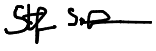
SDP Memo 080: On Scalable Faraday Tomography/Rotation Measure Synthesis for the Square Kilometre Array

Document Number ..... SDP Memo 080
Document Type ..... MEMO
Revision 1
Author ..... Stefano Salvini and Jamie S Farnes
Release Date ..... September 27, 2018
Document Classification ..... Unrestricted
Status ..... Advanced Draft

Lead Author	Designation	Affiliation
Stefano Salvini		Dept. Engineering Sciences, University of Oxford
Signature & Date:  Stefano Salvini (Sep 27, 2018)		

Revision	Date of issue	Prepared by	Comments
1	2018-07-03	Stefano Salvini	Draft

SDP Memo Disclaimer

The SDP memos are designed to allow the quick recording of investigations and research done by members of the SDP. They are also designed to raise questions about parts of the SDP design or SDP process. The contents of a memo may be the opinion of the author, not the whole of the SDP.

Table of Contents

Summary

Faraday Tomography is a critical algorithm for the Square Kilometre Array's (SKA) cosmic magnetism key science goals. However, the current implementations of the algorithm are typically serial and thus not scalable. The unprecedented SKA data rates (20,000 x 20,000 pixel images for 65,000 spectral channels) make existing approaches unfeasible. In this paper, we therefore investigate the parallelism and scalability of the modern spectropolarimetric Faraday Tomography algorithm for the SKA era. This would improve the computational efficiency on the SKA Science Data Processor (SDP) or on SKA Regional Centres. We find that contemporary pixel-by-pixel Faraday Tomography implementations, such as those now used by the SKA predecessors, would require years to process a single SKA dataset. We propose a restructuring of the algorithm, which would reduce this estimate to the order of days on a single node. We also propose a novel parallel approach that provides linear scalability with the number of compute nodes, up to the large systems (500+ nodes) required by SKA. The method accounts for both computational and data transfer costs. The time required to process an SKA-scale dataset would be reduced to the order of an hour, thereby making the study of cosmic magnetism feasible. This parallel approach can also be useful to other SKA computational tasks, such as data visualisation. The technique has immediate applications to forthcoming radio surveys, such as those with ASKAP, LOFAR, and the VLA.

1 Introduction

One of the flagship SKA key science goals is to understand better the cosmic magnetic fields which are ubiquitous throughout our Universe. Magnetic fields exist on all spatial scales, from planets, through the interstellar medium, to the largest known structures in the cosmic web. Little is still known about the definitive role of these magnetic fields in the evolution of our Cosmos. Advances in cosmic magnetism studies require the analysis of polarised radio emission, where the four Stokes parameters I, Q, U, and V define the polarised properties of a radio wave. Linear polarisation, in particular, is defined through Q and U, and enables the measurement of the Faraday effect; a wavelength-squared dependent rotation of the polarisation vector $Q + iU$ which leads to estimating a quantity called *Rotation Measure* or *Faraday Depth*, proportional to the magnetic field strength.

The primary technique in cosmic magnetic field studies is *Faraday Tomography* also sometimes called *Rotation Measure (RM) Synthesis*. This algorithm was first discussed by Burn (1966), and then applied to modern radio data by Brentjens and de Bruyn (2005). A comparison of similar algorithms has continued in more recent works, such as Sun et al. (2015). At its core, Faraday Tomography is a Fourier technique combined with a deconvolution algorithm which allows for the conversion of radio astronomy sky images into *RM datacubes*. The production of these RM cubes is a key requirement for the SKA cosmic magnetism science case, and for numerous pathfinder/precursor surveys: the Multifrequency Snapshot Sky Survey (MSSS) with the LOw Frequency ARray (LOFAR); the POlarisation Sky Survey of the Universe Magnetism (POSSUM) survey with the Australia Square Kilometre Array Pathfinder (ASKAP); the Very Large Array Sky Survey (VLASS) with the Karl G. Jansky Very Large Array (VLA). The Faraday Tomography technique is computationally expensive, of a similar cost to what the de-dispersion of transients require, albeit observations have typically fixed duration. The sheer scale of SKA, which will generate approximately $20,000 \times 20,000$ pixel images of Stokes Q and U, in $65,000$ independent frequency channels, makes Faraday Tomography difficult to achieve.

