DUTCH REQUIREMENTS FOR AIPS++

J.E.Noordam, editor

31 December 1991

Contents

1 INTRODUCTION

The various Dutch astronomical institutes welcome the initiative to create a new Astronomical Information Processing System (AIPS++), and have formed a **Regional AIPS++ Group** to coordinate the Dutch interactions with (and the contributions to) the AIPS++ consortium. These *initial Dutch requirements* for AIPS++ have been compiled by the Regional Group.

For various reasons, we found it difficult to consider all the aspects of AIPS++ in sufficient depth and detail. Therefore, we gratefully acknowledge the extensive use that was made of the requirements formulated by NRAO and ATNF. Both these documents are essentially *checklists* of aspects that should not be forgotten in AIPS++. It is encouraging that there is very little disagreement between them.

Thus, unless stated otherwise, we substantially agree with the NRAO and ATNF checklists, in which most of our requirements are treated in some form. On the whole, we are confident that the **image-plane** side of AIPS++ and the **user interface** will present few problems of principle. We are a little more concerned about the basic structure of the **uv-plane** side of AIPS++, since that is the area where the detailed differences between various "cultures" have to be reconciled.

In the following sections, we *emphasize* those areas that are particularly important for Dutch users of AIPS++. For a quick summary, the reader is referred to the Table of Contents.

2 PROGRAMMABILITY

The need for programmability of AIPS++ cannot be emphasized enough. It is a prominent feature of all the requirement documents. It has been the cherished ideal of many creators of astronomical software packages in the past, but none has been particularly successful yet. This is partially because they concentrated too much on a particular instrument or set of applications, and partly because they were not computer scientists. What we need is a powerful and well-supported **Operating System**, with many applications thrown in.

The need for such an Operating System has been clearly demonstrated by the painful attempts of some institutes to use AIPS1 for this purpose. Some very active groups now **intend to rely** on AIPS++ as an environment to develop their own software, which may or may not find its way into the "official" version of AIPS++. Some of these groups run their own instruments (like NFRA runs the WSRT), others are leaders in a particular field of astronomy (like Groningen in the field of radio spectral line). They all have in common that they cannot afford to write (and maintain) a sufficiently elaborate software package, especially for *outside users*. Successfull programmability of AIPS++ would liberate instruments from isolation, and lower the critical size of productive university institutes.

Obviously it has its dangers to rely collectively upon a single system. Its heaps an enormous responsibility on the central AIPS++ "hub" in Charlottesville. This role will continue in the operational phase, and has two *distinct* parts:

- Development and support for the Operating System part of AIPS++. This includes new hardware platforms and peripherals.

- Integration of new Applications into the "official" version of AIPS++. This includes the enforcement of uniformity of programming style, documentation and nomenclature.

Although this may seem an organisational matter, it is a clear requirement for the revolutionary type of software package which AIPS++ purports to be.

3 WSRT OFF-LINE SYSTEM

NFRA intends to use AIPS++ as an environment for the entire WSRT off-line system. As emphasized before, this is *a novel (and hazardous) use of an external software package.* The planned WSRT off-line system includes the following main parts:

- **Calibration:** Derivation of instrumental parameters from calibrator observations. Results are used to adjust instrumental settings, and to correct "Primary Data" afterwards.
- **Semi-online monitoring:** Detection (and reversible flagging) of "bad data" during the observations, by checking for internal inconsistency (e.g. redundant spacings) and other "unphysical" behaviour.
- **Archiving of Primary Data:** The data is archived as it emerges from the on-line system. It has an associated Data History, which consists of instrumental settings and a description of on-line corrections and operations.
- **Correction of Primary Data:** Application of (interpolated) calibrator gains and phases, and other corrections. A complete description of operations is added to the Data History.
- **Further uv-data processing:**
- **Image analysis:**

Only the last two items are expected to be handled by astronomers in their own institutes, in the same way that AIPS1 is used. For the WSRT, the other items are Observatory tasks, which have to be handled consistently, efficiently, and more or less automatically within AIPS++. It is expected that about 80 percent of these tasks will be covered by *common code*, while for the remaining 20 percent good *programmability* is essential.

It is of the utmost importance that WSRT uv-data fit comfortably into the AIPS++ uvw-data system, and a description of the uv-data Application Programmers Interface is needed at short notice. What is also needed is a *consistent* framework in which all (kinds of) corrections to uv-data find their natural place. NFRA will make its own proposals for these two items, in time for the Green Bank meeting of the uv-data Tiger Team in February 1992.

