Jodrell Bank User Requirements for AIPS++

D.L. Shone (editor[1])
University of Manchester
Nuffield Radio Astronomy Laboratories
Jodrell Bank
Macclesfield, Cheshire, SK11 9DL
United Kingdom

1st January 1992

1 Introduction & Background

AIPS has been important to U.K. radio-synthesis astronomy for many years, and although major software systems have been developed locally (*e.g.* OLAF at Jodrell Bank) which are particularly well suited to local instruments such as MERLIN, it became clear that the primary reduction package in the long term should be AIPS, with vital local tasks incorporated. There were several reasons for this:

- Most image analysis is already carried-out in AIPS, even if the images have been formed in another system.
- Insufficient local manpower to maintain local systems as well as developing new applications; a plan to develop "OLAF 2" was abandoned for this reason.
- AIPS is a global standard, which many astronomers are familiar with. If MERLIN data is to be exported for analysis, adequate software must be available at the remote users institution.
- Most new software developments for synthesis imaging first appear in AIPS.

[1]This document has been prepared after consultation with members of the Jodrell Bank, U.K. and European VLBI user community. I am particularly grateful to Roger Noble, Paddy Leahy, Martin Shepherd, Michael Garrett, Peter Wilkinson and Walter Alef for useful discussions and/or contributions.

However, AIPS is by no means ideal, particularly for interferometer data with comparatively few baselines, and there is little for the single-dish observer. Thus the AIPS++ project is seen as a golden opportunity to achieve the best of all worlds.

Whilst the initial contribution to the project comes solely from Jodrell Bank, there is increasing interest from other parts of the U.K. community, including MRAO (Cambridge) and the JCMT millimetre-wave telescope (Hawaii). At present, manpower constraints preclude full participation for these groups; however, the STARLINK radio astronomy special interest group hope to provide an additional 1.5-2 man-years of effort in the second half of 1992, if funding permits. In addition, Jodrell is member of the European VLBI Network (EVN), and is committed to providing software support for the analysis of EVN data.

Consequently, our user requirements emphasise, first and foremost, constraints imposed by the radio instruments available to the U.K./EVN community. Thus we have to consider not only MERLIN and the Lovell Telescope (as well as other consortium instruments), but also VLBI (and the European VLBI network in particular), MRAO and the JCMT. By catering for a variety of instruments from the beginning, we believe AIPS++ will have the generality required to adapt to new systems far into the future.

Many issues (such as the user interface) are not considered in great depth. This is not because we regard them as unimportant; rather we are aware of the extensive consideration given by others (such as the NRAO group), and believe only a little more has to be added. This document should perhaps be seen as a supplement to the more extensive user requirements produced by NRAO (or maybe those requirements should be considered to be "inherited"). We assume that AIPS++ will have all the functionality of AIPS++, and we suggest ways in which this can be improved.

2 Instrumental Constraints

Here, we indicate generalities which are required to support the data we expect.

2.1 Single Dish

Single dish data are likely to fall into one of three categories:

Spectral line - Series of spectra, not necessarily of equal numbers of channels.

Pulsar - Series of Pulsar profiles - essentially series of time series.

Imagining Collections of spot measurements with position, not necessarily regularly spaced.

No assumptions should be made on the basis of a particular kind of telescope or instrumental geometry, for example, the SCUBA multi-beam array detector on the JCMT will not rotate, and will therefore generate a rotating image. Whilst this is likely to be corrected prior to data export, such data processing problems should be borne in mind when the data system is designed.

In some cases, particularly Pulsar data, there may be a requirement in future to support data which take the form of bit fields

2.2 Interferometer arrays

Our interferometer arrays are generally inhomogeneous in many ways, such as antenna size, receiver temperatures, polarisation characteristics. AIPS++ should not merely be able to cope with data from such systems, it must exploit it's characteristics whenever possible. For example, it should be possible to take full advantage of the case where a very large telescope is part of an array of smaller telescopes, and this will particularly affect calibration.