Scaling current computational schemes to an instrument of the SKA size is neither straightforward nor easy. Parallelism is necessary to satisfy the requirements of both a datasets extent and the computational costs, but the expected distribution of data across *processing units* (be them *islands*, *nodes*, etc.) makes it challenging to implement algorithms that can cut across the frequency–space (as a function of channel).

We investigate here the parallelisation and scalability of modern polarisation algorithms for the SKA era to reduce the computational costs on the SKA Science Data Processor (SDP) or on SKA Regional Centres. Additionally, the numerical algorithm run on each processing unit has been restructured, thus further enhancing performance. Prototyping a new parallelised and scalable algorithm for SKA could be a useful demonstrator of different plausible Execution Frameworks for SDP. In this memo, we consider the application of a novel parallelisation and algorithmic approach to Faraday Tomography and analyse its benefits. Our approaches could be applied to a substantial number of key SKA algorithms, on a case–by–case basis.

The aims of this memo are to show that:

1. Current computational schemes for Faraday Tomography, as implemented within many current procedures and codes, are inadequate and cannot scale to SKA.

2. Highly parallel computational schemes for Faraday Tomography can be envisaged, which also account for the re-distribution of data. These cases are examined.

3. Whenever possible, re-using existing algorithmic and software techniques can minimise the extra effort required.

4. The design and implementation of alternative algorithms can provide the same scientific results with a substantial increase in computational performance.

The document is structured as follows: Section 2 describes the computational method underlying Faraday Tomography, then it describes and discusses in details possible approaches to the algorithm for the SKA-era. Section 3 provides worked examples for major radio facilities, and, after briefly talking about possible future work and some considerations arising in Section 4, finally, we present our conclusions in Section 5.

2 Computational Method

The notation used is described in Table 1.

Symbol	Description
N	Width of the sky image (in pixels)
$N_{\rm pix}$	Total number of pixels: $N_{\rm pix} = N^2$
F	Number of observing frequencies (channels)
P	Number of processing units (islands, nodes, etc.)
R	Number of different Rotation Measures to be used
w	Word length in bytes, e.g. for single precision complex ($w = 8$)
B_p	Input/Output data bandwidth per processing unit (bytes/second)
B_T	Total data bandwidth for the whole system (bytes/second)
B_M	Maximum achievable bandwidth: $B_M = \min(B_P \cdot P, B_T)$
Z	Final RM-cube/3D image matrix for each Rotation Measure (Size: $N_{\rm pix} \times R$)
Q, U	Data cubes, composed of sky images at each frequency, for the Stokes parameters Q and U (both real arrays of size $N \times N \times F$)
Ξ	Complex coefficients for each Rotation Measure and frequency (Size: $F \times R$)
c_ξ	Number of flops required to compute each RM coefficient
s_ξ	Computational speed (flops/sec) achievable during the computation of the matrix of the RM coefficients (Ξ)
s_Z	Computational speed (flops/sec) achievable during the computation of the RM datacube (Z)

Table 1: Symbols used in the document

For each image pixel, we have to compute a weighted sum of the multi-frequency data for a specific Rotation Measure:

$$Z_{i,j}(\varphi_l) = \sum_{k=1}^{F} \xi_k(\varphi_l) \cdot (Q_{i,j,k} + i U_{i,j,k}) \qquad \text{for} \quad 1 \le l \le R,\ 1 \le i,j \le N \tag{1}$$

where i,j are the pixel coordinates, k is the frequency (channel) index, φ_l the l-th Rotation Measure, and $Q_{i,j,k}$ and $U_{i,j,k}$ are the surface brightness in Jy/beam of the i, j pixel for the k-th frequency from the Stokes parameters Q and U, respectively.

The complex coefficients $\xi_k(\varphi_l)$ are given by:

$$\xi_k(\varphi_l) = w_k \cdot \exp\left[2\, i\, \varphi_l (\lambda_k^2 - \lambda_0^2)\right] \tag{2}$$

which depends only on frequency and RM and not on the pixel coordinates. The weights w_k depend only on the frequency; in particular on the r.m.s. noise of the image as either σ^{-1} or

σ^{-2}, and which we assume to have been chosen so as to be spatially non-varying within a given image and to be appropriately scaled/normalised.

For maximum efficiency, the computation can be carried out in two stages:

1. Compute the coefficients $\xi_k(\varphi_l)$ for all k and l. These are placed in the (complex) matrix Ξ so that $\Xi_{k,l} = \xi_k(\varphi_l)$.

