The Development of New Methods for High Resolution Radio Astronomy Imaging

Colm Coughlan
BSc

NATIONAL UNIVERSITY OF IRELAND, CORK

College of Science, Engineering and Food Science

Department of Physics

Thesis submitted for the degree of Doctor of Philosophy

31 July 2014

Supervisor:	Dr. Denise Gabuzda
Head of Department/School:	Prof. John McInerney

Contents

List of Figures

List of Tables

I, Colm Coughlan, certify that this thesis is my own work and I have not obtained a degree in this university or elsewhere on the basis of the work submitted in this thesis.

Colm Coughlan

To my family.

Acknowledgements

First of all I'd like to thank my supervisor, Dr. Denise Gabuzda. You have always been there for a chat when things weren't going well, and you've saved me more than once with an insightful suggestion or comment! Thank you for your patience and your kindness. It has been a pleasure doing research under your supervision.

I'd also like to thank all the people I've shared the lab with over the years. Eoin and Fiona – I always looked forward to coming into work when you were in the lab. Thanks for all the good discussions and great times. Thanks especially to Fiona for your care and support while I was writing this thesis. Andrea, Mark, Juan Carlos, Redmond, Shane and Mehreen – thanks for your friendship, your support and your data! Thanks to all the Summer students and fourth year project students that have come through the lab – you have helped me more than I ever helped you.

Thanks to the staff at UCC – lecturers, secretaries, system administrators, cleaners and more. You make UCC a great place to work and have helped me, in one way or another, every day that I've been here.

Thanks to the Irish Research Council for their financial support, and to the Irish Centre for High End Computing for giving me computational time to test my work.

Thank you to Prof. Russ Taylor, for the great Summer I spent doing research in his group in Calgary. Thank you to my thesis examiners, Dr. Gareth Thomas of UCC and Dr. Neal Jackson of the University of Manchester. Your comments and suggestions have made this a better thesis.

Thanks to my friends, especially my house-mates at 1B Coolgarten Park. You have supported me, made my life happy and taught me so much. I'm proud to count you as friends.

Thanks to my family – John, Sheila, Eoghan and Aoibheann, as well as all my grandparents, aunts, uncles and cousins. You have always been there for me, in the good times and the bad. I couldn't have done this without you.

Abstract

Very Long Baseline Interferometry (VLBI) polarisation observations of the relativistic jets from Active Galactic Nuclei (AGN) allow the magnetic field environment around the jet to be probed. In particular, multi-wavelength observations of AGN jets allow the creation of Faraday rotation measure maps which can be used to gain an insight into the magnetic field component of the jet along the line of sight. Recent polarisation and Faraday rotation measure maps of many AGN show possible evidence for the presence of helical magnetic fields.

The detection of such evidence is highly dependent both on the resolution of the images and the quality of the error analysis and statistics used in the detection. This thesis focuses on the development of new methods for high resolution radio astronomy imaging in both of these areas. An implementation of the Maximum Entropy Method (MEM) suitable for multi-wavelength VLBI polarisation observations is presented and the advantage in resolution it possesses over the CLEAN algorithm is discussed and demonstrated using Monte Carlo simulations. This new polarisation MEM code has been applied to multi-wavelength imaging of the Active Galactic Nuclei 0716+714, Mrk 501 and 1633+382, in each case providing improved polarisation imaging compared to the case of deconvolution using the standard CLEAN algorithm. The first MEM-based fractional polarisation and Faraday-rotation VLBI images are presented, using these sources as examples.

Recent detections of gradients in Faraday rotation measure are presented, including an observation of a reversal in the direction of a gradient further along a jet. Simulated observations confirming the observability of such a phenomenon are conducted, and possible explanations for a reversal in the direction of the Faraday rotation measure gradient are discussed. These results were originally published in Mahmud et al. (2013).

Finally, a new error model for the CLEAN algorithm is developed which takes into account correlation between neighbouring pixels. Comparison of error maps calculated using this new model and Monte Carlo maps show striking similarities when the sources considered are well resolved, indicating that the method is correctly reproducing at least some component of the overall uncertainty in the images. The calculation of many useful quantities using this model is demonstrated and the advantages it poses over traditional single pixel calculations is illustrated. The limitations of the model as revealed by Monte Carlo simulations are also discussed; unfortunately, the error model does not work well when applied to compact regions of emission.

Dá dtuigim gach rún
eolas a bheith agam
ach labhraím gan grá,
ní mise ach guth gan ceol
fuaim folamh ar an ngaoth

List of Abbreviations

AGN - Active Galactic Nucleus

AIPS - Astronomical Image Processing System

CASA - Common Astronomy Software Applications

CSIRO - Commonwealth Scientific and Industrial Research Organisation

DFT - Discrete Fourier Transform

FFT - Fast Fourier Transform

MC - Monte Carlo (simulations)

MEM - Maximum Entropy Method

NASA - National Aeronautics and Space Administration

NRAO - National Radio Astronomy Observatory

PMEM - Polarised Maximum Entropy Method (software)

RM - (Faraday) Rotation Measure

VLA - The Very Large Array

VLBA - The Very Long Baseline Array

VLBI - Very Long Baseline Interferometry

Chapter 1

Introduction

1.1 Radio Astronomy

Astronomical objects emit across a wide spectrum of radiation, but observations on Earth are only possible in windows where the Earth's atmosphere does not block the emitted radiation. Figure 1.1 shows the windows in the spectrum at which observations of astronomical objects are possible and shows clearly a major advantage of radio observations – they can be conducted from the surface of the Earth with little interference from the atmosphere.

This clear window of observation occurs because the wavelength of radio radiation is long enough that it is less easily scattered than other wavelengths of emission. The same long wavelength ensures that atmospheric phenomena such as rainfall do not cause major interference. This means that accurate radio observations can be made with less calibration and with much less sensitivity to weather conditions than ground observations at many other frequencies.

A second major advantage of radio observations is that, while the Sun is a radio emitter, its emission is confined to a relatively small part of the sky. This means that, as long as the source being observed does not lie directly behind the Sun, radio observations can be made 24 hours a day. As long as the radio telescope tracks the source continuous observations are possible on very large time-scales, allowing very high signal to noise ratio data to be recorded.

While in many respects the radio wavelengths may seem the ideal place to conduct observations of astronomical sources of radiation, there is one major disadvantage of observing in the radio – the resolution limit imposed by the Rayleigh criterion.

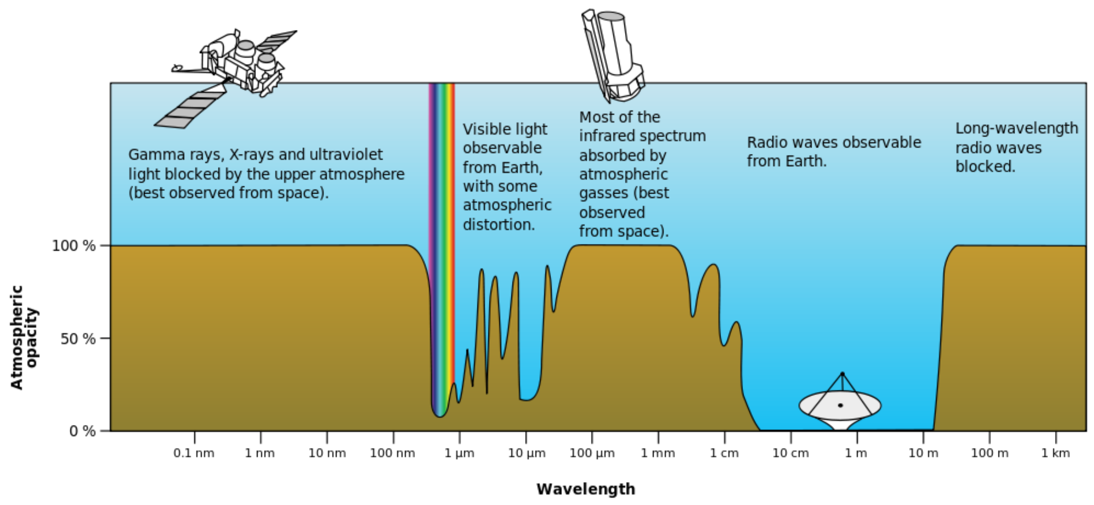


Figure 1.1: The atmospheric opacity of the Earth at different observing wavelengths. NASA image from Wikimedia Commons: http://upload.wikimedia.org/wikipedia/commons/3/34/Atmospheric_electromagnetic_opacity.svg.

The Rayleigh criterion states that the approximate angular resolution, θ, of an observation at a wavelength, λ, with an aperture of diameter d is

$$\theta \simeq \frac{\lambda}{d} \tag{1.1}$$

where the term angular resolution is used to mean the minimum angular separation of two point sources in the sky that results in them being detected as two distinct sources. For example, in order to resolve two sources with an angular separation of 0.1 arc-seconds with observations at the wavelength of yellow optical light (520nm) a telescope with a diameter of 1.2 m is required. However to resolve the same sources at a radio wavelength (for example, at 2 cm) a radio dish with a diameter of 42 km is required.

The construction of a uniform radio dish of this size would be extremely difficult, and funds required to built it would likely dwarf the science case for such an instrument. Indeed, the largest radio dish ever built is located at Arecibo in Puerto Rico with a diameter of just 305 m.

This result initially seems to discount radio observation as being of limited use in astrophysics, however an additional technique, known as radio interferometry can be used to make high resolution radio observations both possible and practical.

1.2 Radio Interferometry

Unless otherwise stated, the material presented in this section is based on Burke & Graham-Smith (2010), and Taylor, Carilli & Perley (1999).

Radio interferometry is a technique which involves the operation of a minimum of two radio dishes together as a combined instrument with an effective resolution proportional to longest distance between two dishes in the array. In the above example, this means that instead of a 42 km wide single radio dish, two much smaller dishes 42 km from each other can observe with a resolution of 0.1 arc-seconds.

Radio interferometry greatly improves the resolution of radio observations without greatly increasing the cost of such observations and has been implemented on scales from hundreds of metres (the Very Large Array in compact configuration) to thousands of kilometres (the Very Long Baseline Array, the European VLBI Network, global interferometry). Interferometry works on the principle that radiation emitted from the same source will arrive at different spatially separated elements of the array at different times. In the same way as interferometric phenomena such as Newton's Rings can be detected in visible light due to the constructive and destructive interference of light, radio fringes can be developed in interferometric arrays by combining the signals from multiple arrays with a suitable time delay on each signal (representing the differing paths travelled by the emission to each element of the array).

In radio interferometry these radio fringes must be found by searching for correlations in the signals from different elements of the array. As there can be many elements in the array, and many more cross correlation terms, this is a computationally expensive operation. Original specialised hardware correlators were designed and used for this purpose, however advancements in computational power mean that in recent years new, more flexible, software based correlators have become available.

Figure 1.2 shows this path difference in the case of a simple two element interferometer. The voltages induced in the two antennas by the incident electromagnetic wave can be written as the following time varying functions

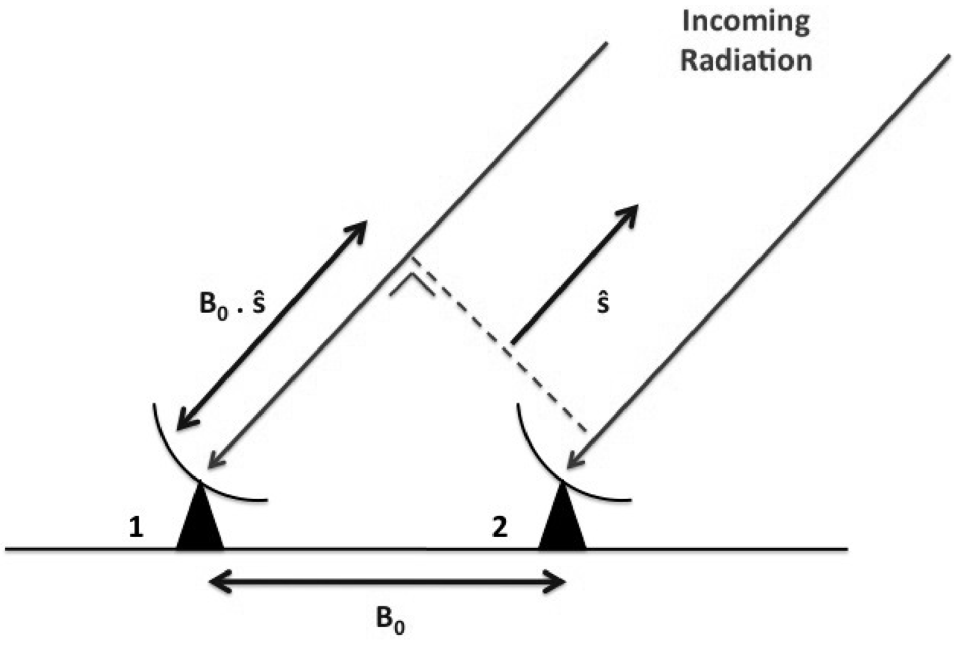


Figure 1.2: Path differences in radio interferometry. Image from Murphy (2013).

$$\begin{aligned} v_1 &\propto E\cos\omega t \\ v_2 &\propto E\cos\omega(t-\tau) \end{aligned}$$

where $\tau = \frac{\vec{B_0}\cdot\vec{s}}{c}$ corresponds to the time for the radio signal to travel the extra distance to antenna 1 and w is the angular frequency of the radiation (related to the frequency by $w = 2\pi f$). The correlation of these two signals is

$$\begin{aligned} \langle v_1 v_2^* \rangle &\propto \langle E^2 \cos\omega t \cos\omega(t-\tau)\rangle \\ &\propto E^2(\cos\omega\tau\langle\cos^2\omega t\rangle + \sin\omega\tau\langle\cos\omega t\sin\omega t\rangle) \\ &\propto E^2\cos\omega\tau. \end{aligned}$$

Substituting the intensity for the square of the electric field, expressing the angular frequency in terms of wavelength and the speed of light, and replacing τ

with the expression above the following equation is obtained

$$\begin{aligned}\langle v_1 v_2^* \rangle &\propto I \cos\left(\tfrac{2\pi c}{\lambda} \tfrac{\vec{B_0} \cdot \hat{s}}{c}\right) \\ &\propto I \cos\left(2\pi \vec{B} \cdot \hat{s}\right)\end{aligned}$$

where $\vec{B}$ is the baseline (distance between telescopes) expressed in wavelengths. Thus a point source of radiation generates a quasi-sinusoidal interference fringe pattern in the response of an interferometer. The amplitude of the fringe is proportional to the intensity of the source and the frequency depends on how $\vec{B} \cdot \hat{s}$ changes in time.

This treatment of the correlator response approximates the radiating source as a single point. To generalise this to an extended source the extended source can be considered as a sum of point sources, each with a different intensity and position in the sky. The position can be described by the vector $\vec{a}$ in the plane of the sky directed to the location of the point source from the direction of the vector $\hat{s}$, known as the "phase-center". The vectors $\vec{a}$ and $\hat{s}$ are thus almost orthogonal to each other ($\hat{s}$ being roughly perpendicular to the plane of the sky). Rewriting the response to a point source for a general location and intensity the following equation is obtained

$$\langle v_1 v_2^* \rangle \propto I(\vec{a}) \cos\left[2\pi \vec{B} \cdot (\hat{s} + \vec{a})\right].$$

If the voltages due to radiation from different parts of the source are uncorrelated, then the total correlator response to the extended sources is

$$\langle v_1 v_2^* \rangle \propto \int_{-\infty}^{\infty} I(\vec{a}) \cos\left[2\pi \vec{B} \cdot (\hat{s} + \vec{a})\right] d\vec{a}.$$

The sine component of the signal, measurable by adding a $\frac{\pi}{2}$ phase shift to the cosine term can then the added onto to the cosine term in Equation (1.2) to give

$$
\begin{aligned}
\langle v_1 v_2^* \rangle &\propto \int_{-\infty}^{\infty} I(\vec{a}) e^{2\pi i \vec{B}\cdot(\hat{s}+\vec{a})} d\vec{a} \\
&\propto e^{i2\pi \vec{B}\cdot\hat{s}} \int_{-\infty}^{\infty} I(\vec{a}) e^{2\pi i \vec{B}\cdot\vec{a}} d\vec{a} \\
&\equiv e^{2\pi i \vec{B}\cdot\hat{s}} V,
\end{aligned}
$$

where V is the visibility function, and $V(\vec{B})$ and $I(\vec{a})$ form a Fourier transform pair (see Section 2.1):

$$
V(\vec{B}) = \int_{-\infty}^{\infty} I(\vec{a}) e^{2\pi i \vec{B}\cdot\vec{a}} d\vec{a} \tag{1.2}
$$

$$
I(\vec{a}) = \int_{-\infty}^{\infty} V(\vec{B}) e^{-2\pi i \vec{B}\cdot\vec{a}} d\vec{B}. \tag{1.3}
$$

As the celestial sphere is nearly flat over a small area, $\vec{a}$ is nearly perpendicular to $\hat{s}$ and $\vec{B}\cdot\vec{a} \approx \vec{b}\cdot\vec{a}$, where $\vec{b} = \vec{B}\cdot\hat{s}$ is the baseline $\vec{B}$ projected onto the sky. Choosing coordinates with axes pointing to the North and to the East the following equations are obtained with (u, v) as the ground coordinates and (x, y) as the sky coordinates

$$
V(u, v) = \int_{-\infty}^{\infty} I(x, y) e^{2\pi i(ux+vy)}\ dx\ dy \tag{1.4}
$$

$$
I(x, y) = \int_{-\infty}^{\infty} V(u, v) e^{-2\pi i(ux+vy)}\ du\ dv, \tag{1.5}
$$

where the second integration sign of the two dimensional integral has been omitted for clarity. This means that as long as $V(u, v)$ is known (i.e. the correlation of the voltages and the lengths of the baselines are known to a high degree of accuracy) it is possible to invert the Fourier transform and obtain a map of the source as it appears in the radio sky. However in practice, due to the finite number of elements in any array, $V(u, v)$ is not known for all values of u and v. This means that the Fourier transform cannot be reliably inverted due to incomplete data, and there is no mathematical technique that can guarantee an authentic reproduction of the original $I(x, y)$ data given only some of the visibility data. This is known as the deconvolution problem in astronomical imaging, and is discussed at length in Section 2.1.

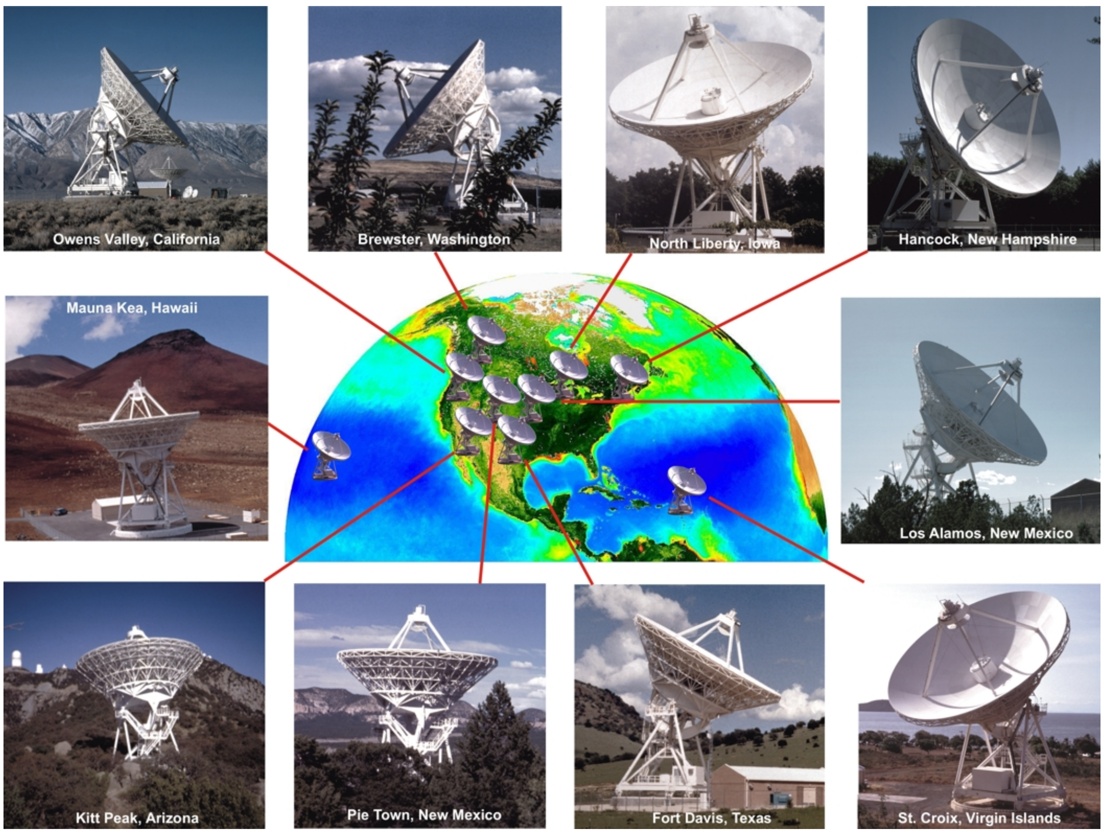


Figure 1.3: The Very Long Baseline Array (USA). 10 telescopes which operate together to perform radio interferometry at resolutions as low as 0.2 mas. Image from http://www.nrao.edu/.

The Fourier pair relationship between $I(x, y)$ and $V(u, v)$ means that a baseline of length u will correspond to a spatial scale on the sky of $x \simeq \frac{1}{u}$, where the u and x are both dimensionless (measured in wavelengths and radians, respectively). This is the Nyquist sampling limit. For any area this means that the smallest resolution possible is

$$\theta \simeq \frac{\lambda}{B} \tag{1.6}$$

where B is the longest baseline in the array (in wavelengths) and λ is the observing wavelength. This effect negates the resolution problem in single dish radio astronomy and makes radio interferometry a good choice for high resolution studies of astronomical objects.

The VLBA (Very Long Baseline Array, see Figure 1.3) is a modern radio interferometer that performs very long baseline interferometry. It comprises of 10 purpose built 25m radio dishes and its longest baseline is over 8,611 km stretching from Mauna Kea on the Big Island of Hawaii to St. Croix in the U.S. Virgin

Islands. Interferometry on these scales is referred to as VLBI (Very Long Baseline Interferometry).

1.3 Interferometric Polarisation Observations

Unless otherwise stated, the material presented in this section is based on Burke & Graham-Smith (2010), and Taylor, Carilli & Perley (1999).

Observations of the polarisation properties of radiation from astronomical objects can give useful indicators of the conditions around the point of emission (see Section 1.6), so it is important to use a clear and concise description of the observed polarisation.

The receivers in the telescopes of the VLBA have two feeds which detect incoming radiation reflected from the collecting dish. One of these feeds responds only to Right-Circularly-Polarised (RCP) emission, and the second (mounted at right angles to the first) responds to Left-Circularly-Polarised (LCP) emission. This means that all the required data to calculate both the intensity and polarisation intensity distribution of a source are collected at once.

The following Stokes parameter based coordinate system is widely used in the radio astronomy community to describe the polarisation properties of observed radiation (see Conway and Kronberg 1969).

$$I = \langle E_A^2 \rangle + \langle E_B^2 \rangle = \langle E_C^2 \rangle + \langle E_D^2 \rangle = \langle E_R^2 \rangle + \langle E_L^2 \rangle \tag{1.7}$$

$$Q = \langle E_A^2 \rangle - \langle E_B^2 \rangle \tag{1.8}$$

$$U = \langle E_C^2 \rangle - \langle E_D^2 \rangle \tag{1.9}$$

$$V = \langle E_R^2 \rangle - \langle E_L^2 \rangle \tag{1.10}$$

where I, Q, U and V are the four Stokes parameters (see Figure 1.5) defined with respect to the component of the electric field in the orthogonal directions A, B, C and D defined in Figure 1.4. R and L correspond to the LCP and RCP intensities. It can be seen that the Stokes I component corresponds to total intensity, while Stokes Q and U together describe the linear polarisation. Stokes V describes the circular polarisation. Figure 1.5 describes the Stokes parameters for simple cases of pure Stokes Q, U and V polarisation.

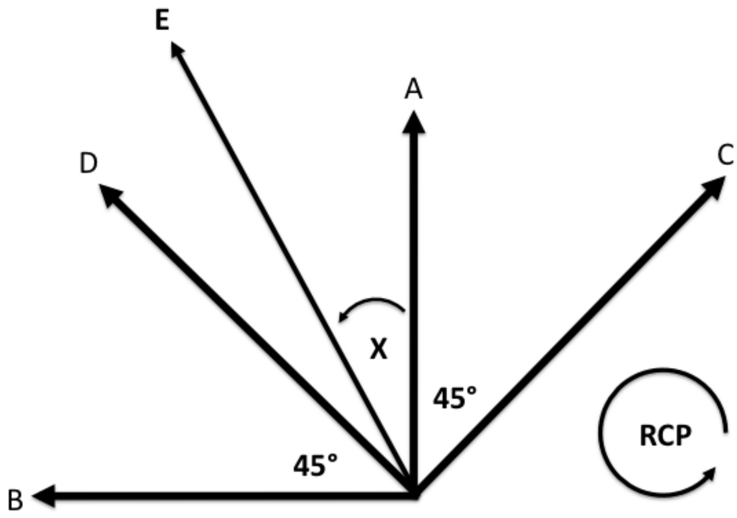


Figure 1.4: Definitions of the directions of the orthogonal modes A, B, C and D for the Stokes parameters and definition for the direction of RCP radiation. Image from Murphy (2013).

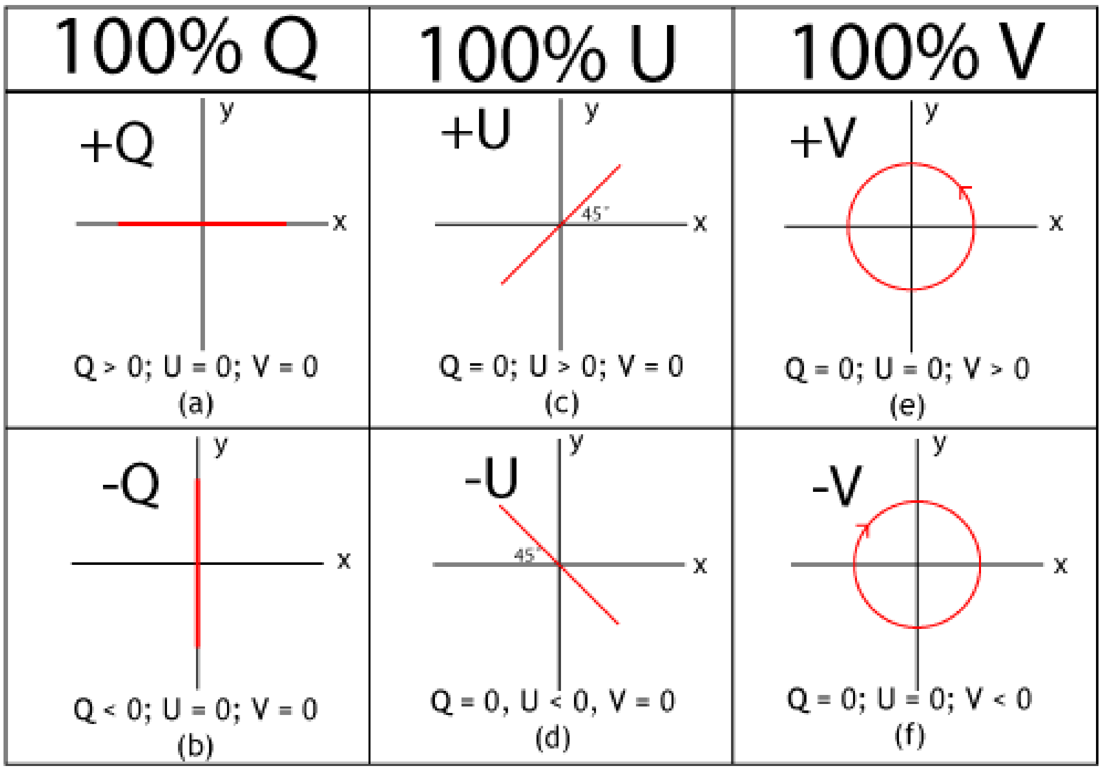


Figure 1.5: The Stokes parametrisation of the polarisation of an electromagnetic wave for some simple cases. Image taken from the Wikimedia Commons: http://en.wikipedia.org/wiki/File:StokesParameters.png.

A connection between Q and U and the source polarisation can be found if the source is considered to have an unpolarised (random) component of its electric field E_u and a polarised component E_p that lies at position angle χ (measured from North to East). This gives

$$E_A = \tfrac{E_u}{\sqrt{2}} + E_p \cos\chi \tag{1.11}$$
$$E_B = \tfrac{E_u}{\sqrt{2}} + E_p \sin\chi \tag{1.12}$$
$$E_C = \tfrac{E_u}{\sqrt{2}} + E_p \sin(45-\chi) \tag{1.13}$$
$$E_D = \tfrac{E_u}{\sqrt{2}} + E_p \cos(45-\chi). \tag{1.14}$$

Note that the power detected in a feed orientated in any of these directions will be the same due to the unpolarised component of radiation. These expressions can then be used to calculate Q and U as follows,

$$Q = (E_u^2 + E_p^2\cos^2\chi) - (E_u^2 + E_p^2\sin^2\chi) \tag{1.15}$$
$$Q = E_p^2\cos(2\chi) \tag{1.16}$$
$$U = (E_u^2 + E_p^2\cos^2(45-\chi)) - (E_u^2 + E_p^2\sin^2(45-\chi)) \tag{1.17}$$
$$U = E_p^2\sin(2\chi) \tag{1.18}$$

where the average of the product of E_p and E_u has been taken as zero, since polarised and unpolarised components of the wave are uncorrelated. Thus the polarisation angle χ, and polarised flux E_p can be calculated as

$$\chi = \frac{1}{2}\arctan\frac{U}{Q} \tag{1.19}$$
$$E_p = \sqrt{Q^2 + U^2} \tag{1.20}$$

and a complex quantity $Q + iU$ can be constructed such that its amplitude is mI, where m is the fractional polarisation, and phase is 2χ. This is the complex polarised flux, P:

$$Q + iU = mIe^{2i\chi} \equiv P. \tag{1.21}$$

Connection with the Correlator Output

Having established the connection between the Stokes parameters Q and U and the linear polarisation of the source radiation, it is now necessary to construct the relationship between Q and U and the output that is obtained from the correlation of the voltages in a VLBI array.

Assume that the VLBI array records the left and right circularly polarised components of the incoming radio signal. Four correlations are possible:

$$\begin{gathered}\langle LL^*(u,v)\rangle \\ \langle RR^*(u,v)\rangle \\ \langle RL^*(u,v)\rangle \\ \langle LR^*(u,v)\rangle\end{gathered}$$

where R and L stand for the RCP and LCP responses and $\langle LL^*(u,v)\rangle$ refers to the correlation of the LCP response of the first antenna with the LCP response of the second, etc. Writing these correlations in terms of the Stokes parameters the following equations are obtained

$$\langle LL^*(u,v)\rangle = I - V \tag{1.22}$$

$$\langle RR^*(u,v)\rangle = I + V. \tag{1.23}$$

This implies that $\langle LL^*(u,v)\rangle$ and $\langle RR^*(u,v)\rangle$ contain information about the total and circularly-polarised intensities of the radiation, but not the linear polarised intensities. In order to find these, the cross correlations $\langle RL^*(u,v)\rangle$ and $\langle LR^*(u,v)\rangle$ need to be examined. This can be done by writing the relationship between the circularly polarised and linearly polarised intensities along axes A and B as follows

$$E_A = E_L \cos(\omega t) + E_R \cos(\omega t) \tag{1.24}$$

$$E_B = E_R \sin(\omega t) - E_L \sin(\omega t). \tag{1.25}$$

Note that χ is chosen to be zero so that the LCP and RCP components of the polarisation line up along the North axis A at t = 0. In complex notation the following equations are obtained

$$E_A = (E_R + E_L)e^{i\omega t} \tag{1.26}$$
$$E_B = -i(E_R + E_L)e^{i\omega t} \tag{1.27}$$
$$E_L = \tfrac{1}{2}(E_A - iE_B)e^{-i\omega t} \tag{1.28}$$
$$E_R = \tfrac{1}{2}(E_A + iE_B)e^{-i\omega t}. \tag{1.29}$$

Thus the correlation $\langle RL^*(u,v)\rangle$ becomes

$$\langle RL^*(u,v)\rangle = \tfrac{1}{2}(E_A + iE_B)(E_A + iE_B) \tag{1.30}$$
$$= \tfrac{1}{2}(E_A^2 - E_B^2) + iE_AE_B. \tag{1.31}$$

The quantity E_AE_B can be identified as U by writing E_C and E_D in terms of E_A and E_B and rewriting U in the following form:

$$E_C = \tfrac{1}{\sqrt{2}}(E_A - E_B) \tag{1.32}$$
$$E_D = \tfrac{1}{\sqrt{2}}(E_A + E_B) \tag{1.33}$$
$$U = \langle E_C^2\rangle - \langle E_D^2\rangle = E_AE_B. \tag{1.34}$$

This implies that Equation (1.31) becomes

$$\langle RL^*(u,v)\rangle = Q + iU. \tag{1.35}$$

Similarly, $\langle LR^*(u,v)\rangle = Q - iU$, and the full set correlations becomes

$$\langle LL^*(u,v)\rangle = I - V \tag{1.36}$$
$$\langle RR^*(u,v)\rangle = I + V \tag{1.37}$$
$$\langle RL^*(u,v)\rangle = Q + iU = P \tag{1.38}$$
$$\langle LR^*(u,v)\rangle = Q - iU = P^*. \tag{1.39}$$

As levels of circular polarisation are usually very low it is often assumed that $V \approx 0$, therefore

$$\langle LL^*(u,v)\rangle \approx \langle RR^*(u,v)\rangle \approx I. \tag{1.40}$$

These Stokes parameters have Fourier transform equivalents in image space give by the following formulae

$$\langle RL^*(u,v)\rangle = P(u,v) \tag{1.41}$$
$$= \int m\widetilde{I}(x,y)e^{2i\chi}e^{2i\pi(ux+vy)}dx\ dy \tag{1.42}$$
$$\langle LR^*(u,v)\rangle = P^*(u,v) \tag{1.43}$$
$$= \int m\widetilde{I}(x,y)e^{-2i\chi}e^{2i\pi(ux+vy)}dx\ dy. \tag{1.44}$$

This gives the important result for the conjugate visibilities

$$\langle LR^*(u,v)\rangle^* = \langle RL^*(-u,-v)\rangle \tag{1.45}$$

meaning that in order to obtain symmetric polarisation visibility coverage (in $u{-}v$ space), all antennae must record both RCP and LCP. This is highly desirable as it simplifies the processing of polarisation data, however it is can be worthwhile to use asymmetric polarisation data if the asymmetric points contribute valuable information (see Section 2.1 for more details).

1.4 Calibrating Polarisation Data

Unless otherwise stated, the material presented in this section is based on Burke & Graham-Smith (2010), and Taylor, Carilli & Perley (1999).

Before the data recorded by the telescope array can be used for science, they must first be calibrated to remove instrumental and atmospheric effects. The presence of the various thermal and systematic errors in the recorded complex visibilities can be understood from the following equation

$$\widetilde{V_{ij}(t)} = g_i(t)g_j^*(t)G_{ij}(t)V_{ij}(t) + \varepsilon_{ij}(t) + \epsilon_{ij}(t) \tag{1.46}$$

where $\widetilde{V_{ij}(t)}$ are the complex visibilities observed with the baseline consisting of array elements i and j, $V_{ij}(t)$ are the true visibilities and g is the gain of an individual element – a measure of how well the antenna converts incoming radio waves into electrical power; G is the non-factorable part of the gain on a baseline, ε represents thermal noise and ϵ is an additive offset term. The effects of G and ϵ are minimised by the design of the array, leaving the effective relationship between the observed and true visibilities as

$$\widetilde{V_{ij}(t)} = g_i(t)g_j^*(t)V_{ij}(t) + \varepsilon_{ij}(t). \tag{1.47}$$

The following Section outlines some of the steps taken to calibrate both the complex visibilities and gain terms in a VLBA polarisation observation. A good description of this process can be found in Taylor, Carilli & Perley (1999).

1.4.1 Amplitude Calibration

In radio astronomy it is convenient to measure the response of a radio telescope to a signal in terms of the equivalent temperature, T, of a source at the input of the receiver. The Rayleigh-Jeans approximation to the Planck radiation law for a black body is

$$P = k_b T \Delta f \tag{1.48}$$

where k_b is the Boltzmann constant, T is the equivalent temperature of the antenna and Δf is the bandwidth of the observation. The equivalent temperature itself is made up of two terms such that $T = T_a + T_{sys}$ where T_a is the antenna temperature of the source, and T_{sys} is the system temperature. The system temperature includes contributions from receiver noise, feed losses, spillover, atmospheric emission, galactic background and cosmic background. The visibility amplitudes must be corrected for these sources of systematic noise such that

$$S_{i,j} = \Delta_{i,j} S_T \sqrt{\frac{T_i T_j}{T_{a,i} T_{a,j}}} \tag{1.49}$$

where the i, j notation refers to the antenna pair in a baseline, $\Delta_{i,j}$ is the correlation coefficient between the two antennae, S_T is the total measured flux of the source and $S_{i,j}$ is the correlated flux on the baseline. Noting that the antenna temperature is the total measured flux scaled by the gain of the antenna ($T_{a,i} = g_i S_T$), the correlated flux on a baseline can be re-written in terms of the gains of the antennae as follows

$$S_{i,j} = \Delta_{i,j} \sqrt{\frac{T_i T_j}{g_i g_j}}. \tag{1.50}$$

1.4.2 Phase Calibration

The first step in phase calibration is to remove the effect of the parallactic angle from the phases. The parallactic angle is the angle between the arcs on the celestial sphere connecting the source and the zenith and the source and the North Pole. The angle describes the rotation of the feeds of the telescope relative to the source as it tracks a source across the sky. These angles are different for antenna mounted at different locations, therefore the parallactic angle difference between two antenna introduces an additional phase term in the visibilities that must be removed.

As demonstrated in Figure 1.2, the emission arriving at one antenna from the source will be out of phase with that arriving at another antenna. The best estimate of the phase delay between two antennas in a baseline is applied during the correlation of the radio signals. The residual phase delays that need to be applied to ensure that each signal phases up correctly must then be determined. When the sources observed are sufficiently compact, this is usually done by modeling them as point sources and calculating the phase delays that maximise the visibility amplitude. For sources that are not sufficiently compact for this approximation, or that are weak, delay solutions for neighbouring compact sources can be interpolated and applied.

1.4.3 Self Calibration

While the initial amplitude and phase calibration perform much of the calibration need to make a good image of a source, many errors can still remain in the visibility data. To remedy this a calibration is performed whereby each source uses itself (or rather, a simplified model of itself) as a calibrator. Care is needed in this technique, as liberal interpretations of what may or may not represent real emission can cause the final "calibrated" visibilities to contain significant errors. The standard method of self calibration uses the CLEAN algorithm (see Section 2.2) to generate a plausible model of the source and applies corrections to the gains by minimising the following function

$$S = \sum_{k} \sum_{i,j \forall i \neq j} w_{ij}(t_k) |\widetilde{V}_{ij}(t_k) - g_i(t_k) g_j^*(t_k) \widehat{V}_{ij}(t_k)| \tag{1.51}$$

where the the indices i and j refer to two telescopes in a single baseline, $t(k)$ is the time at which the observations were taken, w is the weighting, g is the gain of a telescope and $\widetilde{V}$ is the Fourier transform of the data, while $\widehat{V}$ is the Fourier transform of the model (made from CLEAN components – see Section 2.2). By slowly and iteratively changing g_i and g_j for all the telescopes over different time intervals a better estimate of the gains can be calculated and more accurate visibility amplitudes generated, based on the assumption that the model is accurate enough to correct small errors in the data.

Closure Quantities

In addition to minimising Equation (1.51), another important technique used to self-calibrate VLBI sources is the use of closure quantities to remove antenna based errors. This technique is made possible by the property that an appropriate sum of visibility phases around a closed loop of baselines is free of element related errors (see Jennison 1958). This can be seen by considering the following expression for the measured visibility phases derived from Equation (1.47)

$$\widetilde{\phi_{ij}}(t) = \phi_{ij}(t) + \theta_i(t) - \theta_j(t) + noise\ term \tag{1.52}$$

where $\widetilde{\phi_{ij}}$ is the measured phase, ϕ_{ij} is the true phase, θ_i and θ_j are the phases of the gain terms of telescopes i and j. The following quantity, known as the observed closure phase is independent of element related errors

$$\begin{aligned} \widetilde{C_{ijk}}(t) &= \widetilde{\phi_{ij}}(t) + \widetilde{\phi_{jk}}(t) + \widetilde{\phi_{ki}}(t) && (1.53) \\ &= \phi_{ij}(t) + \phi_{jk}(t) + \phi_{ki}(t) + \textit{noise term} && (1.54) \\ &= C_{ijk}(t) + \textit{noise term}. && (1.55) \end{aligned}$$

If one makes a model of the source (with CLEAN for example), one can use the model to provide estimates of the true phases for two terms in this equation for every independent closure phase. This allows the third phase to be derived from the observed closure phase. A similar quantity for the amplitudes known as the closure amplitude can be defined (again from Equation (1.47)) as

$$\Gamma_{ijkl}(t) = \frac{|\widetilde{V_{ij}}(t)|\ |\widetilde{V_{kl}}(t)|}{|\widetilde{V_{ik}}(t)|\ |\widetilde{V_{jl}}(t)|}. \qquad (1.56)$$

In practice both the closure quantities and Equation (1.51) are used to self calibrate the visibility gains. Properly implemented self-calibration can significantly improve the signal to noise ratio of a radio-interferometry image and is an essential step in imaging the jets of AGN.

1.4.4 Instrumental Polarisation Calibration in VLBI

Of the two feeds on radio telescope, one is designed to detect only LCP emission, and the other RCP emission. However, as no instrument is perfect, both feeds actually detect a small amount leakage from the other term – i.e. the LCP feed detects some RCP emission and vice versa. In the most common linear approximation the complex voltages induce in the RCP and LCP feeds can be written

$$\begin{aligned} v_L &= G_L[E_L e^{i\phi_p} + D_L E_R e^{-i\phi_p}] && (1.57) \\ v_R &= G_R[E_R e^{-i\phi_p} + D_R E_L e^{i\phi_p}] && (1.58) \end{aligned}$$

where the G terms are the complex gain factors determined during the calibration process described previously, E_L and E_R are the induced electric fields in complex notation and ϕ_p is the parallactic angle for the relevant telescope. The D terms are the terms that give rise to the "polarisation leakage" and are known as instrumental polarisations, polarisation cross gains, or just simply, the D-terms.

These D-terms can account for polarisation leakage in the range of $1 - 10$ % of the total detected emission. For this reason higher order products of the D terms are ignored and the following expressions for the correlator output can be calculated

$$\begin{aligned}
L_1L_2^* &= G_{L1}G_{L2}^*E_{L1}E_{L2}^*e^{i(\phi_{p1}-\phi_{p2})} && (1.59)\\
R_1R_2^* &= G_{R1}G_{R2}^*E_{R1}E_{R2}^*e^{-i(\phi_{p1}-\phi_{p2})} && (1.60)\\
R_1L_2^* &= G_{R1}G_{L2}^*[E_{R1}E_{L2}^*e^{-i(\phi_{p1}+\phi_{p2})} + D_{L2}^*E_{R1}E_{R2}^*e^{-i(\phi_{p1}-\phi_{p2})}\\
&\quad + D_{R1}E_{L1}E_{L2}^*e^{i(\phi_{p1}-\phi_{p2})}] && (1.61)\\
L_1R_2^* &= G_{L1}G_{R2}^*[E_{L1}E_{R2}^*e^{i(\phi_{p1}+\phi_{p2})} + D_{R2}^*E_{L1}E_{L2}^*e^{i(\phi_{p1}-\phi_{p2})}\\
&\quad + D_{L1}E_{R1}E_{R2}^*e^{-i(\phi_{p1}-\phi_{p2})}]. && (1.62)
\end{aligned}$$

Removing the effect of these D terms is essential if the polarisation properties of the source is to be studied with accuracy. Writing the equations in terms of the Stokes parameters, assuming Stokes V to be zero and applying the parallactic angle corrections (see Section 1.4.2) in such a way as to remove the dependence from the RR and LL correlations these equations become

$$\begin{aligned}
L_1L_2^* &= G_{L1}G_{L2}^*I && (1.63)\\
R_1R_2^* &= G_{R1}G_{R2}^*I && (1.64)\\
R_1L_2^* &= G_{R1}G_{L2}^*[E_p + I(D_{L2}^*e^{2i\phi_{p2}} + D_{R1}^*e^{2i\phi_{p1}})] && (1.65)\\
L_1R_2^* &= G_{L1}G_{R2}^*[E_p^* + I(D_{R2}^*e^{-2i\phi_{p2}} + D_{L1}^*e^{-2i\phi_{p1}})]. && (1.66)
\end{aligned}$$

This means that the D-terms can be calculated by obtaining a series of observations of a calibrator source that is either unpolarised, or has a simple polarisation structure and spans a large range of parallactic angles. The calculated D-terms must then be calibrated out of the visibility data.

1.4.5 Electric Vector Position Angle Calibration

The final step in the calibration of VLBI polarisation data is the calibration of the Electric Vector Position Angle (EVPA). This is most often achieved with the help of integrated polarisation measurements of a calibrator source. At lower resolutions the EVPAs of integrated measurements can be calibrated using standard calibrator sources with known EVPA values. Unfortunately, this method is not feasible with VLBI observations as the polarisation of compact AGN at VLBI scales has been found to be variable in almost every case.

A solution to this problem is to compare relatively low resolution polarisation measurements of the EVPA of a calibrator source with the EVPA corresponding to the total polarisation measured by the VLBI array. This results in an EVPA offset which can then be used to calibrate the VLBI polarisation data at each frequency. The calibrator source used should be sufficiently compact that a large fraction of the source polarisation measured in the low resolution observation is still visible on VLBI scales. If no recent phased array observations of a calibrator source are available for a VLBI experiment, it is also possible to calibrate the VLBI experiment against another, already calibrated, VLBI dataset. However this method is less desirable and if possible an EVPA calibrator source should be included in every VLBI dataset.

1.5 Active Galactic Nuclei

Unless otherwise stated, the material for this Section is based on Peterson (1997), Burke & Graham-Smith (2010), Carroll & Ostlie (2007) and Gabuzda (2008).

Galactic Nuclei are the high density centres of galaxies. A Galactic Nucleus can be considered “Active” if the emission from it is far higher than one might expect simply due to the increased density of stars. The first observation of an AGN was made by E.A. Fath in 1908 at optical frequencies, however AGN were not observed at radio frequencies until the construction of large radio telescopes after the second World War. In general, AGN emission is observed at frequencies ranging from the radio all the way to gamma-ray and are distinguished by their unusually high emission. Emission from AGN can range up to 10^{46} ergs/s, whereas normal galaxies (such as the Milky Way) may have emissions on the order of the $\approx 10^{36}$ ergs/s (Note 1 erg= 10^{-7} J). This means that an AGN can have emission up to 10^{10} times greater than a normal galaxy. The first step in the study of AGN was the identification of a mechanism that could account for these huge amounts of energy.

1.5.1 Energy production in AGN

There are few astrophysical processes that can credibly account for such high rates of emission. The best candidate process for powering AGN is thought to be the high efficiency accretion of material from an accretion disk onto a supermassive black hole at the centre of the AGN (Figure 1.6). The following quick analysis gives an indication of the plausibility of such a model.

Consider an AGN of mass M radiating at the Eddington limit L – that is to say radiating with a outwards radiation pressure which balances the inward force of gravity. Noting that the momentum of a photon with energy E is $p = \frac{E}{c}$, the outward momentum flux at a distance r from a source of luminosity L is

$$P_{rad} = \frac{L}{4\pi r^2 c}. \tag{1.67}$$

This can be converted to an outwards radiation force on a single electron by multiplying the radiation pressure with the Thomson scattering cross section for interaction between a photon and electron σ_e, to obtain

$$F_{rad} = \sigma_e \frac{L}{4\pi r^2 c}. \tag{1.68}$$

By equating this force with the inward gravitational force $F_{grav} = \frac{GMm_p}{r^2}$ between the AGN and an electron-proton pair (of mass $\approx m_p$) the maximum luminosity possible without the disintegration of the source can be calculated as follows

$$\begin{aligned} L = & \frac{4\pi G c m_p}{\sigma_e} M \\ \approx & 7.5 \times 10^7 M \ ergs \ s^{-1}. \end{aligned} \tag{1.69}$$

This is the Eddington luminosity and allows a lower limit to be placed on the mass of a source observed to be emitting at a luminosity L. If a typical AGN luminosity of the order of 10^{46} ergs/s is inserted into this equation, a lower limit of about $10^8 M_{Sun}$ can be determined for the mass of the emitting system.

AGN have been observed to be variable on the time scales of just a few hours. As the size of the AGN core must be comparable to the distance light travels across the core, this means that the smallest AGN must be only a few light hours across. The speed of light is 2.998×10^8 m/s, therefore it travels 1.07928×10^{12} m in an hour, a distance equivalent to 7.2 AU – where 1 AU, an astronomical unit, is the distance from the Earth to the Sun. Thus an AGN core may be of the order of 10s of AU across.

If a object of mass m positioned a distance r from a body of mass M wishes to escape the gravitational attraction of the body, the minimum escape velocity of the object, v_e can be calculated by considering the kinetic energy required to emerge from the body's gravitational potential well

$$\frac{1}{2}mv_e^2 = \frac{GMm}{r} \tag{1.70}$$

$$v_e = \sqrt{\frac{2GM}{R}}. \tag{1.71}$$

If one considers the case of a body so massive that even light cannot escape for it, this formula can be re-arranged to give the radius r_s at which the escape speed from a body of mass M is equal to the speed of light. This is the Schwarzschild radius of the body:

$$r_s = \frac{2GM}{c^2}. \tag{1.72}$$

Inserting an estimate of 10 AU for the radius of the AGN core, a corresponding black hole mass of about $5 \times 10^8 M_{sun}$ is obtained. This compares well with the mass estimated assuming emission at the Eddington limit and is strong evidence for a supermassive black hole based AGN emission.

The question of how much mass such a black hole would need to consume to provide the observed luminosity is important – if it is unrealistically high, then supermassive black holes would seem an unlikely candidate for the engines of AGN. If it is assumed that the energy emitted by the nucleus of the AGN is supplied by matter accretion of rate $\dot{M}$ with an efficiency η the following equation is obtained

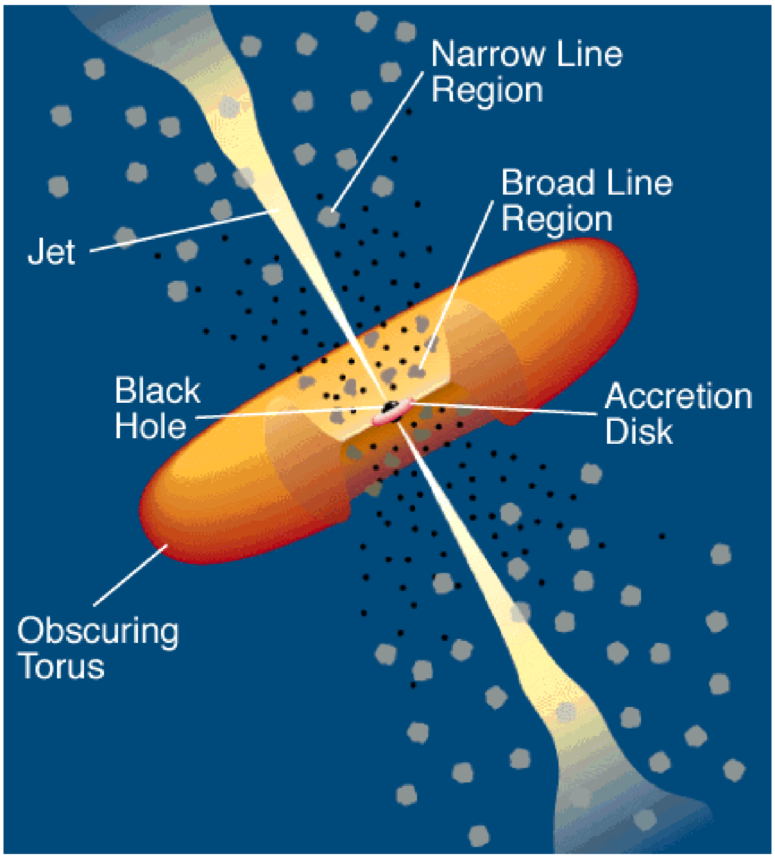


Figure 1.6: Schematic of an Active Galactic Nucleus. Image taken from Urry & Padovani (1995).

$$L = \eta \dot{M} c^2. \tag{1.73}$$

Thus by inserting an AGN luminosity of 10^{46} ergs/s and assuming a moderate efficiency of $\eta = 0.1$ the required mass accretion rate can be calculated as

$$\begin{aligned} \dot{M} =& \frac{L}{\eta c^2} \\ =& \frac{10^{39}}{0.1 c^2} \\ \approx& 1 \times 10^{23} kg/s \\ \approx& 1.8 M_{Sun}/yr. \end{aligned} \tag{1.74}$$

Thus an accretion rate of only a few solar masses a year is needed to fuel AGN at observed luminosities, making an accretion fuelled emission mechanism very plausible.

1.5.2 Extragalactic Origin of the Emission

There was initial confusion as to whether early detections of AGN represented galactic sources, such as stars or novae, or truly extragalactic sources. Indeed, AGN classifications such as Quasar (quasi-stellar object) and BL Lac show the confusion that existed in the early 20^{th} century. Vesto Slipher took the initial steps to solving this problem in 1914 when he published his results on the observed radial velocities of extragalactic objects which had until then been thought to be in our own galaxy. Slipher (1914) determined these velocities by observing the Doppler shift of light from the objects. Instead of the random distribution of radial velocities that had been expected for a source orbiting the same galactic centre as the Earth, Slipher discovered that most of the objects he was studying were shifted towards the red end of the spectrum – implying radial velocities away from the Earth. This implied that the objects were not orbiting the galactic centre, and thus were likely to be extragalactic in nature.

Shortly after this, Edwin Hubble began a search for a particular type of star – Cepheid variables. A Cepheid variable is one of the 'standard candles' of astronomy, an object that belongs to a class of known luminosity. In the case of Cepheid variables there is a strong connection between the intrinsic luminosity of the star and the period of variation as the star pulses between a brighter and a dimmer state. Thus, comparing the apparent brightness and intrinsic luminosity implied by the Cepheid period–luminosity relation made it possible to determine the distance of the variable star.

Hubble found Cepheid variables in some of Slipher's extragalactic sources and used them to determine the distance to these sources. Hubble (1925) made the surprising discovery of a linear relationship between the radial speed of the source away from the Earth, v, and its distance from the Earth, d – nearby sources moved away slowly, while further sources moved away quickly. He expressed this relationship as

$$v = H_0 d \tag{1.75}$$

where H_0 is the Hubble constant. This constant is related to the expansion of the universe and current estimates give its value of about 71 (km/s)/Mpc. Thus if one can find the velocity of an astronomical source, extragalactic or otherwise, the distance to the source can be determined. In practise this velocity is determined just as Slipher did it – by determining the redshift, z, of the source

$$z \equiv \frac{\Delta\lambda}{\lambda_{rest}} \tag{1.76}$$

where $\Delta\lambda$ is the wavelength shift from an original wavelength λ_{rest}. This is related to the radial velocity of the source v_r by the relativistic Doppler formula

$$z = \sqrt{\frac{1+\frac{v_r}{c}}{1-\frac{v_r}{c}}} - 1. \tag{1.77}$$

It is common in astronomy to give the distance of an extragalactic object from the Earth in terms of its redshift and values of z have been calculated for most AGN by spectrometric analysis.

1.5.3 The components of an AGN

Figure 1.6 shows a schematic of the current model of an Active Galactic Nucleus. The centres of all AGN are thought to be supermassive black holes with masses in the range from millions to billions of solar masses. The existence of such a dense concentration for the mass is postulated as a result of the high energies and rapid variability observed in AGN . As discussed in Section 1.5.1 a number of factors suggest that the most likely configuration of mass in the true centre of an AGN is a supermassive black hole.

Surrounding the black hole is an accretion disk that is at most a few parsecs across. Thermal radiation detected from the disk and observed spectra suggests a non-uniform temperature throughout the disk, while optical spectral lines exhibit extreme Doppler broadening – an indicator of rotation and turbulent motion in the disk.

Outside of the accretion disk a torus of cool molecular gas obscures much of the inner structure of AGN. Infra-red radiation indicates the presence of dust molecules in the torus and maser (naturally occurring microwave amplification and stimulated emission of radiation) observations by Miyoshi et al. (1995) have detected a torus rotating at 900 km/s around a massive central object, providing strong evidence for the presence of a central black hole.

Clouds of molecular gas (thought to be H II) surround the torus. For reasons that are not yet fully understood broadened spectra are observed from the clouds in close to the centre (inside the torus), while narrow lines are observed from those further out. The broad lines from the inner clouds are interpreted as being a result of increased Doppler broadening due to the higher rotational velocities closer to the black hole.

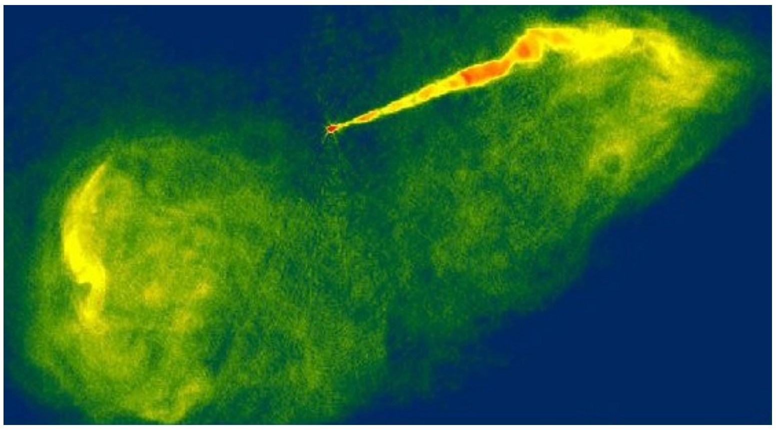
In about 10% of AGN, jets of relativistic plasma have been detected emanating from the central region as in Figure 1.6. It is thought that the jets comprise of accretion disk material launched by either the angular momentum present in the accretion disk (the Blandford and Payne mechanism) or by the extraction of rotational kinetic energy from the rotating supermassive black hole (the Blandford-Znajek mechanism). In both models strong magnetic fields generated by the moving plasma are collimated into two jets by a combination of magnetic fields and pressure shocks. These jets are emitted perpendicularly to the plane of the accretion disk (and probably along the axis of rotation of the spinning supermassive black hole) and can be highly relativistic, moving at speeds very close to the speed of light. These AGN jets have been observed extending up to tens of thousands of light years from the centre of the AGN.

Relativistic jets can give rise to interesting effects, some of which can lead to problems in the classification of AGN. One such effect is the relativistic Doppler beaming of the AGN jets. The jets from AGN are emitted in opposite directions; the emission from each of these will be concentrated in the direction of its motion due to relativistic beaming (Appendix A). Due to this effect, if one of the jets is pointing roughly towards the Earth, it will be observed much brighter than it actually is, while the counter jet (pointing in approximately the opposite direction) will be detected as much weaker, if it is visible at all. Figure 1.7 demonstrates the difference between a Doppler boosted source (Figure 1.7a) and a source where the jets are roughly perpendicular to the line of sight between the Earth and the source (Figure 1.7b).

Appendix A contains a description of how relativistic effects in AGN can give rise to this relativistic Doppler beaming, as well as to the detection of apparently super-luminal speeds in AGN.

1.5.4 Classes of AGN

AGN are divided into a number of different classes. Whether these classes represent different objects, or are simply the same object observed in different ways is an open matter of debate. This section describes the main classes of AGN and


(a) M87.

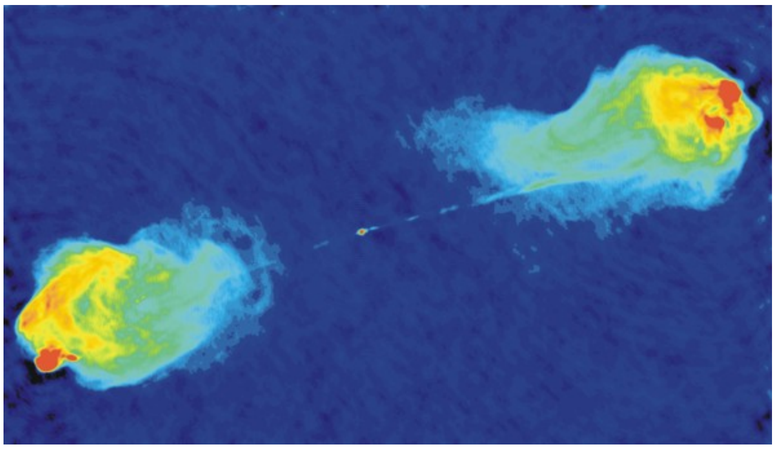

(b) Cygnus A.

Figure 1.7: Radio images of M87 and Cygnus A as observed with the VLA. M87 exhibits strong Doppler boosting in one of its jets, with no sign of the counter jet. Cygnus A has two clear jets, with no evidence of Doppler boosting (they appear to be roughly the same brightness). Images courtesy of the NRAO/AUI : www.nrao.edu.

discusses how they may in fact represent the same objects.

Seyfert Galaxies

Seyfert Galaxies are low luminosity AGN which can have a compact star-like nucleus and a clearly visible host galaxy. Originally identified just using this description, they are now identified by the presence of strong, high ionisation emission lines. Most, if not all, Seyfert AGN occur in spiral galaxies. They can be broken up further into Seyfert 1 and 2 galaxies, where type 1 galaxies show both narrow and broad line emission due to low density ionised gas, while type 2 Seyfert galaxies show only narrow line emission. The reason for this difference is not clear, however it is possible that type 2 galaxies are in fact simply type 1 galaxies where the broad line components of the lines are not visible from the Earth's line of sight.

Quasars

Quasars (Quasi-stellar objects) are the most luminous class of AGN. Originally believed to be a type of star, most quasars are unresolved and appear star-like, however some display suggestions of star light from their host galaxy. Spectra are similar to those of Seyfert galaxies, but the narrow line features appear weaker relative to the broad lines and there is little evidence of any stellar absorption features.

Radio Galaxies

Radio galaxies are galaxies that emit strongly in radio and are often associated with giant elliptical galaxies. Radio galaxies show the same broad line / narrow line separation as Seyfert class 1 and 2 galaxies, but are much more radio loud and occur in different types of galaxies. Many radio galaxies emanate jets perpendicularly to the galactic plane. These jets can come from one side or both sides of the galaxy, though it is more common to see only a single jet due to relativistic Doppler boosting as discussed in Appendix A.

In the case where two jets are observed from a radio galaxy the class is divided once more into Fanaroff-Riley class I or II. FR-I radio galaxies exhibit decreasing luminosity as the distance from the AGN increases, while FR-II galaxies show increased luminosity in the radio lobes at the end of the jet. FR-I galaxies are typically lower luminosity than FR-IIs and the primary difference between the two is thought to be the efficiency of their energy transfer along the jet. FR-I galaxies are less efficient and lose more energy at the start of the jet (near the AGN), while FR-IIs keep their energy until much further out.

LINERs

LINERSs are low-ionisation nuclear emission-line region galaxies first identified by Heckman (1980). They exhibit very low luminosity at the galactic nucleus and spectroscopically resemble Seyfert 2 galaxies with stronger low-ionisation lines. LINERs are very common and may be detectable in up to half of all spiral galaxies.

Blazars

Blazars comprise of two separate AGN classes – optically violent variables (OVVs) and BL Lac objects. Optically violent variables exhibit high degree of polarisation (up to a few percent) and are observed to vary greatly on time-scales of less than a day. BL Lac objects, named after their prototype BL Lacertae – an AGN originally identified as a highly variable star, exhibit similar variability but also do not show strong emission or absorption lines in their spectra. Weak lines have however been detected in observations with very high signal to noise ratios.

Both OVVs and BL Lac objects are thought to be AGN exhibiting strong relativistic beaming close to the line of sight to Earth. For this reason they are both classified as “blazars”.

Narrow Line X-Ray Galaxies

Narrow line X-Ray galaxies are thought to be Seyfert galaxies whose spectra have been reddened and extinguished by dust within the galaxy. Typically they have the same high-excitation emission lines as Seyfert galaxies but a lower luminosity.

1.5.4.1 AGN Unification

All of the AGN classes discussed in this section are thought to originate from the same type of galaxy viewed from different angles as illustrated by Figure 1.8. An AGN viewed with its jet emanating at 90° from the line of sight will not have its jet strongly Doppler boosted and will appear as a Seyfert 2 or a radio galaxy. A jet angle of around 60° may result in a Seyfert 1 galaxy, while as the jet angle reaches 30° Doppler boosting makes the AGN component of the galaxy more visible, resulting in the detection of a quasar galaxy. Finally, a viewing angle of 0° corresponds to ‘looking down the gun’ of a jet from an AGN and a blazar type object is observed.

Differences in intrinsic AGN luminosity and the existence of a radio loud / radio quiet divide between some AGN make a full unification scheme challenging, however efforts continue in the field to include as many of these effects in the scheme

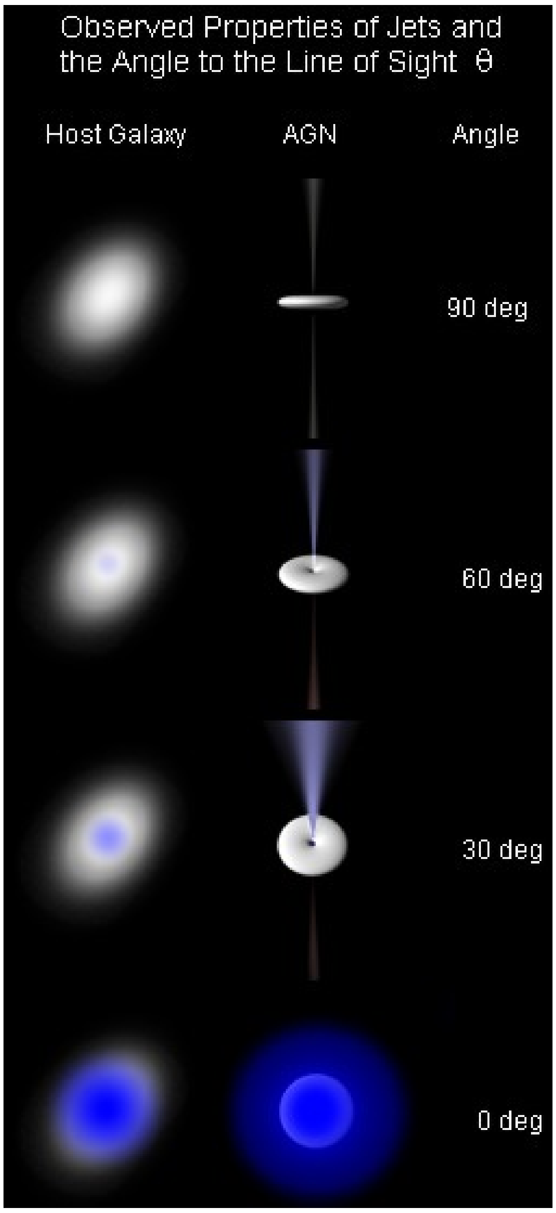

Observed Properties of Jets and the Angle to the Line of Sight θ
Host Galaxy
AGN
Angle
90 deg
60 deg
30 deg
0 deg

Figure 1.8: AGN Unification. Illustration of how the type of galaxy visible from Earth may vary with viewing angle. Image from Wikimedia Commons: http://en.wikipedia.org/wiki/ File: Galaxies_AGN_Jet_ Properties-with-LoS.jpg.

as possible. For a good discussion of AGN unification see Urry & Padovani (1995) and Antonucci (1993). The AGN considered in this thesis are primarily quasars and blazars.

1.5.5 Synchrotron Radiation from AGN

The material in this section is based on Rybiki & Lightman (2008) and Gabuzda (2008).

AGN emit radiation over a wide variety of frequencies. They cannot be well described by a black-body spectrum corresponding to a single temperature or a number of black bodies over a small temperature range. This led to early speculation that non-thermal emission processes were behind the spectral energy distribution of AGN. The continuum emission was originally modelled as a power low of the form

$$F_\nu \propto \nu^\alpha \tag{1.78}$$

where ν is the frequency of radiation, F_ν is the monochromatic flux (the energy in a single frequency interval per unit area) and α is the spectral index of the emission – observed to vary from -1 to small positive values. This power law behaviour is predicted by synchrotron radiation emission from a population of relativistic electrons with a power law distribution in energy as they spiral in a magnetic field. The predicated distribution of the electron energies can be written as

$$N(E)dE = N_0 E^{-p} dE \tag{1.79}$$

where $\alpha = -\frac{p-1}{2}$ and $N(E)dE$ is the number density of electrons between E and $E + dE$. Synchrotron radiation is the relativistic equivalent of cyclotron radiation – the process by which electrons gyrating in a magnetic field emit radiation with a frequency equal to the frequency of gyration. By considering the instantaneous rest frame of the gyrating electron Larmor's formula can be used to calculate the radiated power in the rest frame

$$P' = \frac{2e^2(a')^2}{3c^3} \tag{1.80}$$

where e is the charge of an electron, a' is the acceleration of the electron in the rest frame and c is the speed of light. To find the corresponding power in the observing frame consider the relativistic transformations on the work done over time t' in the rest frame. The work done in the observer frame and the time

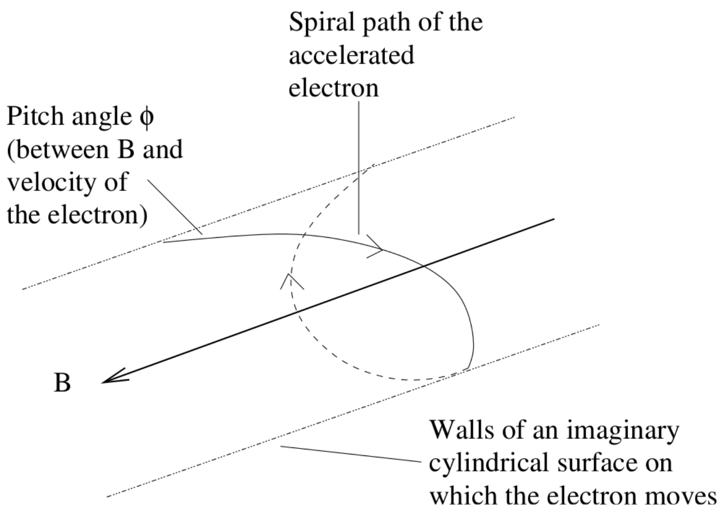

Spiral path of the accelerated electron
Pitch angle ϕ (between B and velocity of the electron)
B
Walls of an imaginary cylindrical surface on which the electron moves

Figure 1.9: An electron moving relativistically in a magnetic field emitting synchrotron radiation. Image from Gabuzda (2008).

elapsed in the observer frame can be calculated as

$$dW = \gamma dW' \tag{1.81}$$
$$dt = \gamma dt' \tag{1.82}$$

where γ is the Lorentz factor. Therefore the power in the observer frame can be calculated as

$$P = \frac{dW}{dt} = \frac{dW'}{dt'} = P'. \tag{1.83}$$

Thus even though the electrons are moving relativistically, Larmor's formula (Equation (1.80)) holds and can be used to calculate the power emitted due to synchrotron radiation.

Consider an electron moving relativistically in a magnetic field as shown in Figure 1.9. The equations of motion governing the electron are

$$\frac{d}{dt}(\gamma m\vec{v}) = q\vec{v} \times \vec{B} \tag{1.84}$$
$$\frac{d}{dt}(\gamma mc^2) = q\vec{v}.\vec{E} = 0 \tag{1.85}$$

where the Equation (1.84) is the Lorenz force on a charge q moving in a magnetic field $\vec{B}$ with a velocity $\vec{v}$ and Equation (1.85) is the rate of change of relativistic energy in an electric field $\vec{E}$. Magnetic fields are believed to play an important role in the dynamics of AGN. If the magnetic forces can be approximated as much larger than the electric forces Equation (1.85) can be set to zero and neglected. This implies that γ is a constant, along with $\vec{v}$ and the pitch and between the magnetic field and the velocity θ. Therefore Equation (1.84) can be rearranged to solve for the acceleration in the observer frame as

$$a = \frac{e}{\gamma mc}\vec{v} \times \vec{B} \tag{1.86}$$

$$a = \frac{e\beta \sin\phi B}{\gamma mc}\hat{a} \tag{1.87}$$

where $\hat{a}$ is a unit vector perpendicular to both the magnetic field and direction of motion. This acceleration can be converted to the rest frame of the electron using the standard acceleration transformations for the components of acceleration parallel and perpendicular to the direction of motion:

$$a'_{\parallel} = \gamma^3 a_{\parallel} \tag{1.88}$$

$$a'_{\perp} = \gamma^2 a_{\perp}. \tag{1.89}$$

As the acceleration in this case is strictly perpendicular to the direction of motion the observed synchrotron radiation in the unprimed frame will be

$$P = \frac{2e^4\gamma^2\beta^2 \sin^2\phi B^2}{3c^3m^2}. \tag{1.90}$$

If one assumes a random distribution of pitch angles θ this power then becomes

$$P = \frac{4e^4\gamma^2\beta^2B^2}{9c^3m^2}. \tag{1.91}$$

This formula for the power emitted due to synchrotron emission in a magnetic field is the primary reason for the belief that the emission that is observed is mostly electronic in origin. The $\frac{1}{m^2}$ term in Equation (1.91) means that electrons are much more efficient at radiating synchrotron radiation that relatively massive protons.

Thus the continuous emission is predicted from electrons gyrating around magnetic field lines. This causes the emission to be tightly beamed along the instantaneous direction of motion at any one time. An indication of the lifetime of synchrotron emission can be gained by dividing the energy in an electron by the rate at which the electron radiates energy to get

$$\tau_{sync} \simeq \frac{E}{P} \tag{1.92}$$

$$\tau_{sync} \simeq \frac{\gamma mc^2}{1}\frac{9c^3m^2}{4e^4\gamma^2\beta^2B^2} \tag{1.93}$$

$$\tau_{sync} \simeq \frac{9m^3c^5}{4e^4\beta^2\gamma B^2}. \tag{1.94}$$

This indicates that more energetic electrons radiate more powerfully and as a consequence have shorter lifetimes. The frequency corresponding to peak radiation can be calculated to be

$$\nu_{peak} \simeq \frac{\gamma eB}{2\pi mc\beta \sin\phi} \tag{1.95}$$

$$\nu_{peak} \simeq 3\times 10^6 B\gamma^2 \sin\phi \; Hz. \tag{1.96}$$

This tells us that the more energetic electrons will radiate at higher frequencies and contribute more to the higher parts of the spectrum.

A consequence of these results is that electrons radiating at higher frequencies will do so for less time than electrons radiating at lower frequencies. Thus if an ensemble of electrons are given a distribution of energies, the higher frequency emission will be exhausted more quickly than the lower frequency emission. In plots of the spectral profile of synchrotron radiation such as in Figure 1.10 this would be observed as a steepening of the high frequency part of the spectrum with time.

Regions of AGN which exhibit relatively flat (or less steep) spectral index may be indicative of regions where the supply of high energy electrons is being replenished by a mechanism which can re-accelerate electrons up to high energy levels. One of the main mechanisms thought to be responsible for such re-accelerations is the

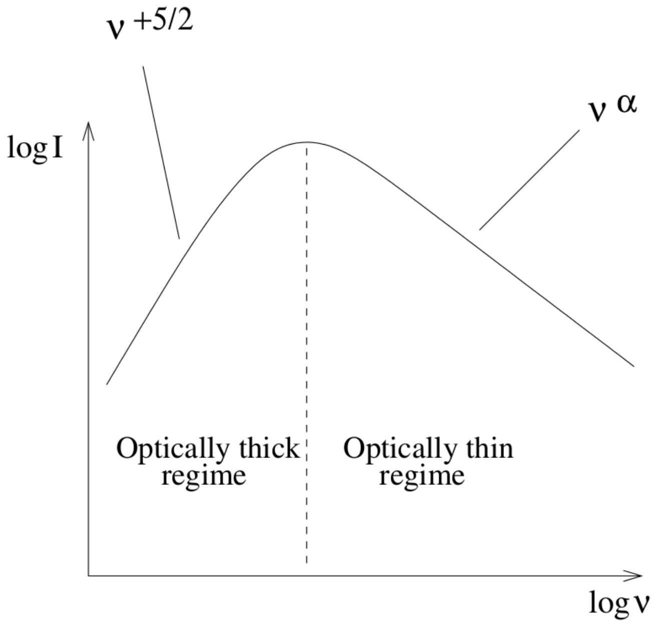


Figure 1.10: Spectrum of Synchrotron Radiation. Image from Gabuzda (2008).

formation of a shock in the jet, where the flow of jet material is greater than the speed of sound in the jet and shockwaves form at various points in the jet.

1.5.5.1 Optical Depth

Different regions of AGN jets exhibit different opacities to synchrotron radiation. If the mean free path of a photon in the radiating region is long enough that the photon is likely to be able to leave the region without being reabsorbed by the jet then that region is said to be optically thin. Correspondingly, if the mean free path of a photon in a region is less than the size of the region, that region is said to be optically thick. Figure 1.10 shows how a positive spectral index in Equation (1.78) corresponds to an optically thick region, while a negative spectral index corresponds to a region where the jet is optically thin.

At low frequencies there is a significant chance for the synchrotron emission produced to be re-absorbed by the same electrons producing the radiation. This is known as synchrotron self-absorption and results in the low frequency part of the synchrotron spectrum being optically thick with a theoretical value for the spectral index of $\alpha = \frac{5}{2}$. While this value is shown in Figure 1.10, in practice observed spectra are only partially optically thick at these low frequencies.

1.5.5.2 Polarisation Properties

Synchrotron radiation naturally produces highly polarised emission. In the case of a completely ordered magnetic field with a pitch angle in Figure 1.9 equal to 90° in all cases the degree of polarisation of the emitted radiation could be as high as 100%. In the more realistic case of randomly orientated pitch angles the theoretical maximum reduces to about 75% for a completely ordered magnetic field in an optically thin region.

The polarisation of the radiation can be broken up into components parallel to and perpendicular to the direction of the magnetic field. In optically thin regions the observed polarisation is perpendicular to the magnetic field and the degree of polarisation can be up to 75%. In optically thick regions the observed polarisation is parallel to the magnetic field and the maximum degree of polarisation is much lower – of the order of 10–15%. The reason for the difference in degree of polarisation is that the emission of polarisation radiation perpendicular to the magnetic field (with the electric field of the emission in the plane of gyration of the electrons) is much more likely than emission of radiation parallel to the magnetic field.

1.6 Polarisation Structures in AGN Jets

Having established the deep link between the polarisation of synchrotron radiation from AGN jets and the local magnetic field environment, it is clear that polarisation observations of jets may provide a valuable insight into the behaviour of any magnetic fields that may be involved in the launch and collimation of the jets. Some common polarisation signatures detected in AGN jets will be discussed in the following section. In particular – the connection between many polarisation features and the possible presence of jet shocks or helical magnetic fields will be highlighted.

The most visible characteristics of polarised emission are the degree of the polarisation (the fraction of the emission which is polarised) and the direction of the EVPA (electric vector position angle), which indicates the direction of the plane of polarisation of radiation at a particular part of the jet. The fractional polarisation, m, is defined as the ratio of the total flux to polarised flux as follows

$$m = \frac{I}{P} \tag{1.97}$$

where I is the Stokes I intensity of the emission and P is the total polarised intensity. The fractional polarisation at any point in a jet is a good indicator of the order in the underlying magnetic field. An ordered field produces relatively highly polarised emission, while a disordered or tangled field will produce significantly less polarisation. As the emission detected from AGN jets is synchrotron radiation, the theoretical maximum fractional polarisation that could be observed is 75%, however in practice much lower figures are normally observed.

EVPAs are observed to vary across the jet for a variety of reasons. The basic morphology of the jet has a major effect – bends and kinds in the jet are often accompanied by changes in the EVPA detected and it is important to note that only the $2D$ projection of the jet on the sky is visible and any emission detected may in fact be integrated through many different regions of the jet. Shear at the edges of the jet can also be detected as a 'drag' in EVPA angles observed in such regions. As synchrotron radiation is polarised in a direction perpendicular to the magnetic field in an optically thin region (see Section 1.5.5), the EVPA can also be an important diagnostic of the jet magnetic field.

Variations in fractional polarisation and degree of polarisation across the jet can be interpreted as corresponding to a variety of physical processes inside the jet. A compact feature with higher than usual fractional polarisation and polarisation aligned parallel to the jet direction (implying an orthogonal magnetic field) may indicate the presence of a transverse shock in the jet. Cawthorne, Jorstad & Marscher (2013) have described the EVPA signature and fractional polarisation pattern corresponding to the presence of a conical shock in an AGN jet and found evidence of such a shock in the jet of 1803+784.

The detection of an extended region of polarisation orthogonal to the jet may indicate a region of the jet experiencing shear stresses, a change in the direction of the jet or the presence of a helical magnetic field with a low pitch angle. In all of these cases, the fractional polarisation may be observed to increase towards the edge of the jet. Such an increase in fractional polarisation results from the increasing order imposed on the magnetic field by the forces acting on the jet. Figure 1.11 demonstrates how a bend in a jet carrying a helical magnetic field could give to an increased degree at the outer edge of the bend, where the longitudinal field component is increased due to the 'stretching out' of the

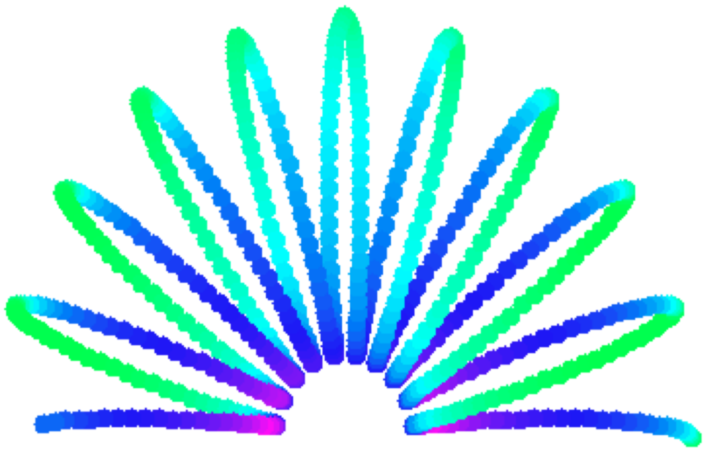


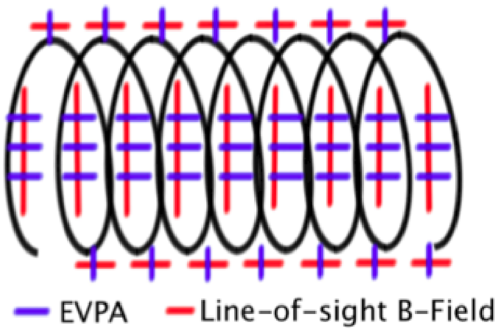
Figure 1.11: The bending of a helical magnetic field can cause an increase in the fractional polarisation at the outer part of the bend. This occurs as the fields are more ordered due to the increased spacing. Colour-scale: The colours indicate the direction of the helix (green for towards the observer, purple for away). Image from Healy (2013).

frozen-in field.

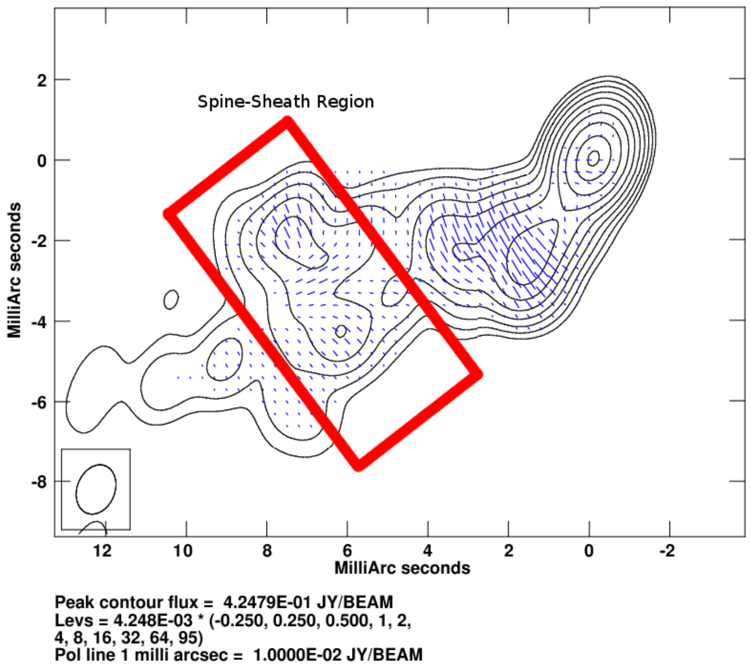
While compact regions of high fractional polarisation and EVPAs parallel to the jet direction are likely to be the result of shocks in the jet, such shocks are unlikely to extend over a large area. Extended regions of parallel EVPAs are likely to indicate the presence of an axisymmetric magnetic field with a toroidal component orthogonal to the jet direction, such as a helical field with a relatively high pitch angle.

Observations of the EVPAs in some sources have shown the existence of a 'spine-sheath' polarisation structure, where the EVPAs in the centre of the jet are parallel to the direction of the jet, but the EVPAs towards the edge of the jet are perpendicular to the jet direction. Figure 1.12a illustrates how a helical magnetic field may give rise to such a structure and Figure 1.12b shows the presence of a spine-sheath structure in polarisation images of Markarian 501 deconvolved with the Maximum Entropy Method as outlined later in this thesis (data originally analysed by Pushkarev et al. (2005)). This type of polarisation structure provides evidence for the presence of a helical magnetic field in the jet, however the detection of such a clear signature is highly dependent on both the viewing angle of the AGN and the pitch angle of the helix.

Murphy, Cawthorne & Gabuzda (2013) investigated this dependence and calculated all of the polarisation signatures possible for a simple model of the helical magnetic field. Figure 1.13 shows these configurations and confirms the presence



(a) Spine-Sheath Model. Image from Healy (2013).



(b) Markarian 501.

Figure 1.12: Spine-sheath structure in theory and reality. Figure 1.12a shows the ideal model of spine-sheath structure. Figure 1.12b shows spine-sheath structure in the optically thin part of the jet of Markarian 501 at 6cm imaged with the VLBA. This image was deconvolved using the Maximum Entropy Method developed in this thesis (data originally analysed by Pushkarev et al. (2005)).

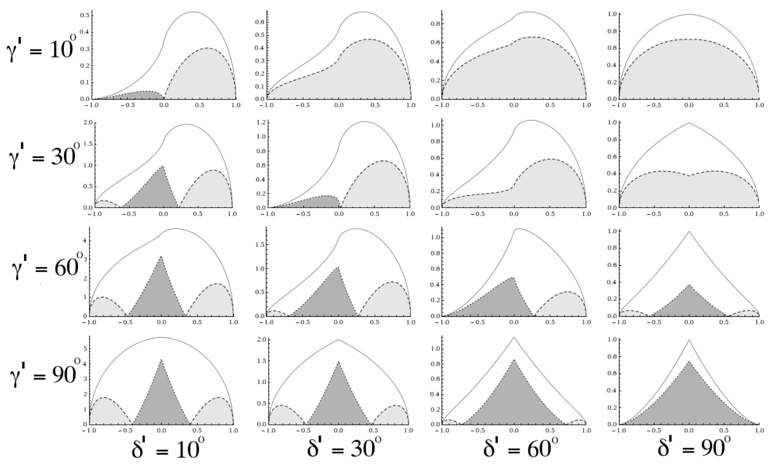


Figure 1.13: Transverse polarisation configurations of a simple helical magnetic field model in an AGN jet. Models parametrised by line of sight with the observer, δ', and pitch angle of the helix γ', both in the rest frame of the jet. Dark regions indicate longitudinal polarisation (EVPAs aligned with the jet), while lighter regions indicate transverse polarisation (EVPAs perpendicular to the jet). The x axis runs transversely across a slice of a model jet. Image from Murphy, Cawthorne & Gabuzda (2013).

of a spine-sheath structure some configurations of pitch angle and line of sight angle, however many other variations are possible. The exact configuration observed also depends on the resolution of the observing instrument. Algorithms such as the Maximum Entropy Method which offer improved resolution over the standard CLEAN algorithm may be able to better image the actual polarisation configuration of the jet – allowing a better estimate of the viewing and pitch angles to be obtained.

Asymmetric features in intensity and polarisation are regularly observed in AGN jets. These could be explained by pressure gradients across the jet or by the presence of a helical magnetic field. However if the asymmetry occurs in a region without any other evidence for the existence of a pressure gradient, such as kinking or twisting, then the presence of such asymmetries together with some of the other features discussed above constitutes strong evidence for a helical magnetic field in the jet.

Murphy et al. (2013) present the results of fitting a simple helical magnetic field model first, developed by Laing (1981), to the jet of Markarian 501. In agreement with earlier analyses by Papageorgiou (2005), they found that the model produced higher than observed levels of polarisation. However, by adding a term dividing the magnetic field into helical and tangled components, Murphy et al. achieved

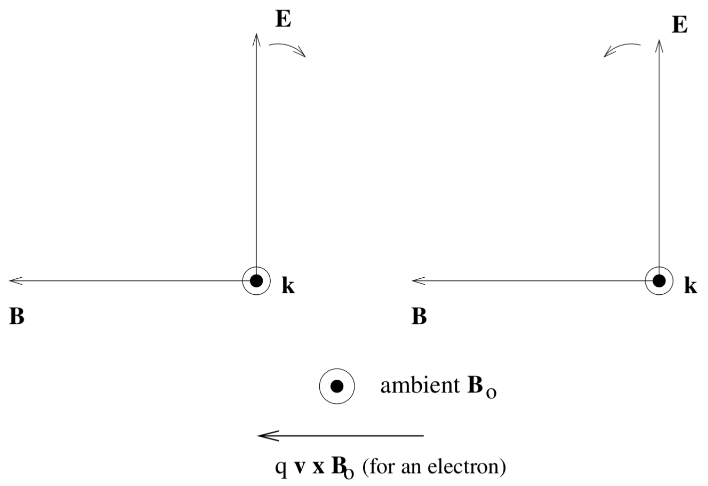


Figure 1.14: The Lorentz Force in Faraday Rotation for the RCP and LCP components of an electromagnetic wave. The Lorentz force is aligned with the rotation of the LCP component, but opposed to the rotation of the RCP component. Image from Gabuzda (2008).

good fits to the asymmetric transverse profiles in Markarian 501 and successfully calculated a value for the rest frame viewing angle of the jet of approximately 83°. This corresponded to a magnetic field that contains approximately 60% of its energy in a helical component and 40% in a tangled component.

1.7 Faraday Rotation and evidence for Helical Magnetic Fields

Faraday rotation is the rotation of the angle of polarisation of light as it passes through a charged plasma in the presence of an external magnetic field. This rotation occurs because the left circularly polarised (LCP) and right circularly polarised (RCP) components of the radio wave respond differently to the magnetic field and propagate at different speeds, thereby leading to a rotation in the overall angle of polarisation.

This can be understood as follows. Consider Figure 1.14 which shows the LCP and RCP components of an electromagnetic wave passing through a region with a magnetic field and a charged plasma. The direction of the Lorentz force, given by the cross product of the velocity of the electron $\vec{v}$ and the magnetic field $\vec{B}$, is aligned with the rotation of the LCP component and opposed to the RCP component.

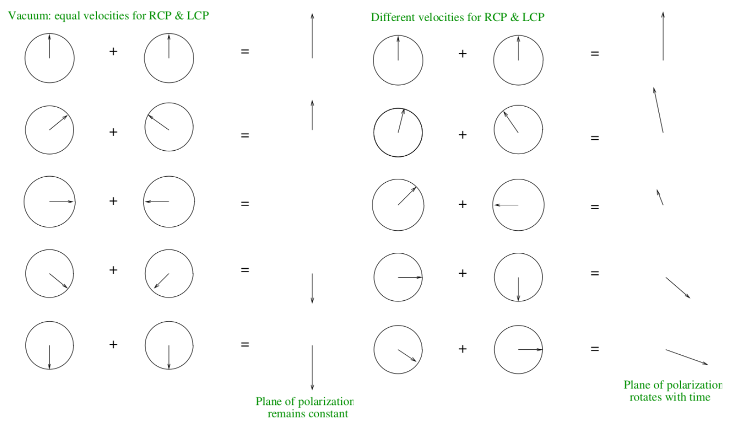

Vacuum: equal velocities for RCP & LCP
Different velocities for RCP & LCP
Plane of polarization remains constant
Plane of polarization rotates with time

Figure 1.15: Faraday rotation caused by different components of the polarisation of the wave experience different refractive indices, and therefore propagation speeds. Image from Gabuzda (2008).

The different responses of the LCP and RCP components of the wave to the Lorentz force result in the two components experiencing different indices of refraction as they travel through the plasma. This introduces a difference in the speed at which each part of the wave propagates, therefore the original polarisation angle is not preserved. Figure 1.15 demonstrates the difference between LCP and RCP components of a wave propagating in a vacuum (without any Faraday rotation) and in a plasma with a magnetic field, where the plane of polarisation rotates as the components travel at different velocities.

The amount by which the polarisation angle rotates at a wavelength λ can be expressed as

$$\chi_{obs} = \chi_0 + RM\lambda^2 \tag{1.98}$$

where RM, known as the Faraday rotation measure, is defined as

$$RM = \frac{e^3}{8\pi^2\varepsilon_0 m_e^2 c^3}\int n_e \vec{B}.\vec{dl} \tag{1.99}$$

where n_e is the number density of the plasma and $\vec{B}.\vec{dl}$ returns the line of sight magnetic field. It is of note that the $\frac{1}{m_e^2}$ term means that any contribution by protons to the Faraday rotation process will be negligible compared to the electronic contribution.