New observing modes will generate data that can no longer be thought of as streams of the same basic block generated at regular intervals. Visibilities from different baselines, frequencies and polarisations may occur at different times, and the number of any of these present may vary during an observation, thus, for example, assuming that a fixed number of baselines will be present throughout an observation will be wasteful of storage space if a maximum fixed number has to be assumed for the whole observation.

Interferometer data involving few baselines, such as from MERLIN or VLBI, are likely to benefit from a number of operations which are not generally applied to VLA or similar data:

- Calibration of data from tables of system temperature and gain *v.* elevation, and generation of such tables from raw data.

- Flagging of data on the basis of observing session logs.

- Model-fitting (with errors) to visibility data, rather than images.
- Self-calibration using a model derived from fitting to visibilities.
- Calibration and self-calibration with constraints from crossing-points in the uv-plane.
- Fringe fitting by baseline, rather than globally.
- Simultaneous processing of "associated" data sets, such as different (*e.g.* by calibration, integration time, fringe-fitting *etc.*) versions of the same observation, so that the best can be selected later.

2.3 Combining Data

Much useful work has already been done in combining data from different instruments, such as MERLIN, with VLA or VLBI data. This is likely to become more commonplace (particularly if the users in the consortium collaborate as much as the software groups!), and must be made easier. AIPS has already acted as an enabling tool for this, but improvements are necessary, such as the ability to transform between different weighting schemes, and "regrid" spectral data for combination where bandwidths and number of channels differ.

3 Data Display and Editing

So far as systems with a few baselines (such as MERLIN and VLBI arrays) are concerned, AIPS has traditionally been weak in displaying and editing data in the most useful way, *i.e.* baseline by baseline. This functionality, available in the CalTech and OLAF systems, is essential. It is also likely to be useful to display and edit data using a variety of "viewing strategies", such as taking arbitrary linear (*e.g.* radial) or circular cuts through the data (gridded or ungridded) on the uv plane. In all cases, it should be possible to superimpose model data, and also subtract or divide by model data, in order to see residual errors more clearly. Some of this is possible in AIPS, but should be made more straightforward to use.

As volumes of data increase, automated techniques for bad data rejection will become important. Some simple statistical techniques already exist, but these often fail to match capabilities of the human eye, and more sophisticated techniques are required, possibly based on neural networks.

Data display and editing should be seen as generic tools, applicable to single dish data as well as interferometer visibilities. To this end, some degree of data visualisation should perhaps be seen as an integral, or at least closely coupled part of the database system. In particular, as well as graphical data display and editing, it should be possible to view the numbers themselves, extract them for other processing, and modify them in the database. The means of doing this is closely tied to the user interface, suffice it to say here that some sort of query language would be one way of achieving this.

4 Image Formation

The title of this section might be construed as covering simply the act of transforming edited, calibrated data and deconvolving the images. However, we wish to emphasise that the process should be viewed as a whole, and whilst data editing, for example, is discussed separately (because it is a generic operation, applicable beyond interferometry), it must be possible to integrate (self-)calibration, data inspection/editing, transformation and deconvolution more closely than is possible in AIPS.

This is what has been achieved in the OLAF system, which has proved particularly valuable for MERLIN and VLBI data. The OLAF MAP program essentially incorporates self-calibration in between major cycles of an MX-like transformation and deconvolution. At each step it is possible to view the current residual image and set windows (as in MX), and, together with the OLAF plot program, the visibility data may be displayed (together with the current clean-component model, or with the latter subtracted) and edited. This sort of thing is, in theory, possible in AIPS, but it is difficult and tedious.

AIPS++ should combine the best of both worlds; instead of a large, monolithic task like OLAF MAP, it should be possible to easily "mix-and-match" self-calibration, transformation, and deconvolution tools, for example, using CLEAN to deconvolve in the early stages, and Maximum Entropy later on when CLEAN would begin to break-down. This also demonstrates the need to make self-calibration use a generic model, which may be CLEAN-components, an image, or a gaussian model.