2. Compute the matrix-matrix product

$$\bar{Z} = (\bar{Q} + i\bar{U}) \cdot \Xi \tag{3}$$

The use of a matrix-matrix product, instead of explicitly applying eq.(1) to each image pixel, is discussed later in the document. The overbars denote that the first two indices of the corresponding matrices Z, Q, and U have been, only notionally, compressed into one index, e.g. $(i, j) \rightarrow i + N(j-1)$, as the matrices Z, Q, and U are presented to matrix-matrix product functions as matrices of size $N_{\text{pix}} \times R$, $N_{\text{pix}} \times F$ and $N_{\text{pix}} \times F$, respectively, without requiring any data movement.

2.1 Initial Assumptions

We make a number of assumptions about the computing environment and the initial data distribution.

Before beginning the generation of the RM–cube Z, we assume that:

- An SKA imaging pipeline would have produced images and data which would be distributed across the processing units by frequency (channels). Hence each processor unit would have the data for F/P frequencies.

- Each processing unit computes and stores the images for its own frequencies. Hence, each processing unit holds F/P images for a total of $(F \cdot N_{\text{pix}})/P$ pixels from the image cube $F \times N \times N$.

- The computation/analysis is carried out on each pixel independently.

2.2 Possible Approaches

We have identified and analysed three possible approaches:

Approach 1
Send all data to a single processing unit, and then carry out the computation there. This is what is currently done in standard Faraday Tomography approaches. Eq.(1) is computed directly, although through the matrix-matrix product of eq.(3).

Approach 2
Tile the image into a number of disjoint (no overlaps) subimages, one for each processing units. A two-dimensional grid of subimages would probably work best, but other topologies could be considered. Data for all frequencies from each subimage are then gathered in each relevant processing unit. Each processing unit then carries out the computation independently. In other words, we split the full image data cubes into vertical slices, cut parallel to the frequency axis.

We denote by the lowercase superscript $^{(p)}$ that portion of an object that is stored in the p-th processing unit. The full matrix Z (distributed across all processing units as subimages) is therefore notionally given by the union of all subimages:

$$Z = Z^{(1)} \cup Z^{(2)} \cup \ldots \cup Z^{(P)} = \bigcup_{p=1}^{P} Z^{(p)} = \bigcup_{p=1}^{P} (Q^{(p)} + iU^{(p)}) \cdot \Xi \tag{4}$$

Approach 3

Compute eq.(1) for only the portion of frequencies $F^{(q)}$ within each processing unit for all pixels in the image. Then add the results, either by splitting them by subimage (as in approach 2) across all processing units, or by adding them up within a single processing unit:

$$Z = Z^{(1)} + Z^{(2)} + \ldots + Z^{(q)} = \sum_{q=1}^{P} Z^{(q)} = \sum_{p=1}^{q} (Q^{(q)} + iU^{(q)}) \cdot \Xi^{(q)} \tag{5}$$

The superscript $^{(q)}$ refers to the portion of frequencies held in the q-th processing unit. Notice that only the portion of the coefficient matrix $\Xi^{(q)}$ is generated in the q-th processing unit.

Approaches 2 and 3 are clearly both substantial improvements over approach 1. However, approach 2 has a further advantage over approach 3 as complete datacubes for each subimage (comprising data from all frequencies) are created and stored. These datacubes are then ready for any further required processing. Processing can be performed on a regional basis across the sky.

As an example of the utility of this approach, one could envisage providing an interface for a typical radio astronomer who we imagine may be observing the Galactic plane. Whilst setting up their SKA observations, the radio astronomer could define a sky region that covers the plane that they wish to be processed for RMs up to 1 million rad/m^2, while also defining a second region off of the plane that they wish to only be processed for RMs up to 1,000 rad/m^2. Approach 2 could easily provide this separate processing on a subimage basis across an SKA field of view, by calculating eq.(1) for different R on each individual processing unit. This provides a somewhat ideal balancing act: enabling highly computationally intensive science requirements, while also minimising computational resources for processing regions of (largely) empty sky which do not contribute to a given science goal.

2.3 Communication Costs Analysis

Communication costs depend on the size of the data to be sent/received, on the bandwidth B_p of each processing unit, on the overall network bandwidth B_T, and on the maximum usable bandwidth $B_M = \min(B_p \cdot P, B_T)$.

For simplicity's sake, we have ignored any network topology effects, and assume that any one processing unit can communicate with any other, subject to the appropriate bandwidth limits. We assume that each processor stores $(N_{\text{pix}} \cdot F)/P$ data each of 8 bytes length (single precision complex for all elements of $(Q + iU)$.