Time-scales: The upgrade of the WSRT software is part of a general upgrade of the WSRT, with new frontends and backend, in 1995. This places a rather severe time-scale limit on the new off-line system, which has to be tested during 1993, and fully operational in 1994. The present official time-scales of AIPS++ are consistent with this goal, but if these seem to have slipped too much by the middle of 1992, alternative routes will have to be considered.

4 WSRT STRONG POINTS

It should be possible in AIPS++ to make the most of the "strong points" of the WSRT. Typically, these are not catered for in AIPS1, because of its strong bias towards the VLA. For some of these strong points, the user may have to invoke special applictaions, provided by NFRA, but built onto the same base. Others should be available as natural generalisations of the common code. Incidentally, most of the WSRT strong points are shared with the AT and Penticton.

4.1 Wide-field mapping

An advantage of a 1D East-West array is the constant Point-Spread Function (PSF, or beamshape) over the map. Because of this, instruments like the WSRT have no socalled "3D problem" with the deconvolution of sources far from the field centre. This makes them particularly suitable for wide-field mapping and mosaicing. It should be easy in AIPS++ to convert between different uvw-coordinate projections.

4.2 Polarisation

The WSRT has very low instrumental polarisation, because of the position of the feed on the symmetry axis of the dish. Also, because of the equatorial telescope mounts, the linear feeds remain stationary with respect to the sky during the observation. These two properties, together with redundant spacing calibration, make the WSRT very suitable for the precise measurement of small amounts of polarisation.

It should be recognised within AIPS++ that a full measurement of the EM field consists of 4 complex numbers, which can be converted to Stokes parameters. This should be built in from the start, and not as an afterthought like in many existing systems. The various differences in telescopes and feeds should be dealt with in a way that is hidden from the user. Fortunately, the WSRT, VLA and AT represent an almost complete set of different polarisation configurations: linear and circular dipoles in equatorial and alt-az telescopes.

4.3 Redundant spacings

The redundancy of the WSRT array can be used for Redundant Spacing Calibration (RSC), which can be seen as a model-independent form of SELFCAL: it has the same basic assumption as SELFCAL, namely that all interferometer-based errors (including noise!) are zero. And like SELFCAL, it solves for antenna-based phase and gain errors. RSC can be used in two modes:

- As an **extra constraint** on SELFCAL. This is easily implemented as extra linear equations (constraints) in the SELFCAL least-squares fit solution matrix. Since the redundancy equations represent a "higher truth" than the model-dependent SELFCAL equations, they should be given higher weights.

- By itself, as a powerful **internal consistency** check on the data. This is useful for the automatic detection of interference and other sources of "bad data".

RSC should be a standard option in AIPS++, to be used whenever there are redundant spacings available (both AT and Penticton have limited redundancy). It is not clear whether "crossover redundancy" in 2D arrays can be treated in the same manner.

5 GIPSY (IMAGE PLANE ANALYSIS)

The group at the Kapteyn Lab in Groningen has been a world leader in radio spectral line astronomy for at least a decade. They have developed their own software package for image plane analysis (GIPSY), which is used at many other sites. However, they intend to switch to AIPS++, provided it has at least the same "functionality" as GIPSY, and provided it is *programmable* enough to allow them to be in control of their own development.

For a detailed description of the GIPSY requirements, the reader is referred to the appendix ("The Groningen Document"). It will be seen that some of the analysis tools of GIPSY have already been adopted by other packages, and many of them are mentioned in some form in the in the NRAO and ATNF checklists. The GIPSY group feels that some elements of the GIPSY data structure could be useful for AIPS++, particularly the treatment of *ancillary data* at the various dimensional levels. They are in direct contact with the KIPS group in Canberra about this.

6 VLBI

Reduction of VLBI observations is important for Dutch radio astronomers, especially when the proposed Processing Centre of the European VLBI Network will be built in Dwingeloo (1996). The European VLBI requirements for AIPS++ will be compiled and submitted by Jodrell Bank.

VLBI processing should be truly integrated in AIPS++, in the sense that there is a minimum of duplication with "normal" aperture synthesis. Therefore, early attention to VLBI in AIPS++ is needed, to make sure that it fits in, and to produce a better package altogether.

7 PULSARS

Pulsars are mentioned explicitly here, because they are expected to become an important area of interest for the WSRT in the near future. Pulsar candidates can be identified by looking for polarised steep-spectrum sources in the Westerbork Northern Sky Survey (WENSS), which is now underway. They can then be detected with the WSRT in phased-array ("single-dish") mode. Once their period and phase is known, their position may be found by making a map with the new WSRT correlator, which will have sufficient (gating) time resolution for msec pulsars.

