AIPS++ and the GBT

Robert W. Garwood

National Radio Astronomy Observatory[1], *520 Edgemont Road, Charlottesville, VA 22903*

Abstract.

AIPS++ is an astronomical data analysis system currently being developed. The Green Bank Telescope (GBT) is under construction in Green Bank, West Virginia. The GBT is expected to be operational in mid 1997. Over the past several months, the AIPS++ group has been working to support the initial tests of the monitor and control software for the GBT. These tests involve placing the NRAO 140-ft. telescope and all data taking hardware under control of the GBT software. This paper describes the AIPS++ support of these ongoing tests. This support includes Glish (a command-line interface and process control hub), a near-real-time data filler, the AIPS++ Table system and some preliminary analysis tools. Real astronomical data has been obtained using GBT software and examined using AIPS++ tools. The future possibilities for AIPS++ single dish support, both for the GBT as well as other single dish telescopes, are discussed.

1. Introduction

AIPS++ is a large astronomical data analysis system currently being developed using object-oriented programming techniques. The development is being done by a consortium of seven radio astronomy observatories from six different countries. This effort is described in more detail in Glendenning, 1996, and through the AIPS++ home page[2].

The Green Bank Telescope (GBT)[3] will be the largest fully steerable radio telescope in the world. Construction is expected to be complete in 1997. The primary reflector is a 100 by 110 meter unblocked parabola. Each of the 2004 panels of the primary reflector can be moved in and out by computer controlled actuators.

A series of tests involving GBT hardware and software have been conducted since mid-1995. For these tests, GBT hardware (primarily receivers) is placed on the NRAO 140-foot telescope and the telescope is controlled using GBT

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

[2]http://www.nrao.edu/aips++/docs/html/aips++.html

[3]http://www.gb.nrao.edu/GBT/GBT.html

software. AIPS++ tools are used to examine and analyze the resulting data. This paper describes those tools.

The original goals of the tests were to demonstrate the integrated operation of GBT systems (electronics, monitor and control, data analysis), to perform observations that will be needed during the initial calibration and operation of the GBT and to use a working telescope for these tests. Within the AIPS++ project, the tests also serve to demonstrate the suitability of the AIPS++ infrastructure to meet real astronomical demands, to gain experience with the individual AIPS++ tools particularly with respect to the interactions between tools, and to meet critical target dates. The tests also provide an opportunity for users to give early feedback to developers.

Three test sessions have occurred so far. In July of 1995 the new digital continuum receiver (DCR) was completed. The telescope and DCR were used under coordinated control. The telescope position, weather, and other data associated parameters (DAP) were available in FITS binary tables format. The rudimentary continuum data analysis and display tools were sufficient to deal with the planned observing tests (atmospheric emission/opacity tipping scans, system temperature determination, cross-scan pointing offset and aperture efficiency calibration, and a raster scan beam map with evenly spaced scans). A second similar round of tests occurred in August of 1995. In October, 1995, telescope receiver focus and rotation were under computer control. More complicated observing capabilities available then included antenna position switching, on-the-fly mapping using repeated rapid raster scans, focus determination, and simple pulsar and polarization observations with the spectral processor.

The tests have been extremely successful. Within about four hours after the start of the first test session real astronomical data had been taken using GBT hardware, GBT control software and AIPS++ analysis tools. This was the first time all of these components had been used together. The first scan is shown in Figure 1. The key to our success continues to be good and frequent communication between the AIPS++ and GBT groups. Examples from these tests can be found in the AIPS++ web pages[4]. The AIPS++ tools described here are by no means a final product. All of the software at this point is rough, ragged, and not for the novice. However, the system has proven to be quite flexible and we see a clear path for improvement.

2. AIPS++ Tools

The GBT monitor and control[5] software was developed using object-oriented design techniques. One of the key features of the software is that it localizes real-time dependencies within separate sub-systems. Data collection is done by independent, autonomous subsystems which are synchronized by initial setup operations. Each subsystem (e.g. weather, timing, position, receivers, etc) writes data to disk as a separate collections of FITS files. These files are currently the