5 Error Handling

Astronomical data analysis is often neglectful of error analysis and propagation. In radio-interferometric synthesis, the nature of the problem is poorly understood, and consequently is usually ignored. Whilst this strategy has not proved disastrous, there is an increasingly strong desire to quantify the "fidelity" of synthesised images, and the use and propagation of reliable error estimates is vital in advanced image analysis such as the determination of optical depths, rotation measures and other polarisation parameters, and in spectral fitting. Less obviously, the wildly inaccurate error estimates sometimes obtained in gaussian fitting and similar tasks are due at least in part to the fact that the real errors in images are functions of both intensity and intensity gradient, and not simply quantifiable as a constant error applicable to each pixel.

In addition, if AIPS++ is to be used in in other radio-astronomical applications where error handling is slightly more rigorous, the ability to associate errors with measured quantities is essential. Thus if a baseline is to be fitted to a single dish spectrum, it should be possible to make use of errors if they are available.

Support for error handling should be available in all statistical analysis and display tasks, for example:

- Plotting error bars on spectra and profiles;
- Automatic warning if contour levels are below noise;
- Error-based blanking in display of results;
- Properly formatted errors when data are extracted for tabulation.

Ideally errors should be propagated right through the system from the errors or weights on the original raw data. Maximum-entropy algorithms can certainly assign pixel errors on this basis; this cannot yet be done for CLEAN. For this reason and because visibility weights will not include all important systematic errors, we also need tools for generating errors based on statistical analysis of images and on "error models" (*e.g.* assumed Poisson errors for photon-count images).

6 User Interfaces

A key feature of the user interfaces should be that they have a similar relation to AIPS++ as the various shells have to Unix; essentially, you can choose the one you want, and all will have some basic functionality (such as the ability to start tasks), whilst others may have more sophisticated features, such as an interpreter which recognises AIPS++ data structures.

The adoption of this sort of model should also ensure that users have full access to normal operating system commands.

6.1 Tasks as Tools

The interface should allow tasks to be combined together (at the command-line, in a command procedure or graphically) to build a new task. We are aware that others are thinking along similar lines, and generally agree with this "toolkit" approach. It would be useful if all tasks (as in AIPS terminology) could be used as subroutine-like operations from a user command shell. This, together with the ability to select and manipulate data from a command-shell (using something like a structured query language) would allow users to perform any operation in their data without resorting to "real programming".

This should be borne in mind in the design of the data system; efficient means of passing data objects will be essential, and something like the pipe facility in Unix shells seems desirable, although this should be capable of passing complex data structures, and perhaps, in general, true objects, incorporating methods associated with data structures. Such piping should also be possible between tasks running on separate systems in a network, as well as on a single system. Distributed processing will be discussed further later. Any really useful data piping system should have "tee-joints"; *i.e.* it should be possible to tap the dataflow for examination at any "joint in the plumbing", so long as the data are in a meaningful state (*e.g.* a Fourier transform has completed).

7 Hardware and Software Environment

7.1 Operating Systems

It is clear that Unix-like standards will dominate for the forseeable future, and whilst it might be argued that this will stifle development of operating systems, and lead to reliance on a particular software model, the evidence is to the contrary. New operating systems are already available which, on the surface, look like Unix, and often comply with standards such as POSIX.1, but "under the hood", these may be very different beasts, in some cases, well suited to new hardware environments. Mach is an example which will form the basis of one mainstream POSIX-compliant system, OSF-1. Chorus and Amoeba are more radical examples. Thus the Unix-like standards will enable us to tap a rich, developing field in operating systems technology, in much the same way as C++ (and hence C) does for other software.

The most difficult area may be in selection of appropriate standards, since not all POSIX committees have produced a standard yet. The X/Open XPG3 should also be considered, since it is well established, and seems to encompass as least as much as POSIX. However, this is a detail which should be left for consideration by the design group.

7.2 Hardware and Performance Considerations

Given the timescale for development of AIPS++, it is not unreasonable to suggest that the least powerful machine it should ever be require to run on is a SPARCstation IPX/SPARC 2. Even this might be too conservative, given that more powerful machines are already available at similar cost (or even less). This baseline, equivalent to 15-20 times the performance of a VAX-11/780, would rule-out most, if not all, existing VAX systems. This is not to say that AIPS++ might not run on these; we have argued that Unix-like standards should be adopted as a software environment, and it seems likely that these will be available on VAXes.

