AT 46.3/021
December 29, 1991

ATNF AIPS++ User Specifications

ATNF Staff and Users

Australia Telescope National Facility[1]
PO Box 76,
Epping, NSW 2121
Australia

1 Introduction

This user specification is intended to reflect the views of ATNF staff and AT users, and to include features of AIPS++ which will be needed for compatibility with the Australia Telescope (AT). It is not intended to be a full AIPS++ user specification, and does not include features which are so obviously necessary that they are certain to be in any full specification for AIPS++.

In this document, requested features range from the mundane to the exotic, and include functionality which probably cannot be realised by currently available hardware. It is assumed that in merging the specifications from different observatories, some priority order will be placed on the different requirements, so that the more exotic requirements may end up as being listed as future developments. The only relevance to the initial design might be that the appropriate hooks should be included so that these features can be added at a later date. Items which are recognised as falling into this category, or which are regarded as having lower priority, are bracketed by [ ].

2 Scope

AIPS++ should, as a minimum, include all the useful functionality of AIPS. AIPS++ should be able to process any radioastronomical data, including single dish and VLBI data. In addition, hooks should be provided for optical aperture synthesis work.

We require AIPS++ to replace AIPS totally within 3 years. If timescales appear to be slipping, then we would prefer to see a basic AIPS++ with limited capability (i.e. mainstream AIPS functionality) released within that time, rather than waiting longer for all functionality to be implemented.

3 Hardware Support

AIPS++ can be restricted to run on Posix machines and in a X-Windows environment. However, to ease possible future migration to other operating systems and windowed environments, operating system and window environment dependencies must be kept to a minimum, and confined to a well-defined set of modules/classes (much like the current AIPS Z and Y routines).

However there must be support for dumb ASCII terminals and for simple graphics terminals (e.g. Tektronix) as well as windowed environments. A user should be capable of performing

[1]The Australia Telescope National Facility is operated in association with the Division of Radiophysics by CSIRO.

the full data reduction/analysis procedures without needing a windowed environment (though with some loss of ease).

Network support should be transparent, so that the number crunching, disk storage, tape storage, display, printer output, etc, can be done on different machines. Though most of these issues are being addressed by the vendors (e.g. NFS and X-Windows), some are not (e.g. remote tape access).

[In the long term, distributed computing functionality should be implemented, so `AIPS++` will decide which of the available machines on the network will best handle a particular operation, thus making use of specialised hardware where available.]

A failing of many current packages is load-time overheads for reading application executables or images from disk. This can make wall-clock time vastly exceed CPU time when doing relatively simple, but repeated, data manipulation. This may be remedied in a number of ways, and we leave it to the implementers to solve this problem (e.g. with an image stack and an application stack).

4 User Interface

4.1 General

The user should have a choice concerning the user interface that he/she uses. User interfaces should include:

- Issuing commands directly to the host system command interpreter. Other user interfaces should be seen as being built on this basic user interface.
- An `AIPS++` shell, intended for use from ASCII terminals and for interpreting procedures. This Command Line Interface (CLI) is discussed further below.
- A modern window-oriented graphical interface, with pull-down menus, etc. This Graphical user Interface (GUI) is discussed further below.

In each case, there should be a high degree of integration between the display and the application (eg fast, windowed data editors).

Interfaces should draw a distinction between required and optional parameters. In all cases, we prefer that such 'adverbs' not be global across applications, unless specifically requested (e.g. turning on calibration for all applications). There should be consistency between adverb names across applications, and it should be possible to transfer (e.g. cut-and-paste) adverbs between applications.

All applications should check input parameters for inconsistencies or unusual or dangerous combinations, and warn the user before execution.

The various interfaces should have a similar "look and feel" on all machines.

The ability to 'undo' the previous operation (e.g. data flagging) should be included.

[For the benefit of remote users accessing `AIPS++` remotely over slow terminal lines, it would be useful to provide a user interface running under DOS-Windows and/or Mac-OS that generates the appropriate line commands.]