[4]http://www.nrao.edu/aips++/docs/sddemo/101/gbt.html

[5]http://info.gb.nrao.edu/GBT/MC/doc/GBTmc.html

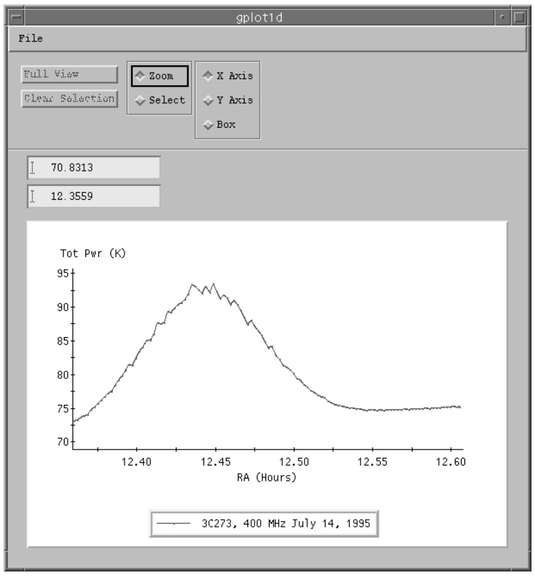

Figure 1. First Source Scan: GBT Monitor & Control, GBT Digital Continuum Receiver, AIPS++, NRAO 140-foot.

point of contact between the GBT monitor and control system and AIPS++ tools.

2.1. AIPS++ Tables

AIPS++ Tables are the fundamental storage mechanism for AIPS++. They are based upon ideas originally proposed by Allen Farris (see also van Diepen, 1995). A Table consists of a number of rows and columns. Keyword/value pairs may be defined for the Table as a whole and for individual columns (e.g. the units of a column in a Table may be defined using a keyword/value pair attached to that column). A cell in a Table may contain a scalar, a "direct" array, an "indirect" array or another Table. A direct array must have the same shape in all cells of a column while an indirect array may have different shapes in different cells. The ability of a cell to contain another Table allows a Table to express arbitrarily complex data relationships. A column in a Table may be "filled" or "virtual". Filled columns contain actual data while virtual columns contain a recipe describing how the data is to be generated (possibly using other columns in the Table, a model calculation, etc.).

An important concept in the Table system is that of a data manager. Data managers handle the reading, writing and generation of data. Each column can be assigned its own data manager, which allows optimization of the data storage mechanism per column. Virtual columns are a type of data manager. The actual form that the data takes behind the scenes (either on disk, or as output from some model or as a combination of values from other locations in the same Table) is determined by each data manager.

The Table system supports a full set of selection and sorting capabilities. The result of a select or sort operation is another Table which references the original Table. Rows or columns can be selected from a Table. Selection operators can be used to form arbitrarily complex expressions. Sorting can occur using one or more columns containing scalars.

2.2. The GBT Data Filler

The GBT data filler is responsible for presenting the data in the multiple FITS files produced by the GBT system as an AIPS++ Table. This currently means that the values are copied from the separate files into a single AIPS++ Table. Eventually, a FITS data manager will be available which will allow an AIPS++ Table to be constructed from the on-line data without actually copying any values. However, since that will not change the appearance of the Table to users it can be made without disrupting the rest of the system.

Each filled Table currently consists of data from a single backend with all of the associated data parameters (weather, telescope position, etc.) collected at the same time. Each row of the Table has a unique combination of timestamp, receiver, and integration phase (on, off, etc.). Since each GBT monitor and control process responsible for writing data to a FITS file is proceeding at its own pace, the data is usually not sampled at the same rate as other FITS files. The rate that data is produced by each receiver determines the rate at which rows are added to the filled Table. Associated data parameters appropriate for each chunk of receiver data are then filled into the table. This often involves interpolating values to arrive at an appropriate value for the time at which the receiver data was taken (e.g. telescope position).

A related filler is used to monitor the health of the various subsystems. This is used by the engineers in Green Bank to examine various engineering data logs. The logs are stored in FITS files in the same format as that used by the data filler. Engineers can then use the same tools to examine the engineering data that observers use to examine the astronomical data. There is no fundamental difference between the types of data.

2.3. Glish

Glish is the backbone of AIPS++ (Schiebel, 1996). It is an interpreted language with distributed clients. The language is loosely based on "S". The Glish interpreter acts as a central "clearinghouse" for forwarding events between processes (clients). Clients may themselves be instances of Glish executing Glish scripts. Glish allows for a loosely-coupled distributed system. Developers can develop Glish clients without regard to the rest of the system. Users can add new clients without remaking the entire system. Clients provide a limited, well defined set of services in order to maintain the distributed nature of the system.

