The Journal of Open Source Software

WODEN: A CUDA-enabled package to simulate low-frequency radio interferometric data

Jack L. B. Line[1, 2]

1 International Centre for Radio Astronomy Research, Curtin University, Perth, WA 6845, Australia
2 ARC Centre of Excellence for All Sky Astrophysics in 3 Dimensions (ASTRO 3D)

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

Summary

WODEN is designed to simulate the response of a class of telescope known as an interferometer, producing output "visibilities" for a given astrophysical sky model. Simulated observations allow us to test other software packages that are designed to calibrate and analyse real interferometric data, including verifying expected behaviour with known inputs, and trialling new sky modelling techniques. The WODEN sky model can be specified in dirac-delta like functions on the sky (known in the field as "point sources"), elliptical Gaussian models, or built out of "shapelet" basis functions (allowing complicated morphologies to be created). Users are able to input a bespoke layout for the interferometer, vary a number of observational parameters including time of day, length of observation and frequency coverage, and select from a number of predefined primary beams which encode the response of the receiving elements of an interferometer. This allows simulations of a number of telescopes to be undertaken. WODEN works with input Stokes I, Q, U, V polarisations as a sky model, simulating telescopes with dual linear polarisations, outputting linear Stokes polarisations.

The core functionality of WODEN is written in CUDA as interferometric simulations are computationally intensive but embarrassingly parallel. The compute performance of CUDA allows for large-scale simulations to be run including emission from all directions in the sky. This is paramount for interferometers with a widefield of view such as the Murchison Widefield Array (MWA, Tingay et al. (2013)). A Python wrapper is used to take advantage of community packages such as astropy (Astropy Collaboration et al., 2018, 2013) and pyerfa (van Kerkwijk et al., 2020) and to present a user-friendly interface to WODEN. Those simulating MWA observations can use the MWA `metafits` file to quickly feed in observational parameters to WODEN to match real data.

WODEN can be run to two levels of precision: a `woden_float` precision (which uses a mix of 32 and 64 bit floating precision), and a `woden_double` (which uses nearly entirely 64 bit precision). In Section Estimation of accuracy and computational speed WODEN is shown to produce visibilities to within 0.2% of expected values when running in `woden_float` mode, and 0.000002% in `woden_double` mode, for baselines of length $<=$ 10km.

Underlying methodolgy

An interferometer creates visibilities V by cross-correlating signals detected between pairs of antennas or dishes (baselines), described by coordinates u, v, w. Each visibility is sensitive to the entire sky, directions of which we describe by the direction cosines l, m, n. Ignoring the antenna response, the full integral over the sky can be discretised as

$$V_s(u_i, v_i, w_i) = \sum_j \mathcal{S}_s(l_j, m_j) \exp[-2\pi i(u_i l_j + v_i m_j + w_i(n_j - 1))], \quad (1)$$

where u_i, v_i, w_i are the visibility coordinates of the i^{th} baseline, l_j, m_j, n_j is the sky position

of the $j^{\rm th}$ component in the sky model, and $\mathcal{S}(l_j, m_j)$ is the flux density of that component in a given Stokes polarisation s. WODEN simulates dual-linear polarisation antennas, with each antenna/station having it's own primary beam shape. I can define the response of a dual polarisation antenna to direction l, m as

$$\mathbf{J}(l,m) = \begin{bmatrix} g_{\rm ns}(l,m) & D_{\rm ns}(l,m) \\ D_{\rm ew}(l,m) & g_{\rm ew}(l,m) \end{bmatrix},$$

where g are gain terms, D are leakage terms, and $\rm ns$ refers to north-south and $\rm ew$ east-west aligned antennas. When calculating the cross-correlation responses from antennas 1 and 2 towards direction l, m to produce linear polarisation visibilities, these gains and leakages interact with the four Stokes polarisations *I,Q,U,V* as

$$\begin{bmatrix} V_{12\,XX}(l,m) \\ V_{12\,XY}(l,m) \\ V_{12\,YX}(l,m) \\ V_{12\,YY}(l,m) \end{bmatrix} = \mathbf{J}_1(l,m) \otimes \mathbf{J}_2^*(l,m) \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & i \\ 0 & 0 & 1 & -i \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{12\,I}(l,m) \\ V_{12\,Q}(l,m) \\ V_{12\,U}(l,m) \\ V_{12\,V}(l,m) \end{bmatrix} \tag{2}$$

where $*$ denotes a complex conjugate, and $\otimes$ an outer product (the result of this outer product is written explicitly in the WODEN documentation here). For each baseline, frequency, and time step, WODEN calculates all four linear Stokes polarisations ($V_{XX}, V_{XY}, V_{YX}, V_{YY}$) as defined above for all l_j, m_j in the sky model, and then sums over j, to produce four full-sky linear Stokes polarisation visibilities per baseline/frequency/time.