The existence of terms describing the magnetic field and electron density in Equa-

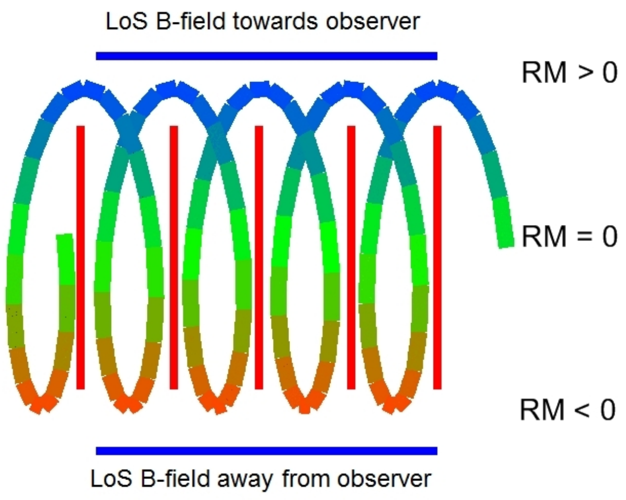


Figure 1.16: Faraday rotation measure gradient due to the changing line of sight component of a helical magnetic field. Image from Reichstein & Gabuzda (2011).

tion (1.99) makes the measurement of the Faraday rotation occurring in any region an important diagnostic of the magnetic field environment of the jet. Changes in the Faraday rotation measure (RM) can be due to either changes in the line of sight magnetic field, or changes in the electronic number density, however if a change in the sign of the RM is observed, the only explanation can be a reversal in the direction of the line of sight magnetic field.

1.7.1 Transverse FR Gradients and Magnetic Fields

The presence of a toroidal or helical magnetic field threading a jet can give rise to a very distinctive Faraday rotation measure signature – namely, a transverse gradient across the jet. Figure 1.16 shows how such a systematic transverse gradient in Faraday RM may come about as the line of sight component of the magnetic field changes across the jet. In some cases observed transverse gradients exhibit the sign change demonstrated in Figure 1.16, however in many cases the finite resolution of the observing array and the geometry of the jet mean that, while a systematic transverse RM gradient is observed, the RM does not change sign over the length of the gradient. Murphy et al. (2013) demonstrate the effects of finite resolution on observed RM distributions for a variety of simple jet models.

The Faraday rotation measure at a specific location in the jet of an AGN can be calculated by obtaining observations of the jet at multiple different frequencies. The resulting maps must be convolved with the same restoring beam (see Section 2.2) in order to have the same units. As the resolution and size of the restoring beam scales with frequency according to Equation (1.6), the conservative choice for this single resolution is the resolution corresponding to the lowest observed frequency. By calculating the observed polarisation angle at each frequency, Equation (1.98) allows both the Faraday rotation measure and the intrinsic polarisation angle to be found using a simple linear fit.

The correct EVPA calibrations must be applied to the polarisation angles before this fitting process, as well as any correction that may be needed to account for local Faraday rotation occurring as the emission from the jet passes through our own galaxy. Section 6.5 discusses the statistical properties of the linear fitting process in detail – including the effect of EVPA calibration errors on Faraday rotation measure gradients. Recent results Murphy et al. (2013) discuss the reliability of the RM gradients calculated in this manner and find them to be extremely robust when the statistical significance of the gradient is greater than 3σ. In the case where the statistical significance of a gradient is over 3σ this method can be used to investigate RM gradients for extremely narrow jets.

The process of combining multi-wavelength data to create Faraday rotation measure maps is surprisingly robust. The method is not sensitive to errors in the EVPA calibration (see Section 6.5), and while errors can be introduced due to the incorrect alignment of the images taken at different frequencies, these are often negligible. This misalignment is due to the phenomenon of VLBI core shift. The process of interferometric imaging loses the absolute position of the source and as a result most images are simply centred on the brightest emission (the "core"). However Konigl (1981) calculated that the position of the VLBI core shifts depending on the frequency of observation according to

$$r_{core} = Cv_{obs}^{\frac{1}{k_r}} \tag{1.100}$$

where C is a constant $k_r = 1$ if the system is assumed to be in equipartition. To correct for this core shift effect identical features must be aligned across the frequencies used in the experiment. Multiple techniques have been developed to do this – Croke & Gabuzda (2008) developed a method of estimating the pixel shift between maps at different frequencies due to this effect using cross correlation coefficient of maps at different frequencies. A catalogue of core shifts across many sources has been developed by Sokolovsky et al. (2011). In many cases

Faraday RM maps do not show sensitivity to image alignment errors, but such errors can be of critical importance in multi-wavelength calculations involving fewer frequencies, such as the calculation of spectral index.

1.8 Thesis Summary

This thesis describes the development of new methods for high resolution radio astronomy imaging. Two different avenues of improving radio imaging are pursued. Firstly – the development of new applications for the maximum-entropy deconvolution method for VLBI polarisation data, which has intrinsically higher resolution than the CLEAN deconvolution that is usually used in radio astronomy. Secondly – a better statistical understanding of the uncertainties associated with new and existing imaging and analysis methods, which can be used to better understand what can reliably be inferred from observations.

The first part of this thesis describes the development of a new implementation of the Maximum Entropy Method (MEM) deconvolution algorithm to be used in multi-wavelength polarisation studies of AGN. Chapter 2 describes the need for deconvolution and the advantages that the MEM offers in the creation of radio images over the more common CLEAN algorithm – namely higher resolution and a better mathematical basis. The chapter goes on to describe previous work on VLBI polarisation data with the MEM and details the computational algorithm which can be used to implement the MEM.

Chapter 3 introduces original changes to the Cornwell-Evans algorithm described in the previous chapter. The development of an new MEM code is detailed and Monte Carlo simulations are performed which show the advantages of the MEM over CLEAN.

Chapter 4 shows the first results of actually using the new MEM code, PMEM, on real VLBA observations of AGN. Intrinsic MEM model maps are shown, as well as the first MEM based Faraday rotation measure maps of AGN at VLBI resolutions. The contribution the MEM can make to high resolution polarisation studies is illustrated.

Chapter 5 of this thesis consists of results published in Mahmud et al. (2013). It details the reliable detection of Faraday rotation measure gradients in two different AGN and the detection of a reversal in the direction of the gradient in one of them. Various explanations for such a reversal are discussed.

Chapter 6 pursues the second aim of this thesis – the development of an improved statistical understanding of the uncertainties associated with current algorithms and techniques. A new error model for the CLEAN algorithm is introduced and tested with Monte Carlo simulations. The results of the simulations and their implications for the error method are discussed. Statistical issues faced by colleagues in the field are detailed and mathematical solutions are proposed.

Finally, Chapter 7 summarises the results of this thesis and offers conclusions on the work.

Chapter 2

The Maximum Entropy Method

The Maximum Entropy Method (MEM) is a deconvolution algorithm widely used in radio astronomy. It was adapted for use with polarisation data by Holdaway (1990), Holdaway and Wardle (1990) and Sault (1999). Holdaway and Wardle (1990) applied the MEM to VLBI polarisation data, however their code could not be interfaced with other standard imaging packages, and never became generally available. In addition, Holdaway and Wardle (1990) did not consider the possibility of investigating multi- wavelength phenomena such as Faraday rotation with polarisation-sensitive MEM. The following section introduces some concepts from Fourier analysis and outlines the need for deconvolution of radio data, discussing the two major algorithms used - the MEM, and the CLEAN algorithm.

2.1 The Problem of Deconvolution

2.1.1 The Fourier Transform

Fourier Analysis is a powerful tool in the study of waveforms of all kinds, including the radio signals that form the basis of radio astronomy. Bracewell (2000) provides an excellent summary of the applicability of the Fourier transform to problems in physics and engineering. However before the practical uses of the Fourier transform can be considered, the mathematical properties of the transform must be defined. The one-dimensional Fourier Transform of a function $f(x)$ can be defined as

$$F(s) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi xs}dx. \tag{2.1}$$

This transform can then be inverted to obtain the original function using a similar formula:

$$f(x) = \int_{-\infty}^{\infty} F(s)e^{i2\pi xs}ds. \tag{2.2}$$

In order for $f(x)$ to be Fourier transformable both $f(x)$ and its transform $F(s)$ must obey the Dirichlet conditions:

1. $f(x)$ and $F(s)$ must be square integrable, i.e. $\int_{-\infty}^{\infty} |f(x)|^2 dx$ must be finite, showing that $\lim_{|x|\to+\infty} f(x) = 0$.
2. $f(x)$ and $F(s)$ must be single valued functions.
3. $f(x)$ and $F(s)$ must be piece-wise continuous.
4. $f(x)$ and $F(s)$ must have both upper and lower bounds.

2.1.2 The Sampling Theorem

Consider a function $f(x)$ such that its Fourier transform $F(s)$ has only finite support, i.e. $F(s) = 0,\ \forall |s| > s_c$. In this case $f(x)$ is said to be "band-limited" and the sampling theorem of Fourier analysis states that sampling $f(x)$ with a frequency given by $\frac{1}{2s_c}$ is sufficient to completely describe its Fourier transform $F(s)$ and therefore reconstitute the original function $f(x)$ by inverse Fourier transform.

This result has great practical significance as it means that accuracy of the waveform $f(x)$ generated by the Fourier transform is dependent on the cut-off frequency, s_c, of the measurements of its transform $F(s)$. In radio interferometry this cut-off frequency for a particular baseline is determined by the observing wavelength and the length of the baseline, and means that the signal generated by inverting the visibilities will only be accurate down to a scale of $\frac{1}{2s_c}$. If the source has structure on scales smaller than this, the corresponding visibilities would be beyond the effective s_c for the array and therefore not measured.

Another important aspect of the interdependence of time and frequency shown by this result is that it is impossible for a Fourier transformable signal to be both band-limited (its Fourier transform describable with a finite number of frequencies) and time-limited (its waveform describable in a finite amount of time). Only a waveform which continues forever in time can be described with a finite number of frequencies in frequency space, and only a waveform consisting of infinite frequencies can be described in a finite amount of time. Consequently, the more precise one's knowledge of the time-span of a wave, the less one can constrain its spectrum (range of frequencies). This can be expressed as the uncertainty relation of Fourier analysis as follows

$$\Delta x \Delta s \geq \frac{1}{4\pi}. \tag{2.3}$$

This result has major consequences for the study of any form of wave, and finds particular fame as the Heisenberg uncertainty principle relating uncertainty in the position and momentum of a matter-wave in quantum mechanics.

2.1.3 Deconvolution in Radio Interferometry

Consider again the relationship between the image and visibility space in radio interferometry (see Fig. 2.1)

$$V(u,v) = \int_{-\infty}^{\infty} I(x,y) e^{2\pi i(ux+vy)}\ dx\ dy \tag{2.4}$$

$$I(x,y) = \int_{-\infty}^{\infty} V(u,v) e^{-2\pi i(ux+vy)}\ du\ dv. \tag{2.5}$$

Note that this relationship assumes that the vertical w coordinate in the visibility space (see Figure 2.1) can be neglected, effectively assuming all of the antennae are on a single $u - v$ plane. This introduces a distortion in the image similar to the effect of projecting a section of the sky dome on a small plane. This effect is negligible for a sufficiently small patch of the sky dome, obeying the condition that $\frac{x^2+y^2}{2w} << 1$. Therefore for high frequencies or longer baselines the amount of the sky that can be mapped ignoring the w coordinate shrinks, but for small sources close to the phase centre w can be safely ignored.

At first glance at these equations it may appear that the visibility function as recorded by the telescope array may simply be inverse Fourier transformed to recover the original sky brightness distribution, however a simple fact prevents

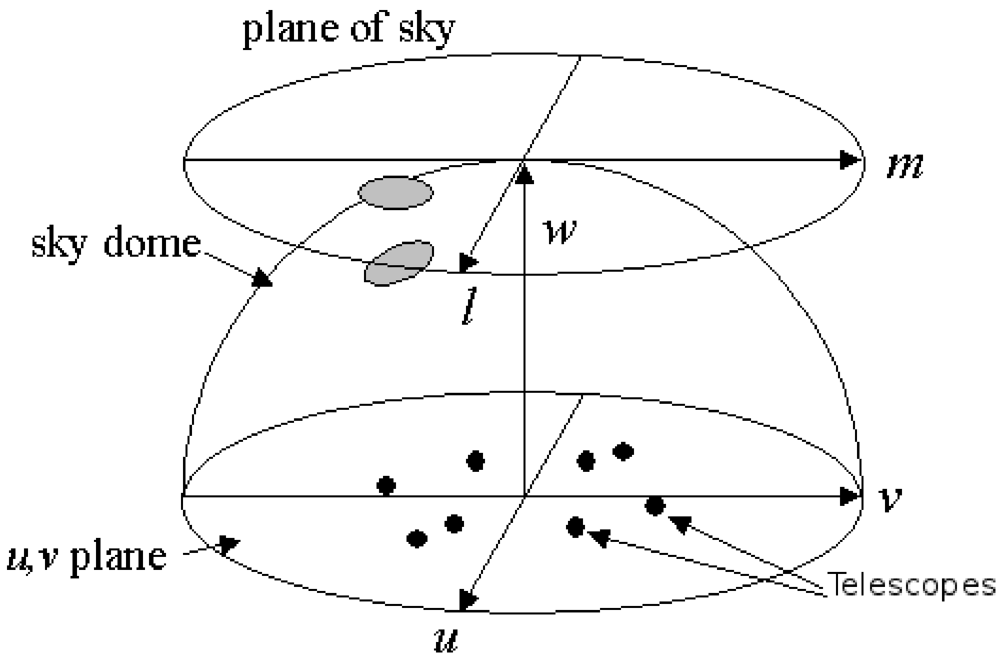


Figure 2.1: The UVW plane. From the lecture notes of Prof. Dale Gary, New Jersey Institute of Technology: http://web.njit.edu/ gary/728/Lecture6.html.

this – not all of the visibility data has been recorded.

An interferometric array can only collect data at the $u-v$ coordinates corresponding to a pair of antennas (a baseline). This situation is remedied somewhat by Earth rotation synthesis, the process by which the rotation of the Earth moves individual baselines across the $u-v$ plane, enabling a more complete sampling, however it is clear from Figure 2.2 that most of the $u-v$ plane still goes unsampled.

This can be represented mathematically by including a binary valued sampling function $S(u,v)$, with a value of 0 for baselines not measured during the observation, and 1 for baselines corresponding to antenna pairs in the array with which data were taken. Thus the recorded visibility data V_r is a subset of $V(u,v)$ and can be written

$$V_r(u,v) = V(u,v)S(u,v). \tag{2.6}$$

Attempting to invert the recorded visibilities V_r with a discrete Fourier transform will not return the original map, instead returning a "dirty" map

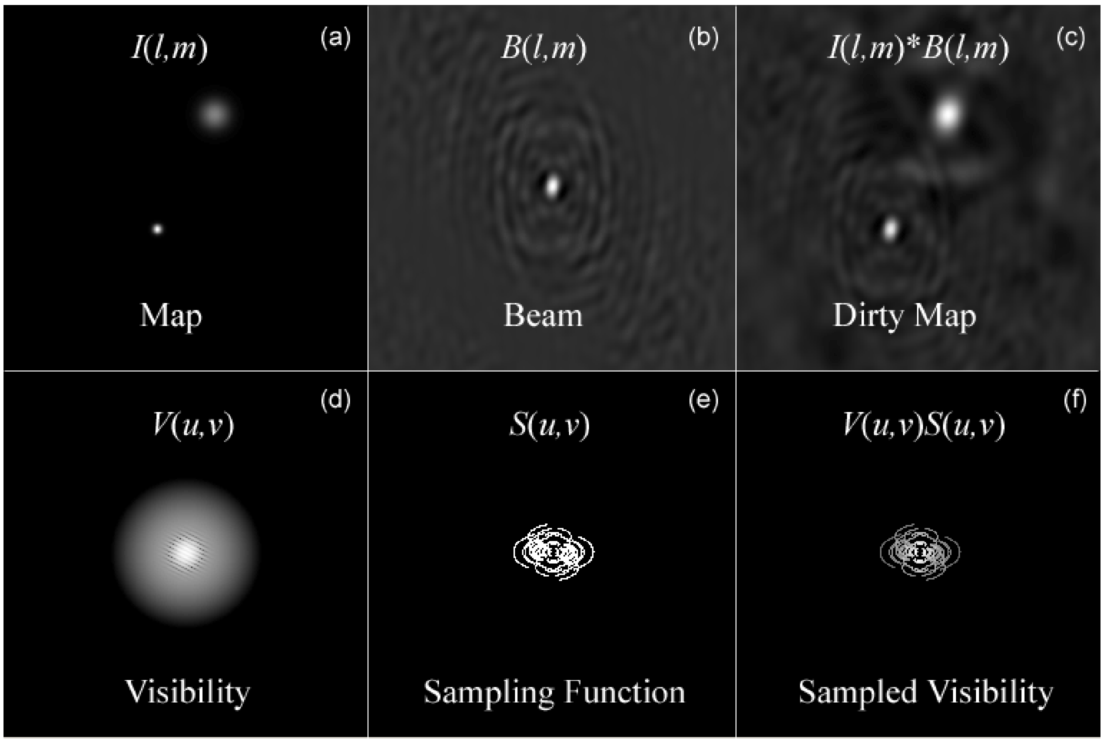


Figure 2.2: The relationship between visibility and image space. Image from the lecture notes of Prof. Dale Gary, New Jersey Institute of Technology: http://web.njit.edu/ gary/728/Lecture6.html.

$$I_{dirty}(x,y) = \sum_r V_r(u,v)e^{2\pi i(ux+vy)}\Delta u\ \Delta v. \tag{2.7}$$

Rewriting $V_r(u,v)$ in terms of $V(u,v)$ and $S(u,v)$ a new formulation of the problem can be derived using the convolution theorem of Fourier analysis (a product in one space is a convolution in the Fourier transformed space)

$$I_{dirty}(x,y) = \int_{-\infty}^{\infty} V(u,v)S(u,v)e^{2\pi i(ux+vy)}du\ dv \tag{2.8}$$

$$I_{dirty}(x,y) = \overline{V(u,v)} * \overline{S(u,v)} \tag{2.9}$$

$$I_{dirty}(x,y) = I_{true}(x,y) * B(x,y) \tag{2.10}$$

where the bar indicates the Fourier transform, "$*$" indicates a convolution and $B(x,y)$, corresponding to the Fourier transform of the sampling function, is known as the dirty beam. In this way, the dirty map produced by the array can be seen as the effect of convolving the true map with the dirty beam of the array as the kernel.

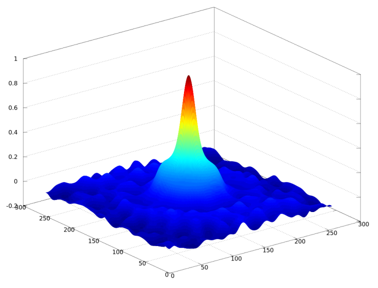
This process can lead to major defects and artefacts in the dirty map, including a relatively low signal to noise ratio and the presence of large "sidelobes" on the map, artificial features introduced due to the systematic asymmetrical sampling of the $u - v$ space. Removing the effect of this convolution, i.e. deconvolving the dirty map, is an essential step in the analysis of spatial data of astronomical objects.

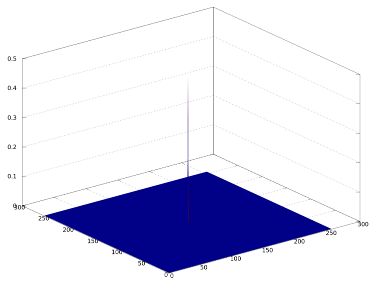
This is not a straightforward task – the only way to truly recover the original sky brightness distribution would be to fully sample the $u - v$ plane. Nevertheless, even though only a small sample of (noisy) visibilities are recorded for each observation, there exist numerous mathematical techniques which attempt to account for the effect of the missing visibilities in a consistent and sensible fashion. The most popular techniques in radio astronomy are variants of the CLEAN and MEM algorithms, the fundamentals of which are outlined in Sections 2.2 and 2.3, respectively. A further explanation of the problem of deconvolution can be found in Starck and Murtagh (2006).

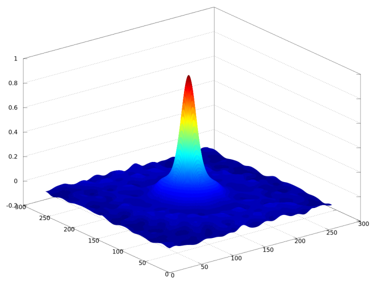
2.2 The CLEAN Algorithm

First developed by Högbom (1974) the CLEAN algorithm is a quick, conceptually clear, computationally efficient method of deconvolving a dirty radio map. It operates as follows

1. The peak of the dirty map is found.
2. The value of the peak is multiplied by a gain factor between 0 and 1 (often 0.1). The resulting number is saved in a list of CLEAN components along with the location of the peak.
3. A δ function with the amplitude and position of the CLEAN component is convolved with the dirty beam. The result is subtracted from the dirty map. This removes the effect of the emission in the CLEAN component from the dirty map (convolving the CLEAN component with the dirty beam gives the response of the array to the CLEAN component of emission alone). See Figure 2.3b.
4. This process is then repeated, removing components of emission from the dirty map until the peak of the residual map is within 3σ of the RMS noise on the map.


(a) The dirty map.


(b) The partially cleaned model.


(c) The partially cleaned map.

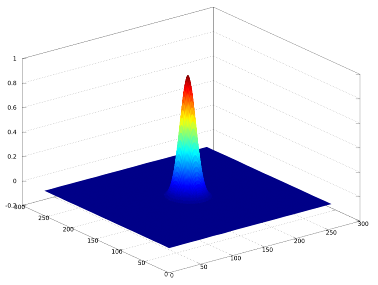

(d) The final cleaned map.

Figure 2.3: Different steps of the CLEAN algorithm as it deconvolves a Gaussian source. The x and y dimensions are in pixels, while the z (height) direction is the Stokes I flux in Jy. Figure 2.3c is a CLEAN map made with a single CLEAN component (visible in Figure 2.3b).

5. The entire list of CLEAN components is then convolved with the CLEAN beam. This is a Gaussian fit to the primary lobe of the dirty beam, thereby having approximately the correct shape and resolution, but without any sidelobes (see the difference in removing the sidelobes associated with a single CLEAN component between Figures 2.3a and 2.3c. The residual map is then added to the convolved CLEAN components to give the final CLEAN map.

The CLEAN algorithm has evolved over the years since it was first introduced and today many different variants of it are in use, including the standard CLEAN algorithm (Clark 1980) and multiscale CLEAN (Wakker 1988). It is suitable for use in imaging both intensity maps and the polarisation Stokes parameters Q, U and V. The Clark CLEAN algorithm is the most widely used deconvolution algorithm in VLBI polarisation studies of AGN.

The CLEAN algorithm, like any algorithm, has systematic flaws that can reduce the quality of the maps it produces. In CLEAN the representation of an emitting source as a series of δ functions may be somewhat accurate for point sources (completely unresolved by the array), but is less suitable for extended emission. There is also often a degree of subjectivity introduced by rejecting certain CLEAN components that may be visually identified as being associated with a sidelobe rather than actual emission (include such components during the self-calibration phase of the data processing can have a major negative effect).

2.3 The Maximum Entropy Method

The Maximum Entropy Method was originally developed by Jaynes (1957) and was first used to deconvolve radio images in the 1970s (Wernecke & D'Addario 1977, Frieden & Wells 1978). Cornwell & Evans (1985) outlined an implementation of the MEM as a constrained optimisation method based on a consideration of the function

$$\begin{aligned} J = H(I_m, P_m) - \alpha\chi^2_I(\widehat{V_I}, \widetilde{V_I}) \\ - \beta(\chi^2_Q(\widehat{V_Q}, \widetilde{V_Q}) + \chi^2_U(\widehat{V_U}, \widetilde{V_U})) - \gamma G \end{aligned} \tag{2.11}$$

where H is the entropy of a model map of the source, χ^2 is a measure of the difference between the model $(\widehat{V})$ and the observed $(\widetilde{V})$ visibilities (there are three χ^2 terms, one for intensity, Stokes I, and two for the polarisation Stokes Q and U parameters), α, β and γ are Lagrangian optimisation parameters and G is a function equal to

$$G = \sum_k I_k - Z.S.F. \tag{2.12}$$

where I_k is the Stokes I intensity at pixel k, $\sum_k I_k$ is the total flux of the MEM model and Z.S.F. stands for the Zero-Spacing Flux, an estimate of the true flux which can be gained from looking at the amplitudes of the shortest visibilities. A form of entropy suitable for polarisation emission developed by Gull & Skilling (1984) and used by Holdaway & Wardle (1990) and Sault, Bock & Duncan (1999) is

$$\begin{aligned} H = & -\sum_k I_k(log(\frac{2I_k}{IB_k e}) + \frac{1+m_k}{2}log(\frac{1+m_k}{2}) \\ & + \frac{1-m_k}{2}log(\frac{1-m_k}{2})) \end{aligned} \tag{2.13}$$

where IB_k is the flux at pixel k of a bias map (normally chosen to be a flat map with a total flux equal to the flux estimated for the source) and I_k and m_k are the Stokes Intensity (I) flux and fractional polarisation, respectively, of pixel k. This exact form of entropy was suggested in Sault et al. (1999), though a very similar form was used by Holdaway & Wardle (1990).

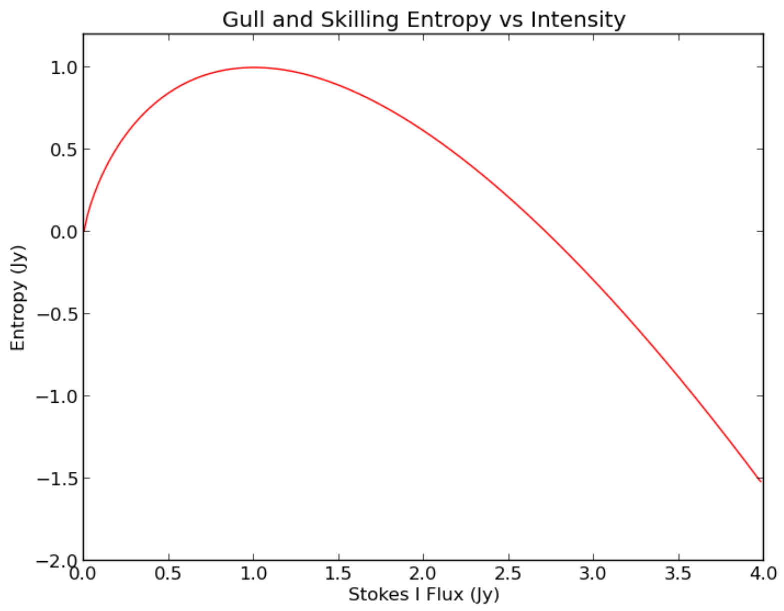
The Gull and Skilling entropy, H, is a form of Shannon entropy (often used to describe the information content of a dataset) which has been generalised to include information on the polarisation of the data. Shannon entropy can be thought of as a measure of disorder in information. The less certain the value of a variable is, the higher Shannon entropy it will have and conversely, a certain outcome has minimal Shannon entropy. The original form of Shannon entropy for a variable x with possible values $x_1, x_2, ..., x_n$ can be expressed as

$$H(x) = -\sum_i^n P(x_i)\log_b P(x_i) \tag{2.14}$$

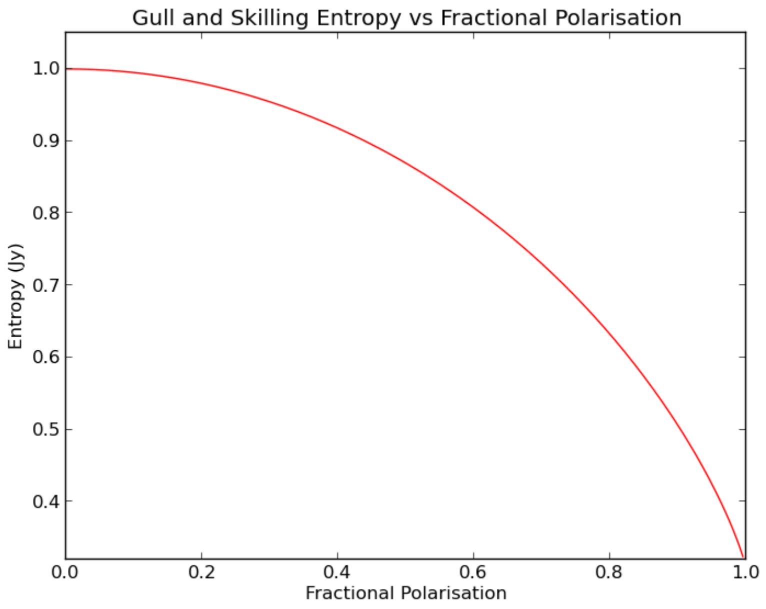
where $P(x_i)$ is the probability that x will take the value x_i and b is the base of the logarithm – often taken to be 2 or e, depending on the application of the concept. Equation (2.13) is a generalisation of Equation (2.14) above treating fractional polarisation as a measure of order in the magnetic field, and biasing the Stokes I part of the equation to rescale the logarithm to the units used in the radio map.

An examination of the form of H gives an indication as to how it will react to different types of sources – see Figure 2.4 for a graphical illustration. The Gull and Skilling entropy of a source that is described well by the bias map is high – the data does not require a meaningful model at all, and the amount of useful information which can be gained from the model is minimal. A source which has low fractional polarisation (i.e. disordered magnetic field) will also have high Gull and Skilling entropy.

The Gull and Skilling entropy is thus maximised for an unpolarised source that is identical to the bias map. This is the map that MEM will produce in the absence of any data that forces it to make a more complicated model. If data is provided to the MEM model, the χ^2 terms in Equation (2.11) force MEM to make a model



(a) Entropy Vs Intensity.



(b) Entropy Vs Fractional Polarisation.

Figure 2.4: Gull and Skilling entropy for a single pixel (Equation (2.13)) plotted with a bias pixel equal to 1 Jy. Figure 2.4a shows how the entropy varies with Stokes I flux with zero fractional polarisation. Figure 2.4b shows how the entropy varies with fractional polarisation with a constant Stokes I flux of 1 Jy. Note that the maximum entropy occurs for a model map with no polarisation and equal intensity to the bias map.

that maximises the Gull and Skilling entropy while reproducing the data and observed flux to within noise levels. In this way the MEM can be thought of as a ‘tug of war’ between the Gull and Skilling entropy, favouring disorder, and the χ^2 terms and flux condition in Equation (2.11), favouring fidelity to the observed data.

2.4 The Cornwell-Evans Algorithm

The Cornwell & Evans (1985) algorithm, with modifications by Holdaway & Wardle (1990) to support deconvolution of polarisation data, implements the MEM by maximising Equation (2.11). The general method can be thought of as follows

1. Generate an initial model (often flat, with a flux equal to the expected flux).
2. Assign appropriate values for the Lagrangian parameters, α, β and γ. These parameters must be set so that the step size they induce is not too large. Section 2.4.3 discusses a method of calculating suitable initial Lagrangian values and updating them as the method converges.
3. Find the step sizes in the Stokes I, Q and U model maps which will maximise J for current values of the Lagrangian parameters.
4. Take the step, with numerical safeguards to ensure that the new Stokes I, Q and U maps have not changed too quickly.
5. Convolve the new model with the dirty beam. Test to see how close the result is to the dirty map.
6. Repeat steps 2 to 5 until the agreement between the model and the measurements is within the estimated uncertainty (or as close to it as possible).
7. Convolve the MEM model with a restoring beam (see Section 2.5 for a discussion about the appropriate restoring beam to use).

The following sections outline how each step may be achieved, beginning with how one can use the Newton–Raphson method to find the maximum of a function.

2.4.1 The Newton–Raphson Method

If one has a one dimensional function $f(x)$ one can take a Taylor series approximation for x close to x_0 to second order as

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \tfrac{f''(x_0)}{2}(x - x_0)^2 \qquad (2.15)$$

where the prime notation indicates differentiation with respect to x. Finding the critical point of this equation using $\frac{df(x)}{d(x-x_0)} = 0$ gives a step in x, Δx that brings f to an extremum such that

$$f'(x_0)+f''(x_0)\Delta x = 0 \qquad (2.16)$$

$$\Delta x = -\frac{f'(x_0)}{f''(x_0}. \qquad (2.17)$$

If f is quadratic, then Δx is exact and leads to the solution. Otherwise it can take some iterations to reach the extremum. This can be generalised to N dimensional functions as follows

$$g(\vec{x}) = g(\vec{x_0}) + \nabla g(\vec{x_0})(\vec{x} - \vec{x_0}) + \tfrac{1}{2}(\vec{x} - \vec{x_0})^T \nabla\nabla g(\vec{x_0})(\vec{x} - \vec{x_0}) \qquad (2.18)$$

where ∇ is the gradient operator, $\nabla\nabla$ is the gradient operator applied twice, and the superscript T indicates the transpose of a vector. In this case, the step that leads to the extremum is

$$\vec{\Delta x} = (-\nabla\nabla g(\vec{x_0}))^{-1}\nabla g(\vec{x_0}). \qquad (2.19)$$

2.4.2 Finding the maximum of J

The outcome of the MEM is three maps (Stokes I, Q and U) that together maximise the function J in Equation (2.11). Using the multidimensional Newton–Raphson method outlined above, where N is now the number of pixels in a map, the step changes needed in the I, Q and U maps are

$$\vec{\Delta I} = (-\nabla\nabla J_I)^{-1}\nabla J_I \qquad (2.20)$$

$$\vec{\Delta Q} = (-\nabla\nabla J_Q)^{-1}\nabla J_Q \qquad (2.21)$$

$$\vec{\Delta U} = (-\nabla\nabla J_U)^{-1}\nabla J_U \qquad (2.22)$$

where $\vec{\Delta I}$ is the change in the Stokes I map (a vector with a length equal to the number of pixels in the map, N), ∇J_I is the partial derivative of J with respect to Stokes I intensity (a vector of the same length) and $\nabla\nabla J_I$ is the matrix of second derivatives (the Hessian) of J with respect to Stokes I, an NxN matrix.

Using the same notation, where the subscript indicates the Stokes parameter used to differentiate, the gradients of J for Stokes I, Q and U can be evaluated from Equation (2.11) and written as follows

$$\nabla J_I = \frac{\partial H}{\partial I_i} - \alpha \frac{\partial \chi_I^2}{\partial I_i} - \gamma \tag{2.23}$$
$$\nabla J_Q = \frac{\partial H}{\partial I_Q} - \beta \frac{\partial \chi_I^2}{\partial Q_i} \tag{2.24}$$
$$\nabla J_U = \frac{\partial H}{\partial I_U} - \beta \frac{\partial \chi_I^2}{\partial U_i}. \tag{2.25}$$

The Hessians of J can then be evaluated as

$$\nabla\nabla J_I = \frac{\partial^2 H}{\partial I_i \partial I_j} - \alpha \frac{\partial^2 \chi_I^2}{\partial I_i \partial I_j} \tag{2.26}$$
$$\nabla\nabla J_Q = \frac{\partial^2 H}{\partial Q_i \partial Q_j} - \beta \frac{\partial^2 \chi_Q^2}{\partial Q_i \partial Q_j} \tag{2.27}$$
$$\nabla\nabla J_U = \frac{\partial^2 H}{\partial U_i \partial U_j} - \beta \frac{\partial^2 \chi_U^2}{\partial U_i \partial U_j}. \tag{2.28}$$

Evaluating the first derivatives of the Gull and Skilling entropy in Equation (2.13) gives the following expressions

$$\frac{\partial H}{\partial I_i} = -\frac{1}{2} \log[\frac{I_i^2}{B_i^2}(1 - m_i^2)] \tag{2.29}$$
$$\frac{\partial H}{\partial Q_i} = \frac{Q_i}{2 m_i I_i} \log[\frac{1 - m_i}{1 + m_i}] \tag{2.30}$$
$$\frac{\partial H}{\partial U_i} = \frac{U_i}{2 m_i I_i} \log[\frac{1 - m_i}{1 + m_i}]. \tag{2.31}$$

The second derivatives can then be calculated:

$$\frac{\partial^2 H}{\partial I_i I_j} = -\delta_{ij}\frac{1}{I_i}(\frac{1}{1-m_i^2}) \tag{2.32}$$

$$\frac{\partial^2 H}{\partial Q_i Q_j} = \delta_{ij}(\frac{1}{2m_i I_i})(\log[\frac{1-m_i}{1+m_i}]+(\frac{Q_i}{I_i m_i})^2(\frac{2m_i}{m_i^2-1} - \log[\frac{1-m_i}{1+m_i}])) \tag{2.33}$$

$$\frac{\partial^2 H}{\partial U_i U_j} = \delta_{ij}(\frac{1}{2m_i I_i})(\log[\frac{1-m_i}{1+m_i}]+(\frac{U_i}{I_i m_i})^2(\frac{2m_i}{m_i^2-1} - \log[\frac{1-m_i}{1+m_i}])). \tag{2.34}$$

Note that the Hessians of the entropy are diagonal NxN matrices. If the Hessians of the χ^2 terms were similar, then evaluating the steps for Stokes I, Q and U in Eqs. 2.20 to 2.22. Unfortunately this is not the case, as can be seen when the χ^2 terms are written out as follows. Note that the χ^2 function for the Stokes I visibilities can be written in terms of the weighted difference between the model and data visibilities:

$$\chi_I^2(\widehat{V_I}, \widetilde{V_I}) = \sum_{k=0}^{N_{vis}} w_k(\widehat{V_I} - \widetilde{V_I})^2 \tag{2.35}$$

$$= \sum_{k}^{N_{vis}} w_k[(\Re[\widehat{V_I}] - \Re[\widetilde{V_I}])^2 + (\Im[\widehat{V_I}] - \Im[\widetilde{V_I}])^2] \tag{2.36}$$

where N_{vis} is the number of visibilities, w_k are any weights applied to the visibilities (weights are often applied according to perceived reliability, or to favour certain baselines) and the scripts $\Re$ and $\Im$ indicate the real and imaginary components, respectively. Note that the model visibilities are just the Fourier transform of the model map with N pixels

$$\widehat{V_I} = \sum_{i=0}^{N} I_i e^{2i\pi(ux+vy)} \tag{2.37}$$

$$= \sum_{i=0}^{N} I_i(\cos(2\pi(ux+vy)) + i\sin(2\pi(ux+vy))) \tag{2.38}$$

$$\frac{\partial\Re[\widehat{V_I}]}{\partial I_i} = \cos(2\pi(ux+vy)) \tag{2.39}$$

$$\frac{\partial\Im[\widehat{V_I}]}{\partial I_i} = \sin(2\pi(ux+vy)). \tag{2.40}$$

Therefore

$$\begin{aligned}
\frac{\partial \chi_I^2}{\partial I_i} &= 2\sum_{k=0}^{N_{vis}} w_k[(\Re[\widehat{V_I}] - \Re[\widetilde{V_I}])\cos(2\pi(ux+vy)) \\
&+ \ [(\Im[\widehat{V_I}] - \Im[\widetilde{V_I}])\sin(2\pi(ux+vy))] \qquad (2.41) \\
&= 2\Re[\sum_{k=0}^{N_{vis}} w_k(\widehat{V_I} - \widetilde{V_I})e^{-2i\pi(ux+vy)}] \qquad (2.42) \\
&= 2(\sum_{j=0}^{N} P_{i,j}I_j - DMI_i) \qquad (2.43)
\end{aligned}$$

where $\sum_{j=0}^{N} P_{i,j}I_j$ represents the convolution of the model I map with the dirty beam P, and DMI_i is the i^{th} pixel of the dirty map. Similarly for Stokes Q and U

$$\frac{\partial \chi_Q^2}{\partial Q_i} = 2(\sum_{j=0}^{N} P_{i,j}Q_j - DMQ_i) \qquad (2.44)$$

$$\frac{\partial \chi_U^2}{\partial U_i} = 2(\sum_{j=0}^{N} P_{i,j}U_j - DMU_i). \qquad (2.45)$$

In each case the second derivative evaluates to

$$\frac{\partial^2 \chi_I^2}{\partial I_i \partial I_j} = \frac{\partial^2 \chi_Q^2}{\partial Q_i \partial Q_j} = \frac{\partial^2 \chi_U^2}{\partial U_i \partial U_j} = 2P_{i,j}. \qquad (2.46)$$

$P_{i,j}$ is a NxN dimensional non-diagonal matrix. The presence of this term in Equations 2.26 to 2.28 means that the Hessian of J would be non-diagonal, vastly increasing the computational power required to evaluate them. Therefore Cornwell & Evans (1985) make the approximation that

$$2P_{i,j} \approx 2Q \qquad (2.47)$$

where the value of Q is not critical, but should represent the power in the main lobe of the primary beam. Sault (1990) finds it appropriate to set $Q = \sqrt{\sum_i^N P_i^2}$, the gain for white noise.

It is notable that even though Equation (2.11) specifies the χ^2 terms in terms of the visibilities, the derivatives of these terms can be expressed in image space.

Thus to avoid using any visibilities and simplify computation using only the dirty maps the following approximation can be made

$$\chi_I^2(\widetilde{V}) \approx \frac{E}{Q} \tag{2.48}$$

where E is the misfit in the image plane, given by

$$E = \sum_i^N (\sum_{j=0}^N P_{i,j} I_j - DMI_i)^2 \tag{2.49}$$

with corresponding approximations made for Stokes Q and U:

$$F = \sum_i^N (\sum_{j=0}^N P_{i,j} Q_j - DMQ_i)^2 + \sum_i^N (\sum_{j=0}^N P_{i,j} U_j - DMU_i)^2. \tag{2.50}$$

Thus efficient expressions have been calculated for all of the terms in Equations (2.20) to (2.22) and, given a suitable choice of values for the Lagrangian parameters α, β and γ, the equations specify steps in Stokes I, Q and U which maximise the value of J in Equation (2.11). However, given the complexity of J these steps will not immediately maximise the function – they are rather the first steps in an iterative process of finding the Stokes I, Q and U maps that maximise the entropy while minimising disagreement with the observed data. Depending on the source and any scaling used to limit the effect of disastrously inaccurate steps in Stokes I, Q and U, it can be many hundreds of iterations before the MEM converges completely. Critical to this convergence is the choice of values for the Lagrangian parameters at the first and all subsequent iterations. The following section prescribes a method of estimating such values.

2.4.3 Updating the Lagrangian Parameters

The Lagrangian parameters α, β and γ need to be updated in such a way that the model data agrees with the observations as best as possible, while also maximising the entropy. All Lagrangian parameters can be initially set to zero and a set of values appropriate for making the first changes to the model map found by the same mechanism used in all subsequent iterations described below. It is useful to use the notation introduced in Cornwell & Evans (1985) where it was noted that the term $(-\nabla\nabla J_I)^{-1}$ as it appears in Equation (2.26) can be considered a metric of the image space, in such a way that the vector scalar product between X and Y is

$$|X \cdot Y| = \sum_{i,j} X_i(-\nabla\nabla J_I)^{-1} Y_j. \tag{2.51}$$

Using this notation the idea that E, F and G are required to be minimal while also having maximum entropy can be stated mathematically as

$$\begin{pmatrix} \nabla E \cdot \nabla J_I \\ \nabla F \cdot \nabla J_{Q,U} \\ \nabla G \cdot \nabla J_I \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{2.52}$$

Expanding this, the following equation in the form $Ax - b = 0$ is achieved

$$\begin{pmatrix} \nabla E \cdot \nabla E & \nabla E \cdot \nabla F & \nabla E \cdot \nabla G \\ \nabla F \cdot \nabla E & \nabla F \cdot \nabla F & \nabla F \cdot \nabla G \\ \nabla G \cdot \nabla E & \nabla G \cdot \nabla F & \nabla G \cdot \nabla G \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} - \begin{pmatrix} \nabla E \cdot \nabla H \\ \nabla F \cdot \nabla H \\ \nabla G \cdot \nabla H \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{2.53}$$

This can be solved for α, β and γ. However the gain of the MEM, the amount J changes by for a change in Stokes I, can be defined as $\frac{\nabla J \cdot \nabla J}{1 \cdot 1}$. In a situation where the gain is small, this means maximising J does not change the Stokes I map by much. A different approach in calculating the Lagrangian parameters can be useful in such cases – namely to change them from their existing values in such a way that the resulting changing in $\nabla E \cdot \nabla J$, for example, is $-E$, the amount by which E must be changed for a perfect agreement. This has the effect of neutralising the E term and increasing J. Applying this logic to all the Lagrangian parameters gives

$$\begin{pmatrix} \nabla E \cdot \nabla E & \nabla E \cdot \nabla F & \nabla E \cdot \nabla G \\ \nabla F \cdot \nabla E & \nabla F \cdot \nabla F & \nabla F \cdot \nabla G \\ \nabla G \cdot \nabla E & \nabla G \cdot \nabla F & \nabla G \cdot \nabla G \end{pmatrix} \begin{pmatrix} \alpha + \Delta\alpha \\ \beta + \Delta\beta \\ \gamma + \Delta\gamma \end{pmatrix} - \begin{pmatrix} \nabla E \cdot \nabla H \\ \nabla F \cdot \nabla H \\ \nabla G \cdot \nabla H \end{pmatrix} = \begin{pmatrix} -E \\ -F \\ -G \end{pmatrix} \tag{2.54}$$

which can be re-arranged to solve for the changes in the Lagrangian parameters as

$$\begin{pmatrix} \Delta\alpha \\ \Delta\beta \\ \Delta\gamma \end{pmatrix} = \begin{pmatrix} -E - \nabla E \cdot \nabla J_I \\ -F - \nabla F \cdot \nabla J_{Q,U} \\ -G - \nabla G \cdot \nabla J_I \end{pmatrix}. \tag{2.55}$$

The recommend values of $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$ can then be clipped to ensure that the resulting change in the image is small enough that the image will eventually converge. Cornwall & Evans (1985) suggest an image can be considered converged when

$$|\nabla J \cdot \nabla J| = \varepsilon |1 \cdot 1| \tag{2.56}$$

where $\varepsilon \leq 0.01$. This condition can be used to place limits on changes in α, β and γ such that if the change in, for example, α is small enough to allow the MEM to converge, $\Delta\alpha$ must lie within the maximum and minimum bounds given by the two solutions to the corresponding quadratic equation in $\Delta\alpha$:

$$|(\nabla J - \Delta\alpha \nabla \chi_I^2) \cdot (\nabla J - \Delta\alpha \nabla \chi_I^2)| = \varepsilon |1 \cdot 1| \tag{2.57}$$

and the similar equations in $\Delta\beta$ and $\Delta\gamma$:

$$|(\nabla J - \Delta\beta \nabla(\chi_Q^2 + \chi_U^2)) \cdot (\nabla J - \Delta\beta \nabla(\chi_Q^2 + \chi_U^2))| = \varepsilon |1 \cdot 1| \tag{2.58}$$

$$|(\nabla J - \Delta\gamma \nabla G) \cdot (\nabla J - \Delta\gamma \nabla G)| = \varepsilon |1 \cdot 1|. \tag{2.59}$$

Thus both (2.53) and (2.55) offer two different methods of estimating appropriate values for α, β and γ. In many cases the recommended values are close, however implementating a MEM which examines both sets of values and chooses the most appropriate set given the current quality of the convergence maximises both the quality and speed of the resulting deconvolution.

2.5 Advantages of the MEM

By iteratively maximising J in equation (2.11), the MEM method develops a model of the source which maximises the Gull and Skilling entropy of the model (the model has lowest possible polarisation, and looks as much like the bias map as the data allows), while also reproducing the observed data to within noise levels. This results in a balance between entropy (representing the effects of unsampled visibilities and thermal noise) and fidelity to the observed data. This method of deconvolution, while not as straightforward as the CLEAN algorithm, is statistically and mathematically well-founded and can produce extremely well deconvolved maps comparable to, and in some cases better than, the CLEAN algorithm.

Unlike the standard CLEAN algorithm, MEM does not model the source as a series of δ functions. Instead MEM models the source as a continuous distribution – a more physically realistic model, but one which is computationally much more demanding. This increases the effective resolution of MEM, as it is not necessary to convolve the MEM model with the CLEAN beam. This means that the theoretical resolution of MEM is the Nyquist sampling theorem limit for the observation (Equation (1.6)), although thermal and systematic noise may prevent drawing useful information at such small scales. It proves useful to convolve the MEM model map with a small beam to smoothen out these variations, although this limits the resolution of the resulting map. From experience, a beam of about $\frac{1}{2}$ to $\frac{1}{4}$ of the CLEAN beam works well for most sources.

MEM is also known for its mathematical property of "super-resolution". This property can be derived as follows by considering the χ^2 function that forces the model to converge in Equation (2.11), see for example Holdaway (1990). χ^2 can be written as follows

$$\chi^2 = \sum_{k=0}^{N_{vis}} \omega_k (V_m - V_{obs})^2 \tag{2.60}$$

where there are N_{vis} observed visibilities, V_{obs}, the same number of model visibilities, V_m, and ω_k is the weight of each visibility. The equation can be re-written in terms of the Real and Imaginary parts of the visibilities and differentiated with respect to the model Stokes I flux as follows,

$$\chi^2 = \sum_{k=0}^{N_{vis}} \omega_k (Re(V_{m,k} - V_{obs,k})^2 + Im(V_{m,k} - V_{obs,k})^2) \tag{2.61}$$

$$\begin{aligned}\frac{\partial\chi^2}{\partial I_i} = 2\sum_{k=0}^{N_{vis}} \omega_k(Re(V_{m,k} - V_{obs,k})\frac{\partial Re(V_{m,k})}{\partial I_i} \\ +Im(V_{m,k} - V_{obs,k})\frac{\partial Im(V_{m,k})}{\partial I_i}).\end{aligned} \tag{2.62}$$

Note that the model visibility $V_{m,k}$ is related to the pixels in the model flux map I_j by the following Fourier Transform relationship

$$V_{m,k}(u,v) = \sum_{j=1}^{j=N} I_j(x,y)e^{2\pi\imath(u_k x_j + v_k y_j)} \tag{2.63}$$

where I_j is the intensity of the j^{th} pixel on the model map, x and y are the coordinates of the pixel on the model map in radians, and u and v are the $u-v$ coordinates of the model visibility in wavelengths. This means that the derivatives of the real and imaginary parts of the model visibilities can be written

$$\frac{\partial Re(V_{m,k})}{\partial I_i} = Cos(2\pi(u_k x_i + v_k y_i)) \tag{2.64}$$

$$\frac{\partial Im(V_{m,k})}{\partial I_i} = Sin(2\pi(u_k x_i + v_k y_i)). \tag{2.65}$$

These formulae can be inserted into Eqn. 2.62 to give

$$\begin{aligned}\frac{\partial\chi^2}{\partial I_i} = 2\sum_{i=0}^{k} \omega_k[Re(V_{m,k} - V_{obs,k})Cos(2\pi(u_k x_i + v_k y_i)) \\ +Im(V_{m,k} - V_{obs,k})Sin(2\pi(u_k x_i + v_k y_i))],\end{aligned} \tag{2.66}$$

which can be re-written as

$$\frac{\partial\chi^2}{\partial I_i} = 2\sum_{i=0}^{k} \omega_k Re(V_{m,k} - V_{obs,k})Cos(2\pi(u_k x_i + v_k y_i)). \tag{2.67}$$

This is the function which drives the convergence of the Maximum Entropy Method. Consider the case of the initial MEM model being set to a flat map with a total flux equal to the expected total flux of the source. The corresponding initial model visibility is a δ function with an amplitude equal to the total flux. For data visibilities with small baselines the cosine term in Equation (2.67) will be positive and close to its maximum value. This causes these baselines to have the effect of increasing or decreasing the flux in all pixels until the model visibilities agree with the data visibilities. These baselines are not sensitive to the exact location of the flux in the image.

As MEM progresses the model visibility function will change from its initial δ function shape, becoming broader and broader as information about the extended structure is captured from longer and longer baselines. This is an iterative process, and information contained in the longer baselines may not be acted upon until larger changes required to make the model agree at shorter baselines have been made. As the model visibility function broadens, for longer visibilities it will remain below the data visibilities (this gap will decrease as convergence proceeds). This means that for longer baselines the term involving the difference in the real visibilities in Eqn. 2.67 will be negative. As a further negative sign is applied to the χ^2 term in Eqn. 2.11 this means that for convergence to continue (J to be maximised), the cosine term in Eqn. 2.67 must be positive. Thus convergence stops when this term becomes negative, i.e., when

$$2\pi(u_k x_i + v_k y_i) = \frac{\pi}{2}. \tag{2.68}$$

This equation gives the resolution limit of MEM (when the MEM model stops converging to data). It is clear from the presence of x_i and y_i in the equation that the resolution of MEM varies over the map. The very smallest values of x_i and y_i for which the equation holds true correspond to

$$x_{min} = \frac{1}{4\ u_{max}} \qquad y_{min} = \frac{1}{4\ v_{max}}, \tag{2.69}$$

where x_{min}, y_{min} are the resolutions in the x and y directions, and u_{max} and v_{max} are the maximum baseline in the u and v directions respectively. This resolution is a factor of 4 below the best-case resolution expected from the Nyquist sampling limit, and therefore details at this resolution scale do not directly reflect information which has been recorded by the array. However, as MEM's model of the source as a continuous distribution is quite realistic, the MEM can model the source at resolution levels below those corresponding to the maximum baselines in the array used for the observations. This modelling is done by creating a structure that can reproduce the data at the observed resolution levels while also having maximum Gull and Skilling entropy. In this way, the MEM produces a conservative model of the source at resolutions below the Nyquist limit.

2.6 Summary

The problem of deconvolution does not have an easy solution – the data required to image the source as it truly appears have not been measured and any attempt to deconvolve an image is, in some sense, an attempt to make a good guess at what contributions the unobserved visibilities might make to the source. The CLEAN algorithm is an excellent choice for deconvolving most images – it is simple, fast and assumes little about the underlying source structure. Nonetheless, the Maximum Entropy Method appears to promise multiple advantages over the CLEAN algorithm.

The MEM operates by making a model of the source that maximises the informational entropy of the source (keeping it as random as possible), while also agrees with the observed visibilities. In this way the MEM model can be thought of as a very conservative vision of the true appearance of the source. Section 2.5 discussed the advantages of such a model and the high resolution maps which may be made from it. The fact that all Stokes parameters are deconvolved together may also yield a more consistent image of a source than CLEAN's independent treatment of each parameter.

Thus the MEM appears to be a very promising algorithm with which to deconvolved polarised VLBI maps of AGN jets. The increased resolution and attention to polarisation may show features that cannot be seen with CLEAN and may show other features in greater detail. The following two chapters of this thesis discuss the implementation of new MEM code based closely on previous implementations in the AIPS and MIRIAD software suites, the testing of the new code on simulated VLBI sources and the use of the code on real observations of AGN jets.

Chapter 3

A New Implementation of the MEM for Polarisation VLBI

As may be evident from the discussion of the MEM in Chapter 2 the mathematical rigour and foundation of the MEM over the CLEAN algorithm comes at a cost of increased complexity and computational demand. Implementing the MEM in code and successfully using such a code to deconvolve a VLBI image can be more challenging than simply using CLEAN – but the rewards are correspondingly greater. Versions of the MEM suitable for the imaging of Stokes I data alone are implemented in many popular imaging suites, including the NRAO's AIPS (Greisen 2002) and CASA (McMullin et al. 2007) suites. While these implementations of the MEM can deconvolve VLBI Stokes I emission using the MEM, they have no support for deconvolving Stokes Q and U maps (this is due to the form of Shannon entropy chosen). Conversely, a version of the MEM suitable for imaging polarised emission is present in CSIRO's MIRIAD data reduction package (Sault, Teuben & Wright 1995), however it is incompatible with data from the VLBA.

A new computer program, Polarised Maximum Entropy Method (PMEM), has been written to address this problem and implement a version of the MEM based on the Cornwell-Evans algorithm (see Section 2.4) with support for polarisation. This implementation builds on work done by Holdaway & Wardle (1990) in studies of the polarisation properties of AGN at VLBI scales and has enabled the creation of new multi-waavelength VLBI polarisation images, as well as the first Faraday rotation measure VLBI maps of AGN made using the MEM.

3.1 Alternations to the Cornwell-Evans Algorithm

While the Cornwell-Evans algorithm describes the overall method of implementing the MEM in a computationally efficient manner, there are many different changes that can be made to optimise the performance of the algorithm for VLBI observations of AGN, and polarisation observations in particular. To this end, a new MEM based deconvolution code has been written which uses the Cornwell-Evans algorithm with the polarisation adaptations by Holdaway & Wardle (1990) and Sault (1999). Although this is not the first implementation of the MEM for polarisation VLBI, this implementation has improved on the algorithms used by Holdaway & Wardle (1990) and Sault et al. (1999) in several ways, as will be described below. It is more user-friendly than the earlier code of Holdaway & Wardle (1990), and, unlike that earlier code, interfaces well with AIPS and CASA. PMEM is also very suitable for multiwavelength studies such as the creation of Faraday rotation measure maps of AGN. The main changes made to the standard Cornwell-Evans algorithm in PMEM are listed in this section.

3.1.1 Improving the polarisation maps

A major improvement in the Stokes Q and U maps produced with the algorithm can be obtained with the inclusion of a term weighting the relative sizes of Equations (2.49) and (2.50). These equations return the misfits between the convolved model map and the dirty map for the Stokes I intensity and polarised intensities, respectively. However as mentioned in Section 1.6 the maximum polarisation that can be expected from synchrotron radiation is 75%. In most sources the actual detected polarisation is far less, often of the order of 10% or so. This results in the misfit for the Stokes I intensity almost always being much greater than the polarisation misfit when both are measured in the same unit. While this is strictly true, it can result in a optimal MEM model that agrees very well with the Stokes I intensity, but quite badly with the polarisation data. As a major purpose of PMEM is to provide increased resolution for polarisation observations of AGN jets this behaviour is unwelcome. To overcome this issue a new parameter w_p is introduced to increase the weight of the polarisation misfits relative to the Stokes I misfit. The parameter is introduced in such a way that the total misfit in polarisation is calculated as

$$F = w_p(\sum_{i}^{N_{pix}}(\sum_{j=0}^{N_{pix}} P_{i,j}Q_j - DMQ_i)^2 + \sum_{i}^{N_{pix}}(\sum_{j=0}^{N_{pix}} P_{i,j}U_j - DMU_i)^2). \qquad (3.1)$$

A suitable value for w_p can vary between different datasets, depending on the relative complexities of the intensity and polarisation structures and the noise present in each. It is suggested that the user divide the total polarised flux of the source being deconvolved by its total intensity, thus determining an average fractional polarisation, m, for the source. Setting $w_p = \frac{1}{m}$ will then equalise the total intensity and polarisation terms in Equation (3.1). The final Stokes I map is relatively insensitive to changes in w_p, however the use of an appropriate value can significantly improve the final Stokes Q and U maps. A value of 2.0 was found to be a good compromise between weighting up the importance of good convergence in the polarisation Stokes parameters while not causing a major change to the Stokes I map, though higher values were often appropriate.

An alternative method of implementing this feature would be to weight the Stokes Q and U maps differently. This would be of use in sources where there was detectable flux in both Stokes Q and U with a significant difference in magnitude between them. In practice it was found that a single parameter was sufficient to allow good convergence in all three Stokes parameters.

3.1.2 Accounting for poorly performing sources

Different observations have different beam profiles and difference noises in the observed visibilities. While the unaltered algorithm performs well for the majority of VLBI jet observations, some jets have structure observed on the same scale as the dirty beam making them difficult for both the MEM and CLEAN to image well. The inclusion of a stepping factor Δ_{step} and an edge pixel exclusion option $N_{exclude}$ give the user some flexibility in responding to challenging sources. More often, they allow the optimisation of maps made for better performing sources.

Δ_{step} is applied as a multiplicative factor to the maximum size of a step allowed during the maximisation of Equation (2.11) using the Newton-Raphson method. As discussed in Section 2.4.3 the gain of the MEM, i.e. the change in J for a change in the Stokes I model map can be calculated as $\frac{\nabla J \cdot \nabla J}{1 \cdot 1}$. If the gain is high, J may change rapidly, while if the gain is low J may be difficult to change. Applying Δ_{step} to the maximum step limit allowed in the Newton-Raphson as

$$\Delta_{max} \propto \Delta_{step} \frac{\nabla J \cdot \nabla J}{1 \cdot 1} \tag{3.2}$$

allows the user to manually slow down the changes in a single iteration of the algorithm in the case of poor performance or increase the changes (reducing the compute time) in the case of well-performing sources. This parameter was especially useful in optimising the performance of PMEM for the Monte Carlo simulations detailed further on in this chapter.

$N_{exclude}$ is a parameter which forces the flux in the outer $N_{exclude}$ pixels of the model map to be zero. This parameter is useful in reducing problems caused by aliasing due to the Fast Fourier Transforms (FFTs) performed on the model map. As repeated convolutions are required for every iteration of the MEM, it is computationally much more efficient to use the FFT method, which has a complexity of $O(n \log n)$, than perform a direct Discrete Fourier Transform (DFT) with a complexity of $O(n^2)$. For example to convolve a 512×512 pixel image with another 512×512 image requires 3 discrete Fourier transforms (twice, to calculate the FT of the two images, and once more to find the inverse FT of the product). Using the DFT this would require roughly $3 \times 512 \times 512 = 786432$ operations, whereas the FFT would require only $3 \times 512 \times \log 512 = 4162$ operations – approximately 100 times faster.

This increase in speed can come at a price – the FFT is susceptible to aliasing. In Fourier aliasing the long wavelength visibilities that would be needed in visibility space to model sharp small scale changes in image space are not recorded in the $N \times N$ FFT of the image. This causes the inverse Fourier transform of the FFT to contain imperfections and artefacts that were not present in the original image. In the case of using FFTs to convolve MEM models with the dirty beam this effect often manifests itself as much higher than would be expected flux in the outer pixels of the resulting convolved map. This effect can perpetuate itself when this false flux creates a local distortion in the residual maps calculated from the convolved map and the MEM actually begins to include the flux in its model of the source, leading to a runaway-type effect.

One way in which it is possible to reduce aliasing related to the use of the FFT is to "zero-pad" the images being transformed, i.e. to add on extra pixels at the edge of the map to contain the higher visibilities that would otherwise go unsampled. However, in addition to the extra computation needed in the FFT of a padded map, the potentially sharp feature in such zero-padded maps can cause further aliasing (the optimisation of such a process is a currently a field of active study). As the majority of sources do not suffer greatly from aliasing effects, it has been found easier to clip the offending aliased pixels rather than zero pad the entire image. The amount of clipping required to remove aliasing effects can vary from a few pixels, to up to 10 or 20% of the image and can be specified using the $N_{exclude}$ parameter. In the case where the amount of clipping needed becomes large, the dirty maps may need to be remade in a larger size. This effectively zero pads the image, and gives more room to clip unneeded pixels from the edge. Note that the clipping is performed in the model and residual maps, therefore the value of the final map at the outer $N_{exclude}$ pixels is not reliable.

While the AIPS task "VTESS" automatically clips the outer 25% of an image, PMEM allows the user to specify any value deemed appropriate. Such a value can be determined by examining the outer pixels of the model and residual maps for any signs of aliasing (higher or lower than expected flux) and setting the $N_{exclude}$ parameter to exclude such pixels. Alternatively, making a large image and taking a similar approach to VTESS would allow a conservative number of pixels to be clipped without needing to examine the model and residuals maps.

3.1.3 Diagonalising the Hessian

Section 2.4.2 describes how the Hessian of J may be diagonalised with the approximation

$$2P_{ij} \approx 2Q \tag{3.3}$$

where P_{ij} is an NxN dimensional matrix based off the dirty beam and Q is a factor which should represent the power in the main lobe of the primary beam. Sault (1990) suggested setting Q equal to the following expression

$$Q = \sqrt{\sum_{i}^{N_{pix}} P_i^2} \tag{3.4}$$

where P_i is the dirty beam. This represents the gain of the array for white noise. It has been found useful to introduce a manual term, q_{factor} to tweak the Q parameter as follows

$$Q = q_{factor} \sqrt{\sum_{i}^{N_{pix}} P_i^2}. \tag{3.5}$$

Varying q_{factor} in conjunction with Δ_{step} as defined in Equation (3.2) is a useful technique in optimising the deconvolution process for poorly performing sources. A value of $q_{factor} = 1$ is equivalent to the suggestion of Sault (1990), however the choice of a low value can slow down the algorithm in steps where changing Δ_{step} would have little effect. A value of $q_{factor} = 0.5$ was found suitable in many cases, though some sources required values around $q_{factor} = 0.05$. In cases where the source performs very well, a values higher than 1 can result in faster deconvolution. It is suggested that the user experiment with a range of values for q_{factor} to achieve maximum convergence.

3.2 The PMEM Software

PMEM was written to implement a VLBI MEM capable of deconvolving polarisation data using Holdaway & Wardle (1990)'s extension of the Cornwell-Evans algorithm to Stokes Q and U with the modifications outlined in Section 3.1. The C++ programming language was chosen as it is fast, efficient and suitable for all forms of numerical and scientific computation. It also has a wide range of external libraries available which were used to further reduce the computational time for the MEM, as well as the time needed to design the software.

Care was taken at all times to write computationally efficient code, using OpenMP to parallelise operations where possible. The external library FFTW (The Fastest Fourier Transform in the West, Frigo and Johnson 2005) was used to perform the Fast Fourier Transforms (FFTs) needed in the convolutions. LAPACK, a high performance linear algebra library, was used to perform the matrix calculations needed for MEM. The FITS (Flexible Image Transport System) file format was used at all times, ensuring compatibility with all of the major astronomical software packages. To achieve this the CFITSIO library was used, and additional programs were written to interface between CFITSIO and PMEM, allowing FITS files to be read and generated.

The open-source code of AIPS's 'VTESS' task, by Tim Cornwell, and MIRIAD's 'PMOSMEM' task, by Robert Sault, were also of great help in writing PMEM. Both tasks contain excellent implementations of the Cornwell-Evans algorithm and were of immense help in avoiding numerical problems and deciding how to structure PMEM.

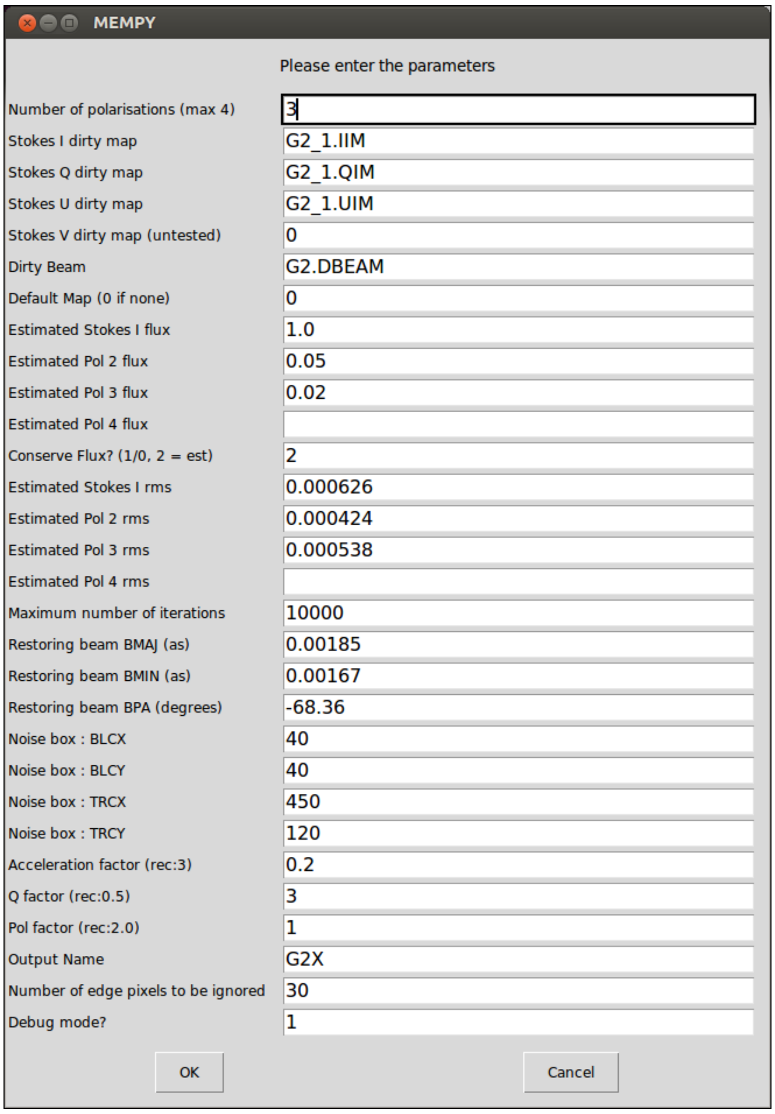
PMEM requires the user to provide the dirty Stokes I, Q and U maps of the source as FITS images. The dirty beam of the observation is also required. The user is then asked to estimate the flux of the source and the final RMS noise that might be achieved. Some other options related to the quantities discussed in Section 3.1 can also improve the performance of the algorithm. To provide a user friendly interface with the C++ code, a Python front-end, "mempy", is used to allow the user to enter these values in a windowed environment (see Figure 3.1). The program can also be used without the MEMPY interface – this form may be suitable for the inclusion in an imaging pipeline or integration with an external software suite.

The code, while numerically intensive, is quick to process maps of 1024x1024 or less and the results are written out into multiple FITS files along with a log of the deconvolution process. For each of the 3 Stokes parameters FITS files containing the following are generated

- The MEM model map for the Stokes parameter.
- The MEM model map convolved with the restoring beam.
- The residual map (the difference between the MEM model convolved with the dirty beam and the dirty map).
- The final MEM map. This is the MEM map convolved with the restoring beam with the appropriately scaled residual map added on (the residual map requires scaling to convert from Jy per old beam to Jy per restoring beam).

3.2.1 Computational Effort

The overall computational effort required to generate a single MEM map is considerably greater than that needed to make corresponding maps with the CLEAN algorithm. The computational cost of both techniques is dominated by the FFTs and inverse FFTs used to convolve their respective models with the dirty beam via the convolution theorem, however performing a FFT on CLEAN's delta function model is computationally much less expensive than performing a FFT on the continuous model used by MEM.



Please enter the parameters

Number of polarisations (max 4)	3
Stokes I dirty map	G2_1.IIM
Stokes Q dirty map	G2_1.QIM
Stokes U dirty map	G2_1.UIM
Stokes V dirty map (untested)	0
Dirty Beam	G2.DBEAM
Default Map (0 if none)	0
Estimated Stokes I flux	1.0
Estimated Pol 2 flux	0.05
Estimated Pol 3 flux	0.02
Estimated Pol 4 flux	
Conserve Flux? (1/0, 2 = est)	2
Estimated Stokes I rms	0.000626
Estimated Pol 2 rms	0.000424
Estimated Pol 3 rms	0.000538
Estimated Pol 4 rms	
Maximum number of iterations	10000
Restoring beam BMAJ (as)	0.00185
Restoring beam BMIN (as)	0.00167
Restoring beam BPA (degrees)	-68.36
Noise box : BLCX	40
Noise box : BLCY	40
Noise box : TRCX	450
Noise box : TRCY	120
Acceleration factor (rec:3)	0.2
Q factor (rec:0.5)	3
Pol factor (rec:2.0)	1
Output Name	G2X
Number of edge pixels to be ignored	30
Debug mode?	1

Figure 3.1: A screenshot of the MEMPY interface to PMEM.

To mitigate this effect, the high performance FFT library, FFTW, was used to optimise PMEM's Fourier transformations. In addition to the immediate speed benefit associated with using a high-speed library, PMEM's performance was further increased by enabling multi-threaded processing within FFTW and allowing FFTW to create an optimised FFT plan for transformations with the dimensions of the specific maps being deconvolved.

Although high efficiency numerical C++ programs often make use of Standard Template Library (STL) vectorization to optimise performance, the use of C and Fortran based external libraries such as FFTW, LAPACK and CFITSIO made reliance on C++ specific features inefficient and inflexible in some cases. This led to the decision to use OpenMP enabled FOR loops as opposed to STL functions in most cases.

PMEM runs efficiently on modern multi-core desktops and laptops, making use of all available cores and deconvolving maps of 1024x1024 or less in a matter of minutes. The exact time required to complete a deconvolution depends on the source and the parameters used, and can range from about a minute to ten minutes. The results presented in Chapter 4 were imaged using a laptop computer, however as the Monte Carlo simulations described in Section 3.3 required repeated imaging of hundreds of sources, the “Stokes” supercomputer at the Irish Centre for High End Computing (ICHEC) was used to perform the MEM imaging. The version of PMEM complied for Stokes was built using the efficient (and expensive) Intel compiler and numerical libraries available on Stokes, resulting in efficiencies over the GNU compiler and open-source libraries used in the normal version of PMEM.

3.3 Monte Carlo Testing

PMEM was initially tested and developed using a variety of real VLBI datasets. Though the software appeared to be functioning correctly it was not possible to fully characterise the performance of the algorithm based on real data. To this end a series of Monte Carlo simulations on model sources were designed that would characterise the ability of PMEM to deconvolve realistic sources. CLEAN based deconvolutions were also performed as a benchmark.

A Gaussian and Triple Gaussian source were designed in Octave (an open-source MATLAB-like numerical software suite) and a typical UV coverage was selected (see Figure 3.2). New C++ software UVFILL2 was written and used to generate simulated observations of the model sources using the chosen UV coverage. Thermal noise was added to the visibilities in such a way as to create realistic noise levels in the final CLEAN maps. This was done by examining the RMS deviation of flux in regions far from the source in real maps – often found to be in the region of 0.5 mJy/Beam, and adding thermal noise to the visibilities until the resulting CLEAN maps had approximately the right level of noise. The random element of the thermal noise added to the visibilities was achieved using the GNU Scientific Library random Gaussian function with zero mean and a user specified standard deviation, seeded with the current time multiplied by the process ID of the current CPU thread running UVFILL2.

One hundred UV datasets were generated for each model, each with different thermal noise added. The data were loaded into AIPS and the AIPS task IMAGR was used to produce dirty maps of each dataset, which were then exported as FITS files. All imaging was performed without any UV tapering and a Brigg's ROBUST parameter of 0, resulting in an even compromise between natural and uniform UV weighting. The cell size used in IMAGR was set to be small enough so that MEM's super-resolution of data could be tested appropriately. A result of this was that the image size had to be significantly increased. This, in combination with the large number of files being processed, and the higher computational demands of the MEM lead to a computer at the Irish Centre for High End Computing (ICHEC) being used to run the MEM Monte Carlo simulations. The MEM imaging was performed using estimates of the final RMS noise from CLEAN images made in the usual way, and using the correct (known) fluxes of the sources.

The final maps were convolved with the CLEAN beam, and beams corresponding to $\frac{1}{4}$, $\frac{1}{3}$, $\frac{1}{2}$ of the CLEAN beam (see Tables 3.1 and 3.3). Results from the CLEAN algorithm as implemented by the IMAGR task in AIPS were also generated using an AIPS script and standard CLEAN imaging techniques. A single model source was imaged manually and used to set the FLUX parameter – the lowest CLEAN component flux allowed, for the imaging script to three times the background noise observed in the manual image. The NITER parameter, governing the maximum number of iterations allowed, was set high enough that the FLUX parameter was the limiting factor in the automated CLEAN performed by the script. Again, all imaging was performed without any UV tapering and a Brigg's ROBUST parameter of 0, resulting in an even compromise between natural and uniform UV weighting. A gain value of 0.1 and a set of CLEAN windows encompassing the source region in the image were used in each case. These images were also convolved with the smaller beams in order to provide a comparison with the results of the MEM.

Maps corresponding to each Stokes parameter and resolution were compared with the model maps convolved with the corresponding beam. Due to small differences in map centering both the MEM and the CLEAN maps had to be shifted by a small number of pixels to align correctly with the convolved model map. The correct shifting was determined by examining the position of various components in CLEAN and MEM maps and identifying the common shift between them. The same shift was found between all CLEAN and MEM maps. Distributions of the difference between the model and imaged fluxes in both total flux and flux in regions of interest on the source were made and the performance of the two algorithms compared. Distributions of the derived quantities, the fractional

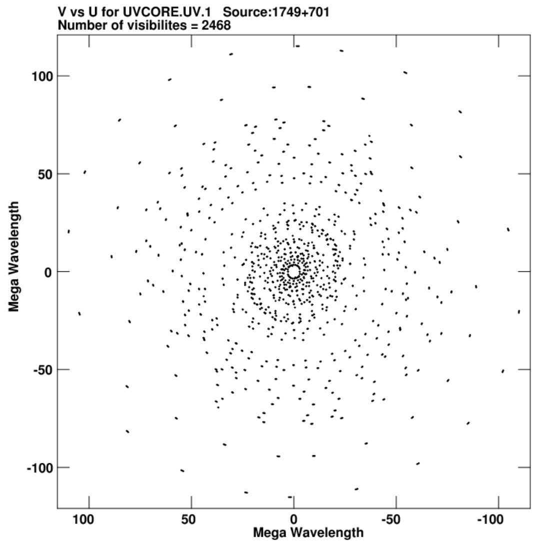


Figure 3.2: The UV distribution used in the Monte Carlo simulations. It corresponds to an observation of 1749+701 at 4.608 GHz in August 2003.

polarisation, m, and the polarisation angle χ were generated as follow:

$$m = \frac{\sqrt{Q^2 + U^2}}{I} \tag{3.6}$$

$$\chi = \frac{1}{2} ArcTan(\frac{U}{Q}). \tag{3.7}$$

Root-Mean-Squared maps of the error for each Stokes parameter and resolution were also made for each algorithm, providing a more general picture of how well the algorithms perform. The following section outlines the results for each of the three source types considered.

3.3.1 Single Gaussian

A Monte Carlo simulation of a single Gaussian with a FWHM of 0.1 mas was carried out using the technique described above. The *UV* coverage used corresponded to an observation with the Very Long Baseline Array at 4.6 GHz, so that 0.1 mas is essentially unresolved. Figure 3.3 shows the single Gaussian convolved with $\frac{1}{4}$ of the normal CLEAN beam for the observation. The Gaussian had a total Stokes I flux of 1 Jy, a Stokes Q flux of 0.05 Jy and a Stokes U flux of 0.02 Jy. These fluxes were chosen to test how MEM performed at realistic intensity and polarisation flux levels. The source structure was otherwise the same (FWHM

Table 3.1: The beams used in the Monte Carlo simulation of the single Gaussian source. The CLEAN beam is a Gaussian fit to the FWHM of the "dirty" beam.

Frac. of CLEAN beam	Major Axis (mas)	Minor Axis (mas)	Position Angle (deg.)
1	1.85	1.67	-68.36
$\frac{1}{2}$	0.925	0.835	-68.36
$\frac{1}{3}$	0.6167	0.5567	-68.36
$\frac{1}{4}$	0.4625	0.4175	-68.36

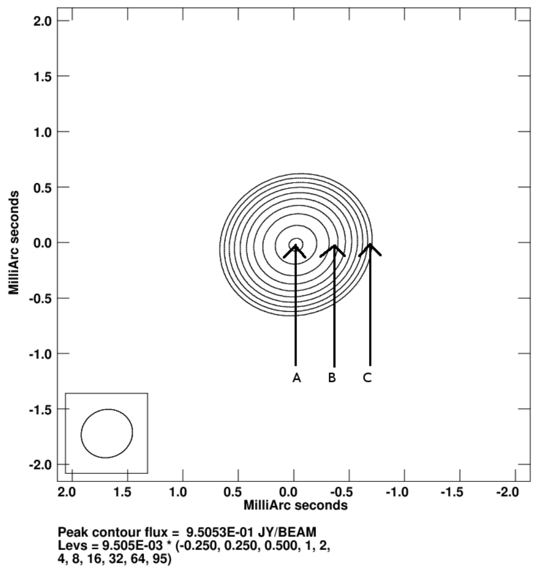
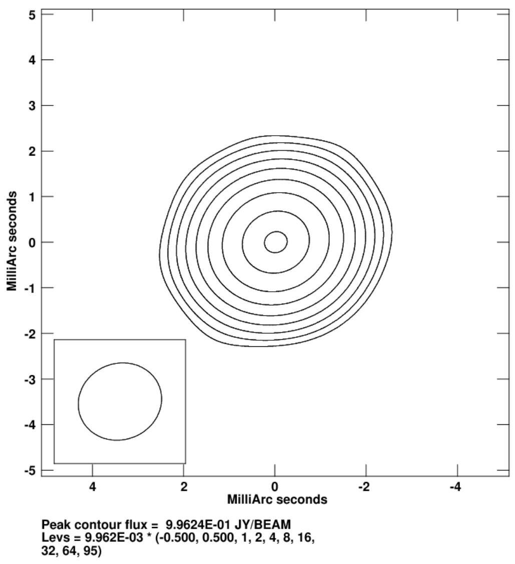


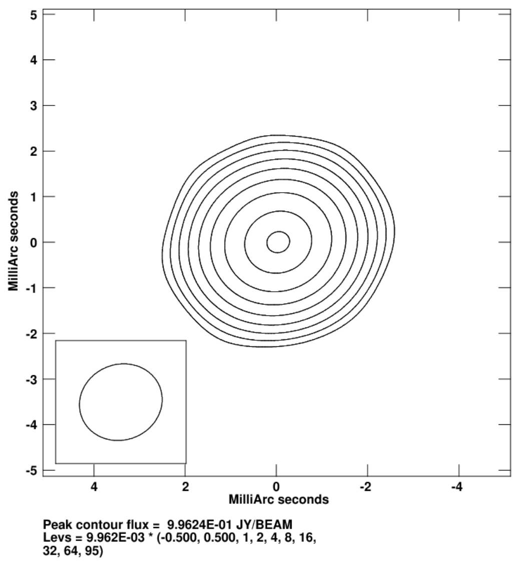
Figure 3.3: The model Gaussian source convolved with 0.25 of the CLEAN beam. Points A, B and C indicate the regions sampled with 0.21 × 0.21 mas boxes in the Monte Carlo experiment.

of 0.1 mas) for all Stokes parameters. Imaging was performed with a cell size of 0.02 mas, 512 × 512 pixel images and the restoring beams outlined in Table 3.1.

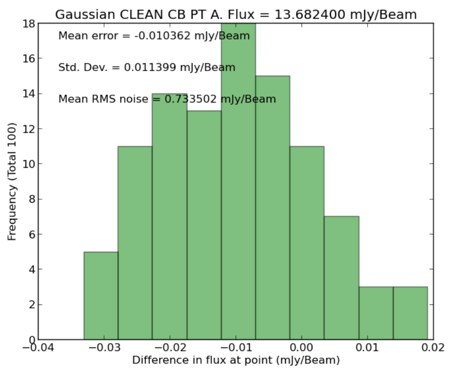
Figure 3.4 shows an example of the criteria used to test how well the MEM and CLEAN algorithms deconvolve the single Gaussian. Sample MEM and CLEAN images from the Monte Carlo simulation are presented, along with histograms indicating the distribution of the difference between the fluxes in the Monte Carlo maps and the real flux in a 0.21 × 0.21 mas region centred on the peak of the maps. The data in this case shows that the CLEAN algorithm is slightly more accurate than the MEM at identifying the correct peak flux at the full CLEAN resolution as the distribution of the CLEAN flux differences has a mean value closer to zero than the corresponding MEM distribution. The narrower profile of the MEM distribution indicates that this inaccuracy is likely to be systematic rather than thermal in nature as the width of the Gaussian is indicative of the size



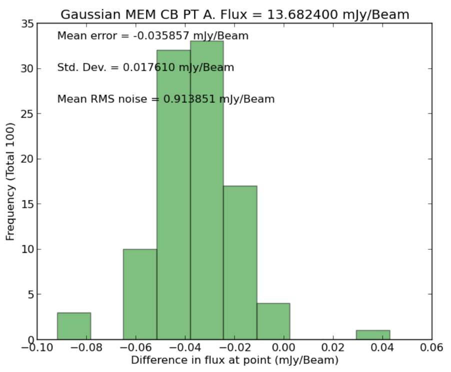
(a) CLEAN



(b) MEM



(c) Distribution of Peak Region



(d) Distribution of Peak Region

Figure 3.4: Sample CLEAN and MEM images of the Gaussian source in Stokes I at CLEAN beam resolution. The histograms give the distribution of the Stokes I flux in a 0.21 $\times$ 0.21 mas region centred point A in Figure 3.9.

of the random variations resulting from the thermal noise added to the visibilities. Similar tests were performed at all resolutions indicated in Table 3.1. In addition to this, the ability of the algorithm to identify the correct Stokes I, Q and U fluxes in regions B and C as indicated in Figure 3.3 was also tested (point A is the peak of the map). The performance of both algorithms in recovering the correct total flux was also examined.

Figures 3.5 and 3.6 (left column) show how well each algorithm performs at identifying the flux (in mJy) in a 0.21 $\times$ 0.21 mas region centred on points A, B and C. A zero error line is included in each plot to show where the data point should be in the case of a perfect deconvolution. Each data point corresponds to the mean error of the algorithm at the corresponding resolution, while the error bars correspond to the standard deviation of the distribution of errors. Thus a large error bar indicates a wide histogram as in Figure 3.4; implying a lot of variation in the accuracy of the algorithm that may not be evident from the mean difference alone.

In all cases at full CLEAN resolution the MEM and CLEAN data points are close together and near the zero error line, indicating a successful deconvolution. As the distance between the data points and the zero error line increases for both algorithms, as the size of the restoring beam is reduced, it is clear that the reliability of both algorithms degenerates at resolutions higher than the CLEAN beam. In particular, both algorithms are generally quite accurate at 1 CLEAN beam and at 0.5 CLEAN beams (as indicated by their proximity to the zero error line), however the performance of both CLEAN and MEM quickly drops off at resolutions lower than 0.5 CLEAN beams.

CLEAN does a better job at finding the correct flux in every region across all Stokes parameters for the higher resolutions. In particular for point A (corresponding to the peak of the Gaussian) CLEAN greatly outperforms the MEM, as might be expected given the suitability of CLEAN's method of modelling the source as a δ function at this point. The MEM however has difficulty in using its continuous model of the source to reach sharp peaks, and ends up underestimating the flux in this region as the size of the convolving beam decreases and the source becomes more sharply defined. This indicates that for real sources where a significant change in flux occurs in an area CLEAN also outperforms the MEM at points B and C, and while the MEM is competitive with CLEAN in Stokes Q and U at these locations, there is little sign of any increase in resolution or accuracy over CLEAN. It is notable that the standard deviations (plotted as error bars) are usually not large enough such that the algorithm is within 3σ of the zero line – a strong indicator of systematic errors in both algorithms.

The plots on the right hand side of Figure 3.6 show how well each algorithm recovers the total flux of the source (defined as the flux inside a box encompassing all of the source). It is immediately obvious that the MEM performs much better than the CLEAN algorithm at higher resolutions, showing little variance at all across all resolutions. This is due to the fact that the MEM maps made at each resolution are generated from the same MEM model, which was obtained from Equation (2.11). This equation explicitly includes a term to ensure that the total

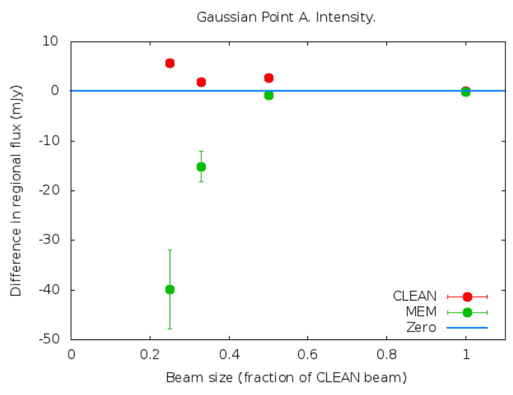
Stokes I flux of the map does not vary far from the estimate of the true flux, therefore the MEM's fidelity to the true flux is not unexpected. It is again of note that the MEM is also similarly successful in recovering the total Stokes Q and U fluxes, even though these Stokes parameters do not have an explicit flux term in Equation (2.11), excepting the χ^2 term.

Figure 3.7 gives the Monte Carlo results of the total polarisation and fractional polarisation in 0.21 × 0.21 regions around points A, B and C. Figs. The plots for the total polarisation (on the left side) show similar results to those for Stokes Q and U where the CLEAN algorithm clearly outperforms the MEM at the core, and there is no suggestion that the MEM is doing a better job at points B and C. Figs. The plots of fractional polarisation on the right side again show that the performance of the two algorithms is similar at lower resolutions, however in this case the MEM outperforms the CLEAN algorithm at higher resolutions at points B and C, where the source is less peaked. This is consistent with MEM's difficulty in modelling sharply peaked sources. It is clear that the MEM is much better at imaging the fractional polarisation that the total polarisation – this is likely due to the explicit inclusion of m rather than p in Equation (2.13).

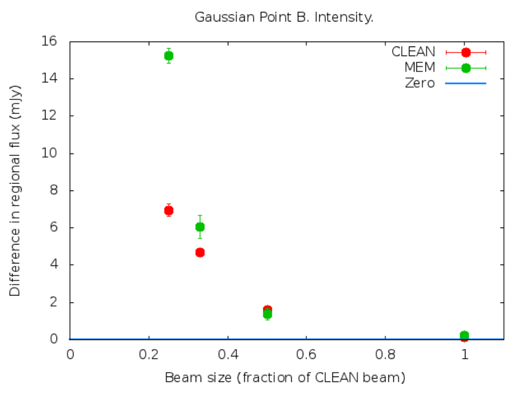
Figure 3.8 gives a similar result for measurements of the polarisation angle. In this case the MEM and CLEAN both do very well in measuring the polarisation angle of the core region (point A) – even at very high resolutions. However at points B and C as the emission becomes more extended the MEM starts to significantly outperform CLEAN, by up to 10 degrees at the highest resolution at point C. It is of note that, just as with fractional polarisation and total flux, the performance of the MEM changes very little as the size of the convolving beam is reduced. This is due to the fact that the form of entropy used in the MEM takes only Stokes I and the fractional polarisation into account (see Equation (2.13)). Thus the form of entropy used does not decide which of the Stokes Q or U components the detected polarised flux belongs to, only the χ^2 term in Equation (2.11) makes this decision. This means that while Stokes Q or U might individually be less accurate than CLEAN, when combining them (or in particular, taking the ratio of them), the MEM data are more likely to be similar to the real data.

3.3.2 Triple Gaussian

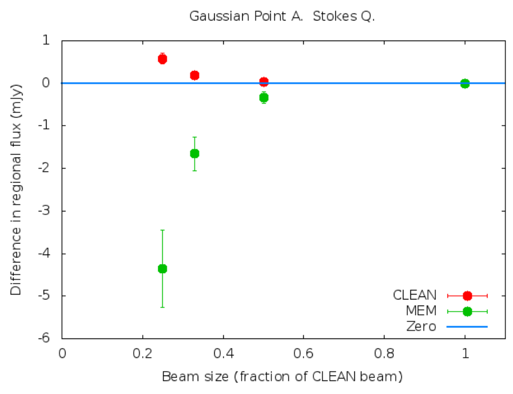
The second source used in the Monte Carlo simulations was a triple Gaussian source of large Gaussian components. The details of the source structure can



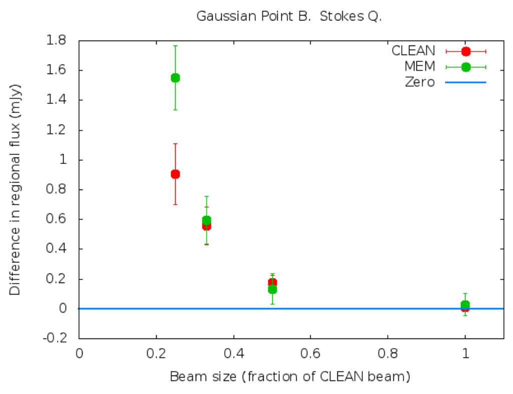
(a) Stokes I



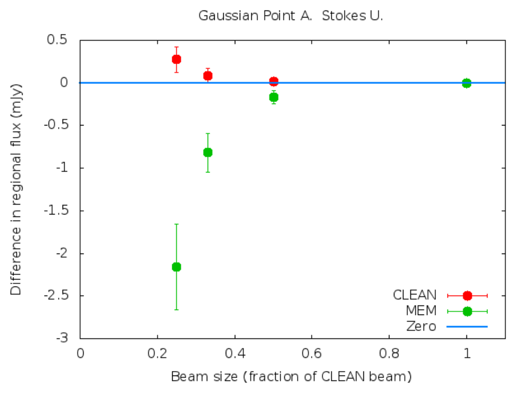
(b) Stokes I



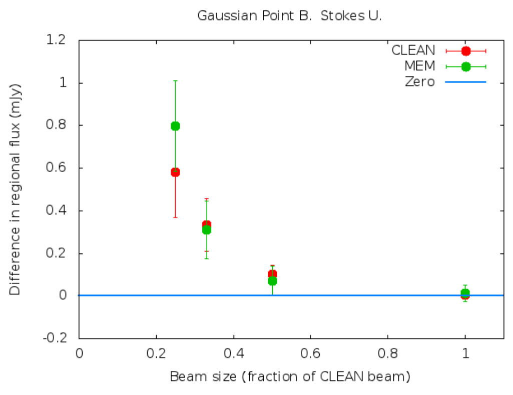
(c) Stokes Q



(d) Stokes Q

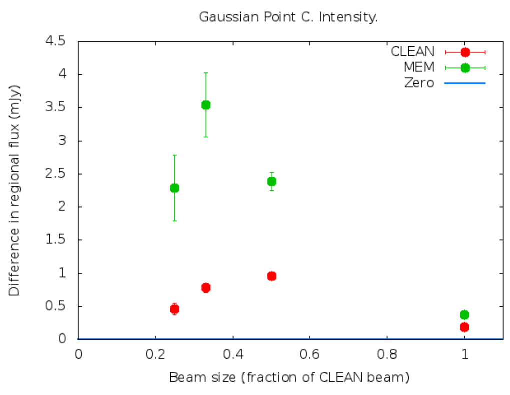


(e) Stokes U

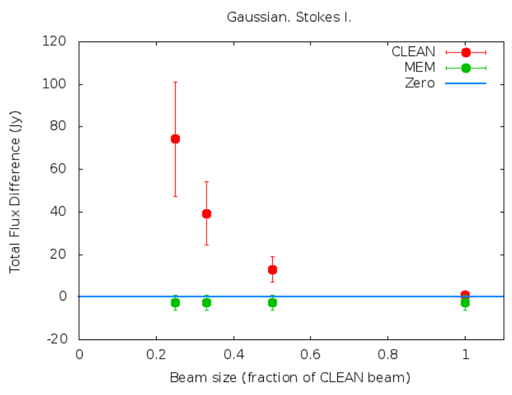


(f) Stokes U

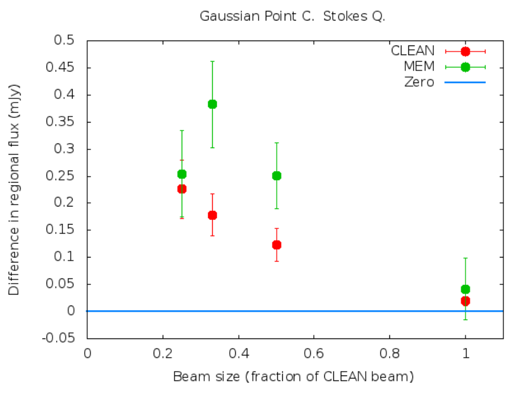
Figure 3.5: The distribution of the mean error in the flux detected with the MEM and CLEAN algorithms in imaging regions A and B in the single Gaussian of FWHM of 0.5 mas for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21×0.21 mas.