Most of the AIPS++ tools described here are Glish clients, although some can be run as stand-alone programs (e.g. the GBT data filler). The arrangement between Glish, its clients and the data is shown in Figure 2. The monitor and control system drops data into several FITS files which are gathered together and filled (by gbtfiller) into an AIPS++ Table. Several Glish clients are available

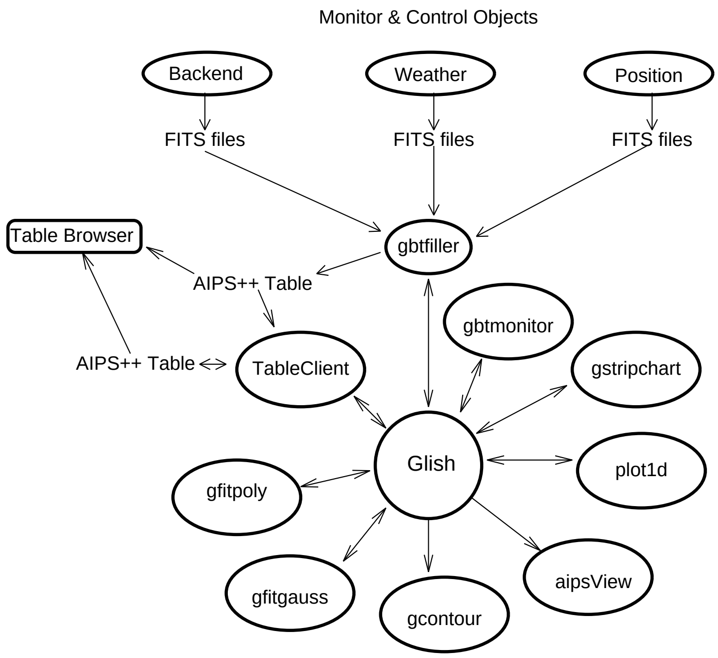

Figure 2. The Distributed System

to the user running Glish (including gbtfiller). The power of the Glish language is such that most of the actual analysis operations necessary for these tests were implemented using Glish functions (e.g. calibration, combining spectra, making simple maps). For these tests, AIPS++ provided the basic tools necessary to access and display the data as well as two tools to allow fitting functions to vectors. More complicated operations were accomplished using Glish functions written by the users in Green Bank. One advantage of Glish is that these functions can be replaced by Glish clients if that is desired for efficiency reasons without such a change affecting how the user interacts with the system.

2.4. The Table Client

The Table Client provides access to AIPS++ Tables though Glish. Tables can be opened read-only, read/write, and created from scratch using Glish commands. The Table Client supports most of the Table functionality available in the C++ interface. Values can be returned from the Table one cell at a time or, more commonly, as a vector of values from a column.

Figure 3 shows a simple example illustrating use of the Table Client and a few of the array manipulation features of Glish. In the first two lines, a sub-Table is selected from a previously opened Table, `OnLineTable`, having all values of the `SCAN` column equal to the value 1234. Four columns are then extracted from this sub-Table and returned as Glish vectors. A vector of booleans (`mask01`) is created having a true value at elements where the value in `rcvr` is 0 and the value in `phase` is 1 and false in all other elements. The length of this vector is the same as the length of the `phase` and `rcvr` vectors. The data and time values at elements that are true in the mask vector are extracted and become the

```
tableGram := paste("select from ", OnLineTable.file,
                   " where SCAN == 1234")
table := table_command(tableGram);
rcvr := get_column(table, "RECEIVER_ID");
phase := get_column(table, "PHASE_ID");
data := get_column(table, "DATA");
time := get_column(table, "Time");
mask01 := (rcvr == 0) && (phase == 1);
data01 := data[mask01];
time01 := time[mask01];
time01 := (time01 - as_integer(time01)) * 24.0;
plotxy(time01, data01, "Rx 0 Phase 1");
```

Figure 3. Table Client Example

elements of two new vectors. These new vectors have a length only equal to the number of true elements in the mask. The vectors are then plotted against each other using the `plotxy` function (which uses one of the graphics tools described in the next section).

Figure 4 shows one of the many Glish functions written for these tests. These functions were written for the most part by the observers and not by the AIPS++ group. In the future, this effort will be better coordinated and a more comprehensive set of analysis functions and clients will exist. This function itself uses several other Glish functions. The first line of the file returns a reference to a subset of the input Table which has all values in the scan column equal to one value (see also Figure 3). The `get_column` function returns the indicated column from the input Table as a Glish vector. In particular, the function in this example gets the data and time columns from the selected sub-Table. Then, two masks are created, one which is true for those elements corresponding to phase 1 and the other which is true for those elements corresponding to phase 2. The data are combined to yield the calibration of counts per degrees Kelvin. This value is used to convert the data into calibrated values with units of Kelvin. Finally, the data are plotted as a function of time. The two variables declared as global are available outside of this function. All other variables are only available within this function.

