AURIS.jl : A High Performance Radio Interferometry Software Suite

Paul Barrett[1,2], Paul Tiede[3,4], & James McGarry[5]

[1]The George Washington University, [2]MACRO Consortium, [3]Black Hole Initiative – Harvard University,
[4]University of British Columbia, Okanagan, [5]University of Hertfordshire



THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Abstract

Most radio astronomy software was designed and developed over 30 years ago before parallel processing was the norm. Therefore, it does not take advantage of the high-performance computing resources that are available today such as new software technology in the form of on-the-fly compilation and advanced numerical algorithms like auto-differentiation. AURIS (**A**stronomical **U**tilities for **R**adio **I**nterferometry and **S**imulation) is a Julia software suite designed to use the latest technologies whether that be now or in the future. This enables AURIS to process large volumes of data efficiently and accurately. All algorithms are designed to take advantage of the inherently parallel nature of radio interferometry data by distributing computations across multicore CPUs and GPUs, when available. In addition, AURIS by default uses half-precision (16-bit) floats for most calculations, which reduces the memory footprint and processing time by half without a decrease in accuracy. As a result, the AURIS calibration pipeline is over fifty times faster than the NRAO CASA calibration pipeline for VLA data, reducing calibration time from several hours to several minutes. This poster will discuss how the various features of the Julia programming language and its associated packages are used to implement a modular, high-performance software suite that can quickly and easily adapt to the increasing volume and wide variety of radio data, such as the next-generation VLA (ngVLA) and the Event Horizon Telescope (EHT).

Resource Constraints

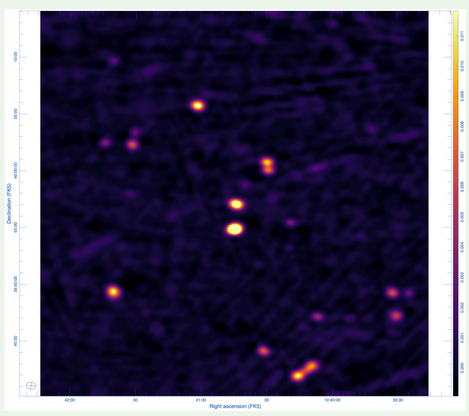
Productivity is Key. Developing a high-performance, general-purpose software suite with no direct funding for full-time scientists and developers requires using the most productive technology available.

- **Julia**: Julia is a high-level, array programming language developed for high-performance scientific computing. Think Matlab, but free. Julia compiles high-level expressions directly to machine code, including GPUs. It has **the performance of C** and **twice the productivity of Python**. **JACC.jl** provides a portable array API for parallel computing on CPUs and GPUs.
- **Claude**: Claude Code is an artificial intelligence (AI) application that expedites code generation. It enables the conversion of software in one language to another (e.g., from C/C++ to Julia). Claude enables our project to quickly implement a functional set of software, which can be modified and improved at a later date.

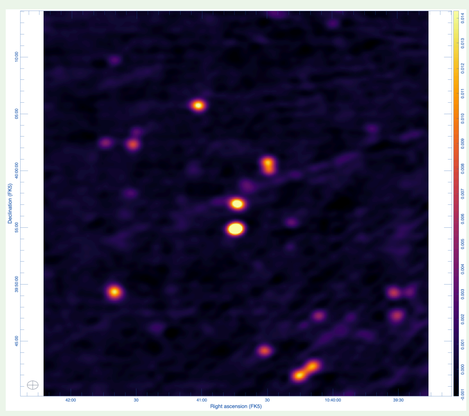
Claude Code

- **Claude generated code is not ideal.** Claude knows Python and C/C++, not Julia, although it is getting better. Claude must be taught to generate idiomatic Julia code by giving Claude the functional signature and having it generate the internal code. Otherwise, Claude will generate unnecessary and restrictive code.
- **Claude** needs to be taught about **multiple dispatch**, which performs a function call using all functional arguments, not just the first argument as with object-oriented languages. Multiple dispatch encourages modular programming by enabling functions to have a simple, specific task. There is rarely a need for `if-then-else` or `switch-case` statements.
- **Claude** needs to be taught about **abstract types**, which enable a single function to be applicable to multiple similar types. For example, `func(a::AbstractFloat)` works with all floating-point types from 4-bit to 128-bit and larger. One function works for multiple types, increasing productivity.

