AIPSLETTER

Volume XIV, Number 1: January 15, 1994

A newsletter for users of the NRAO
Astronomical Image Processing System

Written by a cast of *AIPS*

Edited by
Eric W. Greisen
National Radio Astronomy Observatory
520 Edgemont Road, Charlottesville, VA 22903–2475
804–296–0211, x209; TWX 910–997–0174; Fax 804–296–0278
`aipsmail@nrao.edu`

The Good News ...

The `15JAN94` release of Classic *AIPS* is now available. Contact Ernie Allen at any of the addresses given in the masthead to obtain a copy. As of this writing, 94 copies of the `15JUL93` release have been given out electronically (46 tar.Z, 15 tar.gz) or on magnetic tape (20 8mm, 5 4mm, 4 QIC, 4 9-track). The ports of *AIPS* to DEC's alpha computer (OSF/1 operating system) and HP's 9000 series computers (HP-UX operating system) were cleaned up during the last six months using machines loaned to us by the vendors. A new port to personal computers (80386 and 80486 architectures) using the Linux (a public-domain UNIX-like, *i.e.*, not DOS) operating system was developed for this release as well. (See the article later in this AIPSLETTER.) In addition to the ports, we have written a number of new tasks, enhanced the imaging and Cleaning tasks, improved access to helpful information, and even improved the *POPS* language for the `15JAN94` release.

The NRAO is advertising a support scientist position initially in the Classic *AIPS* group at the Array Operations Center in Socorro, New Mexico. If you are interested, or know anyone who might be interested, please contact Gustaaf van Moorsel at the NRAO in Socorro (505–835–7396 or `gvanmoor@nrao.edu`). A position for a real-time programmer in the VLBA Correlator group is also available. In addition to the new position, we have begun to receive assistance with VLBA-related software from Leonid Kogan at the AOC. Also, Jeff Uphoff, who did the initial port of *AIPS* to the PC while a student at Virginia Tech, now has a student-level position in Charlottesville and spends some of his time on *AIPS* matters.

... and the Bad

The new position at the AOC is needed because Gustaaf van Moorsel has been promoted to Head of the Computer Division at the AOC. While this may be good news for Gustaaf, it means that the *AIPS* Group will get very little of his time in future. He will continue to look after `FILLM` and prepare our weekly meeting notes. We congratulate Gustaaf on his promotion, and wish him well in his new position. In Charlottesville, Bill Cotton has resigned from the *AIPS* Group and moved from the Computer to the Scientific Services Division. It is impossible to overestimate Bill's contribution to Classic *AIPS* and he will be sorely missed. (He still advises your Editor concerning which rocks should be overturned in the search for bugs, fortunately.) We wish him luck with the VLA D-Array survey and his other projects.

NRAO's last Convex, the C-1 called yucca, will be decommissioned during the first quarter of 1994. It has not been used much for *AIPS* in some time, and was not used to test the current release. Although the C-1 is no longer an interesting computer in itself, Convex makes very powerful computers used at other *AIPS* sites and we regret that our support for this architecture will not be at its previous level.

Improvements for Users in 15JAN94

VLBI data formatting and fringe fitting

The task `FITLD` not only performs the functions of `IMLOD` and `UVLOD`, but also acts to convert the FITS format used by the VLBA correlator into the internal $\mathcal{AIPS}$ *uv* format. It has acquired also the jobs of correcting for numerous foibles of the correlator. `FITLD` has been substantially improved for this release. It now is better at handling premature termination of the data, changed antenna numbers during concatenation, crossing of midnight, VLBA lower side band data, data out of time order, global data scaling, and reordering of baselines. It has substantially improved table handling routines.

The fringe-fitting program `FRING` was corrected in its initialization for the multi-band delay solution, in its handling of of small numbers of channels and IFs, and in its choice of solution interval for short data scans. The baseline-based fringe-fitting task `BLING` was given numerous improvements. It uses improved fitting, minimization, interpolation, and initial estimation methods. It now offers a display of the degree of coherence of the fringes. A number of options were added to provide access to convergence criteria and to control deletion of scans where interpolation fails. The help and explain files were improved. The task `BLAPP` applies the results of `BLING`. It was given options to control the weights and the application of the solutions to the CL table and had a couple of bugs exterminated.