2.5. Graphical Tools

Most of the AIPS++ tools produced for these tests are for viewing the data graphically. They are built using Motif widgets and most are Glish clients.

A vector plotter was constructed using a commercial widget, XRT/graph (Shannon, 1996). A commercial widget was chosen for the initial development of a vector plotter because it was already being used by the GBT group. This enabled a more rapid development than adopting a public domain vector plotting tool would have allowed. The vector plotter functionality will eventually be taken over by the BIMA/NCSA group within AIPS++. This will involve non-commercial code.

```
plot_TP_time := function(const cal_value, const table,
                         const scan, const rcvr = 1)
{
   tab := select_by_scan (table, scan);
   data := get_column(tab, "DATA");
   time := get_column(tab, "Time");
   mask1 := get_rcvr_phase_mask(tab, rcvr, 1);
   mask2 := get_rcvr_phase_mask(tab, rcvr, 2);
   data_ph2 := data[mask1];
   data_ph2 := data[mask2];
   global counts_per_K := sum((data_ph2 - data_ph1) /
                           cal_value) / length(data_ph2)
   global cal_data := 0.5 * (data_ph1 + data_ph2) /
                           counts_per_K;
   time_ph1 := (time[mask1] - as_integer(time1[mask1])) * 24.0;
   plot_xy(time, cal_data, paste("Rx", rcvr));
   set_xaxis_label("UTC (Hours)");
}
```

Figure 4. Glish function to plot total power versus time for a specific scan and receiver

The vector plotter is a Glish client. In addition to plotting, this tool allows data selection. Data is selected using the cursor and returned to Glish when requested. The plotter tool evolved rapidly during the tests in response to user feedback. This rapid development was possible because of the distributed nature of the analysis environment that Glish offers.

A strip chart plotter was also built using the same XRT/graph widget. This is a Glish client that allows the user (or other Glish clients) to append data points to an existing plot. The x-axis of the display is time and the plot scrolls as new points are added. This is used as a data monitoring tool.

Some simple 2-dimensional scans on the sky have been made during these tests. Two tools are available to look at these simple images, a contour plotter and AIPSview (Crutcher et. al., 1996).

The contour plotter is a Glish client that has been constructed from a very limited set of PGPLOT functions. It allows for very simple 2-dimensional contour plots of FITS primary arrays, Glish arrays and AIPS++ Image objects. This functionality will eventually be taken over by the BIMA/NCSA group and integrated into AIPSview.

AIPSview is a stand-alone image display tool developed by the BIMA/NCSA group of AIPS++. It can also be used as a Glish client. It was used very briefly during the tests described here.

The final graphical tool available for these tests is a Table browser. This gives a graphical view of the contents of an AIPS++ Table. Whole columns can be selected and plotted. This is not yet a Glish client. The Table browser will be significantly enhanced and possibly largely rewritten during the first few

```
plot_TP_time(1.6, table, 3, 2)
plot_TP_time(1.6, table, 4, 2)

plot_gain_time(1.6, table, 3,2)

plot_TP_time(1.6, table, 3, 2); clear()
plot_gain_time(1.6, table, 3, 2); clear()
plot_xy(cal_data, counts_per_K, "" style="points")

avg_gain := sum(counts_per_K) / len(counts_per_K)
gain := 50.0 * (counts_per_K - avg_gain) / avg_gain
avg_data := sum(cal_data) / len(cal_data)
data := 50.0 * (cal_data - avg_data) / avg_data
cc := cross_cor(data, gain)
clear(); plot_xy(cc,"");

ccd := cross_cor(data, data)
plot_y(ccd,"");
```

Figure 5. Data Exploration Glish Fragment

months of 1996. It will be one of the most important components of the AIPS++ environment.

2.6. Other Glish Clients

Two additional non-graphical Glish clients were required for these tests. These are a client that fits a polynomial to the data for baseline removal and a client that fits multiple Gaussians to the data to evaluate drift scans across sources. For both of these clients, the data to be fit and the parameters to control the fit are sent to the appropriate fitting clients. Once a fit is reached, the fitted results are returned to Glish. The act of selecting the data, calibrating it, sending the data to the fitting client and displaying the fit and removing it from the data can be combined into a single Glish function.

2.7. Flexibility Example

The flexibility of Glish and the AIPS++ tools allows the observers to explore the data set in ways that hadn't been anticipated. During one of the test sessions the observers saw a puzzling result. They were observing 3C273 while tracking at the half-power point to evaluate the tracking accuracy of the telescope under GBT software control. The observed intensity variations were much greater than could be explained given the position errors from the encoders.

