

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY
HAYSTACK OBSERVATORY**

WESTFORD, MASSACHUSETTS 01886-1299

September 3, 2001

*Phone: (978) 692-4764
Fax : (781) 981-0590*

To: LOFAR Group
From: Colin Lonsdale
Subject: LOFAR simulator design

This memo describes the goals and overall design philosophy of the LOFAR simulation package. It lays out information which can be used to guide software design decisions, and establish priorities for development in our manpower and resource-limited situation.

Goals

The LOFAR simulation package is being developed for the following reasons, in priority order:

1. To create simulated LOFAR datasets for the purpose of exploring the performance characteristics of a variety of LOFAR design parameters.
2. To generate datasets as input to test algorithms for LOFAR calibration and data reduction.
3. To provide an environment in which other next-generation imaging interferometer arrays, most notably SKA, can be simulated with minor modifications

The simulator does *not* include the data analysis and algorithm development environment. Algorithm development is a separate activity. A given simulation is considered complete when the resulting dataset is successfully imported into a separate analysis and algorithm development environment.

Note that goals 1 and 2 are not independent. Performance cannot be comprehensively evaluated without an understanding of the algorithms which will be used for data analysis.

Philosophy of operation

There are many performance-related variables involved in the design of LOFAR. These include, but are not limited to:

- Layout and number of antennas within a station
- Layout and number of stations within the array
- Geographical location of the array center
- Instantaneous bandwidth, spanned bandwidth, and distribution of bandwidth

- Nulling beamformer properties

There are also diverse observing conditions for which we are interested in optimizing performance. These include but are not limited to:

- A range of observing frequencies
- Imaging with subarrays
- Observations of varying durations, from seconds to days
- Spectral line observing
- Various declinations and galactic coordinates, in conjunction with array latitude
- A range of ionospheric conditions

In general, then, we are defining a multidimensional parameter space within which we would like to evaluate the performance of LOFAR. It will be impractical to perform an exhaustive exploration of this parameter space, and instead we will have to search for and recognize trends, make consequently educated guesses about what regions of the space are likely to be most interesting, and attempt to iterate to useful conclusions. Such a process cannot reasonably be automated.

Furthermore, for a variety of reasons, simulations cannot be fully realistic. This implies that the interpretation of simulation results will be non-trivial, and will draw heavily upon the knowledge and experience of astronomers well-versed in the intricacies of radio interferometric observation and data analysis. This also dictates that the parameter space exploration cannot be easily automated, since readily calculable performance metrics are not the whole story.

A basic requirement for the simulator is therefore that the execution of a simulation for any chosen set of parameters be straightforward, and amenable to a high level of human control and input. At the same time, the operation of the simulator must lend itself to a limited degree of automation, to relieve drudgery when the work calls for simulations over a range of one or more parameters.

The simulator architecture can be described as follows:

The simulation package will be designed to take advantage of the Unix shell, and standard, powerful Unix utilities. Various parts of the simulator will be implemented as standalone programs, reading from and writing to disk files or pipes, with options governed by command-line flags. In general, each program will not care where its input comes from, or where its output goes, and instead will care only about the format of these inputs and outputs. Simulations will be facilitated by a master shell script¹, which will call other scripts, programs and Unix utilities in the order and with the arguments that lead to a complete, simulated dataset. The master script will manage intermediate data files, and provide a compact and understandable user interface. It must also accommodate

¹ An example of such an approach can be found within HOPS. This is the “mmreduce” script for analyzing millimeter VLBI data, which ties together several HOPS programs, and handles many intermediate data files in the A-file format. At any time, these intermediate files can be manipulated and edited using standard HOPS utilities, and the mmreduce script can then be restarted. This architecture offers a combination of flexibility and automation appropriate to the simulation task at hand.

looping over various parameters in order to provide a useful degree of automation. The script will support stopping and restarting of the simulation process at any stage.