For a telescope like the MWA, the primary beam $\mathbf{J}(l,m)$ is a complicated pattern on the sky, which is sensitive to emission from directly overhead to all the way down to the horizon. To truly capture the effects of astrophysical foregrounds we therefore have to simulate the entire sky. The MWA Fully Embedded Element (FEE, Sokolowski et al. (2017)) model is currently the most accurate representation of the MWA primary beam, and is incorporated into WODEN.

As the sky model of WODEN is a list of Right Ascension and Declinations with associated flux densities, the user has full control over the projection of the sky into visibilities. To simulate discrete foregrounds, one can simply input any sky catalogue specified in RA/Dec. For diffuse sky models, one could for example input a list of point source/elliptical Gaussians following the HEALPix projection (Górski et al., 2005), or employ a TAN or SIN FITS (Calabretta & Greisen, 2002) projection. WODEN will simply calculate the measurement equation for all directions in the sky model.

Statement of need

Under this discrete sky formalism, upwards of $j >= 25 \times 10^6$ components can be required to achieve the angular resolution required. Furthermore, u, v, w are time and frequency dependent, so to sample in frequency of order 500 times and 100 samples in time, there are of order 10^{12} visibility calculations to make. This makes CUDA acceleration paramount.

Alternative approaches to interferometric simulations exist, such as pyuvsim (Lanman et al., 2019, which sacrifices speed for excellent precision), and RIMEz (which decomposes the sky into spherical harmonics rather than discrete points). WODEN was designed with the Australian MWA Epoch of Reionisation (EoR) processing pipeline in mind, which uses a calibration and foreground removal software called the RTS (Mitchell et al., 2008) in search of signals from the very first stars (see Yoshiura et al. (2021) for a recent use of this pipeline). The RTS creates a sky model using the same formalism above, however the code is not optimised enough to handle the volume of sources to simulate the entire sky. To test the RTS method of sky generation, we therefore needed a fast and discretised method. Another excellent CUDA accelerated simulation package, OSKAR (Mort et al., 2010), addresses these two points.

However, the RTS also generates parts of the sky model via shapelets (see Line et al. (2020) for an overview), which OSKAR cannot. Furthermore, in real data, the precession/nutation of the Earth's rotational axis causes sources to move from the sky coordinates as specified in the RA, DEC J2000 coordinate system. The RTS is designed to undo this precession/nutation, and so a simulation fed into the RTS should *contain* precession. WODEN adds in this precession using the same method as the RTS to be consistent. This unique combination of CUDA, shapelet foregrounds, the MWA FEE primary beam, along with source precession, created the need for WODEN. These effects should not preclude other calibration packages from using WODEN outputs however, meaning WODEN is not limited to feeding data into the RTS alone.

Estimation of accuracy and computational speed

The goal of this section is to test the accuracy of the functionality of WODEN, including reading of inputs, the array coordinate calculations, the precession/nutation correction, l, m, n and u, v, w calculations, flux density frequency extrapolation via spectral index, calculation of Equation 2, and writing out of the data to uvfits files.

To test the absolute accuracy of WODEN, we first need a set of input parameters that have an analytically predictable outcome. If we ignore the beam response and polarisation, set the flux density of a source to one, and consider a single baseline and sky direction, the measurement equation (Equation 1) becomes[1]

$$V(u, v, w) = \exp[2\pi i(ul + vm + w(n-1))]. \tag{3}$$

We can use Euler's formula to split V into real and imaginary components. If I label the phase for a particular source and baseline as

$$\phi = 2\pi \left(ul + vm + w(n-1)\right)$$

then the real and imaginary parts of the visibility V_{re}, V_{im} are

$$V_{re} = \cos(\phi), \quad V_{im} = \sin(\phi).$$

If we can therefore set ϕ to a number of values which produce known sine and cosine outputs, by selecting specific combinations of u, v, w and l, m, n, we can simulate visibilities with predictable outputs. First of all, consider the simplified case $\phi_{\rm simple}$ when $u, v, w = 1, 1, 1$. In that case,

$$\frac{\phi_{\rm simple}}{2\pi} = l + m + (n-1).$$

If we further set $l = m$, we end up with

$$\frac{\phi_{\rm simple}}{2\pi} = 2l + (n-1),$$
$$l = \sqrt{\left(\frac{1-n^2}{2}\right)}$$

