December 31, 1991

Miscellaneous Suggestions for AIPS++

R.M.Hjellming et. al.

National Radio Astronomy Observatory[1]

1 Introduction

During the course of work on AIPS++ specifications in August-December 1991, and particularly in response to the invitation to provide material for the AIPS++ User Specification Memo series, many people sent in short statements expressing their views. Most of this were good suggestions, or comments reflecting the views of many users, that were not in a form suitable to a separate item in the memo series. This document is an attempt to include as many as possible of these contributions. In some cases I have added comments that reflect my own opinions and do not necessarily reflect any official view on AIPS++. In some cases I have identified the author by name. I have done some editing based upon my best understanding of the writers intent. In all cases, my own comments, and editing by me, are put in italics. Original material, mostly taken from E-mail messages, are in normal text.

2

I wonder if you would be kind enough to put my name on the AIPS++ Memo Distribution List, please ? Thanks. Good luck with the mammoth task you have taken on in respect of AIPS++. Chris Salter.

I received more than a hundred requests for receiving the memo series. I particularly appreciated those with this sentiment as first expressed by Chris.

3

I'd like to receive the AIPS++ User Specifications Memos Series. At some point soon, I'll be happy to make a contribution of my ideas about how AIPS++ should look. *This was far too common. People expressing a hope that they then did not have time to fill, at least not yet.*

4

Please add me to the AIPS++ user specification memo series. Can I get the back issues of the ones that have been distributed thus far?

PS: Have you considered distributing them electronically? Presumably they're in TeX format. I suspect that most of the recipients have both TeX and e-mail capabilities. It would save you postage, not to mention the hassle of stuffing envelopes. If you do send the AIPS++ memos via e-mail, you can use the address below. Thanks.

This one gives me chance to explain that, although the majority of those expressing interest in AIPS++ fit these criteria, many are not in the habit of using E-mail, and people prefer to

[1] The National Radio Astronomy Observatory is operated by Associated Universities, Inc. under a cooperative agreement with the National Science Foundation.

write with a word processor or non-TEX text formatter. For this reason the official memos are distributed in paper.

5

Please find below my suggestion as a user for the new AIPS++ system:

(AIPS-IRAF) Many astronomers work both in radio and optical/IR. It would be helpful, if AIPS and IRAF are installed in a unified reduction system, "AIPAF" for example. With the AIPAF, one should be able to open two graphic window systems, one for AIPS and one for IRAF, on the same screen. He/she chooses either (1) AIPS mode, (2) IRAF mode, or (3) AIPAF mode. In the (3)rd mode, one may overlay, compare, etc., radio and optical data.

(File names) The present AIPS creates many many files, which cannot be identified and are mostly meaningless for the users. It may be convenient if the files are named by object names, etc. Yoshiaki Sofue

6

A program written to be compatible with AIPS (e.g., using the template TAFFY) should not have to require modification in the new system (or have very few changes).

This will probably be not possible. The improved programmability of AIPS++ is likely to be very incompatible with the old sequence of FORTRAN code which mixes user code and calls on AIPS subroutines in a complicated manner.

7

I will send (in the U.S. mail) copies of the documents on the ISIS system that we have developed for handling imaging spectrometer data. This will include the ISIS User's Manual, the ISIS System Design document (ISD), and the ISIS Programmer's Reference Manual. I will also send a copy of a document that I recently wrote that describes the system update procedures. Most of this would probably be of little interest to you. However, it describes how we automatically extract program descriptions for producing the User's Manual and also how we automatically extract subroutine descriptions for producing the Programmer's Reference Manual.

The Near Infrared Mapping Spectrometer (NIMS) on the Galileo spacecraft is the source of much of the funding for the ISIS development. For the NIMS project, VAX/VMS is the defined operating environment for ISIS. However, when we designed and implemented the so-called "Build 2" system, we expected that there would be a desire to run it in UNIX. Most of the programming environment and non-interactive programs have already been converted to run in UNIX.

I am also enclosing a copy of a paper I wrote that describes the interactive display programs in ISIS. This describes display programs that were part of the "Prototype" version of the system. (The design of the current "Build 2" system began nearly three years ago.) The display programs have been modified to read the Build 2 format cube files, but there is little change in how they appear to the user. Thus, the display programs have had virtually no development effort for about three years. I should point out that the current set of interactive display programs were specifically intended to have maximum interactive response and thus they were not intended to be portable to other operating systems and/or display hardware.

We always recognized a need for less-interactive but highly portable display programs, but we haven't yet gotten around to implementing them.