Approach 1

All data are transferred to a single processing unit by the other $P-1$. Ingestion speed, hence total bandwidth, into that processing unit is limited by its bandwidth B_p.

Approach 2

Each processing unit needs to send $(N_{\rm pix}/P)\cdot(F/P)$ items of data to each of the other processing units and receive as much from them. Any processing unit could communicate with any other, concurrently - the maximum network bandwidth B_T giving an upper limit to the maximum bandwidth achievable.

Approach 3

This is similar to approach 2. However, here the images for each RM are transferred after the computation to be summed in the appropriate processing unit. The entry in table 2 is the same as for case 2. But with the number of RMs, namely, R, instead of the number of frequencies F.

We therefore have the following cost table in terms of time required to complete the communications.

Approach	Time for Data Transfer
1	$w\cdot\frac{N_{\rm pix}\cdot F\cdot(P-1)}{P\cdot B_p}$
2	$w\cdot\frac{N_{\rm pix}\cdot F\cdot(P-1)}{P^2\cdot B_p}\cdot\frac{P\cdot B_p}{B_M}$
3	$w\cdot\frac{N_{\rm pix}\cdot R\cdot(P-1)}{P^2\cdot B_p}\cdot\frac{P\cdot B_p}{B_M}$

Table 2: Parameterised time required for data transfer for the three approaches

.

Approach 1 does not scale with an increasing number of processing units, while approach 2 scales with an increasing number of processing units by a linear factor P, although this may be weighed down by $(P\cdot B_p)/B_M \leq 1$. Approach 3 scales similarly to 2.

2.4 Computational Costs Analysis and Achieving High Performance

Two computational steps are required, and their number of operations are reported in Table 3:

Approach	Number of Flops
Generate the matrix Ξ	$c_\xi F R$
Matrix-matrix product $\bar{Z}=(\bar{Q}+i\bar{U})\cdot\Xi$	$8N_{\rm pix}F R$

Table 3: Parameterised number of operations (flops) required to compute the synthesis matrix Z

1. Compute the RM coefficients matrix, Ξ. We assume that each set of coefficients within the matrix would require c_ξ flops to compute, at the speed s_ξ (in flops/second).

2. Compute the matrix-matrix product, $\bar{Z} = (\bar{Q} + i\bar{U}) \cdot \Xi$, at the speed s_Z (in flops/second).

Splitting the computation of the distributed matrix Z into two steps, first, the computation of the coefficient matrix Ξ, then the matrix-matrix product of eq.(3) is very much faster than applying eq.(1) pixel-by-pixel.

Excellent data re-use can be achieved by using matrix-matrix multiplication, thus leading to very high performance. For example, the LINPACK benchmark used to rank the largest-scale Top 500 supercomputers, consists, in essence, of a large number of matrix-matrix products. Such operations can typically run at close to peak performance, using the BLAS functions CGEMM or ZGEMM for single and double complex numbers, respectively, which are included in most vendors' libraries (QBLAS in NVidia platforms, BLAS in Intel MKL, and AMD ACML). Moreover, both NumPy and MATLAB provide wrappers for the BLAS.

Applying directly eq.(1) pixel-by-pixel would allow only poor data re-use, thus causing intense memory traffic.

The difference in performance is, roughly, given by the ratio between memory access and data consumption (i.e. computation), which is at least a factor of 100. This is borne out by explicit benchmarks, as well as experience on various computational platforms.

The number of flops required by complex matrix-matrix products can be exactly defined: for matrices of size $M \times K$ and $K \times N$, respectively, it is given by $8MKN$. The factor 8 is the number of flops required for each individual product of two complex numbers and summation to a pre-existing one (four products, four additions).

It would not be possible to specify exactly c_ξ (the number of operations required to generate each of the entries in the matrix Ξ) in the same way as can be done for the matrix-matrix product.

In general, $c_\xi \ll 8N_{\mathrm{pix}}$. As a reminder, the symbols s_ξ and s_Z denote the computational speeds (flops per second) achievable during generation of the matrix Ξ and the applying of it to the image cube, respectively. We expect $s_\xi \ll s_Z$ and, for most cases, it is highly likely that:

$$\frac{c_\xi}{s_\xi} \ll \frac{8N_{\mathrm{pix}}}{s_Z} \tag{6}$$

and hence the complex matrix-matrix product would dominate the computational time (and, subsequently, cost).