Lesser changes include adding a multi-band delay correction option to `CLCOR`. A minor error in the times used to plot models in `CLPLT` and `VBPLT` was corrected. The warnings about reference day errors in `MK3IN` were strengthened. And the `VLBA` procedure was speeded up by moving where the spectral averaging is done. It was improved to give the user the choice of display devices and of deleting or keeping intermediate results.

UV data manipulation and display

Three new tasks of relevance to *uv* data appear in the `15JAN94` release. `RFI` searches for interference by looking for high values of the amplitude rms. It does a good job of finding regions of bad interference, which is far more prevalent at the VLA at L band than many people realize. The task `UVCRS` looks for places in the *uv* plane where tracks cross and compares the measured, calibrated data from the baselines involved. It determines gain corrections to apply to make the data agree at the crossing points. Finally, `CCEDT` edits a Clean Components file keeping only components inside specified boxes above a specified cutoff flux (after merging). It is used to improve source models before self-calibration.

Significant changes and improvements were made to `DBCON` and `SPLIT`. `DBCON` was improved to correct the times in the flagging table of the second file to match any change in the apparent time (days) made in the concatenation. It was also improved to offer the option to correct phases for position shifts in a correct frequency-dependent manner. `SPLIT` was given the option to average data across IFs in addition to the previous option to average across spectral channels. It was corrected to use the frequencies in the source table and was given the option to store the full 16-character source name as keywords in the data header.

Less visible changes included raising the maximum number of sources allowed in a data set from 300 or 500 to a parameterized 2500. `UVDIF` was given the ability to detect the presence of sort-order swaps to avoid treating all such as serious differences. `SNPLT` had a number of minor annoyances in its display and handling of empty baselines corrected. The OOPS *uv*-data handling routines were corrected to determine the correct actual frequency and *uvw* scaling to use in the presence of IF selection. This error had led to minor errors in the header frequency and image scaling, especially if IF 1 were not included.

Imaging

Substantial changes were made in the handling of data gridding and the Clark Clean algorithm for the current release. These were driven by experiences with the DDT tests; see the article below for those aspects of the matter. Detailed examination of a number of residual images revealed high-frequency (about 1-cell from plus to minus) oscillations. These are caused by data gridded into the outer parts of the *uv* plane. The cell size may appear to be adequate for the fit beam width, but that beam width relates primarily to the inner portions of the *uv* plane where there is the most data. The outer portions of the *uv* plane are not adequately sampled in the image plane and, hence, cannot be cleaned adequately or correctly. The new adverb GUARD allows the user to define a "guard band" around the outside of the *uv* plane in which data will be discarded (with a warning message) rather than gridded. This is a serious problem and users are strongly encouraged to use this option rather than to shut it off. Actually, what users really should do is choose smaller cell sizes and, possibly, larger image sizes so that no data samples fall into, or even near, the guard-band area.

During the Clark Clean, the histogram of the residual image is taken and only points in the upper few cells are considered for components during the next major cycle. It was found that the histogram was too coarse for large images, so that only a few cells were used rather than a more reasonable number. All imaging tasks were given much finer histograms. The maximum number of cells to be "loaded" and searched for components in any major cycle controls how well and how fast the Clean progresses. If too many are loaded, then many cells are searched and subtracted and never actually selected as having a component. If too few are loaded, then the major cycle has to cut off very early and the expensive processes (model Fourier transform and subtraction, FFTs, etc.) of a major cycle are repeated too frequently. The adverb MAXPIXEL was added to allow the users to control the maximum number of cells to be searched. For small images, a small number would be appropriate; for normal images, the default (20000) is okay; and for large images with extended emission (*e.g.*, Cas A), a rather larger number would be profitable. The Cleaning of each component inside a major cycle is more accurate when more points of the dirty beam are used. All imaging tasks now allow for, and use when possible, a rather larger "beam patch."