Example: CASA and AURIS Pipelines



CASA Pipeline
(140 minutes)



AURIS Pipeline
(1.5 minutes)

CASA and AURIS images of the Seyfert 2 galaxy XCS J1040+3957. The images are generated using similar pipelines with similar input parameters.

Motivation

- **Performance:** Calibration of radio interferometry data should take no more than a tenth of the time that it takes to acquire the data. Compare this to the Common Astronomy Software Applications (CASA) which takes more than three times longer to calibrate data as it does to acquire it. This performance is inadequate for the data volumes of current and future radio arrays.
- **Versatility:** The imaging paradigm of previous radio analysis software is no longer applicable to current and future radio arrays. The sensitivity of current and future arrays enables photo-polarimetry of point sources on timescales of seconds. Generating light curves based on thousands of images is inefficient and unnecessary. A new paradigm that encompasses both imaging and high-time resolution analysis is needed.
- **Technology:** Current radio interferometry software is based on thirty year old technology, when multicore CPUs and GPU were not available. New software is needed that can take advantage of the massively parallel nature of radio interferometry data and today's high-performance software and hardware technology.

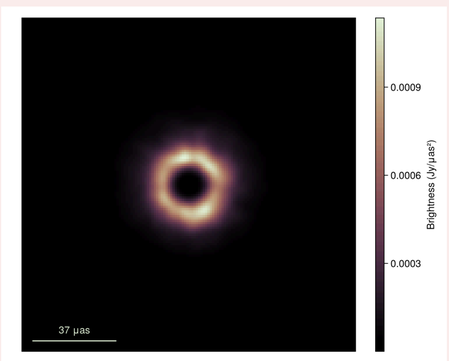
Features and Performance

- **Data Formats:** AURIS is designed to interact with multiple data formats, specifically, Measurement Set v2 (MSv2), Measurement Set V4 (MSv4, aka XRadio), Science Data Model (SDM), and FITS (UVFITS & FITSIDI).
- **Polarization Feeds:** Radio antennas can have linear or circular polarization feeds. Most analysis software can handle only a single type. AURIS is designed to handle any combination as in the case of the Event Horizon Telescope.
- **Stokes Polarimetry:** AURIS will endeavour to process all correlation modes to provide full polarization results.
- **Visibility Fitting:** The processing of radio interferometry data can benefit greatly by performing much of the data analysis in visibility space, instead of Fourier space. For example, a high time-resolution light curve of a point source does not require the creation of multiple images. Visibility fitting is faster and more accurate because the analysis is performed in the statistical space of the data and the processing can be performed in parallel.
- **16-Bit Floats:** Julia has native half-precision (16-bit) floating-point values. An 8-bit analog-to-digital converter (ADC) requires only a 16-bit float. By using 16-bit floats, memory is reduced and performance is improved ($\approx 2\times$) without loss of accuracy.
- **Memory-Mapped Files:** Laptops with limited memory can benefit from memory-mapped files, particularly when the data being processed is greater than the available system memory.

Comrade

Comrade is a Bayesian differentiable modular modeling framework for use with very long baseline interferometry. The goal is to easily combine and modify a set of primitive models to construct complicated source structures. Comrade performs the analysis in visibility space, instead of Fourier space. Comrade will be modified to analyze all interferometry data.

Comrade is used for the analysis of the Event Horizon Telescope (EHT) data of the black holes in M87 and Sgr A*. (See M87 image at right.)



Summary and Future Developments

AURIS is still in the early stages of development and is therefore not production ready. Data access packages are mostly complete and development is focusing on implementing internal data structures, simulation software, and calibration pipelines for ALMA, VLA, and EHT data. AURIS is intended to be a general-purpose software suite that can be used by all radio observatories by adding associated packages.

We encourage the astronomical community to provide feedback, suggestions, and bug reports. We welcome collaboration from the radio astronomy community. Biweekly video calls are on Wednesday, 14:00 UTC. The next meeting is 2026 November 25.

AURIS is part of the JuliaAstro project. A community effort to develop high-performance software to meet the current and future needs of astronomy.

Repository: https://github.com/JuliaAstro/AURIS.jl.