Pulsar observations are an obvious requirement for single-dish instruments. It is not clear whether AIPS++ should be designed to handle the data volume of correlator dumps at msec intervals.

8 MISCELLANEOUS

8.1 Automatic Batch Processing

Although AIPS1 has a batch processing facility, most users tend to sit in front of their terminal until the work is done. As the data volumes increase, and more and more sophisticated processing is required, batch processing will become a necessity. This is particularly true for those observatories that offer an extended user service, ranging from doing "standard" calibration to delivering finished maps.

We emphasize the importance of a good *Automatic Batch Processing* (ABP) system for AIPS++, which is easy to use, is safe (completely separate processing streams), has good reporting, and above all is *intelligent*.

8.2 Combination of data from different instruments

One of the *crucial tests* of AIPS++ is its capability to combine data from different instruments (e.g. VLA, WSRT, MERLIN and GBT) in a natural and straightforward way. It must be possible to do this in such a way that the strong points of the various instruments are fully exploited. For the best results, it will generally be necessary to "adjust" the position and flux level of the input datasets, either in the image plane or the uv-plane.

This requirement automatically implies a high degree of communality (and generality!) of **coordinate systems**, be it in sky coordinates, uvw-coordinates, absolute time etc.

8.3 SELFCAL model parameters

The components of the source model used by SELFCAL should be more than simple CLEAN components, with only position and flux. In the present WSRT system has component may have parameters for position (l,m), polarisation (I,Q,U,V), extendedness (long axis, short axis, position angle of an elliptic gaussian), spectral index, time variability etc.

Such sophisticated source models are very versatile, for instance for the combination of different observations, with multiple frequencies and/or pointing centres. Extended sources may be modelled for much more economically (and accurately?) this way than with thousands of CLEAN components. Improved values of these parameters of source components can be estimated automatically in each SELFCAL iteration. More source parameters can be added, for instance to deal with describe non-isoplanaticity, which will be an important problem for the GMRT and its successors.

8.4 Triple correlation data

In optical interferometry, it is often not possible to obtain individual visibility phases, but only *closure phases*, which are linear combinations of 3 phases. They are derived from "triple correlations", or Triple Data. The same situation may occur with low S/N radio observations.

It is not (yet?) possible to make a map by inverting closure phases, but they may be used as constraints in SELFCAL model-fitting. AIPS++ should be capable of processing "Triple Data" in this way, with the following extra features:

- Redundant Spacing Calibration
- Dealing with triples where one of the visibilities has a different frequency.
- Dealing with spectral line visibilities, which are the product of a high S/N wideband visibility and a low S/N narrowband visibility.

8.5 Vector plots and Sector plots

Vector-plots are useful to plot a 4-dim space (2 spatial coordinates, length and position angle of vector), for instance for polarisation or velocity fields. **Sector-plots** are an extension of vector-plots, with the possibility of a 5th dimension by plotting sectors instead of vectors; the opening angle of the sector is the 5th dimension. Example: spectral components (intensity,velocity,velocity-width). A further extension to a 6th dimension could be the shading (colour) of sectors.

In both cases, multiple vectors and sectors respectively should be possible, e.g. superposition of optical and radio polarisation, multiple spectral components.

A APPENDIX: THE GRONINGEN DOCUMENT (GIPSY)

Requirements for AIPS++ prepared by the Groningen software group.

A.1 Introduction:

Over the past two decades the Groningen Image Processing SYstem (GIPSY) has been developed and has remained the main tool for reduction and analysis of mainly Westerbork image data and IRAS images. A large variety of features has been put into the system, some of which are present in other systems such as the present AIPS. At present GIPSY is being ported to Unix and runs with X Window 11.4; at the same time the Space Research Department in Groningen has developed new software under GIPSY to extract IRAS images from the raw data which are available on-line on a jukebox system of magneto-optical disks.

The perspective now is that in a couple of years AIPS++ will come alive as a new environment for dataprocessing. The Groningen software group envisages that AIPS++, provided the basic system is a success, will be used heavily at the Institute and is planning to provide those applications not available from elsewhere. As the ease of implementing existing software into another system depends very much on the programmer interface it is important to describe what requirements will have to be met in order for GIPSY applications to be put into the future AIPS++ (at some point we will need another name!).

The goal of this document is to describe the functionality and routines important for and sometimes unique to GIPSY and to describe what conditions have to be met by the data-structure.