Data and computational loads are optimally distributed across all processing units, thus also providing good load-balancing. The overall (elapsed) computing times are parameterised in Table 4.

We can see that for SKA scales, in approaches 1 and 3 it is most likely that:

$$\frac{8N_{\mathrm{pix}}}{s_Z} \gg \frac{c}{s_\xi} \tag{7}$$

so that the generation of RM coefficients can be ignored in the computational costs.

Computations could be further optimised by blocking, pipelining, and several other techniques. This is, however beyond, the scope of this current report.

2.5 Overlapping Computation and Communication

Overlapping communication and computation, i.e. transferring data while useful work is being carried out at the same time, can be a viable way to reduce the overall execution times. However, this typically comes at the price of increased code complexity (thereby increasing

Approach	Time for Computation
1	$\frac{c_\xi F R}{s_\xi} + \frac{8 N_{\text{pix}} F R}{s_Z}$
2	$\frac{c_\xi F R}{s_\xi} + \frac{8 N_{\text{pix}} F R}{P s_Z}$
3	$\frac{c_\xi F R}{P s_\xi} + \frac{8 N_{\text{pix}} F R}{P s_Z}$

Table 4: Parameterised computation time required for the three approaches.

monetary costs and development times). Given T_C and T_D, the computation and data transfer times, respectively, the ratio

$$\gamma = \frac{T_C}{T_D} \tag{8}$$

defines the maximum achievable reduction in the overall execution time. The maximum reduction achievable is by $1/2$ when $\gamma = 1$, when communication and computation costs are identical.

The closer to 1 is the ratio γ between computation and communication, the greater is the opportunity for overlapping computation and communication:

$$\gamma \begin{cases} < 1 & \text{data transfer dominates. Maximum gain: } 1+\gamma. \\ = 1 & \text{balance between communication and computation. Maximum gain: 2.} \\ > 1 & \text{computation dominated. Maximum gain: } (1+\gamma)/\gamma. \end{cases} \tag{9}$$

The maximum computation to communication ratio for each of our considered Faraday Tomography approaches is shown in Table 5.

Approach	Maximum Computation to Communication Ratio
1	$\gamma \approx \frac{8 R B_P}{w s_Z}$
1	$\gamma \approx \frac{B_M \left(\frac{c R}{s_\xi} + \frac{8 N_{\text{pix}} R}{P s_Z} \right)}{w N_{\text{pix}}}$
1	$\gamma \approx \frac{8 F B_M}{w P s_Z}$

Table 5: Parameterised computation-to-communication ratio.

3 Worked Examples: LOFAR and SKA

Quantitative estimates of the expected parameter space, performance, and costs for two hypothetical telescopes of sizes similar to LOFAR and SKA respectively, are shown in Table 6. We note that these are simply estimates that indicate the substantial performance improvements that can be obtained via the described approaches. For an SKA-class instrument, the run-time for Faraday Tomography can be improved from 2.1×10^6 seconds (~ 24 days) to 4,200 seconds (~ 1 hour) on infrastructure consisting of 500 compute nodes.

The matrix-matrix approach is used throughout in Table 6. If the computation of eq.(1) were instead carried out pixel-by-pixel, computation times would increase by a factor of 100 or more. For LOFAR-class instruments, approach 1 would require about one hour (which, incidentally, is the order of magnitude required now). For SKA-class instruments, approach 1 would require an unfeasible 6 years, while the (much faster) approach 2 would still require at least 5 days.

	LOFAR-class Instrument			SKA-class Instrument		
	Parameters Values					
Approach	1	2	3	1	2	3
N	2000	2000	2000	20000	20000	20000
N_{pix}	4×10^6	4×10^6	4×10^6	4×10^8	4×10^8	4×10^8
F	280	280	280	65536	65536	65536
R	10,000	10,000	100,000	100,000	10,000	100,000
F	280	280	280	65536	65536	65536
P	20	20	20	500	500	500
B_p (GB/s)	10	10	10	100	100	100
B_T (GB/s)	100	100	100	10000	10000	10000
B_M (GB/s)	100	100	100	10000	10000	10000
w	8	8	8	8	8	8
c_ξ	200	200	200	200	200	200
s_ξ (TFlop/s)	0.01	0.01	0.1	0.1	0.1	0.1
s_Z (TFlop/s)	1	1	1	10	10	10
	Execution Time (seconds)					
Data Transfer	0.2	0.01	0.4	2100	21	33
Compute Ξ	0.06	0.06	0.003	13	13	0.03
Compute Z	23	1.1	2.2	2,100,000	4,200	4,200
Total	23	1.2	1.5	2,100,000	4,230	4,230
gamma	110	110	30	1,000	200	130
gain (%)	0.9	0.9	33.8	0.1	0.5	0.8

Table 6: Example of performance for a LOFAR- and an SKA-class instrument.