The beam width fit to the dirty beam was found to have a small but significant error. This had affected the relative units of the restored components and the residual image and, hence, is of interest. WFCLN offers the option to determine and correct for the relative scaling of the clean and residual portions of an image. WFCLN received a number of corrections to its handling of default values and channel averaging.

Two new tasks appeared in the 15JAN94 release to determine and correct for the variation of instrumental polarization across the single-dish beam. MAPBM uses a VLA raster made in holography mode to compute an image of the beam in the four Stokes parameters. Then VLABP applies this image to sets of I, Q, and U images (as separate images or as a polarization cube). Note that this applies only to snapshot images, where the instrumental polarization has not been smeared by rotation of the parallactic angle.

The option to specify a Gaussian width to the zero-spacing in task MX was removed since the implementation available was too approximate and misleading. Other means should be used to add short-spacing data to a *uv* data set. One possibility would be to use IM2UV on a low resolution image, flag the outer spacings with UVFLG, and then concatenate the short-spacing data with the high-resolution data set. This would allow reasonable weighting and modeling of the short-spacing model, things not allowed by the previous option.

Image handling and display

Walter Jaffe's contributed task SAD (for search and destroy) was substantially revised and generalized for use by the VLA D-Array survey. It searches through an image for regions of flux above a cutoff level and then attempts to fit one or more Gaussians automatically to each region. It now prepares an elaborate MF file containing the complete results of the fitting in such a form that SAD can be restarted with all previously fitted sources removed (and presumably new windows, cutoffs, and the like). SAD can write a residual output file, can add components to a CC file, and uses the line printer to display its results. Provision for polarization has been made in the MF file, but the task to do that part of the fitting has yet to be written. Walter's task CCNTR was enhanced to plot either CC or MF components on top of contours using all the plot symbols available for plotting of "stars."

A new verb, EPOSWTCH, was added to change the coordinates in the image header between B1950 and J2000. Such a conversion is of necessity not exact, but should be sufficient for most non-astrometric uses.

Velocity field analysis task MODVF

The new *AIPS* task **MODVF** (MODel Velocity Field) does what its name implies, using a tilted ring model. This model is specified in a text file, which is read into the task using the adverb **INFILE**. This file contains information about the rotation velocity in each ring and its warping with respect to the central portion of the galaxy — the reference disk. The warping geometry used in **MODVF** is described in the paper "Kinematical modeling of warps in the HI disks of Galaxies" (Christodoulou, *et al.*, 1993, *Ap. J.*, **416**, 74-103). The advantage of this approach is that warps are described in the geometry of the underlying galaxy; instead of using the observer-dependent position angle and inclination, the warp is specified in terms of the angular momentum vector of each ring. This approach makes it much easier for users to test their favorite warp model against actual observations.

New history and help verbs

The **AIPS** verbs dealing with history files were changed and enhanced for this release. **PRTHI** now uses adverbs **HISTART** and **HIEND** to control which lines are printed. **HINOTE**, which adds text to a history file, now has new formatting controls and can be used to create a history file is the old one went missing. The new verbs are **HITEXT** to write out portions of the history file to a text file and **STALIN** to delete portions of the history file with a **DOCONFRM** option.

There are two new verbs to help users find things in *AIPS* now that it is so large. Both of them use the one-line description and the one-line category list typed into help files by the programmers. We have developed programs to build the necessary secondary files from the 946 (±) help files in the system. The first of the verbs is **ABOUT** which lists the one-line descriptions from all help files in the user-specified category (*e.g.*, **ABOUT EDITING**, **ABOUT VERB**, **ABOUT TAPE**). The second is called **APROPOS** and lists all one-line descriptions and category lists for all help files containing the user-specified keyword or keywords (*e.g.*, **APROPOS CLEAN** to display all with words beginning with "clean" and **APROPOS UV,PLOT** to display all that contain both words beginning with "uv" and words beginning with "plot"). The help files for both **ABOUT** and **APROPOS** contain several useful examples and the help file for **ABOUT** gives the complete list of recognized categories.