A.2 Functionality

- **Data structure:** The emphasis in GIPSY has been on image-processing of spectral line synthesis data. To simplify matters we will not discuss in detail how GIPSY is set up since many aspects are already mentioned as new requirements in the SWAG report. The fundamental point is that it must be possible to work easily with multidimensional data. A spectral line experiment for detection of Zeeman splitting for example would be a four dimensional data set. The user and the programmer should not have to worry about how the data is stored and should be able to get an arbitrary plane out of the data set with the only restriction that the two axes of the plane requested are two of the N primary axes of the data set. For example: the user should be able to ask a display task to bring up a velocity - declination map without having to transpose his data set. This requirement has consequences for the data structure, especially the way in which the data structure is described internally. The user and programmer interface to the data should be flexible enough to allow general applications to work for all kinds of rectangular data. For example the same profile plotting/analysis programme should work for one dimensional as well as for multi-dimensional data sets. If the data set has more than one dimension, the user/programmer interface must have a simple way to specify along which axis he/she wants to do the plotting/analysis. In GIPSY this has been resolved in a particular way and a short description is given in appendix A.

- **Coordinates:** A second important aspect is the ability of the software to recognise and work with a variety of coordinate systems. Different kinds of data come with their own specific coordinate or projection system and it is very important to have the ability to make proper transformations from one system to the other when one starts comparing data from different telescopes and at different wavelengths. At present only few software systems recognise an extensive set of coordinate systems, and GIPSY is one of them. A routine for reprojecting data into other coordinate system is available, but more importantly the AIPS++ data descriptors should include a large variety of projections.

- **Masking or blotting:** A feature which is being used extensively in many applications is masking or blotting. In some cases this is done automatically by creating mask maps on the basis of some cut-off or S/N criterion in for example HI channel maps. Often however it is done interactively by drawing polygons on the display. All applications requiring an area as input should also be able to digest such a polygon. This has proven to be very useful for applications such as specifying CLEAN component search areas in the case of complex sources and for indicating areas over which to perform integrations, again in the case of complex sources (these could be HI channel maps).

- **HI line Analysis Software:** This is an area where GIPSY has been quite developed as a result of the interest in the Institute for spectral line observations of galaxies. Rather than describing in detail the different programs we just briefly list the applications which we think are quite important for HI line analysis. These are:

  - the ability to make arbitrary crosscuts through the data set in two dimensions (e.g. velocity - position maps where position is a linear combination of R.A. and Dec);
  - the ability to calculate integrated profiles over a specified area (this could be a blot); the ability to properly analyse the velocity field of galaxies;
  - the ability to interactively fit profiles and inspect the fit; the ability to calculate model galaxy data for a comparison with the observations (such software exists and is heavily used in Groningen; we know private copies of some of these exist which work in AIPS but never made it into the official release)

- **General Analysis Software:** Applications which are used for HI line analysis and photographic measurements:

  - the ability to integrate in ellipses and plot along ellipses
  - the ability to fit ellipses to specified sky levels in an interactive way

- **User interface:**

  - The GIPSY user interface allows multi-tasking in an organised fashion. Users have direct control over their tasks: a running task can be suspended or aborted with a simple command; various settings can be changed while a task is running.
  - Parameters for a task can be specified at any time; if a task needs information that has not been specified yet, the user is prompted.
  - Context sensitive help about tasks is provided at a single keystroke (or button-press).
  - Tasks keep the user informed in two ways: they write in a log file in which the user can page and search and they can provide a one-line status message, which can be frequently updated.
  - The interface between a task and the user interface is highly formalized: tasks communicate with the user exclusively through a collection of interface routines, no other means of communication are allowed. This allows us to provide a number of user

interfaces for different environments, e.g. one for work at an ASCII terminal, one for use with X Window and another to run batch jobs. The choice of the user interface has no influence on the task code.

A.3 GIPSY data structure

- Image data is stored in a "crystalline" structure with unlimited (i.e. up to a ridiculous 20) number of dimensions.
- Image data is only referenced using coordinates on symbolic axes. File coordinates are hidden from application programs. The interface routines allow substructures of any dimensionality and size to be read and written. On write, the last axis can be extended and there are provisions to increase the number of axes in an existing data set.
- Associated with the image data there exists a hierarchical system of descriptor (or "header") items. A descriptor item is identified by the combination of its name and its position in the hierarchy. It can be associated with any of all possible substructures, ranging from the whole data set down to the pixel level. This approach has made it possible to eliminate redundancy by putting information on a sufficiently high level while still allowing exceptions by overriding this information at a lower level.
- Most descriptor items have the same format and semantics as FITS header items. The descriptor system however is capable of storing a wide variety of data formats, including tables.