Now, as we are working on porting to UNIX, portable display programs will be a priority. We are considering many of the same options that you are considering, e.g., IDL and AVS. However, I worry about the efficiency of doing significant processing in IDL. Another option that we are considering for creating the interactive user interface for the display programs is TAE Plus. (By the way, if you want to learn about TAE Plus, the TAE User's Conference will be held Nov. 5-7 in Maryland.)

Another data processing system that you might want to take a look at for ideas is the Spectral Image Processing System (SIPS) produced by the Center for the Study of Earth From Space (CSES) at the University of Colorado. This is another system that is designed for handling imaging spectrometer data such as that produced by the Airborne Visible and Infrared Imaging Spectrometer (AVIRIS). Virtually all of this system is built using the IDL language.

The July 1991 issue of the Advanced Imaging magazine briefly describes yet another imaging spectrometer system. This one is called MIDAS, and was developed by the Analytical Sciences Coproration (TASC) in Reading MA. This system apparently makes use of AVS. I haven't yet found out any additional information on this system.

Happy designing! Jim Torson, USGS, Flagstaff

This illustrates what we all know. Many people are doing the same things in different ways, for different purposes. There are many good systems we could learn from.

8

I have received your letter dated September 25th, about "User Specifications and the User Specifications Memo Series". We are currently using AIPS in a VAX-VMS environment, and from your letter I understand that the AIPS++ version will work only in UNIX environments.

We would appreciate to know in which kind of small computers you are planning to run (and maintain) the AIPS++ version (ie, Spark Stations, etc). This will help us in planning acquisitions of work-stations (or PCs?) for the near future. (At present we do not have any UNIX system.)

Sincerely, Pere Planesas, Centro Astronomico de Yebes, Guadalajara, Spain

This is a very reasonable request indicating that we need to be specific about when portability is likely to be available to which systems.

9

I got your note regarding the AIPS++ user specification memo series. Please keep my name on that list. Also, perhaps I could direct a question or two regarding the selection of C++ from a user's standpoint. I have heard from one local computer scientist that C++ has the following disadvantages: 1) There are actually two definitions of the language currently in use, with no immediate prospect of an IEEE convention being defined, 2) C++ compilers currently on the market produce significantly slower code than, say, C compilers, 3) C++ compilers will probably need to be purchased by most AIPS users for this purpose as they would not already own them for other purposes. Now I am not a computer scientist to know all the ins and outs of this. I received the past year's AIPSLETTERS as a packet in the mail just a few weeks ago including notice of the ongoing email discussion about the new AIPS and have slogged through a portion of it to inform myself. I wonder if you might to able to give me some digested answers to the three concerns my friend raised. Larry Molnar.

Larry delicately addressed a worry commonly addressed to me in language sometimes too pungent to include in a family memo series. There is great concern, amongst what is probably the majority of astronomers, that the C++/OOP implementation of AIPS++ is: (1) based upon a recent CS fad; (2) being done before commonly (and cheaply) available C++ compilers are in hand; (3) going to leave the majority of astronomers out of development in the near future; and (4) a mistake. My own view is that there is no problem with good compilers; however, there could be a problem if it is expensive for users to purchase a commercial development system. This is the same problem one would have if it is assumed that a system like IDL is accessible to AIPS++ users, and the main argument against this is the expense for the users. I am now working with two easily available C++ compilers: Borland C++ 2.0 for PCs, and the GNU C++ that we have running here at the AOC on both Sparc workstations and DOS PCs.

There is going to be a learning curve that the AIPS++ project can help with by paying special attention to astronomer-oriented programming guides, "summer schools", etc. I do not think the C++/OOP approach is a mistake – but rather a classic example of "no pain, no gain".

10

While attending the SETI IWG at Green Bank, I stopped by Charlottesville and talked to Croes about the AIPS++ effort. I think that our SETI software effort should be aware of yours, and we should definitely ensure that our data will seamlessly interface to the new AIPS software. Edward T. Olsen, Jet Propulsion Laboratory.

This is one of the several comments illustrating the need for both effective communication and "seamless" interfacing that I hope will be a pre-eminent requirement for AIPS++.

11

Dear DEVELOPER of AIPS,

I got from you some letters with AIPS USER AGREEMENT , AIPS letters, and a short letter to Potential Users of AIPS++ from R.M.Hjelling. I have some questions now because a month ago I had a talk with L.Gurvits from Astro Space Center USSR. He advised me not to use AIPS till creation AIPS++. There are only IBM PC AT 286/386 in our observatory. They are with XENIX (SCO 2.3) and 386/ix (ISC 2.0) systems. I have filled in a form of AIPS USER AGREEMENT but didn't send it because had a talk with Gurvits (hereinbefore). I am afraid if I send you a form you will send us AIPS which we will not be able to use. Am I right ? RATAN-600 at which I work has some features and data processing is connected with their. RATAN-600 is a telescope working in a transit way. It can use one sector of all surface (all surface is a ring with diameter 600m and each element of surface has height about 10m). It works at different regimes : one sector or all suface, at different wavelengtes simultaneously (till 7 waves as obsevation cabin permits), and 3 main different directions groups of observators works at RATAN. These are continuum group, spectroscopics group and solar group. So there are more then 3 types of data reduction. I am from continuum group and interesting in such types of data reduction. I made and make now software (in language C of UNIX) for working with deep surveys (my main topic), statistics of survey , calculation of instantaneous spectra, working with data base of radiosources.