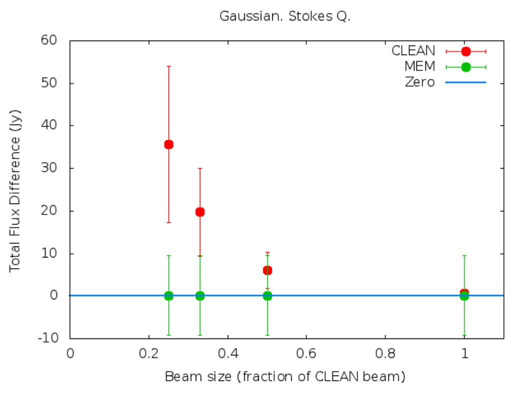
(a) Stokes I



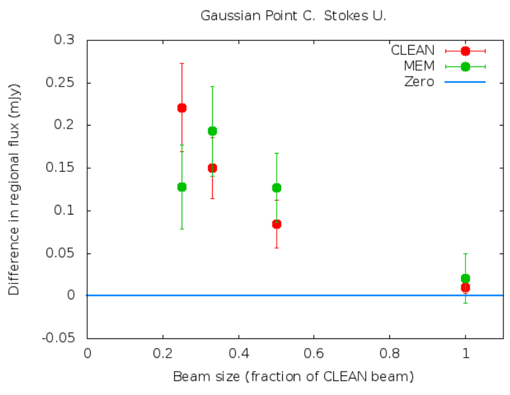
(b) Total Stokes I



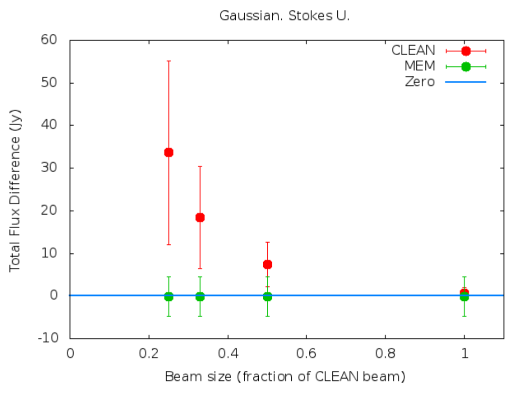
(c) Stokes Q



(d) Total Stokes Q

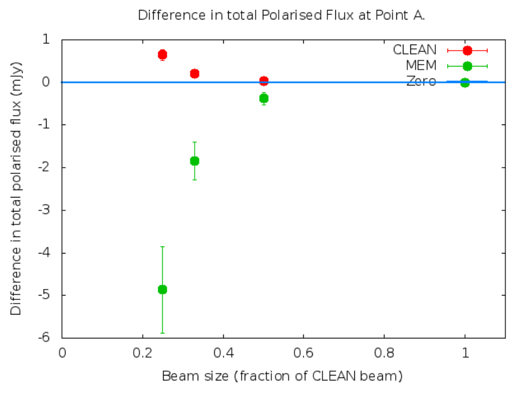


(e) Stokes U

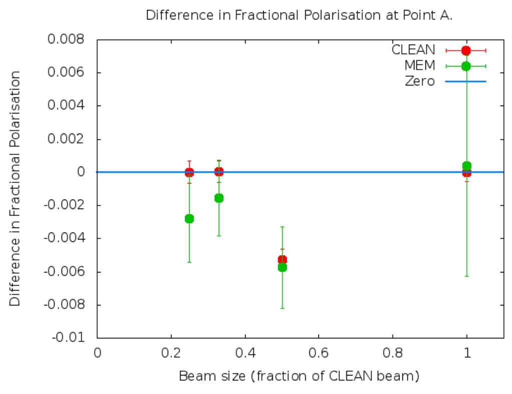


(f) Total Stokes U

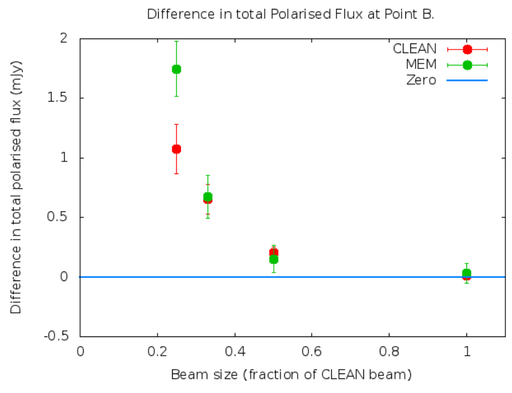
Figure 3.6: The distribution of the mean error in the flux detected with the MEM and CLEAN algorithms in imaging the flux at point C (left) and the total flux (right) in a single Gaussian of FWHM of 0.5 mas for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21×0.21 mas.



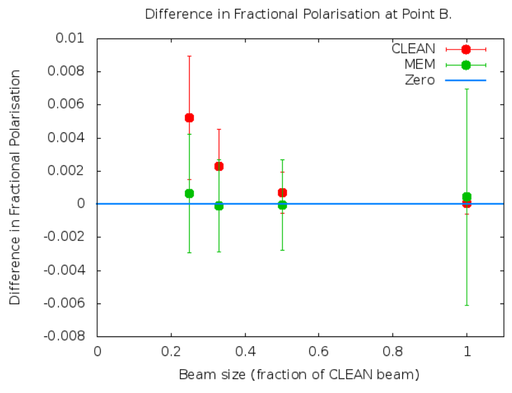
(a) Polarised Flux



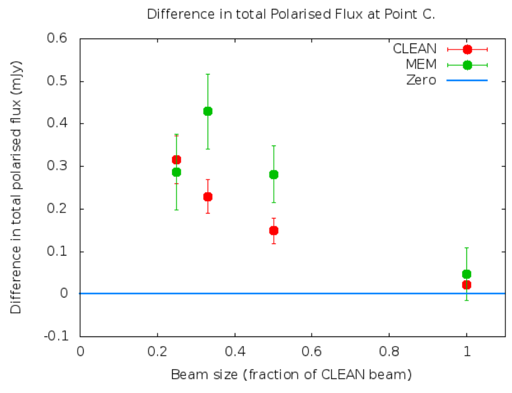
(b) Fractional Polarisation



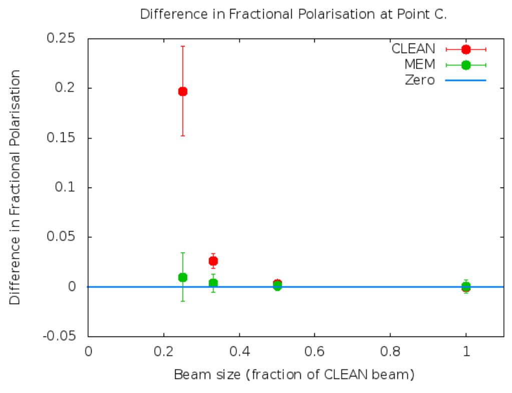
(c) Polarised Flux



(d) Fractional Polarisation

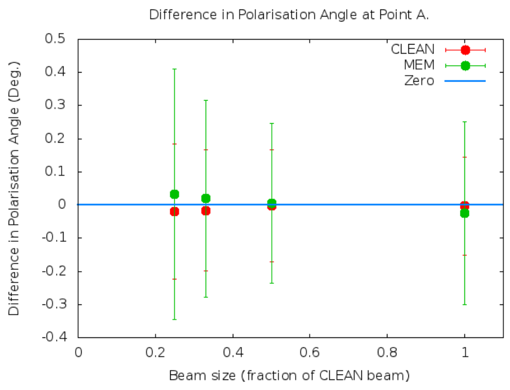


(e) Polarised Flux

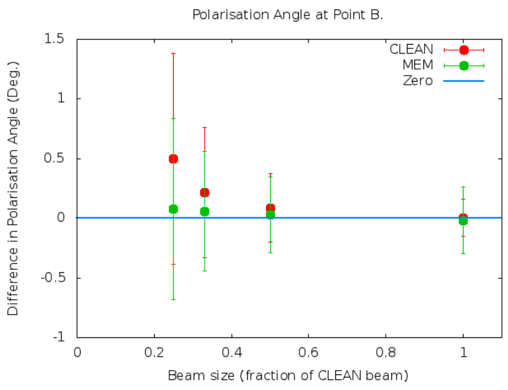


(f) Fractional Polarisation

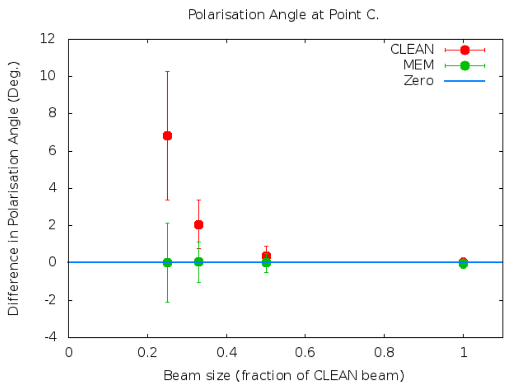
Figure 3.7: The distribution of the mean error in the total polarised flux and fractional polarisation detected with the MEM and CLEAN algorithms in imaging regions A, B and C of the single Gaussian of FWHM of 0.5 mas for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21 $\times$ 0.21 mas.



(a) Polarisation Angle



(b) Polarisation Angle

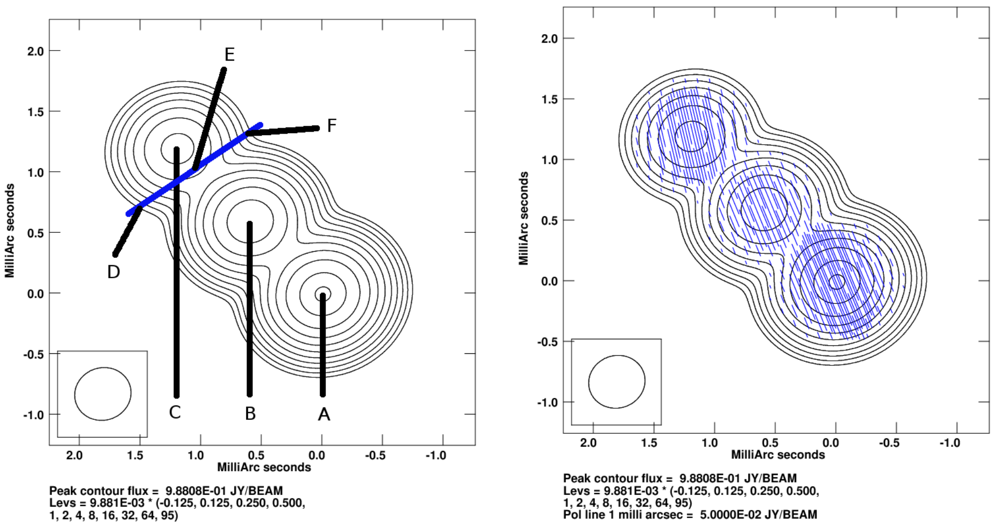


(c) Polarisation Angle

Figure 3.8: The distribution of the mean error in the polarisation angle detected with the MEM and CLEAN algorithms in imaging regions A, B and C of the single Gaussian of FWHM of 0.5 mas for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21 $\times$ 0.21 mas.

Table 3.2: Information about the size and separation of the three Gaussian components making up the triple Gaussian source.

Component	FWHM (mas)	I (Jy)	Q (Jy)	U (Jy)	m	DCORE (mas)
1	0.05	1	0.035	0.035	0.05	0
2	0.1	0.5	0.018	0.018	0.05	0.85
3	0.4	0.1	0.031	0.016	0.35	1.70



(a) Data Points (b) Polarisation Map

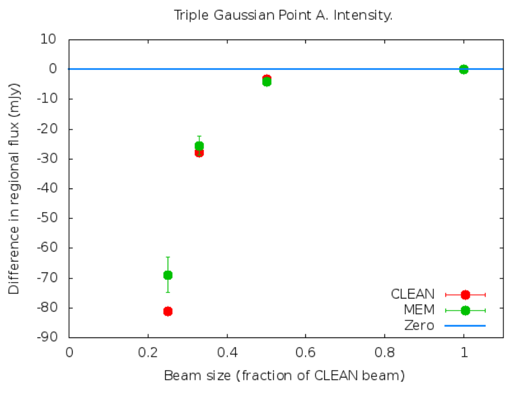
Figure 3.9: The model triple Gaussian source convolved with 0.25 of the CLEAN beam. Points A, B and C indicate the peak regions sampled with 0.21 × 0.21 mas boxes in the Monte Carlo experiment. Points D, E and F indicate regions sampled for a transverse slice across the jet (indicated by the blue line). The blue dashes indicate the polarisation angle.

Table 3.3: The beams used in the Monte Carlo simulations of the Triple Gaussian source. The CLEAN beam is a Gaussian fit to the FWHM of the "dirty" beam.

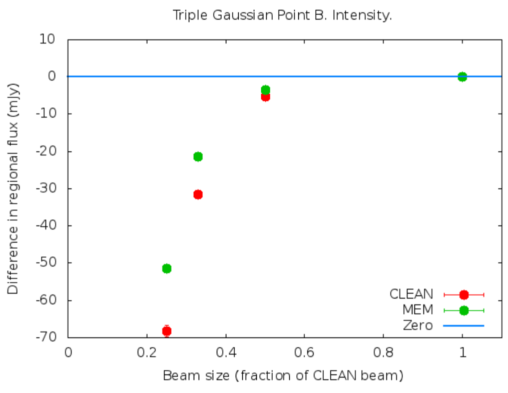
Frac. of CLEAN beam	Major Axis (mas)	Minor Axis (mas)	Position Angle (deg.)
1	1.91	1.74	-71.68
$\frac{1}{2}$	0.955	0.87	-71.68
$\frac{1}{3}$	0.637	0.58	-71.68
$\frac{1}{4}$	0.4775	0.435	-71.68

be seen in Table 3.2 and Figure 3.9. Note the more complicated polarisation structure in Stokes Q and U than was used for the single Gaussian. The same tests were performed on the triple Gaussian source as were performed on the single Gaussian source in the previous section. However in addition to considering the peak region, three additional regions representative of the areas which might be sampled in a transverse slice across the jet were also tested (see Figure 3.9).

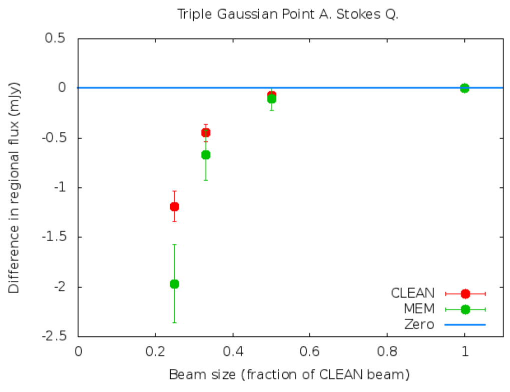
Figures 3.10 and 3.11 (left) give the results of the regional flux test for each algorithm at points A, B and C. These points are the peaks of the three Gaussian components of the source, therefore the CLEAN algorithm is expected to perform well, while the MEM may be at a disadvantage. Both algorithms again do very well at CLEAN beam and half CLEAN beam resolutions, however at higher



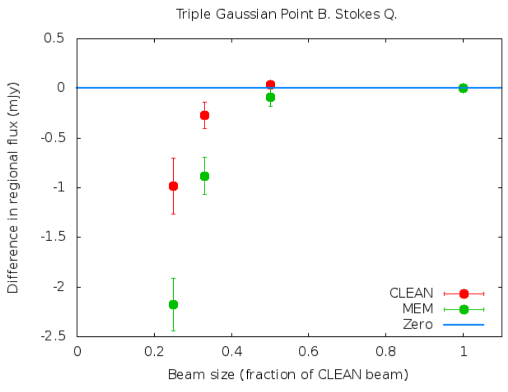
(a) Stokes I



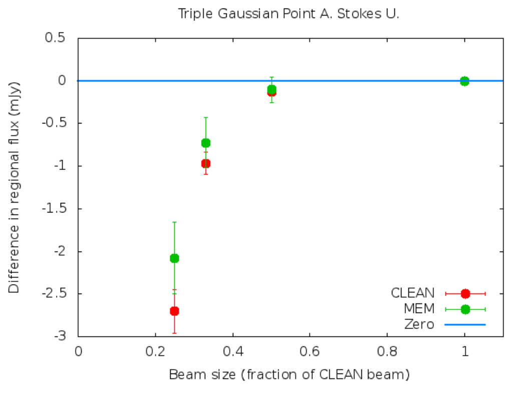
(b) Stokes I



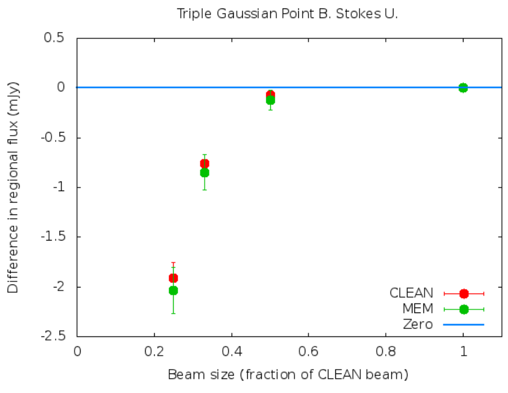
(c) Stokes Q



(d) Stokes Q

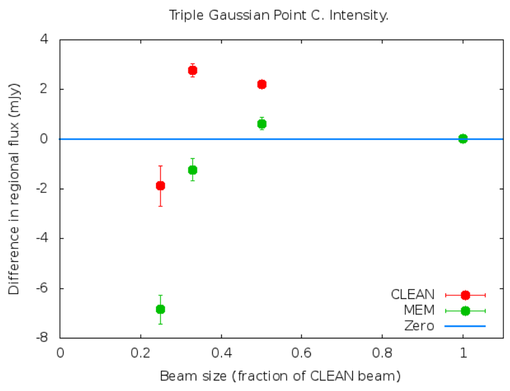


(e) Stokes U

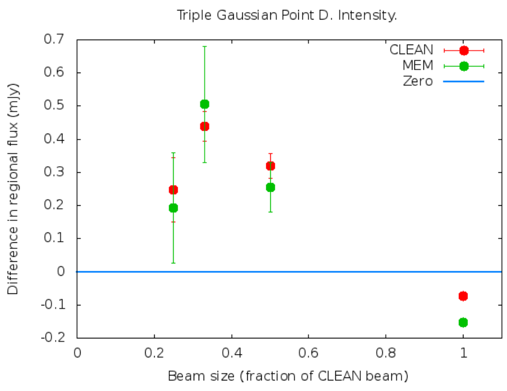


(f) Stokes U

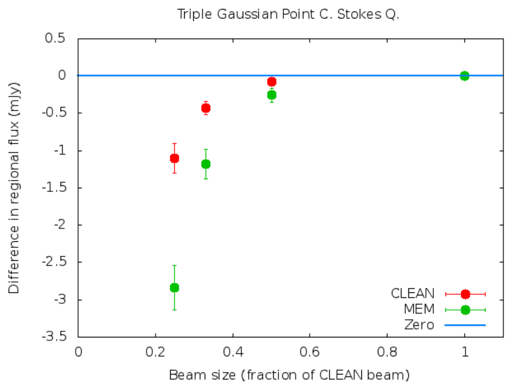
Figure 3.10: The distribution of the mean error in the flux detected with the MEM and CLEAN algorithms in imaging regions A and B of the triple Gaussian source for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21×0.21 mas.



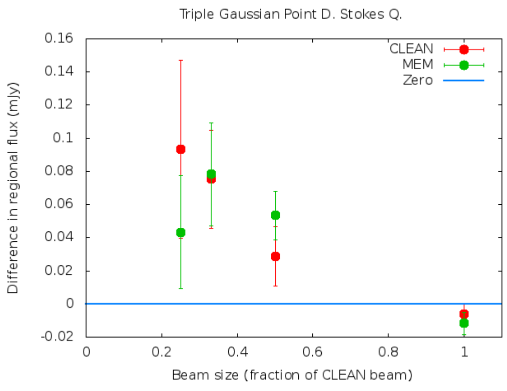
(a) Stokes I



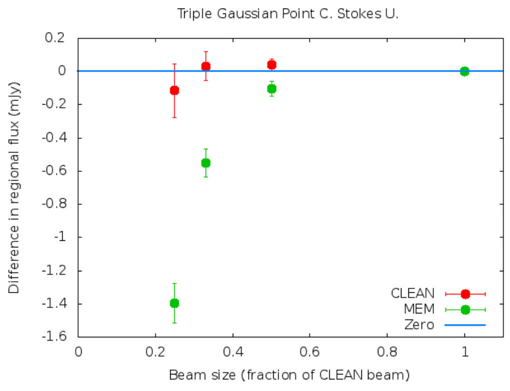
(b) Stokes I



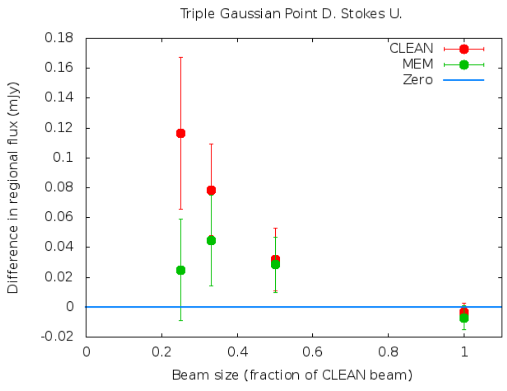
(c) Stokes Q



(d) Stokes Q

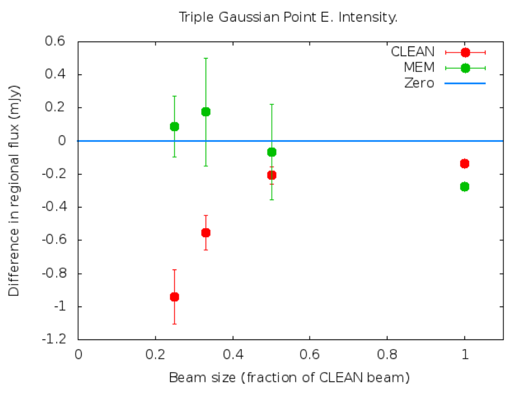


(e) Stokes U

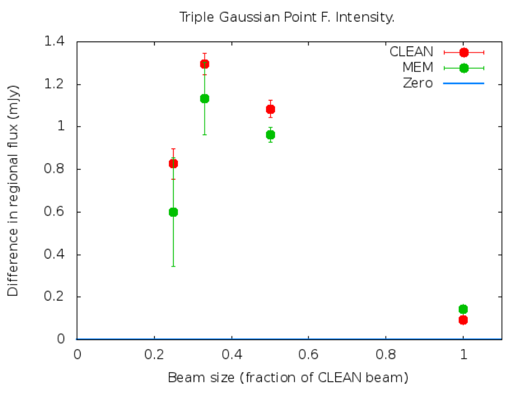


(f) Stokes U

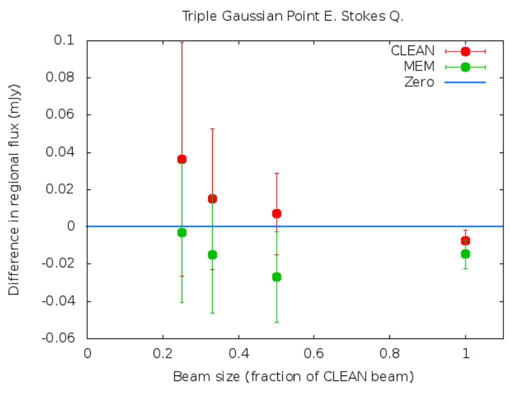
Figure 3.11: The distribution of the mean error in the flux detected with the MEM and CLEAN algorithms in imaging regions C and D of the triple Gaussian source for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21 × 0.21 mas.



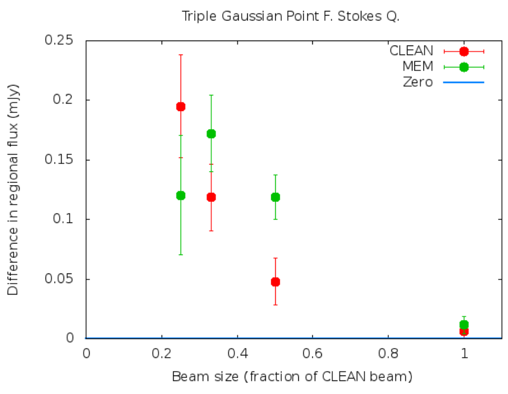
(a) Stokes I



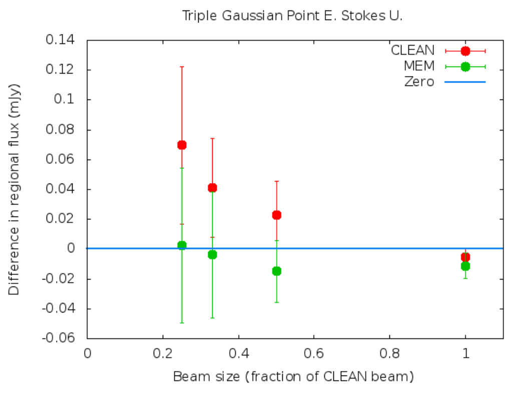
(b) Stokes I



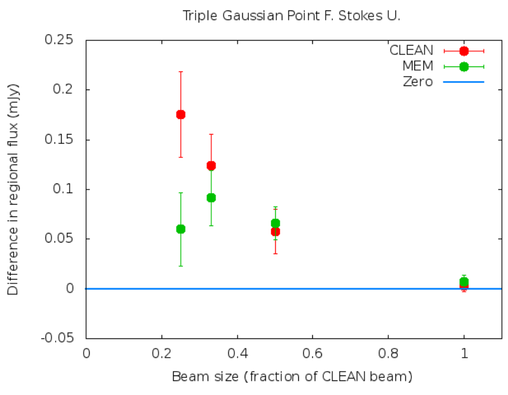
(c) Stokes Q



(d) Stokes Q

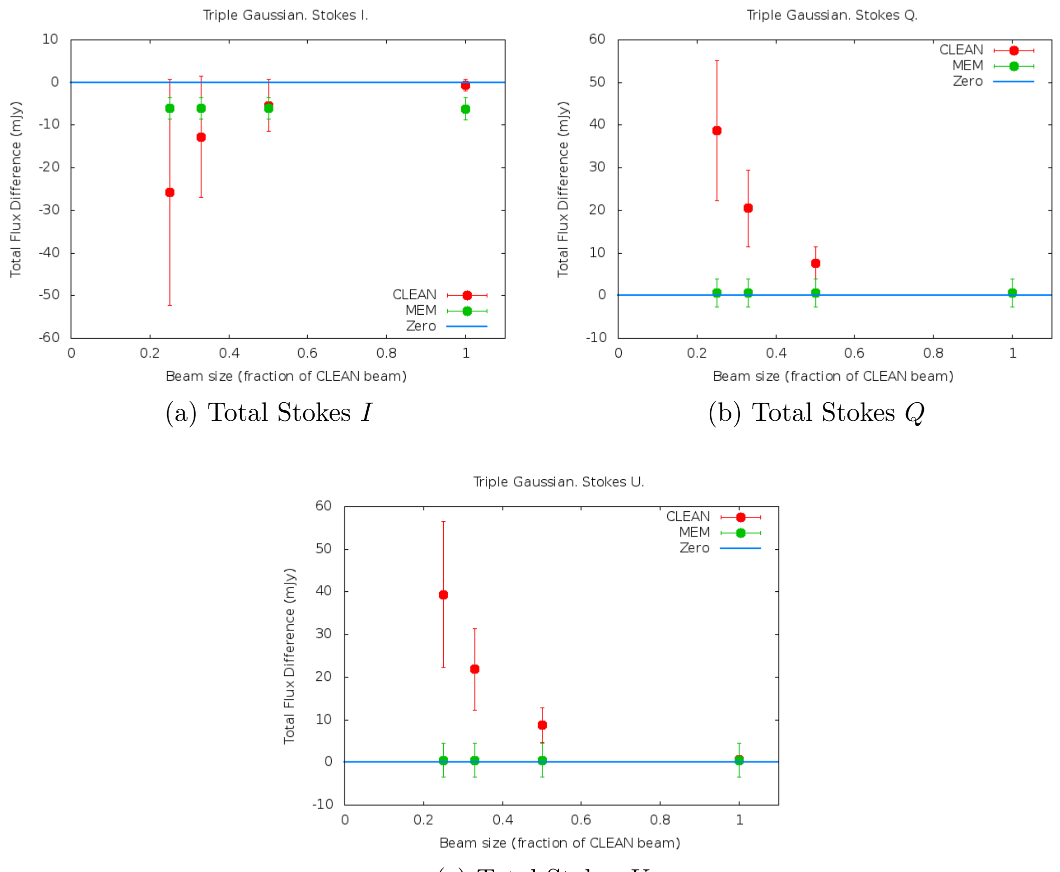


(e) Stokes U



(f) Stokes U

Figure 3.12: The distribution of the mean error in the flux detected with the MEM and CLEAN algorithms in imaging regions E and F of the triple Gaussian source for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21×0.21 mas.



(a) Total Stokes I

(b) Total Stokes Q

(c) Total Stokes U

Figure 3.13: The distribution of the mean error in the total flux detected with the MEM and CLEAN algorithms in the triple Gaussian source for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21 $\times$ 0.21 mas.

resolutions they begin to perform poorly. It is clear that the MEM is much more competitive with the CLEAN algorithm at points A, B and C in the triple Gaussian than it was at point A in the single Gaussian. This is possibly due to CLEAN having trouble imaging the more complicated structure of the triple Gaussian, whereas the MEM, having the advantage of "super-resolution" does not suffer any increased difficulty. Thus while CLEAN was a better choice for imaging the core of the Gaussian in the previous section, the MEM does a better job imaging point A in this section. This is not true for all points and for all Stokes parameters however and neither algorithm excels at imaging regions A, B or C in figures 3.10 to 3.11 at resolutions of less than half a CLEAN beam.

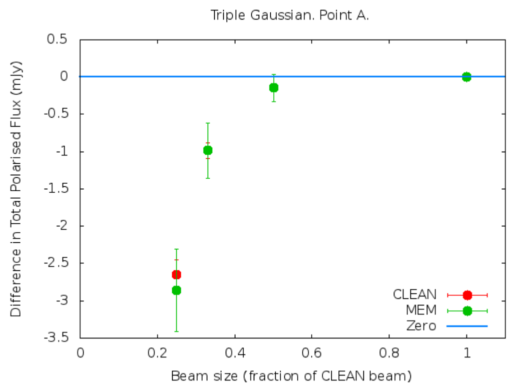
Figures 3.11 (right) and 3.12 show the performance of the two algorithms at measuring the regional flux in regions D, E and F as indicated in Figure 3.9. These points were select to represent likely points of data to take when making a transverse slice across the jet direction, a common operation in testing for gradients in the Faraday rotation measure across the jet. As these points are relatively far from the peaks of the algorithm the continuous model of the source flux employed by the MEM allows it to do a better job than across the majority of data points.

Figure 3.13 shows again that the MEM outperforms CLEAN at measuring the total flux of the source across all three Stokes parameters. Although the CLEAN algorithm does a better job at measuring the Stokes I intensity the normal CLEAN beam resolution, its performance quickly falls off with decreasing beam size while that of the MEM does not vary appreciably. It is also of note that the MEM has a much smaller spread of values, while CLEAN fluctuates wildly.

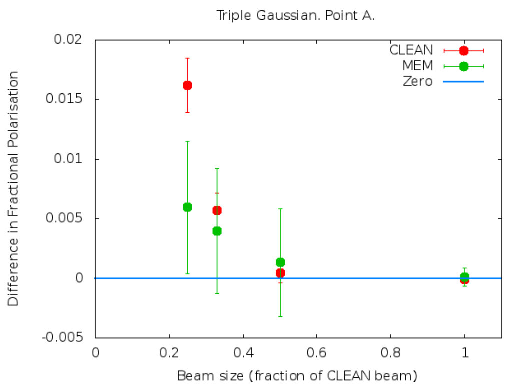
Figures 3.14 and 3.15 show the results for the regional total and fractional polarised flux at all six points. Consistent with the results for the single Gaussian, the MEM consistently outperforms CLEAN at higher resolutions. A similar jump in performance of the MEM is seen in going from total polarised flux to fractional polarised flux. Again, the performance of the MEM appears to vary little as the resolution is increased, while that of CLEAN suffers greatly. Regions A, B and C appear more difficult to image with the MEM in total polarisation than regions D, E and F.

The ability of both algorithms to measure the polarisation angle at all points can be seen in Figure 3.16. The MEM achieves an extremely low uncertainty at all points (notably better at regions A, B and C than the equivalent single Gaussian regions in Figure 3.8). The exceptional performance of the MEM at measuring the polarisation angle in regions D, E and F is of particular interest as this high performance even at very high resolutions make the MEM an excellent candidate for imaging multi-wavelength polarisation data to investigate the presence of Faraday rotation measure gradients across the jet.

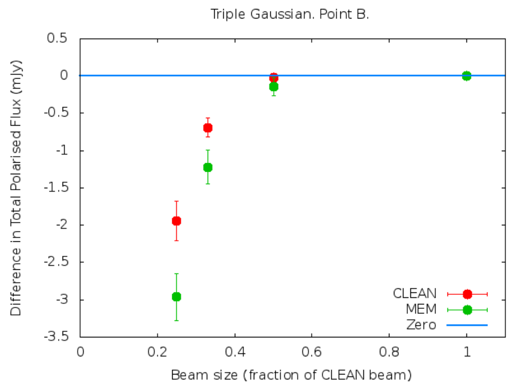
Another interesting result of these Monte Carlo simulations is the relatively tiny error in the polarisation angle as measured using the MEM, even though the error in the individual Stokes Q and U parameters was much higher. Clearly the error in polarisation angle is not a simple propagation of random error, and is more related to systematic effects. It would be more appropriate to estimate the error in the polarisation angle independently from the errors in Stokes Q and U. This estimate would vary depending on the size of the restoring beam, but from Figure 3.16 a value of around 3 degrees may be appropriate in many cases.



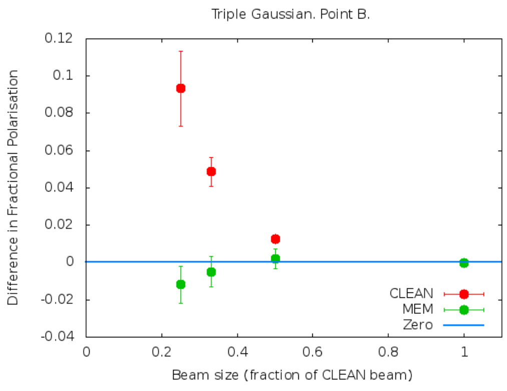
(a) Polarised Flux



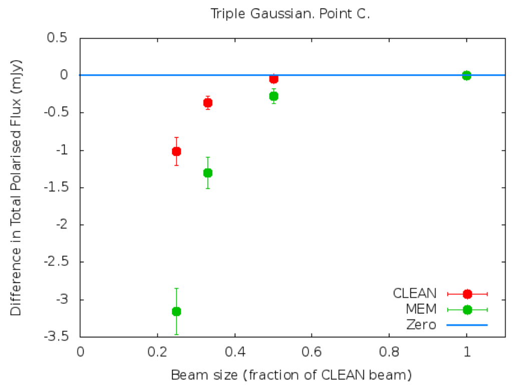
(b) Fractional Polarisation



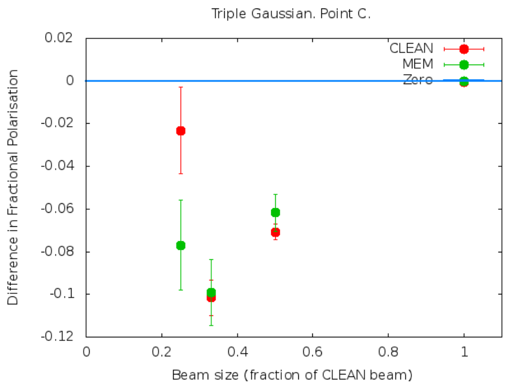
(c) Polarised Flux



(d) Fractional Polarisation

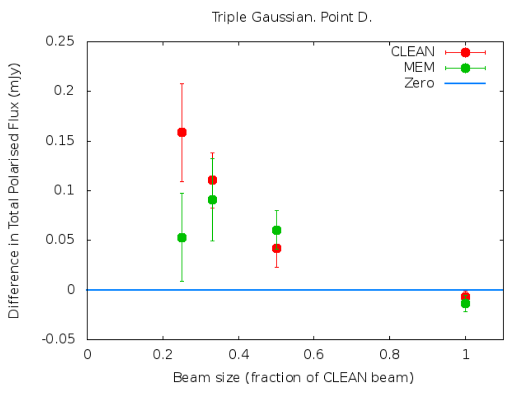


(e) Polarised Flux

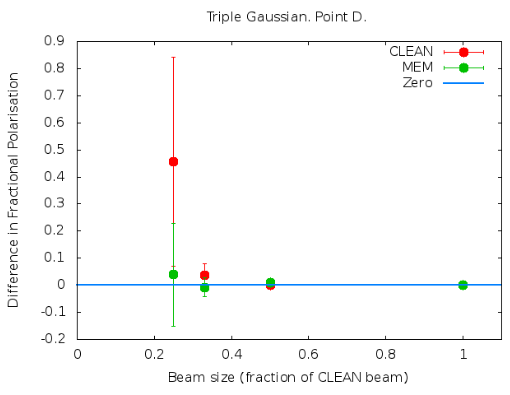


(f) Fractional Polarisation

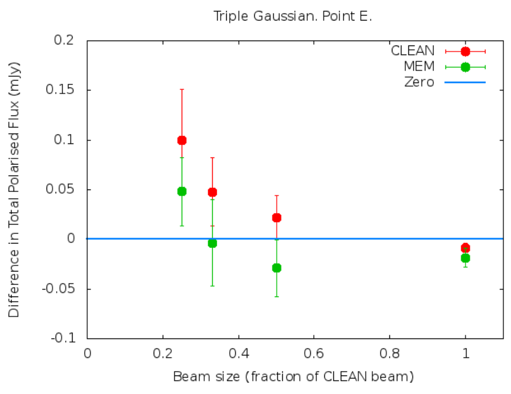
Figure 3.14: The distribution of the mean error in the total and fractional polarisation detected with the MEM and CLEAN algorithms in imaging regions A, B and C of the triple Gaussian source for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21 × 0.21 mas.



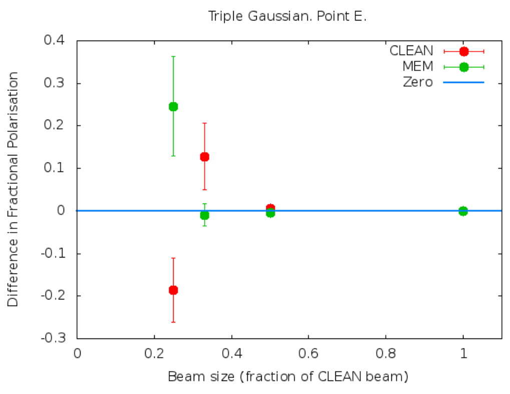
(a) Polarised Flux



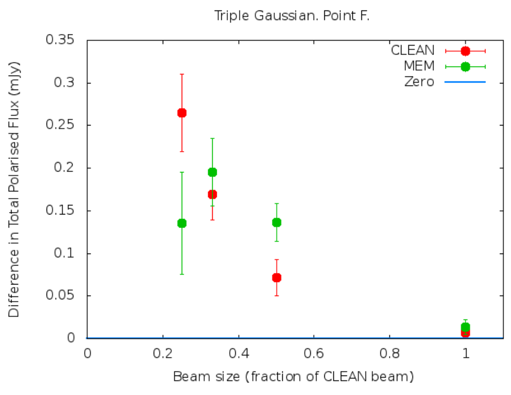
(b) Fractional Polarisation



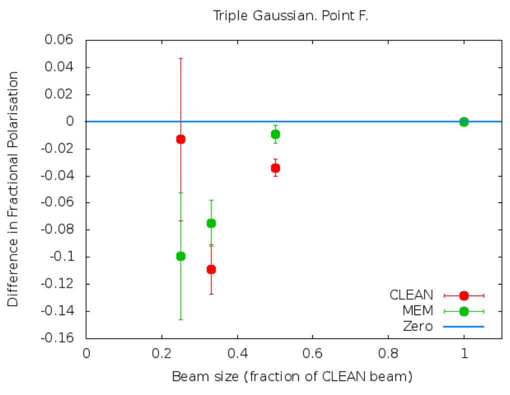
(c) Polarised Flux



(d) Fractional Polarisation

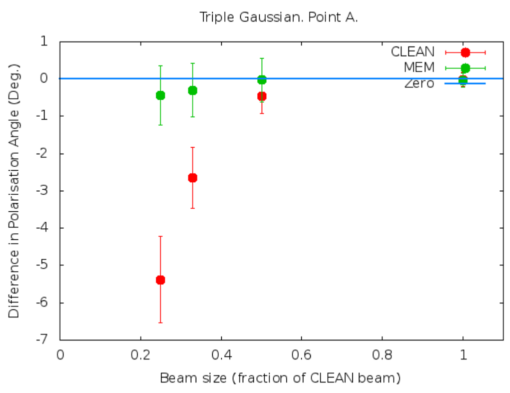


(e) Polarised Flux

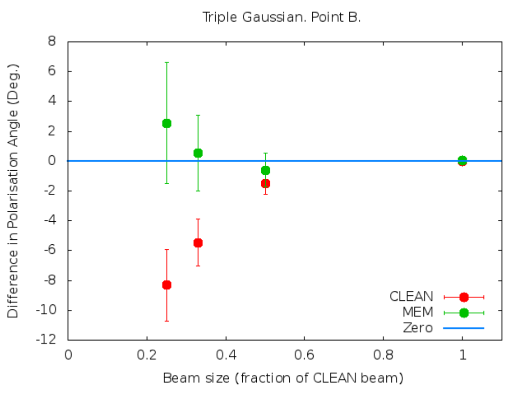


(f) Fractional Polarisation

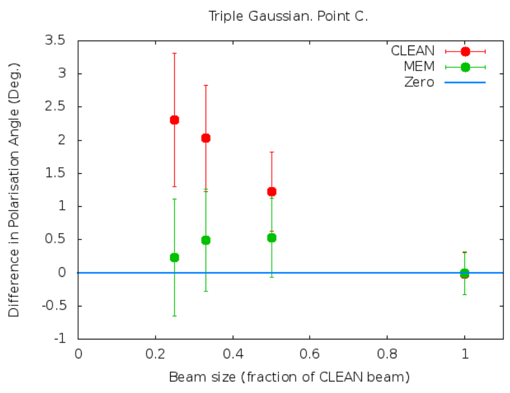
Figure 3.15: The distribution of the mean error in the total and fractional polarisation detected with the MEM and CLEAN algorithms in imaging regions D, E and F of the triple Gaussian source for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21×0.21 mas.



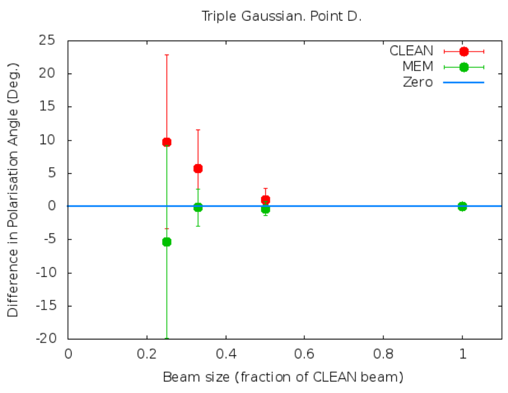
(a) Region A.



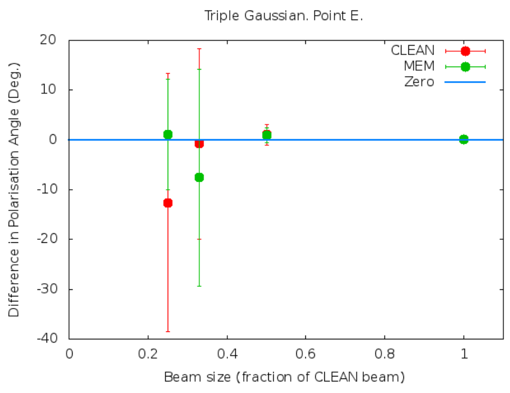
(b) Region B.



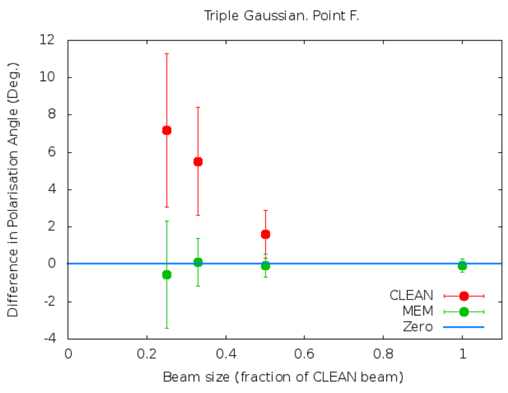
(c) Region C.



(d) Region D.



(e) Region E.



(f) Region F.

Figure 3.16: The distribution of the mean error in the polarisation angle detected with the MEM and CLEAN algorithms in imaging regions A, B, C, D, E and F of the triple Gaussian source for various convolution sizes. The error bars indicate the standard deviation of the distribution resulting from the Monte Carlo simulations. The region size was 0.21 $\times$ 0.21 mas.

3.4 Conclusions

A new C++ code has been written to implement a MEM deconvolution of VLBI intensity (I) and polarisation (Q and U) data. This code was based on the earlier work of Holdaway & Wardle (1990) and Sault (1999), but has improved on their algorithms in a number of ways. The code has been tested and its performance evaluated and compared with the equivalent performance of a standard CLEAN deconvolution using Monte Carlo simulations of the deconvolution of images for a compact single Gaussian model source and a more extended triple Gaussian model source, chosen to have properties qualitatively similar to those of observed AGN on VLBI scales.

On the basis of the Monte Carlo simulations detailed in Section 3.3, the PMEM code developed successfully uses the Maximum Entropy Method to deconvolve VLBI polarisation data. The MEM is very successful at measuring the total flux when given an accurate estimate, but further testing is required to see if this changes when less accurate estimates are given. Both the MEM and CLEAN algorithms produce reliable values for the flux in a small region down to about half of the CLEAN beam, at which point the performance of CLEAN begins to drop away quickly, whereas the MEM is often more reliable.

The MEM exhibits a particular strength in regions of diffuse emission, while CLEAN is often more suitable in more “pointy” regions. This is due to the δ function model of the source used by CLEAN, which assumes the source can be modelled as a series of point sources - which can be true if the source appears as a point source to the observing array. In cases where a source, or a part of a source, appears as a point source to an array – i.e. a single (stacked) CLEAN component accounts for the vast majority of the flux in a region equivalent to the size of the CLEAN beam, CLEAN is likely to outperform the MEM due to its innate assumptions about the structure of the source. Conversely, in regions which cannot be well described as point sources of emission the MEM is likely to outperform CLEAN. This suggests the feasibility of an approach to deconvolution using both CLEAN and MEM, where the CLEAN algorithm could be used to search for “pointy” regions and describe them with a set of CLEAN components with the MEM then being used to deconvolve the remaining map of diffuse and smooth emission. The CLEAN components corresponding to regions well described with a point source model could then be convolved with the same beam as the MEM model and combined with the convolved MEM model and residuals to give a map which should exhibit the strengths of both algorithms. Monte Carlo testing of such an approach could be conducted in the same manner as those in this chapter to test its performance and test for any unforeseen systematic issues.

The MEM has a distinct advantage over CLEAN in imaging the polarisation angle at high resolution, though the exact accuracy can vary (some of the standard deviations in Figure 3.16 are very large). This will make MEM a valuable tool in detecting gradients in Faraday rotation measure that are perhaps hinted at in normal CLEAN images, but need a higher resolution to be firmly detected. The detection of such gradients will then allow the magnetic field present in the jet to be studied and any evidence of a helical field component examined.

The fact that both algorithms perform very well at resolutions down to $\frac{1}{2}$ of the CLEAN beam has implications for the choice of beam used in multi-wavelength studies such as the creation of Faraday rotation measure maps. It is standard practice to restore all maps used with the restoring beam corresponding to the lowest frequency, thereby throwing away information present at higher resolutions in higher frequency maps – even though the natural CLEAN beam for those frequencies might include such information. The results of the Monte Carlo simulations carried out in this chapter strongly suggest that, even for the CLEAN algorithm, a moderate super-resolution of up to half of the CLEAN beam corresponding to the lowest frequency should have little effect on the multi-wavelength-derived map, especially considering the degree of super-resolution will be lower for the other maps. If the MEM is used then higher quality results may be expected, or even higher resolution maps created.

Chapter 4

Application of PMEM to Real VLBI Polarisation Data

Chapter 2 discussed the Maximum Entropy Method and its implementation to deconvolve polarised data using the Cornwell-Evans algorithm, with modifications by Holdaway and Wardle (1990) and Sault (1990). Chapter 3 then discussed the design and realisation of such a method suitable for deconvolving polarised VLBI data and detailed the results of Monte Carlo simulations on the resulting software. This Chapter deals with the application of the new PMEM software to real VLBI polarisation data. Intensity and polarisation maps for various sources are presented, along with the first Faraday rotation measure maps made using the MEM at VLBI scales.

4.1 Markarian 501

The final MEM code was applied to polarisation observations of Markarian 501 at 2cm, 4cm and 6cm taken by Pushkarev et al. (2005). Markarian 501, or J1653+3945, is observed as a BL Lac object (see Section 1.5.4) with a redshift of $z = 0.0337$, corresponding to a distance of 146 Mpc, or 4.5×10^{24} m (data from the MOJAVE project, Lister et al. 2009). This source has a pronounced jet with strong polarisation and a lot of bending. Evidence for the presence of a helical magnetic field threading the jet of the source has been reported by Pushkarev et al. (2005), who pointed out that the spine–sheath polarisation structure observed in the VLBI jet was consistent with a helical magnetic field, and Murphy, Cawthorne and Gabuzda (2013), who successfully fit transverse slices across this spine–sheath polarisation structure using a simple helical field model. In addition, Faraday rotation measure gradients across the jet have been reported by Gabuzda et al. (2004), based on 2cm, 3.6cm and 6cm VLBA polarisation data, and Croke et al. (2010), based on 3.6cm, 6cm, 13cm and 18cm VLBA polarisation data.

All the initial calibration and imaging of these data (using a CLEAN deconvolution) had already been carried out carried out by Pushkarev et al.(2005). The MEM deconvolution of the data was carried out with PMEM using the parameters listed in Table 4.1. The estimates of the total Stokes I, Q and U fluxes were obtained by estimating the zero spacing flux in a plot of the corresponding visibilities against wavelength (see Figure 4.1 for an example). Markarian 501 was found to image well using the default flat bias map and with the fluxes approximately conserved. Images were made corresponding to a variety of estimated final RMS noises, however the best results corresponded to the assumption of 0 final RMS noise in each case. Although this is an unreasonable expectation, in the particular dataset under consideration it resulted in the best convergence between the MEM model and the data and gave realistic final RMS noise values.

Samples of the imaging results for the 6 cm dataset can be seen in Figures 4.2 to 4.4. Figure 4.2 shows the results for Stokes I. The plot of the intrinsic MEM model map shows the intrinsic Stokes I model of Markarian 501 that has maximum information entropy while still agreeing with the data to within a reasonable expectation. As much of the structure shown is at scales smaller than the limit set by the Fourier Sampling Theorem it does not represent real data, rather a conservative model with maximum entropy given the constraints of the data. However, the Monte Carlo simulations detailed in Chapter 3 demonstrate that the MEM does model the source realistically at levels below that accounted for by the Sampling Theorem. This in contrast to CLEAN, where the choice of modelling the intrinsic emission as a series of δ functions can be unrealistic for many sources. It is therefore possible that some useful data can be extracted from the intrinsic MEM model.

The convolved map shown in Figure 4.2 is the result of convolving the intrinsic model with the normal CLEAN beam. The residual map is the difference between the dirty map and the convolution of the intrinsic MEM map with the *dirty* beam. To avoid Fourier related artefacting near the edges of the map degrading the quality of the final MEM image 200 pixels were clipped from the edges of the maps. The image was made sufficiently large to ensure any emission from

Table 4.1: MEM parameters used to image Markarian 501. In all cases the restoring beam used was the CLEAN beam, the maximum number of iterations was $10,000$, a flat bias map was used and flux was conserved with parameter 2. It was found that an estimated final RMS noise of 0 gave the best deconvolution for all Stokes parameters. $N_{exclude}$ was set to 200 pixels in all cases.

	6 cm	4 cm	2 cm
Stokes I Flux	1.05 Jy	0.95 Jy	0.72 Jy
Stokes Q Flux	0.035 Jy	0.03 Jy	0.05 Jy
Stokes U Flux	0.04 Jy	0.03 Jy	0.05 Jy
Δ_{step}	1	1	1
q_{factor}	0.009	0.05	0.04
w_p	2	2	2

the source was not present in clipped regions. There is clearly an increase in the residuals in the on-source area, however as most of the increased residuals are within 3 times the background noise the model can be considered to be well converged. This effect is most pronounced at the core of the source, suggesting that the MEM has difficulty imaging relatively sharply peaked regions. Further evidence of this effect can be seen by analysing the final MEM maps of the source.

The final panel in the figure shows the result of adding the residual map to the convolved model map (note that if the restoring beam is not the CLEAN beam the residuals should be rescaled into the same Jy/Beam units). This final MEM map includes any remaining background noise and source structure that has not been well described by the MEM model. A comparison of the peak Stokes I value as imaged with the MEM and CLEAN can be seen in Table 4.2. It is clear that the CLEAN algorithm systematically measures a higher peak flux than the MEM. This can be explained by the way in which a δ function based model of the source tends to concentrate flux in a small region, whereas the continuous model employed by the MEM prefers to spread it over neighbouring pixels. The analysis in Chapter 3 suggests that, as long as a single pixel value is not used to determine the peak (instead averaging over a small region), both algorithms agree well at the peak region at the full CLEAN beam, while falling off in accuracy at higher resolutions. CLEAN appeared to fare better with a single Gaussian source, while the MEM was better at imaging the peak of the more spread out triple Gaussian source. An imaging approach using both the CLEAN and MEM algorithms as described in Section 3.4 may yield improved results, but would require Monte Carlo testing to determine its accuracy.

Figures 4.3 and 4.4 show the corresponding Stokes Q and U maps of Markarian

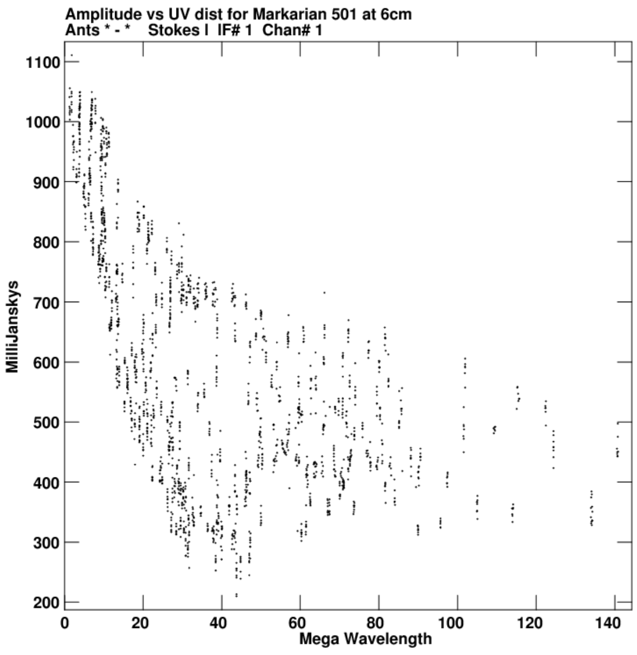


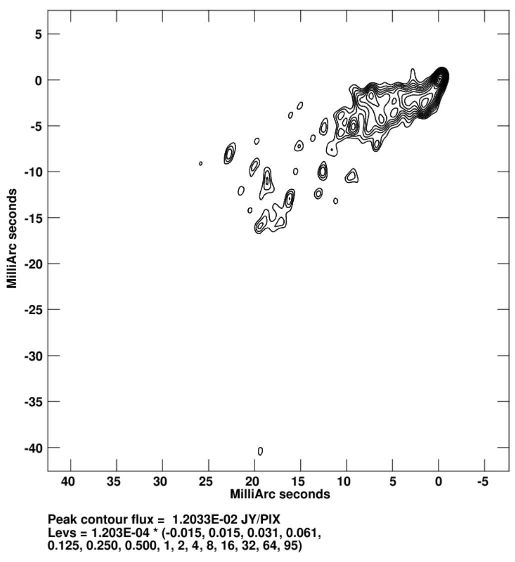
Figure 4.1: Stokes I visibility versus baseline length plot for Markarian 501 at 4.971 GHz (6 cm). The total Stokes I intensity can be estimated by looking at the approximate intensity for a baseline of zero length (thus not resolving any of the source).

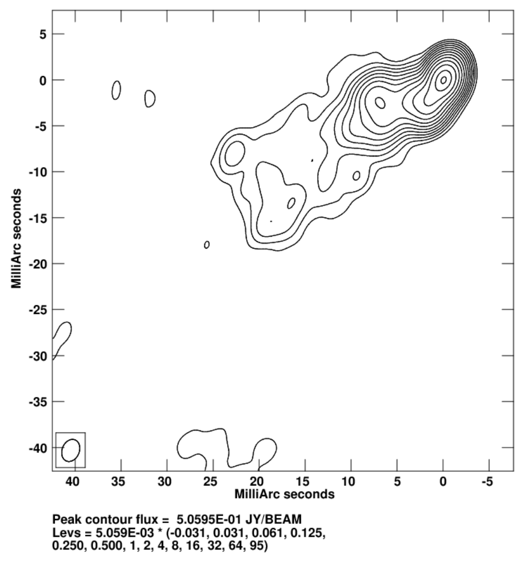
Table 4.2: Peak fluxes achieved with the MEM and CLEAN in Markarian 501 at 6 cm.

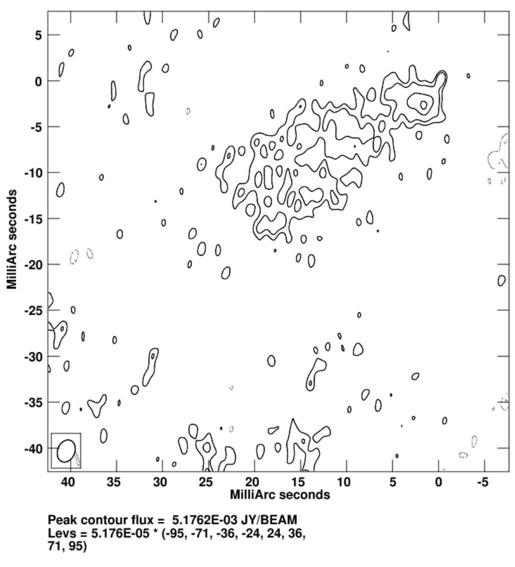
	CB (Jy/Beam)	$\frac{1}{2}$ CB (Jy/Beam)	$\frac{1}{3}$ CB (Jy/Beam)	$\frac{1}{3}$ CB (Jy/Beam)
CLEAN	0.520	0.463	0.436	0.414
MEM	0.509	0.397	0.303	0.223

Table 4.3: The various beams used in the imaging of Markarian 501.

Frac. of CLEAN beam	Major Axis (mas)	Minor Axis (mas)	Position Angle (deg.)
1	2.49	1.88	−22.36
$\frac{1}{2}$	1.245	0.94	−22.36
$\frac{1}{3}$	0.83	0.63	−22.36
$\frac{1}{4}$	0.6225	0.47	−22.36


MilliArc seconds
MilliArc seconds
Peak contour flux = 1.2033E-02 JY/PIX
Levs = 1.203E-04 * (-0.015, 0.015, 0.031, 0.061, 0.125, 0.250, 0.500, 1, 2, 4, 8, 16, 32, 64, 95)
(a) Model Map.


MilliArc seconds
MilliArc seconds
Peak contour flux = 5.0595E-01 JY/BEAM
Levs = 5.059E-03 * (-0.031, 0.031, 0.061, 0.125, 0.250, 0.500, 1, 2, 4, 8, 16, 32, 64, 95)
(b) Convolved Model Map.


MilliArc seconds
MilliArc seconds
Peak contour flux = 5.1762E-03 JY/BEAM
Levs = 5.176E-05 * (-95, -71, -36, -24, 24, 36, 71, 95)
(c) Residual Map.

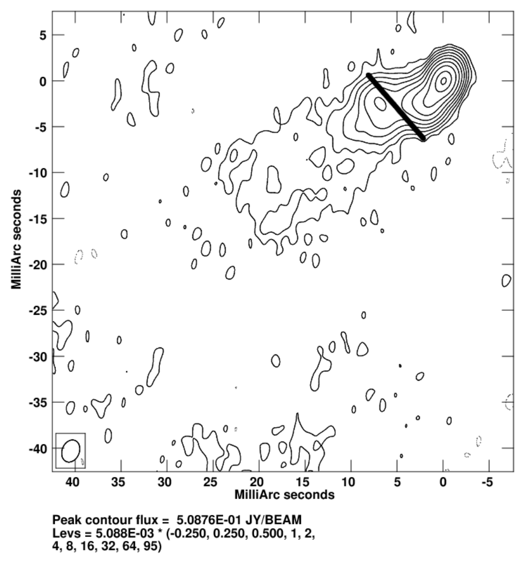

MilliArc seconds
MilliArc seconds
Peak contour flux = 5.0876E-01 JY/BEAM
Levs = 5.088E-03 * (-0.250, 0.250, 0.500, 1, 2, 4, 8, 16, 32, 64, 95)
(d) Final MEM Map.

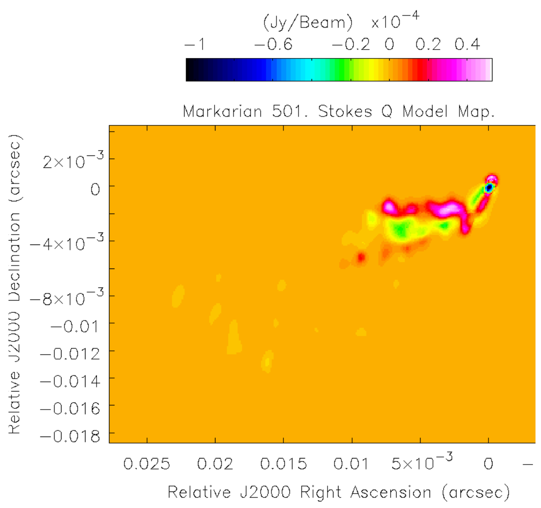
Figure 4.2: Stokes I MEM maps of Markarian 501 at 4.971 GHz (6 cm). The Convolved maps shown have been convolved with the CLEAN beam for the observation. The line in the final map indicates the position of the slices taken across CLEAN maps and presented in Figure 6.12.

501 at 6 cm. The Stokes Q map in particular appears to have achieved excellent convergence, while some structure in the core region of the map is visible in the residual map for Stokes U. However, as this structure is on a scale close to the background noise, the Stokes U map can be said to also be well converged. An important factor in achieving this high degree of convergence was the choice made to assign a higher weight to the polarisation residuals than the Stokes I residuals during the MEM mapping. This resulted in a slightly poorer Stokes I map, but made much better Stokes Q and U maps – a good compromise when primarily investigating the polarisation features of a source. The resulting final MEM maps are displayed in the bottom right of each figure.

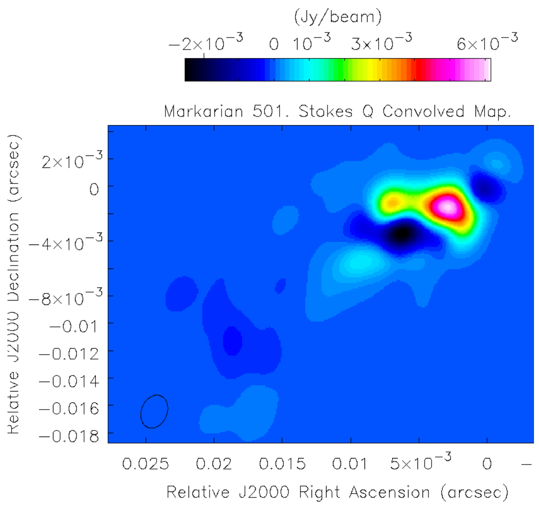
Figures 4.10 to 4.13 show CLEAN and MEM percentage polarisation maps of Markarian 501 at 6 cm convolved with the selection of beams listed in Table 4.3. The cut off point for the fractional polarisation displayed was determined by clipping all polarised flux lower than the maximum polarised flux in region well off the source. The same colour scale is used for the corresponding MEM and CLEAN images.

The MEM and CLEAN percentage polarisation maps at full CLEAN beam resolution are largely similar and both show increasing fractional polarisation at the edges of the jet, a possible indication of the presence of a toroidal or helical magnetic field. At a resolution corresponding to half of the full CLEAN beam both maps still show similar structure, but the CLEAN map is predicting much higher fractional polarisation, at some points exceeding the theoretical maximum of 75% for synchrotron radiation. As the CLEAN maps are convolved with increasingly smaller beams the "bed of nails" effect of the δ function modelling employed by CLEAN becomes increasingly evident in both the Stokes I contours and the percentage polarisation colour scale.

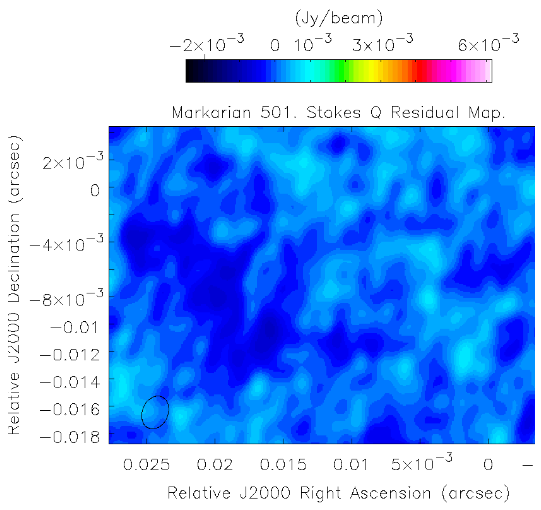
In contrast, the MEM maps smoothly increase in resolution – showing the same features from map to map, but in more detail and with more realistic values than the corresponding CLEAN maps. The background polarisation noise in each MEM map was lower than the corresponding CLEAN map and this became increasingly pronounced as the size of the convolving beam was reduced. The increased polarisation sensitivity and resolution of the MEM can be seen in the region of higher fractional polarisation located at a relative right ascension of about 5 mas and a declination of 0 mas. This is only hinted at in the MEM map convolved with the CLEAN but is clearly visible at all higher resolutions. Interestingly, CLEAN does detect the feature at $\frac{1}{2}$ and $\frac{1}{3}$ CLEAN beam resolutions,



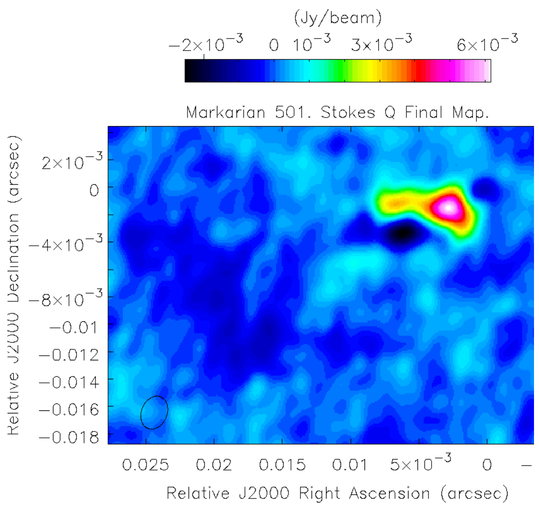
(a) Model Map.



(b) Convolved Model Map

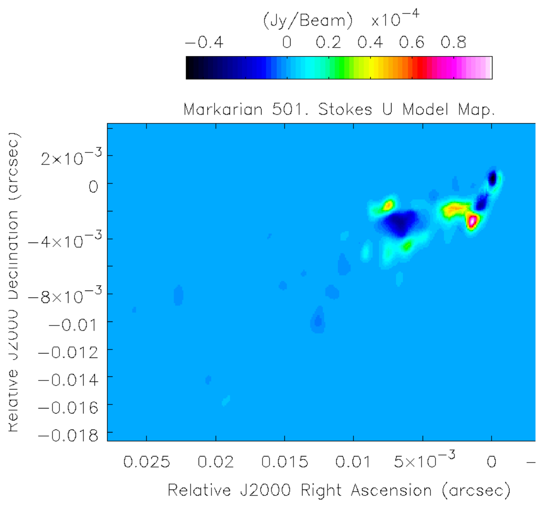


(c) Residual Map

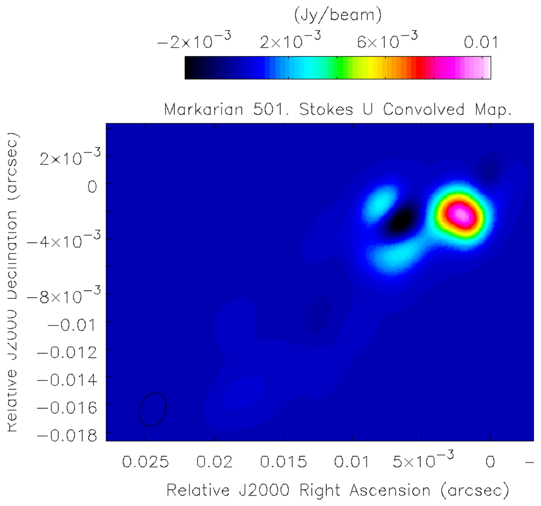


(d) Final MEM Map

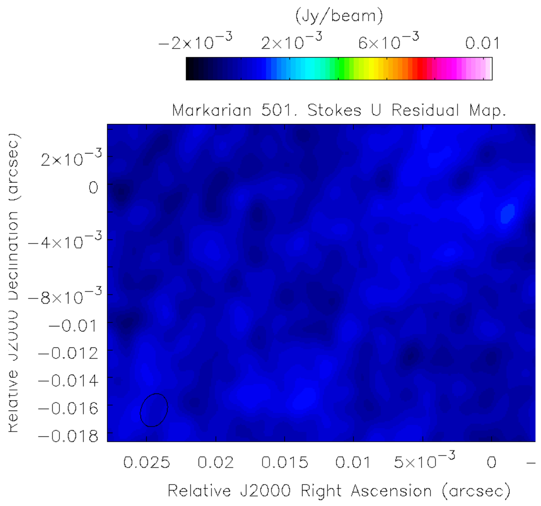
Figure 4.3: Stokes Q MEM maps of Markarian 501 at 4.971 GHz (6 cm). The colour scale indicates intensity in Jy/Pixel or Jy/Beam. The Convolved maps shown have been convolved with the CLEAN beam for the observation.



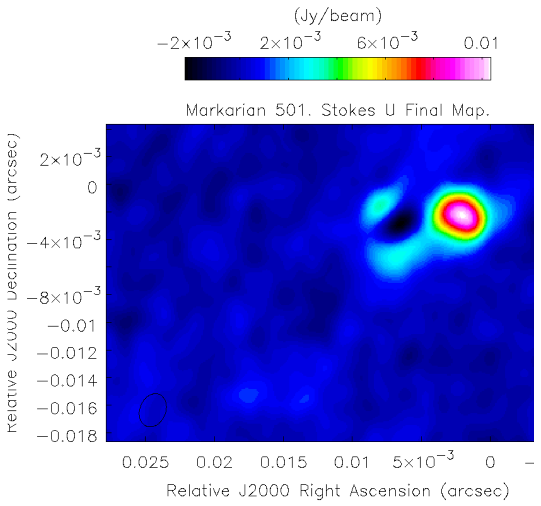
(a) Model Map



(b) Convolved Model Map



(c) Residual Map



(d) Final MEM Map

Figure 4.4: Stokes U MEM maps of Markarian 501 at 4.971 GHz (6 cm). The colour scale indicates intensity in Jy/Pixel or Jy/Beam. The Convolved maps shown have been convolved with the CLEAN beam for the observation.

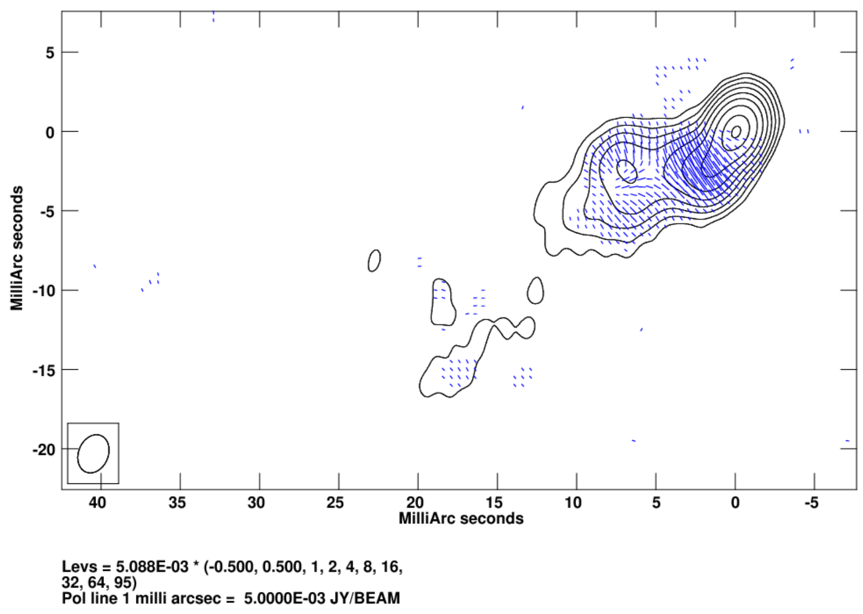
Table 4.4: A comparison of the intrinsic width measurements of Markarian 501 at the slice locations referenced in Figure 4.9. The MEM measurements were made by measuring the approximate width in the MEM intrinsic model map. Murphy et al. achieved their value by a statistical analysis.

Slice	Intrinsic MEM	Murphy et al.
	(mas)	(mas)
1	4.2	3.1
2	4.7	5.7
3	5.4	4.8

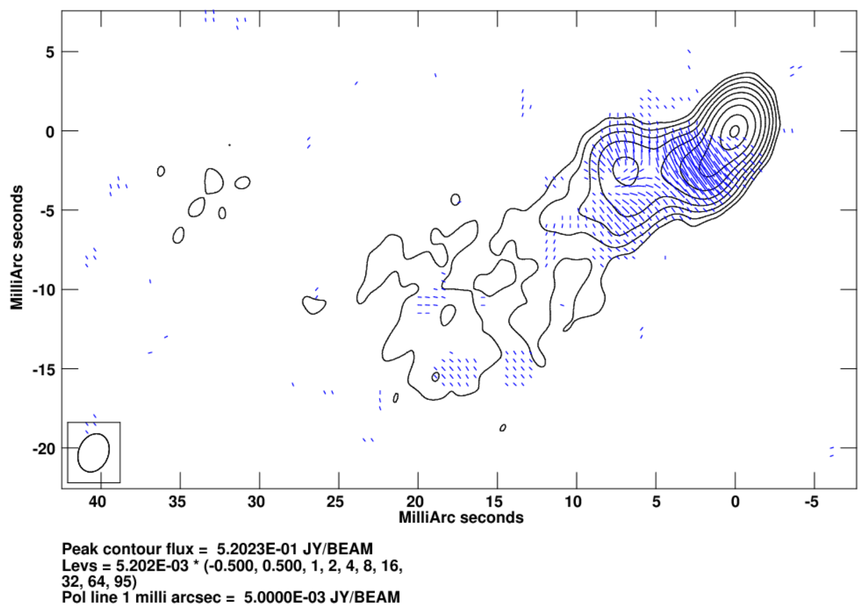
but the maps display unbelievably high fractional polarisation, which would make any other features in the map similarly suspect. The apparent reliability of the MEM maps over the CLEAN maps is in agreement with the Monte Carlo simulation results presented in Chapter 3 which suggested that the MEM is particularly successful in imaging fractional polarisation and polarisation angle.

Figures 4.5 to 4.8 show the polarisation angle maps of Markarian 501 at 4.971 GHz (6 cm) for both MEM and CLEAN at each of the four resolutions. Once again the MEM and CLEAN maps at full CLEAN resolution look broadly similar, however even at resolutions corresponding to half of the CLEAN beam a marked difference emerges between the two algorithms as the MEM image remains smooth while the CLEAN image begins to show signs of the "bed of nails" effect. Only polarisation angles corresponding to a total polarised flux ($P = \sqrt{Q^2 + U^2}$) high enough to rule out most off-source regions were shown. As the MEM produced total polarised flux maps with much lower noise than those produced by the CLEAN algorithm, lower cut–offs were required for the MEM maps than for the CLEAN maps. As the resolution of the maps increase, MEM shows the "spine-sheath" region at Right Ascension 5 mas, Declination -5 mas in increasing clarity and picks up some polarisation in the core region of the map. In contrast, while there is some agreement between the high resolution CLEAN and MEM maps, the CLEAN maps are quickly dominated by artifacts unlikely to represent the true structure of the source, raising questions about the accuracy of CLEAN–based polarisation angles in highly super–resolved images.

As an example of the potential ability to extract some useful data even from the intrinsic (unconvolved) MEM model, a measurement of the intrinsic jet width along a line corresponding approximately to slice 2 in Murphy et al. (2013) gives a result of approximately 4.7 mas (see Figure 4.9). To achieve this the jet width was estimated as the contour line corresponding to 0.05% of the peak model flux. This is in reasonable agreement with the value of 5.7 mas found by Murphy et

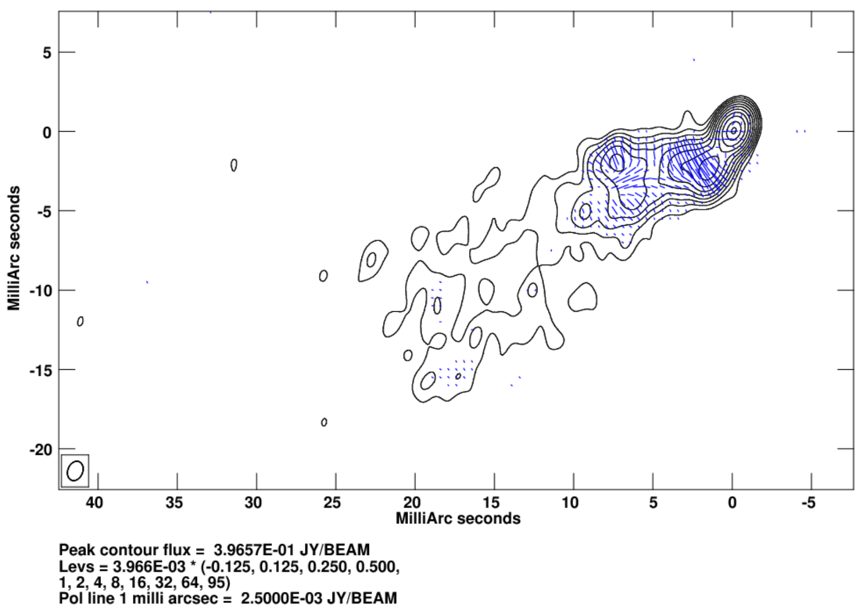


(a) MEM

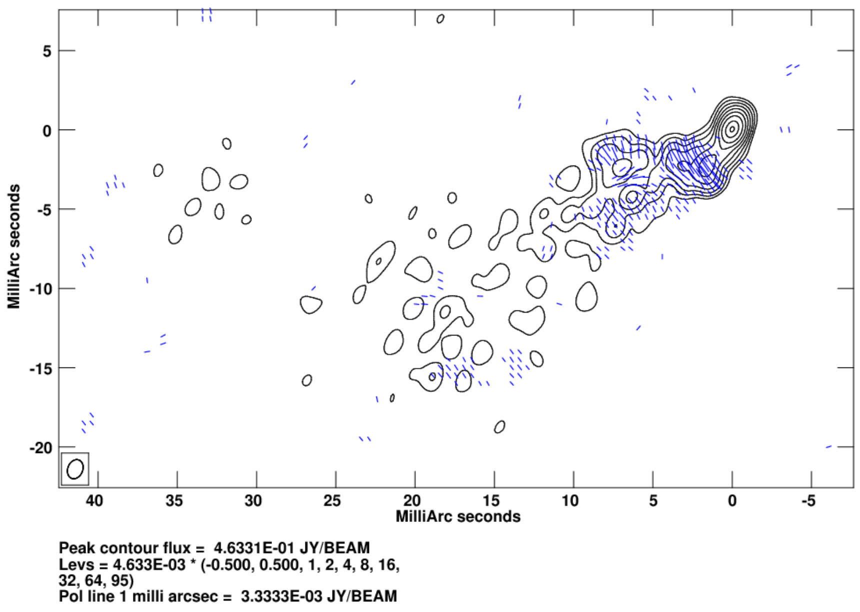


(b) CLEAN

Figure 4.5: MEM and CLEAN polarisation maps of of Markarian 501 at 4.971 GHz (6 cm). The maps have been made with the full CLEAN beam. The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.

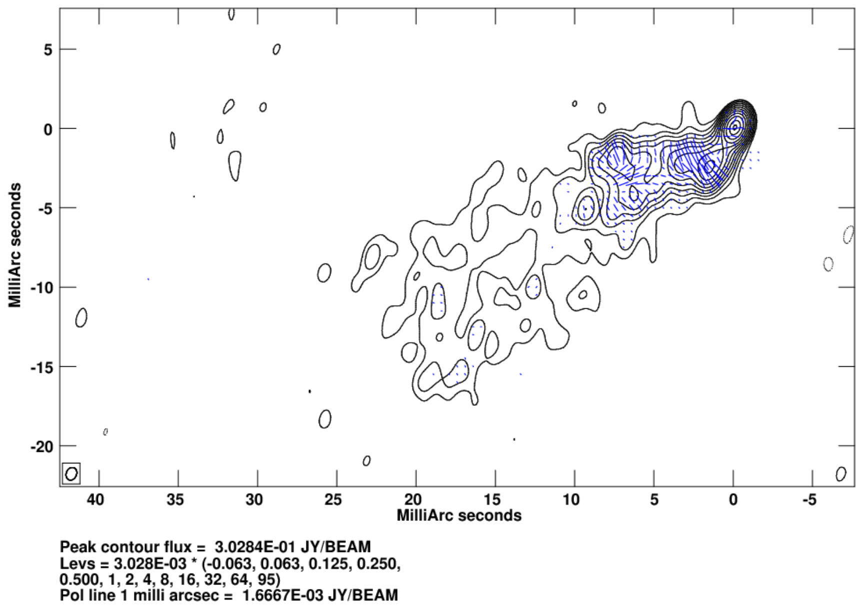

MilliArc seconds
MilliArc seconds
Peak contour flux = 3.9657E-01 JY/BEAM
Levs = 3.966E-03 * (-0.125, 0.125, 0.250, 0.500, 1, 2, 4, 8, 16, 32, 64, 95)
Pol line 1 milli arcsec = 2.5000E-03 JY/BEAM

(a) MEM

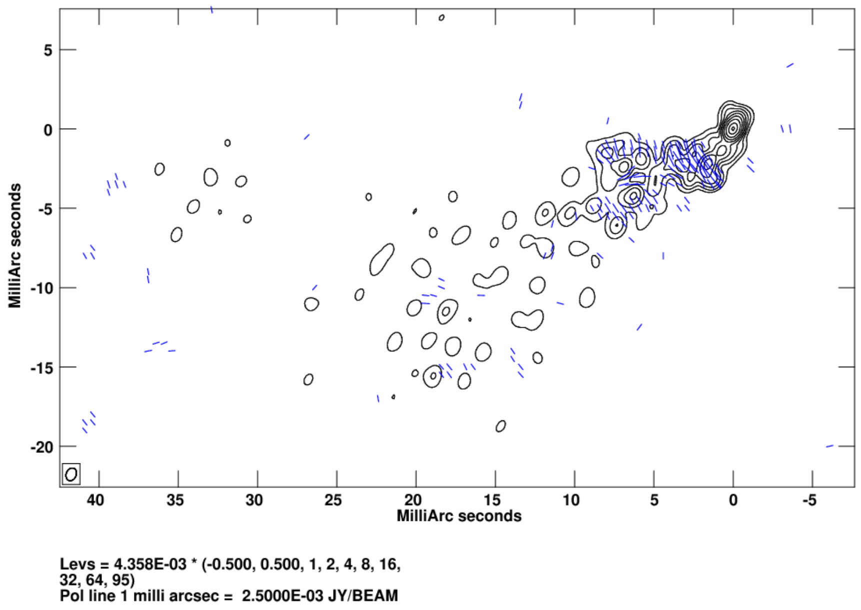

MilliArc seconds
MilliArc seconds
Peak contour flux = 4.6331E-01 JY/BEAM
Levs = 4.633E-03 * (-0.500, 0.500, 1, 2, 4, 8, 16, 32, 64, 95)
Pol line 1 milli arcsec = 3.3333E-03 JY/BEAM

(b) CLEAN

Figure 4.6: MEM and CLEAN polarisation maps of of Markarian 501 at 4.971 GHz (6 cm). The maps have been made with $\frac{1}{2}$ of the CLEAN beam. The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.

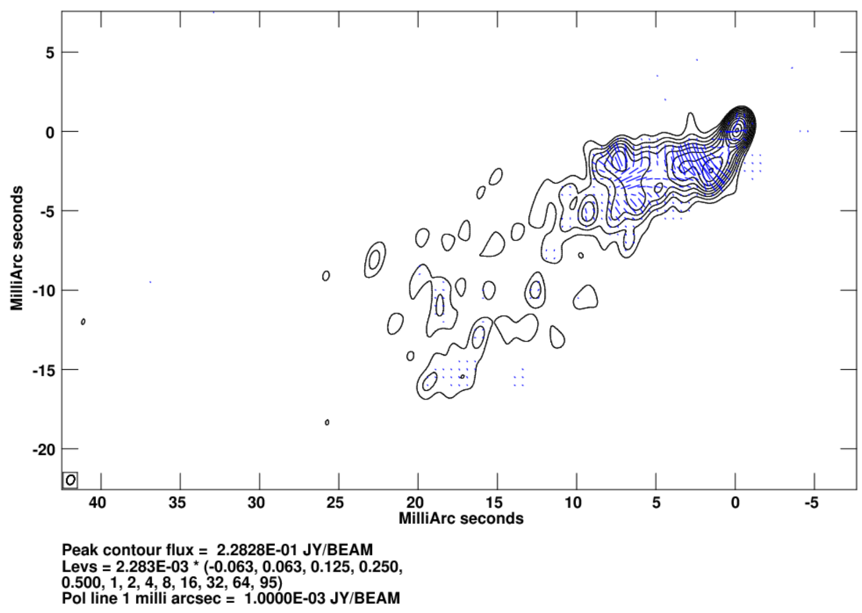

MilliArc seconds
MilliArc seconds
Peak contour flux = 3.0284E-01 JY/BEAM
Levs = 3.028E-03 * (-0.063, 0.063, 0.125, 0.250, 0.500, 1, 2, 4, 8, 16, 32, 64, 95)
Pol line 1 milli arcsec = 1.6667E-03 JY/BEAM

(a) MEM

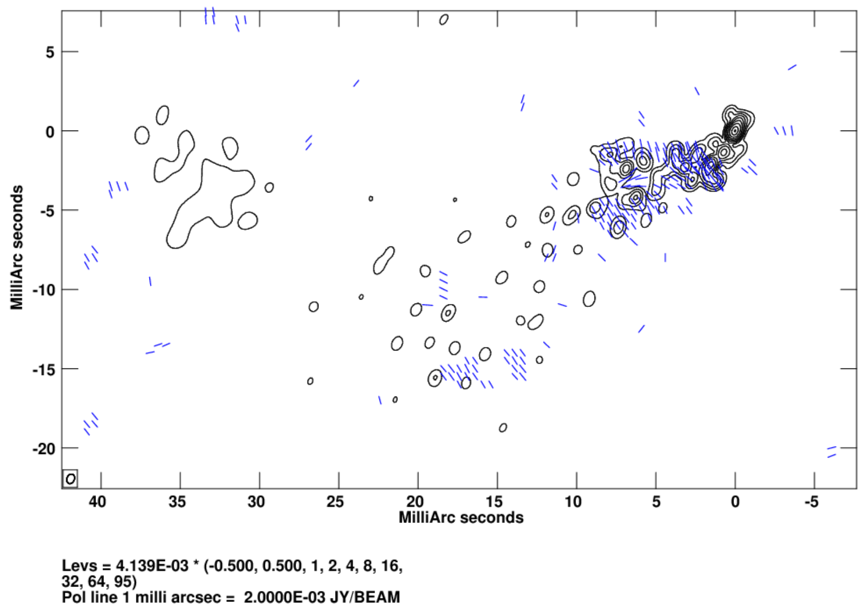

MilliArc seconds
MilliArc seconds
Levs = 4.358E-03 * (-0.500, 0.500, 1, 2, 4, 8, 16, 32, 64, 95)
Pol line 1 milli arcsec = 2.5000E-03 JY/BEAM

(b) CLEAN

Figure 4.7: MEM and CLEAN polarisation maps of of Markarian 501 at 4.971 GHz (6 cm). The maps have been made with $\frac{1}{3}$ of the CLEAN beam. The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.

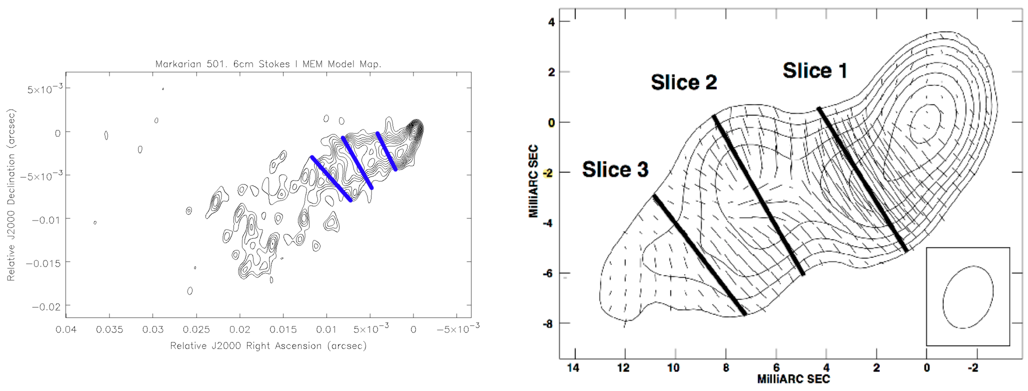


(a) MEM



(b) CLEAN

Figure 4.8: MEM and CLEAN polarisation maps of of Markarian 501 at 4.971 GHz (6 cm). The maps have been made with $\frac{1}{4}$ of the CLEAN beam. The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.



(a) MEM Model Map Slices (b) Murphy et al. Slices

Figure 4.9: Intrinsic width of Markarian 501 at 6cm from Murphy et al. 2013. Murphy et al. found slice 2 to have an intrinsic width of about 5.7mas, which is close to the measurement of 4.7 mas in the corresponding region in the model map in Figure 4.2.

al. (2013) using a Monte Carlo like χ^2 comparison test. Table 4.4 shows the intrinsic width of the jet measured at the three slices. Although the agreement varies and it is unclear which method may be more accurate – particularly as the corresponding slices are necessarily taken at slightly different regions due to the different types of maps under consideration, the MEM model map could be clearly of use in helping to constrain the intrinsic width of the jet. This could lead to increased accuracy and consistency in models such as that of Murphy et al.

Figures 4.14 to 4.17 show Faraday rotation measure maps of Markarian 501 made in AIPS using the 2cm, 4cm and 6cm observations. Estimates of the Faraday rotation were obtained in each pixel, by obtaining a linear fit of the polarisation angles vs. the square of the wavelength in each pixel. The fitted Faraday rotation values were written out only when the significance of the linear fit was high enough that spurious fits in off-source regions were neglected. Different cut off points were used for different convolution sizes, resulting in only relatively reliable RM values were written out. Again results are presented with the restoring beams listed in Table 4.3 corresponding to the full CLEAN beam and $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{4}$ of the CLEAN beam. The same colour scale is used for all images. At full CLEAN beam resolution the rotation measure maps made with each algorithm look very similar, though the MEM map appears somewhat more blotchy at the edges of the jet. The fact that the MEM polarisation maps produced reliable polarisation measurements over a larger area of the source emission region leads to Faraday rotation estimates being available over a larger region of the source in the MEM

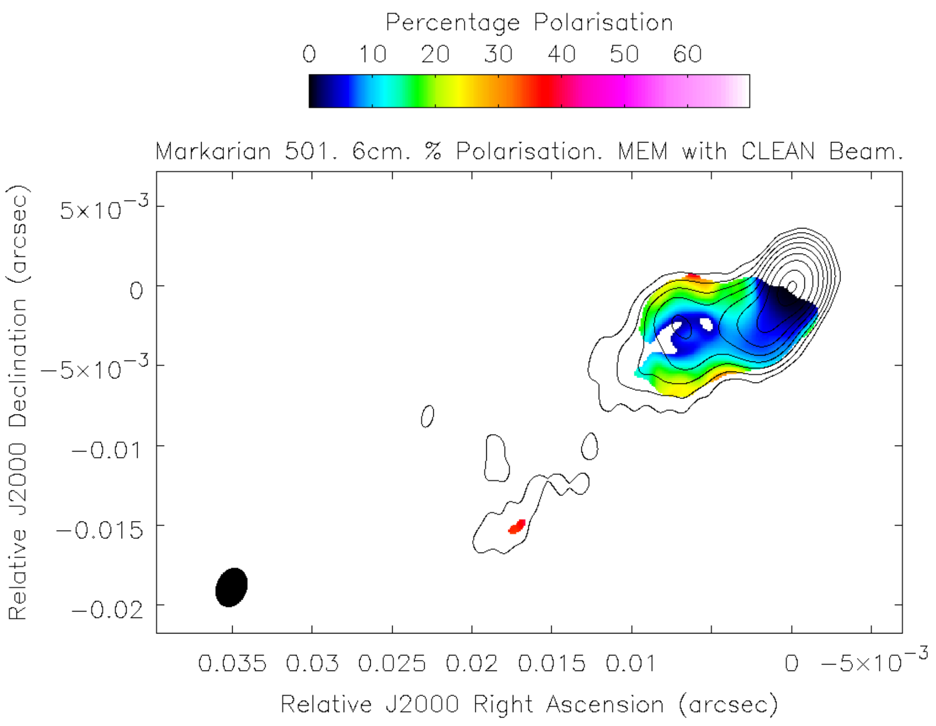
Faraday rotation maps compared to the CLEAN maps; however, this did not lead to the detection of interesting Faraday rotation structures that were not visible in the CLEAN Faraday rotation maps in this case.