Figure 5 shows some of the data exploration that the observers did to attempt to determine what the problem was. All of the figures produced by this example are available within the AIPS++ web pages. The Glish variable `table` is a reference to an already opened AIPS++ Table containing the scans involved (one on-source scan, number 3, and one off-source scan, number 4). The first two lines in Figure 5 plot the data from both scans versus time for receiver 2. The difference in fluctuations seen in the resulting display is what the observers were trying to understand. In order to explore this problem, the receiver gain for the on-source scan was plotted for receiver 2. This indicated that the noise in the gain was too large to rule that out as the cause of the apparent variation. Next, a scatter diagram was generated by plotting the source temperature, `cal_data`, versus the gain, `counts_per_K`. Both of these values are global variables that are calculated in the two plotting functions. These values could easily have been made available through separate function calls that did not require any display. No significant correlation was seen in the scatter plot so the cross-correlation was calculated and displayed. Finally, the autocorrelation of the system temperature was calculated and displayed without clearing the plot client. This fragment represents a small part of the data exploration that the observers conducted in order to understand this problem.

This example indicates how most of the simple analysis needs of these tests were met using Glish functions. One further advantage of Glish is that Glish functions can be turned into Glish clients written in C++ or some other compiled language for efficiency reasons without changing how the user accesses those functions. The function calls remain the same but the internal implementation can change as desired.

3. AIPS++ Single Dish Development Plan

An intermediate-range development plan for single-dish analysis in AIPS++ is nearly complete. This development will be driven by the needs of the GBT and the Parkes Multibeam Project[6]. For the GBT, single dish development will be focused on providing a system that is sufficiently complete to allow for routine calibration and analysis of one-dimensional spectral-line data (external galaxy 21-cm. line profiles, molecular line searches, red-shifted HI in quasars, etc). The components of this system will be developed during the first half of 1996 and a beta version of this one-dimensional spectral-line analysis system will begin to be tested in July of 1996. The Parkes Multibeam project will begin observing in mid-1996. AIPS++ will provide support for the immediate on-line analysis of this data. A member of the Parkes team will work closely with the AIPS++ to provide much of this support. The final version of this plan will be available through the AIPS++ web pages by the end of December, 1995.

These plans include developing a protocol for storing and retrieving help information from Glish functions, clients and distributed objects. We have begun to develop a Glish binding to the Tk widget set. This allows users to treat each Tk widget as a Glish client and therefore to easily construct graphical tools intimately bound to Glish. A help system will be constructed early in the development.

The Tables currently being filled are quite telescope dependent. Within AIPS++, all data will be contained within MeasurementSets. A MeasurementSet is a specialized Table. The MeasurementSet attempts to remove or hide

[6] http://wwwatnf.atnf.csiro.au/Research/multibeam/multibeam.html

as much of the telescope-dependent nature of the data as possible. This primarily involves agreeing to standard column names and definitions as well as to a minimum set of required columns for specific types of data. The set of columns appropriate for single dish data will be finalized by the end of December, 1995. The current GBT filler will be re-written to fill a MeasurementSet by January 1996. A filler will be written for Parkes Multibeam Project data. Additional functionality as required by the GBT and Parkes will be added as Glish clients (simple routine calibration, Fourier transform and its inverse, cross correlation, convolution, interpolation, regridding, filtering) or though Glish functions that use those clients.

A longer range single dish development plan will begin to take shape in the spring of 1996. The focus will then shift to single dish imaging needs.

Acknowledgments. The work described in this paper is the combined work of the entire AIPS++ project in cooperation with the GBT group. Within the AIPS++ project, Brian Glendenning, Darrell Schiebel, Paul Shannon and Ger van Diepen deserve specific mention with respect to the work described here. Rick Fisher and Mark McKinnon were the two GBT users who provided useful feedback to the project and developed several useful Glish functions on their own. The GBT group includes Aron Benett, Joe Brandt, Mark Clark, Roger Norrod and Robert Payne.

References

Crutcher, R. M., Baker, P. M., Bakter, G., Pixton, J., & Ravlin, H. 1996, this volume

Glendenning, B. E. 1996, this volume

Schiebel, D. R. 1996, this volume

Shannon, P. 1996, this volume

van Diepen, G. 1995, in in ASP Conf. Ser., Vol. 61, Astronomical Data Analysis Software and Systems III, ed. Dennis R. Crabtree, R. J. Hanisch & Jeannette Barnes (San Francisco: ASP), 417