A key advantage of this architecture is that the components of the simulator (programs, scripts) remain simple in function, and are self-contained. They can be developed and tested in isolation. This facilitates the development of multiple versions of a module by different people, in order to test diverse options and approaches. A good example of this would be a module which creates a 2-D station layout for the array. Rather than create a family of subroutines, one for each class of layout featuring independent sets of parameters (star, log-spiral, random, fractal etc.), which must be integrated into a larger program, one instead implements several small, standalone programs which write out simple ascii lists of station coordinates. This lowers the threshold for programmers wishing to contribute to the simulator, and enhances flexibility with no loss of capability. One can even hand-edit such lists to test specific, custom configurations if desired. Care need only be taken that the computational efficiency loss inevitably associated with this approach is small. This is anticipated to be the case for the simulator, due to the large amount of number crunching involved in convolutions and Fourier transforms. At issue are the locations and formats of the data file interfaces between modules, which govern file sizes, I/O requirements, and formatting/parsing loads.

Implementation

The following grab-bag of guidelines reflects the wisdom (or otherwise) of the author as of September 2001, and are subject to revision and addition as needed. The object of these guidelines is to maximize the accessibility of simulator components to LOFAR partners, and to potential new members of the Haystack team, throughout the life of the project.

- Compiled programs developed from scratch will be written in C, and must compile and run under Linux. Pre-existing code imported into the simulator may be in any language (most likely Fortran or C++, if not C).
- Intermediate data products in binary format are discouraged in favor of ascii formats, but if binary formats are used, they should be stored in standard IEEE double or single precision floating point, or 16/32 bit signed integer format. We will ignore endian issues, and work around them if they arise.
- Shell scripts may be written in any shell flavor, with the appropriate flavor indicated in the script via the usual `#!/bin/?sh` first line mechanism.
- “Standard” unix utilities may be freely used in shell scripts. These include awk, perl, sed, and so on. Do not use utilities that are not generally available for Linux.
- If plotting functions are to be implemented from within compiled code, the PGPLOT subroutine library should be used.
- Documentation is required for all simulator components, but is most critical for the format of intermediate data products, which constitute the interfaces between simulator modules.

Architecture

Figure 1 shows the general organization of the simulation and algorithm development activities for LOFAR. The diagram is intended to illustrate the large-scale flow of information, and to yield a high-level understanding of the adopted approach. The intent is to place the simulator effort in context. The diagram does not represent a complete description of all information paths. Detailed contents of, and interrelationships between, the various boxes should be the subject of subsequent LOFAR memos.

Schedule

Based upon the statement of work provided to the NSF in August 2001, and subject to modification by the demands of the International Project Office, the following general milestones and target dates for simulator development are in effect:

1. Selection of architecture and methodology for simulator construction – October 1, 2001
(*subject of this memo*)
2. Export to AIPS++ of basic simulated visibility data, including receiver noise and position-independent ionospheric phase delays (requires ASTRON collaboration) – May 1, 2002
3. Creation of mechanisms (e.g appropriate data flow paths and stubbed subroutines) for incorporating full range of corrupting effects – August 1, 2002
4. Acquisition of initial parallel computing hardware – August 1, 2002
5. Develop comprehensive range of simulated sky brightness distributions – November 1, 2002
6. Perform initial suite of simulations for PDR – December 1, 2002
7. Develop full range of data corruption capabilities – August 2002 to August 2003
8. Finish installation of high performance computing cluster – Summer 2003
9. Full range of simulations, and assessment of LOFAR design impacts for CDR – February 1, 2004