Figures 4.18 to 4.21 show intrinsic polarisation angle maps of Markarian 501 at 6 cm made with CLEAN and the final MEM maps restored with the beams listed in Table 4.3. The intrinsic polarisation angle was calculated in the same linear fit as the Faraday rotation measure; i.e., it is the y intercept in the linear fit of the polarisation angle vs. the square of the wavelength obtained for each pixel. The pixels for which values are available correspond to those for which acceptable Faraday rotation estimates could be obtained, based on the same criterion indicated above. Because the MEM Faraday rotation maps contained more pixels with reliable Faraday rotation estimates, this leads to estimates of the intrinsic polarisation angle in a larger number of pixels in the MEM maps, compared to their CLEAN counterparts. There is strong agreement between the images in all cases, with EVPAs aligned transverse to the jet direction from just outside the core to a knot visible in Stokes I, possibly related to a change in jet direction, a shock, or a globule of plasma travelling out along the jet. In an optically thin part of the jet this corresponds to a magnetic field along the jet direction, suggesting a magnetic field structure with a strong toroidal component. The presence of 'spine-sheath' type behaviour at the location of the knot suggests that a helical magnetic field may be responsible for the observed polarisation behaviour.

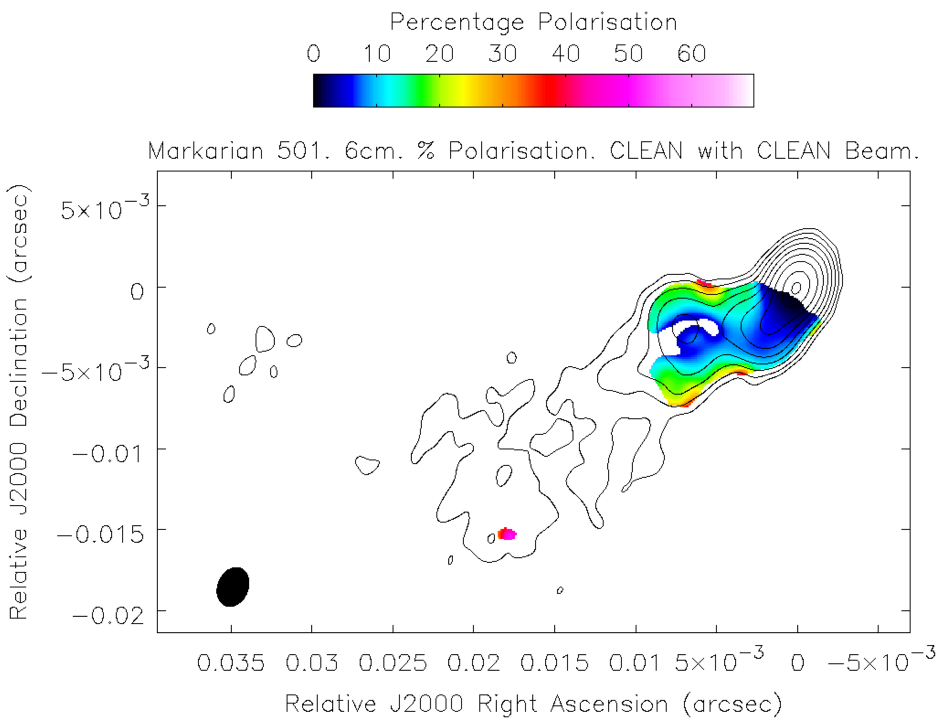
The spine-sheath structure can be seen clearly in the high resolution MEM maps. Comparing the MEM percentage polarisation and intrinsic polarisation angle maps convolved with $\frac{1}{4}$ of the CLEAN beam it is clear that the 'sheath' region of the structure corresponds to a region of higher than normal percentage polarisation. The co-incidence of these features is strong evidence for a helical magnetic field structure at this point in the jet.

4.2 1633+382

J1633+382 is a Quasar with notably high fractional linear polarisation in the optical and a low spectral peak with a redshift of $z = 1.813$, corresponding to a distance of 14,000 Mpc, or 4.3×10^{26} m (data from the MOJAVE project, Lister et al. 2009). Observations were made at six frequencies; 4.612 GHz, 5.088 GHz, 7.904 GHz, 8.8710 GHz, 12.9270 GHz and 15.3710 GHz. The initial

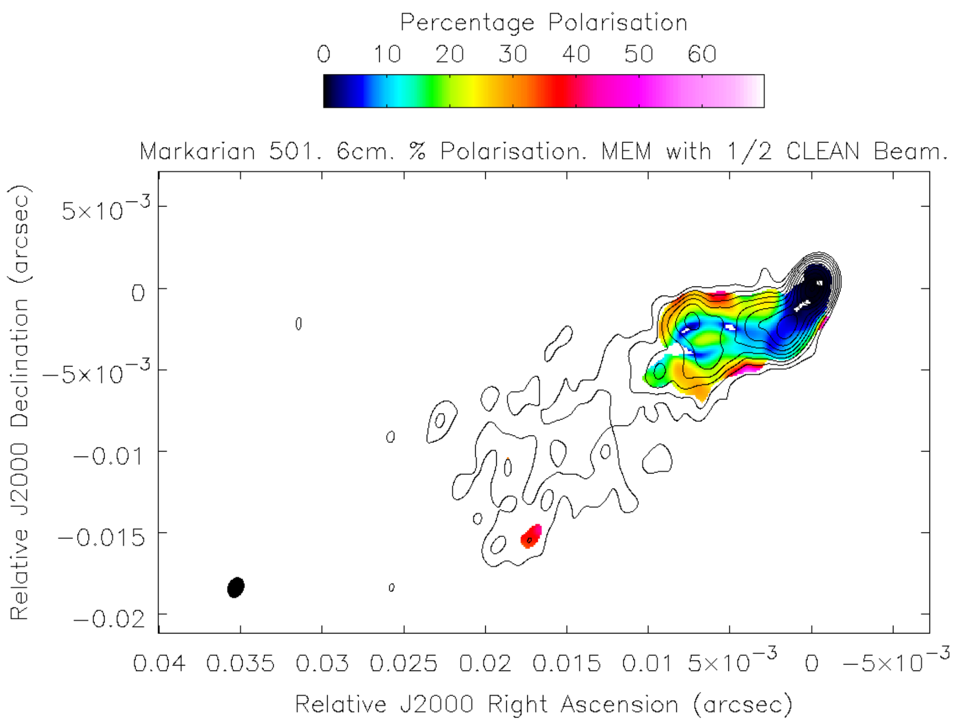


(a) MEM

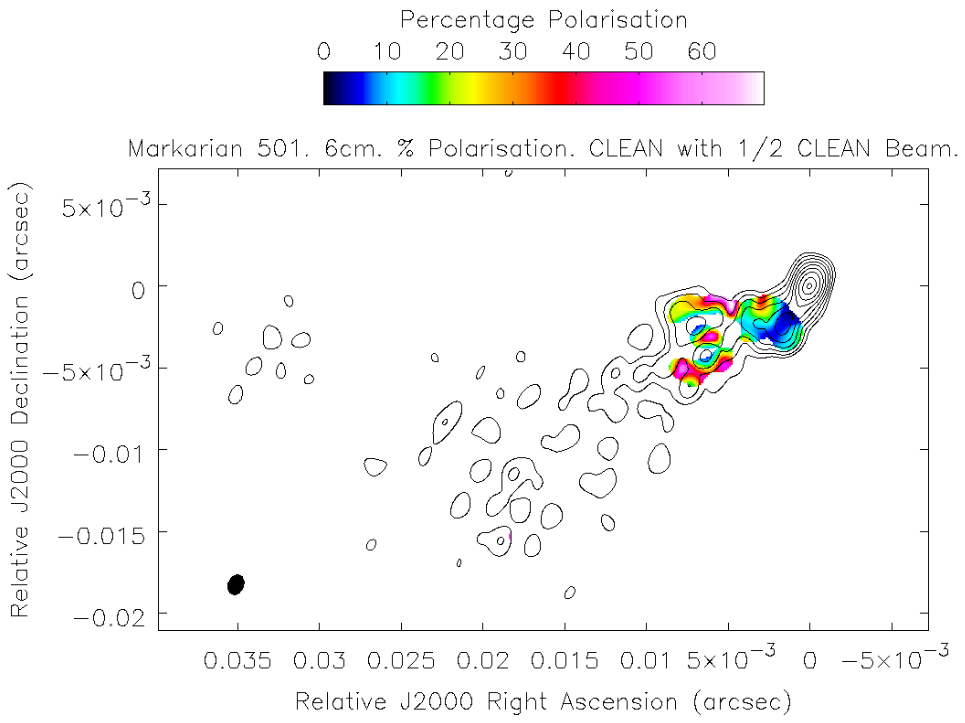


(b) CLEAN

Figure 4.10: Percentage polarisation measure maps of Markarian 501 made with the MEM and CLEAN convolved with the full CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates percentage polarisation.

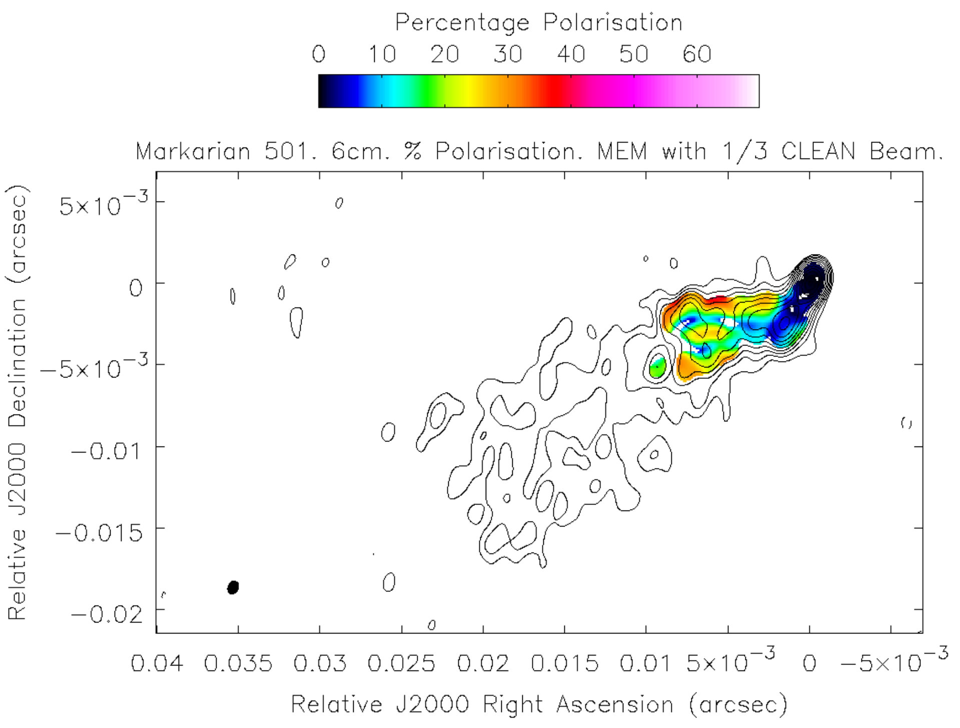


(a) MEM

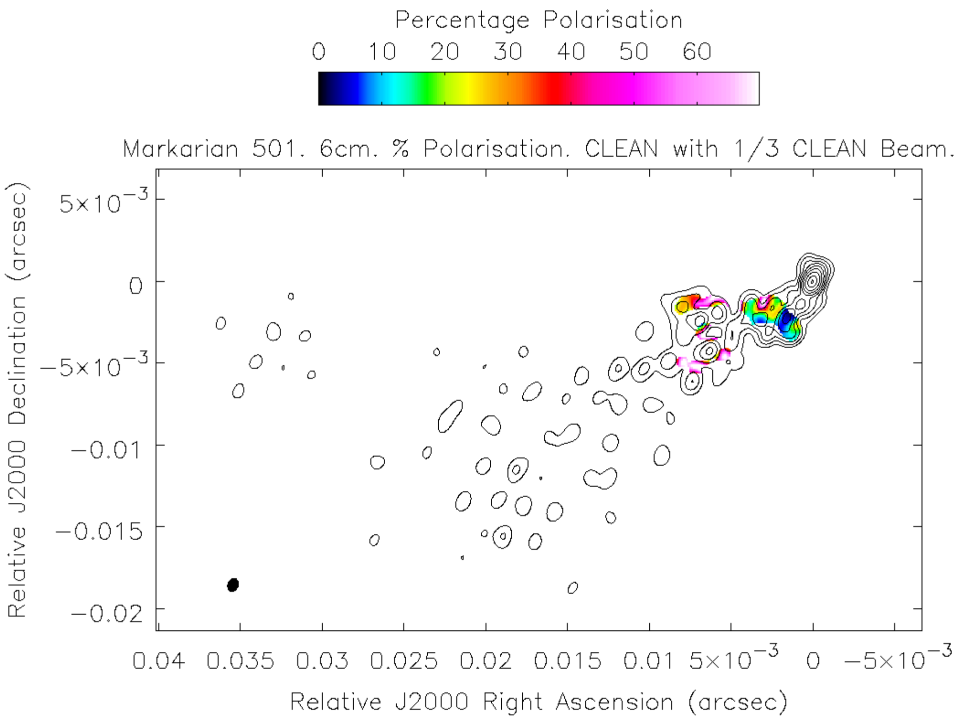


(b) CLEAN

Figure 4.11: Percentage polarisation measure maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{2}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates percentage polarisation.

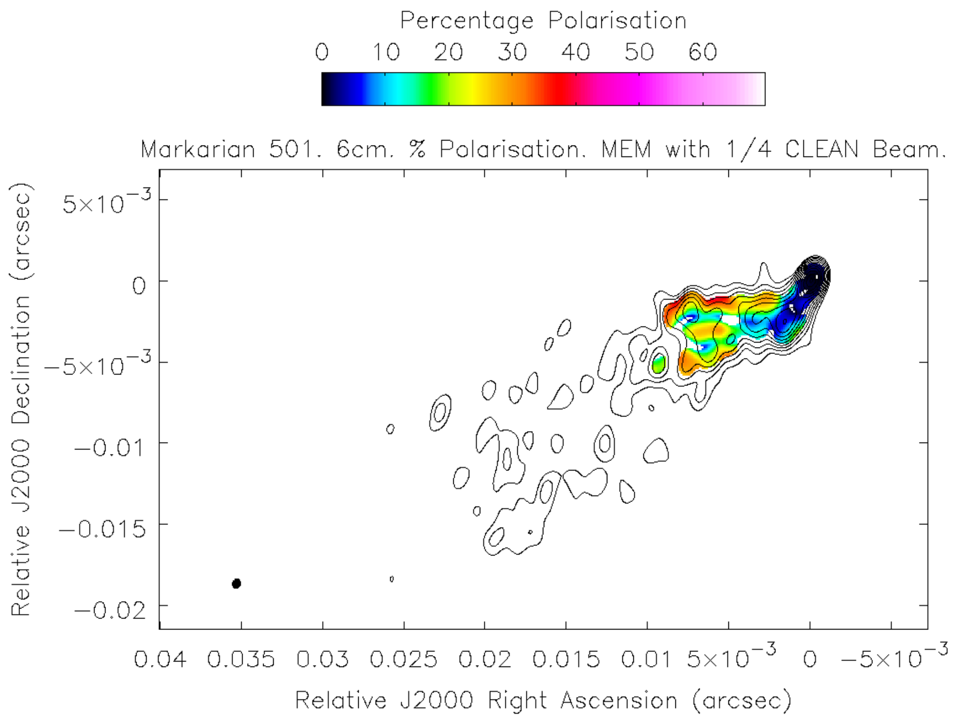


(a) MEM

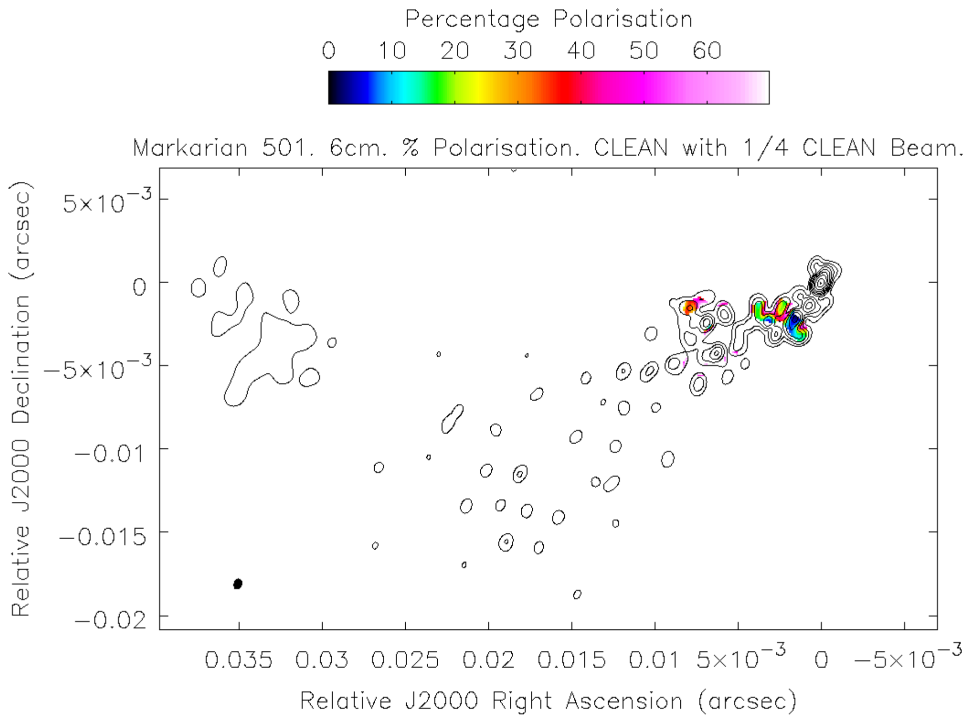


(b) CLEAN

Figure 4.12: Percentage polarisation measure maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{3}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates percentage polarisation.

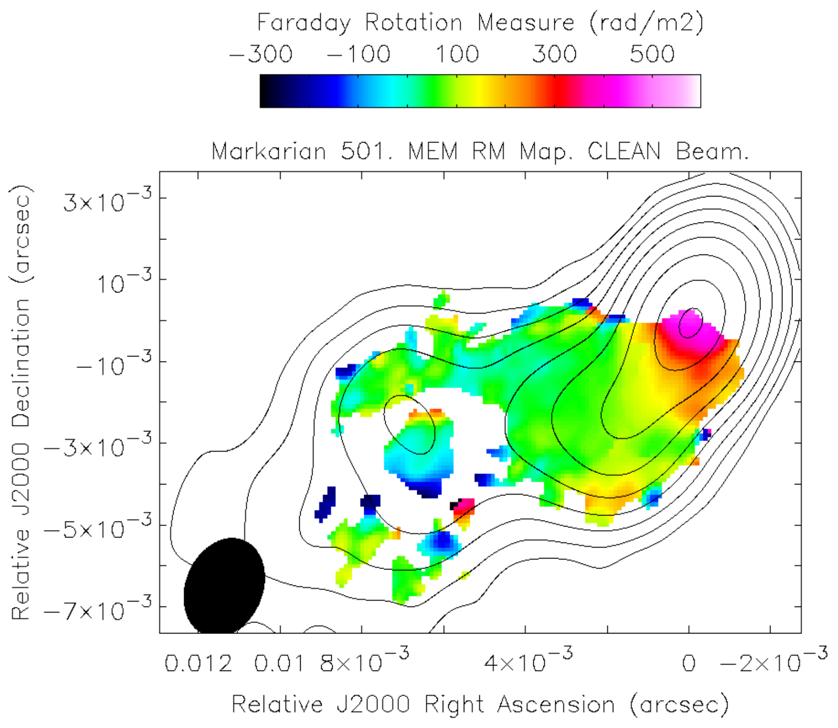


(a) MEM

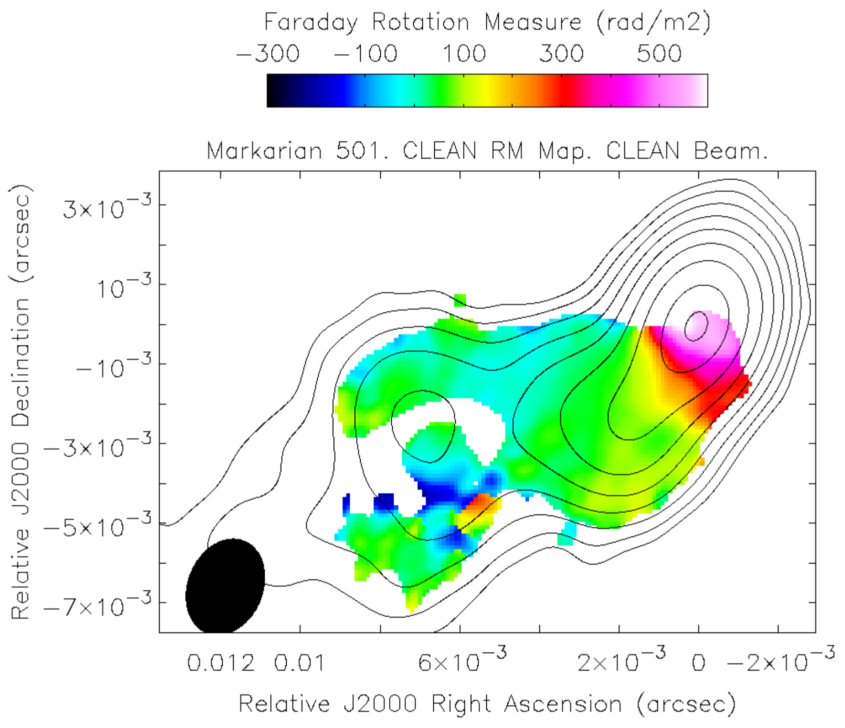


(b) CLEAN

Figure 4.13: Percentage polarisation measure maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{4}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates percentage polarisation.

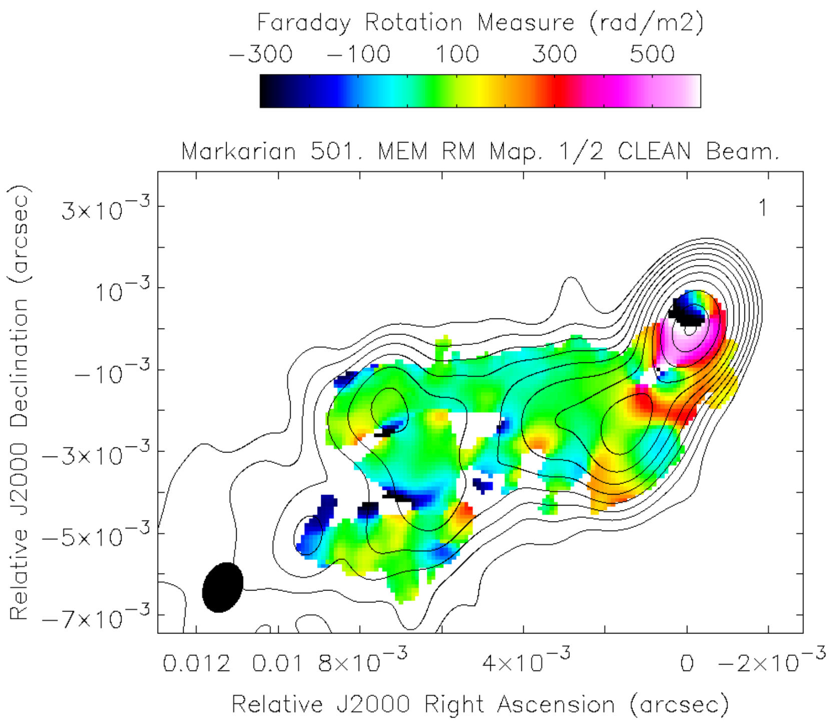

Faraday Rotation Measure (rad/m2)
−300 −100 100 300 500
Markarian 501. MEM RM Map. CLEAN Beam.
3×10⁻³
10⁻³
−10⁻³
−3×10⁻³
−5×10⁻³
−7×10⁻³
Relative J2000 Declination (arcsec)
0.012 0.01 8×10⁻³ 4×10⁻³ 0 −2×10⁻³
Relative J2000 Right Ascension (arcsec)

(a) MEM

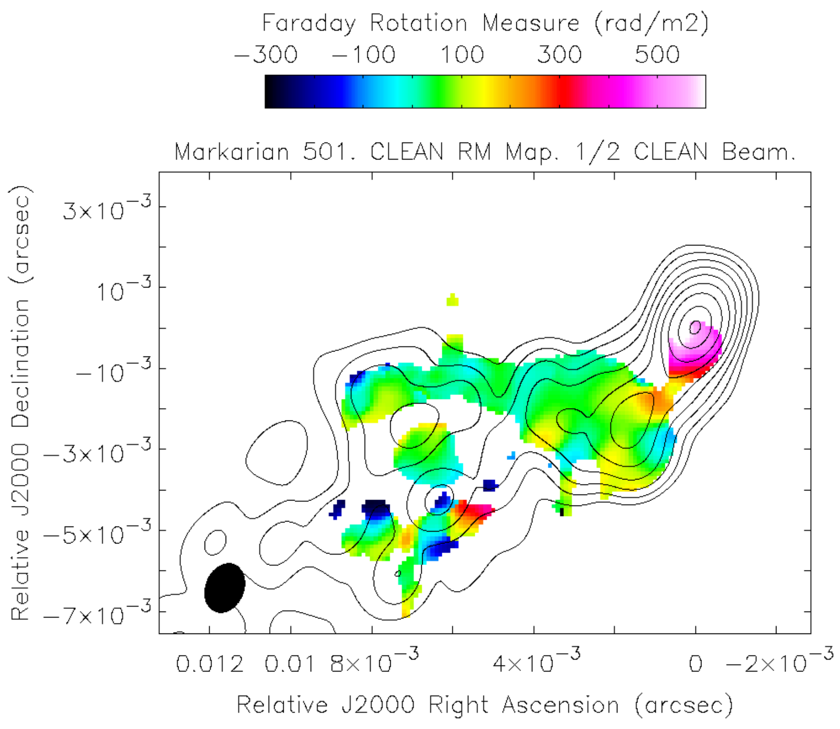

Faraday Rotation Measure (rad/m2)
−300 −100 100 300 500
Markarian 501. CLEAN RM Map. CLEAN Beam.
3×10⁻³
10⁻³
−10⁻³
−3×10⁻³
−5×10⁻³
−7×10⁻³
Relative J2000 Declination (arcsec)
0.012 0.01 6×10⁻³ 2×10⁻³ 0 −2×10⁻³
Relative J2000 Right Ascension (arcsec)

(b) CLEAN

Figure 4.14: Faraday rotation measure maps of Markarian 501 made with the MEM and CLEAN convolved with the full CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates the Faraday rotation measure.

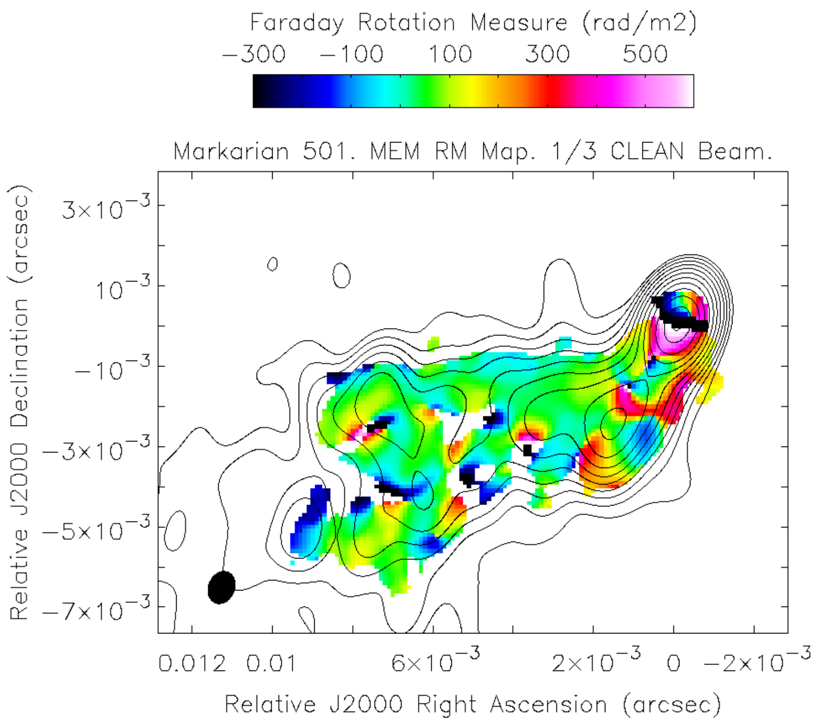

Faraday Rotation Measure (rad/m2)
−300 −100 100 300 500
Markarian 501. MEM RM Map. 1/2 CLEAN Beam.
1
Relative J2000 Declination (arcsec)
Relative J2000 Right Ascension (arcsec)

(a) MEM

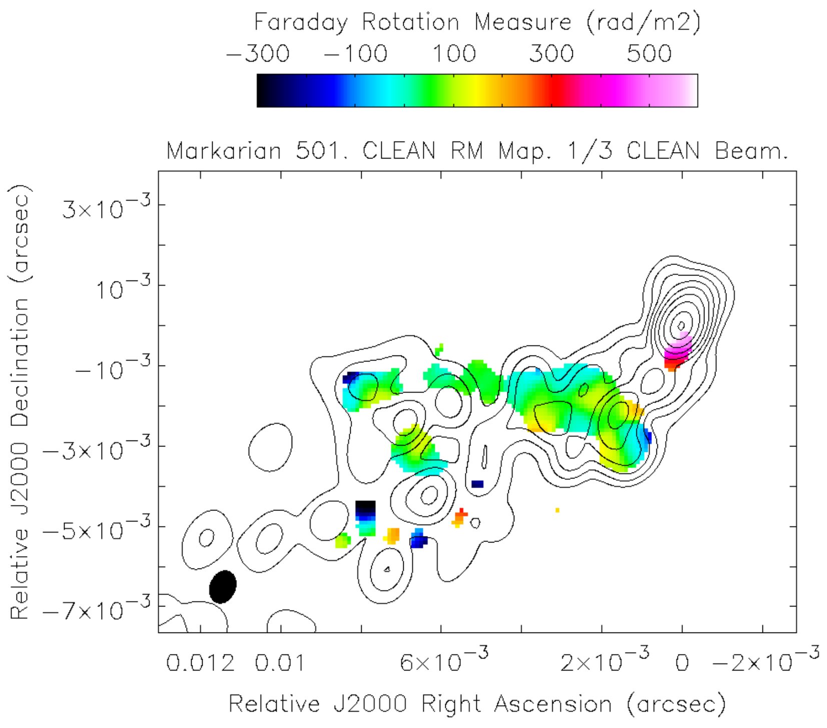

Faraday Rotation Measure (rad/m2)
−300 −100 100 300 500
Markarian 501. CLEAN RM Map. 1/2 CLEAN Beam.
Relative J2000 Declination (arcsec)
Relative J2000 Right Ascension (arcsec)

(b) CLEAN

Figure 4.15: Faraday rotation measure maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{2}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates the Faraday rotation measure.

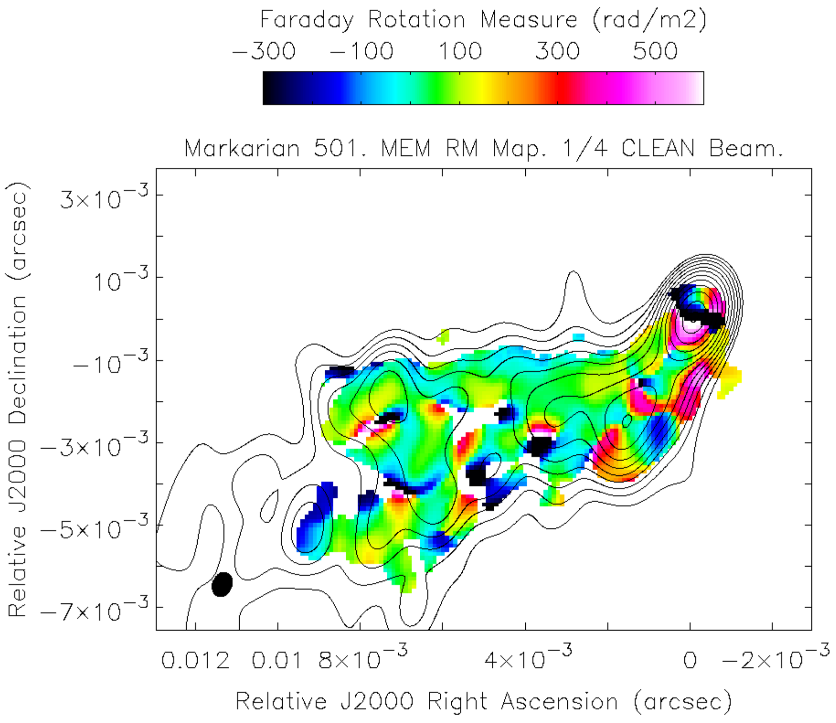


(a) MEM

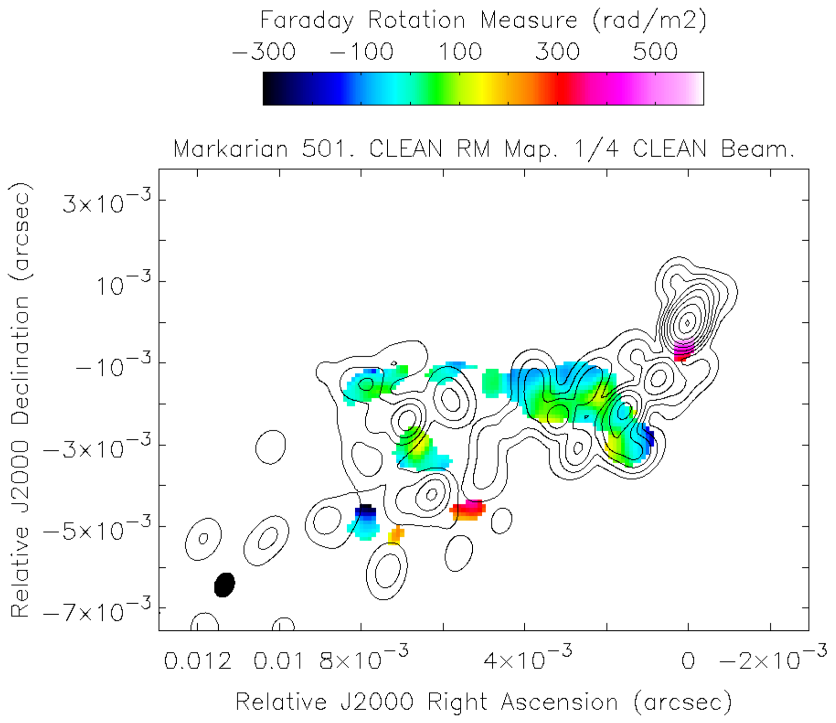


(b) CLEAN

Figure 4.16: Faraday rotation measure maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{3}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates the Faraday rotation measure.

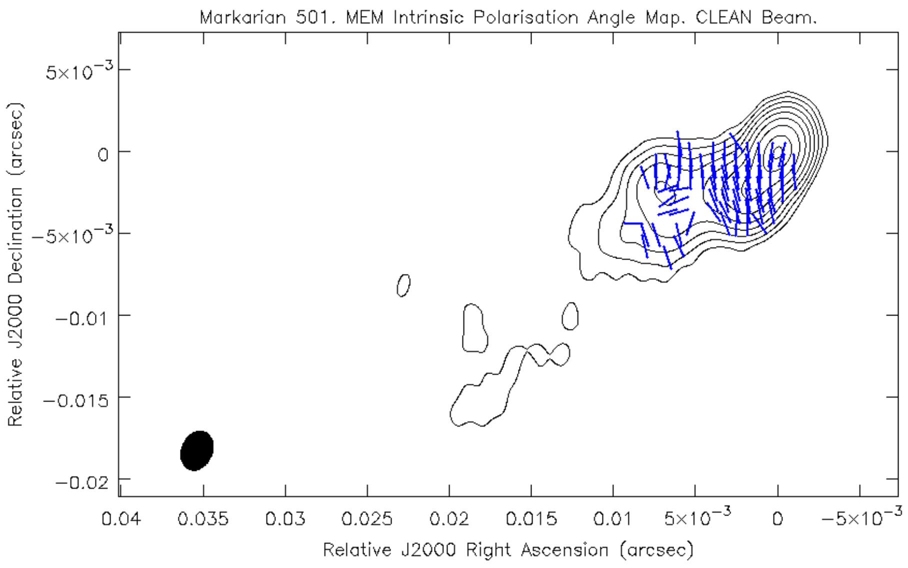


(a) MEM

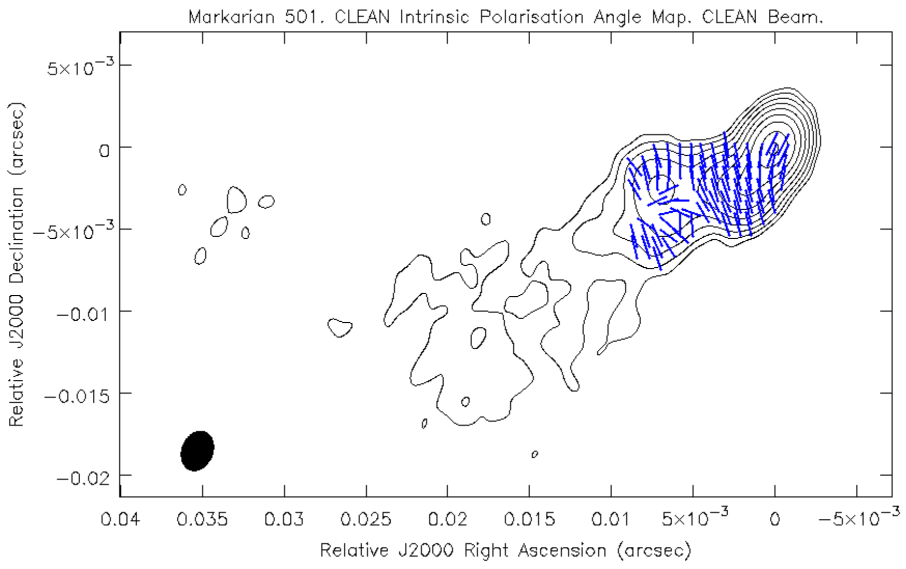


(b) CLEAN

Figure 4.17: Faraday rotation measure maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{4}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The colour scale indicates the Faraday rotation measure.

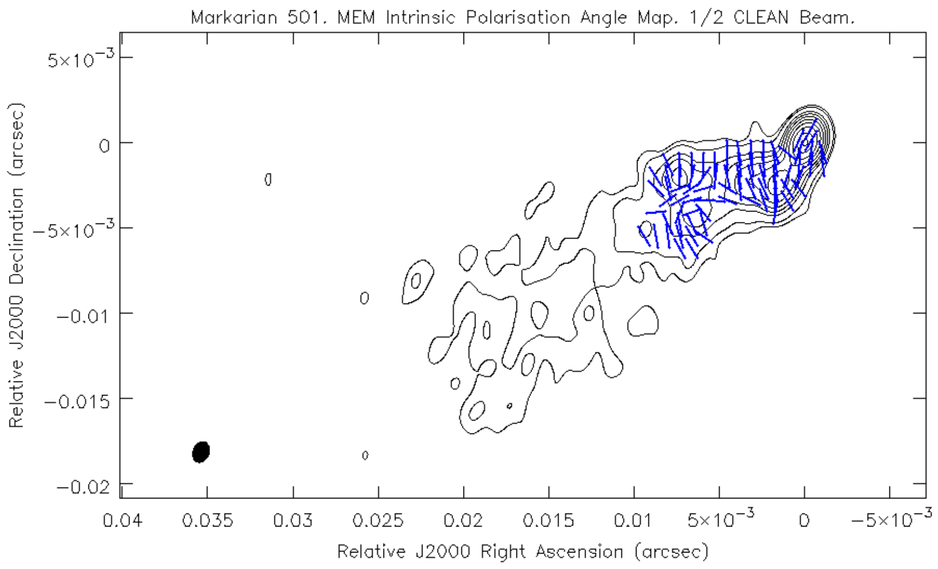


(a) MEM

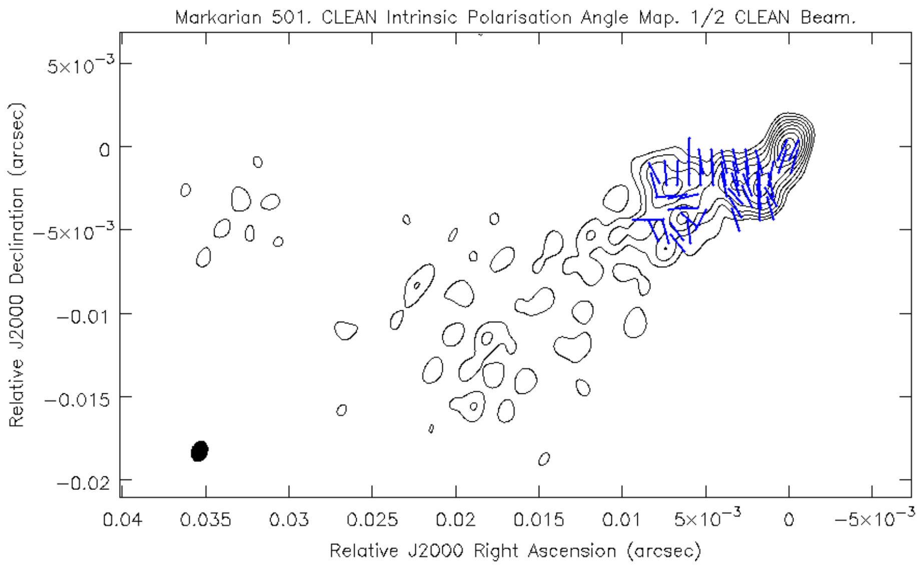


(b) CLEAN

Figure 4.18: Intrinsic polarisation angle maps of Markarian 501 made with the MEM and CLEAN convolved with the full CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The blue dashes indicate the direction of the intrinsic polarisation angle.

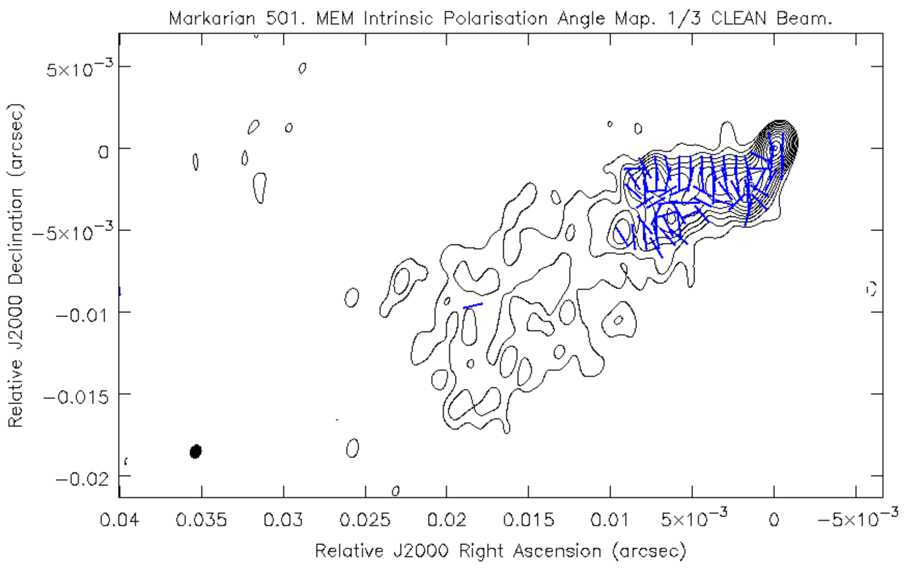


(a) MEM

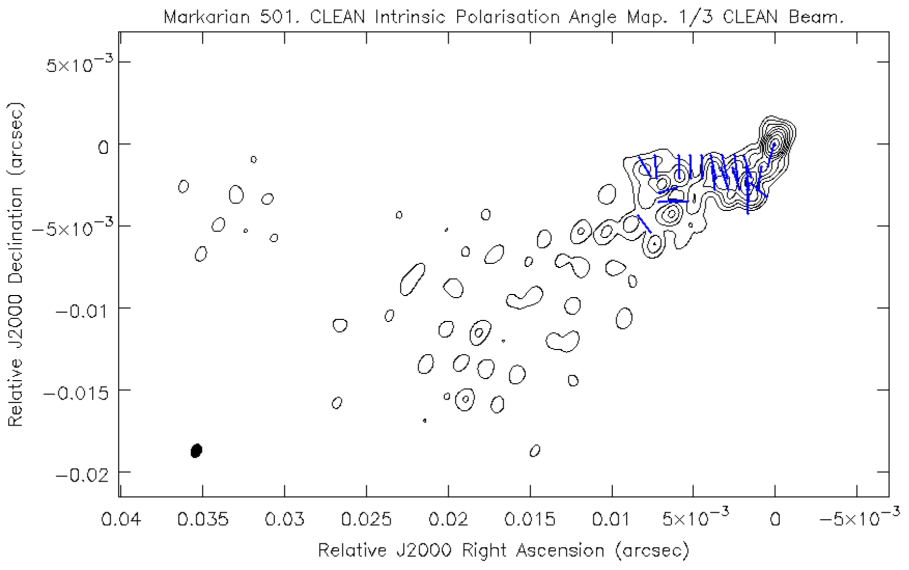


(b) CLEAN

Figure 4.19: Intrinsic polarisation angle maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{2}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The blue dashes indicate the direction of the intrinsic polarisation angle.

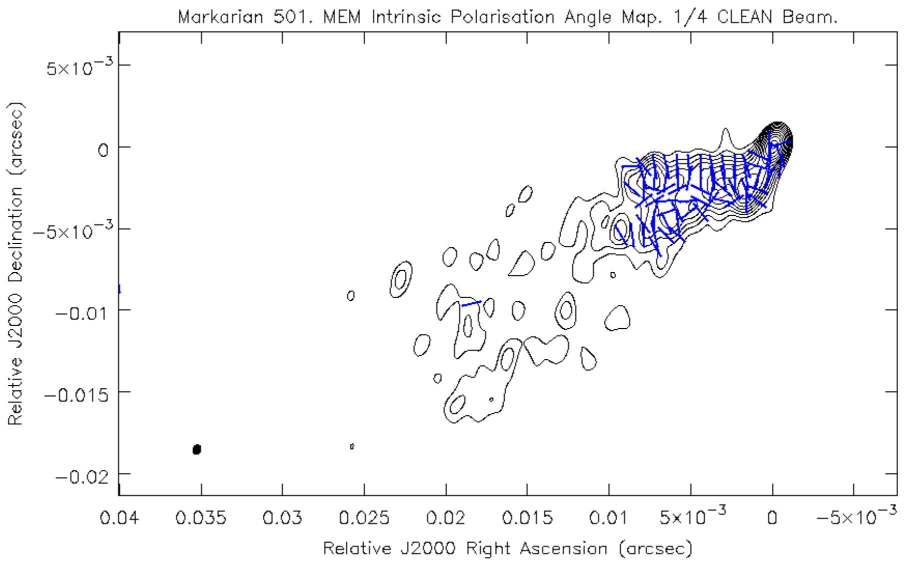


(a) MEM

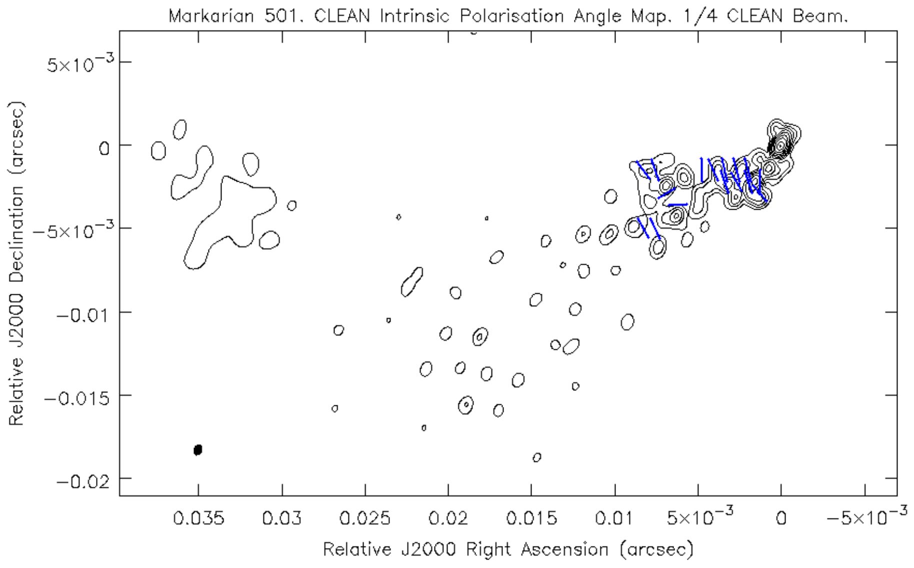


(b) CLEAN

Figure 4.20: Intrinsic polarisation angle maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{3}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The blue dashes indicate the direction of the intrinsic polarisation angle.



(a) MEM



(b) CLEAN

Figure 4.21: Intrinsic polarisation angle maps of Markarian 501 made with the MEM and CLEAN convolved with $\frac{1}{4}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.971 GHz. The blue dashes indicate the direction of the intrinsic polarisation angle.

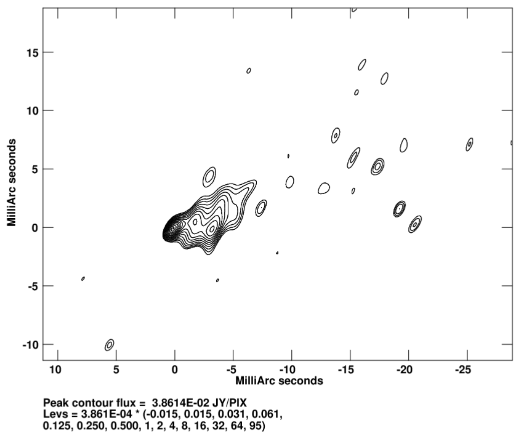
Table 4.5: MEM parameters used to image 1633+382. In all cases the restoring beam used was the CLEAN beam, the maximum number of iterations was 10,000, a flat bias map was used and flux was conserved with parameter 2. It was found that an estimated final RMS noise of 0 for all Stokes parameters gave the best deconvolution results for the lower three frequencies. $N_{exclude}$ was set to 150 pixels in all cases.

Freq. (GHz)	4.6	5.1	7.9	8.9	12.9	15.4
Stokes I Flux (Jy)	2.2	2.2	2.2	2.3	2.5	2.3
Stokes Q Flux (Jy)	0.15	0.15	0.15	0.14	0.15	0.17
Stokes U Flux (Jy)	0.17	0.17	0.17	0.15	0.15	0.17
Stokes I RMS (mJy/Beam)	0	0	0	0.98	0.6	1.24
Stokes Q RMS (mJy/Beam)	0	0	0	0.94	0.94	0.61
Stokes U RMS (mJy/Beam)	0	0	0	1.34	1.35	0.64
Δ_{step}	3	3	1.5	1.5	0.5	1.0
q_{factor}	0.5	2	0.05	2	0.03	0.1
w_p	1	1	2	2	2	4

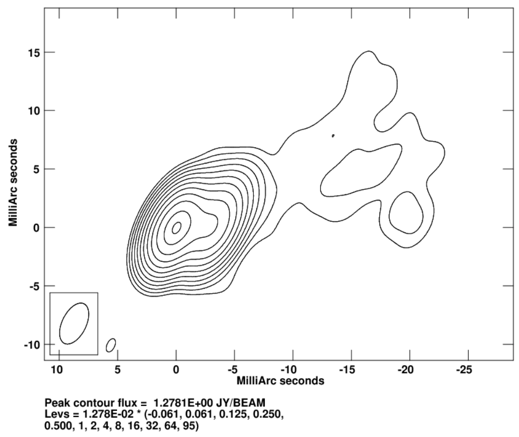
calibration and imaging of the data were carried out by Reichstein (2012). The MEM deconvolution of the data was carried out with PMEM using the parameters listed in Table 4.5. The parameters were selected by trial and error to achieve the best possible deconvolution. The CLEAN maps were used to help select expected RMS residuals for the three highest frequencies, but the three lower frequencies converged best with an RMS value of zero.

Figures 4.22 to 4.24 show the MEM model, convolved, residual and final maps for the three Stokes parameters at 4.612 GHz. Several distinct jet components are clearly visible in the model Stokes I MEM map, which are blended together when convolved with the full CLEAN beam. The residual maps show that good convergence has been reached for all Stokes parameters. Some structure is visible in the residual Q and U maps, notably around the core of the source where the MEM model map has trouble forming the 'spikey' structure needed to describe the sudden increase at the core. Using CLEAN to first image the brighter regions of the source and imaging the remaining flux with the MEM may result in a better image, however as the structure seen in the residual maps is within a few times the RMS of the residual noise, nearly all the information about the source contained in the data is included within the model maps.

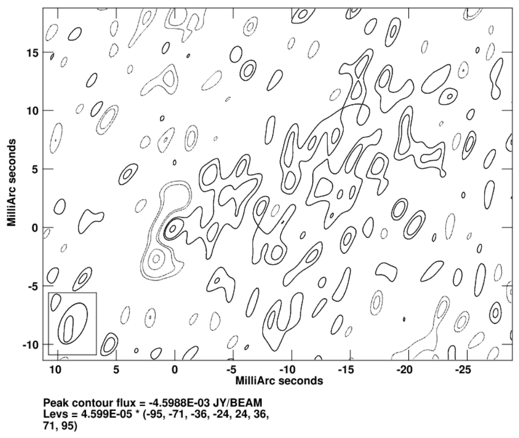
Figures 4.25 to 4.28 show the CLEAN and MEM percentage polarisation maps of 1633+382 at 4.612 GHz imaged with four beams, again corresponding to the full CLEAN beam, $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{4}$ of the CLEAN beam (see Table 4.6 for full details of each beam). The cut off in fractional polarisation shown was determined



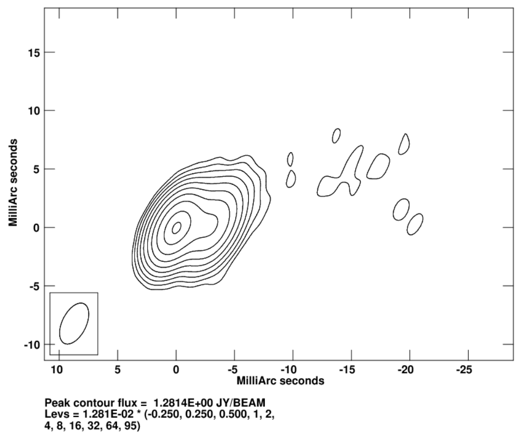
(a) Model Map



(b) Convolved Model Map



(c) Residual Map

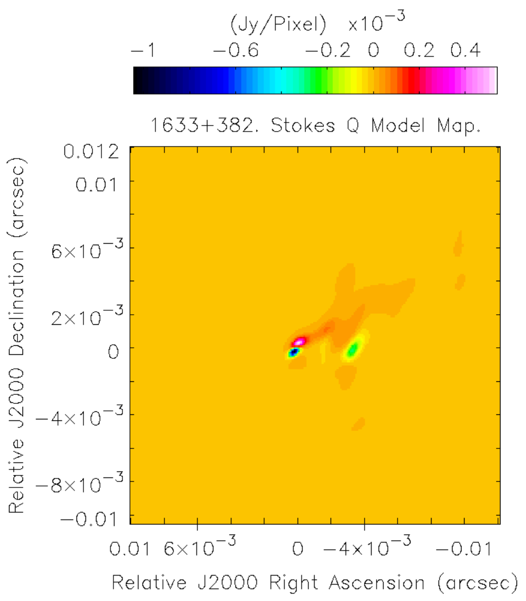


(d) Final MEM Map

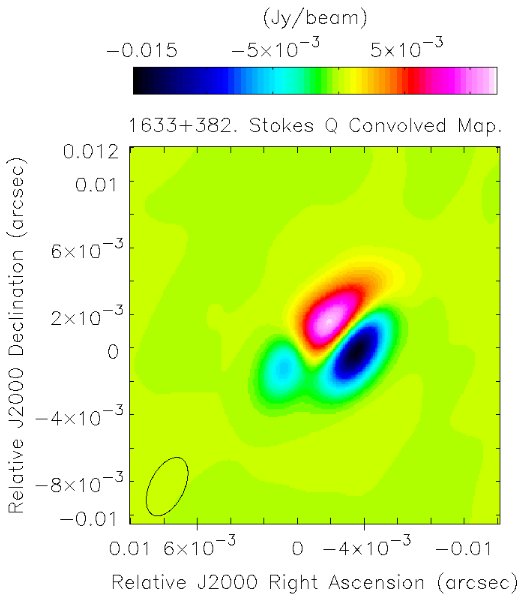
Figure 4.22: Stokes I MEM maps of J1633+382 at 4.612 GHz. The Convolved maps shown have been convolved with the CLEAN beam for the observation.

by finding the maximum total polarised flux in a region away from the source and clipping to show only fractional polarisation corresponding to a higher total polarised flux.

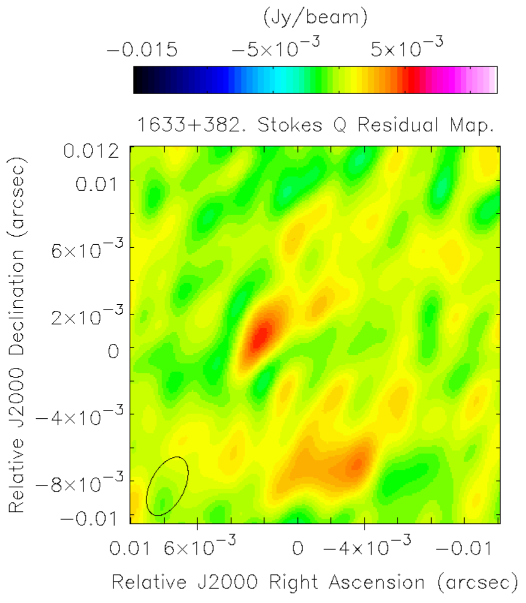
The MEM and CLEAN fractional polarisation maps at full CLEAN resolution are almost identical – clearly both algorithms do a good job at this resolution. In contrast, the MEM image at twice this resolution looks very different to the corresponding CLEAN image. In the MEM image a high resolution version of the full CLEAN beam image is presented, with evidence for an increase in fractional polarisation at the edges of the jet appearing. The CLEAN image has picked up much less statistically significant polarised flux, and the increase in fractional polarisation at the edges is not clear from the CLEAN image alone.



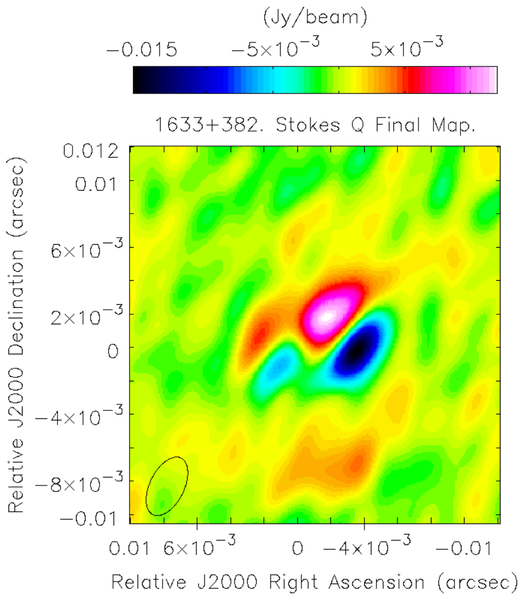
(a) Model Map



(b) Convolved Model Map

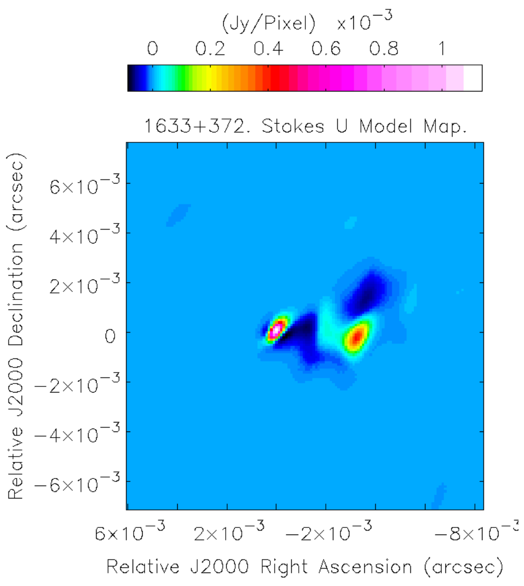


(c) Residual Map

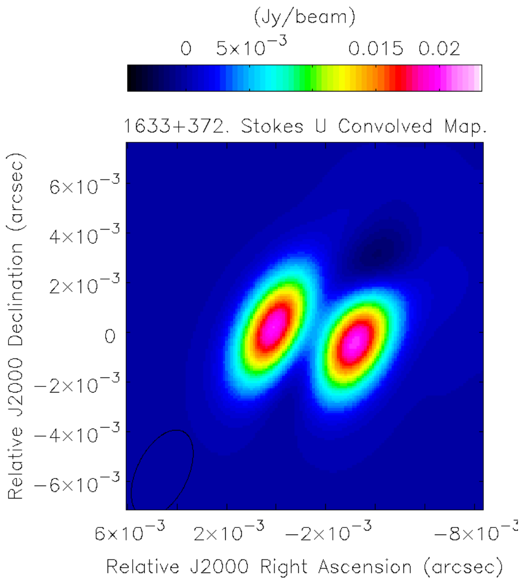


(d) Final MEM Map

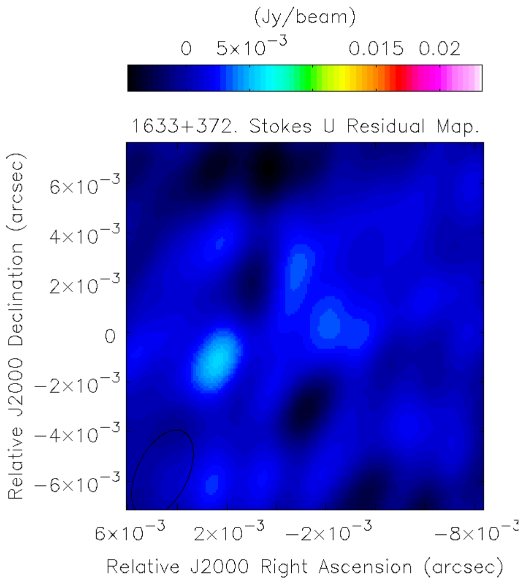
Figure 4.23: Stokes Q MEM maps of J1633+382 at 4.612 GHz. The colour scale indicates intensity in Jy/Pixel or Jy/Beam. The Convolved maps shown have been convolved with the CLEAN beam for the observation.



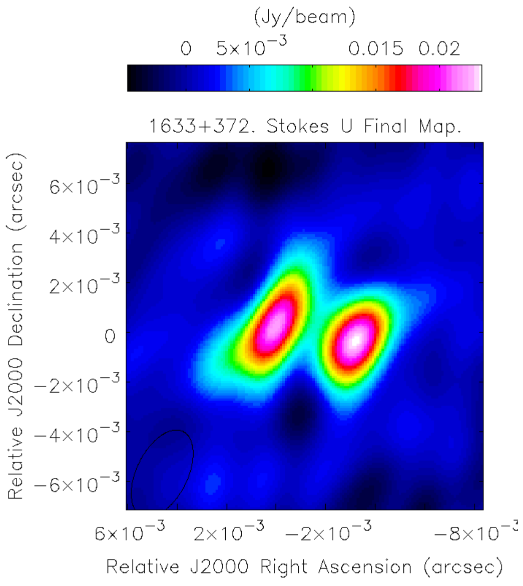
(a) Model Map



(b) Convolved Model Map



(c) Residual Map



(d) Final MEM Map

Figure 4.24: Stokes *U* MEM maps of J1633+382 at 4.612 GHz. The colour scale indicates intensity in Jy/Pixel or Jy/Beam. The Convolved maps shown have been convolved with the CLEAN beam for the observation.

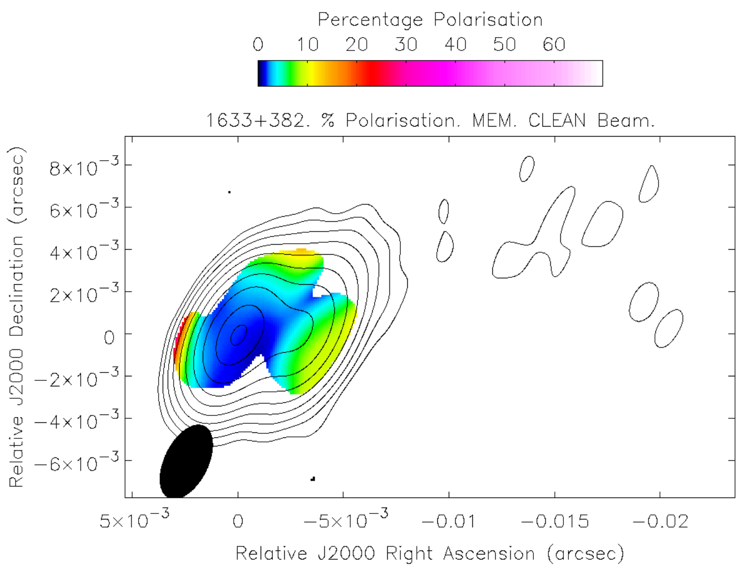
Table 4.6: The various beams used in the imaging of 1633+382.

Frac. of CLEAN beam	Major Axis (mas)	Minor Axis (mas)	Position Angle (deg.)
1	3.77	2.04	-26.53
$\frac{1}{2}$	1.885	1.02	-26.53
$\frac{1}{3}$	1.257	0.68	-26.53
$\frac{1}{4}$	0.9425	0.51	-26.53

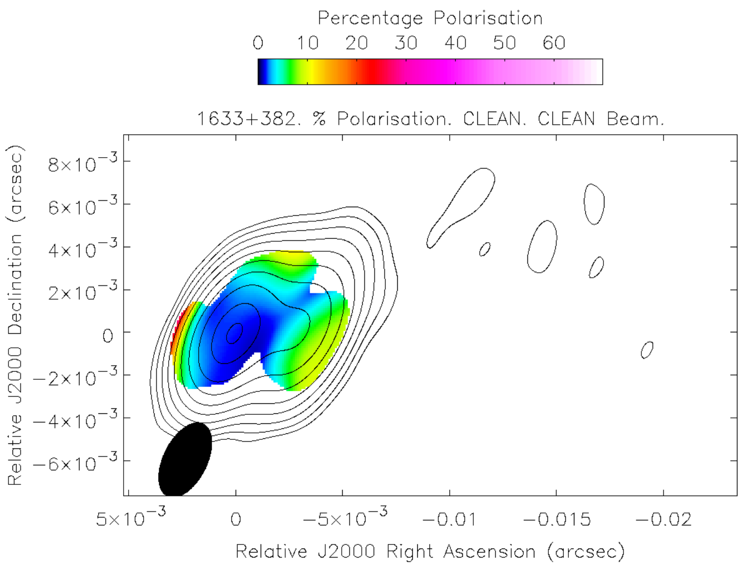
CLEAN images at higher resolutions are badly affected by δ function artefacts and are of little use. In contrast, the high resolution MEM images give a clearer indication of the jet direction and the way in which the increase in fractional polarisation at the jet edges follows the jet direction. In this way, the MEM maps contribute unique evidence for the presence of a toroidal magnetic field in 1633+382.

Figures 4.29 to 4.30 show the polarisation angle maps. Only polarisation angles corresponding to a total polarised flux ($P = \sqrt{Q^2 + U^2}$) high enough to rule out most off-source regions were shown. As the MEM produced total polarised flux maps with much lower noise than those produced by the CLEAN algorithm, lower cut–offs were required for the MEM maps than for the CLEAN maps. The polarisation angle maps follow the same trend as the fractional polarisation maps, with MEM and CLEAN producing similar results at resolutions corresponding to the full CLEAN beam and $\frac{1}{2}$ of the CLEAN beam. At higher resolutions the CLEAN maps are dominated by artefacts, while the MEM maps show the change in polarisation angle between the core and the brightest component of the jet in detail. The longer blue dashes in the core and the brightest component of the jet also link regions of high polarised flux to regions of high total intensity, however the high resolution MEM fractional polarisation map in Figure 4.28 shows that the fraction of the emission that is polarised is actually higher out along the jet than in the core region.

MEM maps of the three Stokes parameters were made for all six frequencies corresponding to the beams listed in Table 4.6. Polarisation angle maps for each frequency were constructed and used to create a Faraday rotation measure map for the entire frequency range in AIPS. Corresponding maps were also made with the CLEAN algorithm. Figures 4.31 and 4.32 show the resulting RM maps for each of the four beams used. It is again clear that at full CLEAN resolution the MEM and CLEAN maps are very similar, though again the MEM map appears to be a little patchier than the CLEAN map. As the resolution increases however,

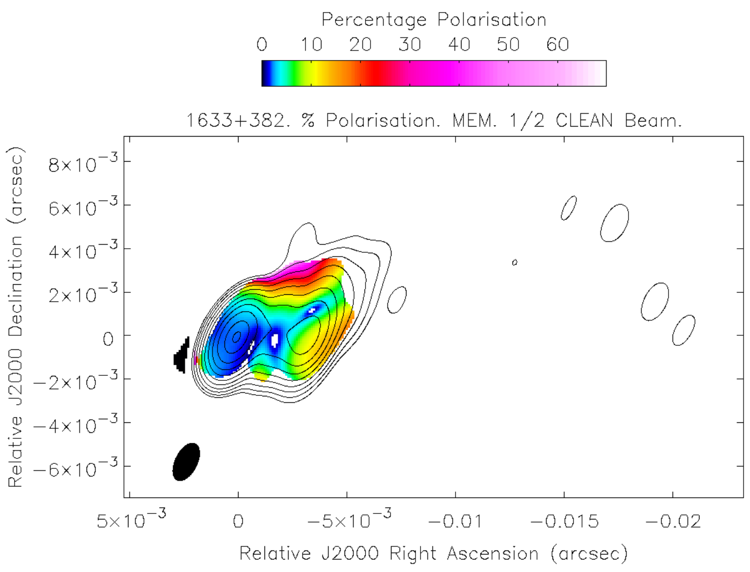


(a) MEM. CLEAN Beam

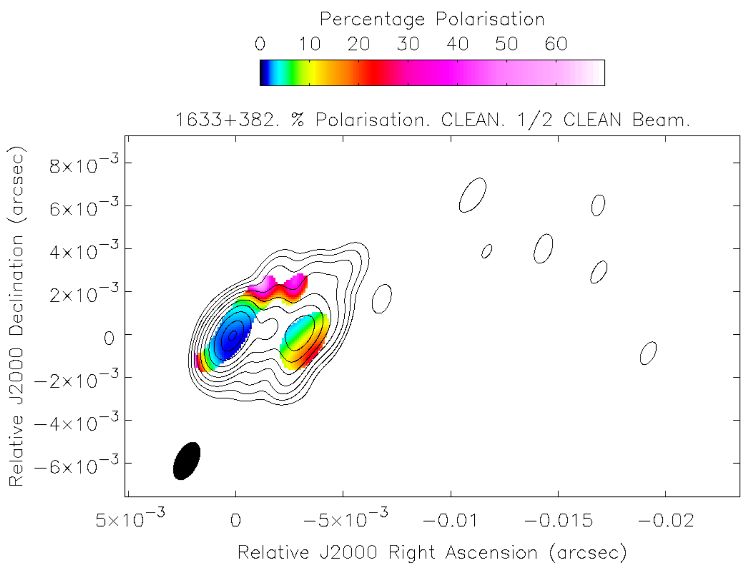


(b) CLEAN. CLEAN Beam

Figure 4.25: Percentage polarisation maps of 1633+382 made with the MEM and CLEAN convolved with the full CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the percentage polarisation.

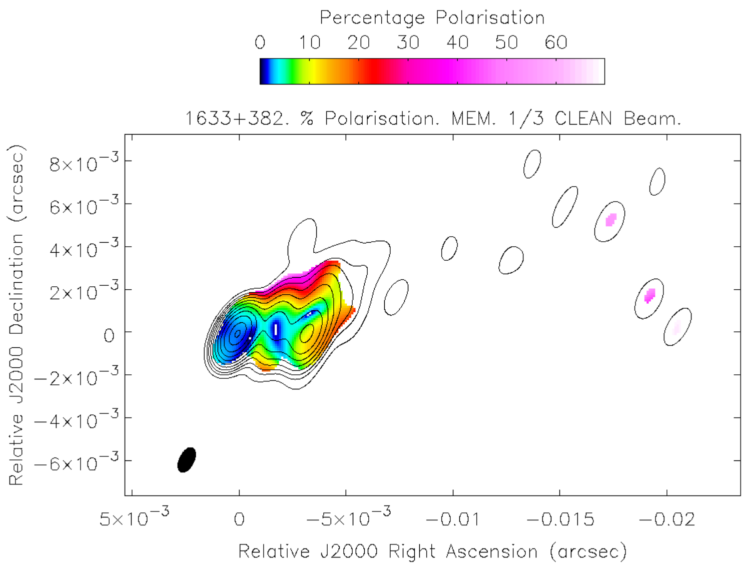


(a) MEM. $\frac{1}{2}$ CLEAN Beam

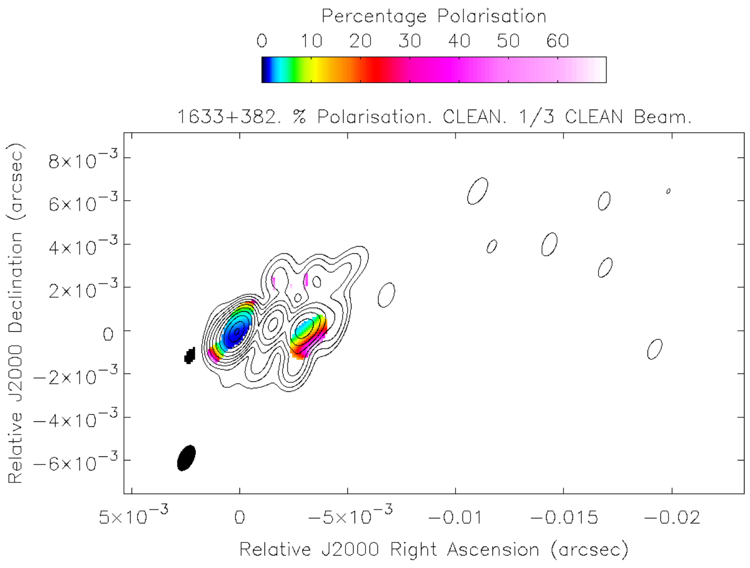


(b) CLEAN. $\frac{1}{2}$ CLEAN Beam

Figure 4.26: Percentage polarisation maps of 1633+382 made with the MEM and CLEAN convolved with $\frac{1}{2}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the percentage polarisation.

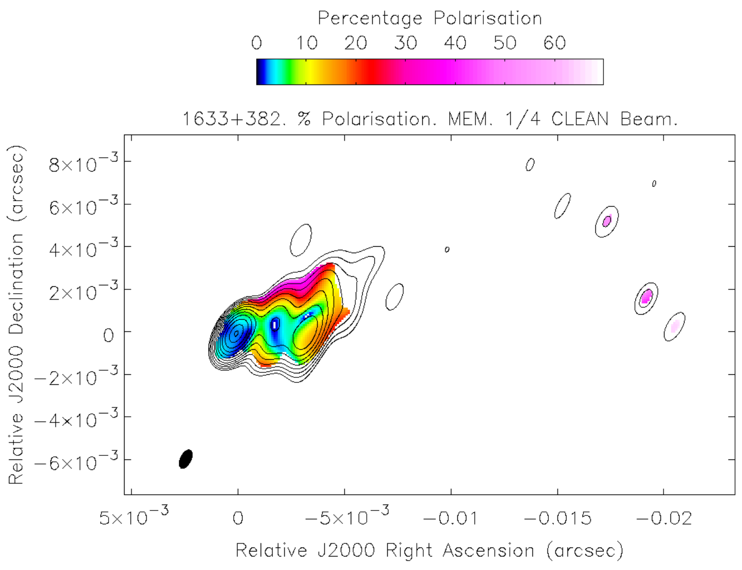


(a) MEM. $\frac{1}{3}$ CLEAN Beam

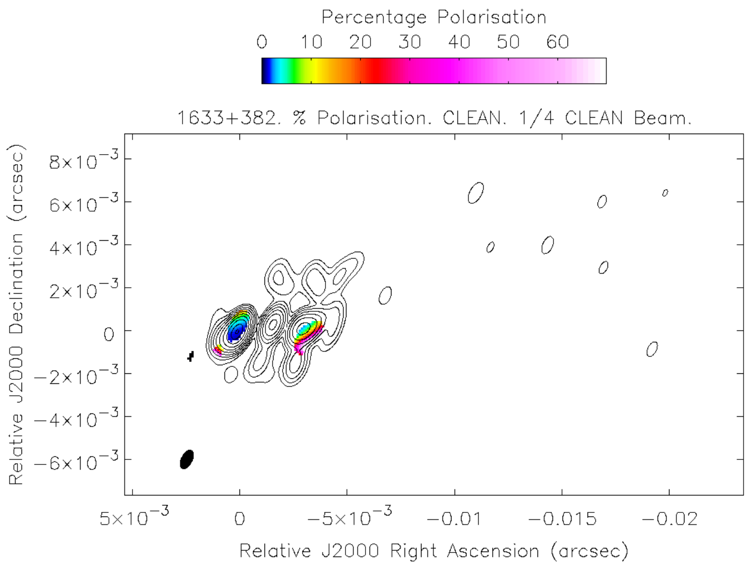


(b) CLEAN. $\frac{1}{3}$ CLEAN Beam

Figure 4.27: Percentage polarisation maps of 1633+382 made with the MEM and CLEAN convolved with $\frac{1}{3}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the percentage polarisation.

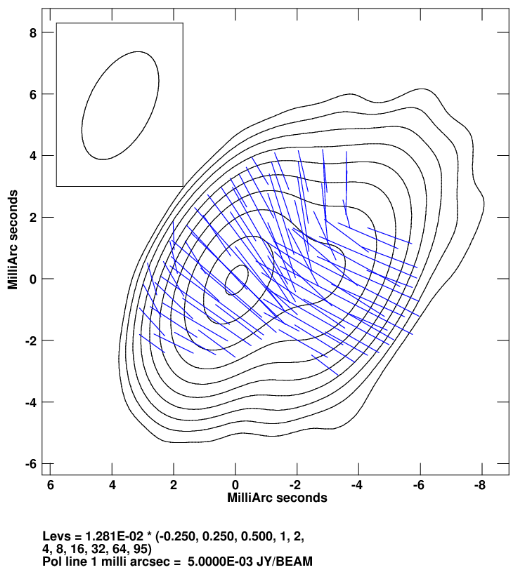


(a) MEM. $\frac{1}{4}$ CLEAN Beam

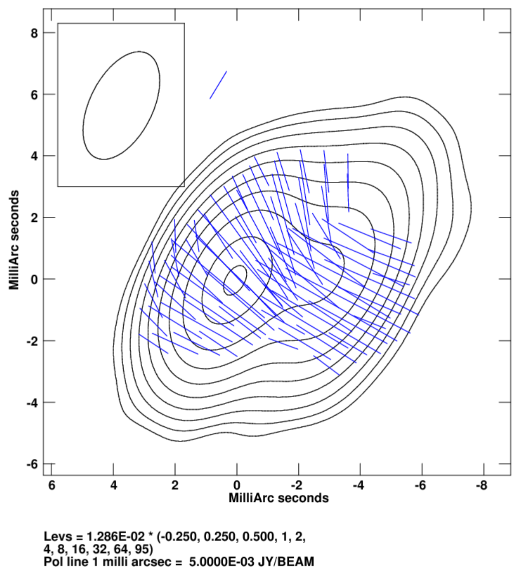


(b) CLEAN. $\frac{1}{4}$ CLEAN Beam

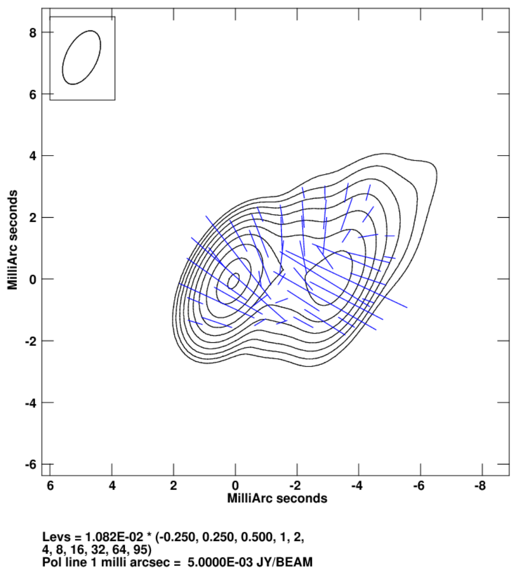
Figure 4.28: Percentage polarisation maps of 1633+382 made with the MEM and CLEAN convolved with $\frac{1}{4}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the percentage polarisation.



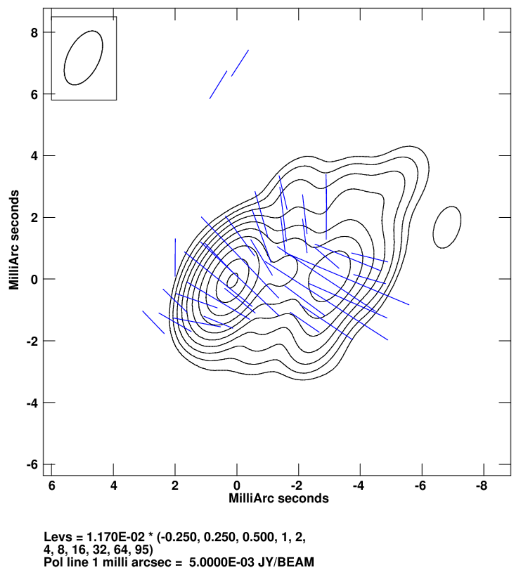
(a) MEM. CLEAN Beam.



(b) CLEAN. CLEAN Beam.

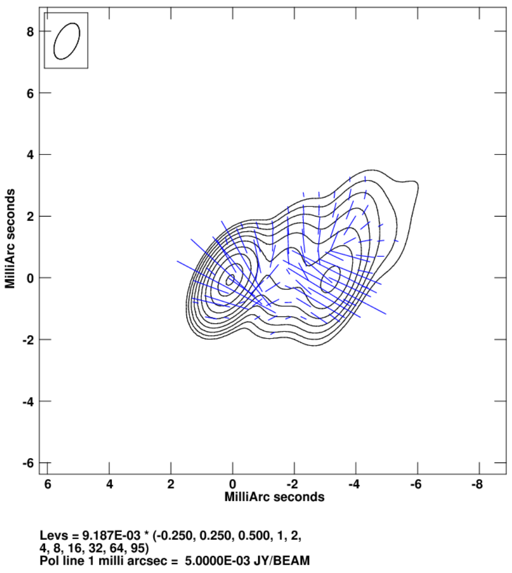


(c) MEM. $\frac{1}{2}$ CLEAN Beam.

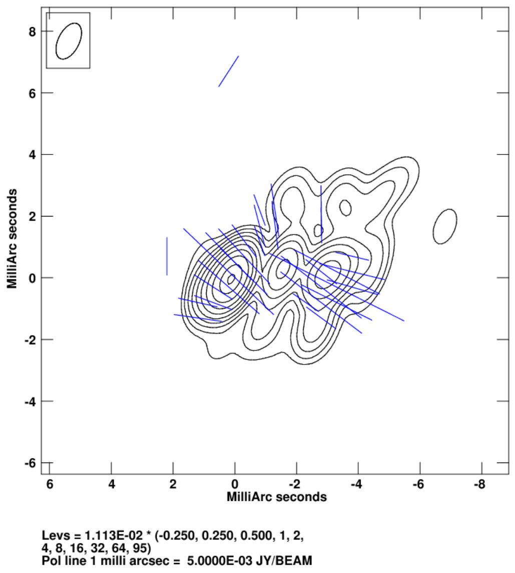


(d) CLEAN. $\frac{1}{2}$ CLEAN Beam.

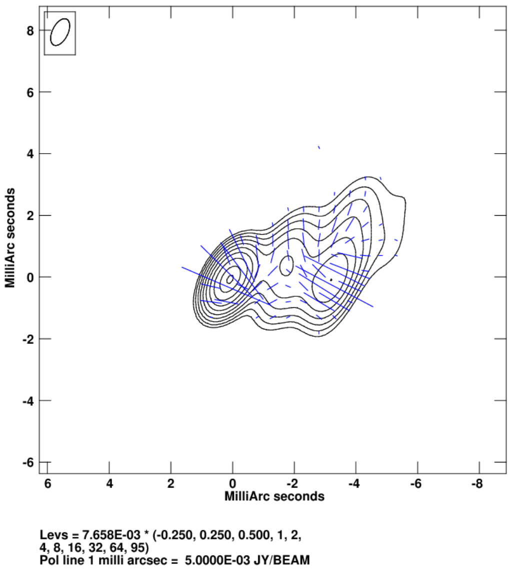
Figure 4.29: MEM and CLEAN polarisation maps of of 1633+382 at 4.612 GHz. The maps have been made with the full CLEAN beam and $\frac{1}{2}$ of the CLEAN beam. The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.



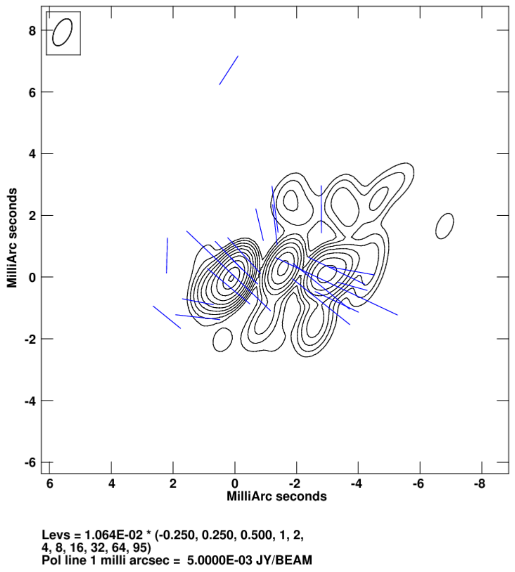
(a) MEM. $\frac{1}{3}$ CLEAN Beam.



(b) CLEAN. $\frac{1}{3}$ CLEAN Beam.



(c) MEM. $\frac{1}{4}$ CLEAN Beam.



(d) CLEAN. $\frac{1}{4}$ CLEAN Beam.

Figure 4.30: MEM and CLEAN polarisation maps of of 1633+382 at 4.612 GHz. The maps have been made with $\frac{1}{3}$ and $\frac{1}{4}$ of the CLEAN beam. The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.

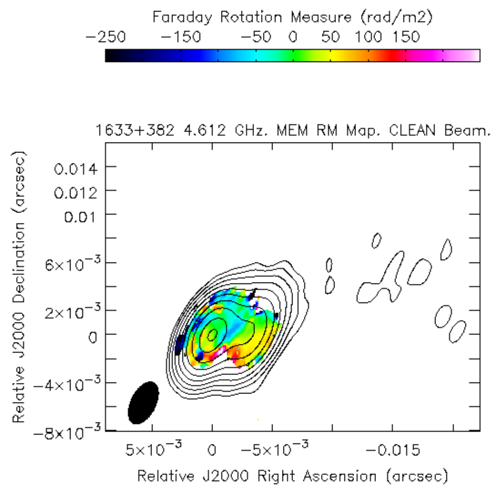
major artefacts appear in the CLEAN maps in both Stokes I intensity and in RM measure. The CLEAN map at the resolution corresponding to $\frac{1}{4}$ of the CLEAN beam is completely dominated by artefacts.

In contrast, the MEM maps smoothly transfer from the lower to the higher resolutions. Though fewer pixels show successful RM fits as the resolution increases (i.e. the beam size lowers), those that are fitted are in agreement with both the CLEAN and the MEM maps at normal CLEAN resolution. The map convolved with $\frac{1}{2}$ of the CLEAN beam is probably the best compromise between resolution and quality of the RM fit. All of the MEM maps shown seem to indicate a Faraday-rotation gradient across the core region, which is also visible in the CLEAN map convolved with the full CLEAN beam, but not in the CLEAN maps convolved with the smaller beams.

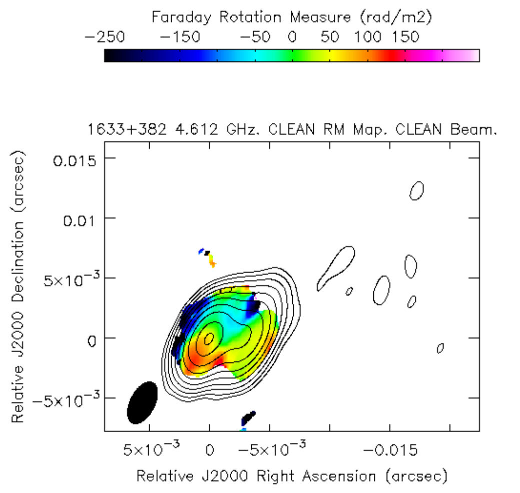
Figures 4.33 and 4.34 show intrinsic polarisation angle maps of 1633+382 at 4.6 GHz made with CLEAN and the final MEM maps, restored with the beams listed in Table 4.6. The intrinsic polarisation angle was calculated in the same linear fit as the Faraday rotation measure, as was described in Section 4.1 above. There is good agreement between all the MEM intrinsic-polarisation images; this structure is sometimes also visible in the corresponding CLEAN images, but the unreliability of the intensity structure shown in the CLEAN images made with beams smaller than $\frac{1}{2}$ the CLEAN beam would cast the reality of this polarisation structure into doubt and make it hard to interpret. The orientation of the polarisation relative to the jet direction roughly 4 mas from the core is not obvious in the CLEAN images; however, the MEM images suggest that this is polarisation along one edge of the jet where it bends toward the north, particularly given the patterns shown by the MEM degree of polarisation images.

4.3 0716+714

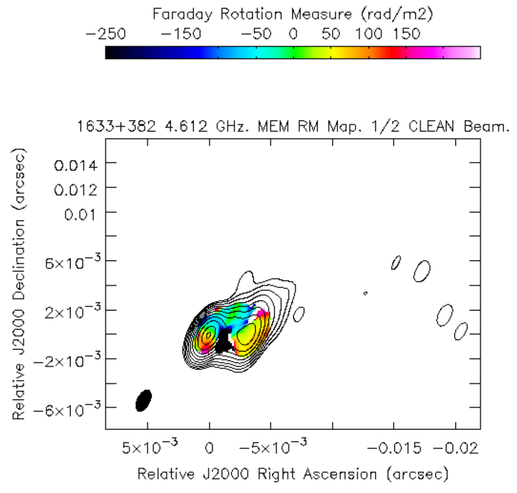
0716+714 (J0721+7120) is a BL Lac object, whose distance is not known due to the lack of a redshift estimate for its spectrum (Lister et al. 2009). Mahmud et al. (2013) published a detection of a transverse gradient in rotation measure that appeared to change direction along the jet (see also Chapter 5). It is of note that Mahmud et al. (2013) used a restoring beam equal to about 60% of the CLEAN beam corresponding to the lowest frequency observed to achieve their results. Although this would be considered unjustified in a traditional approach, the conclusions from my Monte Carlo simulations in Chapter 3 indicate that



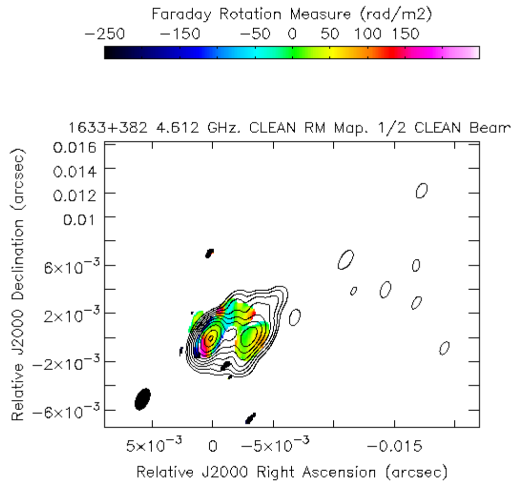
(a) MEM. CLEAN Beam



(b) CLEAN. CLEAN Beam

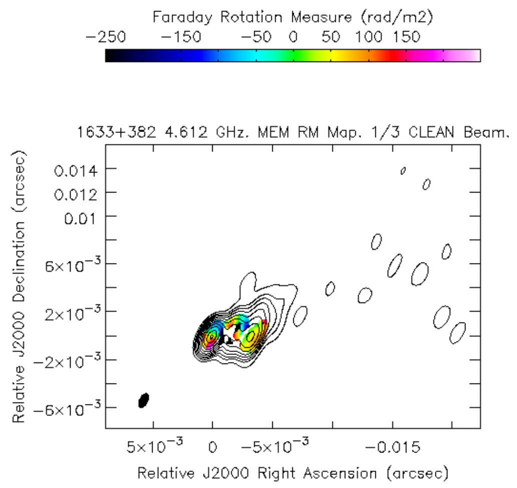


(c) MEM. $\frac{1}{2}$ CLEAN Beam

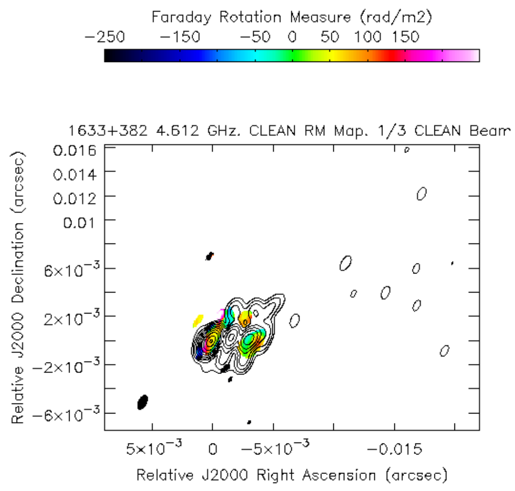


(d) CLEAN. $\frac{1}{2}$ CLEAN Beam

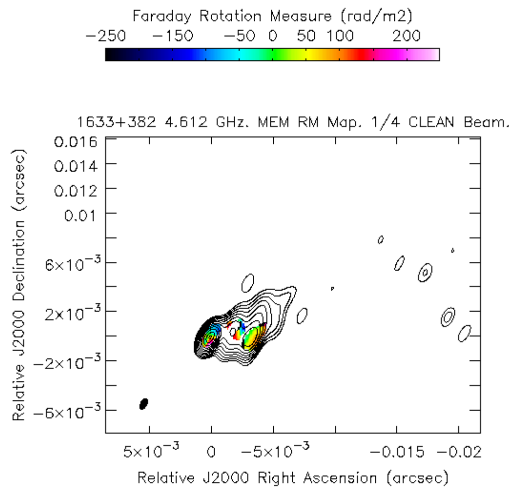
Figure 4.31: Faraday rotation measure maps of 1633+382 made with the MEM and CLEAN convolved with the full CLEAN beam and with $\frac{1}{2}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the Faraday rotation measure.



