Developments on the LOFAR and SKA calibration and imaging software

Tammo Jan Dijkema[1], Maikel Lukkezen[3], Maik Nijhuis[4], André Offringa[1], Chiara Salvoni[3], Sebastiaan van der Tol[1], Mark de Wever[2], and Stefan J. Wijnholds[1]

[1]*ASTRON, Dwingeloo, The Netherlands;* dijkema@astron.nl , offringa@astron.nl , tol@astron.nl , wijnholds@astron.nl

[2]*S[&]T, Delft, The Netherlands;* mark.dewever@stcorp.nl

[3]*CGI, Rotterdam, The Netherlands;* maikel.lukkezen@cgi.com , chiara.salvoni@cgi.com

[4]*TriOpSys, Utrecht, The Netherlands;* maik.nijhuis@triopsys.nl

Abstract.

We present recent developments on calibration and imaging software used in particular for LOFAR. SKA's Team Schaap develops this software. One of the goals is integrating it into the SKA Science Data Processor. The software stack consists of the top-level packages WSClean for imaging and DP3 for calibration and flagging. We have made the software more modular, so that relevant parts of the software can also be used by other packages. Recent improvements in WSClean include a faceting option, which has enabled creating a 10 Gigapixel image. Image domain gridding (IDG) allows applying screen-based corrections. In the pipeline and calibration software DP3, we have implemented support for baseline-dependent averaging (BDA), which can reduce data volumes by an order of magnitude while retaining the image quality. The underlying beam library, EveryBeam, was made generic enough to support any beam model which is expressed in spherical harmonics. This allows also to use recent LOFAR beam simulations as well as SKA models. EveryBeam also allows passing arbitrary A-terms for imaging direction-dependent effects other than beam models.

1. Introduction

The Low Frequency Array (LOFAR) (van Haarlem et al. 2013) is a pathfinder for the Square Kilometre Array (SKA), in particular for its low-frequency receiving system (SKA-LOW) (Labate et al. 2022). Owing to the large conceptual and architectural similarities between LOFAR and SKA-LOW, the data processing challenges to be addressed by the (self-)calibration and imaging workflows for SKA-LOW are similar to those already being addressed by LOFAR. One of the main goals of Team Schaap, which consists of developers from ASTRON and three industry partners, is therefore to adapt the LOFAR calibration and imaging software to make it suitable for the SKA. In the process, the user feedback from LOFAR users helps to ensure that the developed software components are suitable for their purpose. Team Schaap is part of SKA's

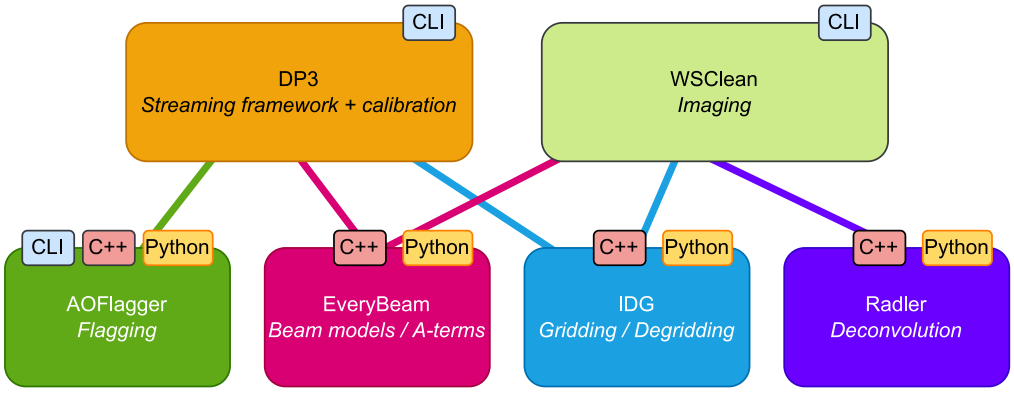

Figure 1. Figure: components in the Schaap software stack with their dependencies. The boxes in the top indicate the interface: command-line interface (CLI), C++, or Python.

Data Processing Agile Release Train working with Scaled Agile Framework (SAFe) methods.

This contribution starts with an overview of the software components that form the heart of the (self-)calibration and imaging pipelines for LOFAR. In Sec. 3 we describe our integration philosophy to make the software components originally developed for LOFAR suitable for execution in SKA's data processing environments. This forms a natural bridge to the overview of new features presented in Sec. 4.

2. Software components

Figure 1 gives an overview of our software stack. Its main components are the Default Pre-Processing Pipeline (DP3) and WSClean which use four other components that can also be used independently.

DP3 (Van Diepen et al. 2018) is LOFAR's pipelined processing code. It can perform all sorts of operations on the data in a pipelined way, so that the data are read and written only once. These operations include flagging, averaging, phase shifting, demixing to remove strong sources and both direction-independent and direction-dependent calibration. It is written in C++ and has a command line interface.

WSClean (Offringa et al. 2014; Offringa & Smirnov 2017) is a generic wide-field imager for radio astronomy that supports multi-scale wideband deconvolution. It supports W-Stacking as well as Image Domain Gridding (IDG) (Van der Tol et al. 2018) for imaging. The resulting dirty images can be deconvolved with using various deconvolution methods including CLEAN and wideband multi-scale CLEAN. WSClean is written in C++ and is executed via its command line interface.

The AOFlagger (Offringa et al. 2012) is a tool that can find and remove radio-frequency interference (RFI) in radio astronomical observations. It can make use of Lua scripts to make flagging strategies flexible, and the tools are applicable to a wide set of telescopes. Its C++ shared library object can be called for flagging in DP3, while its command line interface and Python interface allow stand-alone usage.