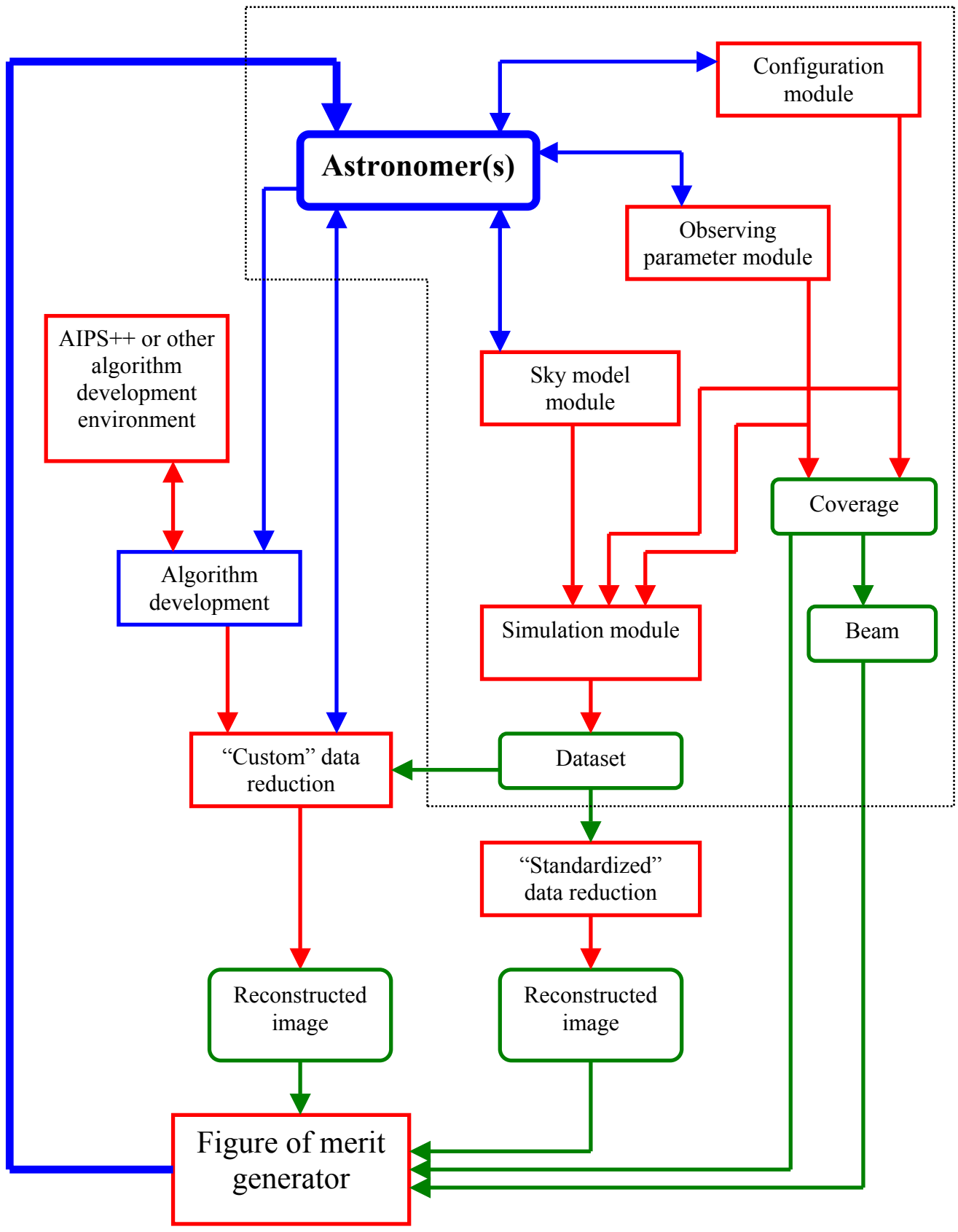


Figure 1. Overall simulation and algorithm development architecture. Blue arrows and boxes indicate interactions with the users. Red boxes indicate software modules of various kinds. Green boxes indicate data-like products of various kinds. The dashed line encloses the functionality addressed by the simulator alone. Only major, direct dependencies are shown. In general, users must be able to provide input to and receive feedback from all red boxes.