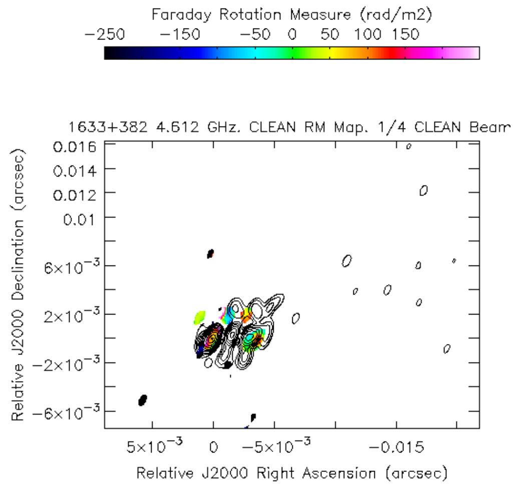
(a) MEM. $\frac{1}{3}$ CLEAN Beam



(b) CLEAN. $\frac{1}{3}$ CLEAN Beam

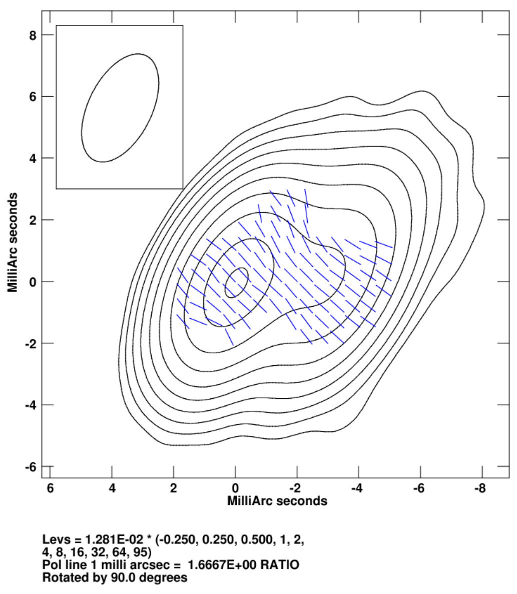


(c) MEM. $\frac{1}{4}$ CLEAN Beam

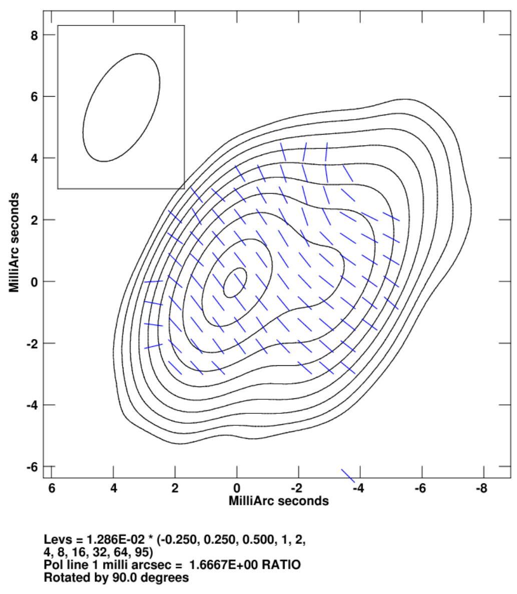


(d) CLEAN. $\frac{1}{4}$ CLEAN Beam

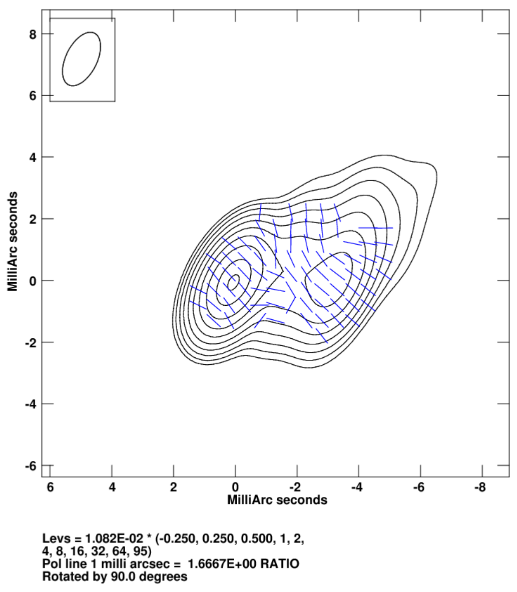
Figure 4.32: Faraday rotation measure maps of 1633+382 made with the MEM and CLEAN convolved with $\frac{1}{3}$ and $\frac{1}{4}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the Faraday rotation measure.



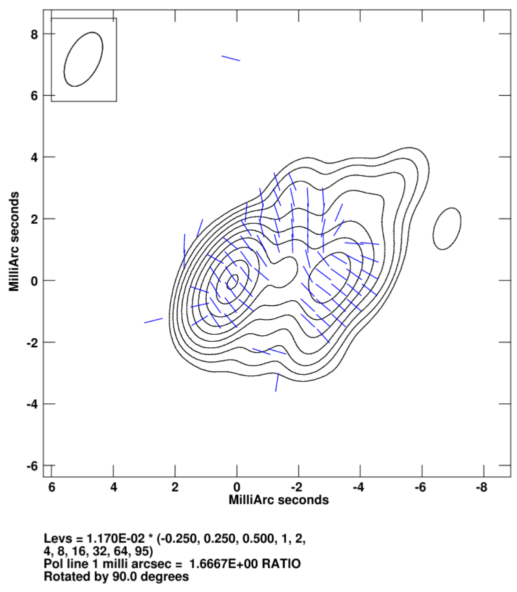
(a) MEM. CLEAN Beam



(b) CLEAN. CLEAN Beam

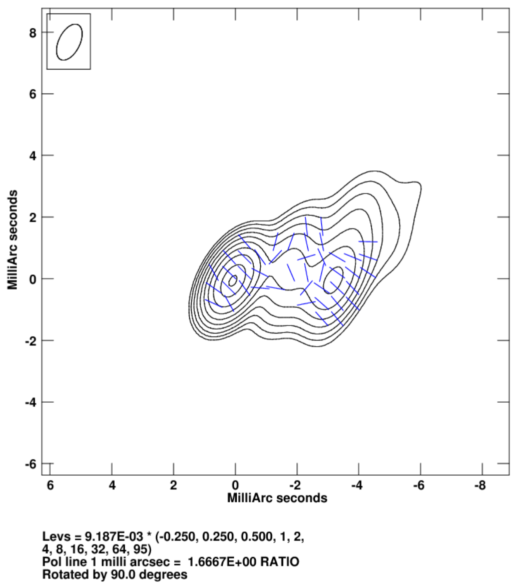


(c) MEM. $\frac{1}{2}$ CLEAN Beam

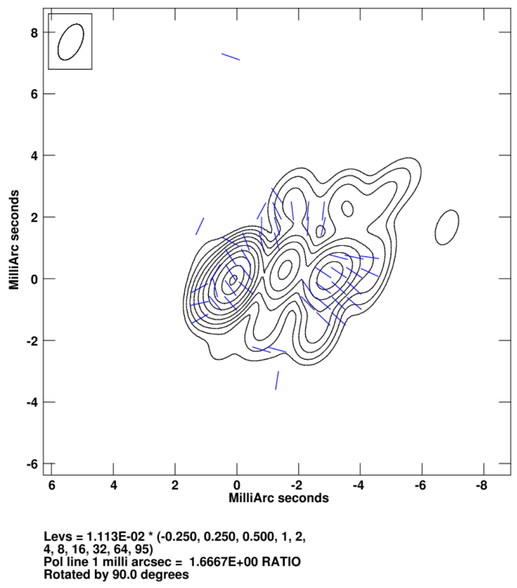


(d) CLEAN. $\frac{1}{2}$ CLEAN Beam

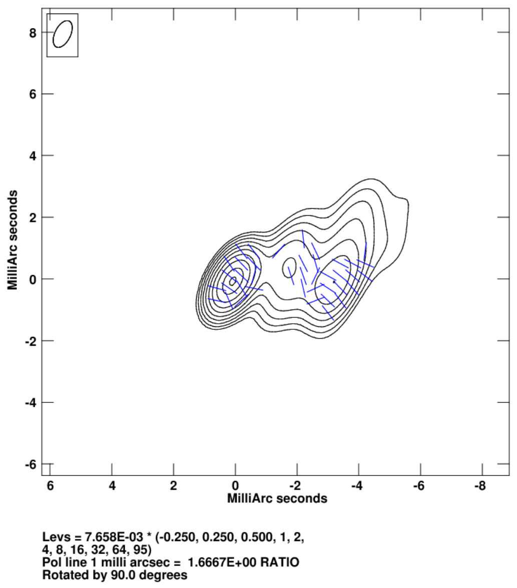
Figure 4.33: Intrinsic polarisation maps of 1633+382 made with the MEM and CLEAN convolved with the full CLEAN beam and with $\frac{1}{2}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The blue dashes indicate the direction of the intrinsic polarisation angle.



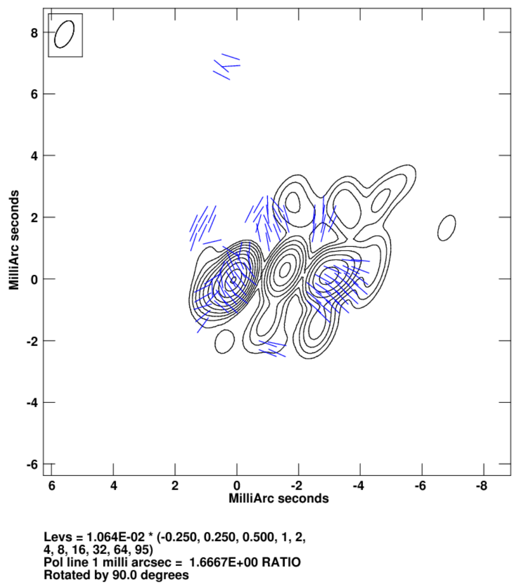
(a) MEM. $\frac{1}{3}$ CLEAN Beam



(b) CLEAN. $\frac{1}{3}$ CLEAN Beam



(c) MEM. $\frac{1}{4}$ CLEAN Beam



(d) CLEAN. $\frac{1}{4}$ CLEAN Beam

Figure 4.34: Intrinsic polarisation maps of 1633+382 made with the MEM and CLEAN convolved with $\frac{1}{3}$ and $\frac{1}{4}$ of the CLEAN beam. The contours indicate the Stokes I intensity at 4.612 GHz. The blue dashes indicate the direction of the intrinsic polarisation angle.

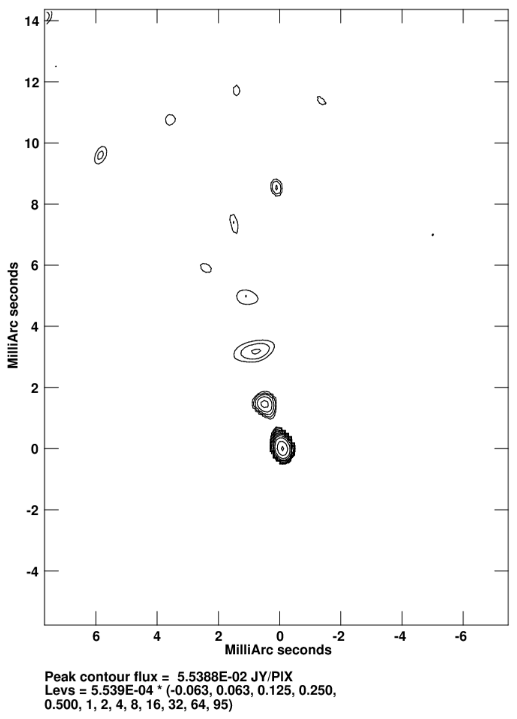
reliable results can be obtained with such modestly “super-resolved” CLEAN maps. The same VLBA data as used by Mahmud et al. (2013) at 6 frequencies between 4.612 and 15.383 GHz were used to create MEM based polarisation and Faraday rotation measure maps of 0716+714, in order to investigate the difference between CLEAN and MEM-based images of this object. The full CLEAN beam is used, as well as the CLEAN beam from Mahmud et al. (2013) and a beam slightly smaller than that beam (Table 4.8).

Table 4.7 gives the PMEM parameters used to image 0716+714 at all 6 frequencies used. The resulting MEM images corresponding to 4.6 GHz are visible in Figures 4.35 to 4.37. The model, residual, convolved and final MEM maps are present for each of the three Stokes parameters I, Q and U. The Stokes I model map captures much of the emission from the extended jet, though the residual map shows errors in the modelling of the core component – specifically a peak that is too low, and an extended core region that is too high. This kind of systematic error is systematic of the MEM algorithm as it struggles to put large amounts of flux into smaller regions. The deconvolution approach using both CLEAN and MEM described in Section 3.4 may improve the appearance of the final residual map and result in a better overall deconvolution, though thorough testing of such an approach would be necessary before any conclusions could be drawn.

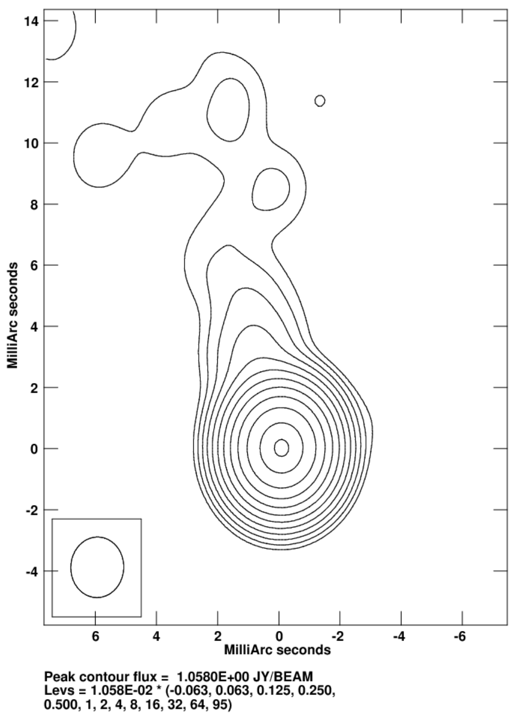
The Stokes Q and U maps also show signs of incomplete convergence as relatively large residuals are visible. However as the residuals are added to the convolved map to create the final map, the final MEM images do include any data that may be missed by the algorithm. Again, MEM appears to be imaging the extended regions with a greater success than the point-like core region.

Figure 4.38 shows percentage polarisation maps of 0716+714 at the full CLEAN beam and at the resolution used in Mahmud et al. The MEM and CLEAN maps show complete agreement, though the MEM maps do show statistically significant polarised flux further out along the jet than is visible in the CLEAN maps. This region appears to have significantly higher percentage polarisation than in the core region, as is a common feature for AGN jets.

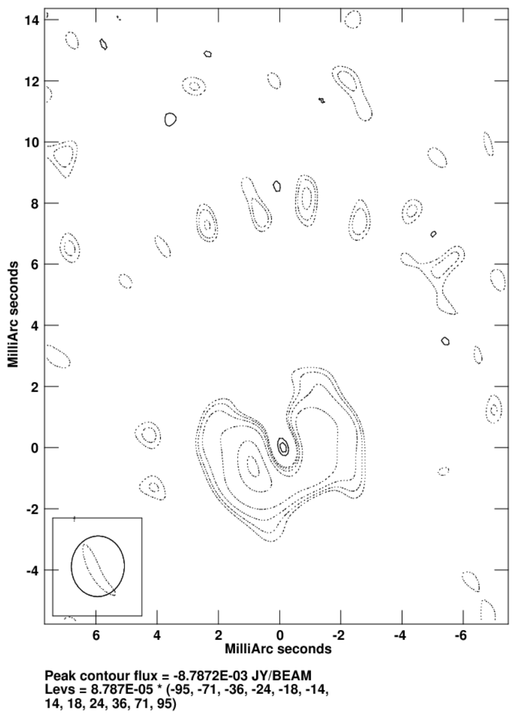
Figure 4.39 continues the same trend for a beam corresponding to $\frac{3}{4}$ of the Mahmud et al. beam, or about 44% of the full CLEAN beam. The MEM continues to pick up a region of high percentage polarisation further out along the jet not visible in the CLEAN map. The difference in scale between the two maps in this case necessitates a change in the colour scale used in the two images. The major difference between MEM and CLEAN based percentage polarisation images in



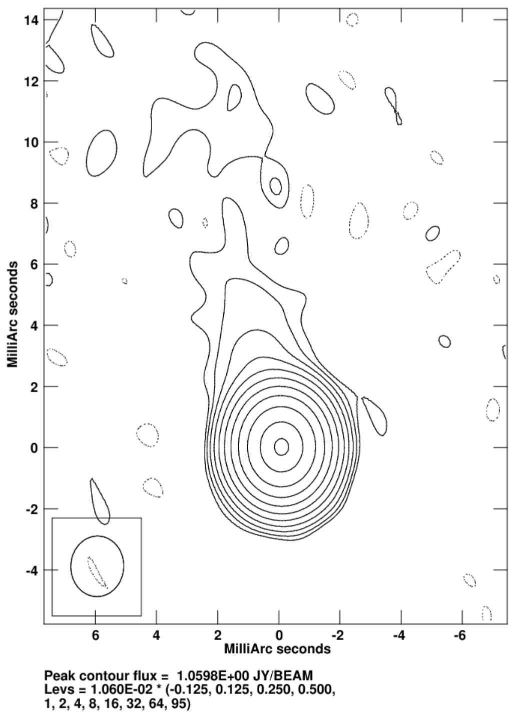
(a) Model Map



(b) Convolved Model Map

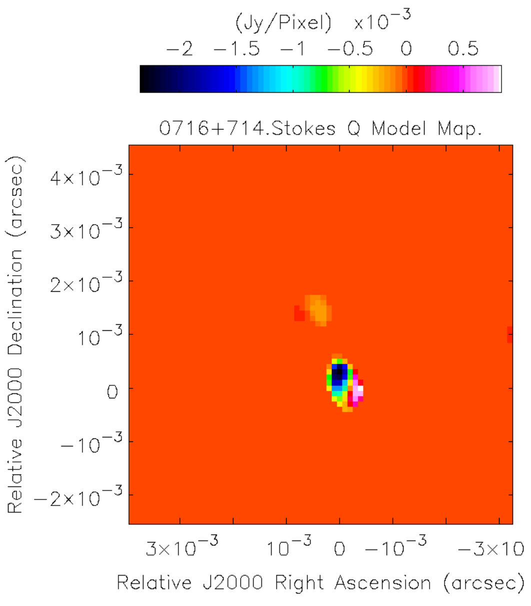


(c) Residual Map

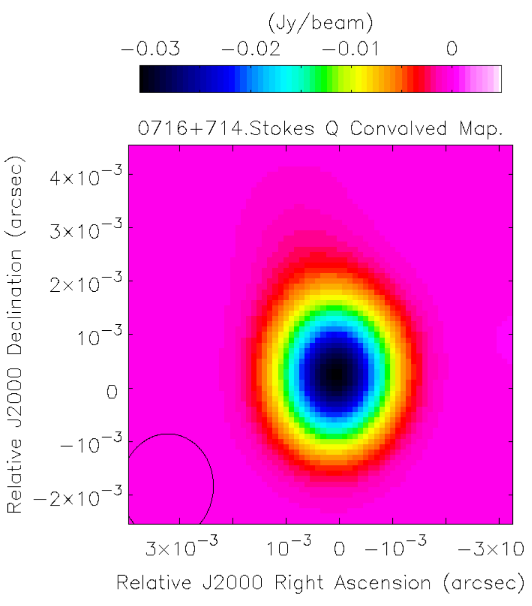


(d) Final MEM Map

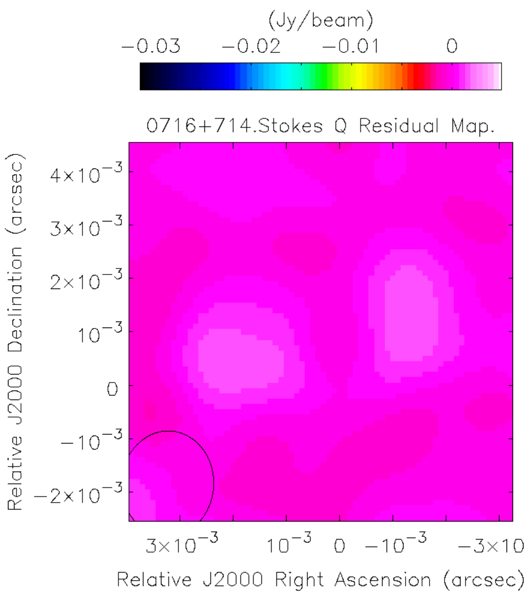
Figure 4.35: Stokes *I* MEM maps of 0716+714 at 4.612 GHz. The Convolved maps shown have been convolved with the CLEAN beam for the observation.



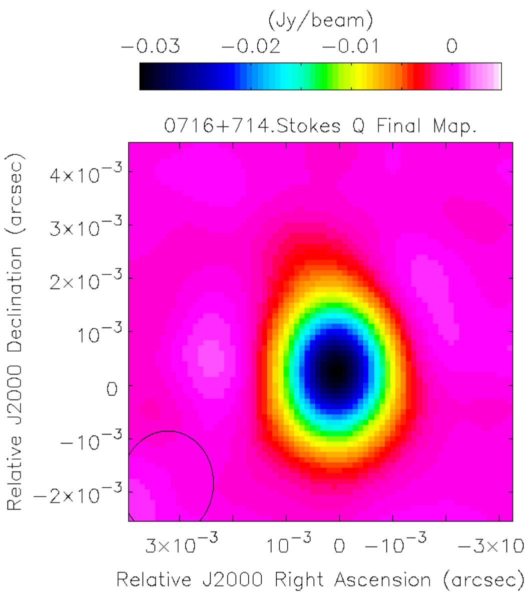
(a) Model Map



(b) Convolved Model Map

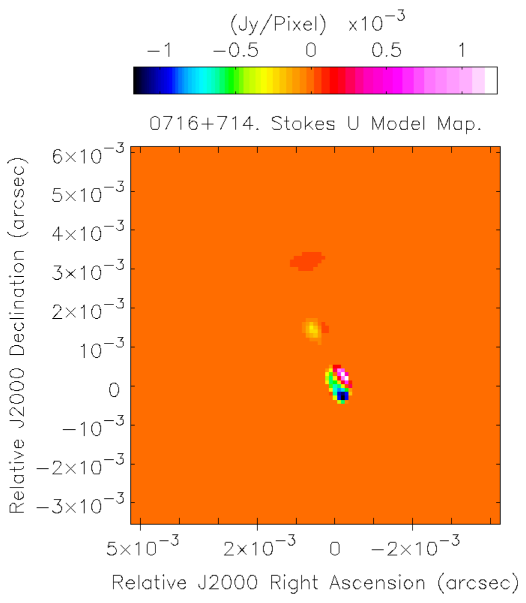


(c) Residual Map

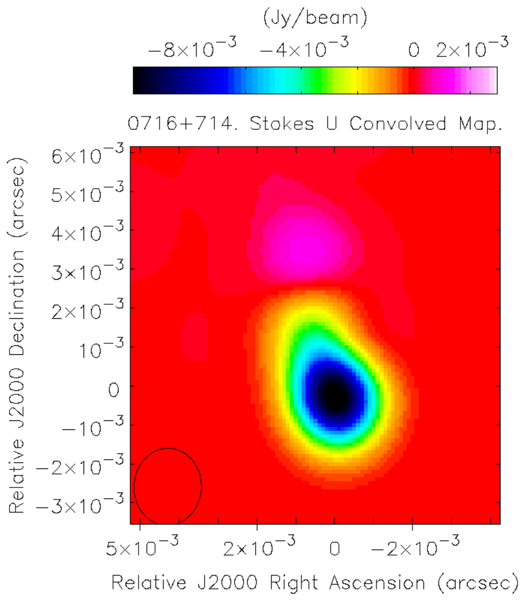


(d) Final MEM Map

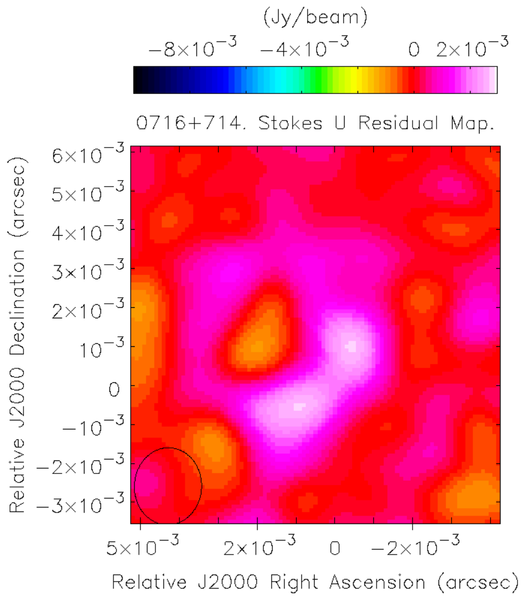
Figure 4.36: Stokes Q MEM maps of 0716+714 at 4.612 GHz. The colour scale indicates intensity in Jy/Pixel or Jy/Beam. The Convolved maps shown have been convolved with the CLEAN beam for the observation.



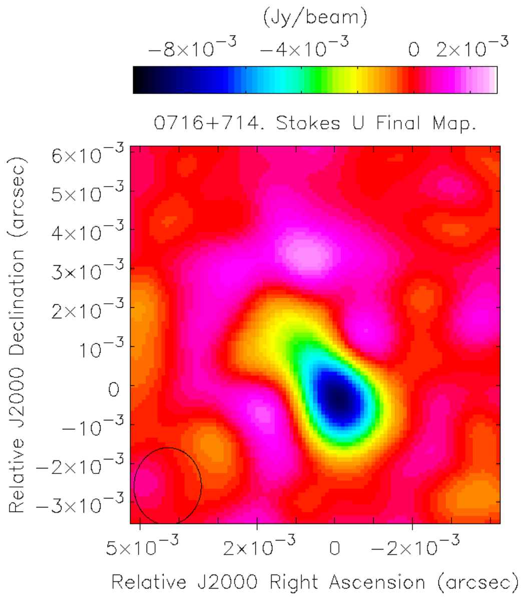
(a) Model Map



(b) Convolved Model Map



(c) Residual Map



(d) Final MEM Map

Figure 4.37: Stokes U MEM maps of 0716+714 at 4.612 GHz. The colour scale indicates intensity in Jy/Pixel or Jy/Beam. The Convolved maps shown have been convolved with the CLEAN beam for the observation.

Table 4.7: MEM parameters used to image 0716+714. In all cases the restoring beam used was the CLEAN beam, the maximum number of iterations was 10,000, a flat bias map was used and flux was conserved with parameter 2. It was found that an estimated final RMS noise of 0 for all Stokes parameters gave the best deconvolution results for the highest frequency. $N_{exclude}$ was set to 100 pixels in all cases.

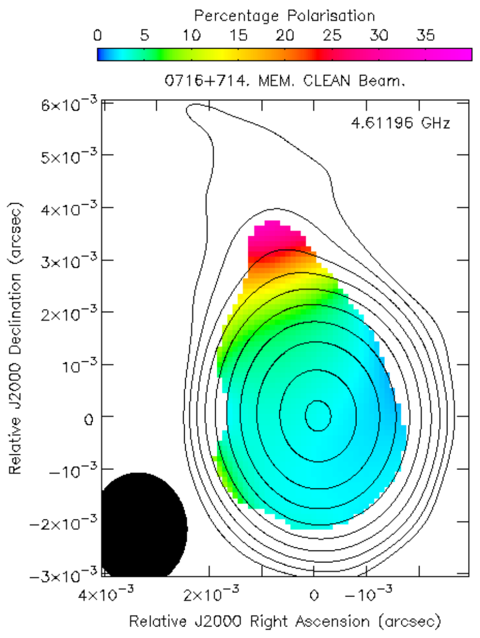
Freq. (GHz)	4.6	5.1	7.9	8.9	12.9	15.4
Stokes I Flux (Jy)	1.17	1.17	1.34	1.5	1.9	1.9
Stokes Q Flux (Jy)	0.05	0.05	0.011	0.05	0	0
Stokes U Flux (Jy)	0.05	0.05	0.011	0.05	0	0
Stokes I RMS (mJy/Beam)	0.4	0.4	0.6	0.6	0.45	0
Stokes Q RMS (mJy/Beam)	0.3	0.3	0.4	0.4	0.35	0
Stokes U RMS (mJy/Beam)	0.3	0.3	0.4	0.4	0.35	0
Δ_{step}	1.5	3	1.45	1.5	1.5	1.5
q_{factor}	0.6	0.48	0.45	0.36	0.0551	0.1
w_p	2	2	2	4	2	2

Table 4.8: The beams used to make MEM images of 0716+714. Mahmud et al. (2013) used a CLEAN beam corresponding to the 7.196 GHz map (see Chapter 5).

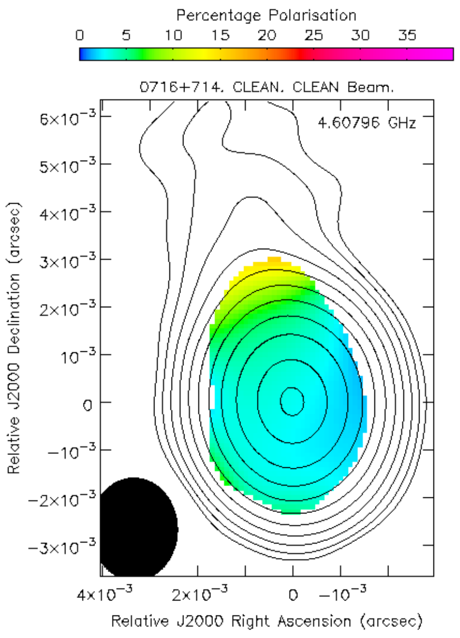
% of CLEAN beam	Major Axis (mas)	Minor Axis (mas)	Position Angle (deg.)
100%	2.15	1.82	1.5
58% (Mahmud et al.)	1.25	1.06	−0.8
44%	0.96	0.795	−0.8

0716+714 appears to be the detection of percentage polarisation much further out along the jet.

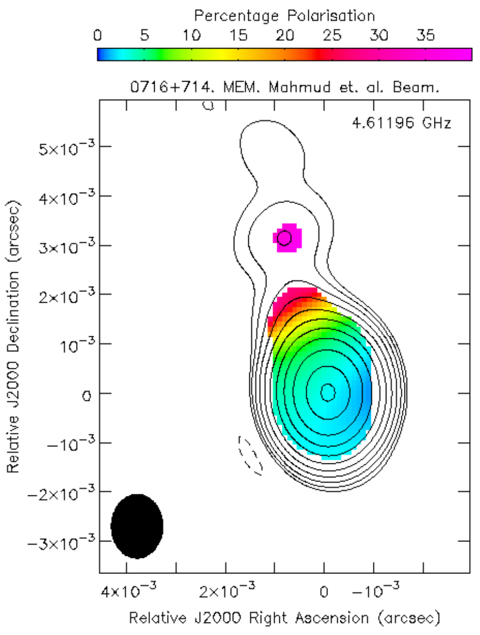
Figures 4.40 and 4.41 show the corresponding polarisation angle maps. Only polarisation angles corresponding to a total polarised flux ($P = \sqrt{Q^2 + U^2}$) high enough to rule out most off-source regions were shown. As the MEM produced total polarised flux maps with much lower noise than those produced by the CLEAN algorithm, lower cut–offs were required for the MEM maps than for the CLEAN maps. Both the MEM and CLEAN maps show identical polarisation angles at all resolutions, though the MEM maps pick up additional polarisation further out along the jet at higher resolutions. The super–resolved CLEAN maps do not show any of the artefacting seen in previous sources and agree closely with the MEM where both methods detect polarised emission. This suggests that super–resolved CLEAN can be reliable in certain sources, if little or no artefacting is observed.



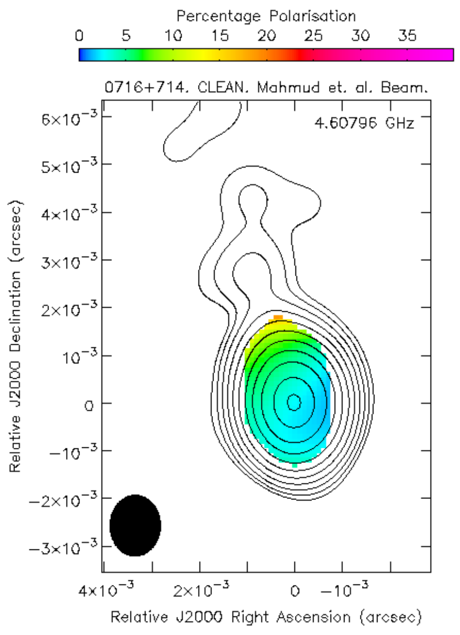
(a) MEM. CLEAN Beam



(b) CLEAN. CLEAN Beam

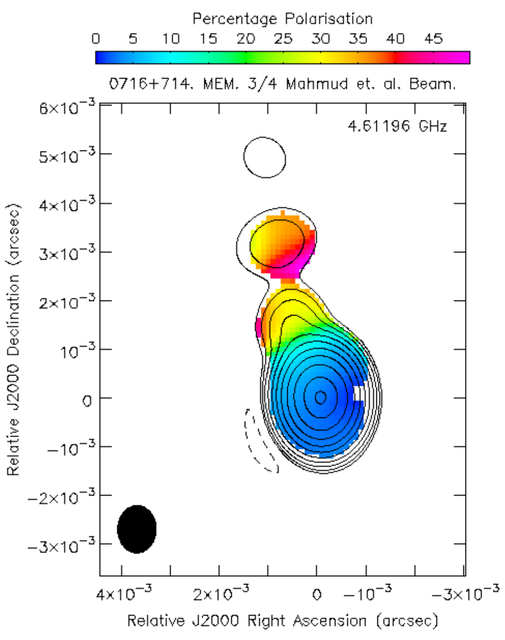


(c) MEM. Mahmud et al. Beam

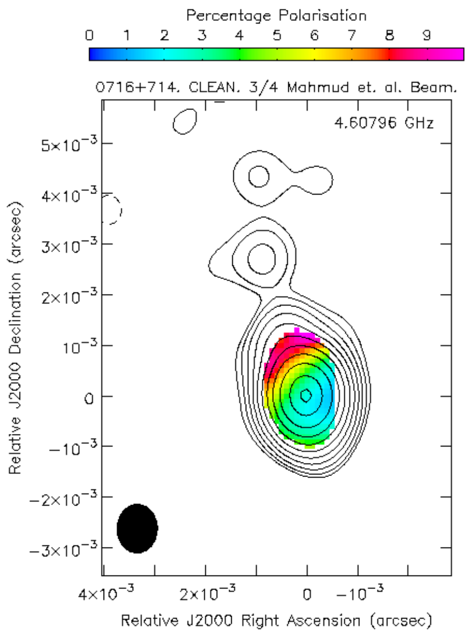


(d) CLEAN. Mahmud et al. Beam

Figure 4.38: Percentage polarisation maps of 0716+714 made with the MEM and CLEAN convolved with the full CLEAN beam and with the beam used in Mahmud et al. (2013). The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the percentage polarisation.



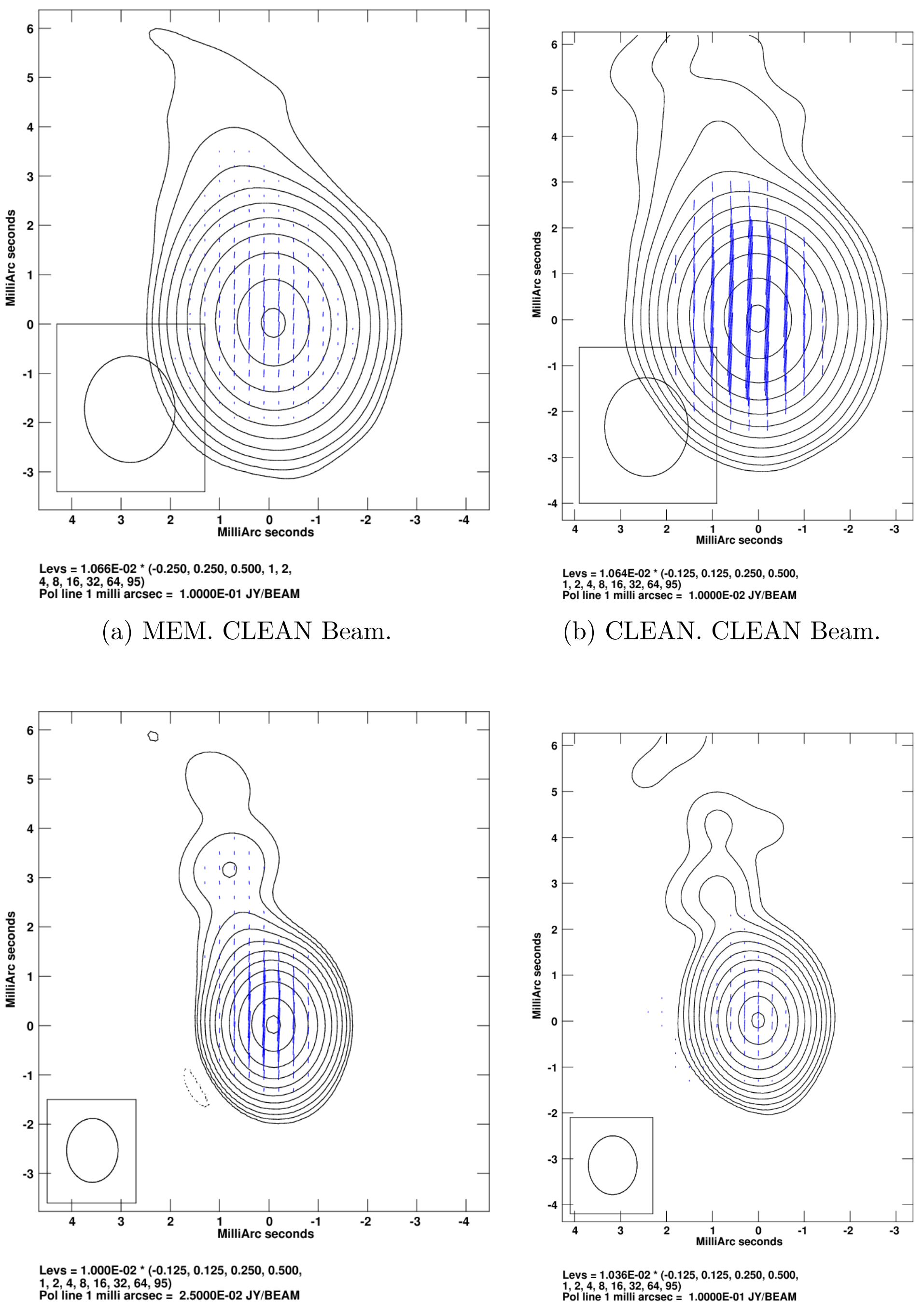
(a) MEM. $\frac{3}{4}$ Mahmud et al. Beam



(b) CLEAN. $\frac{3}{4}$ Mahmud et al. Beam

Figure 4.39: Percentage polarisation maps of 0716+714 made with the MEM and CLEAN convolved with $\frac{3}{4}$ of the beam used in Mahmud et al. (2013). The contours indicate the Stokes I intensity at 4.612 GHz. The colour scale indicates the percentage polarisation. Note that different colour scales have been used because of the large difference in the ranges presented.

Figures 4.42 and 4.43 show CLEAN and MEM Faraday rotation measure maps for 0716+716 at the three resolutions indicated in Table 4.8. The CLEAN maps at full CLEAN resolution and the beam used by Mahmud et al. (2013) both show strong evidence for the existence of transverse gradients. For a full statistical description of the change in direction of the transverse gradient along the jet see Mahmud et al. (2013), or Chapter 5. Figure 4.43 shows that the gradient reversal is still visible at a beam corresponding to 44% of the original CLEAN beam, however the increasing resolution has reduced the significant rotation measure values at the edges of the jet. This can be interpreted as the averaging effect of convolution with a larger beam causing the signal to noise ratio at the edges of the jet to rise to statistically significant levels. As there are no obvious artefacts in the map suggesting that the CLEAN algorithm has been convolved with too small a beam and the map is otherwise consistent with the lower resolution maps it it likely that this high resolution CLEAN map is a good indication of the true Faraday rotation measure at this resolution, but that a lower resolution is necessary to successfully view the gradient.



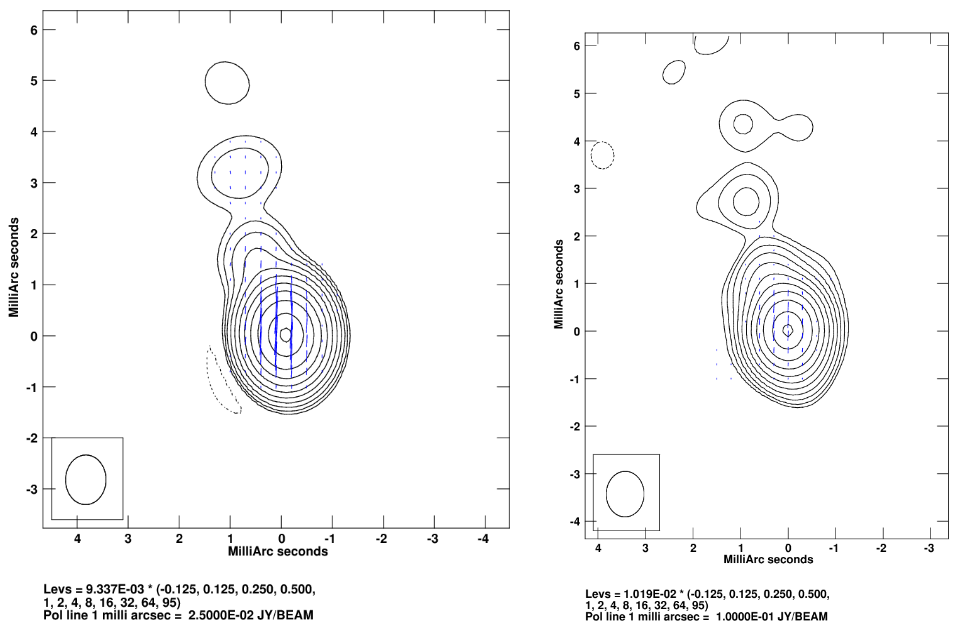
(a) MEM. CLEAN Beam.

(b) CLEAN. CLEAN Beam.

(c) MEM. Mahmud et al. Beam.

(d) CLEAN. Mahmud et al. Beam.

Figure 4.40: MEM and CLEAN polarisation maps of of 0716+714 at 4.612 GHz. The maps have been made with the full CLEAN beam and the beam used in Mahmud et al. (2013). The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.



(a) MEM. CLEAN Beam. (b) CLEAN. CLEAN Beam.

Figure 4.41: MEM and CLEAN polarisation maps of of 0716+714 at 4.612 GHz. The maps have been made with $\frac{3}{4}$ of the beam used in Mahmud et al. (2013). The blue dashes indicate the direction of the observed polarisation angle. The length of the line is proportional to the magnitude of the polarised flux. The contours are Stokes I emission.

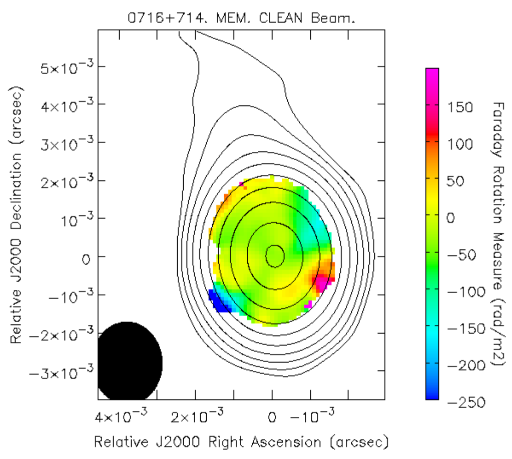
The corresponding MEM maps are not as clear. At the full CLEAN beam the gradients visible in the CLEAN map have analogues in the MEM map, however the MEM map is less smooth and the gradients much less obvious. At the beam used by Mahmud et al. (2013) the MEM sees a blob of high RM at the North of the map that appears closer to the left side of the jet in the CLEAN map. The lower gradient is tentatively visible, but at a different angle and less pronounced. The same trend is continued in the highest resolution MEM map (corresponding to about 44% of the full CLEAN beam). If one were to use only the MEM images for 0716+714 it would be difficult to see any evidence for a significant reversal in the direction of any gradients across the jet. However the detection of a blob of high rotation measure in the North of the jet at the two higher resolutions appears to coincide with the region of high percentage polarisation detected by both CLEAN and the MEM in Figures 4.38 and 4.39. This is a possible suggestion that, either the MEM values for percentage polarisation and Faraday rotation measure in this region are mistaken (though the corresponding CLEAN maps support the increase in percentage polarisation in the region), or there may be a component of polarised emission in the North of the jet that is giving rise to a gradient when convolved with the CLEAN algorithm, but is associated with a peak in Stokes I emission in high resolution MEM maps. A possible cause of such Faraday RM, percentage polarisation and Stokes I behaviour would be the presence of a pressure shock in the jet in the region.

Figures 4.44 and 4.45 show the intrinsic polarisation angle of 0716+714 as calculated by the two algorithms for each of the three resolutions. As the intrinsic polarisation angle is calculated in the same linear fit as the Faraday rotation measure, angles are only available at points where a good straight line fit was found between the polarisation angle at each frequency and the square of the corresponding wavelength. All six maps in Figures 4.44 and 4.45 show much the same behaviour. An approximately constant polarisation angle is detected in the same direction of the jet. This indicates a magnetic field orthogonal to the jet, which could be a signature of either a transverse shock or a region with a significant toroidal magnetic-field component. There is a slight change in the intrinsic polarisation angle in the MEM map in Figures 4.45 corresponding to the location of the blob in RM in Figure 4.43 and the increasing percentage polarisation detected in Figure 4.39, however this is too small too associate with the presence of any shock in the jet.

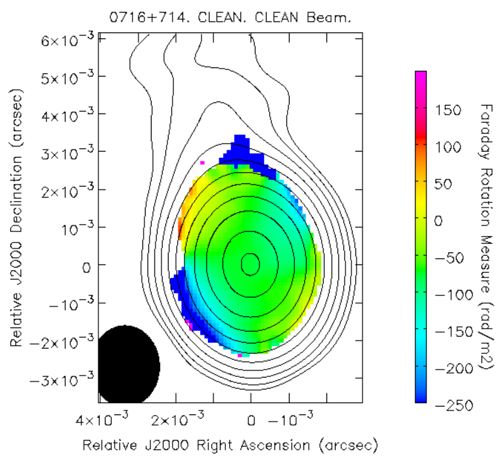
4.4 Conclusions

This chapter has presented the first results of VLBI polarisation maps of AGN deconvolved using the Maximum Entropy Method as implemented in PMEM. For comparison, high resolution images made with the CLEAN algorithm were also presented. All three sources considered showed very similar performance for the MEM and CLEAN at resolutions corresponding to the full standard CLEAN beam. However, MEM shows more detail in polarisation and percentage polarisation images than CLEAN when we consider images convolved with beams smaller than the full CLEAN beam. This is in agreement with the enhanced performance of the MEM in imaging percentage polarisation and polarisation angle noted in Chapter 3. This increased performance is likely driven by the way in which fractional polarisation is included in the Gull and Skilling entropy, which forces the MEM maps to create a model that pays very close attention to the interplay between Stokes I (which has a relatively high signal to noise ratio) and the total polarised flux (which has a much lower signal to noise ratio).

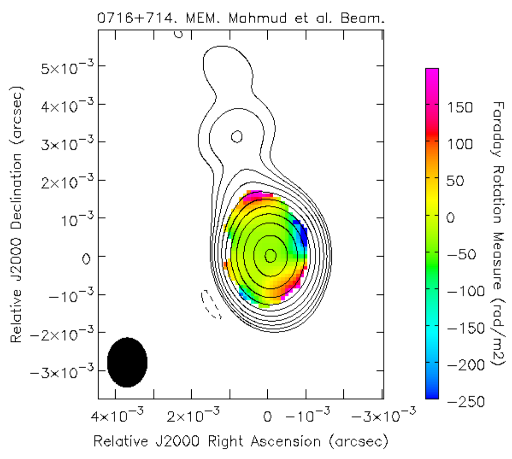
As discussed in Chapter 2, the effective resolution of the MEM is often higher



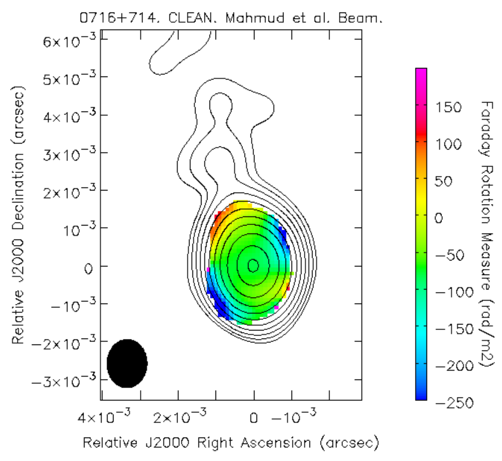
(a) MEM. CLEAN Beam



(b) CLEAN. CLEAN Beam



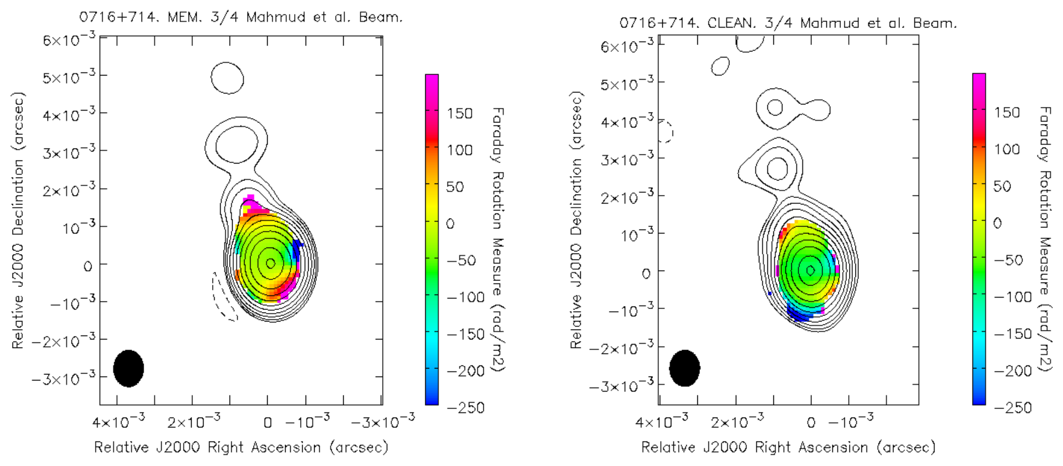
(c) MEM. Mahmud et al. Beam



(d) CLEAN. Mahmud et al. Beam

Figure 4.42: 0716+714 MEM Faraday rotation measure maps convolved with the beams referenced in Table 4.8. A CLEAN Faraday rotation measure map is also presented. The contours indicate the Stokes I intensity at 4.611 GHz. The colour scale is Faraday rotation measure in rad/m^2.

than that of the CLEAN algorithm. The high resolution MEM images presented in this chapter, convolved with fractions of the standard CLEAN beam, have in general been consistent with MEM images at other resolutions and with CLEAN images at the full CLEAN beams. Features identifiable at low resolution are imaged in greater detail and new features in Stokes I and fractional polarisation have become visible as the size of the convolving beam is lowered. These high resolution images can potentially provide a deeper insight into the mechanisms at work in AGN jets and comparing maps at different resolutions and made with different algorithms has proven to be a valuable tool.



(a) MEM. $\frac{3}{4}$ Mahmud et al. Beam

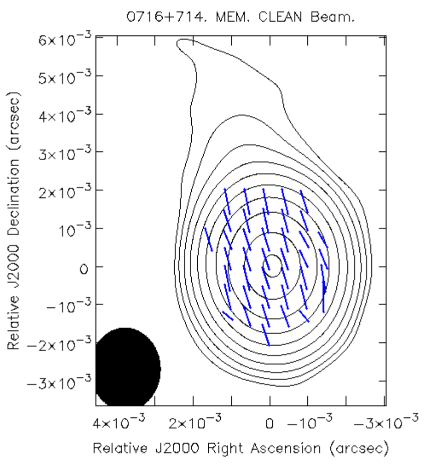
(b) CLEAN. $\frac{3}{4}$ Mahmud et al. Beam

Figure 4.43: 0716+714 MEM Faraday rotation measure maps convolved with the final beam referenced in Table 4.8. A CLEAN Faraday rotation measure map is also presented. The contours indicate the Stokes I intensity at 4.611 GHz. The colour scale is Faraday rotation measure in rad/m^2.

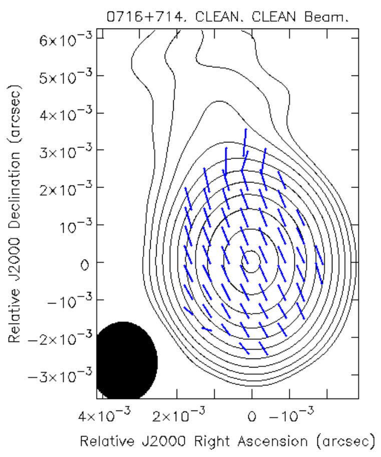
This chapter has also presented high resolution CLEAN maps, made with beams corresponding to those used for the MEM. The CLEAN algorithm has been found to be very comparable to MEM maps at resolutions up to half of the CLEAN beam, as predicted by the simulations discussed in Chapter 3. At resolutions lower than this single frequency maps suffer greatly from CLEAN artefacts, however there has also been some evidence that high resolution CLEAN maps made from multi-frequency data, such as Faraday RM maps, suffer less than a standard single frequency Stokes I or Q map. This can be explained by noting that the degree of super-resolution varies over the frequencies used, so that the lower frequencies may be super-resolved, while the middle ones are not and the higher frequencies may be drastically under-resolved.

In all three sources presented the residual maps showed that the MEM had some difficultly describing the sharply peaked core regions. This was also seen in the Monte Carlo simulations presented in Chapter 3 and is clearly a systematic feature of the MEM algorithm. The combined CLEAN and MEM approach discussed in Section 3.4 may achieve better results in these regions than MEM alone.

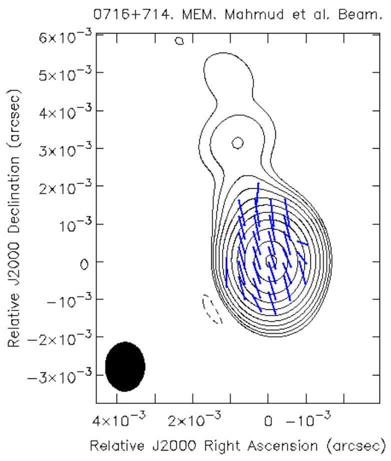
This chapter has presented the first MEM based Faraday rotation measure maps of AGN jets. As discussed for the CLEAN algorithm, Faraday RM maps are excellent candidates for convolution with a smaller than standard CLEAN beam due to the presence of information at various scales due to the various frequencies.



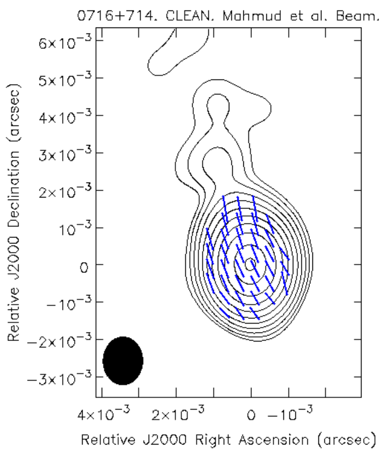
(a) MEM. CLEAN Beam



(b) CLEAN. CLEAN Beam



(c) MEM. Mahmud et al. Beam

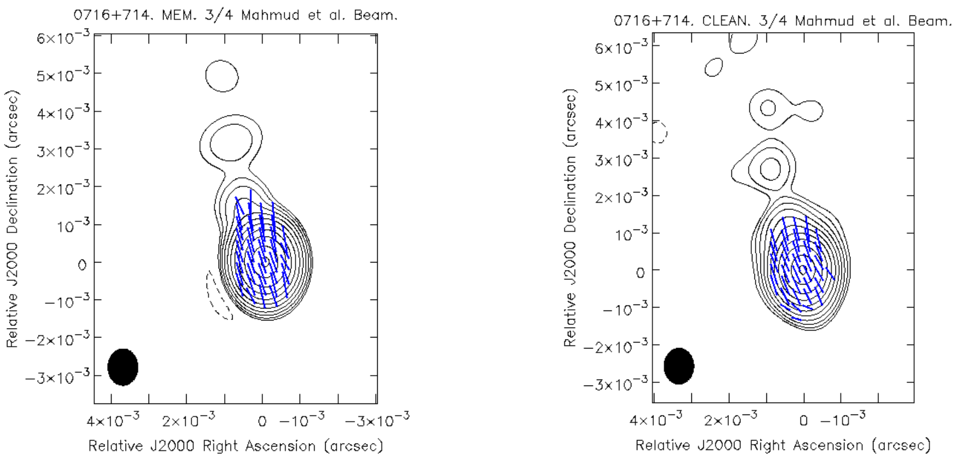


(d) CLEAN. Mahmud et al. Beam

Figure 4.44: Intrinsic polarisation angle maps of 0716+714 made with the MEM and CLEAN convolved with the full CLEAN beam and with the beam used in Mahmud et al (2013). The contours indicate the Stokes *I* intensity at 4.612 GHz. The blue ticks indicate the direction of the intrinsic polarisation angle.

The RM maps at full CLEAN resolution have overall been similar to the CLEAN RM map. At higher resolutions the amount of reliable RM values detected decreases, suggesting the ideal beam to make a RM image with will vary depending on the Faraday RM structure of the source in question. The corresponding intrinsic polarisation angle maps are in close agreement with the CLEAN maps at the full CLEAN beam, though the increased resolution has given a deeper insight into some jet behaviour.

In each of the sources considered here, the MEM polarisation images convolved



(a) MEM. $\frac{3}{4}$ Mahmud et al. Beam (b) CLEAN. $\frac{3}{4}$ Mahmud et al. Beam

Figure 4.45: Intrinsic polarisation angle maps of 0716+714 made with the MEM and CLEAN convolved with the final beam referenced in Table 4.8. The contours indicate the Stokes I intensity at 4.612 GHz. The blue ticks indicate the direction of the intrinsic polarisation angle.

with beams from $\frac{1}{2}$ to $\frac{1}{4}$ the size of the full CLEAN beam have provided new information about the core and jet polarisation that was not evident in the CLEAN images. The region of "spine–sheath" polarisation in Markarian 501 extends over a considerably greater length of the jet in these MEM images, and the rise in the degree of polarisation toward both edges of the jet is very clear; both of these are very suggestive of a helical magnetic field associated with this jet. The additional polarisation detected in the MEM maps for 1633+382 has revealed an increase in the degree of polarisation on either side of the jet beyond a bend toward the north; this bend is shown much more clearly in the MEM than the CLEAN intensity images. The MEM Faraday rotation images show a consistent pattern with a gradient of the Faraday rotation across the VLBI core region, suggestive of a helical magnetic field. The MEM maps for 0716+716 have likewise detected polarisation further from the core, showing a clear tendency for the degree of polarisation to increase along the jet, reaching values of about 35%.

Chapter 5

RM Gradient Reversals

The research presented in this chapter has been published in the Monthly Notices of the Royal Astronomical Society as Mahmud et al. (2013). Mehreen Mahmud calibrated the data and performed the original imaging while Eoin Murphy created the model jets for the Monte Carlo simulations and provided code to conduct the simulations. These simulations were then carried out, together with further imaging and error analysis.

Section 1.7 discussed the use of gradients in Faraday rotation measure (RM) as a diagnostic of the local magnetic field in a jet. The following chapter outlines the observations of such gradients in the jets of the AGN 0716+714 and 1749+701. Reversals in the direction of the gradient are observed in both jets. The chapter goes on to discuss the error analysis and simulations which suggest that the detected gradients are reliable and possible explanations for the phenomenon are presented. All the results presented in this chapter are from maps deconvolved with the CLEAN algorithm exclusively.

5.1 Faraday Rotation Measure Maps

The VLBI jet of the AGN 0716+714 emerges nearly toward the North; the optical spectrum is essentially featureless, and no redshift has been determined, so that its distance is unknown. Observations of 0716+714 were carried out on the 22nd of March 2004 at 6 frequencies: 4.612, 5.092, 7.916, 8.883, 12.939 and 15.383 GHz. It was observed for 25–30 minutes at each frequency in a ‘snap-shot’ mode with 8–10 scans spread out over the observing time period. The calibration and data reduction was performed in AIPS according to standard practice.

The consistency of the EVPA calibration performed using data from the VLA suggested an EVPA accuracy of within 3 degrees. No significant core shifts were found between the images of 0716+714 made at the 6 frequencies (see Section 1.7.1, Equation (1.100)).

The VLBI jet of 1749+701 initially emerges toward the Northwest, then curves toward the North. The redshift is $z = 0.77$. The observations for 1749+701 were taken on the 17th of January 2004 at 4 frequencies: 1.358, 1.430, 1.493 and 1.665 GHz. The data showed no core shift due to the proximity of the 4 frequencies and the consistency of the EVPA calibration suggested an accuracy of within 2 degrees.

Faraday rotation measure maps were made by creating polarisation angle maps for each of the observed frequencies with the same resolution (beam) and performing a linear fit for each pixel as described in Section 1.7. In the case of 0716+714 the beam corresponding to the 7.916 GHz frequency was chosen to provide a compromise between over-resolving the lower frequency maps, and losing the enhanced resolution of the higher frequency maps. The simulated observations presented in Chapter 3 showed that mild super-resolution of the CLEAN algorithm does not have a significant negative effect on the resulting maps. This choice was less important for 1749+701 (as the 4 frequencies used were very close together) so the resolution corresponding to the lowest frequency was used.

Figures 5.1 and 5.2 show the resulting RM maps in the central panel, with slices showing the RM gradients across the core region and inner jet of each of the two AGN. Note that higher-resolution VLBI images of 1749+701 have shown that its jet extends toward the Northwest (upper right corner of the image), although this is not clearly visible in this image due to the relatively low resolution of these observations. Figures 5.1 and 5.2 also show plots of the polarisation angle versus wavelength squared for a single pixel at the end of each slice at the top and bottom of the figure. Two transverse gradients running in opposite directions are clearly visible in each of the images. The gradients are monotonic and the sample plots of χ vs λ^2 shown at the end points of the gradients display the linear relationship characteristic of Faraday rotation. Figures 5.3 and 5.4 show the progress of the RM gradient at three points across the jets, demonstrating visually the 3 σ significance of the gradients. Table 5.1 shows the complete results for the RM gradients of both sources.

The observation of the gradients at a significance of over 3σ in each case indicates that the gradients are real, i.e., not spurious gradients associated with noise in the

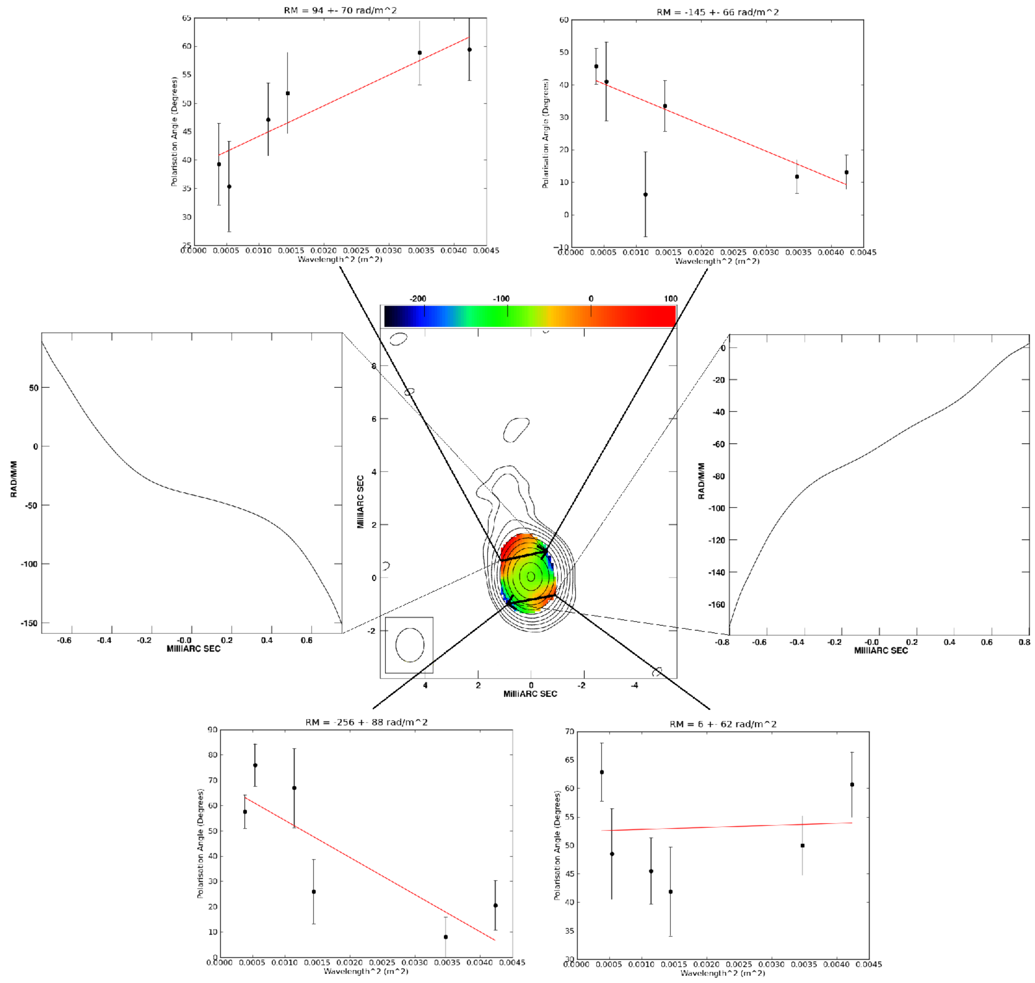


Figure 5.1: RM map of 0716+714 at 6 frequencies between 4.612 and 15.383 GHz. The colour scale is Faraday rotation measure in rad/m^2 and the contours are the Stokes I intensity at 4.612 GHz. The accompanying panels show slices of the RM distribution across the core, and polarisation angle (χ) vs. wavelength-squared (λ^2) plots for pixels on either side of the core and jet. Errors shown are 1σ, and include the estimated random errors and the EVPA uncertainties added in quadrature. The peak of the I map is 1.3 Jy/beam and the bottom contour is 1.0 mJy/beam. The beam used to construct the I and RM maps was 1.28 X 1.06 mas in position angle $-0.8°$.

data or the particular $u - v$ coverage used to make the image. The error analysis involved in calculating this statistical significance is quite involved and a full description is given in the following section. Monte Carlo simulations supporting this analysis are detailed in Section 5.3.

It is notable that a similar result was obtained for 0716+714 by Healy (2014), where the direction of the transverse RM gradient in 0716+714 on somewhat

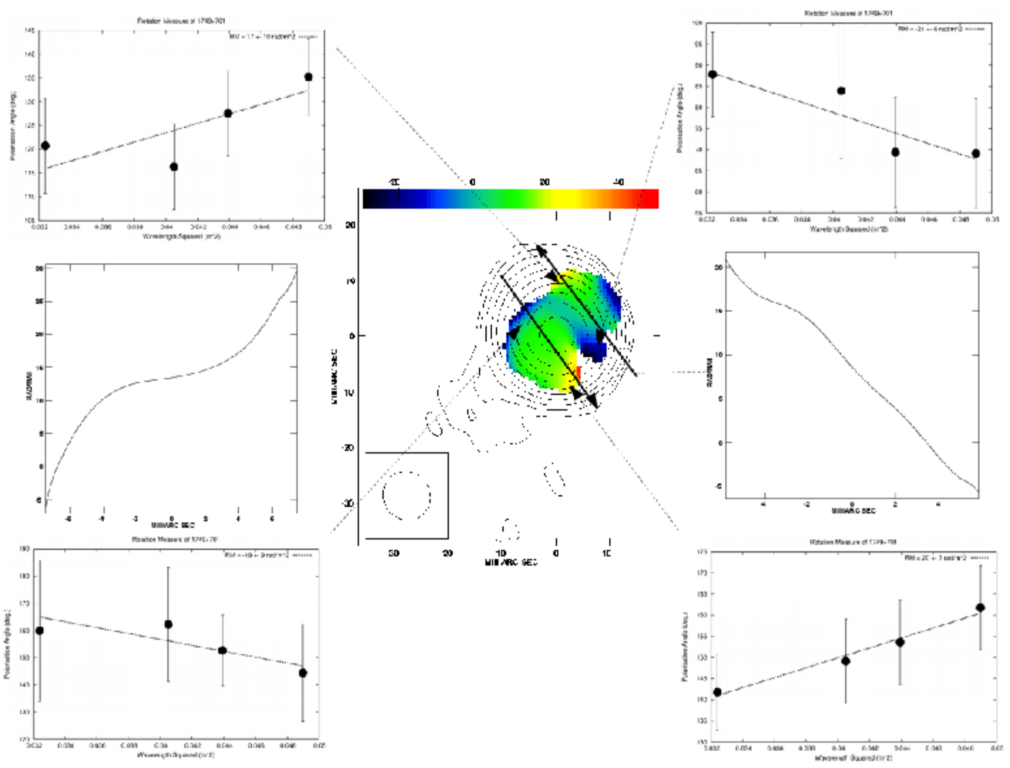
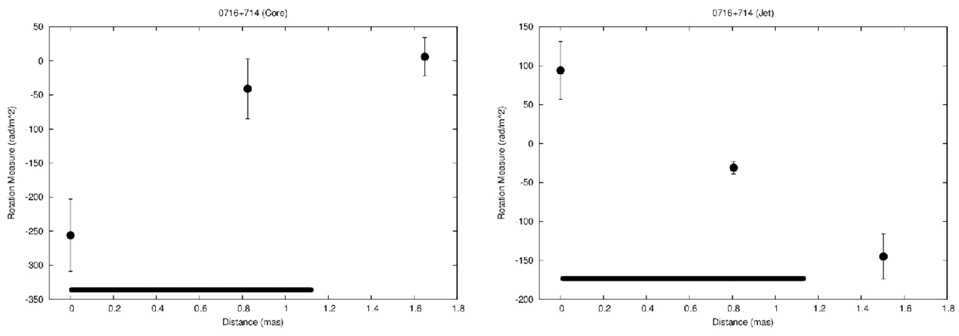
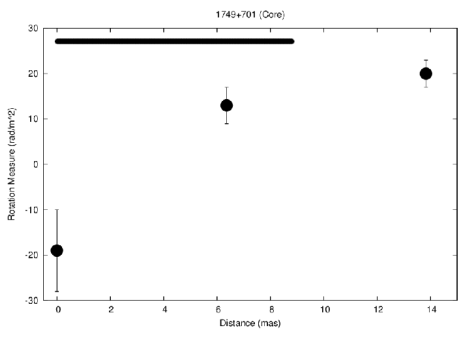


Figure 5.2: RM map of 1749+701 1.36-1.66 GHz. The accompanying panels show slices of the RM distribution across the core, and polarisation angle (χ) vs. wavelength-squared (λ^2) plots for pixels on either side of the core and jet. Errors shown are 1σ, and include the estimated random errors and the EVPA uncertainties added in quadrature. The peak of the I map is 0.6 Jy/beam; the bottom contour is 1.4 mJy/beam (January 2004).The beam used to construct the I and RM maps was 9.16 X 8.57 mas in position angle 49°.

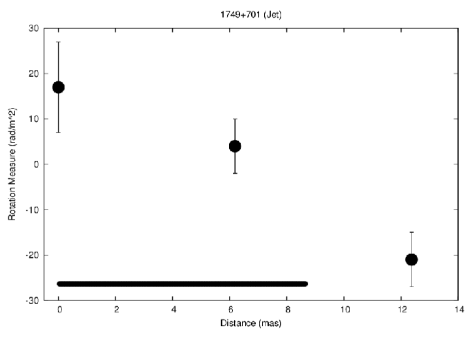


(a) Core Region. (b) Jet.

Figure 5.3: Plots of observed RM as a function of distance from a reference point on one side of the source structure across the core-region (left) and jet (right) RM distributions of 0716+714 at 4.6-15.4 GHz. The positions of each point and the corresponding RM values and their errors are listed in Table 5.1. The horizontal bar shows the approximate size of the beam FWHM.



(a) Core Region.



(b) Jet.

Figure 5.4: Plots of observed RM as a function of distance from a reference point on one side of the source structure across the core-region (left) and jet (right) RM distributions of 1749+701 at 1.36-1.66 GHz. The positions of each point and the corresponding RM values and their errors are listed in Table 5.1. The horizontal bar shows the approximate size of the beam FWHM.

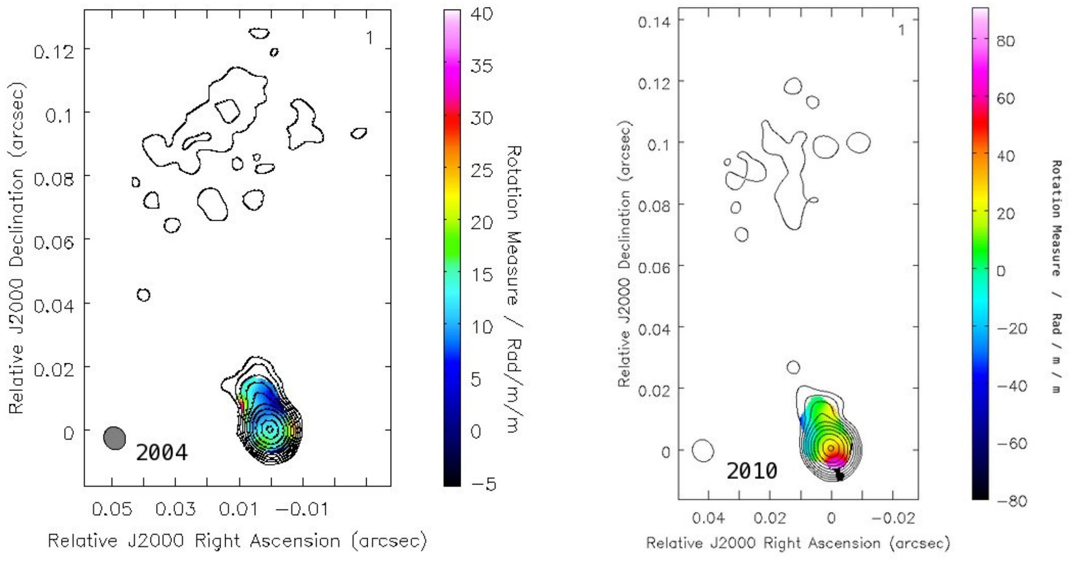

2004
Relative J2000 Declination (arcsec)
Relative J2000 Right Ascension (arcsec)
Rotation Measure / Rad/m/m
2010
Rotation Measure / Rad / m / m

Figure 5.5: Faraday rotation measure maps of 0716+714 from Healy (2014). The 2004 image is from Hallahan and Gabuzda (2008). The change in the direction of the Faraday rotation measure gradient over the six years was found to be statistically significant.

larger scales was observed to change direction over time. This can be seen in Figure 5.5 where both the gradients in the 2004 and 2010 data were found to be statistically significant with significances of 4σ and 7σ respectively. Possible physical origins of such RM gradients that reverse in space or in time are considered in the final section of this Chapter.

Table 5.1: RM measurements in 0716+714 and 1749+701.

Figure	Source	Point in plot	Position (mas)	RM rad/m^2	Left–Right RM Diff	Diff in σ
(1)	(2)	(3)	(4)	(5)	(6)	(7)
5.1	0716+714	Left	$(+0.90, -1.00)$	-256 ± 53		
(left)	(Core)	Mid	$(0.10, -0.80)$	-41 ± 44	$+262 \pm 60$	4.4σ
		Right	$(-0.70, -0.60)$	$+6 \pm 28$		
5.1	0716+714	Left	$(+1.00, +1.00)$	$+94 \pm 37$		
(right)	(Jet)	Mid	$(+0.30, +1.00)$	-31 ± 9	-239 ± 47	5.1σ
		Right	$(-0.50, +1.10)$	-145 ± 29		
5.2	1749+701	Left	$(+9.00, +1.50)$	-19 ± 8		
(left)	(Core)	Middle	$(+4.50, -3.00)$	$+13 \pm 4$	$+39 \pm 9$	4.3σ
		Right	$(-1.50, -7.50)$	$+20 \pm 2$		
5.2	1749+701	Left	$(-4.50, +9.00)$	$+17 \pm 10$		
(right)	(Jet)	Middle	$(-6.00, +3.00)$	$+4 \pm 6$	-38 ± 12	3.2σ
		Right	$(-7.50, -3.00)$	-21 ± 6		

5.2 Estimation of the Uncertainties in χ and RM

The statistical significance of an observed Faraday RM gradient depends crucially on the uncertainties in the observed Faraday rotation measures, which, in turn, depend on the uncertainties in the observed polarisation angles χ at the frequencies used to derive the RM. This section considers estimates of the uncertainties in the χ and RM values based on recent Monte Carlo simulations by Hovatta et al. (2012).

While it was standard practice to adopt the root-mean-square (rms) deviations in the residual map (or in the final CLEAN map far from any regions containing real flux) σ_{rms} as an estimate of the total uncertainty in the measured flux in an individual pixel, Hovatta et al. (2012) investigated this practice empirically using Monte Carlo simulations. They concluded that the uncertainties in Q and U fluxes in individual pixels are described well by the expression

$$\sigma = \sqrt{\sigma_{rms}^2 + \sigma_{Dterm}^2 + (1.5\sigma_{rms})^2} \tag{5.1}$$

where σ_{Dterm} is the error due to the presence of residual instrumental polarisations in the data. Roberts, Wardle & Brown (1994) express this as

$$\sigma_{Dterm} \simeq \frac{\sigma_\Delta}{\sqrt{N_{ant} N_{IF} N_{scan}}} \sqrt{I^2 + (0.3\, I_{peak})^2} \tag{5.2}$$

where σ_Δ is the estimated uncertainty in the individual D-terms, N_{ant} the number of antennas in the VLB array (assuming all have altitude-azimuth mounts), N_{IF} the number of IFs (sub-bands within the total observed band at a given frequency) used for the observations, N_{scan} the number of scans with independent parallactic angles, I the total intensity at the point in question, and I_{peak} the total intensity at the map peak. The term containing I_{peak} was added by Hovatta et al. (2012) to empirically take into account the fact that the residual D-term uncertainty tends to scatter polarized flux throughout the map. Equation (5.1) above explicitly demonstrates that, even if the D-term error term is negligible, the uncertainty in fluxes in regions of source emission is somewhat higher than the map rms in regions far from source emission. This is in agreement with the Monte Carlo simulations performed as part of the error analysis on the CLEAN algorithm in Chapter 6. While Chapter 6 suggests that the full behaviour of the errors is more complicated than the increase in on–source error described by Equation (5.1), using this equation is nevertheless an improvement over the older method of simply assuming the error to be equal to the standard deviation in an area of the map far from the source.

In the case of the data used for this experiment, $N_{ant} = 10$, $N_{IF} = 4$ for the 7.9–15.4 GHz observations and 2 for the 4.6 GHz, 5.1 GHz and 1.36–1.67 GHz observations, and $N_{scan} \simeq 8$. By studying the scatter of the D-terms a value of $\sigma_\Delta \simeq 0.005$ was estimated for all the experiments. As is evident from Equation (5.2), the largest value for σ_{Dterm} will occur at the peaks of the maps; at the positions where we have determined the RM (see Table 5.1), the resulting D-term uncertainties are no more than $\simeq 0.60\sigma_{rms}$ for 0716+714 and no more than $\simeq 0.40\sigma_{rms}$ for 1749+701, making σ_{Dterm} small compared to the other terms contributing to σ.

The Q and U uncertainties determined in this way were then propagated to derive the corresponding uncertainties in the polarisation angles, σ_χ:

$$\chi = \frac{1}{2} ArcTan(\frac{U}{Q}) \tag{5.3}$$

$$\sigma_\chi^2 = \frac{1}{4}[(\frac{Q}{Q^2+U^2})^2\sigma_U^2 + (\frac{U}{Q^2+U^2})^2\sigma_Q^2]. \tag{5.4}$$

The uncertainty in the EVPA calibration σ_{EVPA} were then added in quadrature:

$$\sigma^2_{\chi_{final}} = \sigma^2_{\chi} + \sigma^2_{EVPA}. \tag{5.5}$$

The uncertainties in the polarisation angle were then used to fit a straight line to the χ vs. λ^2 data using standard least squares techniques. See Barlow (1993) for a complete description of this process. A detailed description of the fitting process is also provided in Section 6.5.

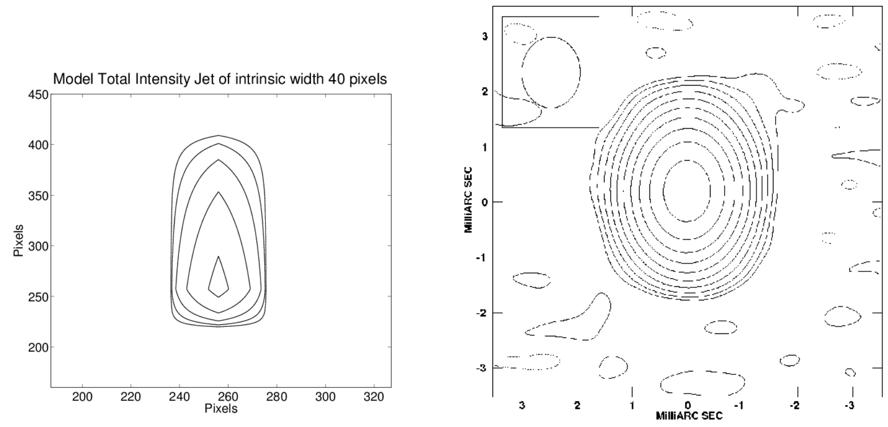
It is notable that, as the same EVPA calibration is applied to each polarisation angle at a given frequency, the uncertainty this introduces is systematic in effect. One consequence of this is that, although the EVPA calibration uncertainties increase the uncertainties in the fitted RM values, EVPA calibration uncertainties do not give rise to spurious RM gradients (see, e.g., Mahmud et al. 2009, Hovatta et al. 2012). The reason for this is essentially that any EVPA calibration error corresponds to a specific systematic offset that affects all EVPA measurements at all points of the maps at the corresponding frequency equally and in the same direction, and so will not induce gradients between points. A full analysis of this behaviour is performed in Section 6.5.

To take into account the fact that EVPA calibration uncertainty increases the total absolute uncertainty in a derived RM value, but essentially cancels out when calculating the difference between two RM values in an RM image, we calculated two different RM values: one RM value whose uncertainty included the effect of the EVPA uncertainty in the polarisation angle values, and a second "relative" RM value whose uncertainty was calculated without including the EVPA uncertainty in the χ uncertainties. The latter was used when calculating the uncertainties in (and thereby significance of) RM gradients.

5.3 New Monte Carlo Simulations

5.3.1 The Question of Resolution

Taylor & Zavala (2010) proposed some criteria for the reliable detection of transverse Faraday rotation gradients. A controversial element of the criteria was the suggestion that the observed RM gradient span at least three "resolution elements" across the jet. This criterion reflects the desire to ensure that it is pos-



(a) Intrinsic Map. (b) Simulated observation.

Figure 5.6: Stokes I maps of the intrinsic model jet and the jet after simulated observation and imaging. The ellipse in the upper left corner of the simulated observation indicates the beam size corresponding to the observation.

sible to distinguish properties between regions located on opposite sides of the jets. The criterion of three "resolution elements" has been taken to correspond to three beamwidths, and coincides with the general idea that structures separated by less than a beamwidth are not well resolved.

To test the validity of this criterion of Taylor & Zavala (2010), core–jet-like sources were constructed with various intrinsic widths and with transverse RM gradients present across their structures, and Monte Carlo simulations were carried out based on these model sources. A model jet with a transverse RM gradient present across the jet was constructed and Monte Carlo simulations carried out based on this model source. The model source was cylindrical, with a fall-off in intensity at either side of the cylinder axis and along the axis of the cylinder from a specified point located near one end of the cylinder (see Figure 5.6). The resulting appearance of the model emission is broadly speaking "core-jet-like".

5.3.2 Monte Carlo Procedure

Model visibility data were generated for each of the six frequencies listed in section 5.1 (4.6-15.4 GHz), including the effect of the transverse RM gradient in the Q and U visibility data, and these model visibility data were sampled at precisely the (u, v) points at which 0716+714 was observed at each of the frequencies. Random thermal noise and the effect of uncertainties in the EVPA calibration by up to 3° were added to the sampled model visibilities. The amount of thermal noise added was chosen to yield rms values in the simulated images that were comparable to those in the actual observations. The noise was added to the model visibilities using the Python SciPy statistics library.

Stokes I, Q and U images were constructed from these visibilities in CASA, using the same beam as was used in the observations of 0716+714 (1.28×1.06 mas in $PA = -0.84°$, where the dimensions given correspond to the full width at half maximum of the beam along its major and minor axes). The polarisation of the model was chosen to yield a degree of polarisation in the lower half of the convolved model image (the "core" region) of about 5% and a degree of polarisation in the upper half of the convolved model image of about 10% – similar to the observed values for 0716+714. The Q and U images were then used to construct the corresponding polarisation angle images at each frequency, which were, in turn, used to construct RM images in the usual way. Finally, Monte Carlo RM maps were constructed, based on 200 independent realizations of the thermal noise and EVPA calibration uncertainty. In each case, the RM values were output to the RM map only in pixels in which the RM uncertainty indicated by the fitting was less than 80 rad/m^2; this value was chosen so that no spurious pixels were written to the output RM maps for any of the 200 realizations of the RM distribution. Finally, an average RM map was derived by averaging together all 200 individual realizations of the RM distribution.

This procedure was carried out for a number of model sources with the general form shown in Figure 5.6, all with a length of 1 mas and with transverse widths of 0.50, 0.35, 0.20. 0.10 and 0.05 mas. A recent observation of 0716+714 with the *RadioAstron* space antenna and the European VLBI Network had measured the size of a feature in the 6.2-cm core region to be 0.07 mas (Kardashev et al. 2013), thus the narrowest jet was designed to have a width somewhat smaller than this.

Two types of monotonic transverse RM gradients were considered: uni-directional along the entire source structure, and oriented in one direction in the "core" region and in the opposite direction in the "jet" region, i.e., showing a reversal. These Monte Carlo simulations complement those carried out by Hovatta et al. (2012), in which simulated RM maps were made from model data that did not contain RM gradients, to determine the frequency of spurious transverse RM gradients appearing in the simulated RM maps.

An example of the total intensity maps of the model sources used in the simulation can be seen in Figure 5.6, and the results of the RM Monte Carlo simulations are shown in Figures 5.7 to 5.10 (the results for the jet width of 0.50 mas are not included as they are very similar for the 0.35-mas jet width). The panels in Figures 5.7 to 5.10 show (i) the RM map obtained by putting data without added thermal noise through the imaging procedure (i.e., the intrinsic RM distribution, but subject to errors due to the CLEAN process and limited *uv* coverage); (ii) two examples of the individual “noisy” RM maps obtained. In all cases, the model source structure corresponds to that shown in Figure 5.6, so that the compact VLBI jet lies directly to the North. Note that the colour scales for the three maps in a corresponding set have been individually chosen to highlight the RM patterns present, and may differ somewhat in some cases.

5.3.3 Simulated Observations

In all cases, the RM gradients that were introduced into the simulated data are visible in the “noisy” RM maps that were obtained, even when the intrinsic width of the jet is approximately 1/20 of the beam full-width at half-maximum (FWHM). The magnitude of the RM gradient is reduced by the convolution more and more as the size of the beam relative to the intrinsic size of the jet width increases, but the RM gradients that were initially introduced into the simulated data remain visible. In the case of jet widths much less than the beam FWHM, the appearance of individual realizations can sometimes be fairly strongly distorted by noise; however, in all cases, averaging together all the individual realizations confirms the presence of the RM gradients in the simulated images.

These results essentially indicate that it is not necessary to impose a restriction on the width spanned by an observed RM gradient, *provided* that the difference between the RM values observed at opposite ends of the gradient is at least 3σ. This is consistent with the results of Murphy & Gabuzda (2012), who investigated the effect of resolution on transverse RM profiles. It is also consistent with Fig.30 of Hovatta et al. (2012), which shows that the fraction of “false positives”, i.e., spurious RM gradients, that were obtained in their Monte Carlo simulations did not exceed $\simeq 1\%$ when a 3σ criterion was imposed for the RM gradient, even when the observed width of the RM gradient was less than 1.5 beamwidths. It becomes important to place some restriction on the width spanned by the gradient if the difference between the RM values being compared is less than 3σ, as was also shown clearly by the Monte Carlo simulations of Hovatta et al. (2012).

The results of these new Monte Carlo simulations thus directly demonstrate that

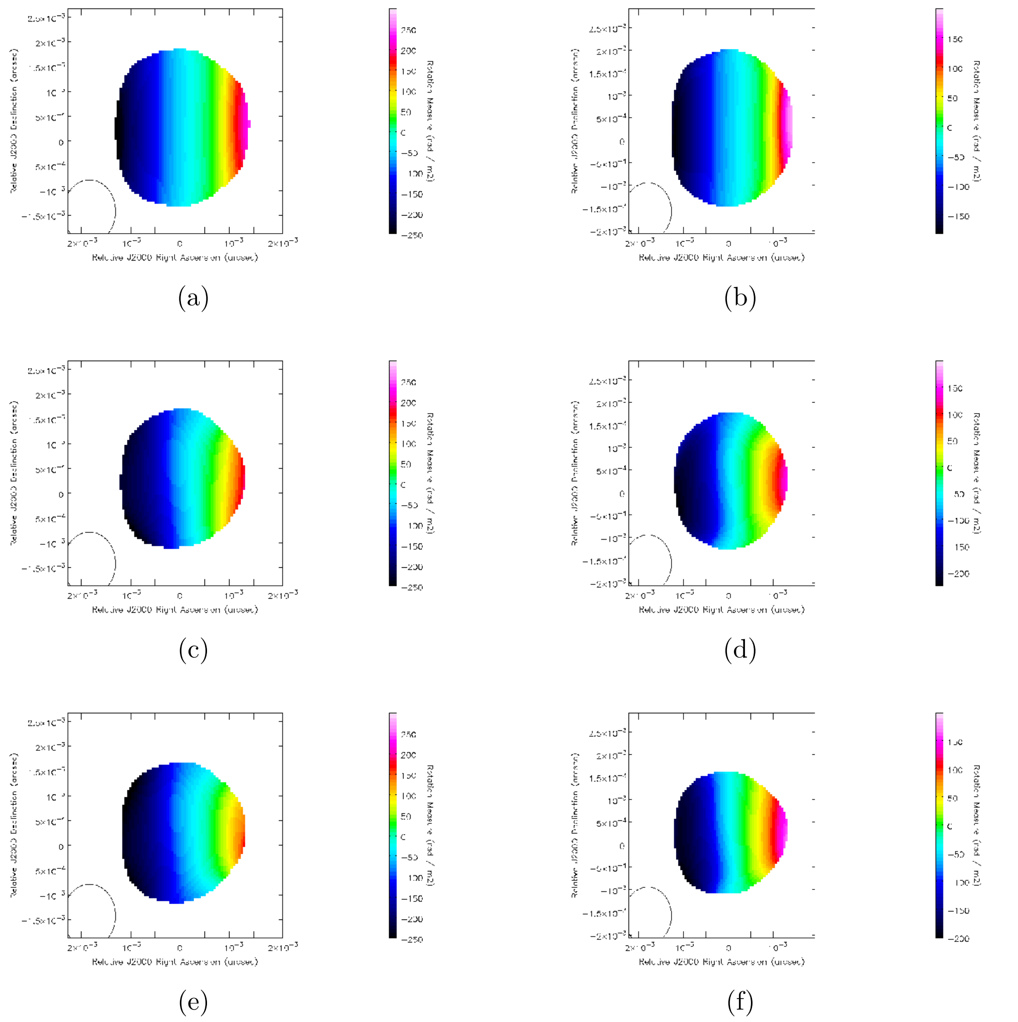
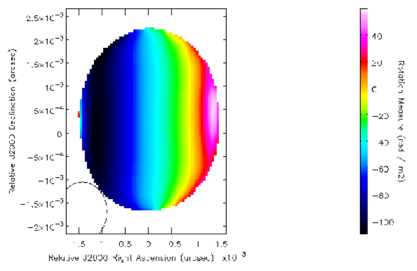
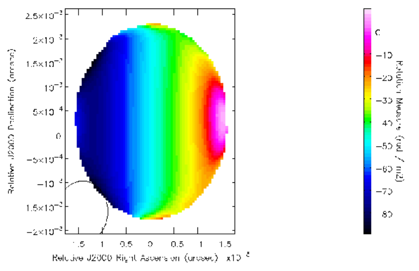
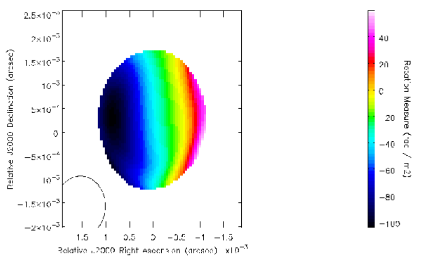
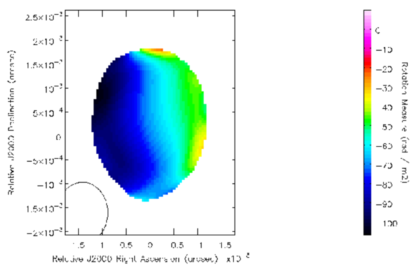


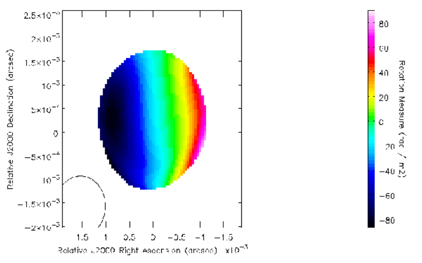
Figure 5.7: Results of Monte Carlo simulations using model core–jet sources with uniformly directed transverse RM gradients. The compact VLBI jet extends toward the North. The intrinsic width of the jet (RM gradient) on the left is 0.35 mas and on the right is 0.2 mas. The convolving beam(1.28 mas×1.06 mas in PA = −0.84°) is shown in the lower left-hand corner of each panel. The top panel shows the RM image obtained by processing the model data as usual, but without adding random noise or EVPA calibration uncertainty; pixels with RM uncertainties exceeding 10 rad/m^2 were blanked. The remaining two panels show two examples of the 200 individual RM images obtained during the simulations; pixels with RM uncertainties exceeding 80 rad/m^2 were blanked.