[It should be recognised that within the lifetime of `AIPS++`, input to the package may be by voice or pen-pad, and hooks, where known, should be provided as appropriate.]

4.2 Graphical User Interface (GUI)

The GUI should be a modern window-oriented graphical interface, with pull-down menus and multiple windows.

In cases where some parameters depend on others (e.g. specifying calibration parameters only if calibration is to be applied), this may best be achieved by popping up a sub-menu as appropriate.

4.3 Command Line Interface (CLI)

This would be the equivalent of the `AIPS` POPS interpreter. This interface should be programmable, in that it supports variables, conditional statements, do-loops, procedures, etc. Procedures should be interpreted (i.e. no need to compile, as POPS currently requires). [For complex procedures, a compiler may be provided at a later stage.] It should be possible to issue host commands from within this shell. Image arithmetic and operations might also be performed by this shell. Both a beginner mode and an advanced mode should be available. For example, where appropriate, it should be possible for novices to run applications in a 'prompt' mode, where the user is asked in plain English what he/she wants to do.

The command-line interface should allow users to set up a user-definable aliases file allowing users to define their own commands and macros. It would be useful if such macros could be 'recorded' from the GUI, in a similar way to that used by several commercial PC packages. The 'batch' system should also be integrated into this system, allowing users to submit scripts from a dumb terminal.

Flexible command-line recall and editing should also be provided.

Data files should be visible through via the machine operating system, and be able to be read and archived independently of `AIPS`. The user should be capable of of using any of the host systems capabilities (e.g. directory structure) and utilities (compression, tar, etc) to store and manage his/her data as he/she sees fit.

4.4 Documentation

We are all agreed that documentation must not be a last minute add-on, but must be a planned part of the `AIPS++` development. We see several levels of user documentation (not including software and technical documentation) as including:

- COOKBOOK (one of the strengths of `AIPS`)
- Application documentation
- On-line and context-sensitive help.

All documentation, including tutorial and cookbook material, should be available on-line. Hypertext may well be an effective way of handling all documentation.

Although application documentation may be written in draft by programmers, the final result must be written by people with the skills, training, and aptitude to write good user documentation. We encourage the `AIPS++` team to find suitable people from within their own organisations and train them appropriately. In addition, uniformity of style, regardless of the application's originating institution, is important, and so there will need to be one central documentation editor.

The present level of AIPS application documentation is inadequate in that it is often unclear exactly what an application is doing. Application documentation must include a complete specification of algorithms used by the application, and a specification of exactly what it does to the data.

Help should be context-sensitive, though again with the ability to rapidly search on keywords or names which are out of context.

5 Visualization and Presentation

The data structure should be flexible enough to allow data to be visualised in non-standard ways. For example, a user may wish to plot calibrator amplitude against telescope elevation, rather than against time or u-v distance; or plot u-v data as a 2-dimensional image (u vs. v with amplitude modulating the pixel intensities).

Multiple image windows should be allowed, and the distinction between line graphics and image windows should be removed. Multiple overlaying of images within windows should be allowed, using different look-up tables if the hardware allows.

It must be possible to present high quality diagrams, with labelling etc. This might best be achieved by providing easy transfer to other packages such as Publisher, Harvard Graphics, or MacDraft.

After choosing labelling, look-up-tables, etc., it should be possible to store the finished image so that it can be recalled again with the same appearance.

One particular area where AIPS is weak is in the overlaying of images from other observatories. For example, digitised data from all-sky Schmidt surveys is now readily available in the form of positions, identifications, isophotal shapes and magnitudes. AIPS can only plot uniformly sized crosses in overlay planes. Handling of overlay planes themselves needs improving over the current implementation on Sun workstations.