When we use one sector an antenna has beam like "propeller" (at elevation 50 degs at 7.6 cm FHPW = 4 time sec by RA and 20 arcmin (') by declination. RATAN works in transit way. And usually we make one or several strips by sky and as a result we have array with dimensions 5 times 6000 sq.pixels. Not square arrays. And all spectra of radiosources we get using transit manner. But we can comove carriage with horns of receivers with source about 2-3 minutes. So our data processing is connected with such featurs. And it will be very interesting and important for us to have such development of AIPS if it's possible.

Reallly I didn't see the work of AIPS (I saw only MIDAS system. It works in our observatory under 386/ix (ISC 2.0) with optical data from 6-meter telescope) and can't image how it looks like. It will be very ineresting to hear about possibilities of AIPS to woork with our data. If it's possible I can work with branch of AIPS connected with our types of data. May be somebody needs it too. Or if it exists we will be very happy to hear about this.

Best wishes, Yours, potential user and friend of AIPS, Oleg Verkhodanov, Special Astrophysical Observatory of USSR AS, Nizhnij Arkhyz, Stavropol Territory, USSR.

This illustrates themes in many of messages I have received. The largest expression of interest is from those outside the US, and in many cases the instruments and computers are diverse.

12

Three suggestions for the AIPS++ user interface:

1. Menu-driven/graphical top level interface (a la Macintosh)
2. Less cryptic task names!! (or at least a much better help system to guide users to the desired tasks)
3. If the extension file concept is used, make as many of them ASCII format as possible to allow easy editing, printing, etc. The use of special purpose editors to modify or examine extension files should not be necessary.

Dayton Jones, JPL.

13

I was just thinking about a possible user environment for AIPS++. It seems to me that a tree structure, such as IRAF uses, might help to organize different aspects of reduction and analysis. For example, when the user "starts up" AIPS++ s/he could be given a menu of initial options, such as

Calibration, Imaging, Image Analysis, Data IO, etc.

By choosing Calibration, another menu could be displayed, such as

AT, MMA, VLA, VLBA, etc.

By choosing one of these options, a list of calibration routines for that particular instrument could be displayed.

Similarly, the submenu under Imaging could display mapping and CLEANing routines.

Image analysis could be further divided into continuum and spectral line, for example.

The user could define a pathname to get to a particular subdirectory instead of having to type the entire pathname. For example "spec" could be defined by the user so that when it is typed the user is placed on the Image Analysis/Spectral Line branch.

This structured environment would allow new users added ease in finding what they are looking for. In addition, the "core" programs and instrument dependent tasks would be, for the most part, clearly divided into separate branches of the tree. The menu choices would have to have understandable names (which is not always the case in IRAF) to be more user friendly.

Just an idea, Dave Mehringer

14

The following is a set of views reacting to a draft of the specifications memo 105. I have copied it almost intact, whether I agree with the sentiments or not, because it expresses many valid concerns and questions.

The Draft specification contains a huge amount of exciting material; probably more than the AIPS++ will be able to deliver for a number of years. In general the document covers the broad picture quite well, but as the authors would expect there are a couple of major areas where I think a change of emphasis and some new thinking are needed. In addition I have many detailed suggestions to add to the wish lists in sections 5 and 6.

It seems to me that a very large part of this document is concerned with ensuring that AIPS++ includes the full functionality of the existing AIPS package. While this is indeed essential, the Draft does not always come to grips with the truely radical possibilities for improvement made possible by ditching the old AIPS model of an analysis system. Where the Draft does go beyond old AIPS it mostly is in those areas which are completely outside the original AIPS remit: single-dish analysis, pre-observing calculations, modelling, etc. An honourable and welcome exception to this is the call for much more flexible access to uv data, which I wholeheartedly support. But other possibilities for more than minor or cosmetic improvements in the core functions of AIPS (synthesis image formation and image analysis) are often missed. One symptom is that many of the functions listed as 'tasks' should really be considered tools which can be combined in many ways and often be run in parallel.