It can be shown (via Wolfram Alpha) that a solution for n is

[1]Note there is no negative at the front inside the exponential for $V(u, v, w)$. After numerous comparisons to other simulation packages, and imaging to check the input source positions match, I find dropping the negative gives the correct outputs.

$$n = \frac{\sqrt{2}\sqrt{-\phi_{\text{simple}}^2 - 4\pi\phi_{\text{simple}} + 8\pi^2} + \phi_{\text{simple}} + 2\pi}{6\pi}$$

which we can then use to calculate values for l, m through

$$l = m = \sqrt{\frac{1 - n^2}{2}}.$$

Practically then, if we input the following combinations of l, m, n into Equation 3 our output visibilities should exactly match the $\cos(\phi)$, $\sin(\phi)$ values.

ϕ_{simple}	l, m	n	$\cos(\phi)$	$\sin(\phi)$
0	0.0	1.0	1.0	0
$\pi/6$	0.0425737516338956	0.9981858300655398	$\sqrt{3}/2$	0.5
$\pi/4$	0.0645903244635131	0.9958193510729726	$\sqrt{2}/2$	$\sqrt{2}/2$
$\pi/3$	0.0871449863555500	0.9923766939555675	0.5	$\sqrt{3}/2$
$\pi/2$	0.1340695840364469	0.9818608319271057	0.0	1.0
$2\pi/3$	0.1838657911209207	0.9656017510914922	-0.5	$\sqrt{3}/2$
$3\pi/4$	0.2100755148372292	0.9548489703255412	$-\sqrt{2}/2$	$\sqrt{2}/2$
$5\pi/6$	0.2373397982598921	0.9419870701468823	$-\sqrt{3}/2$	0.5
π	0.2958758547680685	0.9082482904638630	-1.0	0.0
$7\pi/6$	0.3622725654470420	0.8587882024392495	$-\sqrt{3}/2$	-0.5
$5\pi/4$	0.4003681253515569	0.8242637492968862	$-\sqrt{2}/2$	$-\sqrt{2}/2$

Table 1: l, m, n combinations used in accuracy test

To test for a range of baseline lengths, we can make a simplification where we set all baseline coordinates to be equal, i.e. $u = v = w = b$ where b is some length in units of wavelength. In this form, the phase including the baseline length ϕ_b is

$$\phi_b = 2\pi b\,(l + m + n - 1) = b\phi_{\text{simple}}.$$

As sine/cosine are periodic functions, the following is true:

$$\phi_{\text{simple}} = \phi_{\text{simple}} + 2\pi\text{n}$$

where n is some integer. This means for a given ϕ_{simple}, we can find an appropriate b that should still result in the expected sine and cosine outputs by setting

$$b\phi_{\text{simple}} = \phi_{\text{simple}} + 2\pi\text{n},$$
$$b = \frac{\phi_{\text{simple}} + 2\pi\text{n}}{\phi_{\text{simple}}}$$

for a range of n values. The values of n and the resultant size of b that I use in testing are shown in Table 2.

`WODEN` reads in an input array layout specified in local east, north, height E, N, H coordinates. It then converts those into local X, Y, Z coordinates via the equations

$\phi_{\rm simple}$	$b(\mathrm{n}=1)$	$b(\mathrm{n}=10)$	$b(\mathrm{n}=100)$	$b(\mathrm{n}=1000)$	$b(\mathrm{n}=10000)$
0	6.3	62.8	628.3	6283.2	62831.9
$\pi/6$	13.0	121.0	1201.0	12001.0	120001.0
$\pi/4$	9.0	81.0	801.0	8001.0	80001.0
$\pi/3$	7.0	61.0	601.0	6001.0	60001.0
$\pi/2$	5.0	41.0	401.0	4001.0	40001.0
$2\pi/3$	4.0	31.0	301.0	3001.0	30001.0
$3\pi/4$	3.7	27.7	267.7	2667.7	26667.7
$5\pi/6$	3.4	25.0	241.0	2401.0	24001.0
π	3.0	21.0	201.0	2001.0	20001.0
$7\pi/6$	2.7	18.1	172.4	1715.3	17143.9
$5\pi/4$	2.6	17.0	161.0	1601.0	16001.0