(a)


(b)


(c)


(d)


(e)

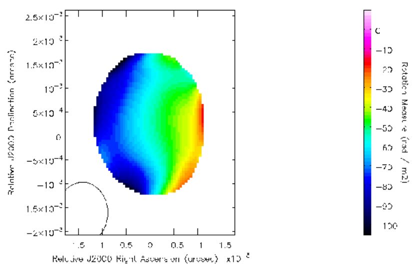
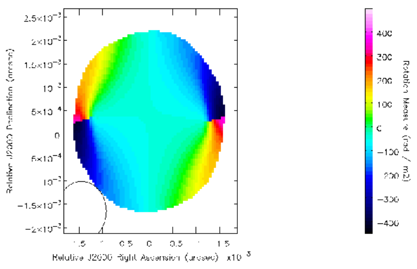
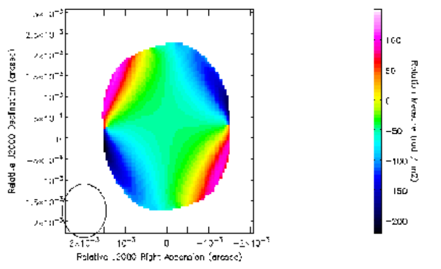
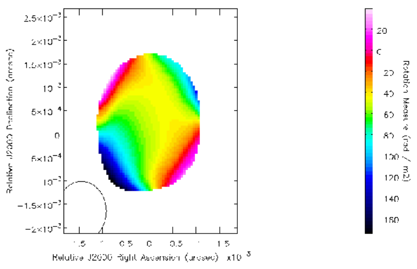
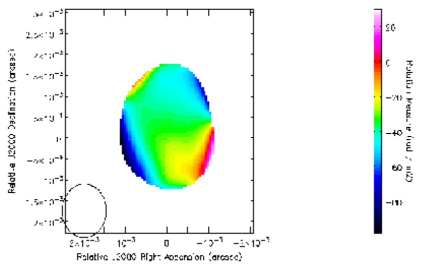

(f)

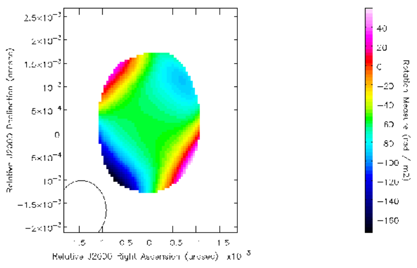
Figure 5.8: Results of Monte Carlo simulations using model core–jet sources with uniformly directed transverse RM gradients. The compact VLBI jet extends toward the North. The intrinsic width of the jet (RM gradient) on the left is 0.1 mas and on the right is 0.05 mas. The convolving beam(1.28 mas$\times$1.06 mas in PA = $-0.84°$) is shown in the lower left-hand corner of each panel. The top panel shows the RM image obtained by processing the model data as usual, but without adding random noise or EVPA calibration uncertainty; pixels with RM uncertainties exceeding 10 rad/m^2 were blanked. The remaining two panels show two examples of the 200 individual RM images obtained during the simulations; pixels with RM uncertainties exceeding 80 rad/m^2 were blanked.


(a)


(b)


(c)


(d)


(e)

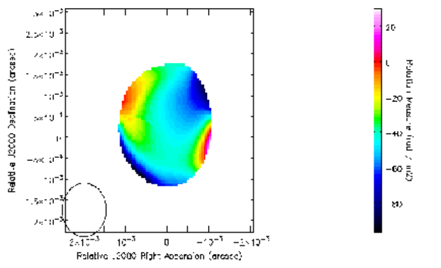
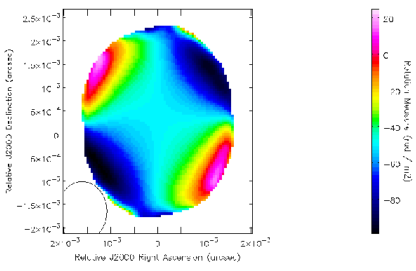
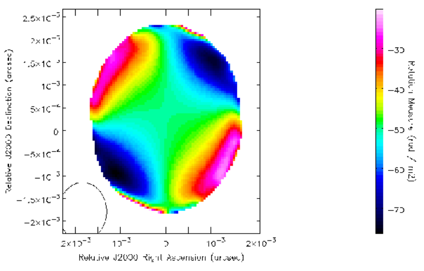

(f)

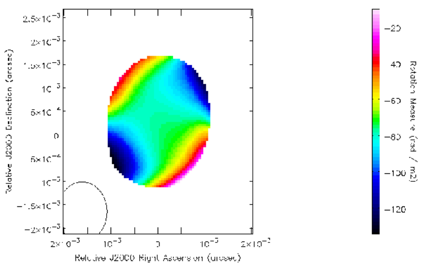
Figure 5.9: Results of Monte Carlo simulations using model core–jet sources with oppositely directed transverse RM gradients in the core region and inner jet. The compact VLBI jet extends toward the North. The intrinsic width of the jet (RM gradient) on the left is 0.1 mas and on the right is 0.05 mas. The convolving beam(1.28 mas$\times$1.06 mas in PA $= -0.84°$) is shown in the lower left-hand corner of each panel. The top panel shows the RM image obtained by processing the model data as usual, but without adding random noise or EVPA calibration uncertainty; pixels with RM uncertainties exceeding 10 rad/m^2 were blanked. The remaining two panels show two examples of the 200 individual RM images obtained during the simulations; pixels with RM uncertainties exceeding 80 rad/m^2 were blanked.



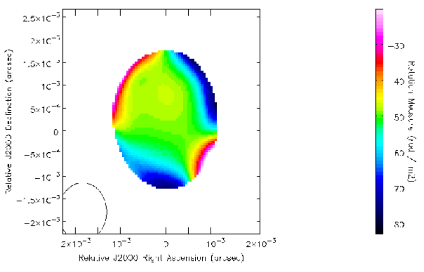
(a)



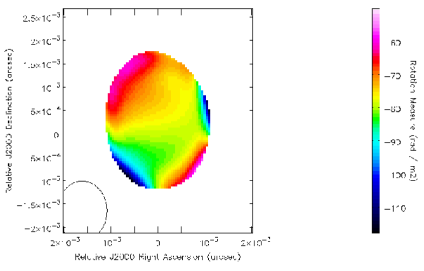
(b)



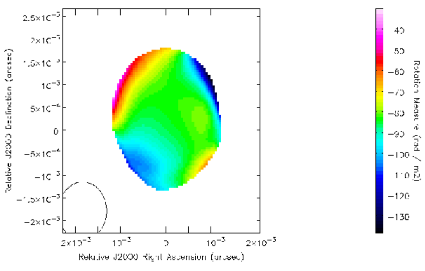
(c)



(d)



(e)



(f)

Figure 5.10: Results of Monte Carlo simulations using model core–jet sources with oppositely directed transverse RM gradients in the core region and inner jet. The compact VLBI jet extends toward the North. The intrinsic width of the jet (RM gradient) on the left is 0.1 mas and on the right is 0.05 mas. The convolving beam(1.28 mas$\times$1.06 mas in PA = $-0.84°$) is shown in the lower left-hand corner of each panel. The top panel shows the RM image obtained by processing the model data as usual, but without adding random noise or EVPA calibration uncertainty; pixels with RM uncertainties exceeding 10 rad/m^2 were blanked. The remaining two panels show two examples of the 200 individual RM images obtained during the simulations; pixels with RM uncertainties exceeding 80 rad/m^2 were blanked.

the three-beamwidth criterion of Taylor & Zavala (2010) is overly restrictive, since the simulations directly show the possibility of detecting transverse RM gradients even when the intrinsic widths of the corresponding source structures are much less than the beamwidth, resulting in RM distributions that span only $1 - 1.5$ beamwidths. This demonstrates that the relatively modest widths spanned by the transverse RM gradients in 0716+714 and 1749+701 described in Section 5.1 should not be taken by themselves as grounds to question the reliability of these gradients. The Monte Carlo simulations described above are not intended to provide a physical model of the observations, or to reproduce our observed RM distributions in any detail; instead, they are intended solely to demonstrate the possibility of detecting a transverse RM gradient in real data, even if the intrinsic jet width is much smaller than the beam FWHM.

Inspection of Fig. 30 of Hovatta et al. (2012) indicates that the fraction of "false positives", i.e., spurious RM gradients, that were obtained in their Monte Carlo simulations did not exceed $\simeq 1\%$ when a 3σ criterion was imposed for the RM gradient, even when the observed width of the RM gradient was less than 1.5 beamwidths. This suggests that there may be up to a $\simeq 1\%$ probability that the RM gradients we report here are spurious, due to their relatively limited widths, although we consider this to be unlikely, given that the RM differences involved correspond to as much as 5σ.

5.4 Discussion

The strong (greater than 3σ) gradients described in Section 5.1 and Table 5.1 suggest that transverse RM gradients have been observed in the AGN 0716+714 and 1749+701. The Monte Carlo simulations detailed in Section 5.3 indicate that, contrary to the resolution criteria of Taylor and Zavala, the VLBA does indeed have the resolution required to detect gradients in Faraday rotation measure across relatively narrow jets, including cases where the gradient is observed to reverse direction over the course of the jet.

Due to optical-depth effects that may be present in the core (the possible transition between an optically thick region, where light from the source is likely to be absorbed and re-emitted in the source before being detected, and an optically thin region, where the emission from the source is unlikely to be absorbed and re-emitted), it is not entirely possible to confirm the reliability of the gradients in the cores of the AGN, thus the detection of an actual gradient reversal remains tentative. However, no unusual polarisation features characteristic of an optically thin/thick transition in the core were observed for either source and the quality of the linear fits is consistent across both the jet and core regions of the sources. Thus while the possibility of such a transition giving rise to a false gradient cannot be ruled out, it can be considered unlikely in these cases.

If the detected reversals in Faraday RM gradient across the jets are indeed real, there are only a few potential physical causes for such an event. The first of these is the possible reversal of the “pole” of the black hole facing the Earth. If the polarity of the rotating black hole at the heart of the AGN reversed, a similar reversal would be expected in the azimuthal component of any helical magnetic field, giving rise to a reversal in the detected gradient. However in the absence of any known mechanism which could cause such a polarity reversal in a supermassive black hole, this explanation is unlikely.

A second possibility is the presence of torsional oscillations in the helical magnetic field. Bisnovatyi-Kogan (2007) demonstrated that such oscillations may help stabilise the jet and may cause the direction of the azimuthal component of the helical field to flip with time (and therefore with distance from the core). In this scenario the reversal caused by the oscillations may be expected to propagate along the jet with time.

A final possible explanation for such reversals in the direction of the Faraday rotation measure gradient is a nested magnetic field structure along the lines of that caused by the Poynting-Robertson battery. A Poynting-Robertson battery in an AGN is a current that may arise from a potential difference created from the differential rotation of electrons and protons in the accretion disk of the AGN. This differential rotation is due to the fact that the Poynting-Robertson radiation drag force on a radiating body is proportional to the Thomson cross sectional area of the body (see, for example Contopoulos and Kazanas 1998). The Thomson cross sectional area of a body is

$$\sigma_t = \frac{8\pi}{3}\frac{q^2}{mc^2}. \tag{5.6}$$

The dependence of the cross section on the mass of the radiating body means that lower mass electrons will be affected much more than protons. As the electrons slow down and the protons pull ahead the difference in charge can cause a current to flow. This azimuthal current (in the plane of the accretion disk) can cause a poloidal magnetic field with a dependence on the direction of rotation of the

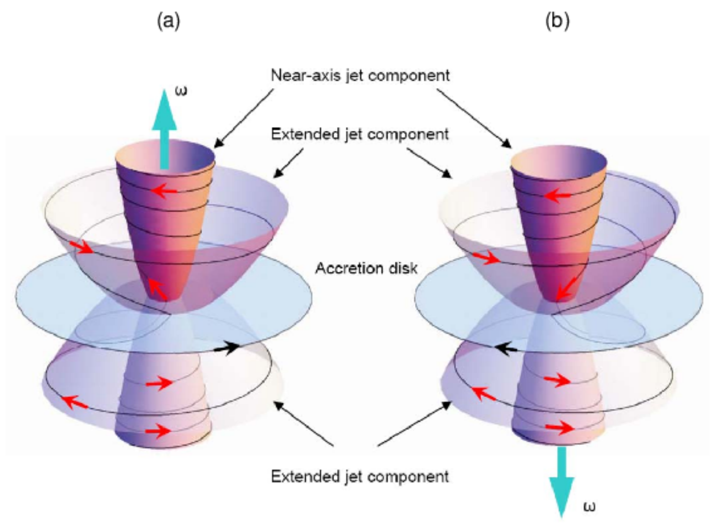


Figure 5.11: The magnetic field in an AGN generated by a Poynting-Robertson battery. The direction of rotation of the disk is indicated by the black arrows, with the corresponding axis of rotation shown by the cyan arrows. The direction of rotation of the helix is indicated by the red arrows. When the direction of rotation is reversed, the direction of the poloidal magnetic field is too – but the azimuthal component of the field does not change. Image from Contopoulos et al. (2009).

accretion disk. The resulting azimuthal component of the field that comes about when the poloidal field is "wound up" by the rotation of the central black hole and its accretion disk is independent of the direction of the rotation (see Figure 5.11).

The existence of inner and outer helices in the magnetic field structure of AGN jets has interesting consequences for Faraday rotation measurements across the jets. It is possible that the inner part of the helix dominates at some regions, while the outer part dominates at other regions. Intermediate regions may contain significant contributes from both fields. The Poynting-Robertson battery, or a mechanism with a similar nested-helix structure for the magnetic field, may be the explanation for observed reversals in the direction of the transverse Faraday rotation measure gradient along the jet. Mahmud et al. (2009) provides a further discussion of such an explanation.

This final explanation is suggested to be the most likely reason for the reversals observed in 0716+714 and 1749+701. Continuous observations of the sources for any changes in the direction of the gradient and the possible propagation of the gradients along the jet with time may yield a deeper insight into the mechanism driving these changes.

The Poynting-Robertson battery also makes the prediction that close to the VLBI core (as near as can be observed to the black hole) the inner helix of the magnetic field should be dominant over the outer helix. This is due to the fact that the outer helical magnetic field can be interpreted as the “return field”. If this were true, based on Figure 5.11 one would expect to see an excess of clockwise Faraday rotation measure gradients in regions closer to the VLBI core (a clockwise gradient is defined as a transverse gradient which would be in a clockwise direction when viewed from the AGN centre). This excess stems from the independence of the azimuthal component of the magnetic field from the rotational direction of the accretion disk. Contopoulos et al. (2009) and Gabuzda et al. (2012) found evidence for such an excess, as well as an increase in counter clockwise gradients as one moves along the jet from the core. This suggests that the Poynting-Robertson battery may indeed be responsible for the magnetic field structure of AGN jets.

5.5 Conclusions

This chapter has detailed the statistically significant observation of reversals in the direction of the Faraday rotation measure gradients across two sources; 0716+714 and 1749+701. Monte Carlo simulations of observations with realistic noise strongly indicate that such a change in direction of a Faraday RM gradient should be visible to the VLBA, though they reveal that the observed RM values may differ significantly from the original values. These results are in contrast to an earlier suggestion by Taylor and Zavala (2010) that the detection of features on scales below 3 beamwidths be considered unreliable. In fact, the simulations discussed in Section 5.3 show that reliable results can be obtained even when the intrinsic width of the jet is $\frac{1}{20}$ of a beamwidth – suggesting that the limiting factor in the observation of Faraday RM gradients in AGN jets is not the intrinsic width of the jet.

Given the results of these Monte Carlo simulations, and the statistical significance of the gradients based on the error method of Hovatta et al. (2012), the gradient reversals observed in 0716+714 and 1749+701 appear to be real. While a number of explanations are possible, the Poynting-Robertson battery model proposed by Contopoulos et al. (2009), or a variant thereof, appears to be be most likely. Further evidence for the presence of this effect, along the lines of the analyses in

Contopoulos et al. (2009) and Gabuzda et al. (2012), would be a major step in the understanding of the launch and collimation of AGN jets.

Chapter 6

A New Approach to Estimating Uncertainties in VLBI Images

This chapter discusses the uncertainty in CLEAN maps of Stokes I and polarisation emission from AGN-like sources. Monte Carlo simulations to determine the uncertainties are conducted and a model of the errors is proposed. The advantages of the model in some important calculations is discussed. A good reference for many of the common statistical formulae used in this chapter is *Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences* by R.J. Barlow.

6.1 Correlation in the CLEAN algorithm

Section 2.2 describes the operation of the CLEAN algorithm. It is summarised again here introducing a new mathematical notation:

- The "dirty map" is found by taking the Fourier transform of the initial visibility data.

$$I_d(x, y) = F.T.(V_I(u, v)) \tag{6.1}$$

- The peak value in this map is found and assumed to represent real emission.
- The amplitude of this peak value is multiplied by a gain factor (often 0.1) and added to a list of "CLEAN components", together with its position.
- The convolution of the new CLEAN component with the dirty beam (the point spread function) is then subtracted from the map.

- The previous three steps are then repeated until one is left with a map with a peak comparable to the rms noise level of the map.
- The source has now been modelled as a sum of point δ function sources (listed in the CLEAN components table). These sources are then convolved with a CLEAN beam to form a partial CLEAN map. The CLEAN beam is normally taken to be an elliptical Gaussian fit to the central lobe of the point spread function. This operation can be expressed mathematically as

  $$I_{Ck} = \sum_{l=1}^{n} B(x_k - x_l, y_k - y_l) i_l \equiv \sum_{l=1}^{n} B_{lk} i_l \tag{6.2}$$

  where I_{Ck} is the value of the partial CLEAN map in pixel k, B_{lk} is the value of the CLEAN beam centred on l evaluated at pixel k, i_l is the CLEAN component l and n is the number of CLEAN components.
- The residuals left over after removing the complete table of CLEAN components from the map are then added to the partial CLEAN map to form the final CLEAN map.

  $$I_k = I_{Ck} + R_k \tag{6.3}$$

  where I is the final CLEAN map, I_C is the partial map and R is the residual map.

It can be determined from the description above that nearby pixels in the final CLEAN map are strongly correlated with each other due to the effect of convolving the list of CLEAN components with the CLEAN beam. This correlation comes into play in Equation (6.2), and although the residuals are added back in by Equation (6.3), the resulting values at any pixel on the source are largely the result of convolved CLEAN components.

This correlation cannot be ignored in performing any statistical analysis of the map. To illustrate the importance of the correlation, consider a CLEAN map made of a single δ function represented by one CLEAN component. When the CLEAN component is convolved with the CLEAN beam as in Equation (6.2) the result will be a Gaussian spread over many pixels. After the residuals are added in, it might be desirable to calculate the mean flux in the core region of the resulting Gaussian. This can be done easily and one might be tempted to make the usual approximation that the standard deviation of the pixels is a suitable estimate for the 1σ error of the mean. However as the neighbouring pixels used to calculate the mean stem from a convolution of a single CLEAN component they are not independent from each other. In fact, in this case, if one assumes the residuals to be small the mean value calculated is not statistically useful in any sense as it is based off a single measurement.

Until the publication of Hovatta et al. (2012), radio astronomers usually took the off-source RMS noise as an estimate of the uncertainty in any pixel. Hovatta et al. noted that in Monte Carlo simulations the on-source error was often significantly higher, suggesting an on-source error for Stokes Q and U maps of

$$\sigma_{on-source} = \sqrt{\sigma_{rms}^2 + \sigma_{Dterm}^2 + (1.5\sigma_{rms}^2)} \tag{6.4}$$

where σ_{Dterm} is the contribution due to uncertainty in the D term calibration (see Sections 1.4.4 and 5.2) determined by Roberts and Wardle (1994). Assuming a small contribution by the D terms, as will be the case in sources that are not especially bright away from the immediate vicinity of the core component, Equation (6.4) becomes

$$\sigma_{on-source} \approx 1.8\sigma_{rms}. \tag{6.5}$$

This description of the errors in a CLEAN map, while more advanced than simply using σ_{rms}, nevertheless continues to ignore the correlation between neighbouring pixels established above. The following section proposes a model of the errors which allows the explicit treatment of this correlation.

6.2 An error model for CLEAN

The correlation present in a CLEAN map is due to the convolution of the CLEAN components with the CLEAN beam. In a sense, the CLEAN components may be considered measurements of the data in the dirty map used to build up a CLEAN component model of the source and, like any measurement, they can have an error associated with them. Assuming that the error in the measurement of an individual CLEAN component is of the order σ_{rms}, the following approximation may be made for the value of that CLEAN component and its uncertainty

$$i_l \pm f\sigma_{rms} \tag{6.6}$$

where f is of order unity. As the units of σ_{rms} are Jy/Beam, while the units of a CLEAN component are Jy, this gives f the unit of Beam, where this refers to the beam area in pixels or radians, etc., as appropriate. The value of f is unlikely to be exactly constant in a single map, as the successive CLEAN components in the same map are correlated with previous CLEAN components, and a component measured in one location may have a different effect than a component measured in another. The gain of the CLEAN algorithm would also affect the value of f, as would the choice of whether to count multiple CLEAN components in the same pixel as a single component with a large value (the collapsed CLEAN component) or as multiple small components (uncollapsed components). As f is related to the beam, the value of f may also vary depending on the ratio of the size of the source to the observing beam. However, if the value of f is taken to be approximately constant and the CLEAN components are treated as being approximately independent, this error can be straightforwardly propagated to the partial CLEAN map. The equation for the propagation of error in a function f of n independent variables x_i with uncertainties σ_{x_i} is

$$\sigma_f^2 = \sum_{i=1}^{n} (\frac{\partial f}{\partial x_i})^2 \sigma_{x_i}^2. \tag{6.7}$$

As CLEAN components are independent variables representing a δ function of emission at their locations this formula can be applied to Equation (6.2), resulting in the following expression for the error in the partial CLEAN map:

$$\sigma_{I_{Ck}}^2 = f^2 \sigma_{rms}^2 \sum_{l=1}^{n} B_{lk}^2. \tag{6.8}$$

The final step in making the CLEAN map is adding the residual map to the partial CLEAN map (Equation (6.3)). Equation (6.7) yields the following expression for the error in the final CLEAN map

$$\sigma_{I_k}^2 = \sigma_{I_{Ck}}^2 + \sigma_{rms}^2. \tag{6.9}$$

Therefore the following expression may be used to calculate the error at any pixel in the CLEAN map.

$$\sigma_{I_k}^2 = \sigma_{rms}^2 (1 + f^2 \sum_{l=1}^{n} B_{lk}^2). \tag{6.10}$$

This expression for the error explicitly includes the effect of correlation. It can been seen from the equation that far from the source the error will be approximately σ_{rms}, however close to any CLEAN components the predicted error will be higher. This is qualitatively consistent with the findings of Hovatta et al. (2012). Detailed Monte Carlo simulations were then carried out to determine the true behaviour of the CLEAN errors and compare this behaviour to the expectations of this model.

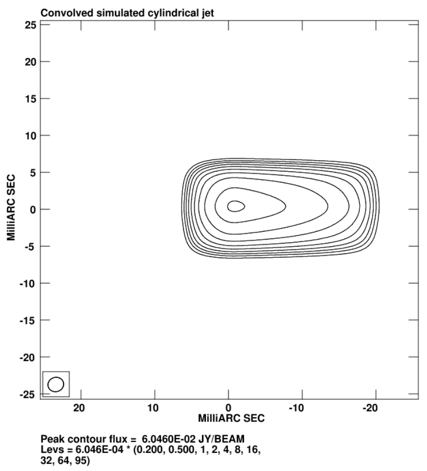
6.3 Monte Carlo Simulations

Although the model proposed in Section 6.2 is conceptually simple and results in a potentially useful mathematical framework, Monte Carlo simulations were required to investigate whether or not the behaviour of the CLEAN algorithm could be approximated with such an approach. If this proved to be true, a suitable value for the f factor in Equation (6.6) could also be determined. A number of model sources were designed to challenge the CLEAN algorithm in different ways, ranging from a fully artificial test, to a tests on realistic sources. As a first test of the method sources many beams across (with many CLEAN components) were tested (Section 6.3.1). These tests were then followed up with much smaller sources (Section 6.3.2). In all cases the collapsed CLEAN component table was used as the basis for the error model.

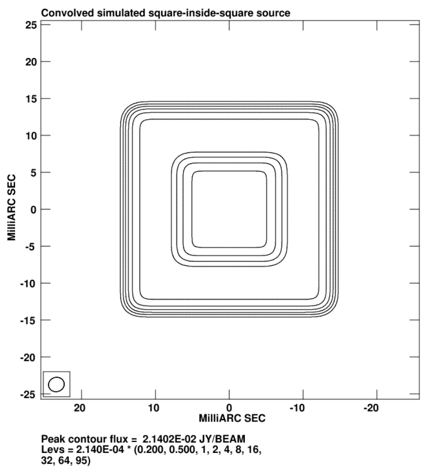
Hovatta et al. (2012) conducted a similar set of Monte Carlo simulations using the AIPS and DIFMAP software suites to determine Equation (6.5), however they only tested individual pixels and did not create error maps of the kind presented in this thesis. These error maps show strong variation in the value of the uncertainty in an individual pixel over the course of the map and suggest that the error is indeed correlated with CLEAN component position.

6.3.1 Initial Testing

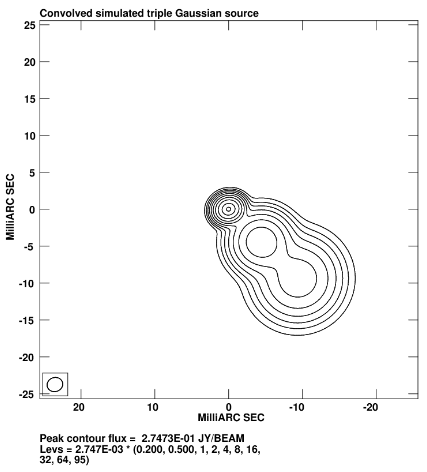
In order to provide an initial test of the method well resolved sources with large number of CLEAN components with both small separations (within the same beam) and large separations were designed to see if the errors on a CLEAN map were really associated with the CLEAN component locations. The three models chosen for this first set of Monte Carlo simulations can be seen in Figure 6.1.



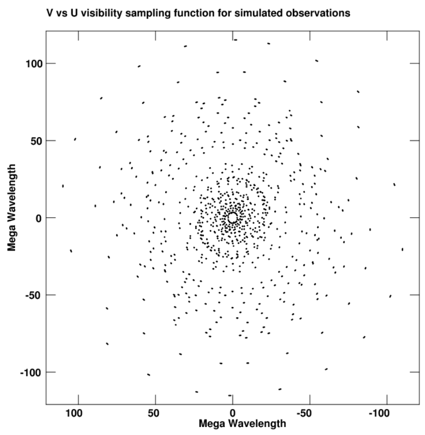
(a) Cylindrical jet with an intrinsic linear decrease in intensity



(b) Square-within-square source



(c) Triple Gaussian source



(d) UV sampling function

Figure 6.1: Convolved model maps and UV distribution used. Total flux in each case is 1 Jy. Beam = 2.14 × 1.93 mas, position angle −75.11°. Contours shown are 0.2, 0.5, 1 , 2, 4, 8, 16, 32, 64, 95 % of peak. Sources have peak values of (a) 60.46 mJy/Beam, (b) 21.40 mJy/Beam and (c) 274.7 mJy/Beam. The cylindrical jet in Figure 6.1a has an intrinsic length of 25 mas and width of 10 mas. The Square-within-square source in Figure 6.1b has an intrinsic length of 14 mas for the inner square, and 28 mas for the outer square. The 3 components of the triple Gaussian source in Figure 6.1c have intrinsic full widths at half maximum of 1 mas, 8 mas and 12 mas.

These models were generated in OCTAVE and exported as CSV files with a total flux of 1 Jy in each case. A cylindrical jet with a linear decrease in intensity (Figure 6.1a) was designed to test how CLEAN responded to continuous emission that decreased monotonically. A square-within-square source (Figure 6.1b) was designed to test how CLEAN coped with very sudden changes in flux. Both the inner and outer squares had constant fluxes, with the inner square having a value equal to ten times that of the outer square. This is not a realistic model of any AGN, but serves to test the CLEAN algorithm and help understand the behaviour of the f factor introduced in Section 6.2. The final model used in this set of simulations was designed to be a more realistic model of a core–jet source with three Gaussian components (see Figure 6.1c).

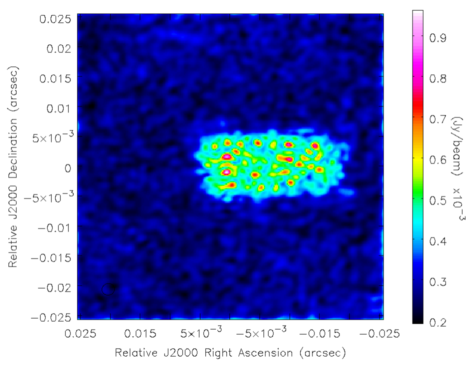
The C++ program UVFILL2 was written to take in these files and the UV distribution shown in Figure 6.1d and produce simulated observations of the sources using a direct Fourier Transform to calculate the simulated visibilities at the given UV coordinates. Thermal noise was then added to the visibilities to ensure a realistic noise level in the final maps. As in Chapter 3, the random element of the thermal noise added to the visibilities was achieved using the GNU Scientific Library random Gaussian function with zero mean and a user specified standard deviation, seeded with the current time multiplied by the process ID of the current CPU thread running UVFILL2. One hundred simulated observations were generated for each of the three models described above and these were then imaged with the CLEAN algorithm using an AIPS script written by myself. As each model was made with the same cell size and imaged with the same UV distribution the resulting CLEAN beam was the same in each case (2.14×1.93 mas, position angle $-75.11°$). This beam was much smaller than the intrinsic size of each of the sources, resulting in length to beam ratios of approximately 15.

The final maps were then exported as FITS files and another C++ program, MCAVERAGE, was written and used to compare the final maps to a convolution of the original model with the same CLEAN beam. Due to the differences in map centring the simulated maps had to be shifted slightly to align with the same position on the convolved model map. Thus the exact RMS error at every pixel could be calculated by finding the root mean square deviation of each of the simulated observations from the convolved model map. These results can be seen in the left panels of Figure 6.2.

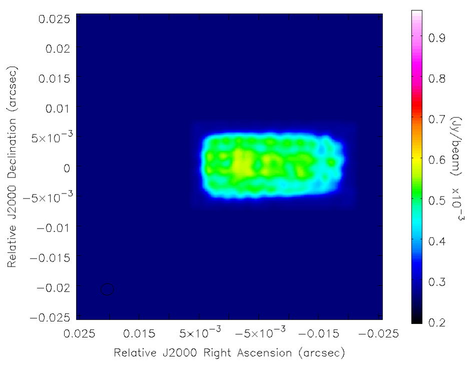
A C++ program, ERROR_MAP_MC, was written to calculate the errors for each of the individual simulated maps using only the final map and the collapsed CLEAN components on it according to the error model described in Section 6.2. OpenMP parallel processing was used to speed up the process of calculating the error maps and ensure that the full one hundred map batch could be processed quickly. Individual error maps were generated for each of the simulated observations and averaged together to give a final map representing the average predicted error in each pixel. The results can be seen in the right panels of Figure 6.2. These figures are analytical predictions of the RMS error in the CLEAN algorithm's attempt to image the source under realistic thermal noise and are directly comparable to the error distributions resulting from the Monte Carlo simulations shown in the corresponding left panels.

The Monte Carlo error maps on the left panels of Figure 6.2 show clearly that the uncertainty of a pixel in the source is, generally, higher than the error in a pixel off the source, as was also shown by Hovatta et al. (2012). There are also certain regions on the source which appear to have much higher errors than the rest of the source region. In the square-within-square source in the centre panels of Figure 6.2 it is clear that these regions of enhanced error appear to be correlated with the points in the model where the flux changes suddenly. The core-jet source has a much higher error at its peak than anywhere else. There are also regions of high error in the linear jet that do not seem to be associated with a sudden change in flux. Thus, at least for these relatively large sources (on scales of tens of milli-arcseconds, compared to a beam of approximately 2 mas) it appears that while the suggestion of Hovatta et al. (2012) that the error on the source is higher than off the source is true, there appears to be significant structure in the pattern of errors.

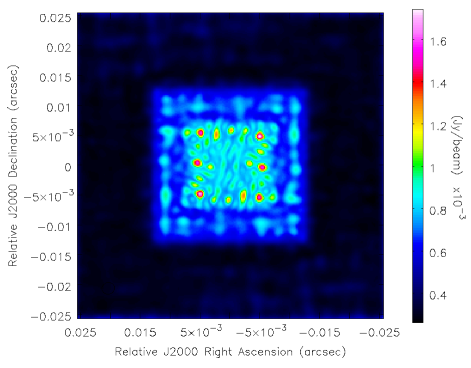
The calculated error maps on the right panels of Figure 6.2 show the error pattern created by the error model described in Section 6.2 with $f = 0.75$ Beam. There is strong agreement between the Monte Carlo and calculated error maps in both the magnitude and structure of the error patterns across all three sources. In particular, the ridges and bumps in the inner and outer squares in the square within square source in the centre panels are very well described by the error model, as is the sudden increase in error at the peak of of the triple Gaussian in the lower panels. This suggests that the regions of enhanced error are indeed correlated with the locations of CLEAN components and that at least some component of the error on the maps can be modelled along the lines suggested in Section 6.2. A comparison of the corresponding left and right panels of Figure 6.2 shows that there are individual compact (point-like) regions where the calculated model er-



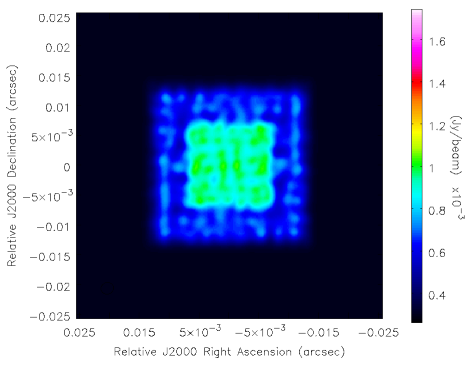
(a) Cylindrical jet MC errors



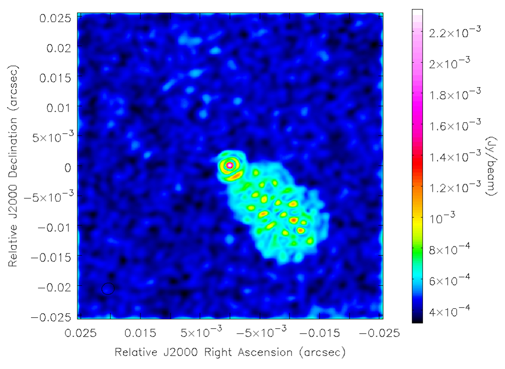
(b) Cylindrical jet calculated errors



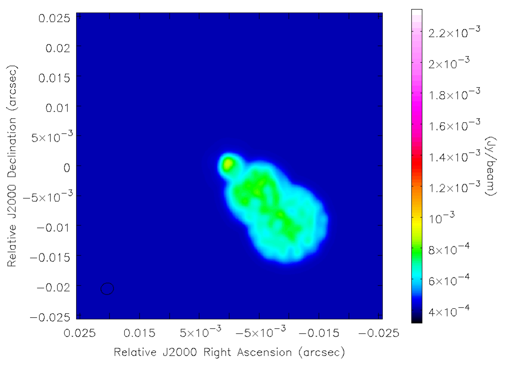
(c) Square-within-square MC errors



(d) Square-within-square calculated errors



(e) Triple Gaussian MC errors



(f) Triple Gaussian calculated errors

Figure 6.2: Monte Carlo and calculated error maps for large simulated sources. Monte Carlo results are in the left panels, and calculated errors on the right panels. The top panels show a cylindrical jet with a linear decrease in intensity. The centre panels show the square-within-square source and the lower panels show the triple Gaussian source. In all cases $f = 0.75$ Beam

rors are significantly lower than the Monte Carlo errors, by a factor of about 1.5–2. Although this discrepancy cannot be removed by adjusting the value of f, the possibility of underestimating uncertainties in individual regions when using the calculated model errors could be avoided by increasing the value of f so that the model agreed well with the Monte Carlo simulations in these compact regions of higher uncertainty, but at the expense of overestimating the uncertainties in the remaining regions of emission by a factor of 1.5–2.

Further investigation was required to test the performance of the model for smaller (more realistic) model sources and, in particular, to test the applicability of the model at the lower signal to noise regime present polarisation maps.

6.3.2 Testing Flux and Resolution Dependence

A second series of Monte Carlo simulations was conducted to examine any flux or resolution dependency in the behaviour of uncertainties in images produced using the CLEAN algorithm. As the Square-within-Square type source used in Section 6.3.1 was very successful in establishing the link between CLEAN component locations and uncertainty for very well resolved sources, similar sources with different sizes relative to the beam were used for the second series of tests. The inner square was offset slightly to the left relative to the original Square-within-square source. The full details of the new Square-within-Square sources can be seen in Table 6.1 and corresponding $u - v$ and convolved model data can be seen in Figures 6.3 and 6.4. There is no reason that Stokes Q and U should behave differently with regard to their uncertainties; however we have assigned different flux levels to Q and U in the models (total fluxes of 100 mJy and 50 mJy, respectively) in order to explore the error model's performance for these different flux levels. Below, we will accordingly refer to Q as P_{High} and U as P_{Low}.

The $u - v$ data in Figure 6.3 show the amplitude of the sampled Fourier Transform of the intensity distributions for each of the new test sources, with thermal noise added (i.e., one example of the Monte Carlo realisations), plotted against baseline length. As the size of the square compared to the size of the beam decreases from SQV1 to SQV4 the pattern shown by the visibility amplitude versus baseline length shifts to the right until only a small part of the Fourier Transform is successfully sampled. This corresponds to the Fourier Analysis result that features on small scales in image space correspond to high frequencies in UV space. This is manifest in Figure 6.4 as the convolved map of the model becoming in-

Table 6.1: The 4 Square-within-Square sources used to test the dependence of CLEAN errors on resolution (See Figure 6.4 for convolved models). All sources were made from the same 512 pixel CSV file with a 130 pixel wide outer square, a 60 pixel wide inner square and a 10:1 flux ratio between the inner and outer squares. Stokes I corresponds to a flux of 1 Jy, P_{High} to a flux of 0.1 Jy and P_{Low} to 0.05 Jy. The geometric mean of the beam is given. The inner square is slightly offset to the left. The ratio of the outer length to the beam for each source is also given.

Source ID	Cellsize (mas)	Outer Length (mas)	Inner Length (mas)	Beam (mas)	Outer $\frac{length}{beam}$
SQV1	0.2	26	12	2.27	11.45
SQV2	0.11	14.3	6.6	2.06	6.94
SQV3	0.065	8.5	3.9	1.91	4.45
SQV4	0.02	2.6	1.2	1.76	1.48

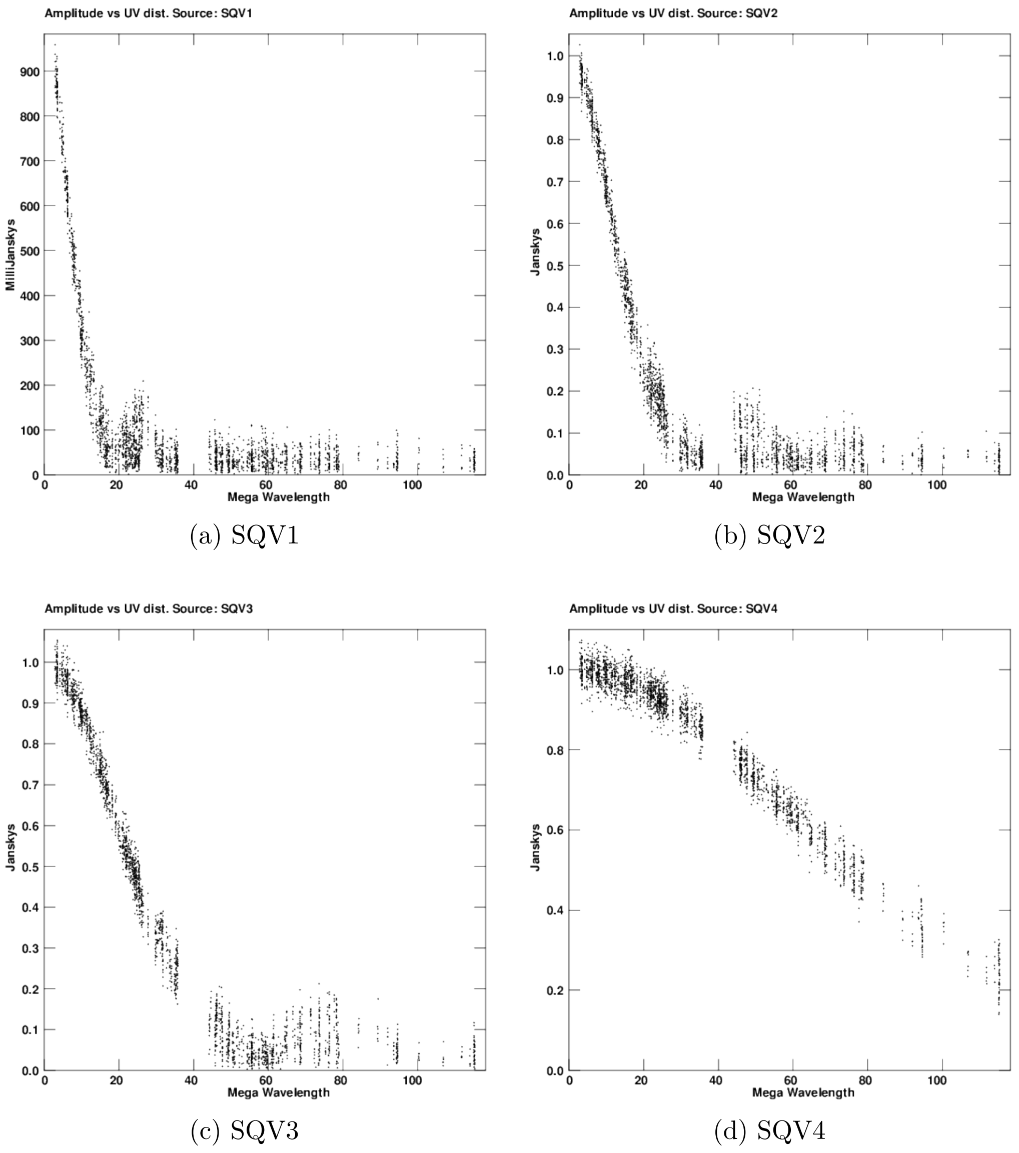
Table 6.2: CLEAN beams fitted to SQV1-SQV4. The major (BMAJ) and minor (BMIN) axes of the restoring elliptical Gaussian beam, along with the position angle (BPA). The geometric mean of the beam is also listed.

Source ID	BMAJ (mas)	BMIN (mas)	BPA (mas)	Geom. Mean (mas)
SQV1	2.37	2.18	-85.34	2.27
SQV2	2.17	1.96	-75.84	2.06
SQV3	2.02	1.80	-73.75	1.91
SQV4	1.85	1.67	-68.3	1.76

creasingly smeared out as the dimensions of the model decrease. Note that, even though the same $u - v$ distribution is used in all cases, the varying cell size in each case leads to a different CLEAN beam being fitted for each of the 4 sources. The geometric means of the beams fitted are listed in Table 6.2.

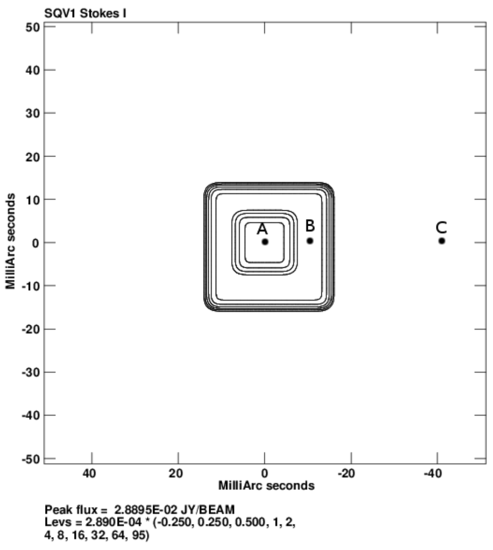
The Monte Carlo results for SQV1 in Figure 6.5 for Stokes I are very similar to the results outlined in Figure 6.2. The same tendency for higher errors in the inner region of the map is present, although the symmetry present in the Figure 6.2 is no longer present due to the slightly asymmetric design of these square-within-square sources (see Figure 6.4). The same pattern of errors is seen in P_{High} and P_{Low}, although the magnitude of the errors has changed to reflect the lower fluxes involved. The P_{High} and P_{Low} maps highlight the tendency of errors to increase at the interface between the inner and outer squares.

The calculated error maps on the right hand panels in Figure 6.5 show good agreement with the Monte Carlo maps for all Stokes parameters with a constant f value equal to 0.75 Beam. This suggests that the dependency of the f parameter

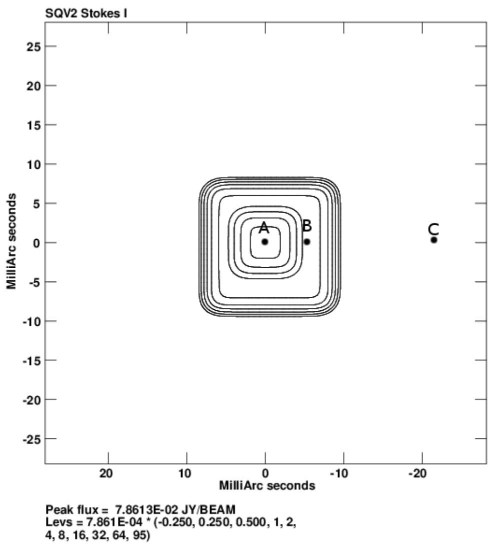


(a) SQV1
(b) SQV2
(c) SQV3
(d) SQV4

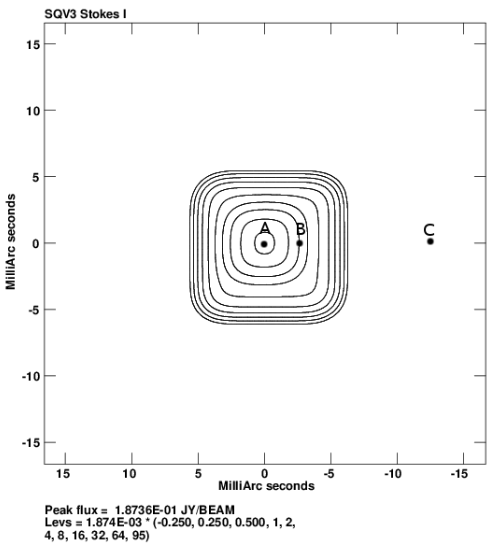
Figure 6.3: Visibility amplitude (including thermal noise) versus baseline length for each of the square-in-square models used to investigate flux and resolution dependence of the errors. Note that the UV function appears to be stretched to the right as increasingly smaller model sources are sampled with the same UV distribution. By SQV4 much of the UV data goes unsampled. Thermal noise is present in each UV distribution to an extent designed to give realistic RMS noises in the final images.



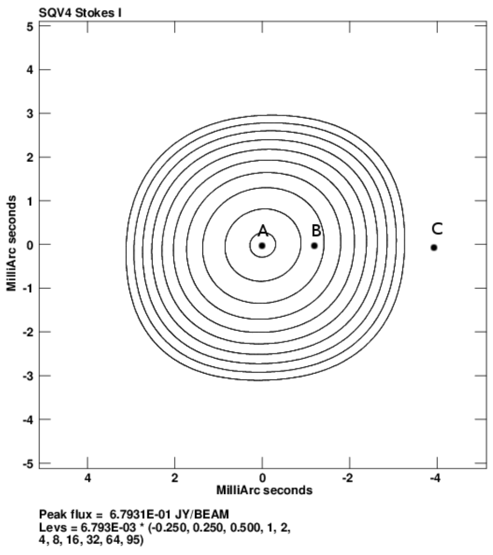
(a) SQV1



(b) SQV2



(c) SQV3



(d) SQV4

Figure 6.4: Convolved maps of square-within-square model sources. Detail is lost as the overall dimensions of the model source are decreased. Note that the inner square is offset to the left. By SQV4 the convolved model appears to be approximately Gaussian. The locations marked in the map of SQV1 are used in testing Section 6.4.1.1.

on flux is fairly weak, although the maximum errors in individual compact regions in the maps would require a higher value of f to describe them, as was also the case in the Monte Carlo studies described in the previous section. All of the calculated error maps reproduce to some degree the tendency of the Monte Carlo errors to increase at the interfaces between the inner and outer squares and the outer square and the off-source region. There is a trade off in choosing an f value that describes both the maximum and average errors well in a given region. This suggests that the error model proposed in Section 6.2 only approximately holds, though it may represent a step forward from modelling the error as a constant value as in previous analyses.

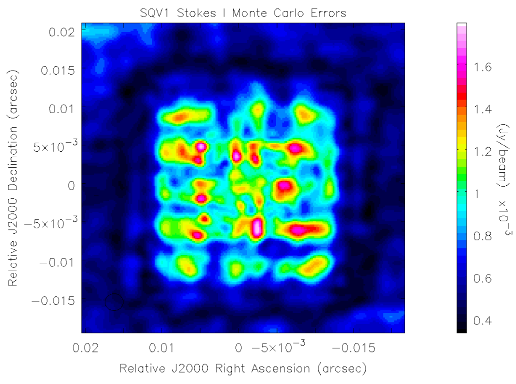
Figure 6.6 is laid out in the same way for SQV2. The Monte Carlo results on the left side of the image appear broadly similar to those for SQV1, however the decreasing source size is evident in the way that the ‘perimeter’ of high errors at the interface between in the inner and outer squares visible for P_{High} and P_{Low} in SQV1 has changed to a central blob of errors in SQV2. The calculated errors do a fairly good job of describing the overall errors with an f value equal to 0.5 Beam for all Stokes values, although the agreement is not as good as in the previous cases considered. Systematic differences are also beginning to be present, as the Monte Carlo errors peak at the center of the inner square, while the calculated errors do not show this behaviour. As the effective size of SQV2 is almost half that of SQV1 the change in the value of f from 0.75 Beam to 0.5 Beam is likely due to the difference in ratio of the size of the source to the beam (see Table 6.1).

Figure 6.7 shows the Monte Carlo and calculated error results for SQV3. The two distinct regions of high errors corresponding to the inner and outer squares are visible in the Stokes I and P_{High} Monte Carlo maps but are less distinct in the P_{Low} map. There continues to be a good general correspondence between the Monte Carlo error maps and the calculated error maps for all three Stokes parameters. The calculated maps do not describe the detailed errors, but continue to approximately describe the regions of high error within the fluctuations of the background noise. A single value for f of 0.3 Beam yields good results for all three Stokes parameters.

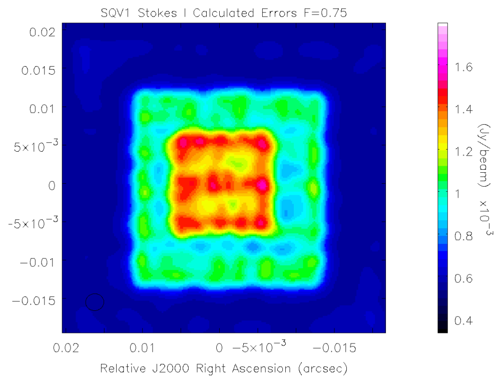
The final set of results for SQV4 are contained in Figure 6.8. This was the smallest of the 4 models with an outer length of just 2.6 mas compared to a beam with a geometric mean of 1.76 mas (see Table 6.2). There is a significant change in the Monte Carlo error maps at this resolution, with no systematic increase in error in the on-source region at all being observed for P_{High} and P_{Low}, while the increase in Stokes I is much narrower than the model in the right panel would predict. Thermal (random) uncertainties dominate the P_{High} and P_{Low} Monte Carlo maps and there is no evidence of the patterns predicted in the corresponding calculated error maps. While an f value of $f = 0.2$ Beam was used for all Stokes parameters, only the Stokes I map shows any correspondence with the Monte Carlo simulations, and even then the agreement is poor. This could be interpreted as evidence for a flux dependency in the value of f at small scales, however as no sign of such a dependency was evident at large scales it is more probable that the CLEAN algorithm is simply doing a better job of imaging smaller structures than the error model predicts. The performance of CLEAN should increase as the source being imaged more closely resembles a δ function and, as the Fourier transform of a δ function is a Gaussian, the UV visibilities for SQV4 in Figure 6.3 suggest that SQV4 appears fairly close to a δ function to the UV dataset used in the simulated observations. The symmetric error pattern visible in the Stokes I Monte Carlo error map in Figure 6.8 is likely the result of an interaction between the source and the dirty beam and is occasionally visible in real observations. Errors from this effect are distributed symmetrically about the map and do not show any correspondence with CLEAN component location.

Figure 6.9 shows some results for the (a) peak error, (b) mean overall error, (c) mean core error rms error for the SQV series of sources. Note that some care should be taken in examining Figures 6.10 and 6.10 as the straight lines joining each successive data point are not intended to suggest a linear relationship, but rather to highlight the apparent trend in the data. It is unlikely that the underlying relationship between the points is linear in nature. As the error at any point varies with source and Stokes parameter the mean errors have been corrected for this variation by dividing by the root-mean-square (RMS) error corresponding to the map. This means any variation in peak error, total mean error or mean error in the core region visible in Figure 6.9 is independent of beam effects and due to source size alone. Note that this treatment of the errors factors out the increase in RMS error that can be associated with interference due to source structure on scales less than but close to the CLEAN beam (visible in the RMS error plot).

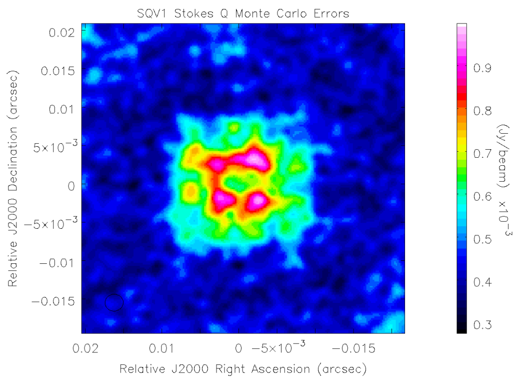
The plots of the peak, mean and mean core errors as a function of source size shown in Figure 6.9a–c show the decrease in the on-source error with the size of the source. The third plot restricts the analysis to the central (inner square) region of the SQV sources and appears to show the clearest trends – namely



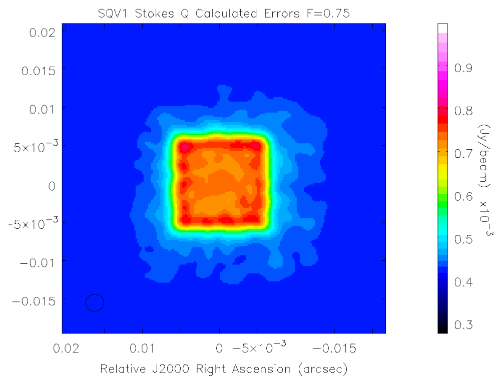
(a) Stokes I Monte Carlo Errors



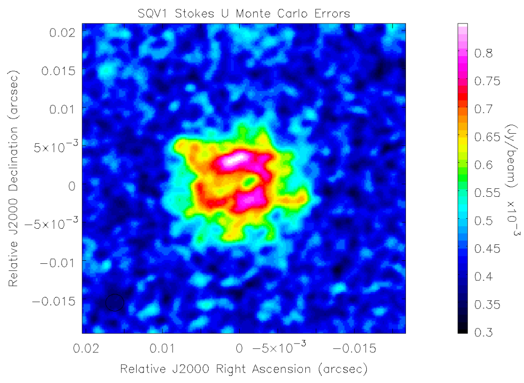
(b) Stokes I Calculated Errors



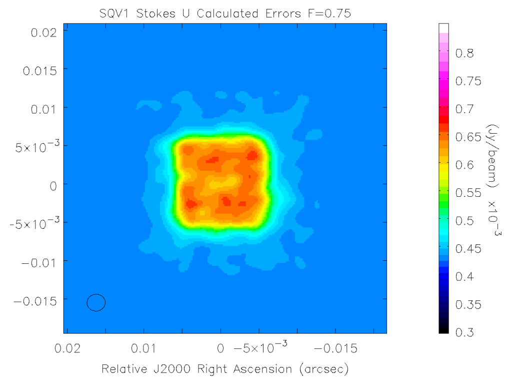
(c) P_{High} Monte Carlo Errors



(d) P_{High} Calculated Errors

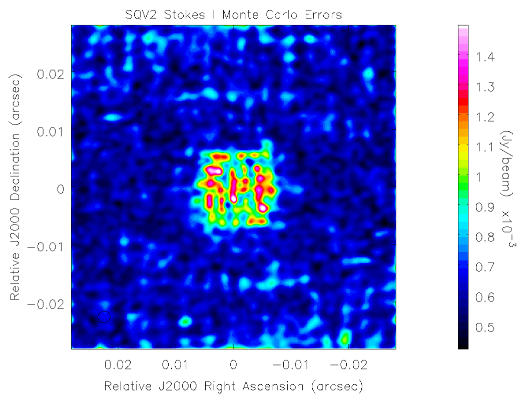


(e) P_{Low} Monte Carlo Errors

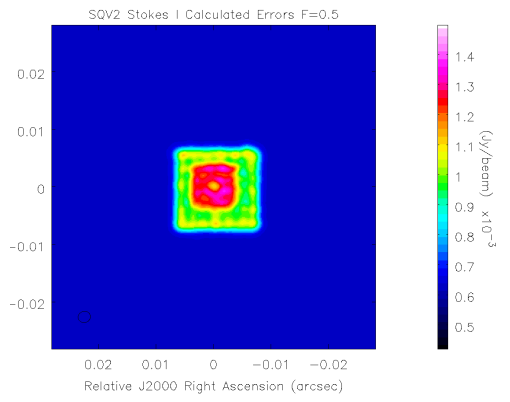


(f) P_{Low} Calculated Errors

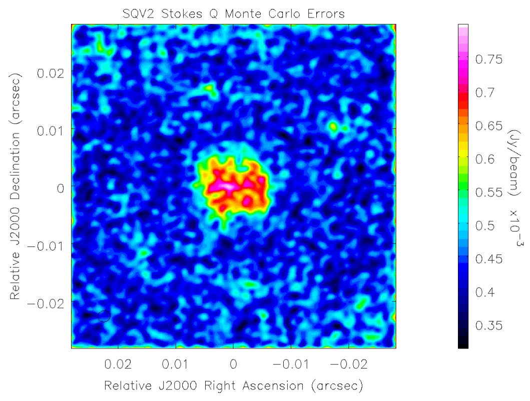
Figure 6.5: Monte Carlo and calculated error maps for SQV1. $f = 0.75$ Beam for all Stokes parameters. The same colour scale has been used in both the Monte Carlo error map and the calculated error map for each Stokes parameter, however the colours have been slightly altered for P_{High} and P_{Low} to highlight the pattern in the calculated error maps. The flux levels in the inner square for Stokes I, P_{High} and P_{Low} are 0.029 Jy, 0.0029 Jy and 0.0015 Jy respectively. The flux levels in the outer square are approximately a tenth of the inner square.



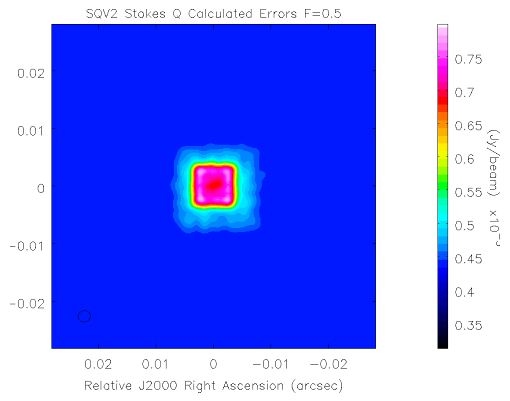
(a) Stokes I Monte Carlo Errors



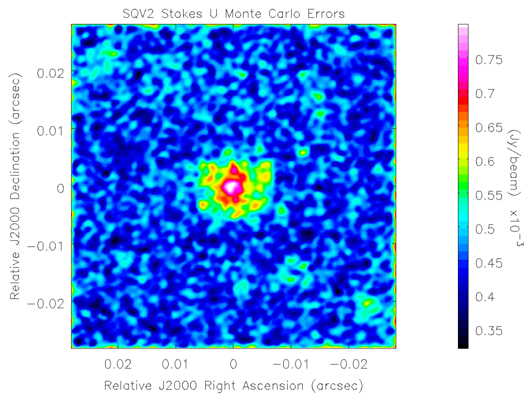
(b) Stokes I Calculated Errors



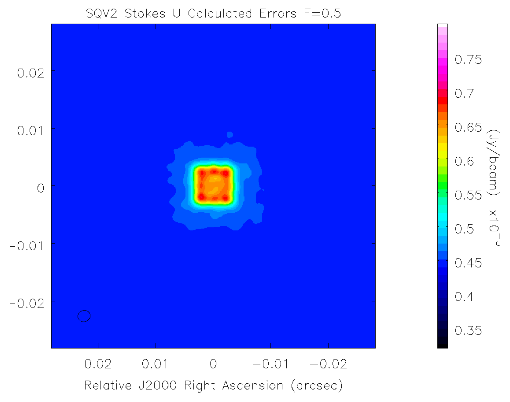
(c) P_{High} Monte Carlo Errors



(d) P_{High} Calculated Errors

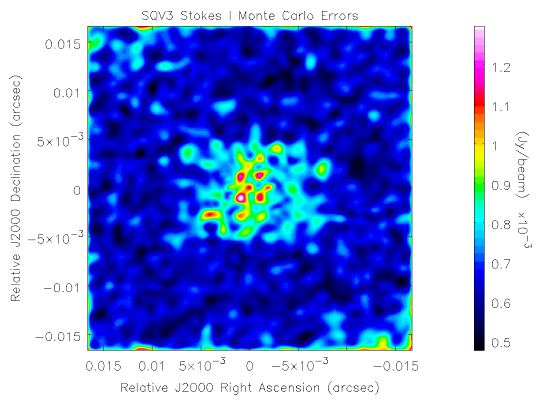


(e) P_{Low} Monte Carlo Errors

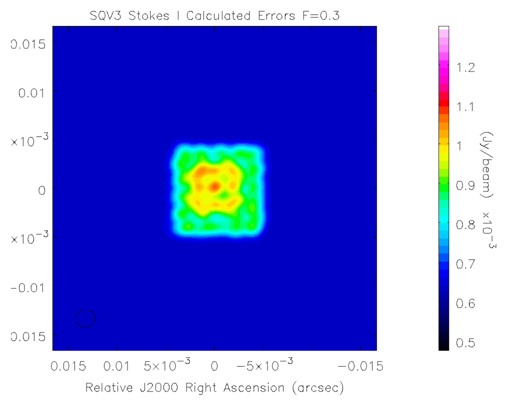


(f) P_{Low} Calculated Errors

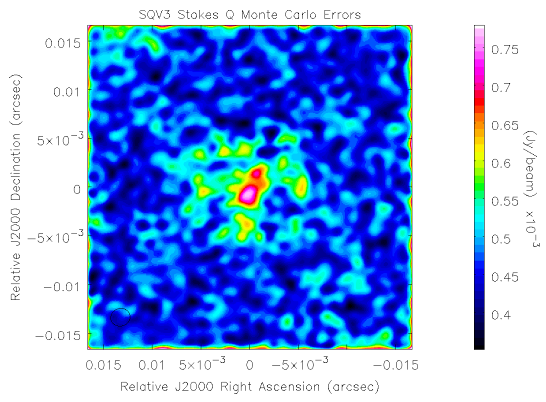
Figure 6.6: Monte Carlo and calculated error maps for SQV2. $f = 0.5$ Beam for all Stokes parameters. The same colour scale has been used in both the Monte Carlo error map and the calculated error map for each Stokes parameter. The flux levels in the inner square for Stokes I, P_{High} and P_{Low} are 0.079 Jy, 0.0079 Jy and 0.0039 Jy respectively. The flux levels in the outer square are approximately a tenth of the inner square.



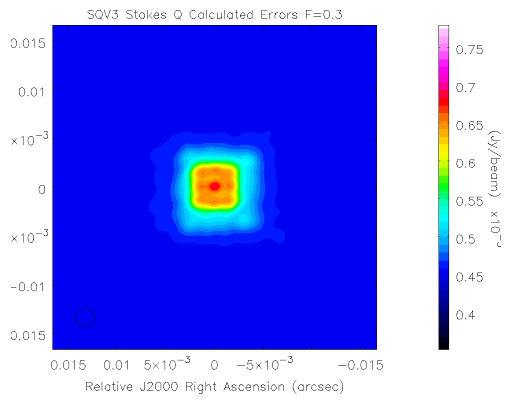
(a) Stokes I Monte Carlo Errors



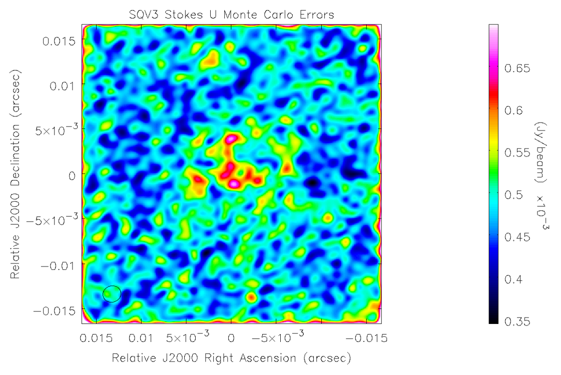
(b) Stokes I Calculated Errors



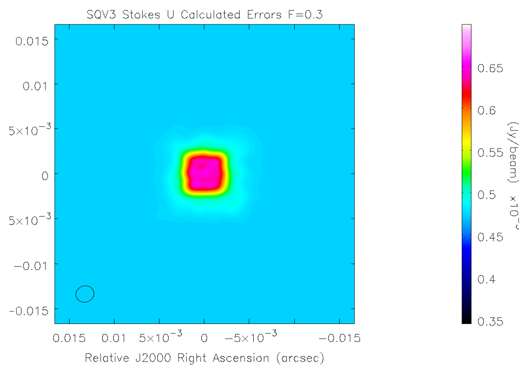
(c) P_{High} Monte Carlo Errors



(d) P_{High} Calculated Errors

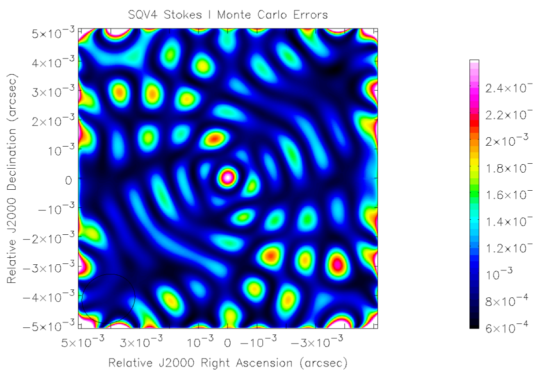


(e) P_{Low} Monte Carlo Errors

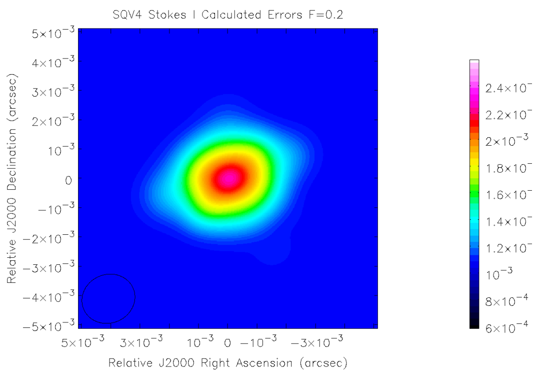


(f) P_{Low} Calculated Errors

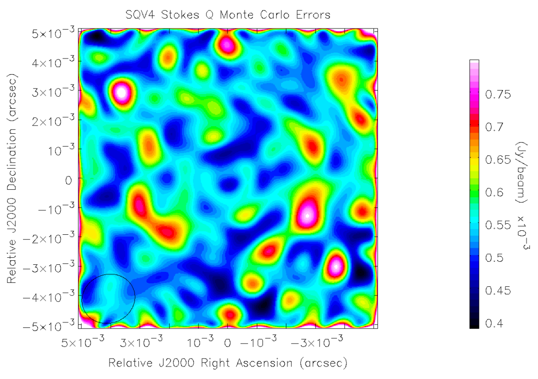
Figure 6.7: Monte Carlo and calculated error maps for SQV3. $f = 0.3$ Beam for all Stokes parameters. The same colour scale has been used in both the Monte Carlo error map and the calculated error map for each Stokes parameter. The flux levels in the inner square for Stokes I, P_{High} and P_{Low} are 0.19 Jy, 0.019 Jy and 0.009 Jy respectively. The flux levels in the outer square are approximately a tenth of the inner square.



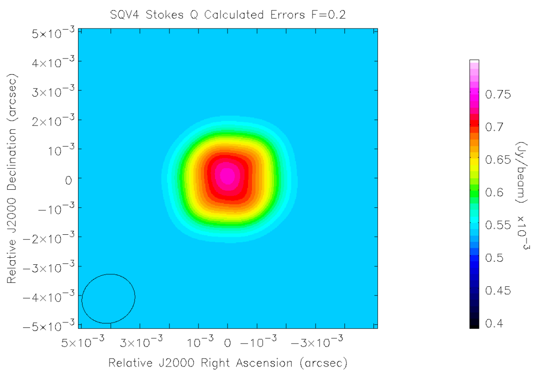
(a) Stokes I Monte Carlo Errors



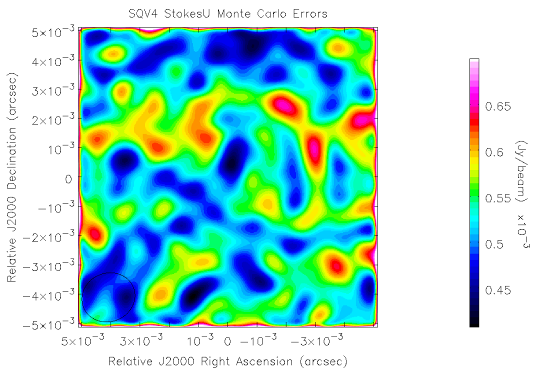
(b) Stokes I Calculated Errors



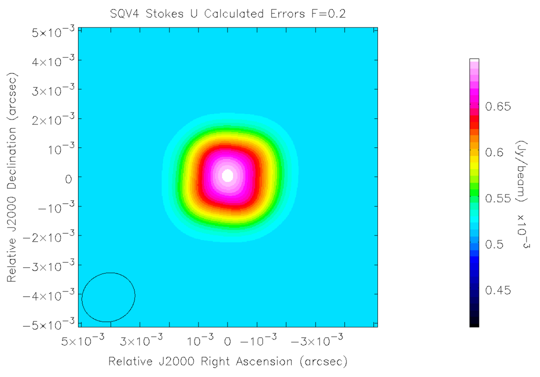
(c) P_{High} Monte Carlo Errors



(d) P_{High} Calculated Errors



(e) P_{Low} Monte Carlo Errors



(f) P_{Low} Calculated Errors

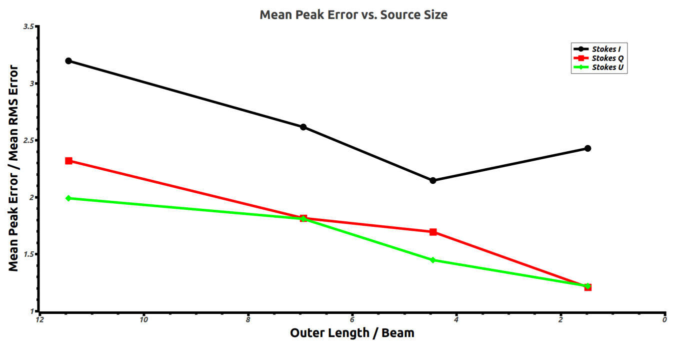
Figure 6.8: Monte Carlo and calculated error maps for SQV4. $f = 0.2$ Beam for all Stokes parameters. The same colour scale has been used in both the Monte Carlo error map and the calculated error map for each Stokes parameter. The flux levels in the inner region for Stokes I, P_{High} and P_{Low} are 0.68 Jy, 0.068 Jy and 0.034 Jy respectively.

that the on-source error is higher for higher fluxes (Stokes $I > P_{High} > P_{Low}$) and that in all cases the error decreases with the size of the source. The on-source uncertainty of approximately 1.8 times the rms implied by the Equation (6.5) proposed by Hovatta et al. (2012) for Stokes Q and U in the absence of appreciable uncertainties due to residual D-terms is fairly close to the factors seen in the plot of the peak errors for Stokes $Q$and U in Figure 6.9. The fact that the mean errors in the source region (particularly the core source region) are lower than this suggest that the uncertainty levels proposed by Hovatta et al. (2012) may be on the conservative side, especially for compact sources.

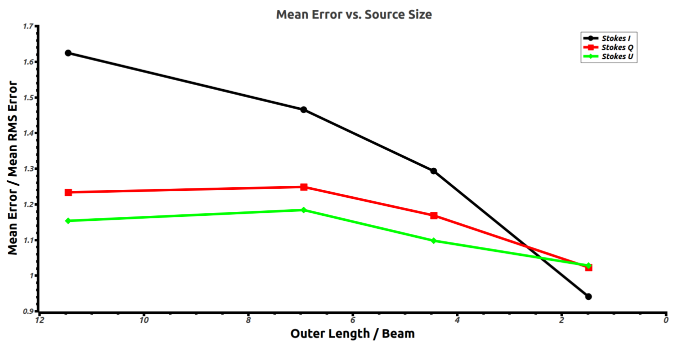
Figure 6.10 shows the dependence of f on the size of the source. The values of f are approximate and were selected by hand, however it is clear that successively smaller values of f are needed to model smaller sources. Though the plot appears to suggest an approximate relationship between the value of f for various source sizes, the fact that the lowest value of f used in SQV4 failed to describe the errors for P_{High} and P_{Low} suggests that the use of a value of f smaller than 0.2 may be appropriate if the error method is to be applied to compact sources; it is also possible that the approach used in this error model breaks down in the case of sufficiently compact emission that is described very well by the CLEAN deconvolution. As the unit of f is Beam, this implies that f is a value partly defined by the size of the convolving beam used in a particular map. As indicated in Table 6.2, the beam varies by a small but significant amount between the four model sources used in the Monte Carlo testing. Thus, the choice of an appropriate value of f will also depend on the size of the convolving beam, regardless of the intrinsic size of the source.

6.4 Calculations with the new error model

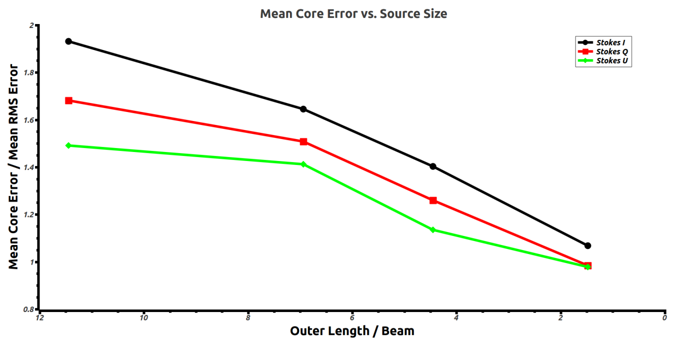
To briefly summarise the results of the Monte Carlo tests described in Sections 6.3.1 and 6.3.2, the proposed error model given by Equation (6.10) is in good overall agreement with the results of Monte Carlo simulations of the uncertainties in individual map pixels when the emission region is relatively large compared to the beam size; i.e., when the source is not too compact to the available resolution. In some cases the agreement in details is striking, indicating that the proposed error model correctly represents at least some component of the overall uncertainties in CLEAN images. However, the model appears to break down in regions of emission that are relatively compact; whether it is possible to modify



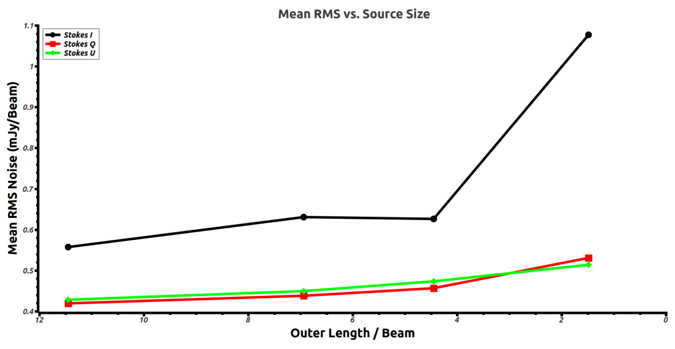
(a) Peak Error vs. Source Size



(b) Mean Error vs. Source Size

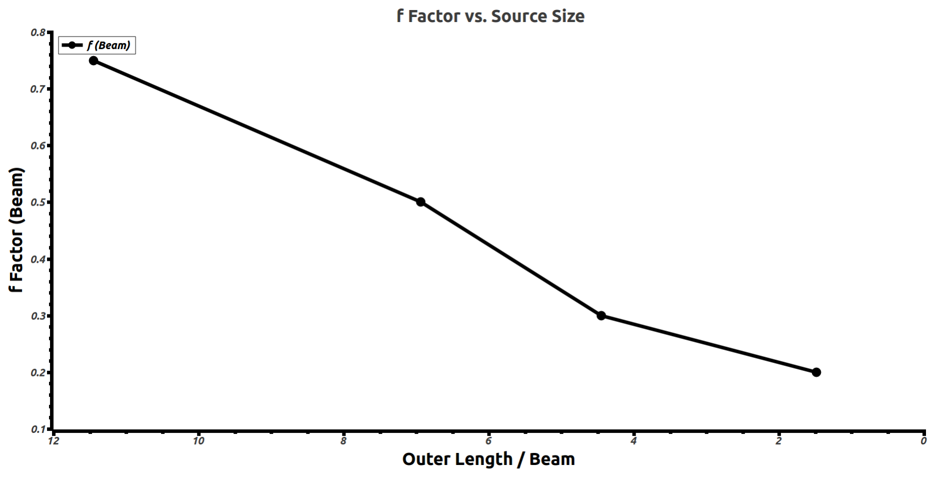


(c) Mean Core Error vs. Source Size



(d) RMS Error vs. Source Size

Figure 6.9: The dependence of the errors in the CLEAN algorithm on resolution for the SQV series of sources (source size given as the ratio of the length of the outer square to the geometric beam of the restoring beam). Note that the straight lines used to connect the data are intended purely to highlight the data trends.



(a) f vs. Source Size

Figure 6.10: The dependence of the f factor on source size for the SQV series of sources (source size given as the ratio of the length of the outer square to the geometric beam of the restoring beam). Note that the values of f used were approximate (a best-fit by eye) and that the straight lines used to connect the data are intended purely to highlight the data trends and are not representative of actual data. The actual values in the intermediate regions are unmeasured.

the error model to better describe the uncertainties in compact regions is a subject for future study. Because many regions imaged in VLBI maps of AGN jets are, in fact, quite compact, the error model is probably not directly applicable to such images in its current form, although it may be suitable for more extended regions such as those imaged on larger (kiloparsec) scales.

When applicable, the error model enables a much fuller treatment of the errors in individual pixels and the correlations between them. The following sections explore this in some detail.

6.4.1 Taking the average of correlated variables

It is standard practice in radio astronomy to create maps with a cell size appreciably smaller than the beamwidth. This helps ensure that features in the data are present in the map, and that the dirty beam is correctly reproduced. This means that a pixel is typically much smaller than a resolution element, making it reasonable to consider averages over some number of neighbouring pixels when estimating local values in maps. Complications can arise in finding the error associated with that average if the variables being averaged are correlated with each other. This is true when taking the average of a region in a CLEAN map, as every pixel is dependent on the same variables (the CLEAN components). As the

CLEAN beam falls off quite slowly this means that nearby pixels in the CLEAN map are highly correlated, and ignoring this correlation in calculating the error of the average would result in a significant underestimation of the error.

The equation for the propagation of error in a function f of n dependent variables x_i with uncertainties σ_{x_i}, where $x_i = g_i(y_k)$ and y_k are m independent variables with uncertainty σ_y is

$$\sigma_f^2 = \sum_{i=1}^{n}(\frac{\partial f}{\partial x_i})^2\sigma_{x_i}^2 + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\frac{\partial f}{\partial x_i}\frac{\delta f}{\delta x_j}cov(x_i, x_j) \tag{6.11}$$

where the $cov(x_i, x_j)$, the covariance matrix, is defined by

$$cov(x_i, x_j) = \sum_{k=1}^{m}\frac{\partial g_i}{\partial y_k}\frac{\partial g_j}{\partial y_k}\sigma_{y_k}^2. \tag{6.12}$$

A simple unweighted average over n pixels where the pixel values Q_i are each dependent on every CLEAN component q_i as described by equations [6.2] and [6.3] can be expressed as

$$\overline{Q} = \frac{1}{n}\sum_{i=1}^{n}Q_i. \tag{6.13}$$

Equation (6.11) yields the following expression for the uncertainty in the average

$$\sigma_{\overline{Q}}^2 = \frac{1}{n^2}(\sum_{i=1}^{n}\sigma_{Q_i}^2 + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\sum_{k=1}^{m}B_{ki}B_{kj}f^2\sigma_{rms}^2). \tag{6.14}$$

Note that including the correlation effects in the CLEAN algorithm guarantees a bigger error than simply ignoring them. The weighted average is defined as follows

$$\overline{Q} = \sum_{i=1}^{n}w_iQ_i \tag{6.15}$$

$$w_i = \frac{1}{W}\sum_{i=1}^{n}\frac{1}{\sigma_i^2} \tag{6.16}$$

where w_i are the normalised weights and W is the sum of the unnormalised weights, $\frac{1}{\sigma_i^2}$. The uncertainty in the weighted average can be found using the same method as for the unweighted average, yielding

$$\sigma_{\bar{Q}}^2 = \sum_{i=1}^{n} w_i^2 \sigma_{Q_i}^2 + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} w_i w_j \sum_{k=1}^{m} B_{ki} B_{kj} f^2 \sigma_{rms}^2. \tag{6.17}$$

6.4.1.1 Test of the validity of the Error Method

The method of calculating the error of of an unweighted mean of all the pixels in a region of the map given by Equation (6.14) can be used to test the validity of the error method. If the error method accurately takes into account the correlation between different pixels, the predicted uncertainty in the mean should be close to the uncertainty calculated using a Monte Carlo Method. New code was written to take average values of the simulated observations of the four simulated sources generated in Section 6.3.2 in 3×3 and 9×9 pixel regions centred at the locations marked in Figure 6.4. The code then compared each of the average values against the known true average value in the corresponding region in the convolved model maps. The root mean square of the deviations was calculated as a measure of the uncertainty in the average at each region. The 3×3 regions are typical of small averaging operations useful in increasing the signal to noise ratio of a quantity on the map while still representing data from the same region, while the use of a 9×9 region resulted in 81 correlated data points, ensuring that an inaccurate treatment of the correlation should cause a major disagreement between the error method and the results from the Monte Carlo simulations. The results are presented in Tables 6.3 to 6.5. Note that Stokes I represents flux levels typical of total intensity observations, while P_{High} and P_{Low} represent fluxes typical of the linearly polarised component of the emission.

Table 6.3 gives the results for 3×3 and 9×9 pixel regions around the core of each of the model sources. It is clear that the measured uncertainty in the mean value of the region is significantly higher than the off-source RMS noise in almost all cases. If the pixels were truly independent the mean of a 9×9 region (81 samples) should have an uncertainty of $\frac{1}{9}$ that of the uncertainty in a single pixel. The fact that the uncertainty of the mean is generally higher than the RMS noise (which is of the order of the error in a single pixel) demonstrates strongly that correlation cannot be ignored in any statistical analysis of a CLEAN map. The Monte Carlo errors in the 9×9 pixel region are slightly smaller than those for the corresponding 3×3 pixel region. This is as expected for a highly correlated set of data – increased sampling does reduce uncertainty, but at a slow rate. The errors predicted using Equation (6.14) are in general close to the actual errors

Table 6.3: Error in unweighted average of core regions in SQV1 to SQV4. The core region used (point A) can be seen in the maps of each source in Figure 6.4. Averages were taken over 3 × 3 and 9 × 9 pixel regions. The Monte Carlo RMS errors are shown, together with the errors predicted using the new error model and the average off-source RMS noise.

Region A	σ_{rms} (mJy)	MC Error 3×3 (mJy)	Pred. Error 3×3 (mJy)	MC Error 9×9 (mJy)	Pred. Error 9×9 (mJy)
Stokes I					
SQV1	0.56	1.19	1.30	1.03	1.14
SQV2	0.63	1.33	0.96	1.22	0.91
SQV3	0.63	0.84	0.87	0.78	0.83
SQV4	1.61	1.52	2.99	1.47	2.93
P_{High}					
SQV1	0.42	0.60	0.62	0.56	0.54
SQV2	0.44	0.75	0.54	0.71	0.51
SQV3	0.46	0.70	0.53	0.68	0.50
SQV4	0.55	0.49	0.56	0.49	0.53
P_{Low}					
SQV1	0.43	0.65	0.44	0.58	0.38
SQV2	0.45	0.78	0.49	0.74	0.45
SQV3	0.47	0.58	0.47	0.56	0.44
SQV4	0.53	0.47	0.52	0.47	0.49

and, critically, do not show the rapid drop in uncertainty with increasing sample size that would be seen if the model did not successfully describe the correlation in neighbouring pixels. The agreement is especially good for the Stokes I results, and appears to diminish in quality for P_{High} and P_{Low}.

It is of note that the performance of the method improves for P_{High} and P_{Low} as the model source shrinks in size – this is in contrast to the maps of the error distribution presented in Figures 6.7 and 6.8. Thus, while the value of f may be so low for SQV3 and SQV4 that the random background noise dominates the error maps, the error model does appear to successfully model the correlation in nearby pixels due to the CLEAN algorithm. These results show that it may be justifiable to apply the model to compact sources with a suitable value for f close to that used for SQV4.

Table 6.4 shows the results for 3 × 3 and 9 × 9 pixel regions around point B as labelled in Figure 6.4 for SQV1 to SQV4. The general trend is similar to that for point A, where the predicted error in Stokes I (high flux) matches up well with the observed errors, while there is a disagreement in P_{High} and P_{Low} that becomes

Table 6.4: Error in unweighted average of point B in SQV1 to SQV4. Point B can be seen in the maps of each source in Figure 6.4. Averages were taken over 3 × 3 and 9 × 9 pixel regions. The Monte Carlo RMS errors are shown, together with the errors predicted using the new error model and the average off-source RMS noise.

Region B	σ_{rms} (mJy)	MC Error 3×3 (mJy)	Pred. Error 3×3 (mJy)	MC Error 9×9 (mJy)	Pred. Error 9×9 (mJy)
Stokes I					
SQV1	0.56	0.89	0.82	0.82	0.72
SQV2	0.63	1.18	0.75	1.14	0.69
SQV3	0.63	0.73	0.62	0.70	0.57
SQV4	1.61	2.43	2.32	2.40	2.26
P_{High}					
SQV1	0.42	0.40	0.17	0.36	0.09
SQV2	0.44	0.62	0.25	0.58	0.19
SQV3	0.46	0.46	0.27	0.44	0.23
SQV4	0.55	0.54	0.46	0.54	0.43
P_{Low}					
SQV1	0.43	0.39	0.15	0.32	0.06
SQV2	0.45	0.53	0.17	0.45	0.09
SQV3	0.47	0.56	0.22	0.54	0.15
SQV4	0.53	0.58	0.41	0.57	0.37

less pronounced as the source becomes more compact. The disagreement between the uncertainty in the mean predicted by the method for large scale sources and the measurements from the Monte Carlo analysis is somewhat surprising, as the error maps in Figures 6.5 and 6.6 appear to show reasonable agreement for single pixel errors. The results for P_{Low} also show a marked reduction in quality between the 3 × 3 and 9 × region size for the larger sources – though as in region A the poor results for P_{High} and P_{Low} appear to improve slightly for compact sources.

Although the agreement between the model and the simulations in region B improves for more compact sources, it falls far short of the level of agreement seen in region A for SQV4. An examination of the final set of results for region C in Table 6.5 suggests why – the error method does not successfully describe the behaviour of correlation in an off-source region. As outlined in Section 6.2, the error method models the background error, σ_{rms}, as being completely random – therefore taking 81 pixels with an individual certainty of σ_{rms} should yield an uncertainty of $\frac{\sigma_{rms}}{9}$ in the resulting mean. The actual uncertainties measured are in general smaller than σ_{rms}, but not by much. As region C is far from any

Table 6.5: Error in unweighted average of point C in SQV1 to SQV4. Point C can be seen in the maps of each source in Figure 6.4. Averages were taken over 3 × 3 and 9 × 9 pixel regions. The Monte Carlo RMS errors are shown, together with the errors predicted using the new error model and the average off-source RMS noise.

Region C	σ_{rms} (mJy)	MC Error 3×3 (mJy)	Pred. Error 3×3 (mJy)	MC Error 9×9 (mJy)	Pred. Error 9×9 (mJy)
Stokes I					
SQV1	0.56	0.40	0.19	0.36	0.06
SQV2	0.63	0.49	0.21	0.45	0.07
SQV3	0.63	0.62	0.21	0.60	0.07
SQV4	1.6	0.71	0.54	0.70	0.18
P_{High}					
SQV1	0.42	0.36	0.14	0.29	0.05
SQV2	0.44	0.38	0.15	0.35	0.05
SQV3	0.46	0.48	0.15	0.38	0.05
SQV4	0.55	0.56	0.18	0.55	0.06
P_{Low}					
SQV1	0.43	0.37	0.14	0.30	0.05
SQV2	0.45	0.40	0.15	0.37	0.05
SQV3	0.47	0.39	0.16	0.38	0.05
SQV4	0.53	0.50	0.18	0.50	0.06

CLEAN components this behaviour strongly indicates the presence of a correlation term in the noise completely independent of the CLEAN algorithm and likely the result of a systematic correlations introduced by a combination of the physical observing mechanism and electronics, and the calibration process.

It is outside the scope of the error method developed in this chapter to deal with such correlations; however the problem is largely confined to off source regions. This effect may cause the error method to give an inaccurate estimate for the error in regions of low signal to noise, where the contribution of background noise to the total error in a pixel is comparable to the contribution due to CLEAN component related errors. This may be evident in the slightly lower quality results for P_{High} and P_{Low} in region B compared to region A, as the CLEAN components are grouped more tightly around region A and region B experiences a larger proportion of its noise from the addition of the residual map as described in Section 6.2.

In any case, it is clear that the errors predicted for averaging over some number of neighbouring pixels cease to reproduce the corresponding Monte Carlo errors well when the flux levels in the regions sampled become relatively low. This makes this approach of only limited use when applied to VLBI polarisation images.

A final result from this analysis is the suitability of σ_{rms} as a measure of the uncertainty in averaged flux values for compact sources with low flux. Although the results presented in Sections 6.3.1 and 6.3.2 demonstrated that σ_{rms} is often a poor estimate for the single pixel uncertainty in a source, it appears that by taking an average over a number of pixels the error quickly becomes approximately equal to the background rms fluctuations. Thus while the method described in this Chapter or the approximation of Hovatta et. al (2012) may be necessary to describe a single pixel error, it may be faster and just as accurate to estimate the uncertainty in the mean of a number of pixels as σ_{rms}.

6.4.2 Error analysis of a data slice

When investigating various types of jet intensity and polarisation structure, it can at times be useful to take a slice of an image in a particular direction of interest, e.g. either across or along the jet direction. Such slices are important in the study of transverse gradients in Faraday rotation measure (see Chapter 5) and in the fitting of helical magnetic field profiles to observed jets (see, for example, Murphy et al. 2013). As the taking of a slice of an image often includes the use of bilinear interpolation any scheme which extends the error method outlined in Section 6.2 to the taking of slices needs to take this into account. The following section outlines the bilinear interpolation technique and suggests how the error model outlined in this chapter can be applied to it.