A user must be able to interact with and edit his/her data much more effectively than current AIPS routines such as TVFLG, SPFLG, POSSM, IBLED, etc. An application to do this (and replace all these four named tasks) might appear as follows. The user might display his/her u-v data averaged over a number of channels in a way similar to TVFLG. He/she then clicks on a data point, or clicks and drags to select a rectangle of data, at which point a pull-down menu invites him/her to look at the spectrum, plot amplitude against time, edit the data, etc.

Alternatively, he/she might move a 'magnifying glass' around the data, and get a magnified view appearing in a separate window. As a further alternative, he/she opens another window to display spectral information, and gets a grey-scale display (like SPFLG) corresponding to the region of data selected by his/her click-and-drag in the 'TVFLG window'.

Such applications might have have individual tasks represented as icons, and the passage of data through them represented as lines drawn by the user. Such visual representation of pipelined algorithms is already available in some packages (e.g. APE).

Research is currently being done at the AT and elsewhere on using advanced visualisation engines to provide rapid interactive visualisation to data cubes. Such cubes are composed of voxels distinguished not only by colour and brightness but also by opacity, and ray tracing then gives efficient visual cues to the user. These cues are enhanced by techniques such as interactively manipulating the cube, and displaying it stereoscopically. While such facilities are at present available only on specialised hardware, the increasing power of workstations means that such facilities should be accessible to all users within a few years, and AIPS++ must be capable of supporting such concepts.

6 Data Formats

6.1 Internal Data Structures

AIPS++ should not have a distinction between 'multi-source' and 'single-source' files. Applications should apply calibration/flagging, etc, on the fly. It should not be necessary to generate a copy of a u-v data-set in the entire reduction process. Similarly, applications should not require that u-v data be sorted in a particular order (TB or XY), or that data (hyper-)cubes be sorted in a particular order (e.g. velocity/RA/DEC). The application program may dictate the order it requires, but the mechanism of that ordering should be hidden from the user. There is no computational advantage on most modern machines with pre-sorted data as, for smaller datasets, the entire dataset can usually be fitted into memory. For larger data sets, we urge the design team to consider ways of getting this efficiency without involving the user.

Under certain circumstances (e.g. reducing the data set size or filtering out extraneous sources), it may be convenient for the user to apply the calibration or sorting the data once and for all. Therefore, there should remain the ability to split off, or to sort data sets. However, doing so must not be required by applications.

User-editing of header parameters (coordinates, projections, etc.) or tables should be made easy using a proper full-screen editing application. Rather than inventing yet another editor, this may best be done by sending the data out to the editor of the users choice and letting him/her edit the data before replacing it in AIPS++.

Data with non-regular increments in frequency (eg AOS data; data in optical velocity convention) or RA, DEC (scanned plates) should be supported. Hierarchical data structures can support this easily, but it is important to get the structure right to begin with. The data format should be machine independent.

Non RA-DEC coordinate systems should be supported, and sufficiently precise definitions of terms and storage of ephemeris information, to allow AIPS++ to be used in astrometric applications.

It should be possible to store closure phase explicitly, both for VLBI and optical aperture synthesis.

It should be possible to store complex images explicitly, rather than as 'REAL' and 'IMAG' images as in present AIPS.

6.2 Data Import/Export

Image plane data from any observatory should be supported, and not require the presence of specific data headers. FITS input/output should be the main data transfer route, but FITS readers should be able to cope with generic FITS rather than the subset supported by AIPS.

Simple one or two dimensional ASCII files should also be able to be both imported and exported, and support should be provided for import/export to/from other astronomical packages.

U-V data from all existing radio telescopes should be supported. The data input routines should be general enough to read related data such as from optical interferometers where the photons may be time, position and detector-tagged. This data may come in a non baseline-dependent manner: eg in the form of triple correlations. The FITS readers/writers should not lock out such data by, for example, demanding that PTYPE1=UU, etc.