Specifically: On image formation, the Draft does not sufficiently take into account that data examination, editing, calibration (including self-calibration), fourier inversion, CLEAN, MEM, MaxEmpt, etc. are all aspects of a single task: in a truely flexible system these are not separate stages in a pipeline but fully interlinked activities. Thus in MERLIN data analysis the MAP program automatically updates the self-cal solution after each major cycle of CLEAN, and the revised visibilities are used to calculate the next iteration residual image. Moreover it is commonplace to stop in the middle of MAP to examine the fit between data and model visibility, and, on the basis of this, to edit the data and then re-activate MAP starting from where it left off.

In a similar vein, the success of the MX and VTESS programs seems to have blinded users to the fact that each of these combines two quite separate functionalities: CLEAN + multiple subfields, and MEM + mosaic. The mosaic vs. sub-field distinction appears fundemental in that it is hard to imagine datasets requiring both facilities at once. On the other hand many experienced mappers now use a combination of MEM and CLEAN to produce a single final image, and I for one have had occasion to wish for both an MEM-based sub-field facility (which is straightforward to implement), and a CLEAN-based mosaic (which would require some development). An advanced imaging task would allow the user to swap deconvolution algorithms in mid-process in either mosaic or sub-field mode, and for sub-fields one should be able to choose the algorithm applied to each sub-field independently. As a consequence of this fusion of the MEM and CLEAN approaches, self-calibration programs should be indifferent as to the origin of the model supplied: old AIPS is optimised only for CLEAN models.

On the image analysis front, to me a glaring omission was any request for built-in pixel-based error estimates. We should remember the truism that a number is only meaningful when accompanied by a measure of its accuracy. Use and propagation of reliable error estimates is really essential in complex image analysis such as the determination of optical depths, rotation measures and other polarization parameters, and spectral fitting. Less obviously, the wildly inaccurate error estimates provided by gaussian fitting and similar programs 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 on each pixel. These errors estimates (hereafter simply errors) should be available for use in statistical analysis and in display programs (e.g. the plotting of error bars on profiles; automatic warning if contour intervals are less than the noise; error-based blanking in display [rather than calculation] of results; properly formatted errors when image values are extracted for tabular presentation). In fact the author of every application program should try to make appropriate use of error information.

Ideally errors should be propagated right through the system from the original visibility weights. MEM 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).

In comparison with proper treatment of errors, the use of 'blanks' as discussed extensively in section 8.1 is extremely crude, and can lead to appalling biases in image analysis: for instance see VLA Sci.Mem. 161 on the adverse effects of blanking in polarization and especially depolarization analysis.

Some of you know that last year Bill Cotton and I spent quite a lot of time devising a system for including the above possibilities in the old AIPS. Our work was eventually aborted when old AIPS was frozen, but I learned enough to convince me that errors should be included right at the beginning of system design, as it is difficult to insert such a fundemental property at a later date.

Other minor comments/suggestions:

- Self-calibration and/or crossing-point calibration of 'global' parameters when combining datasets, i.e. amplitude scale and position offset (also perhaps field rotation in some cases). At present the necessity for a final 'joint' self-cal to align datasets can often corrupt their individual self-cal solutions.

- Limited-parameter self-calibration, e.g. for a coherent phase wedge over the array rather than for uncorrelated phases at each telescope. Useful for calibrating at low S/N.

- Intelligent image-based continuum subtraction in spectral line cubes need not use the same channels for the continuum on each pixel: instead find line-free channels for each pixel separately.

- 'Profiles' should handle cases with uneven increments between values, e.g. broad-band spectra containing points at 0.3, 1.4, 1.6, 5... GHz. This is important for multi-frequency spectral and polarization work.

- The calls for fitting of gaussian components should be supplemented with routines to fit other (or general) functions, e.g. projected ellipsoids and cylinders, King and de Vaucoulers models, Lorentzians for spectral line fitting, etc. Optionally these should be convolved with the beam.

- Coordinate conversion facilities for image headers, plus facilities to re-project images onto a specified coordinate grid. (B1950, J2000, galactic, ecliptic, etc).