6.4.2.1 Bilinear Interpolation

Consider the calculation of a value in an image that does not lie exactly on a pixel. Figure 6.11 shows a point P which is located partly between the pixels Q_{11}, Q_{12}, Q_{21} and Q_{22}. The method of bilinear interpolation calculates a value for the image at P by first interpolating values for the image at points R_1 and R_2 which have the same x coordinate as P, and then interpolating a value for P using R_1 and R_2 (see, for example *Numerical Recipes in C*). Taking a function f to represent the image value, this scheme can be described as follows:

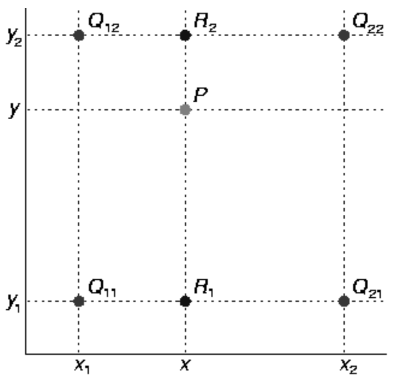


Figure 6.11: Bilinear interpolation labelling scheme. The Q points indicate actual pixels, the point P is the point being interpolated and R_1 and R_2 are the intermediate interpolation points. Image credit: Wikimedia Commons.

$$f(R_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \tag{6.18}$$

$$f(R_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \tag{6.19}$$

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2). \tag{6.20}$$

Noting that in an image the pixel grid ensures that $x_2 - x_1 = y_2 - y_1 = 1$, the following expression for $f(P)$ is obtained

$$\begin{aligned} f(P) = {} & (x_2 - x)(y_2 - y) f(Q_{11}) + (x - x_1)(y_2 - y) f(Q_{21}) \\ & + (x_2 - x)(y - y_1) f(Q_{12}) + (x - x_1)(y - y_1) f(Q_{22}) \end{aligned} \tag{6.21}$$

which can be written in the form

$$f(P) = \sum_{i=1}^{4} w_i f(Q_i) \tag{6.22}$$

where w_i represent the weights applied to the contribution of each of the four pixels to the interpolated value.

6.4.2.2 Error analysis of an interpolated slice

Section 6.1 described how a pixel final CLEAN map could be expressed as the convolution of all the CLEAN components evaluated at that pixel with a residual value added in

$$I_k = \sum_{l=1}^{n} B_{lk} i_l + R_k. \tag{6.23}$$

Therefore the value of the interpolated pixel can be written

$$f(P) = \sum_{i=1}^{4} w_i (\sum_{l=1}^{n} B_{lk} i_l + R_k). \tag{6.24}$$

Using Equation (6.12) the covariance matrix due to variations in the CLEAN components alone can be written as

$$cov(f(P_i), f(P_j))_{CC} = \sum_k \frac{\partial f(P_i)}{\partial q_k} \frac{\partial f(P_j)}{\partial q_k} \sigma q_k^2. \tag{6.25}$$

Using the notation

$$\frac{\partial f(P_i)}{\partial q_k} = \sum_{l=1}^{4} w_{il} h_{kl}, \tag{6.26}$$

where w_{il} is the weight corresponding to interpolated point i, pixel l, this becomes

$$cov(f(P_i), f(P_j))_{CC} = \sum_k (\sum_{l=1}^{4} w_{il} h_{kl})(\sum_{l=1}^{4} w_{jl} h_{kl}) \sigma q_k^2. \tag{6.27}$$

Equation (6.27) gives the covariance matrix of $f(P)$ due to errors in the CLEAN components alone. To calculate the final covariance matrix a term describing the residuals must be added. As the residuals are very small, the correlation between neighbouring interpolated points sharing the same residuals may be ignored, yielding a final equation of

$$cov(f(P_i), f(P_j)) = cov(f(P_i), f(P_j))_{CC} + \delta_{i,j} \sigma_{rms}^2. \tag{6.28}$$

Alternatively, the correlation may be taken into account using Equation (6.12), yielding

$$cov(f(P_i), f(P_j)) = cov(f(P_i), f(P_j))_{CC} + \sigma^2_{rms} \sum_{k=1}^{4} w_{ik} w_{jk'}, \tag{6.29}$$

where k and k' refer to the same pixel (which may have a different index for for different interpolated points).

6.4.2.3 Fitting a model to an interpolated slice

The following method for attempting to correct for correlation when performing a χ^2 based fit of a model to a data slice was developed with Eoin Murphy. The method was used to make fits similar to those in Murphy et. al 2013.

The standard χ^2 method of fitting a model to data is the search for a model that minimises the function

$$\chi^2 = (d_i - m_i)V^{-1}_{ij}(d_j - m_j), \tag{6.30}$$

where d_i corresponds to the i^{th} element of the data, m_i to the corresponding model value and V is the covariance matrix of the data. The quality of the resulting fit (and the statistical likelihood of rejecting the null hypothesis) can normally be determined from the χ^2 value corresponding to the best fit for the given degrees of freedom (DOF). However if this technique is used to find the best fitting model to a set of data points which are correlated with each other the resulting χ^2 value may indicate a much higher quality fit than is truly the case. This can be understood by considering the χ^2 value of a linear fit to two data points. When judged by eye the fit will appear to be perfect, but statistically the hypothesis that the data series is not linear cannot be rejected. However if the two points were interpolated to two hundred points, the apparent statistical quality of the fit would be artificially inflated.

In order to correct for this a correction to the degrees of freedom value was developed such that

$$DOF_{corrected} = \frac{DOF}{G}, \tag{6.31}$$

where G is a correction equal to the sum of the entire Pearson correlation matrix divided by the number of variables

$$G = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} V_{ij}/V_{ii}. \tag{6.32}$$

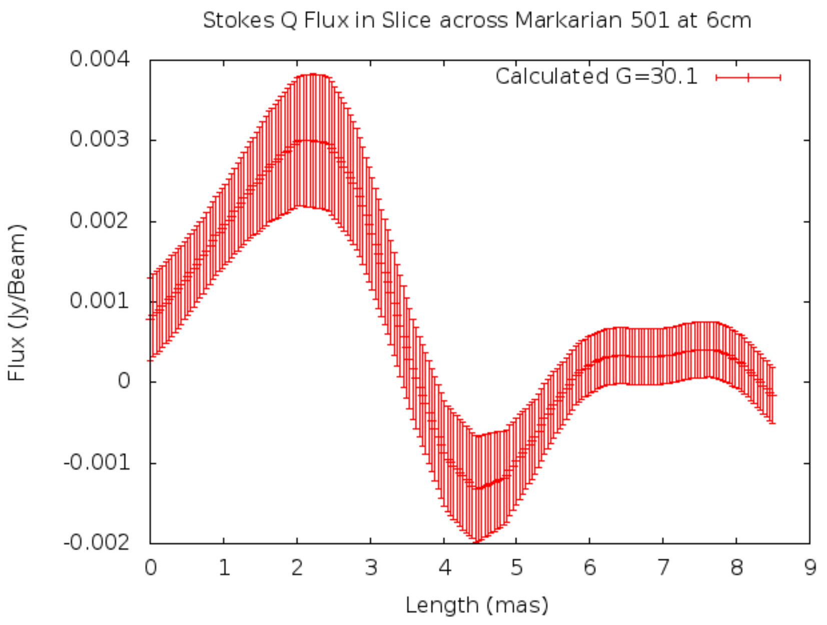
Note that in the case of completely uncorrelated variables the covariance matrix is diagonal and the Pearson correlation matrix is equal to the identity matrix. The value of G in this case is 1 and the number of degrees of freedom does not need to be corrected.

Figure 6.12 shows Q and U slices taken across the jet of Markarian 501 at 6 cm. The error bars shown are generated using the new technique introduced at the beginning of the chapter and resulted in a value of G equal to 30.1. It is notable that the 8.49 mas long slice is approximately 3.9 beams across (using the geometric mean of the beam, 2.1951 mas). Therefore with an extremely conservative perspective on resolution one can say that there are roughly 4 independent measurements made across the slice. Of course the true number of measurements is higher than this, but removing correlation effects is difficult. The value for G calculated would result in $DOF = \frac{200}{30.1} \approx 6.6$. Thus this technique suggests that there are 6.6 independent measurements across the slice, in agreement with out intuition that the effective size of a resolution element is somewhat but not dramatically smaller than the beamwidth.

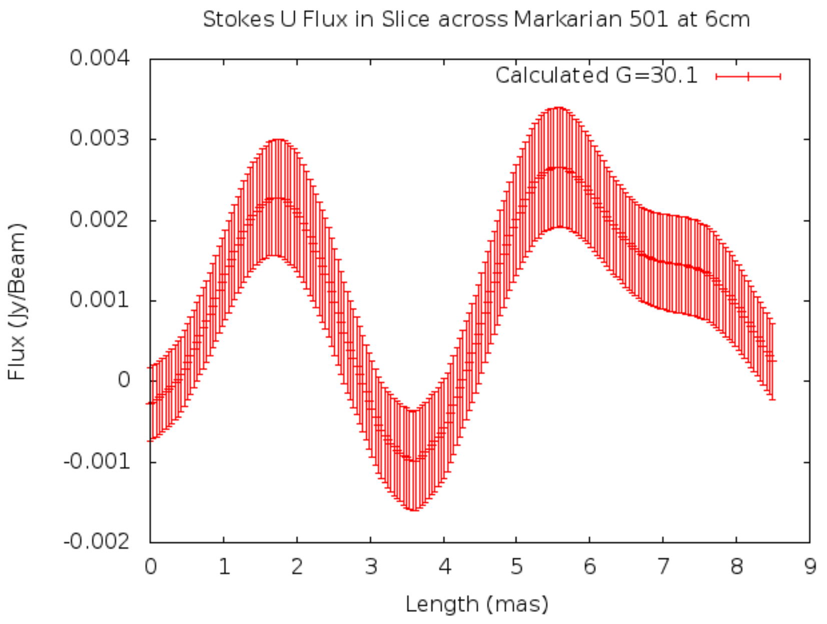
While this method attempts to account for correlation in the use of a χ^2 test and does in some cases work better than the normal χ^2 on correlated data, in order to perform a fully rigorous statistical fit to correlated data a different statistical test (not based on the χ^2 statistic) may be more useful.

6.5 Uncertainty in a rotation measure gradient

The following section describes the standard method of calculating a rotation measure gradient and calculates the effect of the correlation introduced by the EVPA calibration. It also briefly describes a more rigorous method of calculating a Faraday rotation measure taking into account correlation introduced by correcting for the local Galactic contribution to the total Faraday rotation. It is of note that the AIPS task 'RMCUB' as kindly provided by R. Zavala does not implement the standard weighted least squares technique described in the following section, thus all Faraday rotation measure gradients reported were made in



(a) Stokes Q.



(b) Stokes U.

Figure 6.12: Q and U slices from a CLEAN image Markarian 501 at 6cm. The error bars have been generated using the method described in Section 6.4.2. The errors and correlation correspond to $G = 30.1$. The convolving beam was 2.55 $\times$ 1.89 mas with a position angle of -25.56 degrees. The original slice covered approximately 28.3 pixels with a cell size of 0.3 mas. The slice was then interpolated to 200 pixels. The approximate position of the Slice can be seen in the MEM contour map in Figure 4.2

external software.

6.5.1 Calculating a Rotation Measure Gradient

Given values for Stokes Q and U and their uncertainties in a particular region, the polarisation angle in that region can be found. The polarisation angle is defined as

$$\chi = \frac{1}{2}ArcTan(\frac{U}{Q}). \tag{6.33}$$

Using equation [6.7], the uncertainty in χ can be calculated as

$$\sigma_\chi^2 = \frac{1}{4}[(\frac{Q}{Q^2+U^2})^2\sigma_U^2 + (\frac{U}{Q^2+U^2})^2\sigma_Q^2]. \tag{6.34}$$

Calibration corrections may need to be added to each polarisation angle. If these corrections have an error associated with them it can be added to the existing error in quadrature as derived from equation [6.7] to give

$$\sigma_{\chi_{final}}^2 = \sigma_\chi^2 + \sigma_{cal}^2. \tag{6.35}$$

It should be noted that as the same calibration correction is added to each polarisation angle at the same frequency, this introduces a systematic error into the polarisation angle. This systematic error is a type of correlation and must be treated carefully to ensure that its effect on subsequent calculations is taken into account. See Mahmud, Gabuzda and Bezrukovs (2009) and Section 6.5.2 for a detailed description of how this correlation affects the error in the rotation measure.

If one measures χ_{final} at the same location over a number of frequencies one can then investigate the change in χ_{final} across frequencies. If the change is due to Faraday rotation caused as the wave passes through a plasma in a magnetic field the following linear relationship is expected to be observed

$$\chi = \chi_0 + RM\lambda^2, \tag{6.36}$$

where λ is the wavelength in metres, and RM, the rotation measure, is a constant related to the plasma density and magnetic field along the line of sight between the plasma and the observer.

By plotting observed values of χ against λ^2 and fitting a line to the distribution the value of the rotation measure (the slope of the line) and its uncertainty can be calculated using standard least squares techniques and some extra measures needed due to the correlations in the polarisation angles discussed above.

The weighted least squares technique fits a line $f(x) = ax + b$ to a set of n data points (x, y) with uncertainties in y_i of σ_i by minimising

$$\chi^2 = \sum_{i=1}^{n} w_i (y_i - a x_i - b)^2 \tag{6.37}$$

relative to a and b. Note in this case that χ^2 is used in the statistical sense, and has nothing to do with the polarisation angle. w_i are the unnormalised weights $\frac{1}{\sigma_i^2}$. This minimisation yields the following equations for a and b and their uncertainties:

$$a = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}, \tag{6.38}$$

$$\sigma_a^2 = \frac{1}{W(\overline{x^2} - \overline{x}^2)}, \tag{6.39}$$

$$b = \overline{y} - a\overline{x}, \tag{6.40}$$

$$\sigma_b^2 = \frac{\overline{x^2}}{W(\overline{x^2} - \overline{x}^2)}, \tag{6.41}$$

where W is the sum of the unnormalised weights and $\overline{x}$ and $\overline{x^2}$ indicate the weighted averages of x and x^2 respectively. Fitting the linear relationship in Equation (6.36) using this method allows the RM and its error to be found using the above formulae, where the x axis is λ^2 in metres squared, and the y axis is χ_{final}; the polarisation angle after calibration corrections have been applied.

6.5.2 Removing the correlation due to EVPA calibration

To find the difference between rotation measures at two different points the following formula is used

$$\Delta RM = RM_1 - RM_2. \tag{6.42}$$

To find the uncertainty in the difference Equation (6.11) must be used as RM_1 and RM_2 are correlated due to the calibration corrections made to χ. This results in the formula

$$\sigma^2_{\Delta RM} = \sigma^2_{RM_1} + \sigma^2_{RM_2} - \frac{2}{W_1 W_2} \frac{1}{(\overline{x_1^2} - \overline{x_1}^2)} \frac{1}{(\overline{x_2^2} - \overline{x_2}^2)} \sum_{k=0}^{nfreq} (x_k - \overline{x_1})(x_k - \overline{x_2}) w_{1k} w_{2k} \sigma_C^2, \tag{6.43}$$

where σ_C is the error in the calibration and W_1, $\overline{x_1}$ etc. refer to the parameters from the least squares fit of each RM value.

6.5.3 Correlation due to Galactic Faraday rotation

The method for calculating a Faraday rotation measure as described above ignores any correlation between the polarisation angles introduced by EVPA calibration and correction for local Galactic Faraday rotation. A fully general method of linear fitting can be derived from Equation (6.30) as

$$\chi^2 = \sum_{i=1}^{n} (y_i - a x_i - b) V_{ij}^{-1} (y_j - a x_j - b), \tag{6.44}$$

where x_i and y_i correspond to the square wavelength and polarisation angle of frequency i, respectively, a is the slope of the fitted line, b is the y-intercept and V_{ij} is the covariance matrix. The resulting expressions for a and b are complicated, but can be found in standard statistics texts such as Barlow (1993). The covariance matrix for χ can be written

$$V_{ij} = cov(\chi_i, \chi_j) = \sum_{k=0}^{Nvar} \frac{\partial \chi_i}{\partial v_k} \frac{\partial \chi_j}{\partial v_k} \Delta v_k^2, \tag{6.45}$$

where the index k varies over all variables involved v_k. Any given final polarisation in degrees can be thought of as being dependent on three variables, χ_d, the uncalibrated polarisation angle, $\Delta\chi_E$, the correction due to EVPA calibration and ΔRM_G, the local Galactic Faraday rotation, such that

$$\chi_{final} = \chi_d + \Delta\chi_E - \frac{180}{\pi}\lambda^2 \Delta RM_G. \tag{6.46}$$

Thus the above equation can be written

$$V_{ij} = \sum_{k=0}^{Nfreq} \frac{\partial\chi_i}{\partial\chi_{dk}} \frac{\partial\chi_j}{\partial\chi_{dk}} \Delta\chi_{dk}^2 + \sum_{k=0}^{Nfreq} \frac{\partial\chi_i}{\partial\chi_{Ek}} \frac{\partial\chi_j}{\partial\chi_{Ek}} \Delta\chi_{Ek}^2 + \frac{\partial\chi_i}{\partial RM_G} \frac{\partial\chi_j}{\partial RM_G} \Delta RM_G^2 \tag{6.47}$$

$$= \delta_{ij}(\Delta\chi_{dk}^2 + \Delta\chi_{Ek}^2) + (\frac{180}{\pi})^2 \lambda_i^2 \lambda_j^2 \Delta RM_G^2, \tag{6.48}$$

where χ_d and $\Delta\chi_E$ are only correlated at the same frequency, but ΔRM_G introduces an off-diagonal term to the covariance matrix as the same calibration is applied to all frequencies. Given accurate errors for χ at a pixel or region (generated as described in this section), this method allows statistically rigorous Faraday rotation measure values to be generated, which can then be tested for a significant gradient using Equation (6.43).

6.6 Conclusions

The proposed method introduced in Section 6.2 for estimating the uncertainties in individual pixels in a CLEAN image is mathematically straightforward. It is based on the hypothesis that the uncertainty in each CLEAN component is proportional to the rms noise off-source, $\sigma = f\sigma_{rms}$, and represents a first attempt to describe the image uncertainties mathematically as being a result of the CLEAN algorithm used to produce the image.

Section 6.3 detailed Monte Carlo simulations which suggest that the error method proposed works well for large, well-resolved, sources – successfully predicting regions of high error and scaling properly between typical fluxes associated with Stokes I and Q or U. The agreement in the error patterns shown by the Monte Carlo simulations and error maps calculated using the proposed method is striking, indicating that the proposed error-calculation method successfully reproduces at least some component of the overall image uncertainties. However the model does not appear to work well for sources whose size is comparable to or smaller than the beam size, as is typical on the scales probed by VLBI observations of AGN. This breakdown may indicate that the appropriate value of f in Equation

(6.6) is so low that the corresponding error pattern is below the variations in the background noise (which are random and not modelled by the technique developed in Section 6.3); however, it is also possible that this approach is not suitable for compact sources for some other reason related to differences in the way that CLEAN deconvolves compact and more extended regions of emission.

A modification of the error model may result in better agreement with the results of the Monte Carlo simulations, but at the cost of increased mathematical complexity. Some exploration of different models based on Section 6.2 was conducted, however no model emerged that offered a clear advantage over the one proposed and investigated in this chapter. It may also be possible to instead perform a series of Monte Carlo simulated observations to generate an accurate error map and correlation matrix for a real source. This would be computationally demanding and time consuming, but would allow the full and accurate exploitation of the real data independent of any analytical model for the errors.

Section 6.4.1.1 considers the uncertainty in the average of values within pixels in a 3×3 or 9×9 region, and compares estimates of these uncertainties obtained using Monte Carlo simulations and using the proposed analytical error formula. One striking result to come out of this analysis is that the uncertainty in these average values tends to be close to the background RMS fluctuations σ_{rms} in off-source pixels. This means that σ_{rms} provides a reasonable estimate of the uncertainty in averaged flux values for compact sources with relatively low flux levels (e.g. for polarised fluxes).

Section 6.5 describes how an accurate value and error for a Faraday rotation measure gradient can be calculated, and includes a more rigorous treatment of correlations introduced due to EVPA calibration (Section 6.5.2) and local Faraday rotation (Section 6.5.3). These improvements to the calculation of Faraday rotation measures and gradients are independent of the error model from Section 6.2 and will result in formally more accurate Faraday rotation measure maps of any AGN.

In conclusion, the error method developed in this chapter shows considerable promise in its ability to describe single-pixel errors in CLEAN maps of sources that are not too compact compared to the beam size and is easily extendible to more complicated calculations where the analytical description of the correlations introduced by the CLEAN algorithm allows accurate uncertainties for many useful quantities to be determined.

Chapter 7

Conclusions

This thesis has developed new methods for high resolution radio astronomy imaging and investigating the uncertainties in VLBI intensity and polarisation images.

Chapter 3 of this thesis has described a new software implementation of the Maximum Entropy Method for use in the the deconvolution of multi-wavelength VLBI polarisation images of AGN jets. Using the Cornwell-Evans algorithm and building on previous work done on MEM-based VLBA polarisation deconvolution by Holdaway and Wardle (1990) and Sault (1990), a new C++ program, PMEM was written. This program is suitable for polarisation observations and capable of working closely with existing software suites. PMEM produces MEM images which have the resolution advantage enjoyed by the MEM over the CLEAN algorithm outlined in the introductory chapters of this thesis.

PMEM uses multi-threaded coding techniques and high performance external libraries to ensure a rapid deconvolution of the typical VLBI polarisation data. It has an easy-to-use Python based graphical user interface and uses the FITS file format to input and output data. This should make PMEM useful to astronomers using popular astronomy software packages such as AIPS and CASA, while requiring a minimum amount of knowledge of the underlying code. The source code of PMEM is fully commented and modular so advanced users may easily edit and adapt it for new purposes, such as the possible inclusion of PMEM in a CASA-based imaging "pipeline", where large numbers of images are created with minimal human input.

PMEM has been tested using Monte Carlo simulations of realistic VLBI sources. The UVFILL2 code included in Appendix B was developed to enable such simulations and was also used to generate Monte Carlo simulations for the error technique proposed in Chapter 6. The performance of the MEM was compared against that of the CLEAN algorithm for a variety of conditions. An early interesting result of this comparison was that, even though the CLEAN algorithm specifies the use of a restoring beam that is a Gaussian fit to the central lobe of the dirty beam, the CLEAN algorithm actually performs very well when used with a restoring beam up to 50% smaller than this. This is equivalent to double the normal CLEAN resolution. Thus a mild to mid-range CLEAN based "super-resolution" may be appropriate in many maps as long the resulting images do not show any signs of CLEAN based artefacts, such as any evidence of δ function type structures.

The performance of both MEM and CLEAN was found to be very close for restoring beams down to about half the size of the normal CLEAN beam. After this point performance drops off significantly for both algorithms, with each algorithm demonstrating particular strengths and weaknesses. As might be expected in an algorithm which models sources with a series of δ functions, the CLEAN algorithm outperforms the MEM method at the more "pointy" parts of the source, where it appears that MEM's continuous model of the emission struggles to rise to a point. Conversely, the MEM outperforms CLEAN at more diffuse regions. This is of interest in one of the major applications for PMEM – the detection of Faraday rotation measure gradients.

In order to examine the statistical significance of a Faraday rotation measure gradient across a jet the greatest range of the gradient is found and the difference between the two end points is calculated. The MEM is particularly strong at imaging these end points as they correspond to the diffuse edges of the jet. The simulations have also demonstrated that the MEM performs particularly well in measurements of fractional polarisation and polarisation angle, even for restoring beams up to a quarter of the normal CLEAN beam (this is presumably a result of the way in which polarised emission is included in the Gull and Skilling entropy).

The combination of MEM's high quality measurements of the polarisation angle at high resolutions and its suitability for imaging diffuse regions gives it a major advantage over the CLEAN algorithm right at the critical points for the measurements of gradients in Faraday rotation measure and suggest that the MEM is an excellent choice to image multi-wavelength polarisation data. The MEM also outperforms CLEAN at determining the total Stokes I, Q and U flux in the source, though this will depend on out accurately the zero spacing flux is estimated from the Stokes I visibility data for a source.

The use of PMEM on real VLBI observations of Markarian 501, 1633+632 and 0716+714 in Chapter 4 supports the conclusions of the Monte Carlo simulations. In all three cases high resolution MEM maps of the fractional polarisation show significantly more detail than the corresponding CLEAN maps. The higher resolution also allows the polarisation angle structure of each jet to be examined in greater detail. These results in particular are in close agreement with the high level of accuracy in MEM based fractional polarisation and polarisation angle maps predicted by the simulations in Chapter 3. The MEM polarisation maps offer a deeper insight into the polarisation structure than the corresponding CLEAN maps. In addition to this, the high resolution makes the total intensity (Stokes I) structure visible in much greater detail, allowing the association of particular polarisation features with jet components only clear at resolutions corresponding to beams smaller than the CLEAN beam.

The first MEM-based VLBI Faraday rotation measure maps have been made by combining MEM polarisation-angle images at multiple wavelengths. These maps have then been compared to their CLEAN algorithm equivalents. The high resolution achieved in these MEM maps allows the magnetic field structure of AGN jets to be probed on smaller scales than possible with the CLEAN algorithm, and will prove useful in the analysis of new and existing gradients in Faraday rotation measure. Intrinsic polarisation angle maps were also generated and show strong agreement with existing CLEAN maps – sometimes even at resolutions corresponding to a fraction of the CLEAN beam. This again suggests that a mild amount of super–resolution can be applied to the CLEAN algorithm without negatively affecting its accuracy.

Chapter 5 of this thesis presents observations of transverse gradients in Faraday rotation measure in CLEAN images of 0716+714 and 1749+714 first published in Mahmud et al. (2013). The detection of these gradients, all of which involve a change from negative to positive rotation measure, constitute strong evidence for the presence of helical magnetic fields in the two sources. The detection of reversals in the direction of the gradients further out along the jet is interesting. Some possible explanations are proposed, the most likely being a phenomenon along the lines of the Poynting-Robertson battery, whereby the magnetic field lines in an inner helix return to the AGN core in an outer helical field. Monte Carlo results showing the ability to detect transverse Faraday rotation gradients even when the intrinsic width of the jet is appreciably smaller than the width of the beam (at least down to jet widths of about $\frac{1}{20}$ of the beam width) are also presented.

Chapter 6 of this thesis has presented a new model for errors introduced by CLEAN in maps deconvolved using this algorithm. Previous practices for estimating the single-pixel uncertainties in CLEAN maps have largely ignored the issue of correlation between neighbouring pixels, and have taken the single-pixel uncertainties to be equal to the root-mean-square deviations far from regions of source emission. The error model introduced attempts to both predict the distribution of uncertainties across CLEAN maps, as well as to determine the exact correlation between neighbouring pixels.

Monte Carlo simulations showing the distribution of CLEAN errors for various sources are presented and compared to the patterns predicted by the error method proposed. While the error method works very well for sources spread across many CLEAN beams, the CLEAN algorithm performs better than predicted by my model for compact sources similar to realistic observations of AGN jets. Thus, although the ability of the error model to reproduce the error patterns visible in the corresponding Monte Carlo error maps for well resolved sources is striking, this method cannot in its present form be applied to sources with sizes appreciably smaller than the CLEAN beam. It may be possible to modify the technique to provide better predictions of the uncertainties for compact sources, but such a modification remains illusive.

Chapter 6 also explores one of the major advantages of the new error model – the ability to calculate rigorous errors for simple statistical quantities that have, up until now, been impossible to determine accurately. Calculations for weighted averages, linear fits and the fitting of a general model to an interpolated slice of data are made. The predicted unweighted average generated by the error model is compared to the result from Monte Carlo simulations, again showing good agreement for larger sources, but poor agreement for compact sources. These simulations also demonstrate that the off-source rms flux fluctuations σ_{rms} provide a reasonable estimate of the uncertainty in averaged flux values for compact sources with relatively low flux levels (e.g. for polarised fluxes). A simple method for estimating the number of degrees of freedom in a χ^2 statistic for correlated data is also presented. This method, while crude, does gives much better results than simply ignoring the correlation between pixels when making a fit and provides a first step on a path to rigorous test statistics on the applicability of a fit to a transverse slice across a jet.

In conclusion, this thesis has developed a range of new methods for high resolution radio astronomy imaging, along with codes that may be useful for MEM deconvolution, Monte Carlo analyses and error analyses involving radio-interferometry data. Much of the code is included in Appendix B, and any additional code (for example, to implement the error method described in Chapter 6 is available on request.

Bibliography

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A. and Sorensen, D. (1999), *LAPACK Users' Guide*, third edn, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Antonucci, R. (1993), 'Unified Models for Active Galactic Nucle and Quasars', *Annual Review of Astronomy and Astrophysics* **31**(1), 473–521.

Barlow, R. J. (1993), *Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences (Manchester Physics Series)*, Wiley.

Bisnovatyi-Kogan, G. S. (2007), 'Dynamic confinement of jets by magneto-torsional oscillations', *Monthly Notices of the Royal Astronomical Society* **376**(1), 457–464.

Blandford, A. and Payne, A. (1982), 'Hydromagnetic flows from accretion discs and the production of radio jets', *Monthly Notices of the Royal Astronomical Society* **199**, 883–903.

Blandford, A. and Znajek, A. (1977), 'Electromagnetic extraction of energy from Kerr black holes', *Monthly Notices of the Royal Astronomical Society* **179**, 433–456.

Bracewell, R. N. (2000), *The Fourier Transform and Its Applications*, McGraw-Hill Higher Education.

Burke, B. F. and Graham-Smith, F. (2010), *An Introduction to Radio Astronomy*, Cambridge University Press.

Carroll, B. W. and Ostlie, D. A. (2007), *An introduction to modern astrophysics*, Pearson Addison-Wesley.

Cawthorne, T. V., Jorstad, S. G. and Marscher, A. P. (2013), 'Polarization structure in the core of 1803+784: A signature of recollimation shocks?', *The Astrophysical Journal* **772**(1), 14.

Clark, A. (1980), 'An efficient implementation of the algorithm 'CLEAN'', *Astronomy and Astrophysics* **89**.

Contopoulos, I., Christodoulou, D. M., Kazanas, D. and Gabuzda, D. C. (2009), 'The Invarient Twist of Magnetic Fields in the Relativistic Jets of Active Galactic Nuclei', *The Astrophysical Journal* **702**(2), L148–L152.

Contopoulos, I. and Kazanas, D. (1998), 'A Cosmic Battery', *The Astrophysical Journal* **508**(2), 859–863.

Conway, R. G. and Kronberg, P. P. (1969), 'Interferometric measurement of polarization distribution in radio sources', *Monthly Notices of the Royal Astronomical Society* **142**.

Cornwell, T. J. and Evans, K. F. (1985), 'A simple maximum entropy deconvolution algorithm', *Astronomy and Astrophysics (ISSN 0004-6361)* **143**, 77–83.

Croke, S. M. and Gabuzda, D. C. (2008), 'Aligning VLBI images of active galactic nuclei at different frequencies', *Monthly Notices of the Royal Astronomical Society* **386**(2), 619–626.

Croke, S. M., O'Sullivan, S. P. and Gabuzda, D. C. (2010), 'The parsec-scale distributions of intensity, linear polarization and Faraday rotation in the core and jet of Mrk 501 at 8.4-1.6 GHz', *Monthly Notices of the Royal Astronomical Society* **402**(1), 259–270.

Frieden, A. and Wells, A. (1978), 'Restoring with maximum entropy. II - Poisson sources and backgrounds', *Optical Society of America* **68**, 93–103.

Frigo, M. and Johnson, S. (2005), 'The Design and Implementation of FFTW3', *Proceedings of the IEEE* **93**(2), 216–231.

Gabuzda, D. C., Christodoulou, D. M., Contopoulos, I. and Kazanas, D. (2012), 'Evidence for Helical Magnetic fields in Kiloparsec-Scale AGN Jets and the Action of a Cosmic Battery', *Journal of Physics: Conference Series* **355**(1), 012019.

Gabuzda, D. C., Murray, E. and Cronin, P. (2004), 'Helical magnetic fields associated with the relativistic jets of four BL Lac objects', *Monthly Notices of the Royal Astronomical Society* **351**(4), L89–L93.

Gabuzda, A. (2008), ‘Radiation Processes in the Universe: Synchrotron Radiation and Propagation Effects’, *"Proceedings of the 2nd MCCT-SKADS Training School. Radio Astronomy: fundamentals and the new instruments. 26th August - 4th September 2008. Siguenza (Spain). Published online at http://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=65* .

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M. and Rossi, F. (2009), ‘GNU Scientific Library Reference Manual-(v1. 12)’, *Network Theory Ltd* .

Greisen, A. (2002), *Information Handling in Astronomy - Historical Vistas*, Vol. 285 of *Astrophysics and Space Science Library*, Kluwer Academic Publishers, Dordrecht.

Gull, S. F. and Skilling, J. (1984), ‘The Maximum Entropy Method’, *Indirect Imaging. Measurement and Processing for Indirect Imaging* **-1**, 267.

Hallahan, R. and Gabuzda, D. (2008), ‘The Deca-parsec Scale Radio Structures of BL Lac Objects’, *"Proceedings of the 9th European VLBI Network Symposium on The role of VLBI in the Golden Age for Radio Astronomy and EVN Users Meeting. September 23-26* .

Healy, F. (2013), Multi-Epoch 18-22cm VLBA Observations of Several BL-LAC Objects, Masters thesis, University College Cork.

Heckman, A. (1980), ‘An optical and radio survey of the nuclei of bright galaxies - Activity in normal galactic nuclei’, *Astronomy and Astrophysics* **87**, 152–164.

Högbom, J. A. (1974), ‘Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines’, *Astronomy and Astrophysics Supplement* **15**.

Holdaway, M. A. (1990), ‘Maximum entropy imaging of radio astrophysical data’, *Ph.D. Thesis Brandeis Univ.* .

Holdaway, M. A. and Wardle, J. F. C. (1990), Maximum Entropy Method Imaging of Polarization in Very Long Baseline Interferometry, *in* A. F. Gmitro, P. S. Idell and I. J. LaHaie, eds, ‘IN: Digital image synthesis and inverse optics; Proceedings of the Meeting’, pp. 714–724.

Hovatta, T., Lister, M. L., Aller, M. F., Aller, H. D., Homan, D. C., Kovalev, Y. Y., Pushkarev, A. B. and Savolainen, T. (2012), ‘Mojave: Monitoring of Jets in Active Galactic Nuclei With Vlba Experiments. Viii. Faraday Rotation in Parsec-Scale Agn Jets’, *The Astronomical Journal* **144**(4), 105.

Hubble, A. (1925), 'Cepheids in spiral nebulae', *The Observatory* **48**, 139–142.

Jaynes, E. (1957), 'Information Theory and Statistical Mechanics', *Physical Review* **106**(4), 620–630.

Jennison, A. (1958), 'A phase sensitive interferometer technique for the measurement of the Fourier transforms of spatial brightness distributions of small angular extent', *Monthly Notices of the Royal Astronomical Society* **118**.

Kardashev, N. S., Khartov, V. V., Abramov, V. V., Avdeev, V. Y., Alakoz, A. V., Aleksandrov, Y. A., Ananthakrishnan, S., Andreyanov, V. V., Andrianov, A. S., Antonov, N. M., Artyukhov, M. I., Arkhipov, M. Y., Baan, W., Babakin, N. G., Babyshkin, V. E., Bartel', N., Belousov, K. G., Belyaev, A. A., Berulis, J. J., Burke, B. F., Biryukov, A. V., Bubnov, A. E., Burgin, M. S., Busca, G., Bykadorov, A. A., Bychkova, V. S., Vasil'kov, V. I., Wellington, K. J., Vinogradov, I. S., Wietfeldt, R., Voitsik, P. A., Gvamichava, A. S., Girin, I. A., Gurvits, L. I., Dagkesamanskii, R. D., D'Addario, L., Giovannini, G., Jauncey, D. L., Dewdney, P. E., D'yakov, A. A., Zharov, V. E., Zhuravlev, V. I., Zaslavskii, G. S., Zakhvatkin, M. V., Zinov'ev, A. N., Ilinen, Y., Ipatov, A. V., Kanevskii, B. Z., Knorin, I. A., Casse, J. L., Kellermann, K. I., Kovalev, Y. A., Kovalev, Y. Y., Kovalenko, A. V., Kogan, B. L., Komaev, R. V., Konovalenko, A. A., Kopelyanskii, G. D., Korneev, Y. A., Kostenko, V. I., Kotik, A. N., Kreisman, B. B., Kukushkin, A. Y., Kulishenko, V. F., Cooper, D. N., Kut'kin, A. M., Cannon, W. H., Larionov, M. G., Lisakov, M. M., Litvinenko, L. N., Likhachev, S. F., Likhacheva, L. N., Lobanov, A. P., Logvinenko, S. V., Langston, G., McCracken, K., Medvedev, S. Y., Melekhin, M. V., Menderov, A. V., Murphy, D. W., Mizyakina, T. A., Mozgovoi, Y. V., Nikolaev, N. Y., Novikov, B. S., Novikov, I. D., Oreshko, V. V., Pavlenko, Y. K., Pashchenko, I. N., Ponomarev, Y. N., Popov, M. V., Pravin-Kumar, A., Preston, R. A., Pyshnov, V. N., Rakhimov, I. A., Rozhkov, V. M., Romney, J. D., Rocha, P., Rudakov, V. A., Räisänen, A., Sazankov, S. V., Sakharov, B. A., Semenov, S. K., Serebrennikov, V. A., Schilizzi, R. T., Skulachev, D. P., Slysh, V. I., Smirnov, A. I., Smith, J. G., Soglasnov, V. A., Sokolovskii, K. V., Sondaar, L. H., Stepan'yants, V. A., Turygin, M. S., Turygin, S. Y., Tuchin, A. G., Urpo, S., Fedorchuk, S. D., Finkel'shtein, A. M., Fomalont, E. B., Fejes, I., Fomina, A. N., Khapin, Y. B., Tsarevskii, G. S., Zensus, J. A., Chuprikov, A. A., Shatskaya, M. V., Shapirovskaya, N. Y., Sheikhet, A. I., Shirshakov, A. E., Schmidt, A., Shnyreva, L. A., Shpilevskii, V. V., Ekers, R. D. and Yakimov, V. E. (2013), '"RadioAstron"-A telescope with a size of 300 000 km: Main parameters and first observational results', *Astronomy Reports* **57**(3), 153–194.

Konigl, A. (1981), 'Relativistic jets as X-ray and gamma-ray sources', *The Astrophysical Journal* **243**, 700.

Laing, R. A. (1981), 'Magnetic fields in extragalactic radio sources', *The Astrophysical Journal* **248**, 87.

Lister, M. L., Aller, H. D., Aller, M. F., Cohen, M. H., Homan, D. C., Kadler, M., Kellermann, K. I., Kovalev, Y. Y., Ros, E., Savolainen, T., Zensus, J. A. and Vermeulen, R. C. (2009), 'MOJAVE: Monitoring Of Jets in Active Galactic Nuclei with VLBA Experiments. V. Multi-Epoch VLBA Images', *The Astronomical Journal* **137**(3), 3718–3729.

Mahmud, M., Coughlan, C. P., Murphy, E., Gabuzda, D. C. and Hallahan, D. R. (2013), 'Connecting magnetic towers with Faraday rotation gradients in active galactic nuclei jets', *Monthly Notices of the Royal Astronomical Society* **431**(1), 695–709.

Mahmud, M., Gabuzda, D. C. and Bezrukovs, V. (2009), 'Surprising evolution of the parsec-scale Faraday Rotation gradients in the jet of the BL Lac object B1803+784', *Monthly Notices of the Royal Astronomical Society* **400**(1), 2–12.

McMullin, A., Waters, A., Schiebel, A., Young, A. and Golap, A. (2007), 'CASA Architecture and Applications', *Astronomical Data Analysis Software and Systems XVI ASP Conference Series* **376**.

Miyoshi, M., Moran, J., Herrnstein, J., Greenhill, L., Nakai, N., Diamond, P. and Inoue, M. (1995), 'Evidence for a black hole from high rotation velocities in a sub-parsec region of NGC4258', *Nature* **373**(6510), 127–129.

Murphy, E. (2013), Computational Studies of the Transverse Structure of AGN Jets, PhD thesis, University College Cork.

Murphy, E., Cawthorne, T. V. and Gabuzda, D. C. (2013), 'Analysing the transverse structure of the relativistic jets of active galactic nuclei', *Monthly Notices of the Royal Astronomical Society* **430**(3), 1504–1515.

Murphy, E. and Gabuzda, D. C. (2012), 'Investigating the Effects of Finite Resolution on Observed Transverse Jet Profiles', *Journal of Physics: Conference Series* **355**(1), 012009.

Murphy, A., Coughlan, A. and Gabuzda, A. (2013), 'Investigating the effects of finite resolution on observed transverse Rotation Measure distributions', *eprint arXiv:1309.1718* .

Papageorgiou, A. (2005), Transverse polarization structure of parsec-scale radio jets, PhD thesis, University of Central Lancashire.

Peterson, B. M. (1997), *An Introduction to Active Galactic Nuclei*, Cambridge University Press.

Pushkarev, A. B., Gabuzda, D. C., Vetukhnovskaya, Y. N. and Yakimov, V. E. (2005), 'Spine-sheath polarization structures in four active galactic nuclei jets', *Monthly Notices of the Royal Astronomical Society* **356**(3), 859–871.

Reichstein, A. (2012), Active Galactic Nuclei with Spine-Sheath Polarisation Structure, PhD thesis, University College Cork.

Reichstein, A. and Gabuzda, D. (2011), 'The Use of Faraday Rotation Sign Maps as a Diagnostic for Helical Jet Magnetic Fields', *Proceedings of Beamed and Unbeamed Gamma-Rays from Galaxies* p. 5.

Roberts, D. H., Wardle, J. F. C. and Brown, L. F. (1994), 'Linear polarization radio imaging at milliarcsecond resolution', *The Astrophysical Journal* **427**, 718.

Rybicki, G. B. and Lightman, A. P. (2008), *Radiative Processes in Astrophysics*, John Wiley & Sons.

Sault, R. J. (1990), 'A modification of the Cornwell and Evans maximum entropy algorithm', *The Astrophysical Journal* **354**, L61.

Sault, R. J., Bock, D. C.-J. and Duncan, A. R. (1999), 'Polarimetric imaging of large fields in radio astronomy', *Astronomy and Astrophysics Supplement Series* **139**(2), 387–392.

Sault, A., Teuben, A. and Wright, A. (1995), 'A Retrospective View of MIRIAD', *Astronomical Data Analysis Software and Systems IV* **77**.

Skilling, A. and Bryan, A. (1984), 'Maximum Entropy Image Reconstruction - General Algorithm', *Monthly Notices of the Royal Astronomical Society* **211**.

Slipher, A. (1914), 'The Radial Velocity of the Andromeda Nebula', *Popular Astronomy* **22**.

Sokolovsky, K. V., Kovalev, Y. Y., Pushkarev, A. B. and Lobanov, A. P. (2011), 'A VLBA survey of the core shift effect in AGN jets', *Astronomy & Astrophysics* **532**, A38.

Starck, J.-L. and Murtagh, F. (2006), *Astronomical Image and Data Analysis (Astronomy and Astrophysics Library)*, Springer.

Taylor, G. B., Carilli, C. L. and Perley, R. A. (1999), 'Synthesis Imaging in Radio Astronomy II', *Synthesis Imaging in Radio Astronomy II* **180**.

Taylor, G. B. and Zavala, R. (2010), 'Are there Rotation Measure Gradients across Active Galactic Nuclei Jets?', *The Astrophysical Journal* **722**(2), L183–L187.

Urry, C. M. and Padovani, P. (1995), 'Unified Schemes for Radio-Loud Active Galactic Nuclei', *Publications of the Astronomical Society of the Pacific* **107**, 803.

Wakker, B. P. and Schwarz, U. J. (1988), 'The Multi-Resolution CLEAN and its application to the short-spacing problem in interferometry', *Astronomy and Astrophysics (ISSN 0004-6361)* **200**, 312–322.

Wells, A., Greisen, A. and Harten, A. (1981), 'FITS - a Flexible Image Transport System', *Astronomy and Astrophysics Supplement* **44**.

Wernecke, A. and D'Addario, A. (1977), 'Maximum Entropy Image Reconstruction', *IEEE Trans. Comput.* pp. 351–364.

William H. Press, Saul A. Teukolsky, Willian T. Vetterling, B. P. F. (2007), *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press.

Appendix A

Basic Relativistic Effects in AGN

A.1 Relativistic Beaming

Consider a body moving at a relativistic speed u in the x direction. The Lorentz transformations of special relativity give the following relationship between the observer's frame of reference (unprimed) and the co-moving frame (primed)

$$dx = \frac{dx' + udt'}{\sqrt{1 - \frac{u^2}{c^2}}} \tag{A.1}$$

and similarly for the y and z directions and time

$$dy = dy' \tag{A.2}$$

$$dz = dz' \tag{A.3}$$

$$dt = \frac{dt' + dx'\frac{u}{c^2}}{\sqrt{1 - \frac{u^2}{c^2}}} \tag{A.4}$$

Defining $v_x = \frac{dx}{dt}$ and $v'_x = \frac{dx'}{dt'}$ the following equations can be obtained for the velocities

$$v_x = \frac{v'_x + u}{1 + v'_x \frac{u}{c^2}} \tag{A.5}$$

$$v_y = \frac{v'_y}{(1 + v'_x \frac{u}{c^2})\gamma} \tag{A.6}$$

$$v_z = \frac{v'_z}{(1 + v'_x \frac{u}{c^2})\gamma} \tag{A.7}$$

where $\beta = \frac{u}{c}$ and the Doppler factor, γ, is $\gamma = (1-\beta^2)^{-\frac{1}{2}}$. In a more compact form this can be written

$$v_\parallel = \frac{v'_\parallel + u}{1 + v'_\parallel \frac{u}{c^2}} \tag{A.8}$$

$$v_\perp = \frac{v'_\perp}{1 + v'_\parallel \frac{u}{c^2}} \tag{A.9}$$

where $v_\parallel$ and $v_\perp$ are the velocity components parallel and perpendicular to u, respectively. This gives the expression for the angle along which the object is observed to move as

$$\tan\phi = \frac{v_\perp}{v_\parallel} = \frac{v_\perp}{\gamma(v'_\parallel + u)} \tag{A.10}$$

$$\tan\phi = \frac{v' \sin\phi'}{\gamma(v' \cos\phi i + u)} \tag{A.11}$$

Consider a photon with velocity c emitted at an angle of 90° to u in it's rest (primed) frame. The following equations can be are obtained

$$\tan\phi = \frac{1}{\gamma\frac{u}{c}} = \frac{1}{\gamma\beta} \tag{A.12}$$

$$\cos\phi = \frac{1}{\sqrt{\tan^2\phi + 1}} = \beta \tag{A.13}$$

$$\sin\phi = \sqrt{1 - \cos^2\phi} = \frac{1}{\gamma} \tag{A.14}$$

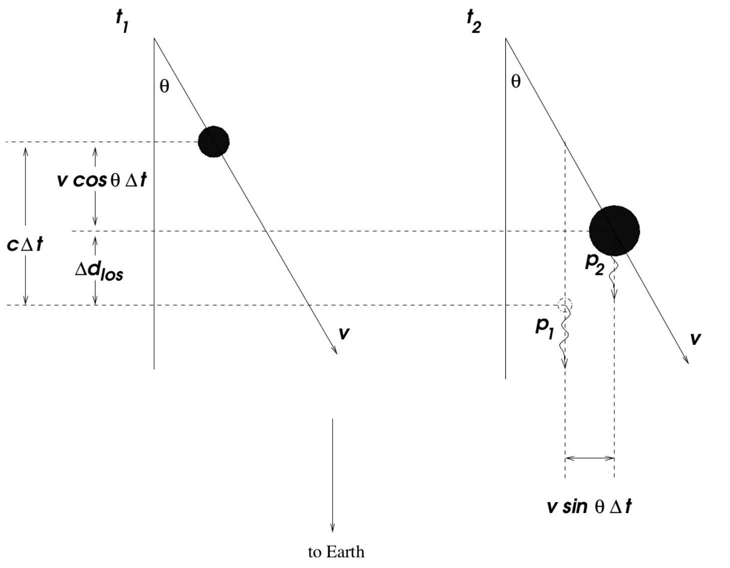


Figure A.1: A source moving with speed v at an angle θ with respect to the Earth emitting photons at times t_1 and t_2. Image from Gabuzda (2008).

For a jet with a relativistic speed $\beta \approx 1$, therefore γ is very large. This implies

$$\sin\phi \approx \phi = \frac{1}{\gamma} \tag{A.15}$$

The consequence of this is that a moving source emitting radiation isotropically will appear brighter than it is in the rest frame when approaching an observer, but dimmer than in the rest frame when it is receding from the observer. In the example above half of the source's total emission will be detected in an angle of just $\phi = \frac{1}{\gamma}$ in the observer frame. This is sometimes called the "Headlight Effect", Doppler boosting or relativistic beaming.

A.2 Apparent superluminal motion

Consider a source of radiation moving with speed v at an angle θ to the line of sight towards the observer as shown in Figure A.1. The distance moved on the sky can be written as

$$\Delta d_{sky} = v\sin\theta\Delta t \tag{A.16}$$

The distance between photons emitted at time t_1 and t_2 is

$$\Delta x = c\Delta t - v\cos\theta\Delta t = \Delta t(c - v\cos\theta) \tag{A.17}$$

where $\Delta t = t_2 - t_1$. The time measured between the arrival of the two photons is therefore

$$\Delta t_{arr} = \frac{\Delta x}{c} = \Delta t(1 - \frac{v}{c}\cos\theta) \tag{A.18}$$

This gives the apparent speed of motion on the sky as

$$v_{app} = \frac{\Delta d_{sky}}{\Delta t_{arr}} = \frac{v\sin\theta}{1 - \frac{v}{c}\cos\theta} \tag{A.19}$$

$$\beta_{app} = \frac{\beta\sin\theta}{1 - \beta\cos\theta} \tag{A.20}$$

where $\beta = \frac{v}{c}$. Equation (A.20) reveals that the speed of the object as deduced from it's motion across the sky differs from the true speed of the object. The maximum β_{app} for a given β can be calculated by differentiating with respect to θ and setting the resulting function to zero. The resulting angle which gives a maximum value for β_{app} gives

$$\cos\theta = \beta \tag{A.21}$$

$$\sin\theta = \sqrt{1 - \cos^2\theta} = \frac{1}{\gamma} \tag{A.22}$$

The corresponding maximum value of β_{app} is

$$\beta_{app} = \beta\gamma \tag{A.23}$$

Therefore for highly relativistic speeds ($B \approx 1$, $\gamma > 1$), the observed speed can be far greater than the true speed, leading to the apparent superluminal motion of the source. This effect is regularly observed in blobs of gas in the relativistic jets of AGN.

Appendix B

Source Code

This appendix contains the C++ source code for the main PMEM Maximum Entropy Method deconvolution code, the UVFILL simulation observation code and various subroutines needed to interface with the FITS file format. The header files are also included. All of the code is released under the GNU GPL version 3. You can obtain the source code by emailing me at colm.coughlan_at_umail.ucc.ie. For the purposes of brevity, not all of the code described in this thesis is included in this appendix. The code corresponding to the Monte Carlo simulations of the error method in Chapter 6 and the implementation of various fits and error analysis is available on request.

B.1 PMEM Code

The following section contains the source code to the main PMEM C++ program, along with a script for a Python based graphical user interface and a Linux make file to compile the program.

B.1.1 MEMPY Interface

```
'''
    This program is called mempy. It is a GUI to pmem.
    Copyright (C) 2013 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

'''

import easygui as eg
from subprocess import call

msg        = "Please enter the parameters"
title      = "MEMPY"
fieldNames = ["Number of polarisations (max 4)","Stokes I dirty map","Stokes Q dirty map","Stokes U dirty map","Stokes V
      dirty map (untested)","Dirty Beam", "Default Map (0 if none)","Estimated Stokes I flux","Estimated Pol 2 flux","
      Estimated Pol 3 flux","Estimated Pol 4 flux","Conserve Flux? (1/0, 2 = est)","Estimated Stokes I rms","Estimated Pol 2
      rms","Estimated Pol 3 rms","Estimated Pol 4 rms","Maximum number of iterations","Restoring beam BMAJ (as)", "
      Restoring beam BMIN (as)", "Restoring beam BPA (degrees)","Noise box : BLCX","Noise box : BLCY","Noise box :
      TRCX","Noise box : TRCY","Acceleration factor (rec:3)","Q factor (rec:0.5)","Pol factor (rec:2.0)","Output Name","
      Number of edge pixels to be ignored","Debug mode?"]
fieldValues = []   # we start with blanks for the values

loadmsg = "Do you want to load the last values?"
loadtitle = "Welcome to MEMPY"
if eg.ynbox(loadmsg, loadtitle):      # show a Continue/Cancel dialog

        try:
                f=open('mempy_driver.dat','r')
        except IOError:
                print 'Error loading old data\n'


        for i in range(len(fieldNames)):
                try:
                        line = f.readline().rstrip('\n')
                except IOError as e:
                        print 'Error loading old data\n'
                        break
                fieldValues.append(line)

        f.close()


fieldValues = eg.multenterbox(msg,title, fieldNames, fieldValues)
errmsg = ""
npol=0
temp=0

# make sure that none of the fields was left blank
while 1:  # do forever, until we find acceptable values and break out
        if fieldValues == None:
                errmsg = "Blank field"
                break
        errmsg = ""

    # look for errors in the returned values

        # check that the number of polarisations is okay

        try:
                npol = int(fieldValues[0])
        except ValueError as e:
                errmsg += 'Invalid choice of number of polarisations\n'



        # check that the dirty imap is okay



        try:
                with open(fieldValues[1]) as f: pass
        except IOError as e:
                errmsg += 'Error locating dirty Stokes I map\n'
```

```
# check that the dirty Q,U,V maps are okay



for i in range(npol-1):
        try:
                with open(fieldValues[2+i]) as f: pass
        except IOError as e:
                errmsg += 'Error locating dirty polarisation map '+str(1+i)+'\n'


# check that the dirty beam is okay



try:
        with open(fieldValues[5]) as f: pass
except IOError as e:
        errmsg += 'Error locating dirty beam\n'

# check that the default map is okay



try:
        with open(fieldValues[6]) as f: pass
except IOError as e:
        if float(fieldValues[6]) <> 0:
                errmsg += 'Error locating default map\n'


# check that the estimated fluxes are okay

try:
        temp = float(fieldValues[7])
except ValueError as e:
        errmsg += 'Need a (positive) value for estimated I flux\n'
if temp <= 0:
        errmsg += 'Need a (positive) value for estimated I flux\n'


for i in range(npol-1):
        try:
                temp = float(fieldValues[8+i])
        except ValueError as e:
                errmsg += 'Need a value for estimated polarised flux\n'

# check that the conserve flux option is okay

try:
        temp = int(fieldValues[11])
except ValueError as e:
        errmsg += 'Need a decision about weather or not to conserve flux\n'
if (temp <> 0 and temp <> 1) and temp <> 2:
        errmsg += 'Please enter 0 , 1 or 2 for flux conservation\n'

# check that rms options are okay

for i in range(npol):
        try:
                temp = float(fieldValues[12+i])
        except ValueError as e:
                errmsg += 'Need a value for estimated rms\n'

# check that niter is okay

try:
        temp = int(fieldValues[16])
except ValueError as e:
        errmsg += 'Need a maximum number of iterations\n'
if temp <= 0:
        errmsg += 'Please enter a strictly positve maximum number of iterations\n'


# check that bmaj is okay

try:
        temp = float(fieldValues[17])
except ValueError as e:
        errmsg += 'Need a BMAJ\n'
if temp < 0:
        errmsg += 'Please enter a positive or zero-valued BMAJ\n'


# check that bmin is okay

try:
        temp = float(fieldValues[18])
except ValueError as e:
        errmsg += 'Need a BMIN\n'
if temp < 0:
        errmsg += 'Please enter a positive or zero-valued BMIN\n'


# check that bmaj is okay

try:
        temp = float(fieldValues[19])
except ValueError as e:
        errmsg += 'Need a BPA\n'

# check that the BLC TRC is okay

try:
        temp = int(fieldValues[20])
except ValueError as e:
        errmsg += 'Need a valid BLCX\n'

try:
        temp2 = int(fieldValues[21])
except ValueError as e:
        errmsg += 'Need a valid BLCY\n'

try:
        temp3 = int(fieldValues[22])
except ValueError as e:
        errmsg += 'Need a valid TRCX\n'

try:
        temp4 = int(fieldValues[23])
except ValueError as e:
        errmsg += 'Need a valid TRCY\n'

if temp3 -temp <= 0:
        errmsg += 'Please enter a sensible noise box coord for x\n'

if temp4 -temp2 <= 0:
        errmsg += 'Please enter a sensible noise box coord for x\n'

# check that the acceleration factor is ok

try:
        temp = float(fieldValues[24])
except ValueError as e:
        errmsg += 'Need an acceleration factor\n'
if temp <= 0:
        errmsg += 'Please enter a positive acceleration factor\n'

# check that the q factor is ok

try:
        temp = float(fieldValues[25])
except ValueError as e:
        errmsg += 'Need a q factor\n'
if temp <= 0:
        errmsg += 'Please enter a positive q factor\n'

# check that the pol upweight factor is ok

try:
        temp = float(fieldValues[26])
except ValueError as e:
        errmsg += 'Need a polarisation weighting factor\n'
if temp <= 0:
```

```
                errmsg += 'Please enter a positive q factor\n'

        # check that the output name option is okay (check that the use has entered one)

        temp = len(fieldValues[27])
        if temp <= 1:
                errmsg += 'Please set an output name\n'

        # check that the ignore edge pixels option is okay

        try:
                temp = int(fieldValues[28])
        except ValueError as e:
                errmsg += 'Need a valid number of edge pixels to ignore. Try 0 as a default\n'
        if temp < 0:
                errmsg += 'Error in number of pixels to ignore (needs to be an integer >=0).\n'


        # check that the debug option is okay

        try:
                temp = int(fieldValues[29])
        except ValueError as e:
                errmsg += 'Need a decision about weather or not to use debug mode\n'
        if temp <> 0 and temp <> 1:
                errmsg += 'Please enter 0 or 1 for debug mode\n'



        if errmsg == "":
                break # no problems found
        else:
                # show the box again, with the errmsg as the message
                fieldValues = eg.multenterbox(errmsg, title, fieldNames, fieldValues)

if errmsg <> "":
        exit()





f=open('mempy_driver.dat','w') # write instructions to a driver for the C++ file


for i in range(len(fieldNames)):
        f.write(fieldValues[i]+'\n')

f.close()

err=call("cp mempy_driver.dat "+fieldValues[27]+".pmemo",shell=True) # copy driver file to a log version

err=call("./pmem > "+fieldValues[27]+".pmemlog",shell=True) # call C++ program, printing the output to a log

if err!=0:
        if err==2:
                print "Run complete: Desired criteria not achieved."
        else:
                print "Error "+str(err)+" running c++ code"
        exit()

print('Program appears to have executed correctly')
```

B.1.2 PMEM Headers

```
/*
    pmem_heads.hpp is a header file used by pmem and some programs it links with
    Copyright (C) 2012 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.
*/


#include <iostream>
#include <fstream>
#include <string>
#include <string.h>
#include <cmath>
#include <stdlib.h>
#include <sstream>
#include <fftw3.h>
#include <fitsio.h>
#include <stdio.h>
#include <omp.h>
#include <armadillo>



int cfits_write(const char* filename , double* array , int imsize , double cell , double ra , double dec , double*
      centre_shift , double* rotations , double freq , double freq_delta , int stokes , char* object , char* observer , char*
      telescope , double equinox , char* date_obs , char* history , double bmaj , double bmin , double bpa , int niter ,
      bool jy_per_beam);

int cfits_read_header_map(const char* filename , int* dim , double* cell , double* ra , double* dec , double*
      centre_shift , double* rotations , double* freq , double* freq_delta , int* stokes , char* object , char* observer ,
      char* telescope , double* equinox , char* date_obs , double* bmaj , double* bmin , double* bpa , int* ncc);

int cfits_read_map(const char* filename, double* tarr, int dim2 , double* cc_xarray, double* cc_yarray, double*
      cc_varray , int ncc);

extern "C" {
        void dgetrf_( const int * , const int * , double * , const int * , int * , int * );
        void dgetrs_( const char * , const int * , const int * , double * , const int * , int * , double * , const int* ,
              int * );
};
```