Applications which generate listings of data should consider the needs of users who want to export their data (as ASCII tables) to spreadsheet and database applications. For example, headings and comments in generated lists should be prefixed by a # character, to allow these to be easily stripped out, or treated as comments.

Tasks should be provided explicitly to allow transfer of data and tables to common packages such as Excel, 1-2-3, and dBase.

7 Applications

7.1 General

As a starting point, we require all the functionality of `AIPS`, but recognise that many of these are now obsolete and their functions can be handled in better ways than a specific application. In addition, there are newer requirements, such as mosaicing and multi-frequency synthesis, which must also be catered for.

There should be more image analysis and display applications than are supported by `AIPS` (eg median filtering; arbitrary cuts through cubes). Fourier transforms should be more generic (ie not tied to powers of 2, or to specific, or any, coordinate systems). Mosaicing should be well-supported from early on and enough single-dish software to enable the easy interfacing to interferometer data.

Production of faked u-v data should be possible starting with a user-defined model or image, together with knowledge of instrumental parameters. Thus a user, when preparing his/her observing application, will be able to try out configurations and see what result he/she would expect (e.g. If my source looks like Cen-A but at z=0.1, what will I get if I do a 10-minute snapshot with a given array at a given frequency ?)

Another new type of application comes under the general heading of "Interactive Hypothesis Testing," in which a user might "paint" an area of a map to remove a suspected artefact, or introduce a suspected source, and then see whether the resulting image is consistent with the data.

A tree-like application structure is preferred (ie applications falling into certain classes like image manipulation) with the proviso that rapid application searches are provided so that applications in other classes can readily be located by name or keyword.

7.2 Single Dish Applications

Single dish data are vital for mosaicing of AT data and should therefore be supported, preferably using the same internal format as for interferometer data. Similarly, telescopes like the AT can output autocorrelations in the same way, at the same time, and in the same format, as cross-correlations, and `AIPS++` must handle these data properly.

`AIPS++` must also support single-dish applications in their own right. As well as being used in conjunction with interferometry data, applications must contain the functionality of existing single-dish programs, and in particular those committed to spectral-line data. However, we recognise that some users will want only the single-dish spectral-line part of the package, and so it must be possible to construct a stripped-down version of `AIPS++` containing only those parts useful to a single-dish spectral-line astronomer for installation on his (Posix-compliant) PC.

In addition, such applications must have the rapid interactive 'feel' currently afforded by existing single- dish programs and lacking from `AIPS`.

Support of large-area (eg all sky) image data is requested. Wrap-round contouring is implemented in plot packages such as PGPLOT.

7.3 VLBI

Applications must provide full VLBI support. This must include not only VLBA correlator formats, but also MkII (which will continue to be a cost-effective medium for smaller groups, especially for spectral-line data, for many years), S-2, K-4, etc.

Modelling tools should be provided for sparse data, and will be especially important for Space VLBI.

More robust global fringe-fitting and Selfcal, such as in OLAF, are needed.

[It would be useful to include astrometry and geodesy software, such as JPL's Masterfit.]

7.4 Data Reduction Methodologies

Data reduction would be made simpler by the grouping together of trivial applications. This would shorten the length of the data 'reduction' pipeline. For the simplest type of data, all that is required is reading, editing, calibration and imaging. Present procedures for the simplest of AT data involve the use of at least 8 separate tasks. The 'inputs' to these groups should be amenable to beginners and advanced users with sufficient information given to the user about what applications are actually doing to the data.

[Consideration should be given to the implementation of 'real-time' mapping which will be possible, even for large data sets, with future generations of computers. Ideally, a user would like to see u-v data displayed in one window, and the resultant image in another window, with a FFT-MAP-CLEAN-SELFCAL pipeline running continuously between them. Thus the user can edit his/her data in one window and see the resultant image a few seconds later in the other. Alternatively, a calibration icon may be double-clicked, a parameter tweaked, and the effect seen in real time.]