- Surface integration bounded by specified contour; automatic recognition and contour integration of extended sources on images. (Liz Waldram of MRAO, Cambridge, has developed a sophisticated package along these lines for analysing the Cambridge survey data).
- 'Blotch' boundaries should be recordable and able to be passed between tasks/tools to enable consistency, possibly over extended periods (e.g come back 6 months later and measure the flux over exactly the same region of an image based on new data).
- 'Cubes' should allow for random positions of planes on the frequency axis for multi-frequency spectral and polarization analysis. Frequency information for each plane should be retained.
- General 'vector' pixels (e.g. I/Q/U/V, intensity/velocity/dispersion, value/error/chi-squared/n.d.f.) should be available.
- Vector plots for polarization and other angle-related quantities. All comments re contour plots apply here too: e.g. arrays of plots, intelligent grey-scale superposition, etc.
- Possibility of specifying several sub-regions for histograms and pixel vs pixel plots, differentiated by different linestyles/symbols.
- Visibility data may well contain autocorrelation data, e.g. for noise diode calibration of polarization position angles.
- I'm horrified by the suggestion that AIPS should adopt the optical convention of interpolating (with fake noise) across blanked pixels. This is a recipe for confusion at best (who will remember that a region was faked when it later turns out to contain a 'variable source' invisible in the first observation?). At worst it is an invitation to scientific dishonesty (e.g. in scrubbing out presumed calibration artefacts to increase the dynamic range).

Patrick Leahy, NRAL Jodrell Bank.

15

Here are some inputs for AIPS++. I'm not sure if these are the kinds of suggestions u are looking for, but will throw then into the hat for consideration.

1) LISTR. OPTYPE = 'scan' option needs improved printout format and additional information. The observing date needs to be printed out. Timerange needs to be annotated properly (e.g. 'IAT'). Az and el info would be useful. IF(1)and IF(2) should be separately labeled in the 'source summary' section, so that IF 1/2 info can be clearly distinguished. The old DEC10 version LISTER had a well-designed output format (for the 'scan' option) which was easy to read and had all of the above info. The AIPS output format was a step backward.

The above was submitted as AIPS gripe 4169, but to my knowledge none of the suggestions were ever implemented. I use the 'scan' listing, along with the observing logs, as a permanent record of the run (expect other observers may as well), so having an easy- to-read, complete scan printout is very helpful.

2) AIPS BATCH capability does not work. Was submitted as AIPS gripe 4168. Dont know if it was ever fixed. BATCH capability would be useful, but we have found ways to get by without it.

3) FITTP can not handle multi-volume files. It would be nice to have a tape utility that does not abort when it encounters an eof while writing to tape. This may be difficult to implement, because of the wide variety of tape drives in use. This was submitted as AIPS gripe 4167.

4) Entering baseline corrections during calibration for runs that took place during antenna moves is tedious. Can this procedure be streamlined? (esp. keying in the long sequences of numbers)

5) New versions of the AIPS Cookbook do not have an index at the back which gives the page or section number for a particular task. As a result, one has to thumb thru the cookbook to find the task of interest. Older versions of the cookbook had an index with section numbers.

6) During continuum data calibration, the user is required to keep track of table version numbers (e.g. GAINVER, GAINUSE). Can the software be structured so that this housekeeping becomes automatic?

Regards, Steve Skinner

16

The following was sent to me relatively early in the AIPS++ Development Seminar series that started at the Socorro AOC in August 1991. It contains material that I had hoped Tim would put in a memo, particularly since it expresses a passionate and valid view about how people want to do science with a system.

It seems to be widely accepted that AIPS++ should be a data reduction and analysis package capable of dealing with a wide variety of instruments operating across the electromagnetic spectrum (nobody has requested support of particle data, as far as I am aware). It seems to me that the problem presented by AIPS++ naturally divides itself into two parts: 1) data reduction; 2) data analysis. There is and must be, of course, a certain degree of feedback and overlap between the two parts, but let's set that aside for now.

DATA REDUCTION

Let me first deal with data reduction. First, what do I mean by "data reduction"? By "data reduction", I mean those operations performed on a "raw" data set from a given instrument which render it usable for scientific "analysis". That is, the end product of "data reduction" is the starting point for "data analysis".

Now it seems to me that one may think of the data reduction process as one of weaning the data from the instrument. The raw data are necessarily tightly coupled with the instrument which obtained them. It is through data editing and calibration followed by various kinds of signal processing in the spatial, temporal, or frequency domains (e.g., image deconvolution, spectral deconvolution, filtering, etc.), that one progresses from the data as acquired by a given instrument, to the final map of radiation intensity as a function of frequency and/or time and/or position on the sky, or line intensity as a function of frequency and position, or a calibrated pulse profile, etc. These end products have little to do with the original instrumentation used to acquire them but everything to do with the science one wishes to do.

Since data reduction is so tightly coupled to the instruments responsible for acquiring the data, one may well ask if we intend to support the reduction of data from "all" astronomical instruments? The answer is "No, that would be absurd". For example, it would be silly to support the reduction of Einstein archive data – software to do so already exists. Ditto for many other instruments. What about supporting the reduction of "most" radio-astronomical instruments; i.e., single dish, linked array, and VLB array data? This is what most people have in mind, I think, and I'll proceed on this assumption.