B.1.3 PMEM Source Code

```
/*
    This program is called pmem. It deconvolves polarised VLBI maps using MEM.
    Copyright (C) 2012 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.


        Polarised Maxmimum Entropy Method
        Colm Coughlan - colmcoughlanirl %%%at%%% gmail.com
*/
```

```
#include "pmem_heads.hpp"

        using namespace std;



// structure to make passing around all the gradients a bit easier

struct gradient_structure {
        double EE;
        double EF;
        double EG;
        double EH;
        double EJ;
        double FF;
        double FG;
        double FH;
        double FJ;
        double GG;
        double GH;
        double GJ;
        double HH;
        double JJ;
        double II;
};


int write_csv(string filename, double* array, int imsize);
void arrange_ft(double* arr, int imsize);
int ft(fftw_complex* in, fftw_complex* out, int imsize, int direction);
int convolve(double* data, fftw_complex* response, int imsize, int pad_factor, double* output , fftw_plan&
      forward_transform, fftw_plan& backward_transform, double* double_buff, fftw_complex* complex_buff);
string int2str (int num);

int get_residual_map(double* dirty_map , double* convolved_model , double* residual_map, int imsize, int
      ignore_pixels);
int get_info(double** model , double** residual, double* default_map2 , gradient_structure& grad , double* chi2_rms,
      double& total_flux , double alpha , double beta , double gamma , double& imin, double& imax , int imsize , int
      npol, double q, int ignore_edge_pixels);
int new_ABG(gradient_structure grad , double delta_E , double delta_F , double delta_G , double& alpha , double&
      beta , double& gamma , bool conserve_flux , int npol);
int cal_step(double** model , double** residual, double* default_map2 , double alpha , double beta , double gamma ,
      int imsize , int npol , double q , double& J0 , double** step_map, int ignore_edge_pixels);
int take_step(double** model , double** step , double step_length , double step_limit , int imsize , int npol, int
      ignore_edge_pixels);
int check_step(double** old_model , double** new_model , double** new_residual, double* default_map2 , double
      alpha , double beta , double gamma , int imsize , int npol , double q , double& J1, int ignore_pixels);
int interpolate_models(double** current_model , double** new_model , double frac_new , int imsize , int npol, int
      ignore_pixels);
int interpolate_residuals(double** current_residuals , double** new_residuals , double frac_new , int imsize2 , int npol);
int copy_model(double**& model1, double**& model2);

int gen_gauss(double* matrix, int imsize, double cellsize, double bmaj, double bmin, double bpa);
int ft_beam(double* beam, fftw_complex* ft_beam, int imsize, int pad_factor, fftw_plan& plan, double* double_buff,
      fftw_complex* complex_buff);
double average_region(double* map, int blcx, int blcy, int trcx, int trcy, int imsize);
int clip_edges(double* map, double replacement_value, int edge_limit, int imsize);
double rms_region(double* map, int blcx, int blcy, int trcx, int trcy, int imsize);
int zero_array(double* array, int imsize);

const double min_flux = 1e-18; // set the minimum flux allowable for the Stokes I models (must be > 0 for model)

int main()
{
        double ra;
        double dec;
        int imsize;
        double cell;      // stored in degrees

        double centre_shift[2]; // where the peak of the source is on the map (x and y coords)
        double rotations[2];     // any rotation applied to the map
        int stokes[4];           // enough room to hold stokes parameters for 4 maps

        double freq;     // stored in Hz
        double freq_delta;
```

```
char object[FLEN_VALUE];
char observer[FLEN_VALUE]; // information about the source
char telescope[FLEN_VALUE];
double equinox;
char date_obs[FLEN_VALUE];

char history[] = "MEM deconvolution performed by PMEM. See PMEM logs for details.";

double bmaj;   // stored in degrees
double bmin;   // stored in degrees
double bpa;    // stored in degrees
int ncc;       // number of clean components

string line;
string filename_dirty_imap;    // strings for filenames, IO
string filename_dirty_beam;
string *filename_pol;
string output_name;
string filename_default_map;
gradient_structure grad;       // room to transport gradient information
fstream fout;

bool converged;
bool converged_temp;
bool conserve_flux;
bool estimate_flux;
bool debug;

double current_rms[4]; // current Stokes I, Q, U , V residual rms
double min_rms[] = {1e99,1e99,1e99,1e99};    // minimum Stokes I residual rms achieved (should be updated almost
    every iteration)
int min_rms_i_ctr;    // counter value at which the minimum Stokes I residual rms has been achieved
int convergence_limit; // the number of allowed iterations where the Stokes I residuals are not decreasing before
    the program quits

int i,j , k,ctr;
int err;
int imsize2;
int npol;              // no. of polarisations being deconvolved
int niter;             // max no. of iterations
int pad_factor = 1;    // 1 for no padding, 2 for zero padding
int blcx , blcy , trcx , trcy;
int ignore_edge_pixels; // number of pixels at the edge of the map to ignore (useful for aliasing problems)

double* null_double;

double acceleration_factor;
double q_factor;
double pol_upweight_factor;
double temp,temp2;
double pixels_per_beam;
double total_flux;
double alpha , beta , gamma;
double imin , imax;
double step_length1 , step_length2;
double old_step_length1 , old_step_length2;
double step_limit;
double delta_E, delta_F, delta_G;
double J0 , J1;
double rms_tolerance , flux_tolerance , convergence_tolerance;
double bmaj_restoring , bmin_restoring , bpa_restoring;



double** dirty_map;
double** current_model;
double** new_model;
double** current_residuals;
double** new_residuals;
double** convolved_model;
double** best_model;

double* dirty_beam;
fftw_complex* dirty_beam_ft;
fftw_complex* complex_buff;
double* double_buff;
double* default_map2;
```

```
double* zero_spacing_flux;
double* chi2_rms;
double* rms_theoretical;

double q = 0.0;



// start up some of the fourier tranform settings and variables from FFTW


fftw_init_threads();    // initialise fftw parallisation
fftw_plan_with_nthreads(omp_get_max_threads());


fftw_plan forward_transform, backward_transform;






fout.open("mempy_driver.dat",ios::in);  // read in filename, number of frequencies to be used and number of 3x3 grids
        required
if (fout.is_open())
{
        getline(fout, line);
        npol = atoi(line.c_str());                // read number of pols

        getline(fout,filename_dirty_imap);        // read imap filename
}
else
{
        cout<<"Exiting program..."<<endl;
        return(1);
}

rms_tolerance=1.1;    // allow a tolerance of 10 % in guess for noise
flux_tolerance=0.1;
convergence_tolerance=0.1;



err = cfits_read_header_map( filename_dirty_imap.c_str() , &imsize , &cell , &ra , &dec , centre_shift , rotations ,
       &freq , &freq_delta , &stokes[0] , object , observer , telescope , &equinox , date_obs , &bmaj , &bmin , &
       bpa , &ncc );
if( err != 0 )
{
        cout<<"Error reading header from "<<filename_dirty_imap<<endl;
        cout<<"Exiting program..."<<endl;
        return(1);
}

if(ncc!=0)
{
        cout<<"Warning: "<<filename_dirty_imap<<" does not appear to be a dirty map..."<<endl;
        ncc = 0;
}

imsize2=imsize*imsize;



j = pad_factor * imsize * ( pad_factor * imsize / 2 + 1 );      // allowing for padding in buffers and reducing
       memory needed for r2c and c2r transform by taking Herm. conjugacy into account

// first declaration of memories − only for things with npol dependencies


dirty_map=new double*[npol];
current_model=new double*[npol];
new_model=new double*[npol];
current_residuals=new double*[npol];
new_residuals=new double*[npol];
convolved_model=new double*[npol];
best_model = new double*[npol];
zero_spacing_flux=new double[npol];
```

```
chi2_rms=new double[npol];
rms_theoretical=new double[npol];

if(npol>1)
{
        filename_pol = new string[npol];
}


dirty_beam=new double[imsize2];
dirty_beam_ft = ( fftw_complex* ) fftw_malloc( sizeof( fftw_complex ) * j );  // saving memory with herm.
    conjugacy
complex_buff = ( fftw_complex* ) fftw_malloc( sizeof( fftw_complex ) * j );    // allowing room for padding in
    buffers
double_buff = ( double* ) fftw_malloc( sizeof( double ) * pad_factor * pad_factor * imsize2 );
default_map2=new double[imsize2];


for(i=0;i<npol;i++)
{
        dirty_map[i]=new double[imsize2];
        current_model[i]=new double[imsize2];
        new_model[i]=new double[imsize2];
        current_residuals[i]=new double[imsize2];
        new_residuals[i]=new double[imsize2];
        convolved_model[i]=new double[imsize2];
        best_model[i] = new double[imsize2];

        chi2_rms[i] = 0.0;
        rms_theoretical[i] = 0.0;
        zero_spacing_flux[i] = 0.0;
}

if(err!=0)
{
        goto free_mem_exit;
}

for(i=0;i<npol;i++)
{
        zero_spacing_flux[i] = 0.0;
        rms_theoretical[i] = 0.0;
}

if (fout.is_open())
{
        for(i=0;i<npol−1;i++)
        {
                getline(fout,filename_pol[i]);  // read in the filenames of the Q,U,V files
        }

        for(i=npol;i<4;i++)
        {
                getline(fout, line);        // read in any blank lines
        }


        getline(fout,filename_dirty_beam);      // filename of dirty beam

        getline(fout,filename_default_map);     // filename of default map (if any)

        for(i=0;i<npol;i++)
        {
                getline(fout, line);
                zero_spacing_flux[i] = atof(line.c_str());        // read in estimated fluxes
        }

        for(i=npol;i<4;i++)
        {
                getline(fout, line);        // read in any blank lines
        }

        estimate_flux = false;
        getline(fout, line);
        i = atoi(line.c_str());
        if(i>0)
        {
```

```
                        conserve_flux = true;
                        if(i==2)
                        {
                                estimate_flux = true;
                        }
                }
                else
                {
                        conserve_flux=false;
                }

                for(i=0;i<npol;i++)
                {
                        getline(fout, line);
                        rms_theoretical[i] = atof(line.c_str());        // read in estimated fluxes
                }

                for(i=npol;i<4;i++)
                {
                        getline(fout, line);    // read in any blank lines
                }

                getline(fout, line);
                niter = atoi(line.c_str());     // read in number of iterations

                getline(fout, line);
                bmaj_restoring = atof(line.c_str());    // read in restoring beam

                getline(fout, line);
                bmin_restoring = atof(line.c_str());

                getline(fout, line);
                bpa_restoring = atof(line.c_str());

                getline(fout, line);
                blcx = atoi(line.c_str());      // read in noise box info
                blcx--;

                getline(fout, line);
                blcy = atoi(line.c_str());      // read in blc and trc. Convert to 0 base.
                blcy--;

                getline(fout, line);
                trcx = atoi(line.c_str());
                trcx--;

                getline(fout, line);
                trcy = atoi(line.c_str());
                trcy--;

                getline(fout, line);    // read in acceleration factor name
                acceleration_factor = atof(line.c_str());

                getline(fout, line);    // read in q factor name
                q_factor = atof(line.c_str());

                getline(fout, line);    // read in polarisation upweight factor name
                pol_upweight_factor = atof(line.c_str());

                getline(fout,output_name);      // read in output name

                getline(fout, line);
                ignore_edge_pixels = atoi(line.c_str());

                getline(fout, line);
                if(atoi(line.c_str())==1)       // read in debug mode option
                {
                        debug=true;
                }
                else
                {
                        debug=false;
                }


                fout.close();
        }
```

```
else
{
        return(1);
}


// prepare  FT plans



forward_transform = fftw_plan_dft_r2c_2d(imsize * pad_factor , imsize * pad_factor , double_buff , complex_buff
      , FFTW_MEASURE ); // optimise FFT

backward_transform = fftw_plan_dft_c2r_2d(imsize * pad_factor , imsize * pad_factor , complex_buff ,
      double_buff , FFTW_MEASURE ); // r2c is always a forward transform etc



if(  bmaj_restoring == 0 || bmin_restoring ==0) // if no restoring beam is given,  use  the  beam from the  dirty  maps
{
        bmaj_restoring = bmaj;
        bmin_restoring = bmin;
        bpa_restoring = bpa;    // storing  in  degrees
}
else
{
        bmaj_restoring /= 3600.0;
        bmin_restoring /= 3600.0;
}


// print  out  some information

cout<<"Running PMEM."<<endl<<endl;
cout<<"Target files:"<<endl;
cout<<"\t Stokes I : "<<filename_dirty_imap<<" with an estimated flux of "<<zero_spacing_flux[0]<<" Jy."<<
      endl;
for(  i  = 1; i  < npol ; i++ )
{
        cout<<"\t Stokes "<<i+1<<" : "<<filename_pol[i-1]<<" with an estimated flux of "<<zero_spacing_flux[i
              ]<<" Jy."<<endl;
}
cout<<"\t Dirty beam : "<<filename_dirty_beam<<"."<<endl;
if(  filename_default_map.length() > 1)
{
        cout<<"\t Default map : "<<filename_default_map<<"."<<endl;
}
else
{
        cout<<"\t No default map used."<<endl;
}
cout<<endl<<"Restoring beam information:"<<endl;
cout<<"\t BMAJ = "<<bmaj_restoring*3600.0<<" as, BMIN = "<<bmin_restoring*3600.0<<" as, BPA = "<<
      bpa_restoring<<" deg."<<endl<<endl;
cout<<"Map details :"<<endl;
cout<<"\t Imsize = "<<imsize<<" pixels."<<endl;
cout<<"\t Cellsize = "<<cell*(3600.0*1000.0)<<" mas."<<endl<<endl;
cout<<"Running parameters :"<<endl;
cout<<"\t Acceleration factor = "<<acceleration_factor<<endl;
cout<<"\t Q factor = "<<q_factor<<endl;
cout<<"\t Polarisation upweight factor = "<<pol_upweight_factor<<endl;
cout<<endl<<endl;




ncc=0;  // turn  off  any clean  component handling
err  = cfits_read_map(filename_dirty_imap.c_str() , dirty_map[0] , imsize2 , null_double , null_double ,
      null_double , ncc );
if(err!=0)
{
        cout<<endl<<"Error detected reading map from "<<filename_dirty_imap<<", err = "<<err<<endl<<endl
              ;
        cout<<"Program closing"<<endl;
        goto free_mem_exit;
}
```

```
for(i=1;i<npol;i++)    // read in all the polarisation maps (if any)
{
        err = cfits_read_header_map( filename_pol[i−1].c_str() , &imsize , &cell , &ra , &dec , centre_shift ,
              rotations , &freq , &freq_delta , &stokes[i] , object , observer , telescope , &equinox , date_obs ,
              &temp , &temp , &temp , &ncc );  // note the only new piece of information here is the stokes value
              (read in temp to avoid overwriting bmaj etc.)
        if(err!=0)
        {
                cout<<endl<<"Error detected reading header from "<<filename_pol[i−1]<<", err = "<<err<<endl
                      <<endl;
                cout<<"Program closing"<<endl;
                goto free_mem_exit;
        }

        ncc = 0;

        err = cfits_read_map( filename_pol[i−1].c_str() , dirty_map[i] , imsize2 , null_double , null_double ,
              null_double , ncc );
        if(err!=0)
        {
                cout<<endl<<"Error detected reading map from "<<filename_pol[i−1]<<", err = "<<err<<endl<<
                      endl;
                cout<<"Program closing"<<endl;
                goto free_mem_exit;
        }
}


err = cfits_read_map( filename_dirty_beam.c_str() , dirty_beam , imsize2 , null_double , null_double ,
      null_double , ncc );
if(err!=0)
{
        cout<<endl<<"Error detected reading beam from "<<filename_dirty_beam<<", err = "<<err<<endl<<
              endl;
        cout<<"Program closing"<<endl;
        goto free_mem_exit;
}




// estimate number of pixels per beam

pixels_per_beam = bmaj ∗ bmin ∗ M_PI / ( 4.0 ∗ log(2) ∗ cell ∗ cell );
if(pixels_per_beam <= 0 )
{
        cout<<"Error in estimating beamize. Strictly positive number of pixels per beam required."<<endl;
        goto free_mem_exit;
}
cout<<"Pixels per beam = "<<pixels_per_beam<<endl;



//  initialize  default map

if( filename_default_map.length() > 1)
{
        cout<<"Using given map as default."<<endl;

        ncc = 0;
        cfits_read_map( filename_default_map.c_str() , default_map2 , imsize2 , null_double , null_double ,
              null_double , ncc );       // load default map if given

        temp = 0.0;
        for(i=0;i<imsize2;i++)
        {
                temp += default_map2[i];
        }
        temp = zero_spacing_flux[0] / temp;    // normalise flux of default map to guess
        temp ∗= temp;

        for(i=0;i<imsize2;i++)
        {
                default_map2[i] ∗= (default_map2[i]∗temp);
        }
}
```

```
else
{
        cout<<"Making flat default map."<<endl;

        temp=zero_spacing_flux[0]/(imsize2);   // temp becomes the pixel value of the (flat) default map
        temp = temp * temp;   // the default map is never actually used, but its square is

        for(i=0;i<imsize2;i++)
        {
                default_map2[i]=temp;
        }
}

// initialize current models and zero residual maps

if( imsize - 2 * ignore_edge_pixels <= 0)
{
        cout<<"Ingore edge pixels = "<<ignore_edge_pixels<<" is too large for image of size "<<imsize<<endl;
        goto free_mem_exit;
}

ctr = (imsize - ignore_edge_pixels);
k = ctr * ctr;


for(int tpol=0; tpol<npol;tpol++)
{
        err = zero_array(new_residuals[tpol], imsize);
        err = zero_array(current_residuals[tpol], imsize);
        err = zero_array(current_model[tpol], imsize);

        temp = zero_spacing_flux[tpol] / k;
        cout<<"Initial model of polarisation "<<tpol<<" is set to a flat map with pixel value = "<<temp<<endl;

        #pragma omp parallel for collapse(2)
        for( i = ignore_edge_pixels ; i < ctr ; i++ )
        {
                for( j = ignore_edge_pixels ; j < ctr ; j++ )
                {
                        current_model[tpol][i*imsize + j] = temp;
                }
        }
}



cout<<"Set to ignore "<<ignore_edge_pixels<<" edge pixels."<<endl;
cout<<"Acceleration factor set to "<<acceleration_factor<<endl;

// find ft of the dirty beam

if( pad_factor == 1)
{
        arrange_ft( dirty_beam , imsize );      // rearrange beam to prepare for use in convolution
}

ft_beam(dirty_beam , dirty_beam_ft , imsize , pad_factor , forward_transform , double_buff , complex_buff); //
      get ft




// convolve initial model maps with dirty beam and get residual maps

for(i=0;i<npol;i++)
{
        convolve( current_model[i] , dirty_beam_ft , imsize , pad_factor , convolved_model[i] , forward_transform ,
                backward_transform , double_buff , complex_buff);
        get_residual_map( dirty_map[i] , convolved_model[i] , current_residuals[i] , imsize, ignore_edge_pixels );
}

// initialize alpha, beta and gamma

alpha = 0.0;
beta = 0.0;
gamma = 0.0;
```

```
// Caculate Q, a factor used in the approximation of the Hessian matrix as diagonal. Important value. (see Cornwell
       & Evans 1984, Sault 1990)

q = 0.0;
#pragma omp parallel for reduction( +: q)
for( i = 0 ; i < imsize2 ; i++ )
{
        q += dirty_beam[i]*dirty_beam[i];
}
q = q_factor * sqrt( q );
cout<<"Q = "<<q<<endl;


// get some information

err=get_info( current_model , current_residuals , default_map2 , grad , chi2_rms, total_flux , alpha , beta ,
       gamma , imin, imax , imsize , npol , q , ignore_edge_pixels);
if(err!=0)
{
        cout<<endl<<"Error detected in initial get_info, err = "<<err<<endl<<endl;
        cout<<"Program closing"<<endl;
        goto free_mem_exit;
}


if(debug)
{
        cout<<endl<<endl<<"Initial values"<<endl;
        cout<<"Alpha, beta, gamma = "<<alpha<<" , "<<beta<<" , "<<gamma<<endl;
        cout<<"Total flux, max and min = "<<total_flux<<" , "<<imax<<" , "<<imin<<endl;
        cout<<"First step, second step, step limit = "<<step_length1<<" , "<<step_length2<<" , "<<step_limit
            <<endl; // output some info
        cout<<"GradJ.J, Grad1.1, J0, J1 = "<<grad.JJ<<" , "<<grad.II<<" , "<<J0<<" , "<<J1<<endl;
        cout<<"Delta E, F, G = "<<delta_E<<" , "<<delta_F<<" , "<<delta_G<<endl;
        cout<<"GradE.E, GradF.F, GradG.G = "<<grad.EE<<" , "<<grad.FF<<" , "<<grad.GG<<endl;
        cout<<"GradE.F, GradF.G, GradE.G = "<<grad.EF<<" , "<<grad.FG<<" , "<<grad.EG<<endl; // output
            even more info
        cout<<"GradE.H, GradF.H, GradG.H = "<<grad.EH<<" , "<<grad.FH<<" , "<<grad.GH<<endl;
        cout<<"GradE.J, GradF.J, GradG.J = "<<grad.EJ<<" , "<<grad.FJ<<" , "<<grad.GJ<<endl;
}




// start iterations

converged=false;
ctr=0;
old_step_length1=0.0;
old_step_length2=0.0;
min_rms_i_ctr = 0;
convergence_limit=100;

while(!converged && ctr < niter)
{
        ctr++;
        k = (imsize - ignore_edge_pixels) * (imsize - ignore_edge_pixels);    // number of pixels being used

        delta_E = (chi2_rms[0] - (rms_theoretical[0] * rms_theoretical[0] * k) ) / q;   // find differences in E, F
               and G

        delta_F = 0.0;
        for(i=1;i<npol;i++)
        {
                delta_F += ( ( chi2_rms[i] - (rms_theoretical[i] * rms_theoretical[i] * k) ) / q);
        }
        delta_F *= pol_upweight_factor; // upweight polarisation error

        delta_G = total_flux - zero_spacing_flux[0];

        err = new_ABG( grad , delta_E , delta_F , delta_G , alpha , beta , gamma , conserve_flux , npol );    //
               update values of alpha, beta, gamma
        if(err!=0)
        {
                cout<<endl<<"Error detected in new_ABG, err = "<<err<<endl<<endl;
                break;
```

```
}

err = cal_step( current_model , current_residuals , default_map2 , alpha , beta , gamma , imsize , npol , q ,
        J0 , new_model, ignore_edge_pixels);    // find a good step
if(err!=0)
{
        cout<<endl<<"Error detected in cal_step, err = "<<err<<endl<<endl;
        break;
}


step_limit = 1.0;
if( grad.JJ > 0 )
{
        step_limit = min( 2.0 , acceleration_factor * 0.15 * grad.II / grad.JJ );       // this line is very
                very very important... ( especially the 0.15 factor)
}
step_length1 = min( 0.5 * (1.0 + old_step_length1) , step_limit);       // step length etc. is measured as a
        fraction (between 0 and 1)
old_step_length1 = step_length1;
J0 *= step_length1;

err = take_step( current_model , new_model , step_length1 , step_limit , imsize , npol, ignore_edge_pixels)
        ;  // take the step calculated, but scaled by step_length1
if(err!=0)
{
        cout<<endl<<"Error detected in take_step, err = "<<err<<endl<<endl;
        break;
}


// convolve new model maps with dirty beam and get new residual maps

for(i=0;i<npol;i++)
{
        convolve( new_model[i] , dirty_beam_ft , imsize , pad_factor , convolved_model[i] ,
                forward_transform , backward_transform , double_buff , complex_buff);
        get_residual_map( dirty_map[i] , convolved_model[i] , new_residuals[i] , imsize, ignore_edge_pixels
                );
}

err = check_step( current_model , new_model , new_residuals , default_map2 , alpha , beta , gamma ,
        imsize , npol , q , J1 , ignore_edge_pixels);       // check if the step was near optimal
if(err!=0)
{
        cout<<endl<<"Error detected in check_step, err = "<<err<<endl<<endl;
        break;
}

if( J0 − J1 != 0.0 )
{
        step_length2 = J0 / (J0 − J1);
}
else
{
        step_length2 = 1.0;
}
step_length2 = 0.5 * (step_length2 + old_step_length2);
step_length2 = min( step_length2 , step_limit / step_length1 );
old_step_length2 = step_length2;


if( fabs( step_length2 − 1.0) > 0.05 ) // if step 1 was okay, just use it, but if step 2 offers a decent
        advantage take the average of step1 and step2
{
        err = interpolate_models( current_model , new_model , step_length2 , imsize , npol,
                ignore_edge_pixels);       // interpolate between old and new models, scaling with step2
        if(err!=0)
        {
                cout<<endl<<"Error detected in interpolation of new and old models, err = "<<err<<endl
                        <<endl;
                break;
        }

        err = interpolate_residuals( current_residuals , new_residuals , step_length2 , imsize2 , npol);
                        // interpolate residuals too
        if(err!=0)
```

```
		{
			cout<<endl<<"Error detected in interpolation of new and old residuals, err = "<<err<<endl
				<<endl;
			break;
		}

		if(debug)
		{
			cout<<"Interpolating current and new models."<<endl;
		}
	}
	else
	{
		err = copy_model( current_model , new_model ); // just copy new model into old model
		if(err!=0)
		{
			cout<<endl<<"Error detected replacing new and old models, err = "<<err<<endl<<endl;
			break;
		}

		err = copy_model( current_residuals , new_residuals ); // just copy new residuals into old model
		if(err!=0)
		{
			cout<<endl<<"Error detected replacing new and old residuals, err = "<<err<<endl<<endl;
			break;
		}

		if(debug)
		{
			cout<<"Replacing current model with new model."<<endl;
		}
	}


	err = get_info( current_model , current_residuals , default_map2 , grad , chi2_rms, total_flux , alpha ,
		beta , gamma , imin, imax , imsize , npol , q , ignore_edge_pixels); // get grads etc.
	if(err!=0)
	{
		cout<<endl<<"Error detected in get_info, err = "<<err<<endl<<endl;
		goto free_mem_exit;
	}

	if(debug || (ctr%100 == 0) )
	{
		cout<<endl<<endl<<"Iteration number "<<ctr<<endl;
		cout<<"Alpha, beta, gamma = "<<alpha<<" , "<<beta<<" , "<<gamma<<endl;
		cout<<"Total flux, max and min = "<<total_flux<<" , "<<imax<<" , "<<imin<<endl;
	}

	if(debug)
	{
		cout<<"First step, second step, step limit = "<<step_length1<<" , "<<step_length2<<" , "<<
			step_limit<<endl; // output some info
		cout<<"GradJ.J, Grad1.1, J0, J1 = "<<grad.JJ<<" , "<<grad.II<<" , "<<J0<<" , "<<J1<<endl;
		cout<<"Delta E, F, G = "<<delta_E<<" , "<<delta_F<<" , "<<delta_G<<endl;
		cout<<"GradE.E, GradF.F, GradG.G = "<<grad.EE<<" , "<<grad.FF<<" , "<<grad.GG<<endl;
		cout<<"GradE.F, GradF.G, GradE.G = "<<grad.EF<<" , "<<grad.FG<<" , "<<grad.EG<<endl;
			// output even more info
		cout<<"GradE.H, GradF.H, GradG.H = "<<grad.EH<<" , "<<grad.FH<<" , "<<grad.GH<<endl;
		cout<<"GradE.J, GradF.J, GradG.J = "<<grad.EJ<<" , "<<grad.FJ<<" , "<<grad.GJ<<endl;
	}

	converged_temp = true; // check for convergence



	// update the rms that has been reached

	for(i=0;i<npol;i++)
	{
		current_rms[i] = rms_region( current_residuals[i] , blcx , blcy , trcx , trcy , imsize);		//
			this might actually go up at the start, as the residual map is not a noise map initially
	}


	// Check if the current iteration is the best (best average polarisation, if available)
```

```
if(npol==1)
{
        if( current_rms[0] < min_rms[0] )
        {
                min_rms[0] = current_rms[0];
                min_rms_i_ctr = ctr;

                #pragma opm parallel for
                for(k=0; k<imsize2; k++)
                {
                        best_model[0][k] = current_model[0][k];
                }
        }
}
else
{
        temp = 0.0;
        temp2 = 0.0;
        for(k = 0; k < npol ; k++)
        {
                temp += current_rms[k];
                temp2 += min_rms[k];
        }


        if( temp < temp2 )
        {
                for(k=0; k<npol; k++)
                {
                        min_rms[k] = current_rms[k];
                }
                min_rms_i_ctr = ctr;

                #pragma opm parallel for collapse(2)
                for(j=0; j<npol; j++)
                {
                        for(k=0; k<imsize2; k++)
                        {
                                best_model[j][k] = current_model[j][k];
                        }
                }
        }
}


// Test for convergence

for(i=0;i<npol;i++)
{
        converged = ( current_rms[i] < rms_theoretical[i] );

        if(debug or (ctr%100 == 0) )
        {
                if(converged)
                {
                        cout<<"Stokes "<<i+1<<" has converged. "<<current_rms[i]<<" < "<<
                               rms_theoretical[i]<<endl;
                }
                else
                {
                        cout<<"Convergence test S"<<i+1<<": "<<current_rms[i]<<" needs to be < "<<
                               rms_theoretical[i] <<endl;
                }
        }

        converged = converged && converged_temp;
        converged_temp = converged;
}


// Add flux condition if required

if(conserve_flux && (!estimate_flux) )
{
        converged = converged && ( total_flux − zero_spacing_flux[0] < flux_tolerance ∗ zero_spacing_flux
               [0]);
```

```
                if(debug&&converged)
                {
                        cout<<"Flux has converged."<<endl;
                }
        }
        converged = converged && (grad.JJ / grad.II < convergence_tolerance);

        if(alpha!=alpha)
        {
                err=1;
                cout<<endl<<"Error : Infinity detected in alpha."<<endl<<endl; // a check to make sure no
                      infinities are around
                goto free_mem_exit;
        }
        if(total_flux < 0)
        {
                err=1;
                cout<<endl<<"Error : Negative total Stokes I flux detected in model."<<endl<<endl;
                goto free_mem_exit;
        }

        if( ctr - min_rms_i_ctr > convergence_limit )
        {
                cout<<endl<<"Convergence appears to have stopped."<<endl;
                cout<<"If you have not achieved the desired convergence try changing some parameters."<<endl;
                break;
        }
}


if(converged)
{
        cout<<endl<<"Successful convergence after "<<ctr<<" iterations."<<endl;
        for(i=0;i<npol;i++)
        {
                cout<<"Stokes "<<i+1<<" has converged. "<<current_rms[i]<<" < "<<rms_theoretical[i]<<endl;
        }
}
else
{
        cout<<endl<<"Failed to converge after "<<ctr<<" iterations."<<endl;
        for(i=0;i<npol;i++)
        {
                cout<<"Convergence test S"<<i+1<<": "<<current_rms[i]<<" needs to be < "<< rms_theoretical[i
                      ] <<endl;
        }
}

cout<<endl<<endl<<"Final iteration number "<<ctr<<endl;
cout<<"Alpha, beta, gamma = "<<alpha<<" , "<<beta<<" , "<<gamma<<endl;
cout<<"Total flux, max and min = "<<total_flux<<" , "<<imax<<" , "<<imin<<endl;

if(debug)
{
        cout<<"First step, second step, step limit = "<<step_length1<<" , "<<step_length2<<" , "<<step_limit
              <<endl; // output some info
        cout<<"GradJ.J, Grad1.1, J0, J1 = "<<grad.JJ<<" , "<<grad.II<<" , "<<J0<<" , "<<J1<<endl;
        cout<<"Delta E, F, G = "<<delta_E<<" , "<<delta_F<<" , "<<delta_G<<endl;
        cout<<"GradE.E, GradF.F, GradG.G = "<<grad.EE<<" , "<<grad.FF<<" , "<<grad.GG<<endl;
        cout<<"GradE.F, GradF.G, GradE.G = "<<grad.EF<<" , "<<grad.FG<<" , "<<grad.EG<<endl; // output
              even more info
        cout<<"GradE.H, GradF.H, GradG.H = "<<grad.EH<<" , "<<grad.FH<<" , "<<grad.GH<<endl;
        cout<<"GradE.J, GradF.J, GradG.J = "<<grad.EJ<<" , "<<grad.FJ<<" , "<<grad.GJ<<endl;
}

if( (min_rms[0] < current_rms[0]) && (!converged) )
{
        cout<<endl<<endl<<"A better Stokes I convergence of "<<min_rms[0]<<" was attained during iteration "
              <<min_rms_i_ctr<<endl;
        cout<<"Corresponding S2 convergence is "<<min_rms[1]<<endl;
        cout<<"Corresponding S3 convergence is "<<min_rms[2]<<endl;
        cout<<"Loading results from there."<<endl;


        for(i=0;i<npol;i++)     // calculate convolved final models from best_model
        {
```

```
                        convolve( best_model[i] , dirty_beam_ft , imsize , pad_factor  , convolved_model[i] ,
                               forward_transform , backward_transform , double_buff , complex_buff);
                        get_residual_map( dirty_map[i] , convolved_model[i] , current_residuals[i] ,   imsize,
                               ignore_edge_pixels );
                }
        }

        gen_gauss(new_model[0], imsize , cell, bmaj_restoring , bmin_restoring , bpa_restoring);          // make restoring
               beam

        if( pad_factor == 1)
        {
                arrange_ft( new_model[0] , imsize );      // arrange beam so that the convolution will work well
        }

        ft_beam(new_model[0] , dirty_beam_ft , imsize , pad_factor , forward_transform , double_buff , complex_buff); //
               get ft of restoring beam


        for(i=0;i<npol;i++)     // calculate convolved final models
        {
                convolve( best_model[i] , dirty_beam_ft , imsize , pad_factor  , convolved_model[i] , forward_transform ,
                       backward_transform , double_buff , complex_buff);
        }


        temp = bmaj_restoring * bmin_restoring * M_PI / ( 4.0 * log(2) * cell * cell ); // set time = number of pixels in
               restoring beam
        // there is a good possiblity that temp = pixels per beam, but if the restoring beam is different to the initial
               beam, then the units of Jy/Beam in the residuals need to be converted to Jy/restoring beam.

        temp = temp / pixels_per_beam;

        cout<<endl<<endl<<"Ratio of restoring beam to \"natural\" beam = "<<temp<<endl;

        #pragma omp parallel for collapse(2)
        for(i=0;i<npol;i++)     // add in residuals
        {
                for(j=0;j<imsize2;j++)
                {
                        current_residuals[i][j] = − current_residuals[i][j] * temp;      // residuals defined as dirty model
                               − dirty map, redefine and rescale
                        new_model[i][j] = convolved_model[i][j] + current_residuals[i][j];                       // save final
                                map in new_model
                }
        }


        cout<<"Writing out maps..."<<endl;


        for(i=0;i<npol;i++)     // write out convolved models
        {
                line.assign(output_name);
                line.append("_Model_S");
                line.append(int2str(i+1));
                line.append(".fits");
                err = cfits_write( line.c_str() , best_model[i] , imsize , cell , ra , dec , centre_shift , rotations , freq
                       , freq_delta , stokes[i] , object , observer , telescope , equinox , date_obs , history ,
                       bmaj_restoring , bmin_restoring , bpa_restoring , ctr , false);
                if(err!=0)
                {
                        cout<<endl<<"Error detected in attempting to write to "<<line<<", err = "<<err<<endl<<endl;
                }


                line.assign(output_name);
                line.append("_Convolved_S");
                line.append(int2str(i+1));
                line.append(".fits");
                err = cfits_write( line.c_str() , convolved_model[i] , imsize , cell , ra , dec , centre_shift , rotations ,
                       freq , freq_delta , stokes[i] , object , observer , telescope , equinox , date_obs , history ,
                       bmaj_restoring , bmin_restoring , bpa_restoring , ctr , true);
                if(err!=0)
                {
                        cout<<endl<<"Error detected in attempting to write to "<<line<<", err = "<<err<<endl<<endl;
                }
```

```
            line.assign(output_name);
            line.append("_Residual_S");
            line.append(int2str(i+1));
            line.append(".fits");
            err = cfits_write( line.c_str() , current_residuals[i] , imsize , cell , ra , dec , centre_shift , rotations
                    , freq , freq_delta , stokes[i] , object , observer , telescope , equinox , date_obs , history ,
                    bmaj_restoring , bmin_restoring , bpa_restoring , ctr , true);
            if(err!=0)
            {
                    cout<<endl<<"Error detected in attempting to write to "<<line<<", err = "<<err<<endl<<endl;
            }


            line.assign(output_name);
            line.append("_Final_S");
            line.append(int2str(i+1));
            line.append(".fits");
            err = cfits_write( line.c_str() , new_model[i] , imsize , cell , ra , dec , centre_shift , rotations , freq ,
                     freq_delta , stokes[i] , object , observer , telescope , equinox , date_obs , history ,
                    bmaj_restoring , bmin_restoring , bpa_restoring , ctr , true);
            if(err!=0)
            {
                    cout<<endl<<"Error detected in attempting to write to "<<line<<", err = "<<err<<endl<<endl;
            }

    }





    // free up memory

    free_mem_exit:
    cout<<"Freeing up memory..."<<endl;

    fftw_destroy_plan(forward_transform);
    fftw_destroy_plan(backward_transform);

    fftw_cleanup_threads();

    for(i=0;i<npol;i++)
    {
            delete[] dirty_map[i];
            delete[] current_model[i];
            delete[] new_model[i];
            delete[] current_residuals[i];
            delete[] new_residuals[i];
            delete[] convolved_model[i];
            delete[] best_model[i];
    }
    delete[] dirty_map;
    delete[] current_model;
    delete[] new_model;
    delete[] current_residuals;
    delete[] new_residuals;
    delete[] convolved_model;
    delete[] best_model;
    delete[] chi2_rms;
    delete[] rms_theoretical;
    delete[] zero_spacing_flux;

    delete[] default_map2;
    delete[] dirty_beam;
    fftw_free(dirty_beam_ft);
    fftw_free(double_buff);
    fftw_free(complex_buff);

    if(npol > 1)
    {
            delete[] filename_pol;
    }
```

```
        cout<<"End of program"<<endl;

        if(err == 0 && !converged)
        {
                return(2);
        }
        else
        {
                return(err);
        }
}

/*
        write_csv

        write_csv writes out a matrix into a comma separated volume file

        inputs:
                filename = the name of the output file
                array = the matrix to be written out
                imsize = the dimension of the matrix
        outputs:
                on return = err (0 if no error)
*/


int write_csv(string filename, double* array, int imsize)
{
        int i,j,k;
        int err;
        ofstream fout;
        char* cfilename;

        cfilename=new char[filename.length()];          // get name into C format
        strcpy(cfilename,filename.c_str());

        fout.open(cfilename,ios::out);
        err=fout.is_open();

        k=0;
        for(i=0;i<imsize;i++)
        {
                fout<<array[k];
                k++;
                for(j=1;j<imsize;j++)
                {
                        fout<<","<<array[k];
                        k++;
                }
                fout<<endl;
        }
        fout.close();

        delete[] cfilename;
        return(err);
}

/*
        int2str converts an integer to a string
*/

string int2str (int num)
{
        stringstream ss;

        ss<<num;

        return(ss.str());
}


/*
        arrange_ft

        arrange_ft changes an image to "wrap-around" order by swapping quadrants 1 --> 3, and 2 --> 4
        This can be useful in performing an FFT
```

```
	At the moment this function is needed only for FFTs with no zero padding (not the default case)

	inputs:

	arr = the image (stored in doubles) that needs to be rearranged
	imsize = the dimension of the image (length of one side)

	outputs:

	arr = the image now in "wrap-around" order
*/


void arrange_ft(double* arr, int imsize)
{
	int i,j,k,l;
	int half=(imsize/2);	// 0.5*dimension

	double temp;


	#pragma omp parallel for collapse(2) private(k,l,temp)
	for(i=0;i<half;i++)	// swap values from q1 to q3, and q2 to q4
	{
		for(j=0;j<half;j++)
		{
			k=i*imsize+j;	// location in q1
			l=(i+half)*imsize+half+j;	// location in q3

			temp = arr[k];	// re
			arr[k] = arr[l];
			arr[l] = temp;


			k=i*imsize+j+half;	// location in q2
			l=(i+half)*imsize+j;	// location in q4

			temp = arr[k];	// re
			arr[k] = arr[l];
			arr[l] = temp;


		}
	}
}


/*
	"convolve"

	Convolves data with the repsonse function response and copies output to output.

	Inputs:

	data = data to be convolved
	response = response function (same size as data)
	imsize = size of data, response and output
	pad_factor = amount of zero padding (1 = no zero padding, 2 = imsize zero padding, etc.)
	forward_transform = fftw plan for forward transform
	backward_transform = fftw plan for backward transform
	double_buff = work space for fftw plans
	complex_buff = work space for fftw plans

	Outputs:

	output = the convolved data
	on return, 0

*/

int convolve(double* data, fftw_complex* response, int imsize, int pad_factor, double* output, fftw_plan&
	forward_transform, fftw_plan& backward_transform, double* double_buff, fftw_complex* complex_buff)
{
	int i, j, k;

	int imsize_pad = pad_factor * imsize;
	double temp1, temp2;
```

```
        #pragma omp parallel for collapse(2) private(k) // initialise (padded) working array for forward transform
        for(i=0;i<imsize_pad;i++)
        {
                for(j=0;j<imsize_pad;j++)
                {
                        k = (i * imsize_pad) + j ;
                        if( ( i < imsize ) && ( j <imsize ) )
                        {
                                double_buff[k] = data[ (i * imsize) + j ];
                        }
                        else
                        {
                                        double_buff[k] = 0.0;
                        }
                }
        }



        fftw_execute( forward_transform );       // forward transform



        j = pad_factor * imsize * ( pad_factor * imsize / 2 + 1 );        // this is the point where Hermitian conjugacy
               means the FFTW routine has stopped spitting out more data
                                                                           // no need to process the additional data, as the c2r
                                                                                   transform assumes conjugacy too

        k = imsize_pad * imsize_pad;

        #pragma omp parallel for private(temp1,temp2)
        for( i=0 ; i < j ; i++ )          // convolution theorem
        {
                temp1 = ( complex_buff[i][0] * response[i ][0] - complex_buff[i][1] * response[i ][1] ) / double( k );
                temp2 = ( complex_buff[i][1] * response[i ][0] + complex_buff[i][0] * response[i ][1] ) / double( k );     //
                        the k scales data_pad and response_pad correctly

                complex_buff[i][0] = temp1;
                complex_buff[i][1] = temp2;
        }


        fftw_execute( backward_transform );    // inverse transform


        if(pad_factor==1)       // k is the displacement necessary to read from the centre of the padded array
        {
                k = 0;
        }
        else
        {
                k = (pad_factor - 1) * imsize / 2;
        }


        #pragma omp parallel for collapse(2)    // copy data to result , reading from the "centre" of the padded output
        for(i=0;i<imsize;i++)
        {
                for(j=0;j<imsize;j++)
                {
                        output[ (i * imsize) + j ] = double_buff[ ( (i + k) * imsize_pad) + j + k];
                }
        }



        return(0);
}

/*
        ft_beam

        ft_beam FFTs a beam (dirty or restoring) and saves the result in a complex array
        As the FFT is real to complex, the array does not need the same dimensions as the input due to the Hermitian
               conjugacy of the resulting FT frequencies
        This FFT is done outside the convolution function because the beam only needs to be FFTed once. This saves processing
                time.

        inputs:

        beam = the beam image to be FFTed (dirty or restoring)
        imsize = the length/width of the image
        pad_factor = amount of zero padding (1 = no zero padding, 2 = imsize zero padding, etc.)
        plan = fftw plan for forward transform
        double_buff = work space for fftw plans
        complex_buff = work space for fftw plans

        outputs:

        ft_beam = the FFT of the beam
        on return, 0

*/

int ft_beam(double* beam, fftw_complex* ft_beam, int imsize, int pad_factor, fftw_plan& plan, double* double_buff,
       fftw_complex* complex_buff)
{
        int imsize_pad = pad_factor * imsize;
        int i, j, k;

        #pragma omp parallel for collapse(2) private(k) // initialise (padded) working array for transform
        for(i=0;i<imsize_pad;i++)
        {
                for(j=0;j<imsize_pad;j++)
                {
                        k = (i * imsize_pad) + j;
                        if( ( i < imsize ) && ( j < imsize ) )
                        {
                                double_buff[k] = beam[ (i * imsize) + j ];
                        }
                        else
                        {
                                double_buff[k] = 0.0;
                        }
                }
        }

//      pad_image( beam, double_buff, imsize, pad_factor);      // initialise (padded) working array for forward transform,
       using padding correctly

        fftw_execute(plan);       // forward transform

        j = pad_factor * imsize * ( pad_factor * imsize / 2 + 1 );        // only read in as much as necessary, taking the
               Herm. cong. into account

        for(i=0;i<j;i++)          // copy result to output array
        {
                ft_beam[i][0] = complex_buff[i][0];
                ft_beam[i][1] = complex_buff[i][1];
        }

        return(0);
}

/*
        "zero_array"

        sets array to zero. useful for initial declaration of residuals

        inputs:

        array = map
        imsize = length of one side of the array

        outputs:

        array = map, now zero everywhere
*/

int zero_array(double* array, int imsize)
```

```
{
        #pragma omp parallel for
        for(int i = 0; i < imsize ; i++ )
        {
                array[0] = 0.0;
        }

        return(0);
}

/*
        "get_residual_map"

        gets the residual map of the convolved model − the dirty map (units = Jy/beam)

        inputs:

        dirty_map = dirty map, nothing done to it (units = Jy/beam)
        convolved_model = the current model convolved with the dirty beam (units = Jy/beam)
        size = the number of pixels

        outputs:

        residual_map = difference between convolved model and dirty map
        on return, 0
*/

int get_residual_map(double* dirty_map , double* convolved_model , double* residual_map , int imsize, int
      ignore_pixels)
{
        int i , j , ctr;
        int right_pixel_limit = imsize − ignore_pixels;

        #pragma omp parallel for collapse (2) private(ctr)
        for(i=ignore_pixels;i<right_pixel_limit;i++)
        {
                for(j=ignore_pixels;j<right_pixel_limit;j++)
                {
                        ctr = i * imsize + j;
                        residual_map[ctr] = convolved_model[ctr] − dirty_map[ctr];
                }
        }

        return(0);
}

/*
        "get_info"

        Calculates inner products. Finds the current total flux and total rms residuals

        Inputs:

        model = Current model imap
        residual = Current residual maps (I,Q,U,V)
        default_map2 = Default imap squared (normally flat)
        alpha = Lagrange parameter for chi2 for Stokes I
        beta = Lagrange parameter for chi2 for polarisation Stokes parameters
        gamma = Lagrange paramter for flux conservation
        imsize2 = The number of pixels in the images
        npol = The number of polarisations
        q = Factor converting Jy/pix to Jy/beam

        outputs:

        grad = Structure holding all the inner products that need to be calculated
        chi2_rms = Calculated rms residual for I,Q,U,V
        imin = Minimum Stokes I flux
        imax = Maximum Stokes I flux
        on return = error (0 if none)
*/

int get_info(double** model , double** residual, double* default_map2 , gradient_structure& grad , double* chi2_rms,
       double& total_flux , double alpha , double beta , double gamma , double& imin, double& imax , int imsize , int
       npol, double q, int ignore_pixels)
{
        double metric,dh,dh2,de,df,dg,temp,p,pexp,plog,m;
        double EE , EF , EG , EH , FF , FG , FH , GG , GH , HH , II;
        int i , j , k , ctr , err;
        int right_pixel_limit = imsize − ignore_pixels;

        EE = 0.0;
        EF = 0.0;
        EG = 0.0;
        EH = 0.0;
        FF = 0.0;
        FG = 0.0;        // note EF and FG will stay as zero
        FH = 0.0;
        GG = 0.0;
        GH = 0.0;
        HH = 0.0;
        II = 0.0;

        temp=0.0;

        err=0;

        #pragma omp parallel for collapse(2) reduction( +: EE , EG , EH , FF , FH , GG , GH , HH ,temp) private(ctr,k,p,
                m,plog,pexp,dh,dh2,de,df,dg,metric)
        for(i=ignore_pixels;i<right_pixel_limit;i++)
        {
                for(j=ignore_pixels;j<right_pixel_limit;j++)
                {
                        ctr = i ∗ imsize + j;

                        p = 0.0;
                        for(k=1;k<npol;k++)
                        {
                                p += (model[k][ctr] ∗ model[k][ctr]) ;    // find polarised intensity for each pixel
                        }
                        p = sqrt(p);


                        if(p > 0.01 ∗ model[0][ctr])
                        {
                                plog = 0.5 ∗ log( ( model[0][ctr] − p ) / ( model[0][ctr] + p) ) / p;
                                pexp = ( ( model[0][ctr] / (p ∗ p − model[0][ctr] ∗ model[0][ctr] ) ) − plog ) / (p ∗ p);
                                                // calculate some expressions needed for evaluating the derivative of the
                                        entropy
                        }
                        else
                        {
                                m = p / model[0][ctr];
                                plog = − (1.0 + (m ∗ m / 3.0) ) / model[0][ctr];
                                pexp = − ( (2.0 / 3.0) + 0.8 ∗ m ∗ m) / (model[0][ctr] ∗ model[0][ctr] ∗ model[0][ctr]) ;
                                                // Taylor series expansion of above for small m (see Mathematica)
                        }


                        for(k=0;k<npol;k++)
                        {
                                if(k==0)        // Stokes I only
                                {
                                        dh = −0.5 ∗ log( ( model[0][ctr] ∗ model[0][ctr] − p ∗ p ) / ( default_map2[ctr] ) );
                                                // derivative of the entropy contribution of pixel
                                        dh2 = − model[0][ctr] / ( model[0][ctr] ∗ model[0][ctr] − p ∗ p);
                                                                // second derivative of the entropy contribution of
                                                pixel
                                        de = 2.0 ∗ residual [0][ ctr ];
                                                // derivative of the Stokes I chi2 contribution of the pixel
                                        df = 0.0;
                                                // derivative of the Stokes Q,U,V chi2 contribution of the pixel
                                        dg = 1.0;
                                                // derivative of the flux conservation term contribution of the pixel
                                        metric = 1.0/( 2.0 ∗ q ∗ alpha − dh2);
                                                // metric of the minimisation at this point
                                }
                                else    // Other Stokes parameters (same idea)
                                {
                                        dh = model[k][ctr] ∗ plog;
                                        dh2 = plog + model[k][ctr] ∗ model[k][ctr] ∗ pexp;
                                        de = 0.0;
                                        df = 2.0 ∗ residual [k][ ctr ];
                                        dg = 0.0;
```

```
                                        metric = 1.0/( 2.0 * q * beta - dh2);
                                }

                                EE += de*metric*de;
                                EG += de*metric*dg;
                                EH += de*metric*dh;   // contributions to the inner products
                                FF += df*metric*df;
                                FH += df*metric*dh;
                                GG += dg*metric*dg;
                                GH += dg*metric*dh;
                                HH += dh*metric*dh;
                                temp += metric; // inner product 1.1
                        }
                }
        }

        total_flux=0.0;


        imin=9999.0;
        imax=-9999.0;

        for(j=0;j<npol;j++)
        {
                chi2_rms[j]=0.0;
        }

        for(i=ignore_pixels;i<right_pixel_limit;i++)   // find max, min, total flux and residual chi2 term
        {
                for(j=ignore_pixels;j<right_pixel_limit;j++)
                {

                        ctr = i * imsize + j;

                        for(k=0;k<npol;k++)
                        {
                                chi2_rms[k] += residual[k][ctr] * residual[k][ctr];
                        }

                        total_flux += model[0][ctr];
                        imin=min(imin,model[0][ctr]);
                        imax=max(imax,model[0][ctr]);
                }
        }


        // save inner products into the structure

        grad.EJ = EH - alpha * EE - beta * EF - gamma * EG;
        grad.FJ = FH - alpha * EF - beta * FF - gamma * FG;
        grad.GJ = GH - alpha * EG - beta * FG - gamma * GG;
        grad.JJ = HH + alpha * alpha * EE + beta * beta * FF + gamma * gamma * GG - 2.0 * alpha * EH - 2.0 * beta *
                FH - 2.0 * gamma * GH + 2.0 * alpha * beta * EF + 2.0 * alpha * gamma * EG + 2.0 * beta * gamma * FG;
        II = HH + alpha * alpha * EE + beta * beta * FF + gamma * gamma * GG;

        if(II < 0)
        {
                II=temp;
        }

        grad.EE = EE;
        grad.EF = EF;
        grad.EG = EG;
        grad.EH = EH;
        grad.FF = FF;
        grad.FG = FG;
        grad.FH = FH;
        grad.GG = GG;
        grad.GH = GH;
        grad.HH = HH;
        grad.II = II;


        return(err);
}

/*
```

```
        new_ABG

        new_ABG gets new values for alpha, beta and gamma
        This function uses LAPACK functions dgetrf (factorise into LU) and dgetrs (solve A x = b when A is LU factorised)

        inputs:
                grad = a structure containing all the norms calculated in get_info
                delta_E = the difference between current residuals and expected residuals for Stokes I
                delta_F = the difference between current residuals and expected residuals for Stokes Q and U
                delta_G = the difference between current flux and expected flux
                conserve_flux = a boolean that is true if flux is being conserved, false otherwise
                npol = the number of polarisations being deconvolved
        outputs:
                alpha = the Lagrangian parameter associated with minimising Stokes I residuals
                beta = the Lagrangian parameter associated with minimising Stokes Q and U residuals
                gamma = the Lagrangian parameter associated with conserving the Stokes I flux
                on return, 0
*/


int new_ABG(gradient_structure grad , double delta_E , double delta_F , double delta_G , double& alpha , double&
      beta , double& gamma , bool conserve_flux , int npol)
{
        double l , arg;
        double alpha1 , beta1 , gamma1;
        double alpha2 , beta2 , gamma2;
        double dalpha , dbeta , dgamma;

        double epsilon1 = 0.1;  // tolerance for checking if the changes in alpha etc. are "small enough". This is the MEM'
                s equivalent of Gain. Keep very small.
        double epsilon2 = 0.05; // tolerance for checking if the changes in l are "small enough". This is the MEM's
                equivalent of Gain.

        int err;

        int n;  // parameters for LAPACK functions dgetrf (factorise into LU) and dgetrs (solve A x = b when A is LU
                factorised)
        char trans='T'; // indicate that the transpose should be used in LAPACK functions (row major to column major)
        int nrhs = 1;


        l = fabs( grad.JJ / grad.II );   // a measure of how quickly the cost function is changing scaled with how quickly the
                I map is changing

        if(alpha <= 0)
        {
                l=0.0;
        }

        if(npol ==1 )
        {
                if(conserve_flux)
                {
                        n=2;
                        double A[4] = { grad.EE , grad.EG , grad.EG , grad.GG};
                        double b[2] = { grad.EH , grad.GH};
                        int ipiv [2];


                        dgetrf_(&n , &n , A , &n , ipiv, &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrf_ : Error performing LU factorisation."<<endl;
                                return(1);
                        }


                        dgetrs_( &trans , &n , &nrhs , A , &n , ipiv , b , &n , &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrs_ : Error solving linear equation."<<endl;
                                return(1);
                        }

                        alpha1 = b[0];
                        beta1 = 0.0;
                        gamma1 = b[1];
                        b[0] = delta_E + grad.EJ;      //      reset b
                        b[1] = delta_G + grad.GJ;

                        dgetrs_( &trans , &n , &nrhs , A , &n , ipiv , b , &n , &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrs_ : Error solving linear equation."<<endl;
                                return(1);
                        }


                        dalpha = b[0];
                        dbeta = 0.0;
                        dgamma = b[1];
                }
                else
                {
                        // no beta or gamma
                        // need only to solve gradE.gradJ = 0, i.e. gradE.(gradH - alpha * grad E)=0, or alpha = grad HE /
                              grad EE

                        alpha1 = grad.EH / grad.EE;
                        beta1 = 0.0;
                        gamma1 = 0.0;

                        // need to solve grad EE * dalpha = delta E + grad EJ

                        dalpha = ( delta_E + grad.EJ ) / grad.EE;
                        dbeta = 0.0;
                        dgamma = 0.0;
                }
        }
        else
        {
                if(conserve_flux)
                {
                        // If conserving flux solve 3x3 matrix to find changes in alpha, beta, gamma


                        n=3;
                        double A[9] = { grad.EE , grad.EF , grad.EG , grad.EF , grad.FF , grad.FG , grad.EG , grad.FG ,
                              grad.GG };
                        double b[3] = { grad.EH , grad.FH , grad. GH};
                        int ipiv [3];


                        dgetrf_(&n , &n , A , &n , ipiv, &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrf_ : Error performing LU factorisation."<<endl;
                                return(1);
                        }

                        dgetrs_( &trans , &n , &nrhs , &A[0] , &n , ipiv , &b[0] , &n , &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrs_ : Error solving linear equation."<<endl;
                                return(1);
                        }

                        alpha1 = b[0];
                        beta1 = b[1];
                        gamma1 = b[2];


                        b[0] = (delta_E + grad.EJ);    //      reset b
                        b[1] = (delta_F + grad.FJ);
                        b[2] = (delta_G + grad.GJ);

                        dgetrs_( &trans , &n , &nrhs , A , &n , ipiv , b , &n , &err );

                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrs_ : Error solving linear equation."<<endl;
                                return(1);
                        }

                        dalpha = b[0];
                        dbeta = b[1];
                        dgamma = b[2];
                }
                else
                {
                        n=2;
                        double A[4] = { grad.EE , grad.EF , grad.EF , grad.FF };
                        double b[2] = { grad.EH , grad.FH};
                        int ipiv [2];



                        dgetrf_(&n , &n , A , &n , ipiv, &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrf_ : Error performing LU factorisation."<<endl;
                                return(1);
                        }


                        dgetrs_( &trans , &n , &nrhs , A , &n , ipiv , b , &n , &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrs_ : Error solving linear equation."<<endl;
                                return(1);
                        }


                        alpha1 = b[0];
                        beta1 = b[1];
                        gamma1 = 0.0;


                        b[0] = delta_E + grad.EJ;       //      reset b
                        b[1] = delta_F + grad.FJ;

                        dgetrs_( &trans , &n , &nrhs , A , &n , ipiv , b , &n , &err );
                        if(err != 0)
                        {
                                cout<<"Error : new_ABG : dgetrs_ : Error solving linear equation."<<endl;
                                return(1);
                        }

                        dalpha = b[0];
                        dbeta = b[1];
                        dgamma = 0.0;
                }
        }



        // check if change in alpha is "small enough"

        arg = grad.EJ * grad.EJ - (grad.JJ - epsilon1 * grad.II) * grad.EE;
        if( arg > 0 )
        {
                arg = sqrt(arg);
                dalpha = max( (grad.EJ - arg) / grad.EE , min( (grad.EJ + arg) / grad.EE , dalpha ) );
        }
        else
        {
                dalpha=max( dalpha , 0.0);
        }
        alpha2 = alpha + dalpha;

        // check if change in beta is "small enough"

        if(npol>1)
        {
                arg = grad.FJ * grad.FJ - (grad.JJ - epsilon1 * grad.II) * grad.FF;
                if( arg > 0 )
                {
                        arg = sqrt(arg);
                        dbeta = max( (grad.FJ - arg) / grad.FF , min( (grad.FJ + arg) / grad.FF , dbeta ) );
                }
                else
                {
                        dbeta=max( dbeta , 0.0);
                }
                beta2 = beta + dbeta;
        }

        // check if change in gamma is "small enough"

        if(conserve_flux)
        {
                arg = grad.GJ * grad.GJ − (grad.JJ − epsilon1 * grad.II) * grad.GG;
                if( arg > 0 )
                {
                        arg = sqrt(arg);
                        dgamma = max( (grad.GJ − arg) / grad.GG , min( (grad.GJ + arg) / grad.GG , dgamma ) );
//                      cout<<"Max, min = "<<(grad.GJ + arg) / grad.GG<<" , "<<(grad.GJ − arg) / grad.GG<<endl;
                }
                else
                {
                        dgamma=max( dgamma , 0.0);
                }
                gamma2 = gamma + dgamma;
        }



        // decide on alpha

        if( l >= epsilon2 or alpha2 <=0.0)
        {
                alpha=max(alpha1,0.0);
        }
        else
        {
                alpha=max(alpha2,0.0);
        }


        // decide on beta


        if(npol>1)
        {
                if( l >= epsilon2 or beta2 <=0.0)
                {
                        beta=max(beta1,0.0);
                }
                else
                {
                        beta=max(beta2,0.0);
                }
        }

        // decide on gamma

        if(conserve_flux)
        {
                if( l >= epsilon2 or gamma2 <=0.0)
                {
                        gamma=max(gamma1,0.0);
                }
                else
                {
                        gamma=max(gamma2,0.0);
                }
        }


        return(0);
}

/*
        cal_step

        cal_step finds the next step to take for the image using the Newton−Ralphson method
        It also uses the plog, m and pexp terms to try to be computationally effecient

        inputs:
                model = the model Stokes I, Q and U maps
                residuals = the residual Stokes I, Q and U maps
                default_map2 = the square of the default map
                imsize2 = the number of pixels in the new map
                alpha = the Lagrangian parameter associated with minimising Stokes I residuals
                beta = the Lagrangian parameter associated with minimising Stokes Q and U residuals
                gamma = the Lagrangian parameter associated with conserving the Stokes I flux
                q = the number of pixels in a beam
                npol = the number of polarisations being deconvolved

        outputs:
                J0 = a measure of the change in J due to the step taken
                step_map = the steps for the Stokes I,Q,U model maps
                on return, 0
*/

int cal_step(double** model , double** residual, double* default_map2 , double alpha , double beta , double gamma ,
      int imsize , int npol, double q , double& J0 , double** step_map, int ignore_pixels)
{
        double p , plog , pexp , m , dh , dh2 , gradJ , metric , step;
        int i, j, k, ctr;
        int right_pixel_limit = imsize - ignore_pixels;

        double J0_temp = 0.0;

        #pragma omp parallel for collapse(2) reduction(+:J0_temp) private(ctr,k,p,m,plog,pexp,dh,dh2,gradJ,metric,step) //
                eff : rewrite to reduce operations ( default map^2, 2* q )
        for(i=ignore_pixels; i < right_pixel_limit; i++)
        {
                for(j=ignore_pixels; j < right_pixel_limit; j++)
                {
                        ctr = i * imsize + j;

                        p = 0.0;
                        if(npol>1)
                        {
                                for(k=1;k<npol;k++)
                                {
                                        p += (model[k][ctr] * model[k][ctr]);
                                }
                                p = sqrt(p);

                                if(p > 0.01 * model[0][ctr])
                                {
                                        plog = 0.5 * log( ( model[0][ctr] - p ) / ( model[0][ctr] + p) ) / p;
                                        pexp = ( ( model[0][ctr] / (p * p - model[0][ctr] * model[0][ctr] ) ) - plog ) / (p
                                              * p);
                                }
                                else
                                {
                                        m = p / model[0][ctr];
                                        plog = - (1.0 + (m * m / 3.0) ) / model[0][ctr];
                                        pexp = - ( (2.0 / 3.0) + 0.8 * m * m) / (model[0][ctr] * model[0][ctr] * model[0][ctr
                                              ]);          // Taylor series expansion of above for small m (see Mathematica)
                                }
                        }

                        for(k=0;k<npol;k++)
                        {
                                if(k==0)
                                {
                                        dh = - 0.5 * log( (model[0][ctr] * model[0][ctr] - p * p) / (default_map2[ctr]));
                                        dh2 = - model[0][ctr] / (model[0][ctr] * model[0][ctr] - p * p);
                                        gradJ = dh - 2.0 * alpha * residual[0][ctr] - gamma;
                                        metric = 1.0 / ( 2.0 * q * alpha - dh2);
                                }
                                else
                                {
                                        dh = model[k][ctr] * plog;
                                        dh2 = plog + model[k][ctr] * model[k][ctr] * pexp;
                                        gradJ = dh - 2.0 * beta * residual[k][ctr];
                                        metric = 1.0/( 2.0 * q * beta - dh2);
                                }
                        step = metric * gradJ;
                        J0_temp += gradJ * step;
                        step_map[k][ctr] = step;
                    }
                }
            }

            J0 = J0_temp;

            return(0);
}

/*
            take_step

            take_step actually takes the step that was generated by cal_step
            it also enforces a minimum flux and makes sure that the fractional polarisation does not leave the region 0<m<1

            inputs:
                model = the model Stokes I, Q and U maps
                step = the steps for the Stokes I, Q and U model maps
                step_length = the factor by which the steps should be scaled
                step_limit = the maximum allowed change
                imsize2 = the number of pixels in a map
                npol = the number of polarisations being deconvolved

            outputs:
                model = the new model I, Q and U maps (after the step)
                on return, 0
*/

int take_step(double** model , double** step , double step_length , double step_limit , int imsize , int npol, int
        ignore_pixels)
{

            // First declare variables

            double pold , pnew , iold , inew , istep , factor;
            int i,j,k,ctr;
            int right_pixel_limit = imsize - ignore_pixels;

            // Start openmp for loop if possible to efficiently loop over every pixel

            #pragma omp parallel for collapse(2) private(pold, pnew, k, ctr, iold, istep, inew, factor)
            for(i=ignore_pixels; i < right_pixel_limit; i++)
            {
                for(j=ignore_pixels; j < right_pixel_limit; j++)
                {
                    ctr = i * imsize + j;
            /*
                    First take the Stokes I step
                    Record the old value
                    Set istep = step (which is the step length times the recommended step). If the step is negative,
                        make sure it's not too negative
                    Update Stokes I value with new flux, or min_flux - whichever is greater
            */

                    iold = model[0][ctr];
                    istep = step_length * max( step[0][ctr] , -0.9 * model[0][ctr] / step_limit);
                    step[0][ctr] = max( model[0][ctr] + istep , min_flux );
                    inew = step[0][ctr];


            //      Polarisation section

                    if(npol > 1 )
                    {
                        // Find P, the total polarised flux, for the old and new cases

                        pold = 0.0;
                        pnew = 0.0;

                        for(k=1;k<npol;k++)
                        {
                            pold += model[k][ctr] * model[k][ctr];
                            pnew += pow( model[k][ctr] + step_length * step[k][ctr] , 2);
                        }
                        pold = sqrt(pold);
                        pnew = sqrt(pnew);

                        // Set a factor to ensure that m, the fractional polarisation, is an element of [0,1]

                        if( pnew < inew )
                        {
                                factor = min( 1.0 , (0.4 * inew / pnew + 0.4 * pold / pnew) );
                        }
                        else
                        {
                                factor = 0.8 * inew / pnew;
                        }

                        // Update the Q and U maps with their new values

                        for(k=1;k<npol;k++)
                        {
                                step[k][ctr] = factor * ( model[k][ctr] + step_length * step[k][ctr] );
                        }
                }
        }
    }

    return(0);
}

/*
    check_step

    check_step checks to see if the previous step length was near optimal

    inputs:
        old_model = the model Stokes I, Q and U maps from before the last step
        new_model = the model Stokes I, Q and U maps from after the last step
        new_residuals = the residual Stokes I, Q and U maps from after the last step
        default_map2 = the square of the default map
        imsize2 = the number of pixels in the new map
        alpha = the Lagrangian parameter associated with minimising Stokes I residuals
        beta = the Lagrangian parameter associated with minimising Stokes Q and U residuals
        gamma = the Lagrangian parameter associated with conserving the Stokes I flux
        q = the number of pixels in a beam
        npol = the number of polarisations being deconvolved

    outputs:
        J1 = a measure of the change in J due to the step taken. To be compared with J0 to see if the step should
            have been bigger or smaller.
        on return, 0 if no error, 1 if error detected
*/

int check_step(double** old_model , double** new_model , double** new_residual, double* default_map2 , double
    alpha , double beta , double gamma , int imsize , int npol, double q , double& J1, int ignore_pixels)
{
    // Declare some variables

    double p , plog , m , dh , gradJ , step;
    int i , j , k , ctr;
    int right_pixel_limit = imsize - ignore_pixels;

    double J1_temp = 0.0;

    // Start openmp for loop if possible to efficiently loop over every pixel

    #pragma omp parallel for collapse(2) reduction(+:J1_temp) private(ctr,k,m,p,plog,dh,gradJ,step)
    for(i=ignore_pixels; i < right_pixel_limit; i++)
    {
        for(j=ignore_pixels; j < right_pixel_limit; j++)
        {
            ctr = i * imsize + j;

            // Find the total polarised flux, P


            p = 0.0;
            for(k=1;k<npol;k++)
            {
```

```
                                        p += (new_model[k][ctr] * new_model[k][ctr]);
                                }
                                p = sqrt(p);

                /*
                                Need to implement dH and gradJ according to the MEM formulae
                                To do this  efficiently , make use of plog and m to prevent calculating the same thing more than once
                                If the total polarised flux is sufficiently small, use the Taylor expansion of the logs to save
                                        computation
                */

                                if(p > 0.01 * new_model[0][ctr])
                                {
                                        plog = 0.5 * log( ( new_model[0][ctr] - p ) / ( new_model[0][ctr] + p) ) / p;
                                }
                                else
                                {
                                        m = p / new_model[0][ctr];
                                        plog = - (1.0 + (m * m / 3.0) ) / new_model[0][ctr];
                                }

                                for(k=0;k<npol;k++)
                                {
                                        if(k==0)
                                        {
                                                dh = -0.5 * log( (new_model[0][ctr] * new_model[0][ctr] - p * p ) / (default_map2[
                                                        ctr]));
                                                gradJ = dh - 2.0 * alpha * new_residual[0][ctr] - gamma;
                                        }
                                        else
                                        {
                                                dh = new_model[k][ctr] * plog;
                                                gradJ = dh - 2.0 * beta * new_residual[k][ctr];
                                        }

                                        step = new_model[k][ctr] - old_model[k][ctr];  // Step = difference between the two maps
                                        J1_temp += gradJ * step;                        // The change in J due to this is gradJ
                                }
                        }
                }

                J1 = J1_temp;

                if(J1 == J1)    // make sure J1 is not infinity
                {
                        j = 0;
                }
                else
                {
                        j = 1;
                }

                return(j);
}

/*
        interpolate_models

        interpolate_models interpolates current models and new models if necessary.
        It also enforces a minimum flux and makes sure the polarisation doesn't do anything strange

        inputs:

                current_model = the current model Stokes I, Q, U maps
                new_model = the new model Stokes I, Q, U maps
                frac_new = the weighting factor of the new Stokes I, Q, U maps
                imsize2 = the number of pixels in a map
                npol = the number of polarisations being deconvolved

        outputs:
                current_model = the interpolated model maps
                on return, 0
*/

int interpolate_models(double** current_model , double** new_model , double frac_new , int imsize , int npol, int
        ignore_pixels)
{
    double frac_old , iold , inew , pold , pnew , factor;
    int i, j, k, ctr;
    int right_pixel_limit = imsize − ignore_pixels;

    frac_old = 1 − frac_new;

    #pragma omp parallel for collapse(2) private(ctr, iold , inew , pold , pnew , factor , k)
    for(i=ignore_pixels; i < right_pixel_limit; i++)
    {
        for(j=ignore_pixels; j < right_pixel_limit; j++)
        {
            ctr = i * imsize + j;
            // First interpolate the Stokes I part

            iold = current_model[0][ctr];
            current_model[0][ctr] = max( frac_old * current_model[0][ctr] + frac_new * new_model[0][ctr] ,
                min_flux);
            inew = current_model[0][ctr];

            // Now interpolate Stokes Q and U, making sure that the resulting m is an element of [0,1]

            if(npol > 1)
            {
                pold = 0.0;
                pnew = 0.0;
                for(k=1;k<npol;k++)
                {
                    pold += current_model[k][ctr] * current_model[k][ctr];
                    current_model[k][ctr] = frac_old * current_model[k][ctr] + frac_new * new_model[k
                        ][ctr];
                    pnew += current_model[k][ctr] * current_model[k][ctr];
                }
                pold = sqrt(pold);
                pnew = sqrt(pnew);

                // New P has been found − need a factor to make sure it's safe

                if( pnew < inew )
                {
                    factor = min( 1.0 , (0.4 * inew / pnew + 0.4 * pold / pnew) );
                }
                else
                {
                    factor = 0.8 * inew / pnew;
                }

                // apply interpolation

                if(factor < 1.0)
                {
                    for(k=1;k<npol;k++)
                    {
                        current_model[k][ctr] = factor * current_model[k][ctr];
                    }
                }

            }
        }
    }

    return(0);
}

/*
     interpolate_residuals

     interpolate_residuals interpolates current resdiuals and new resdiuals if necessary.

    inputs:

          current_resdiuals = the current model Stokes I, Q, U maps
          new_resdiuals = the new model Stokes I, Q, U maps
          frac_new = the weighting factor of the new Stokes I, Q, U maps
          imsize2 = the number of pixels in a map
          npol = the number of polarisations being deconvolved

    outputs:
```

```
                current_resdiuals = the interpolated resdiual maps
                on return, 0
*/

int interpolate_residuals(double** current_residuals , double** new_residuals , double frac_new , int imsize2 , int npol)
{
        double frac_old;

        frac_old = 1.0 − frac_new;

        #pragma omp parallel for collapse(2)
        for(int i=0;i<imsize2;i++)
        {
                for(int j=0;j<npol;j++)
                {
                        current_residuals[j][i] = frac_old * current_residuals[j][i] + frac_new * new_residuals[j][i];
                }
        }

        return(0);
}

/*
        copy_model

        copy_model updates either the model or residual maps by just copying the new maps into the old
        All it actually does is swap the pointers
        model2 points at model1's old memory afterwards

        inputs:
                model1 = a pointer to the map to be updated
                model2 = a pointer to the map to be copied

        outputs:
                model1 = model2
                model2 = model1
*/

int copy_model(double**& model1, double**& model2)
{
        double** temp;

        temp = model1;
        model1 = model2;
        model2 = temp; // this actually swaps model 1 and model 2

        return(0);
}

/*
        gen_gauss

        gen_gauss makes a Gaussian distribution using the given beam parameters

        inputs:
                imsize = the dimensions of the matrix
                cellsize = the size of a single cell in degrees
                bmaj = the major axis of the ellipse at the FWHM of the beam in degrees
                bmin = the major axis of the ellipse at the FWHM of the beam in degrees
                bpa = the position angle of the ellipse at the FWHM of the beam in degrees

        output
                matrix = a matrix containing the generated Gaussian
                on return, 0
*/

int gen_gauss(double* matrix, int imsize, double cellsize, double bmaj, double bmin, double bpa)
{
        int imsize2=imsize*imsize;
        int i,j,k;
        double x,y,a,b,c;
        int yorigin = (imsize/2);         // centre of distribution (pixels). Assumed centre is of the form (255,256) for a
              512 map.
        int xorigin = yorigin;

        bpa = 90 − bpa; // convert from CCW measurement used in astronomy
        bpa*=(M_PI/180.0);    // convert to radiens
        bmaj*=((M_PI/180.0)/(2.354820045)); // convert to radiens from degrees and from FWHM to sigma (2*sqrt(2*log(2))
               )) = 2.354820045
        bmin*=((M_PI/180.0)/(2.354820045));

         cellsize *=M_PI/180.0; // convert from degrees to radiens

        a=0.5*(pow(cos(bpa)/bmaj,2)+pow(sin(bpa)/bmin,2));
        b=0.25*sin(2.0*bpa)*(-1.0/pow(bmaj,2)+1.0/pow(bmin,2));
        c=0.5*(pow(sin(bpa)/bmaj,2)+pow(cos(bpa)/bmin,2));


        j=0;
        k=0;

        for(i=0;i<imsize2;i++)
        {
                x=double(k-xorigin)*cellsize;
                y=double(j-yorigin)*cellsize;
                matrix[i]=exp(-(a*pow(x,2)+2.0*b*x*y+c*pow(y,2)));  // 2d gaussian general form
                k++;
                 if(k==imsize)
                {
                        k=0;
                        j++;
                }
        }

        return(0);
}

/*
        average_region

        average_region takes the average of a given region in the map

        inputs:
                map = the map in question
                imsize = the size of the map (one side)
                blcx = the x coord of the bottom left corner
                blcy = the y coord of the bottom left corner
                trcx = the x coord of the top right corner
                trcy = the y coord of the top right corner

        outputs:
                on return = the average of the region
*/

double average_region(double* map, int blcx, int blcy, int trcx, int trcy, int imsize)
{
        int i,j;
        double sum = 0.0;

        trcx++; // increment ends to make the for loop design a bit easier
        trcy++;

        #pragma omp parallel for collapse(2) reduction(+:sum)
        for(i=blcx;i<trcx;i++)
        {
                for(j=blcy;j<trcy;j++)
                {
                        sum += map[j * imsize + i];      // add up over entire region, with openmp if possible
                }
        }

        i = trcx - blcx;          // find number of pixels in region
        j = trcy - blcy;

        i = i *j;

        return(sum/double(i));
}

/*
        rms_region_region

        rms_region takes the root mean square of a given region in the map
        inputs:
                map = the map in question
                imsize = the size of the map (one side)
                blcx = the x coord of the bottom left corner
                blcy = the y coord of the bottom left corner
                trcx = the x coord of the top right corner
                trcy = the y coord of the top right corner

        outputs:
                on return = the rms of the region
*/

double rms_region(double* map, int blcx, int blcy, int trcx, int trcy, int imsize)
{
        int i,j;
        double sum = 0.0;
        double mean;

        trcx++; // increment ends to make the for loop design a bit easier
        trcy++;

        #pragma omp parallel for collapse(2) reduction(+:sum)
        for(i=blcx;i<trcx;i++)
        {
                for(j=blcy;j<trcy;j++)
                {
                        sum += pow(map[j * imsize + i] , 2.0);
                }
        }

        i = trcx - blcx;         // find number of pixels in region
        j = trcy - blcy;

        i = i *j;

        return(sqrt(sum/double(i)));
}

/*
        clip_edges

        clip_edges clips the edges of a map, replacing the values with a single given value

        inputs:
                map = the map in question
                imsize = the dimension of the map (one side)
                replacement_value = the value with which to replace all pixels in the edge region
                edge_limit = the number of pixels from the edge of the map to consider "edge" pixels

        outputs:
                map = the map, now clipped
                return value = 0
*/

int clip_edges(double* map, double replacement_value, int edge_limit, int imsize)
{
        int i,j,k;

        k = imsize - edge_limit;

        #pragma omp parallel for private(j)
        for(i=0;i<imsize;i++)
        {
                for(j=0;j<edge_limit;j++)
                {
                        map[ i * imsize + j] = replacement_value;
                }

                for(j = k; j<imsize;j++)
                {
                        map[ i * imsize + j] = replacement_value;
                }
        }


        #pragma omp parallel for private(i)
        for(j=edge_limit;j<k;j++)
        {
                for(i=0;i<edge_limit;i++)
                {
                        map[ i * imsize + j] = replacement_value;
                }

                for(i = k; i<imsize;i++)
                {
                        map[ i * imsize + j] = replacement_value;
                }
        }

        return(0);
}
```

B.1.4 PMEM Make File

```
# Set compiler to g++.
CC=g++
LINK=g++
# Set options for the compiler
CCFLAGS= −c −O3 −fopenmp
LINKOPTS= −lm −lfftw3 −fopenmp −llapack −lblas −lfftw3_threads −lcfitsio −O3

all: cfits pmem link

cfits_read: cfits_read_map.cpp
        $(CC) $(CCFLAGS) cfits_read_map.cpp

cfits_read_header_map: cfits_read_header_map.cpp
        $(CC) $(CCFLAGS) cfits_read_header_map.cpp

cfits_write: cfits_write.cpp
        $(CC) $(CCFLAGS) cfits_write.cpp

cfits : cfits_read cfits_write cfits_read_header_map

pmem: pmem.cpp
        $(CC) $(CCFLAGS) pmem.cpp

link:
        $(LINK) −o pmem pmem.o cfits_read_map.o cfits_write.o cfits_read_header_map.o $(LINKOPTS)

clean:
        rm −rf *.o pmem
```

B.2 CFITSIO Interface Code

The following sections contains the C++/C source code to interface between the PMEM program and FITS files using the external CFITSIO program.

B.2.1 CFITS Read Map Header

```
/*
    This program is called cfits_write . It interacts with the CFITSIO library to read metadata from a FITS file
    Copyright (C) 2012 Colm Coughlan
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

        INPUTS:
                char* filename : c string = name of FITS file to be read from
        OUTPUTS:
                various metadata

        NB : see FITS file format documentation to see what each parameter means (though AIPS may interpret it differently
                ...)
*/

#include "pmem_heads.hpp"

int cfits_read_header_map(const char* filename , int* dim , double* cell , double* ra , double* dec , double*
        centre_shift , double* rotations , double* freq , double* freq_delta , int* stokes , char* object , char* observer ,
        char* telescope , double* equinox , char* date_obs , double* bmaj , double* bmin , double* bpa , int* ncc)
{

/*
        INPUTS:
                char* tfilename : c string = name of FITS file to be read
        OUTPUTS:
                ra = right ascention
                dec = declination
                key_string = name of source
                freq = frequency
                 cell = cellsize (degrees)
                dim = image size
                bmaj, bmin, bpa = beam information (degrees)
*/
         fitsfile *fptr;

        int status,i;
        int err;
        char comment[FLEN_VALUE];
        char beamhdu[]="AIPS CG ";
        char cchdu[]="AIPS CC ";
        char bmajname[]="BMAJ";
        char bminname[]="BMIN";
        char bpaname[]="BPA";
        int colnum;
        double temp;
        float floatbuff;
        float float_null=0;
        int int_null=0;
        long longbuff;


        status = 0;     // for error processing
        err=0;



        if ( fits_open_file(&fptr,filename, READONLY, &status) )     // open file and make sure it's open
        {
                printf("ERROR : cfits_read_header_map --> Error opening FITS file, error = %d\n",status);
                return(status);
        }

        i=ASCII_TBL;
        if (fits_movabs_hdu(fptr,1,&i,&status))          // move to main AIPS image HDU (assuming it's the first one)
        {
                printf("ERROR : cfits_read_header_map --> Error locating AIPS ACSII table extension, error = %d\n",
                       status);
                printf("ERROR : cfits_read_header_map --> Did you remember to use the AIPS FITAB task instead of
                       FITTP?\n");
                return(status);
        }

        // read in some optional keys

        fits_read_key(fptr,TSTRING,"OBJECT",object,comment,&status);
        fits_read_key(fptr,TSTRING,"OBSERVER",observer,comment,&status);
        fits_read_key(fptr,TSTRING,"TELESCOP",telescope,comment,&status);
        fits_read_key(fptr,TDOUBLE,"EQUINOX",equinox,comment,&status);
        fits_read_key(fptr,TSTRING,"DATE-OBS",date_obs,comment,&status);

        // read in important keys

        status = 0;

        fits_read_key(fptr,TDOUBLE,"OBSRA",ra,comment,&status);
        err+=status;
        fits_read_key(fptr,TDOUBLE,"OBSDEC",dec,comment,&status);
        err+=status;

        fits_read_key(fptr,TDOUBLE,"CDELT1",&temp,comment,&status);
        err+=status;
        cell[0]=fabs(temp);
        fits_read_key(fptr,TDOUBLE,"CRPIX1",&temp,comment,&status);
        err+=status;
        centre_shift[0]=temp;
        fits_read_key(fptr,TDOUBLE,"CROTA1",&temp,comment,&status);
        err+=status;
        rotations[0]=temp;


        fits_read_key(fptr,TDOUBLE,"CRPIX2",&temp,comment,&status);
        err+=status;
        centre_shift[1]=temp;
        fits_read_key(fptr,TDOUBLE,"CROTA2",&temp,comment,&status);
        err+=status;
        rotations[1]=temp;


        fits_read_key(fptr,TDOUBLE,"CRVAL3",freq,comment,&status);
        err+=status;
        fits_read_key(fptr,TDOUBLE,"CDELT3",freq_delta,comment,&status);
        err+=status;


        fits_read_key(fptr,TINT,"CRVAL4",stokes,comment,&status);
        err+=status;



        fits_read_key(fptr,TDOUBLE,"NAXIS1",&temp,comment,&status);
        err+=status;
        dim[0]=int(temp);        // dim is stored as a double in the FITS file

        if(err!=0)
        {
                printf("ERROR : cfits_read_header_map --> Error reading keywords, custom error = %d\n",err);
        }


        // move to AIPS CG HDU for beam information

        if (fits_movnam_hdu(fptr,BINARY_TBL,beamhdu,0,&status))  // move to beam information hdu
        {
//              printf("cfits_read_header_map --> Beam information not found in %s\n",filename);
//              return(status);
                bmaj[0] = 0.0;
                bmin[0] = 0.0;  // changed this because model files don't have any beam information. Should check to make
                        sure beam info is valid in other code
                bpa[0] = 0.0;
        }
        else
        {
                fits_get_colnum(fptr,CASEINSEN,bmajname,&colnum,&status);
                if(status!=0)
                {
```

```
                printf("ERROR : cfits_read_header_map --> Error locating BMAJ information, error = %d\n",
                        status);
        }
        fits_read_col(fptr,TFLOAT,colnum,1,1,1,&float_null,&floatbuff,&int_null,&status);
        if(status!=0)
        {
                printf("ERROR : cfits_read_header_map --> Error reading BMAJ information, error = %d\n",
                        status);
        }
        bmaj[0]=double(floatbuff);

        fits_get_colnum(fptr,CASEINSEN,bminname,&colnum,&status);
        if(status!=0)
        {
                printf("ERROR : cfits_read_header_map --> Error locating BMIN information, error = %d\n",
                        status);
        }
        fits_read_col(fptr,TFLOAT,colnum,1,1,1,&float_null,&floatbuff,&int_null,&status);
        if(status!=0)
        {
                printf("ERROR : cfits_read_header_map --> Error reading BMIN information, error = %d\n",
                        status);
        }
        bmin[0]=double(floatbuff);

        fits_get_colnum(fptr,CASEINSEN,bpaname,&colnum,&status);
        if(status!=0)
        {
                printf("ERROR : cfits_read_header_map --> Error locating BPA information, error = %d\n",
                        status);
        }
        fits_read_col(fptr,TFLOAT,colnum,1,1,1,&float_null,&floatbuff,&int_null,&status);
        if(status!=0)
        {
                printf("ERROR : cfits_read_header_map --> Error reading BPA information, error = %d\n",
                        status);
        }
        bpa[0]=double(floatbuff);
    }



    // move to AIPS CC HDU for clean component information

    if (fits_movnam_hdu(fptr,BINARY_TBL,cchdu,0,&status))      // move to main AIPS UV hdu
    {
//              printf("WARNING : cfits_read_header_map --> No clean component table detected.\n");
        ncc[0]=0;
    }
    else
    {
        fits_get_num_rows(fptr,&longbuff,&status);
        ncc[0]=longbuff;
        if(status!=0)
        {
                printf("ERROR : cfits_read_header_map --> Error reading number of clean components, error =
                        %d\n",status);
        }
    }




    status=0;
    if ( fits_close_file (fptr, &status) )
    {
        printf("ERROR : cfits_read_header_map --> Error closing FITS file, error = %d\n",status);
        return(status);
    }


    return(status);
}
```

B.2.2 CFITS Read Map

```
/*
    This program is called cfits_write . It interacts with the CFITSIO library to read in a FITS file to an array of doubles
    Copyright (C) 2012 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

        INPUTS:
                char* filename : c string = name of FITS file to be read from
                array : map of doubles to be read from FITS file
                various metadata
        OUTPUTS:
                Reads a FITS file from disk

        NB : see FITS file format documentation to see what each parameter means (though AIPS may interpret it differently
            ...)
*/

#include "pmem_heads.hpp"

int cfits_read_map(const char* filename, double* tarr, int dim2 , double* cc_xarray, double* cc_yarray, double*
    cc_varray , int ncc)
{
/*
        INPUTS:
                char* tfilename : c string = name of FITS file to be read
                int dim2 : number of pixels to be read
                int ncc : number of clean components to be read (0 if no cc table expected/needed)
        OUTPUTS:
                tarr : 1D floating point array containing pixel values. ****** NB! This is in row major order, converted
                    Fortran code ****
                cc_xarray : 1D fp array containing x coords of clean components in degrees
                cc_yarray : 1D fp array containing y coords of clean components in degrees
                cc_varray : 1D fp array containing values of clean components in degrees
*/
         fitsfile *fptr;

        int status;
        char comment[FLEN_VALUE];
        float nullval=666.0;
        int int_null=0;
        double double_null=0;
        long fpixel=1;
        char cchdu[]="AIPS CC ";
        char fluxname[]="FLUX";
        char xname[]="DELTAX";
        char yname[]="DELTAY";
        int colnum;
        int i;


        status = 0;     // for error processing


        if ( fits_open_file(&fptr,filename, READONLY, &status) )      // open file and make sure it's open
        {
                printf("ERROR : cfits_read_map --> Error opening FITS file, error = %d\n",status);
                return(status);
        }


        if (fits_movabs_hdu(fptr,1,IMAGE_HDU,&status))      // move to main AIPS image hdu
        {
                printf("ERROR : cfits_read_map --> Error locating AIPS primary image extension, error = %d\n",status);
                return(status);
        }
        // read in main image data data

        fits_read_img(fptr, TDOUBLE, fpixel, dim2, &nullval, tarr, &int_null, &status);
        if(status!=0)
        {
                printf("ERROR : cfits_read_map --> Error reading map, error = %d\n",status);
        }

        if(ncc > 0)     // read in cc data if present/required
        {
                if (fits_movnam_hdu(fptr,BINARY_TBL,cchdu,0,&status))     // move to main AIPS image hdu
                {
                    printf("ERROR : cfits_read_map --> Error locating AIPS clean component extension, error = %d\
                        n",status);
                    return(status);
                }
                else
                {
                    fits_get_colnum(fptr,CASEINSEN,xname,&colnum,&status);
                    if(status!=0)
                    {
                        printf("ERROR : cfits_read_map --> Error locating CC x position information, error = %
                            d\n",status);
                    }
                    fits_read_col(fptr,TDOUBLE,colnum,1,1,ncc,&double_null,cc_xarray,&int_null,&status);
                    if(status!=0)
                    {
                        printf("ERROR : cfits_read_map --> Error reading CC x position information, error = %d
                            \n",status);
                    }

                    fits_get_colnum(fptr,CASEINSEN,yname,&colnum,&status);
                    if(status!=0)
                    {
                        printf("ERROR : cfits_read_map --> Error locating CC y position information, error = %
                            d\n",status);
                    }
                    fits_read_col(fptr,TDOUBLE,colnum,1,1,ncc,&double_null,cc_yarray,&int_null,&status);
                    if(status!=0)
                    {
                        printf("ERROR : cfits_read_map --> Error reading CC y position information, error = %d
                            \n",status);
                    }

                    fits_get_colnum(fptr,CASEINSEN,fluxname,&colnum,&status);
                    if(status!=0)
                    {
                        printf("ERROR : cfits_read_map --> Error locating CC flux position information, error =
                            %d\n",status);
                    }
                    fits_read_col(fptr,TDOUBLE,colnum,1,1,ncc,&double_null,cc_varray,&int_null,&status);
                    if(status!=0)
                    {
                        printf("ERROR : cfits_read_map --> Error reading CC flux position information, error =
                            %d\n",status);
                    }
                }
        }

        if ( fits_close_file (fptr, &status) )
        {
                printf("ERROR : cfits_read_map --> Error closing FITS file, error = %d\n",status);
                return(status);
        }

        return(status);
}
```

B.2.3 CFITS Write Map

```
/*
    This program is called cfits_write . It interacts with the CFITSIO library to write out a FITS file from an array of
         doubles
    Copyright (C) 2012 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

        INPUTS:
                char* filename : c string = name of FITS file to be created
                array : map of doubles to be converted to a FITS file
                various metadata (source and observation information, cellsize , frequency, beam (if any) etc.)
        OUTPUTS:
                writes a FITS file to disk

        NB : see FITS file format documentation to see what each parameter means (though AIPS may interpret it differently
              ...)
*/

#include "pmem_heads.hpp"


int cfits_write(const char* filename , double* array , int imsize , double cell , double ra , double dec , double*
      centre_shift , double* rotations , double freq , double freq_delta , int stokes , char* object , char* observer , char*
      telescope , double equinox , char* date_obs , char* history , double bmaj , double bmin , double bpa , int niter ,
      bool jy_per_beam)
{

         fitsfile *fptr; // pointer to fits file
        int status, i, j;
        long fpixel = 1, naxis = 4, nelements; // fpixel is the coordinate of the first pixel to be read
        long naxes[4] = { imsize, imsize , 1 , 1 };
        double temp;
        char comment[]="";
        char tstring[FLEN_VALUE];
        char fname[strlen(filename)+1];

        strcpy(fname,"!");
        strcat(fname,filename);



        status = 0;
        // status is an error variable

         fits_create_file (&fptr,fname, &status);

        // create new file

        // Create the primary array image (64-bit floating point pixels)
        fits_create_img(fptr, DOUBLE_IMG, naxis, naxes, &status);


        fits_update_key(fptr, TSTRING, "OBJECT", object , comment , &status);
        fits_update_key(fptr, TSTRING, "OBSERVER", observer , comment , &status); // write out information about the
              source
        fits_update_key(fptr, TSTRING, "TELESCOP", telescope , comment , &status);
        fits_update_key(fptr, TDOUBLE, "EQUINOX", &equinox , comment, &status);
        fits_update_key(fptr, TSTRING, "DATE-OBS", date_obs , comment , &status);
        fits_update_key(fptr, TDOUBLE, "OBSRA", &ra , comment , &status);
        fits_update_key(fptr, TDOUBLE, "OBSDEC", &dec , comment , &status);

        temp=1.0;
        fits_update_key(fptr, TDOUBLE, "BSCALE", &temp,comment, &status);

        temp=0.0;
```

```
fits_update_key(fptr, TDOUBLE, "BZERO", &temp,comment, &status);


if(jy_per_beam)
{
        sprintf(tstring,"JY/BEAM");
        fits_update_key(fptr, TSTRING, "BUNIT", tstring , comment , &status);
}
else
{
        sprintf(tstring,"JY/PIX");
        fits_update_key(fptr, TSTRING, "BUNIT", tstring , comment , &status);
}



sprintf(tstring,"RA---SIN");
fits_update_key(fptr, TSTRING, "CTYPE1", tstring , comment , &status); // write out information about the first axis
    , Right Ascention

fits_update_key(fptr, TDOUBLE, "CRVAL1", &ra , comment , &status);

temp = -cell;
fits_update_key(fptr, TDOUBLE, "CDELT1", &temp , comment , &status);

temp = centre_shift[0];
fits_update_key(fptr, TDOUBLE, "CRPIX1", &temp , comment , &status);

temp = rotations[0];
fits_update_key(fptr, TDOUBLE, "CROTA1", &temp , comment , &status);




sprintf(tstring,"DEC--SIN");
fits_update_key(fptr, TSTRING, "CTYPE2", tstring , comment , &status); // write out information about the second
    axis, Declination

fits_update_key(fptr, TDOUBLE, "CRVAL2", &dec , comment , &status);

fits_update_key(fptr, TDOUBLE, "CDELT2", &cell , comment , &status);

temp = centre_shift[1];
fits_update_key(fptr, TDOUBLE, "CRPIX2", &temp , comment , &status);

temp = rotations[1];
fits_update_key(fptr, TDOUBLE, "CROTA2", &temp , comment , &status);




sprintf(tstring,"FREQ");
fits_update_key(fptr, TSTRING, "CTYPE3", tstring , comment , &status); // write out information about the third
    axis, Frequency

fits_update_key(fptr, TDOUBLE, "CRVAL3", &freq , comment , &status);

fits_update_key(fptr, TDOUBLE, "CDELT3", &freq_delta , comment , &status);

temp=1.0;
fits_update_key(fptr, TDOUBLE, "CRPIX3", &temp , comment , &status);

temp=0.0;
fits_update_key(fptr, TDOUBLE, "CROTA3", &temp , comment , &status);



sprintf(tstring,"STOKES");
fits_update_key(fptr, TSTRING, "CTYPE4" , tstring , comment , &status); // write out information about the fourth
    axis, Stokes parameter

temp = double(stokes);
fits_update_key(fptr, TDOUBLE, "CRVAL4", &temp , comment , &status);

temp=1.0;
```

```
fits_update_key(fptr, TDOUBLE, "CDELT4", &temp , comment , &status);

fits_update_key(fptr, TDOUBLE, "CRPIX4", &temp , comment , &status);

temp=0.0;
fits_update_key(fptr, TDOUBLE, "CROTA4", &temp , comment , &status);


fits_write_history(fptr , history , &status);    // write history and date

fits_write_date(fptr , &status);



nelements = naxes[0] * naxes[1];
// number of pixels to write

// Write the array of double size floating point to the image
fits_write_img(fptr, TDOUBLE, fpixel, nelements, array, &status);

if(jy_per_beam) // write out beam information for AIPS if necessary. note AIPS CG --> AIPS CLEAN gaussian,
    which is not true in this case, but is used for compatability with AIPS
{
    char bmaj_name[] = "BMAJ";
    char bmin_name[] = "BMIN";
    char bpa_name[] = "BPA";
    char freq_name[] = "FREQUENCY";
    char freq_units[] = "HZ";
    char beam_units[] = "DEGREES";
    char data_type[] = "1D";
    char* beaminfo_names[4] = {freq_name , bmaj_name, bmin_name, bpa_name};
    char* beaminfo_units[4] = {freq_units , beam_units , beam_units , beam_units};
    char* beaminfo_datatype[4] = {data_type , data_type , data_type , data_type};
    char tbl_name[] = "AIPS CG ";

    sprintf(tstring,"COMMENT : WRITING OUT BEAM IN AIPS FASHION (NOT REALLY FROM AIPS)");
        // write beam as AIPS-style note
    fits_write_history(fptr , tstring , &status);

    // NB - do no use more than 8 places here. Make sure to use the same accuracy for BMAJ and BMIN
    sprintf(tstring,"AIPS   CLEAN BMAJ= %.8lf BMIN= %.8lf BPA= %.8lf" , bmaj , bmin , bpa );
    fits_write_history(fptr , tstring , &status);

    if(niter>0)
    {
        sprintf(tstring,"COMMENT : NITER BELOW NOT NECESSARILY FROM CLEAN"); // write
            beam as AIPS-style note
        fits_write_history(fptr , tstring , &status);


        sprintf(tstring,"AIPS   CLEAN NITER= %d PRODUCT=1",niter);
        fits_write_history(fptr , tstring , &status);
    }

    fits_create_tbl(fptr , BINARY_TBL , 1 , 4 , beaminfo_names , beaminfo_datatype , beaminfo_units ,
        tbl_name , &status); // make beam table
    if(status!=0)
    {
        printf("cfits_write_map --> Error creating beam information table for %s, error code %i\n",
            filename,status);
    }

    if(fits_movnam_hdu(fptr , BINARY_TBL , tbl_name , 0 , &status))
    {
        printf("cfits_write_map --> Error writing beam information to %s, error code %i\n",filename,
            status);
    }

    fits_write_col(fptr, TDOUBLE , 1 , 1 , 1 , 1 , &freq , &status);
    fits_write_col(fptr, TDOUBLE , 2 , 1 , 1 , 1 , &bmaj , &status);        // write out values (beam info in
        degrees)
    fits_write_col(fptr, TDOUBLE , 3 , 1 , 1 , 1 , &bmin , &status);
    fits_write_col(fptr, TDOUBLE , 4 , 1 , 1 , 1 , &bpa , &status);

}

fits_close_file(fptr, &status);
```

```
        fits_report_error(stderr, status);

        return(status);

        fits_close_file (fptr, &status);
        fits_report_error(stderr, status);

        return(status);
}
```

B.2.4 CFITS Overwrite UV Data

```
/*
    This program is called cfits_overwrite_uvdata. It overwrites visibility data in a FITS file using the FITSIO library.
    Copyright (C) 2012 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

*/

#include "uvfill_heads.hpp"

int cfits_overwrite_uvdata(char* filename, int nvis, double* u_array, double* v_array, double* tvis)
{
/*
        INPUTS:
                char* tfilename : c string = name of FITS file to be read
                int nvis : number of visibilities to be read
        OUTPUTS:
                u_array : nvis u coords, converted to physical units by fitsio
                v_array : nvis v coords, converted to physical units by fitsio
                tvis : (nvis*24) visibilities , in Jy
*/
        fitsfile *fptr;

        int status, i, j;
        int err;
        char extname[]="AIPS UV ";
        char comment[]="UVFILL";
        double d_null=0;
        int anynull;
        double temp;

        status = 0;     // for error processing
        err=0;

        if ( fits_open_file(&fptr,filename, READWRITE, &status) )   // open file and make sure it's open
        {
                printf("ERROR : cfits_overwrite_uvdata --> Error opening FITS file, error = %d\n",status);
                return(status);
        }

        if (fits_movnam_hdu(fptr,BINARY_TBL,extname,0,&status))  // move to main AIPS UV hdu
        {
                printf("ERROR : cfits_overwrite_uvdata --> Error locating AIPS UV binary extension, error = %d\n",
                        status);
                printf("ERROR : cfits_overwrite_uvdata --> Did you remember to use the AIPS FITAB task instead of
                        FITTP?\n");
                return(status);
        }
        // read in data

        fits_write_col(fptr, TDOUBLE, 1, 1, 1, nvis, u_array, &status);
        err+=status;
        if(err!=0)
        {
                printf("ERROR : cfits_overwrite_uvdata --> Error writing uarray, custom error = %d\n",err);
        }

        fits_write_col(fptr, TDOUBLE, 2, 1, 1, nvis, v_array, &status);
        err+=status;
        fits_write_col(fptr, TDOUBLE, 9, 1, 1, nvis*12.0, tvis, &status);
        err+=status;
        if(err!=0)
        {
                printf("ERROR : cfits_overwrite_uvdata --> Error writing keywords, custom error = %d\n",err);
        }



        if ( fits_close_file (fptr, &status) )
        {
                printf("ERROR : cfits_overwrite_uvdata --> Error closing FITS file, error = %d\n",status);
                return(status);
        }

        return(status);
}
```

B.2.5 CFITS Replace Antenna Information

```
/*
    This program is called cfits_replace ant info. It replaces antenna information in a FITS file using the FITSIO library.
    Copyright (C) 2012 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

*/

#include "uvfill_heads.hpp"

int cfits_replace_ant_info(char* filename, double* rdterm, double* ldterm)
{
/*
        INPUTS:
                char* tfilename : c string = name of FITS file to be read
                int nvis : number of visibilities to be read
        OUTPUTS:
                u_array : nvis u coords, converted to physical units by fitsio
                v_array : nvis v coords, converted to physical units by fitsio
                tvis : (nvis*24) visibilities , in Jy
*/
         fitsfile *fptr;

        int status, i, j;
        char extname[]="AIPS AN ";
//      char comment[FLEN_VALUE];
        char comment[]="UVFILL";
        char poltype[]="VLBI  ";
        char rdtermname[]="POLCALA"; // assumes column a is the r term and column b is the l term
        char ldtermname[]="POLCALB";
        double d_null=0;
        int colnum;
        double temp;

        status = 0;      // for error processing

        if ( fits_open_file(&fptr,filename, READWRITE, &status) )   // open file and make sure it's open
        {
                printf("ERROR : cfits_replace_ant_info --> Error opening FITS file, error = %d\n",status);
                return(status);
        }

        if (fits_movnam_hdu(fptr,BINARY_TBL,extname,0,&status))  // move to main AIPS UV hdu
        {
                printf("ERROR : cfits_replace_ant_info --> Error locating AIPS UV binary extension, error = %d\n",
                      status);
                printf("ERROR : cfits_replace_ant_info --> Did you remember to use the AIPS FITAB task instead of
                      FITTP?\n");
                return(status);
        }
        // write out d-term data to correct columns

        fits_get_colnum(fptr,CASEINSEN,rdtermname,&colnum,&status);
        if(status!=0)
        {
                printf("ERROR : cfits_replace_ant_info --> Error finding r dterm column to write to, error = %d\n",
                      status);
        }


        fits_write_col(fptr, TDOUBLE, colnum, 1, 1, 10*2, rdterm, &status);
        if(status!=0)
        {
                printf("ERROR : cfits_replace_ant_info --> Error writing keywords, error = %d\n",status);
        }

        fits_get_colnum(fptr,CASEINSEN,ldtermname,&colnum,&status);
        if(status!=0)
        {
                printf("ERROR : cfits_replace_ant_info --> Error finding l dterm column to write to, error = %d\n",
                      status);
        }

        fits_write_col(fptr, TDOUBLE, colnum, 1, 1, 10*2, ldterm, &status);
        if(status!=0)
        {
                printf("ERROR : cfits_replace_ant_info --> Error writing new antenna information, error = %d\n",status)
                      ;
        }

        fits_update_key(fptr, TSTRING, "POLTYPE", &poltype,comment, &status);
        if(status!=0)
        {
                printf("ERROR : cfits_replace_ant_info --> Error writing polarisation type to header file, error = %d\n",
                      status);
        }



        if (  fits_close_file (fptr, &status) )
        {
                printf("ERROR : cfits_replace_ant_info --> Error closing FITS file, error = %d\n",status);
                return(status);
        }

        return(status);
}
```

B.3 UVFILL Code

The following section contains the C++ source code for the UVFILL program used to simulate observations by replacing the UV visibilities corresponding to a real observation with a simulated set generated from a CSV model. The code also allows thermal noise to be added to the resulting visibilities and can also simulate errors in the D-term calibration.

B.3.1 UVFILL Headers

```
/*
    This is a header file for functions and libraries releating to uvfill2 .
    Copyright (C) 2012 Colm Coughlan

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

*/

#include <iostream>
#include <fstream>
#include <string>
#include <string.h>
#include <cmath>
#include <stdlib.h>
#include <sstream>
#include <fftw3.h>
#include <fitsio.h>




extern "C"
{
        #include <gsl/gsl_rng.h>
        #include <gsl/gsl_randist.h>

        int cfits_write(const char* filename , double* array , int imsize , double cell , double ra , double dec , double*
              centre_shift , double* rotations , double freq , double freq_delta , int stokes , char* object , char* observer
              , char* telescope , double equinox , char* date_obs , char* history , double bmaj , double bmin , double
              bpa , int niter , bool jy_per_beam);
        int cfits_read_header(char* filename, double* ra, double* dec, char* key_string, double* freq, int* nvis);
        int cfits_read_data(char* filename, int nvis, double* u_array, double* v_array, double* tvis);
        int cfits_overwrite_uvdata(char* filename, int nvis, double* u_array, double* v_array, double* tvis);
        int cfits_replace_ant_info(char* filename, double* rdterm, double* ldterm);
}

// g++ -o uvfill uvfill .cpp cfits_replace_ant_info.c cfits_read_data.c cfits_overwrite_uvdata.c cfits_read_header.c - lcfitsio
        -fopenmp -lgsl -lblas -O3
// g++ -o uvfill2 uvfill2 .cpp cfits_replace_ant_info.c cfits_read_data.c cfits_overwrite_uvdata.c cfits_read_header.c
        cfits_write .cpp - lcfitsio -fopenmp -lgsl -lblas -O3
```

B.3.2 UVFILL Source Code

```
/*
    This program is called uvfill2 . It creates simulated UV data corresponding to a given CSV model of an AGN.
    It writes results using an existing UV file as a basis .
    Copyright (C) 2012 Colm Coughlan

    This program is free software : you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful ,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

*/

#include "uvfill_heads.hpp"
        using namespace std;

class fitsinfo
{
        private:

        double ra_val;
        double dec_val;
        int nvis_val;
        double freq_val;
        string name_val;

        double cellsize;
        int imsize;

        public:

         fitsinfo ()        // constructor defaults everything to zero
        {
                ra_val=0.0;
                dec_val=0.0;
                nvis_val=0;
                freq_val=0.0;
        }

        ~ fitsinfo (){}    // destructor

        double* set_ra()
        {
                return(&ra_val);
        }
        double* set_dec()
        {
                return(&dec_val);
        }
        int* set_nvis()
        {
                return(&nvis_val);
        }
        double* set_freq()
        {
                return(&freq_val);
        }
        int set_name(string temp)
        {
                name_val=temp;
                return(0);
        }

        double freq()
        {
                return(freq_val);
        }
        double ra()
        {
                return(ra_val);
        }
        double dec()
        {
                return(dec_val);
        }
        int nvis()
        {
                return(nvis_val);
        }
        string name()
        {
                return(name_val);
        }

        int set_image_info(double cell, int im)
        {
                cellsize =cell;
                imsize=im;

                return(0);
        }

        double give_cellsize()
        {
                return(cellsize);
        }

        int give_imsize()
        {
                return(imsize);
        }
};


class rawuv
{
        private:

        double* u;
        double* v;
        double* weight;

        fftw_complex* i;
        fftw_complex* rl;
        fftw_complex* lr;

        double* fits_uv_data;


        int size;
        int blocksize;

        double umax;
        double vmax;
        double maxweight;
        double sum_weights;

        public:

        rawuv(int n)
        {
                size=n;
                blocksize=n*12;
                u=new double[n];
                v=new double[n];
                i=new fftw_complex[n];
                rl=new fftw_complex[n];
                lr=new fftw_complex[n];
                weight=new double[n];
                fits_uv_data=new double[blocksize];
        }

        ~rawuv()
        {
                delete[] u;
                delete[] v;
                delete[] i;
                delete[] rl;
                delete[] lr;
                delete[] weight;
                delete[] fits_uv_data;
        }

        int filluv (double* uvals, double* vvals, double* tvis, double freq)
        {
                double temp;

                umax=0.0;
                vmax=0.0;
                sum_weights=0.0;
                maxweight=0.0;
                for(int ctr=0;ctr<size;ctr++)
                {


//      unweighted averaging of IF1 and IF2 occuring
//      assumed order in line : u,v,<6 unneeded random pars>,rr1,ll1,rl1,lr1,rr2,ll2,rl2,lr2
//      rr1 --> real, imag, weight. => 3*8 = 24 per entry
//      assumes the weighting remains the same for each individual visibility



                        u[ctr]=uvals[ctr]*freq;
                        v[ctr]=vvals[ctr]*freq;


                        temp=fabs(u[ctr]);
                        if(temp>umax)
                        {
                                umax=temp;
                        }
                        temp=fabs(v[ctr]);
                        if(temp>vmax)
                        {
                                vmax=temp;
                        }



                        i[ctr][0]=0.5*(tvis[(ctr*12)]+tvis[(ctr*12)+3]);
                        i[ctr][1]=0.5*(tvis[(ctr*12)+1]+tvis[(ctr*12)+4]);
                        rl[ctr][0]=tvis[(ctr*12)+6];
                        rl[ctr][1]=tvis[(ctr*12)+7];
                        lr[ctr][0]=tvis[(ctr*12)+9];
                        lr[ctr][1]=tvis[(ctr*12)+10];
                        weight[ctr]=tvis[(ctr*12)+2];

                        temp=weight[ctr];
                        if(temp>maxweight)
                        {
                                maxweight=temp;
                        }
                        sum_weights+=weight[ctr];
                }
                cout<<"maxweight = "<<maxweight<<endl;



                for(int i=0;i<blocksize;i++)
                {
                        fits_uv_data[i]=tvis[i];
                }
                return(size);
        }

        double* update_vis_block()
        {
                for(int ctr=0;ctr<size;ctr++)
                {
                        fits_uv_data[(ctr*12)]=i[ctr][0];               // ll, rr, rl, lr
                        fits_uv_data[(ctr*12)+3]=i[ctr][0];
                        fits_uv_data[(ctr*12)+6]=rl[ctr][0];
                        fits_uv_data[(ctr*12)+9]=lr[ctr][0];

                        fits_uv_data[(ctr*12)+1]=i[ctr][1];
                        fits_uv_data[(ctr*12)+4]=i[ctr][1];
                        fits_uv_data[(ctr*12)+7]=rl[ctr][1];
                        fits_uv_data[(ctr*12)+10]=lr[ctr][1];
                }

                return(fits_uv_data);
        }

        double* give_vis_block()
        {
                return(fits_uv_data);
        }

        int change_vis(double *imap, double *qmap, double* umap, int imsize, double cellsize)
        {
                int ctr,j,k,n;
                double temp1,temp2;
                int beg,end;


                double* grid_image;

                grid_image=new double[imsize];

                k=0;
                beg=-imsize/2;
                end=imsize/2;
                if((imsize%2)==0)
                {
                        beg++;
                }
                end++;
                for(ctr=beg;ctr<end;ctr++)     // create x and y axes for maps (in radiens) and gridded uv data (in lambda)
                {
                        grid_image[k++]=ctr*cellsize;
                }

                #pragma omp parallel for private(n,ctr,j,temp1,temp2)
                for(k=0;k<size;k++)
                {
                        i[k][0]=0.0;
                        i[k][1]=0.0;
                        rl[k][0]=0.0;
                        rl[k][1]=0.0;
                        lr[k][0]=0.0;
                        lr[k][1]=0.0;
                        n=0;
                        for(j=0;j<imsize;j++)
                        {
                                for(ctr=0;ctr<imsize;ctr++)
                                {
                                        temp1=-2.0*M_PI*((u[k]*grid_image[ctr])-(v[k]*grid_image[j]));
                                        temp2=sin(temp1);
                                        temp1=cos(temp1);
                                        i[k][0]+=imap[n]*temp1; // assuming imap is real
                                        i[k][1]+=imap[n]*temp2;
                                        rl[k][0]+=qmap[n]*temp1-umap[n]*temp2;
                                        rl[k][1]+=umap[n]*temp1+qmap[n]*temp2;
                                        lr[k][0]+=qmap[n]*temp1+umap[n]*temp2;
                                        lr[k][1]+=qmap[n]*temp2-umap[n]*temp1;
                                        n++;
                                }
                        }
                }

                delete[] grid_image;

                return(0);
        }

        double giveu(int n)
        {
                return(u[n]);
```

```
}

double givev(int n)
{
        return(v[n]);
}

double* giveu_array()
{
        return(u);
}

double* givev_array()
{
        return(v);
}

double giveumax()
{
        return(umax);
}

double givevmax()
{
        return(vmax);
}

double givei(int n,int comp)
{
        return(i[n][comp]);
}

double giverl(int n,int comp)
{
        return(rl[n][comp]);
}
double givelr(int n,int comp)
{
        return(lr[n][comp]);
}

int set_i(int n,int comp,double val)
{
        i[n][comp]=val;
        return(0);
}

int set_rl(int n,int comp,double val)
{
        rl[n][comp]=val;
        return(0);
}

int set_lr(int n,int comp,double val)
{
        lr[n][comp]=val;
        return(0);
}

double giveweight(int n)
{
        return(weight[n]);
}

double give_sum_weights()
{
        return(sum_weights);
}

double give_max_weights()
{
        return(sum_weights);
}

int givesize()
{
        return(size);
        }
};

int get_fitsi(string filename, fitsinfo& fi);
int get_vis(string filename,rawuv& visarr,fitsinfo& fi);          // put uv visibilities into rawuv
int csv_read(string filename, int imsize, double* matrix);
int rewrite_vis(string filename,rawuv& visarr,fitsinfo& fi);
string int_to_str(int i);
int rewrite_ant_data(string filename,double* rdterm, double* ldterm,fitsinfo& fi);
int write_csv(string filename, double* array, int imsize);
int row_major_to_col_major(double *mat, int imsize);




int main()
{
         fitsinfo fi;

        string uvfits;
        string modelname;


        int err;
        int i,j,k;
        int nvis;
        int question;
        int nmaps;


        double temp;
        double stddev;
        double stddev_dterm;

        int imsize;
        double cellsize;

        double *imap;
        double *qmap;
        double *umap;
        fftw_complex* original_i;
        fftw_complex* original_rl;
        fftw_complex* original_lr;

        double* rdterm;
        double* ldterm;

        ifstream fin;
        ofstream fout;

        char outdata[]="simmodel";



        cout<<"Please enter the imsize of the model map (pixels)"<<endl;
        cin>>imsize;
        cout<<"Please enter the cellsize of the model map (as)"<<endl;
        cin>>cellsize;

        imap=new double[imsize*imsize];
        qmap=new double[imsize*imsize];
        umap=new double[imsize*imsize];
         cellsize *=(M_PI)/(180.0*3600);



        cout<<"Please enter the name of the I model map"<<endl;
        cin>>modelname;
        err=csv_read(modelname,imsize,imap);
        if(err==0)
        {
                cout<<"Imap read successful"<<endl;
        }
        else
        {
                cout<<"Problem reading model map."<<endl;
                return(1);
        }

        cout<<"Please enter the name of the Q model map"<<endl;
        cin>>modelname;
        err=csv_read(modelname,imsize,qmap);
        if(err==0)
        {
                cout<<"Qmap read successful"<<endl;
        }
        else
        {
                cout<<"Problem reading model map."<<endl;
                return(1);
        }

        cout<<"Please enter the name of the U model map"<<endl;
        cin>>modelname;
        err=csv_read(modelname,imsize,umap);
        if(err==0)
        {
                cout<<"Umap read successful"<<endl;
        }
        else
        {
                cout<<"Problem reading model map."<<endl;
                return(1);
        }


        cout<<"Please enter the name of the FITS file with the UV information"<<endl;
        cout<<"This FITS file is assumed to have 1 IF."<<endl;
        cin>>uvfits;



        cout<<"Attempting to open "<<uvfits<<endl;
        err=get_fitsi( uvfits , fi );          //get fits information from file
        if(err!=0)
        {
                cout<<"Error attempting to open fits file."<<endl;
                cout<<"Program closing."<<endl;
                return(1);
        }
        nvis=fi.nvis();

        rawuv visarr(nvis);

        get_vis( uvfits , visarr , fi );
        cout<<"Raw UV data and visibility read complete."<<endl;
        cout<<"\t "<<nvis<<" visibilites read."<<endl;
        cout<<"\t First U coord = "<<visarr.giveu(0)/(1000000.0)<<" megalambda."<<endl;
        cout<<"\t First V coord = "<<visarr.givev(0)/(1000000.0)<<" megalambda."<<endl;
        cout<<"\t First I  visibility  = "<<visarr.givei(0,0)<<" "<<visarr.givei(0,1)<<" i."<<endl;
        cout<<"\t First RL visibility = "<<visarr.giverl(0,0)<<" "<<visarr.giverl(0,1)<<" i."<<endl;
        cout<<"\t Sum of FITS weights = "<<visarr.give_sum_weights()<<endl;


        err = cfits_write( "model_imap.fits" , imap , imsize ,  cellsize *(180.0/M_PI) , fi.ra()  ,  fi .dec()  , &temp , &stddev
               , fi.freq()  ,  0  ,  1  ,  outdata , outdata , outdata , 0  ,  outdata , outdata , 0  ,  0  ,  0  ,  0  ,  false);
        err = cfits_write( "model_qmap.fits" , qmap , imsize ,  cellsize *(180.0/M_PI) , fi.ra()  ,  fi .dec()  , &temp , &stddev
               , fi.freq()  ,  0  ,  2  ,  outdata , outdata , outdata , 0  ,  outdata , outdata , 0  ,  0  ,  0  ,  0  ,  false);
        err = cfits_write( "model_umap.fits" , umap , imsize ,  cellsize *(180.0/M_PI) , fi.ra()  ,  fi .dec()  , &temp , &stddev
               , fi.freq()  ,  0  ,  3  ,  outdata , outdata , outdata , 0  ,  outdata , outdata , 0  ,  0  ,  0  ,  0  ,  false);

//      write_csv( "model_imap.csv", imap , imsize);

//      err = row_major_to_col_major( imap, imsize);
//      err = row_major_to_col_major( qmap, imsize);
//      err = row_major_to_col_major( umap, imsize);


        visarr.change_vis(imap,qmap,umap,imsize,cellsize);

        cout<<"DFT complete - writing results to fits file"<<endl;

        fin.open(uvfits.c_str(),  fstream::binary);
        fout.open("output_no_noise.fits", fstream::trunc|fstream::binary);
```

```
fout<<fin.rdbuf();
fout.close();


rewrite_vis("output_no_noise.fits",visarr, fi);

cout<<"Writing out model maps as FITS images."<<endl;

temp=imsize/2;
stddev=(imsize/2)+1;

cout<<"Would you like to add normally distributed noise to the DFT? (1 = yes, 0 = no)"<<endl;
cin>>question;

if(question==1)
{
        const gsl_rng_type * T;
        gsl_rng * r;
        gsl_rng_env_setup();
        unsigned long int seed;

        T = gsl_rng_default;
        r = gsl_rng_alloc (T);

        seed = time (NULL) * getpid();
        gsl_rng_set ( r, seed);


        original_i=new fftw_complex[nvis];
        original_rl=new fftw_complex[nvis];
        original_lr=new fftw_complex[nvis];

        rdterm=new double[2*10];
        ldterm=new double[2*10];


        for(i=0;i<nvis;i++)
        {
                original_i[i][0]= visarr.givei(i,0);
                original_i[i][1]= visarr.givei(i,1);
                original_rl[i][0]= visarr.giverl(i,0);
                original_rl[i][1]= visarr.giverl(i,1);
                original_lr[i][0]= visarr.givelr(i,0);
                original_lr[i][1]= visarr.givelr(i,1);
        }

        for(i=0;i<20;i++)       // format is (real, imag)*10
        {
                rdterm[i]=0.0;
                ldterm[i]=0.0;
        }

        cout<<"How many files would you like to make?"<<endl;
        cin>>nmaps;
        cout<<"What standard deviation would you like the noise distributed about zero to have?"<<endl;
        cin>>stddev;
        stddev/=sqrt(2.0);
        cout<<"Sample Noise values."<<endl;
        for(i=0;i<10;i++)
        {
                cout<<"\t "<<gsl_ran_gaussian(r,stddev)<<endl;
        }

        cout<<"What D term noise would you like to add to the antenna table?"<<endl;
        cin>>stddev_dterm;
        stddev_dterm/=sqrt(2.0);

        for(i=0;i<nmaps;i++)
        {
                fin.seekg(0, ios::beg);
                fin.clear();

                modelname.assign("output_");
                modelname.append(int_to_str(i+1));
                modelname.append(".fits");
                fout.open(modelname.c_str(), fstream::trunc|fstream::binary);
                fout<<fin.rdbuf();
```

```
                        fout.close();

                        for(j=0;j<nvis;j++)
                        {
                                visarr.set_i(j,0, original_i[j][0]+gsl_ran_gaussian(r,stddev));
                                visarr.set_i(j,1, original_i[j][1]+gsl_ran_gaussian(r,stddev));
                                visarr.set_rl(j,0, original_rl[j][0]+gsl_ran_gaussian(r,stddev));
                                visarr.set_rl(j,1, original_rl[j][1]+gsl_ran_gaussian(r,stddev));
                                visarr.set_lr(j,0, original_lr[j][0]+gsl_ran_gaussian(r,stddev));
                                visarr.set_lr(j,1, original_lr[j][1]+gsl_ran_gaussian(r,stddev));
                        }

                        rewrite_vis(modelname.c_str(),visarr,fi);

                        for(j=0;j<20;j++)        // format is (real, imag)*10
                        {
                                rdterm[j]=gsl_ran_gaussian(r,stddev_dterm);
                                ldterm[j]=gsl_ran_gaussian(r,stddev_dterm);
                        }

                        rewrite_ant_data(modelname.c_str(),rdterm,ldterm,fi);
                }

                delete[] original_i;
                delete[] original_rl;
                delete[] original_lr;
                delete[] rdterm;
                delete[] ldterm;
                gsl_rng_free (r);
        }
        else
        {
//              visarr.make_vis_map(imap,imsize,cellsize,fi);
                cout<<"Closing Program"<<endl;
        }

        delete[] imap;
        delete[] qmap;
        delete[] umap;
        fin.close();
/*
        fi.set_image_info(cellsize,imsize);
        write_map("qmap.fits",pmap,0,fi);
*/
        return(0);
}

string int_to_str(int i)
{
        stringstream ss;
        string str;

        ss<<i;

        str=ss.str();
        ss.flush();

        return(str);
}


int get_fitsi(string filename, fitsinfo & fi)        // fill up fi with header information
{

        char* cfilename;         // c and fitsio are a bit fussy about exactly what type of character array / string they
              recieve
        char tstring[FLEN_VALUE];

        int err;

        cfilename=new char[filename.length()];
        strcpy(cfilename,filename.c_str());

        err=cfits_read_header(cfilename,fi.set_ra(), fi.set_dec(),tstring, fi.set_freq(), fi.set_nvis());   // write header
              information into fi

        fi.set_name(tstring);
        delete[] cfilename;

        return(err);
}

int get_vis(string filename,rawuv& visarr,fitsinfo& fi) // put uv visibilities into rawuv
{
        int i;
        int nvis;
        double freq;
        char* cfilename;        // c and fitsio are a bit fussy about exactly what type of character array / string they
              recieve
        double* u_array;
        double* v_array;
        double* tvis;

        int err;

        nvis=fi.nvis();
        freq=fi.freq();

        cfilename=new char[filename.length()];
        strcpy(cfilename,filename.c_str());

        u_array=new double[nvis];    // hold u coordinates
        v_array=new double[nvis];    // hold v coordinates
        tvis=new double[nvis*12];    // hold all visibilities

        err=cfits_read_data(cfilename,nvis,u_array,v_array,tvis);
        if(err!=0)
        {
                cout<<"Error reading data from fits file."<<endl;
                return(1);
        }



        err=visarr.filluv(u_array,v_array,tvis,freq);
        if(err!=nvis)
        {
                cout<<"Error loading fits information into memory."<<endl;
                return(1);
        }

        delete[] cfilename;
        delete[] tvis;
        delete[] u_array;
        delete[] v_array;

        return(0);
}

int rewrite_vis(string filename,rawuv& visarr,fitsinfo& fi)     // put uv visibilities into rawuv
{
        int i;
        int nvis;
        double freq;
        char* cfilename;        // c and fitsio are a bit fussy about exactly what type of character array / string they
              recieve

        int err;

        nvis=fi.nvis();
        freq=fi.freq();

        double* ucorr;
        double* vcorr;

        ucorr=new double[nvis];
        vcorr=new double[nvis];

        for(i=0;i<nvis;i++)
        {
                ucorr[i]=visarr.giveu(i)/freq;
                vcorr[i]=visarr.givev(i)/freq;
        }
        cfilename=new char[filename.length()];
        strcpy(cfilename,filename.c_str());


        err=cfits_overwrite_uvdata(cfilename,nvis,ucorr,vcorr,visarr.update_vis_block());
        if(err!=0)
        {
                cout<<"Error rewriting data in fits file."<<endl;
                return(1);
        }

        delete[] cfilename;
        delete[] ucorr;
        delete[] vcorr;

        return(0);
}

int csv_read(string filename, int imsize, double* matrix)
{
        fstream fin;

        string str;
        int i,j;
        size_t p1,p2;

        fin.open(filename.c_str(), ios::in);

        if(fin.is_open())
        {
                i=imsize-1;
                while(getline(fin,str))
                {
                        p1=0;
                        for(j=0;j<imsize;j++)
                        {
                                p2=str.find_first_of(",",p1);
                                matrix[i*imsize+j]=atof((str.substr(p1,p2-p1)).c_str());          // left to right, from the
                                        end of the file
                                p1=p2+1;                                                          // this is the Octave way...
                        }
                        i--;
                }
        }
        else
        {
                return(1);
        }


        if(i!=-1)
        {
                cout<<"Problem reading file. Look at dimensions."<<endl;
                return(1);
        }

        fin.close();
        return(0);
}



int rewrite_ant_data(string filename,double* rdterm, double* ldterm,fitsinfo& fi)
{
        char* cfilename;          // c and fitsio are a bit fussy about exactly what type of character array / string they
                recieve

        int err;


        cfilename=new char[filename.length()];
        strcpy(cfilename,filename.c_str());


        err=cfits_replace_ant_info(cfilename,rdterm,ldterm);
        if(err!=0)
        {
                cout<<"Error rewriting antenna data in fits file."<<endl;
                return(1);
        }

        delete[] cfilename;

        return(0);
}

int write_csv(string filename, double* array, int imsize)
{
        int i,j,k;
        int err;
        ofstream fout;
        char* cfilename;

        cfilename=new char[filename.length()+1];                // get name into C format
        strcpy(cfilename,filename.c_str());

        fout.open(cfilename,ios::out);
        err=fout.is_open();

        k=0;
        for(i=0;i<imsize;i++)
        {
                fout<<array[k];
                k++;
                for(j=1;j<imsize;j++)
                {
                        fout<<","<<array[k];
                        k++;
                }
                fout<<endl;
        }
        fout.close();

        delete[] cfilename;
        return(err);
}

int row_major_to_col_major(double *mat, int imsize)
{
        double *buffer;
        int i,j;
        int imsize2 = imsize * imsize;

        buffer = new double[imsize2];

        for(i=0;i<imsize2;i++)
        {
                buffer[i] = mat[i];
        }

        #pragma omp parallel for collapse(2)
        for(i=0;i<imsize;i++)
        {
                for(j=0;j<imsize;j++)
                {
                        mat[(i*imsize)+j] = buffer[(j*imsize)+i];
                }
        }

        delete[] buffer;
        return(0);
}
```