8 Programmability

It is important to have good user-control of the data through the command level or, for more complex operations, through template routines. Efficiency may be sacrificed for ease of use here. `AIPS` is poor in this area, for example the `AIPS` template image reading routine only allows the user to access 1 row of data at a time!

Users should have the ability to develop their own applications in some shell language, as mentioned under the User-Interface section. In the case of simple arithmetic, this might consist of a single line such as "C = A + sqrt(B)," where A,B, and C are all images.

As with UNIX, there should be no distinction (as far as the user is concerned) between an application written in the shell language and an application written in a compiled language (C++). That is, both the shell language application and the C++ application should have the same user interface, and the ability to retrieve help information.

It should be possible for programmers/users to develop their own personal code. This would reside in their personal directories. The `AIPS++` user interfaces should treat such applications as it treats and standard `AIPS++` applications (e.g. the application is initiated in the same way, and the same sorts of help are potentially available). It should be possible for the programmer/user to be able to override an existing `AIPS++` application with their personal version (e.g. for debugging or development purposes). Probably there would be a search-path mechanism.

Although C++ may be the language of the future, it is presently a very minor language in astronomical applications. Until such time as the user community gains experience with C++, `AIPS++` should allow people to create tasks by allowing Fortran subroutines or providing extensive templates. Users should also be able to link in with the subroutine libraries of their choice to facilitate numerical manipulations.

9 Error Handling

Data must be uncorruptable by errors! When error handling is implemented, it should give a true indication of the stage at which the error occurred.

10 AT Technical Requirements

`AIPS++` should:

- support a variable number of channels within each frequency band.
- eliminate the distinction between processing frequencies by "FREQI" or using IF axis.
- support correlators which can observe multiple lines simultaneously (e.g. multiple rest frequencies).
- provide full support for linear and circular polarisations.
- support rapid time-switched mosaicing.
- support mosaiced observations with many ($\sim$1000) pointing centres, both at the u-v dataset and image dataset level.
- support rapid time-switched frequencies.
- support time-switched polarisation measurement (i.e. for observatories which do not correlate all 4 polarisation parameters simultaneously).
- support interferometers with focal plane arrays (e.g. multiple simultaneous pointings).
- support different integration times on different baselines.
- provide clean frequency and velocity handling, including full support for the standard extragalactic 'optical' convention where equal velocity increments do not correspond to equal frequency increments.
- allow single-dish data-set formats to be the same as interferometer u-v data-sets.
- allow sufficiently precise definitions of terms, and storage of ephemeris information, to allow `AIPS++` to be used in astrometric applications.
- allow programs to process multiple input u-v data-sets (e.g. less need to concatenate different u-v data-sets together).
- provide flexible u-v data selection mechanism.

- provide on-the-fly calibration (antenna gains, passband, baseline-dependent offsets and gains, polarisation leakage).
- provide on-the-fly polarisation conversion (e.g. convert raw linear or circular to Stokes).
- allow reversible flagging.
- have no single source/multi-source u-v data-set distinction.
- allow easy storage of ancillary/monitor/meterological data with the u-v data-set.
- provide the ability to flag correlations dependent on ancillary/monitor/meteorological data.
- provide support for 16-bit scaled integers and 32- bit floating point correlation data.
- have a machine independent data format.
- allow different Stokes/polarisation parameters in a single image data-set.
- allow arbitrary region-of-interest (e.g. multiple polygonal regions of interest, as well as boxes).
- support pulsar synthesis data, in which pulsar bin number is one axis of the data hypercube.
- allow complex and double precision pixel values.
- allow data (hyper)cubes to be read in "arbitrary order". That is, the the physical ordering on disk is hidden. Cubes may be stored with dimensions RA/DEC/velocity, or velocity/RA/DEC, etc. The application program would dictate the order it requires (i.e. it would indicate that it wants to read velocity/RA planes, or velocity profiles, or whatever. This all amounts to including the AIPS task "TRANS" within the image class.