For various stages of processing, knowledge about the direction-dependent instrument response is needed. For LOFAR, this was captured in a package called LOFAR-

Beam. This package was generalised to support calculation of the instrument response over the field-of-view of the observation of phased array systems prior and after beamforming as well as the primary beam response of dish-based telescopes. The package was thus aptly renamed to EveryBeam, which is now used by both DP3 and WSClean via its C++ interface. Telescope responses can also be calculated via a Python interface.

In an effort to make the software more modular, Team Schaap have split off the deconvolution functions from WSClean into a separate library, the Radio Astronomical Deconvolution Library (Radler). This separation of imaging and deconvolution resulted in cleaner interfaces that should facilitate further integration of both imaging and deconvolution functionality in SKA's processing function library. WSClean obviously uses the C++ interface to Radler by linking to a shared library. For integration in SKA's workflows, a Python interface has been provided.

Image Domain Gridding (IDG, Van der Tol et al. (2018)) is a fast method for convolutional resampling (gridding/degridding) of radio astronomical data (visibilities). Direction dependent effects (DDEs) or A-terms can be applied in the gridding process. This library has implementations for both the CPU and the GPU. WSClean uses IDG via its C++ interface. The Python interface to the IDG library started as a convenient tool for experimenting but has matured to enable integration in Python-based processing workflows, like those currently being developed for SKA.

3. Integration philosophy

Our software components should be modular, usable from Python, and have high-performance. Since most LOFAR software is written in C++, part of our work is to make Python wrappers. We use pybind11 (Jakob et al. 2017). A design goal is to keep the Python interface as close as possible to the C++ interface, while making the Python interface feel 'pythonic'. Packages should all be easily installable. We aim to make Python packages available as binary wheels. For packaging pure C++ packages, we rely on CMake and apt packaging.

4. New features developed by Team Schaap

4.1. Baseline-dependent averaging (BDA) in DP3

As described by Salvoni et al. (2022), BDA can reduce data volumes, specifically for interferometers with a dense core. We have adapted several DP3 steps to work with data which have been averaged in a baseline-dependent way. The direction-dependent calibration methods have been adapted to be agnostic on possible averaging in the data.

4.2. Support for beam models expressed in spherical harmonics

A new beam model for LOFAR is being simulated using full-wave electromagnetic simulations that compute the beam pattern for every individual dipole. These simulation results are interpolated using spherical harmonics. We have added functionality to EveryBeam to evaluate these spherical harmonics, and thus the beam simulations.

4.3. Direction-dependent point-spread functions in deconvolution

The central stations of LOFAR each have 24 high-band tiles (where a tile consists of 4×4 dipoles). There are also stations with 48 or 96 tiles. When using these stations together, typically the inner tiles are used, so that the beam is approximately equal for every station in the array. This has the disadvantage that a lot of collecting area of the larger stations is not used.

In deconvolution, normally the assumption is made that the point-spread function is a shifted version of the point-spread function in the phase center. If the beam is not the same over the array, this approximation becomes worse. By evaluating a grid of point-spread functions over the image, the approximations made in deconvolution are better and the minor deconvolution cycle works better.

4.4. Faceted imaging

Deep imaging requires imaging out to the first sidelobes and correction for direction-dependent effects. Combined with LOFAR's long baselines, this results in images of 10 Gigapixels in size. In facet-based imaging approaches, the image is split into facets that are imaged separately. This does not only split the imaging problem in a number of smaller subproblems, but also provides a natural way to make direction-dependent corrections in a number of discrete directions across the image as demonstrated by Tasse et al. (2018). We have reimplemented this concept in WSClean to enable its use in the calibration and imaging pipelines supporting LOFAR's non-expert users. This functionality has been picked up by some expert users already who are now regularly making images of 60k-by-60k pixels and larger and provide useful feedback that will allow us to deliver this as field-tested functionality to the SKA.

5. Summary

In this contribution, we gave an overview on recent updates in LOFAR's calibration and imaging stack. These updates include BDA in the direction-dependent calibration solver, facet imaging, which enabled creation of a 10 Gigapixel image, and deconvolution with a direction-dependent psf. These features are already used in LOFAR workflows, which results in useful user feedback. This allows us to further improve these components and deliver field-tested functionality to the SKA.

References

Jakob, W., Rhinelander, J., & Moldovan, D. 2017, pybind11 – seamless operability between c++11 and python. Https://github.com/pybind/pybind11

Labate, M. G., et al. 2022, SPIE Journal of Astronomical Telescopes, Instruments and Systems, 8

Offringa, A. R., McKinley, B., Hurley-Walker, et al. 2014, MNRAS, 444, 606

Offringa, A. R., & Smirnov, O. 2017, MNRAS, 471, 301

Offringa, A. R., vam der Gronde, J. J., & Roerdink, J. B. T. M. 2012, A&A, 539, 1

Salvoni, C., van den Bergh, R. R., Dijkema, T. J., Maljaars, J., Nijhuis, M., Offringa, A. R., van der Tol, S., de Wever, M., & Wijnholds, S. J. 2022, in 2022 3rd URSI Atlantic and Asia Pacific Radio Science Meeting (AT-AP-RASC), 1

Tasse, C., et al. 2018, Astronomy & Astrophysics, 611, 1

Van der Tol, S., Veenboer, B., & Offringa, A. R. 2018, A&A, 616, A27. URL `https://doi.org/10.1051/0004-6361/201832858`

Van Diepen, G., Dijkema, T. J., & Offringa, A. 2018, DPPP: Default Pre-Processing Pipeline, Astrophysics Source Code Library, record ascl:1804.003. 1804.003
van Haarlem, M. P., et al. 2013, Astronomy & Astrophysics, 556, 1