It is obvious that we should strive for data base structures which can accomodate most kinds of radio astronomical measurements. The types of measurements which spring to mind are continuum, spectral line, and time series measurements made by interferometers and single dishes. I am certainly in no position to say what data base structures are best or possible; whether one structure can handle all foreseeable data types, or whether it would be best to have a number of data structures, each tailored to a specific reduction problem (one data structure for pulsar observations from the GBT vs. another data structure for HI mapping with the WSRT, for example).

Regardless of how things are done internally, I would think it easiest on the user if one could enter a reduction environment designed to the reduction problem at hand (continuum, spectral line, single dish, array, etc.). Once in a given environment, it would be extremely useful to have a means of further constraining the environment by identifying the particular instrument(s) used to acquire the data. The point of this latter step would be to allow the environment to set its "commons" with all available instrumental parameters of interest, to alert it to pull in additional instrumental or site-related parameters of interest when the data are loaded, and to provide an organizational "filter" for reduction tasks relevant to the particular instrument and data type in question. For example, if I were doing a reduction of a continuum data base from the VLA, I would want to set up an environment which would know about the VLA and what kinds of information are important to the calibration of VLA data. Furthermore, I would want the environment to know which data reduction tasks are useful to calibrating VLA continuum data and to have their names and documentation easily accessible. I would not be interested in having lots of extra baggage around such as VLBI or spectral line reduction software. Similarly, environments could exist for standard VLBA reductions, GBT spectral line work, etc. Of course, it would have to be a fairly straightforward business to create additional reduction environments as needed. Furthermore, because personal tastes and styles vary, users should be allowed to create personal reduction environments.

I would think that data reduction would in many cases remain "task" oriented. That is, AIPS tasks like CALIB or BPASS which compute calibration parameters, or CLCAL and SPLIT which interpolate and apply the calibration, or tasks designed to do image deconvolution, may best be run as standalone tasks. An alternative is for data reduction environments to provide many single purpose "modules" (e.g., one to solve for amplitude and phase corrections, one to do a bandpass calibration, one to apply a Tsys correction, one to map data, one to CLEAN maps, ...) which could be piped together in flexible ways. On the other hand, most people wish data editing and troubleshooting to be less task oriented and more interactive. It should be possible to access a given data base so that one can quickly and easily examine the data in a large number of ways (printout, graphically, as image data, in 1, 2, or 3D representations). For example, it shouldn't be necessary to have to run UVPLT every time one wants to examine a given baseline. One should be able to plot a baseline (or the visibilities corresponding to a specific timestamp, or any other number of variations) with one or two command lines.

While the above general conception of data reduction in AIPS++ may not be accepted, I do think that we have to think about the nature of data reduction in AIPS++ and to specify our desires rather precisely before getting down to the nitty-gritty of what functionality the package should provide. If we don't have a context for the functionality of AIPS++, how can we hope to specify it in any meaningful way?

DATA ANALYSIS

As stated earlier, the end product of "data reduction" is the starting point for "data analysis". It is here that one has freed oneself of a given instrument and has presumably obtained some physically meaningful result (a map or spectrum calibrated in physical units, etc.). One can then proceed to exploit any and all available tools to extract whatever science is of interest from the data. It seems to me that it is at this stage of the game that we would like to open things up in AIPS++ and support the manipulation and analysis of data from essentially "all" astronomical instruments (optical, IR, UV, SXR, HXR, etc.) Consequently, it is the data analysis part of the AIPS++ package that must present the user with enormous flexibility.

In the area of data analysis, one has exchanged one set of problems for another. For data reduction, the data structure(s) necessary to support a large number of radio-astronomical instruments will possibly be quite complex. On the other hand, the range of reduction environments that one needs to support, and the variety of tasks needed for the reduction of radio-astronomical data, is limited and the application of such tasks, repetitive. In the case of data analysis, it is the other way around; the data structures needed to describe "most" astronomical observations of electromagnetic radiation are simple but the variety of tasks needed to manipulate the data is enormous.

The end product of data reduction is calibrated radiation intensities as a function of time and/or space and/or frequency, etc. Simple n-dimensional, floating point arrays with an appropriate descriptor of each axis should be sufficient for most if not all kinds of data (the FITS-like approach). It has been mentioned in past discussions that support of irregularly sampled data (temporally, spatially, or spectrally) is desirable; this should be possible with a suitably general axis desciptor. True, to get data into and out of AIPS++ from a wide variety of instruments will require filters of one kind and another, both "standard" and user defined, to convert data into a convenient internal format. In any event, the idea of supporting data analysis for a large variety of astronomical instruments does not seem to be a daunting task.