Table 6 also confirms that overlapping computation with communication (Subsection 2.5) would only result in marginal improvements, particularly for instruments of SKA size.

4 Future Work and Further Considerations

Future work on Faraday Rotation Tomography would investigate including RM-Clean, the primary deconvolution algorithm for RM datacubes, in this scheme. We believe that it could be achieved rapidly. In addition, we also intend to develop: code to apply to existing precursors and show that that it would scale up to SKA-size systems; an algorithmic prototype for SKA-scales using various approaches such as Dask; of course, implementation on GPUs, where this would be much simplified by using appropriate QBLAS, as shown above.

We would also like to point out that our parallel approach of choice could be extended easily to other algorithms requiring pixel processing, including some local neighbouring pixels. Pixels on the tiles edges would require information from neighbouring tiles. When the image has been split into P tiles, one per processing unit, arranged in a 2D grid, only data from neighbouring tiles need to be exchanged, thus halo exchanges could take place in parallel. As each tile would have a maximum of eight neighbours, halo exchange would be very rapid, only requiring a fraction of the overall data communication time, namely, $8/(P-1)$. Of course, tile sizes could be made smaller by using multiple tiles per processing unit.

In the future, we aim to investigate the application of this approach to other algorithm of importance to Radio Astronomy.

5 Conclusions

The algorithm of Faraday Tomography is a critical piece of software infrastructure that is required to enable the full SKA science case. The substantial computational costs for data on SKA-scales requires careful consideration of new approaches. We have estimated that existing software implementations of this algorithm would require about 6 years to process a single SKA image cube. We have presented a full analysis that details a significant reduction of SKA computing costs per dataset from 24 days to just over 1 hour using approximate SKA compute capabilities. These estimates are inclusive of both data computation and communication, and scale with the number of processing units. We believe that a redistribution of data would lead to almost ideal parallelism across the processing units, and changing the computational engine would increase performance by a considerable factor.

In summary, we have found the following:

1. For LOFAR-class telescopes, parallelising the processing would only provide minor improvements when judged against the typical human timescale. Moving to a much faster matrix-matrix product approach would reap rewards, but these may perhaps be too small in absolute scale to warrant changes.

2. For SKA-class telescopes, the situation is very different:

   (a) Current serialised procedures would be unfeasible, and the science objectives of the SKA would be unachievable:

      i. Current pixel-by-pixel codes would require enormous, unrealistic, amounts of time.

      ii. Even the matrix-matrix product approach would have serious difficulties, due to the sheer amount of resources required.

   (b) A parallelised approach would dramatically reduce both the data transfer and the total computation times by a factor close to P (more so for the computation time).

In conclusion, a full parallelised approach - such as described in this document - that uses matrix-matrix products is essential for ensuring both feasibility and scalability for SKA. In our opinion, the second approach (the first of the two parallel approaches described) would be best and most suitable for SKA. Firstly, it guarantees the re-use of existing codes: RM algorithms would not be applied to the whole image datacube but to portions of it (subimages). Secondly, the algorithm would split the image cube along the frequency-axis into subimages. This scheme would also be useful on a range of other algorithms requiring a similar organisation, for example, data visualisation.

List of Tables

References

Brentjens, M. A. and de Bruyn, A. G.: Faraday rotation measure synthesis, A & A, 441, 1217–1228, 2005.

Burn, B. J.: On the depolarization of discrete radio sources by Faraday dispersion, MNRAS, 133, 67, 1966.

Sun, X. H., Rudnick, L., Akahori, T., Anderson, C. S., Bell, M. R., Bray, J. D., Farnes, J. S., Ideguchi, S., Kumazaki, K., O'Brien, T., O'Sullivan, S. P., Scaife, A. M. M., Stepanov, R., Stil, J., Takahashi, K., van Weeren, R. J., and Wolleben, M.: Comparison of Algorithms for Determination of Rotation Measure and Faraday Structure. I. 1100-1400 MHz, AJ, 149, 60, 2015.