Table 2: Range of baseline lengths used in conjunction with the l, m, n coordinates in Table 1.

$$X = -\sin(\phi_{\rm lat})N + \cos(\phi_{\rm lat})H \tag{4}$$
$$Y = E \tag{5}$$
$$Z = \cos(\phi_{\rm lat})N + \sin(\phi_{\rm lat})H \tag{6}$$

where $\phi_{\rm lat}$ is the latitude of the array. X, Y, Z are used to calculate the u, v, w coodinates (c.f. Chapter 4 in Thompson et al. (2017)). If we place our interferometer at a $\phi_{\rm lat} = 0.0^\circ$ and set the local sidereal time (LST) to zero, the calculation of u, v, w becomes

$$u = E;\ v = N;\ w = H; \tag{7}$$

allowing us to set $E, N, H = b$ for our values on b in Table 2. Furthermore, we can convert our l, m values from Table 1 into RA,Dec (α, δ) via:

$$\delta = \arcsin(l) \tag{8}$$
$$\alpha = \arcsin\left(\frac{l}{\cos(\arcsin(l))}\right) \tag{9}$$

Following the `RTS`, `WODEN` first of all calculates *X,Y,Z* using the array latitude at the time of the observation. It then uses the `PAL` (Jenness & Berry, 2013) palPrenut function to generate a rotation matrix to rotate the local *X,Y,Z* coordinates back to the J2000 epoch, as well as the LST and latitude of the array. This accounts for the precession/nutation of the Earth with respect to the J2000 RA/Dec coordinates that the sky model is specified in. To manifest the outcomes of Equations 7, 8, and 9, we have to apply the opposite rotation about $\phi_{\rm lat}$ as defined by Equations 4, 5, and 6, as well as the rotations applied via palPrenut to account for precession/nutation, to our input E, N, H coordinates.

Figure 1 shows the result of running multiple simulations, each with: an array layout with a single baseline; a single time step and frequency channel; a single point source sky model; a primary beam model with gains of one and zero leakage. All possible combinations of l, m, n and b as listed in Tables 1 and 2 are run. Each simulation is run with the parameters specified in Table 3.

Parameter	Value	Manifestation in simulation
Date (UTC)	2020-01-01 12:00:00.0	WODEN must correct for precession and nutation
Latitude (deg)	0.1095074	After precess/nut correction, latitude is 0.0°
Longitude (deg)	79.6423588	After precess/nut correction, LST is 0.0°
Frequency (MHz)	299.792458	Means $\lambda = 1$, so wavelength scaled $u, v, w = E, N, H$
Reference frequency for sky model (MHz)	150	WODEN has to extrapolate the flux density
Spectral Index	-0.8	Needed to extrapolate flux density
Reference Stokes I flux density (Jy)	1.7401375	Should be extrapolated to a flux of 1.0 at the simulation frequency

Table 3: Common settings for the simulations run to produce the results in Figure 1