However, much of the functionality we require may only be possible with the relatively high scalar performance of RISC systems; for example, suppose some valuable facility in a command interpreter results in a SPARCstation IPX-class machine taking 0.3 seconds to return a prompt. This is probably alright, and is likely to improve on future machines; however a MicroVAX 3000-class machine may take several seconds, which is likely to be unacceptable.

Basic functionality, as discussed in this and other requirements specifications, should not be constrained by a need to run on obsolescent hardware. If the additional functionality can be made optional, so that it works if the appropriate hardware is available, then this is satisfactory, and this philosophy may be more widely applicable to applications, such as 3-D visualisation, which may require special purpose hardware.

At the other extreme, AIPS++ must be able to take advantage of many forms of parallelism for the highest performance. By comparison with most major astronomical software systems, AIPS has been quite successful in exploiting the mini-supercomputer systems, such as Alliant and Convex, which appeared during the mid-1980's. This was largely due to the fact that they ran Unix, (hence supporting our earlier arguments for the operating system environment), and also to the way almost all the performance-critical code is encapsulated in the AIPS Q-routines. This limited the amount of work required to take advantage of new vector and parallel hardware. However, the original array-processor paradigm is limited to some extent by the memory model assumed in AIPS, and this must be addressed.

The highest performance machines may be massively parallel SIMD or MIMD machines with distributed memories. It is not yet clear whether distributed memory systems will be made to look like a large global memory in the long term, but AIPS++ should be designed with these problems in mind. This might go hand-in-hand with considerations for networked systems proposed next.

7.3 Networked Systems

Clusters of Unix workstations linked via ethernet are already common. Internet Protocal (IP) seems likely to dominate the software connection for some time, although the impact of OSI standards should be considered.

Over the next few years, most significant hardware changes are likely to be transparent to the software, taking the form of performance improvements with fibre-optic (FDDI) links replacing ethernet, and High-Performance Parallel Interface (HIPPI) links will make it possible to couple computers intimately enough to distribute an application in a way analogous to the distribution of functions in AIPS among a basic host (such as a VAX) and array and display processors. The difference which AIPS++ should be able to exploit is that each component may be a fully functional computer in it's own right, differentiated only by an ability to perform a particular operation (such as array operations or 3-D display) particularly well.

AIPS++ should be able to make better use of networks than AIPS does in a number of ways:

Data serving - AIPS file serving relies on NFS (Sun's Network File System) for accessing remote files over a network. This can be quite detrimental to performance, and, although remote access will be limited by the hardware interconnection, it should be possible to do better, particulary if the problem of slow NFS-writes can be circumvented (deferred writes are possible with some NFS servers). An asynchronous data server system, making greater use of file-server intelligence within the AIPS++ database system is likely to be desirable to make best use of networks, and could also take advantage of multi-processor systems, with truly asynchronous I/O.

Intelligent display servers - AIPS remote-display capabilities rely on the client system for most computation associated with display. AIPS++ should be able to run display applications entirely on the server where possible. This would be particulary useful in the of widely-separated systems (such as the case when using a remote supercomputer). Of course, this does not preclude the use of X-Windows; a display server might, itself, consist of separate, networked systems, if appropriate.

Distributed Processing - AIPS++ should be capable of distributed either by running different tasks on different machines (and piping them together, if necessary), or by distributing different parts of a single task. In both cases, it might be useful to make the selection at run time, with the ability to set defaults in system-wide and personal initialisation files

8 Miscellaneous General Requirements

History and Session Logs - It is essential that a user may be able to determine what has happened to a dataset, and support should be available to import history information from "previous incarnations", such as on-line observing-system logs.

Archives - AIPS++ should be able to support data archives comprised of many raw and/or processed datasets, with a standard means of transporting such archives. There should be support for searching for, and retrieving a desired dataset.

Electronic Publication - Given the kind of archive facility described above and the hypertext documentation system described by others (*e.g.* the NRAO group), a useful long term goal would be to go beyond the objective of "publication-quality" output; there is clearly a mismatch between what can be produced at the end of an astronomical project, and what can be published. FITS has done a great deal to assist the transportation of astronomical data; AIPS++ could do the same for astronomical *information.*