What may seem, at first, a daunting task is that of providing the user with a comprehensive set of software with which to analyze his or her data. If we grant that it is simply not possible to anticipate all of the tasks or procedures that a user will want, the question becomes one of providing a sufficiently extensive software toolbox and a sufficiently powerful and flexible command-line language that users can quickly and easily write their own analysis software. This is not so daunting a task – examples of the sort of thing I'm talking about already exist and have been used with great success (i.e., IDL/PV Wave).

CLOSING THE LOOP

So far, I've described my ideas concerning AIPS++ in terms of the two, rather separate, processes one performs on data, reduction and analysis, and have not said much about the interplay between the two. After all, why not just write two separate packages? Obviously, there will be many situations where reduction and analysis functions will overlap, or the tools provided by one will be useful for the other. An obvious example is the production and manipulation of model data. Another example, already mentioned, is the examination and manipulation of "raw" or partially calibrated data. So I think it's pretty clear that all of the software tools available for the purposes of data analysis should be applicable at almost any stage of the reduction process. This implies that "raw" data can be dealt with, or put in a form that can be dealt with, by the software tool box. Let me provide a concrete example. Calibration of VLA continuum observations of the Sun involves making an offline Tsys correction. Before doing so, one examines the Tsys variation for each antenna and polarization channel, flagging out bad antennas or time stamps. To do so in AIPS1 involves running the tasks SNPLT, TCLIP, and TABED repeatedly followed by a run of SOLCL. One possible way of doing it in AIPS++ is (in analogy to PV Wave) to do it in a few command lines.

1) Pull the Tsys measurements from a "common" data structure for antenna N, IF 1, RCP into an array and plot it on a display device:

```
ARRAY = TSYS(N, IF, POL); PLOT ARRAY   or

ARRAY = TSYS(N, IF, POL, START = [HH, MM, SS], END = [HH, MM, SS]);
   PLOT ARRAY
```

I've assumed that parameters IF and POL may be set with values 1 and RCP, respectively, and that some "common" called TSYS contains all of the antenna TSYS measurements.

2) Edit low values (as often occur at the the beginning of a scan) with a simple clip

```
TSYS(N, IF, POL) = ARRAY > VAL    or

TSTS(N, IF, POL, START, END) = ARRAY > VAL
```

That is, place Tsys values in ARRAY greater than VAL back into the "common".

3) At some point, run a task which applies TSYS to the data.

One can also imagine that the command line language used to write complex analysis procedures could be used to set up calibration procedures. For example, one might want to set up a procedure to do self-calibration iterations, displaying information (printed, graphical, or image) along the way. This should be easy to do.

To conclude, while data reduction and analysis are in many ways distinct from each other, one can imagine many situations where the two overlap. A powerful command-line language and software toolbox should be the common thread between the two.

SUMMARY

I have suggested that the AIPS++ problem naturally divides into two parts. The data reduction part is concerned with allowing the user to reduce radio-astronomical data from a wide variety of single dish, linked arrays, and VLB arrays. It would be useful to the user to be able to define a reduction environment tailored to the data type and instrument at hand. Because of the relatively simple and repetitive nature of data reduction, it may be appropriate for it to be "task" oriented. However, since there are often variations on the theme, a "modular" approach to data reduction may be preferable. The tasks of data display, editing, troubleshooting, etc. are best served by highly interactive, "non-task" oriented, computing. These latter aspects of data reduction would be best served by providing the user with an array of tools designed to grab, display, and modify data. These tools might best be used in the context of a command line language.

Data analysis involves the display and manipulation of data that is divorced from any particular instrument. It is this part of AIPS++ that can and should support all types of astronomical data. I claim that the (internal) data structures required to do so can and should be relatively simple. The import and export of astronomical data will likely require data filters of one kind or another, however. A powerful and flexible command line language which supports user programing and provides a large library of software tools is essential. The ability to write software modules using the command line language, or a lower level language, is highly desirable. Complex data analysis schemes should then be quite easy to implement.

The two parts of AIPS++ need not and should not be strictly separate from each other. While the software toolbox embodied in the command line language should be primarily designed for data analysis, it also lends itself to interaction with raw data (display, editing, etc, as mentioned previously). Similarly, for certain kinds of data analysis, it will be more convenient to treat it as a reduction problem, running repetitive precedures in a restricted environment.

To conclude, I agree with the remark made by Chris Flatters that one must describe the overall structure of AIPS++ in order for the programmers to make headway. I have attempted to do so above, but in terms that are still much too vague to be of practical use. I think that a fair amount of effort should go into sharpening up the concept of how AIPS++ should approach the problem of data reduction and analysis from the user's point of view. It is only after the structural specifications are made clear can one begin to think about a detailed specification of the baseline functionality of AIPS++ and of the user interface.