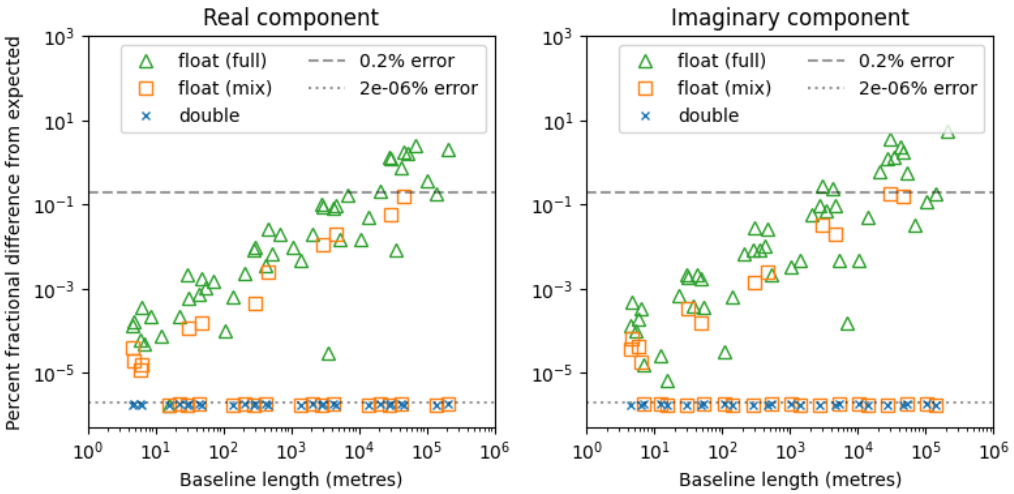

Figure 1: The absolute fractional difference (in percent) of visibilities calculated by WODEN, compared to their expected values, with the real component shown on the left, and the imaginary shown on the right. The green triangles show an older version of WODEN which used only 32 bit precision; the orange square show the v1.1 `woden_float` version which uses a mixture of 32 and 64 bit precision; the blue crosses show the `woden_double` mode which uses nearly entirely 64 bit precision.

All array layouts, sky models, and simulations are run by `WODEN/test_installation/absolute_accuracy/run_the_absolute_accuracy_test.sh`, which can be run as part of a test suite bundled with WODEN. This script reads the values out of the output `uvfits` files, and produces the plot in Figure 1.

Version 1.0 of WODEN was fully 32 bit, which produced the green triangles in Figure 1, with longer baselines consistently a few percent off expectations. A two time processing slow down by moving to a combined 32 and 64 bit `woden_float` mode (orange squares) improves the accuracy to $<=$ 0.2% on the longer baselines. The entirely 64 bit `woden_double` precision mode is consistent in precision across baseline length, sitting at $<$ 2e-6% accuracy. The `woden_float` and `woden_double` executables are available in Version 1.1 (TODO link to release once release is made), and can be switched between via a command line option in `run_woden.py`. It should be noted that these offset errors are deterministic, meaning comparison

between different simulations out of Version 1.0 WODEN are consistent; these errors matter most when comparing to real data.

As 32 and 64 bit precision calculations are performed in physically different parts of an NVIDIA GPU, with cards typically having less double precision hardware that single, the `woden_double` version is slower that the `woden_float`. Each card will show a different slow-down between the two modes. As a test, I ran a simulation using a catalogue of over 300,000 sources. The number of sources above the horizon and the simulation settings used are listed in Table 4, along with the speed difference between the `woden_float` and `woden_double` executables for two different NVIDIA GPU cards.

Parameters	Value
Time steps	14
Frequency channels	80
Point sources components	207673
Gaussian components	1182
Shapelet components (basis functions)	62 (10400)
Primary beam model	MWA FEE
GTX 1080 Ti `woden_float` simulation time	10min 39sec
GTX 1080 Ti `woden_double` simulation time	55min 46sec
V100 `woden_float` simulation time	4min 35sec
V100 `woden_double` simulation time	5min 55sec

Table 4: Benchmark simulation to compare `woden_float` and `woden_double` speeds. Each shapelet component can have several basis function calculations, each more expensive that a point source component calculation. The MWA FEE is the most computationally expensive beam model included with WODEN.

Given this > 5 times slow down on a desktop card, having the option to toggle between `woden_float` and `woden_double` allows quick experimentation using `woden_float` and longer science-quality runs with `woden_double`. Luckily, for cards like the V100, the slow down is around 1.3. Note that these simulations can easily be broken up and run across multiple GPUs if available, reducing the real time taken to complete the simulations.

Example application

In Line et al. (2020), we compared two methods to model Fornax A: a combination of point and elliptical Gaussians, compared to shapelets (see Figure 2). We were able to quickly compare the computational efficiency of the methods using a desktop, and comment on their respective strengths and weaknesses in regard to foreground removal for EoR purposes. Furthermore, as we could control the simulations, we could compare the methods in the absence of other processing systematics that are present in the real data from the MWA, which dominated the comparison when using the RTS alone.