B APPENDIX: THE LEIDEN DOCUMENT

B.1 Command Line Interface

Requirements for the AIPS++ command language, in addition to things already mentioned in earlier documents.

My chief concern is that data reduction and interpretation not be limited to scheduling large preprogrammed tasks. A powerful and flexible language should be directly available to the user in the command language so she can perform normal numerical and graphic operations on small data sets created as she goes along.

Vectors, Matrices and perhaps Structures should be easy to construct and operate upon. Models of such systems exist in APL and IDL. Facilities that should be present are:

- **Unary and Binary operations between matrices** should be implicit, and not require do–loops. Thus the line a = b + c; d = exp(a) where b and c are 35 by 50 arrays, should create implicitly the output array a (so statements like "array a(35, 50)" are not necessary); it should add b and c and put them in a, then it should create the array d of similar dimensions, and fill it, element by element with e to the power a.

- **Expansion of an element to higher order** when combined with a higher order entity should be implicit. Thus if a is a scalar, and b is a vector; c = a * b should expand a by repitition to the length of b, then perform the multiplication and create c with the answer. Similarly a vector can be added to a matrix by column or row duplication.

- **An index generator:** index(k) generates an array with elements 1,2... k. A dimension generator: if A is an m x n matrix, DIMEN(A) is the two element vector (m, n) and DIMEN(DIMEN(A)) is 2, the rank of A.

- **Concatenation:** a scalar can be concatenated on to the end of a vector, a vector to the end of a matrix, etc.

- **An outer product operator O(op):** Thus if a and b are vectors of length m and n, a O(+) b creates an m x n matrix with all the possible combinations of a + b. Arbitrary binary operators can be substituted for "op".

- **An inner product operator I(op):** If a and b are vectors of equal length, a I(*) b generates a1*b1 + a2*b2 ... an*bn.

- **A matrix * vector operator:** This can in fact be built up from the previous operators.

- **A matrix division operator** so that a(vector) / M(matrix) produces Inverse(M) * a, for M square and the same size as a. This can in fact be expanded to find least square solutions if M is rectantular.

- **Plotting:** Obviously you should be able to plot any vector against any other vector, with control of format as in any reasonable plotting package like MONGO or GREG. Higher dimensional structures can be displayed on TV devices.

- **Curve fitting:** Given optionally a mask vector, you can fit a polynomial function of given order to a given vector of equally spaced samples. Optionally you fit a polynomial given a vector of independent coordinates, and a vector of dependent coordinates (the function values).

You can find a number of other useful functions in (for example) the IDL manual. Some statistical facilities should also be supplies. To use these numerical facilities, one needs need i/o facilities with respect to AIPS++ internal, and external data structures. Thus:

You can copy (pieces) of any AIPS internal file into a command language variable. This includes subsets of maps in an obvious fashion. Something like a = getdata (mapspecification, [1, 5; *]) should create a two dimensional array a in the command language, consisting of rows 1 and 5 of the specified map. With a more complicated syntax, pieces of UV data should also be accessible. It should be possible to read any column of an AIPS table into a command language variable.

You can also read any external list (a file containing data in character string form) into a command language variable, as in GREG or MONGO.

You can write command language variables into AIPS structures, reversing the above processes. For creating a new AIPS file, this requires creating a dummy header, or copying one from an existing file.

Some conversion tools to external data base structures (e.g. for TABLES to LOTUS) should be available to allow the user to use these programs.

B.2 Data base and catalogues

A simple relational data base should be available to the users. You should be able to copy AIPS tables into such a data base, if you haven't written them in that form in the first place. Columns from these data bases can be read into command language variables. The result of all of this is that you can issues a straightforward query to select data from one or more tables, then operate upon this data numerically and display it.

The user data catalog can be a restricted version/view of this data base. Thus the naive user will see a limited number of fields in a simple list. For instance (name, qualifier,type, sequencenumber) which could all be strings, or perhaps sequence is a number. Precanned queries allow listing of all entries of given qualifier or type etc. (This is equivalent to but more general then the current system.

Further relations/tables contain implicit or explicit connections between files. Thus if a "Source File" is created from a map: in the first place an entry is main in the main catalog, with type="SOURCE". Then in the connection table an entry (MAP, SOURCE) is made, connecting the two. The connection table allow general/selective deleting or copying of all the connected files.

While connection entries are made implicitly when "extention" files are created, they can also be created explicitly by the user. In this way she can, say, connect all files related to ORIONA, and then treat these as a unit, at least as far as copying and backup operations are concerned.

To a yet to be determined degree, it should be able to give a task a query field instead of a file name as input. The task front end should perform the query, derive a list of files, and operate on the list sequentially.