Tim Bastian, NRAO, Socorro

17

Hi Bob, Here is a suggestion for aips++:

Here is my idea of one way to organize the overall structure of aips++. I am writing as if it is menu driven, but it is important that there should also be a command driven option, so that the experienced user does not have to wait around until the appropriate menu is presented. I have used current AIPS tasks as examples, although these may not appear in the same form in aips++.

One of the things which I think would be useful in aips++ is a good way of pipelining the processing of data, allowing for the use of several atttempts at any stage of the work. This leads to a tree structure, starting with the raw uv data at the root and extending up to the final maps as the tips of the branches. Ideally aips++ would make it easy to see the structure, which might involve several tries at self-cal, for example, and easy to navigate around in it, both forwards and backwards.

A good display of the tree would be helpful in directory maintenance, since it would be easy to see which uv files have led to the current maps, and which were just dead ends or have been superseded.

Here is an example tree. It is not quite this simple, since some tasks have more than one input (e.g. ASCAL, COMB), but I think it is still useful as a way of thinking about the data flow.

```
raw uv -- 1st cal -- map
       |
       -- edit -- 2nd cal -- map
                          |
                          -- selfcal -- map A
                          |          |
                          |          -- map B
                          |
                          -- selfcal -- map -- selfcal -- map
```

Navigating backwards: As an example, suppose I have a map, but would like to remake it with different parameters. In aips++, I would like to be able to SELECT that map, and say something like REDO MX. That should look up the history and put me back in the parameter setup that was used for MX (assuming that the necessary uv file exists). It would then be a simple matter to change one or two parameters and rerun the MX. If I wanted to go back and do another set of editing or self-cal, I should be able to SELECT the map and say something like ANOTHER SELFCAL. Ideally, this would give me the option of going right back through the map-making stage automatically using the same parameters as in the previous iteration. (At this point one might think of making a loop which repeats the selfcal automatically, according to the improvement in noise on the map.)

Navigating forwards: Navigating forwards automatically is a bit more difficult because there are more choices, but I would like to SELECT a uv dataset, say, and be given a list of appropriate actions, perhaps grouped into passive (display) and active. For example, if I pick a spectral-line, uncalibrated uv dataset, the basic options should include amp/phase calibration, editing, and passband calibration, in whatever is the preferred order. If I have already done the first two, then the third should be top of the list. Once the calibration is complete, then SPLIT would be the prime option (using current aips as an example). Since SPLIT creates a new dataset, that would become the SELECTED one and a new set of options would be presented, with MX at the top.

So, in my view of aips++, one would have a particular dataset SELECTED at any one time and there would be 3 groups of commands available. The first would be appropriate display tasks, the second would be appropriate processing tasks, in order of most common use, and the third would be tasks to go back up the tree some number of steps and try again. For flexibility, of course, inappropriate commands would be available also, but not so prominently displayed.

The next thing which follows from this idea of having a SELECTED dataset, with context-dependent commands, is to have context-dependent parameters. Of course this is more difficult, since there are many more choices, but it should be easy to make a lot of improvement over the current situation, for example, by having the number of channels in a spectral line dataset, for example, be the default for channel settings in each processing task. Cell size would follow from the uv range, etc.

If this is implemented well, then a beginner could just load the raw uv dataset, SELECT it, and follow the suggested tasks in order, with the default parameters, to get a reasonable map. The more experienced user would accept most of the defaults but only have to change a few of the parameters or tasks to get his or her more sophisticated map.

The next stage of sophistication would be to allow the user to specify his/her own context-dependent preferences in some sort of configuration file to customize the navigation through the tasks. Perhaps this could even be done in a learn mode (in real time or by looking through history files).

Another advantage would be in script (RUN) files. The less that parameters have to be specified each time, the easier, more general, and more reliable become any script files which might be used. In addition, it would be very helpful to have access to internal variables of datasets and results of programs from the script files for conditional tests. For example, one might want to set CLEAN parameters according to the results of IMEAN, or use the map rms to decide whether to do another round of self calibration.

I hope this is useful, and I look forward to seeing the other suggestions. Please put me on your mailing list. Colin R Masson, Center for Astrophysics.

18

I don't know what to say about AIPS++ except that it should be well documented with a simple introduction for innocents. If showed your mail to J.S.-B. who has been struggling with the *local* version of AIPS. He says that:

"Improvement is needed in the programmibility of Tasks. At present, one cannot introduce even small alterations for private use, nor can one understand what the routines do internally. This in particular applies to the editing of data files."

He also says that documentation is very good in parts and very bad in other parts.