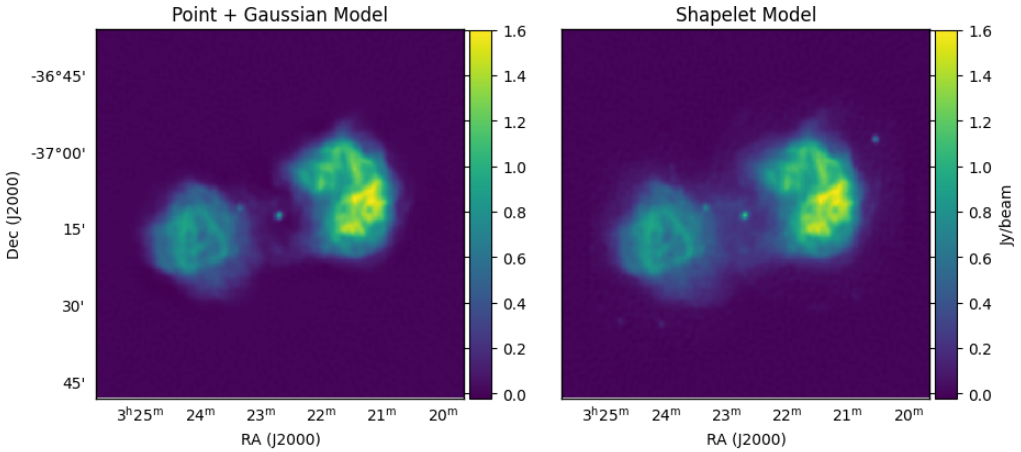

Figure 2: Two methods to simulate Fornax A visibilities are compared here (both imaged using `WSClean` (Offringa et al., 2014; Offringa & Smirnov, 2017)), with point and elliptical Gaussians on the left, and shapelets on the right.

Documentation

The documentation for `WODEN` can be found here on readthedocs, including a detailed installation guide, ways to test a local installation, details of the calculations `WODEN` makes under the hood, and worked examples, which are also included in the `github` repo.

Acknowledgements

I acknowledge direct contributions from Tony Farlie (who taught me how pointer arithmetic works in `C`) and contributions from Bart Pindor and Daniel Mitchell (through their work in the `RTS` and through advising me on `CUDA`). I'd like to thank Chris Jordan who acted as a sounding board as I learnt `C` and `CUDA`. I like to thank both Matthew Kolopanis and Paul La Plante for reviewing the code and giving useful suggestions on how to improve the code.

This research was supported by the Australian Research Council Centre of Excellence for All Sky Astrophysics in 3 Dimensions (ASTRO 3D), through project number CE170100013.

References

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., Günther, H. M., Lim, P. L., Crawford, S. M., Conseil, S., Shupe, D. L., Craig, M. W., Dencheva, N., Ginsburg, A., Vand erPlas, J. T., Bradley, L. D., Pérez-Suárez, D., de Val-Borro, M., Aldcroft, T. L., Cruz, K. L., Robitaille, T. P., Tollerud, E. J., … Astropy Contributors. (2018). The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. *156*(3), 123. https://doi.org/10.3847/1538-3881/aabc4f

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., … Streicher, O. (2013). Astropy: A community Python package for astronomy. *558*, A33. https://doi.org/10.1051/0004-6361/201322068

Calabretta, M. R., & Greisen, E. W. (2002). Representations of celestial coordinates in FITS. *395*, 1077–1122. https://doi.org/10.1051/0004-6361:20021327

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., & Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *622*(2), 759–771. https://doi.org/10.1086/427976

Jenness, T., & Berry, D. S. (2013). PAL: A Positional Astronomy Library. In D. N. Friedel (Ed.), *Astronomical data analysis software and systems XXII* (Vol. 475, p. 307).

Lanman, A. E., Hazelton, B. J., Jacobs, D. C., Kolopanis, M. J., Pober, J. C., Aguirre, J. E., & Thyagarajan, N. (2019). Pyuvsim: A comprehensive simulation package for radio interferometers in python. *Journal of Open Source Software*, *4*(37), 1234. https://doi.org/10.21105/joss.01234

Line, J. L. B., Mitchell, D. A., Pindor, B., Riding, J. L., McKinley, B., Webster, R. L., Trott, C. M., Hurley-Walker, N., & Offringa, A. R. (2020). Modelling and peeling extended sources with shapelets: A Fornax A case study. *Publications of the Astronomical Society Australia*, *37*, 15. https://doi.org/10.1017/PASA.2020.18

Mitchell, D. A., Greenhill, L. J., Wayth, R. B., Sault, R. J., Lonsdale, C. J., Cappallo, R. J., Morales, M. F., & Ord, S. M. (2008). Real-Time Calibration of the Murchison Widefield Array. *IEEE Journal of Selected Topics in Signal Processing*, *2*(5), 707–717. https://doi.org/10.1109/JSTSP.2008.2005327

Mort, B. J., Dulwich, F., Salvini, S., Adami, K. Z., & Jones, M. E. (2010). OSKAR: Simulating digital beamforming for the SKA aperture array. *2010 IEEE International Symposium on Phased Array Systems and Technology*, 690–694. https://doi.org/10.1109/ARRAY.2010.5613289

Offringa, A. R., McKinley, B., Hurley-Walker, N., Briggs, F. H., Wayth, R. B., Kaplan, D. L., Bell, M. E., Feng, L., Neben, A. R., Hughes, J. D., Rhee, J., Murphy, T., Bhat, N. D. R., Bernardi, G., Bowman, J. D., Cappallo, R. J., Corey, B. E., Deshpande, A. A., Emrich, D., … Williams, C. L. (2014). Wsclean: An implementation of a fast, generic wide-field imager for radio astronomy. *Monthly Notices of the Royal Astronomical Society*, *444*(1), 606–619. https://doi.org/10.1093/mnras/stu1368

Offringa, A. R., & Smirnov, O. (2017). An optimized algorithm for multiscale wideband deconvolution of radio astronomical images. *Monthly Notices of the Royal Astronomical Society*, *471*(1), 301–316. https://doi.org/10.1093/mnras/stx1547

Sokolowski, M., Colegate, T., Sutinjo, A. T., Ung, D., Wayth, R., Hurley-Walker, N., Lenc, E., Pindor, B., Morgan, J., Kaplan, D. L., Bell, M. E., Callingham, J. R., Dwarakanath, K. S., For, B.-Q., Gaensler, B. M., Hancock, P. J., Hindson, L., Johnston-Hollitt, M., Kapińska, A. D., … Zheng, Q. (2017). Calibration and Stokes Imaging with Full Embedded Element Primary Beam Model for the Murchison Widefield Array. *Publications of the Astronomical Society Australia*, *34*, e062. https://doi.org/10.1017/pasa.2017.54

Thompson, A. R., Moran, J. M., & Swenson, Jr., George W. (2017). *Interferometry and Synthesis in Radio Astronomy, 3rd Edition*. https://doi.org/10.1007/978-3-319-44431-4

Tingay, S. J., Goeke, R., Bowman, J. D., Emrich, D., Ord, S. M., Mitchell, D. A., Morales, M. F., Booler, T., Crosse, B., Wayth, R. B., Lonsdale, C. J., Tremblay, S., Pallot, D., Colegate, T., Wicenec, A., Kudryavtseva, N., Arcus, W., Barnes, D., Bernardi, G., … Wyithe, J. S. B. (2013). The Murchison Widefield Array: The Square Kilometre Array Precursor at Low Radio Frequencies. *Publications of the Astronomical Society Australia*, *30*, e007. https://doi.org/10.1017/pasa.2012.007

van Kerkwijk, M., Tollerud, E., Woillez, J., Robitaille, T., Bray, E. M., Valentino, A., Sipocz, B., Droettboom, M., Deil, C., Seifert, M., Conseil, S., Aldcroft, T., Price-Whelan, A., StuartLittlefair, Lim, P. L., Sulzbach, B., Beaumont, C., Cara, D., Crichton, D., … Sumak, J. (2020). *Pyerfa* (Version 2.0.0). Zenodo. https://doi.org/10.5281/zenodo.3940699

Yoshiura, S., Pindor, B., Line, J. L. B., Barry, N., Trott, C. M., Beardsley, A., Bowman, J., Byrne, R., Chokshi, A., Hazelton, B. J., Hasegawa, K., Howard, E., Greig, B., Jacobs, D., Jordan, C. H., Joseph, R., Kolopanis, M., Lynch, C., McKinley, B., ... Zheng, Q. (2021). A new MWA limit on the 21 cm power spectrum at redshifts 13–17. *Monthly Notices to the Royal Astronomical Society*, *505*(4), 4775–4790. https://doi.org/10.1093/mnras/stab1560