Magnetic tapes

A new task, **TCOPY** appears in the **15JAN94** release. It copies selected contents, in any format including ones otherwise unknown to *AIPS*, from one magnetic tape to another. It is designed particularly to copy many tapes to one tape, an operation which is difficult to do properly with UNIX utilities. Disk FITS files are also supported.

For the **15JAN94** release, many *AIPS* tape functions were made more efficient by removing excess tape operations where possible. This may be a little less safe, but certainly speeds up the writing of ends-of-file at the beginning of **FITTP** and after each FITS file. An extra back-file 0 operation was added at the end of **FITTP** to keep IBM (and perhaps other) systems from appending an extra, unwanted end-of-file mark after the one **FITTP** had just written. The default blocksize in **FITTP** was changed to 10, which is more efficient than smaller blockings. The dangerous adverb **DOTABLE** was removed from **FITTP** and the **DONEWTAB** adverb was changed to refer to the name convention used on FITS binary tables. Older releases of *AIPS* only recognize the non-standard name "A3DTABLE" while the now-standard name for such tables is "BINTABLE". **FITTP** was also corrected to keep the coordinate projection information on u, v, and w axes. One should also use **DONEW = FALSE** for this change when carrying data to older releases of *AIPS*. A bug in **FITTP** that caused it to write two consecutive ends-of-file when no data were found was corrected.

PRTTP was also improved by having it recognize the binary table *uv* format used by the VLBA and to display various things accordingly. It also acquired the option to do a very brief display to a text file (with **PRTLEV = -3**). **PRTTP** is currently undergoing development to make it more robust against errors and to have all of its error display appear in the printer file. These improvements were not complete in time to make it to the **15JAN94** version, but will probably be offered in the patch area for **15JAN94**.

POPS language processor changes

Two new pseudoverbs were written to declare verb numbers for verbs and pseudoverbs. This will allow new verbs and pseudoverbs to be declared in the `NEWPARMS` run file and thus to let users keep all their environments (`SAVE/GET` files) intact. Use of `RESTORE 0` should now be optional under all normal circumstances. The `RUN` pseudoverb was corrected to use the `VERSION` adverb correctly and to give real precedence to the `$RUNFIL` area (over the `$RUNSYS` area). A new verb `CLRTEMP` was written to clear the temporary literal area of all scratch numeric and string data. This verb can be used (carefully) in long looping procedures to keep the use of character strings, substrings and concatenation from gradually eating up all of the available space. If you get a "`BLEW TEMP C`" message, then you need this verb in your procedures.

Adverb values are no longer found through a complex Fortran equivalence which depended on an exact match between a Fortran include file and the `POPSDAT` file from which the POPS environment is defined. Instead, the language routines are used by subroutines `ADVERB` and `ADVRBS` to get and set, respectively, the adverb values. The language processor was corrected to enforce an upper limit of 132 characters for character strings and to handle such long strings appropriately in such verbs as `INPUTS` and `PRINT`. Subtle bugs affecting `IF` and `SUBSTR` statements were also corrected. All verbs and pseudoverbs were renumbered and rearranged in the `POPSDAT` file to make space for new ones and to make the lists easier to read and process. Thus, all users will have to do one more `RESTORE 0` when they begin using the `15JAN94` release and will need to replace all of their old `SAVE/GET` files.

Miscellaneous changes

Other corrections and improvements made to *AIPS* for the `15JAN94` release include:

batch Batch jobs now run at a new, much lower priority under UNIX systems. They will use all the available cpu, but not interfere at all with interactive usage of the computer.

printing All printer output now shows the host and release names at the top of each page. In this way, users may determine which computer and release actually produced their hard-copy output.

printing The deletion of temporary printer and plotter files is now done by a detached process after a time delay. This should reduce the incidence of lost files due to overburdened printer/plotter queues.

plotting Plot files are now selected for display by the new adverb `PLVER` rather than the over-used adverb `INVERS`.

HOLGR Numerous options and corrections were made to this antenna holography task.

UVNOU Renamed from `UVNOV`, this useful little task deletes data near $u = 0$, where the fringe rate is zero, which may therefore be corrupted by offsets in the correlator and by terrestrial interference.

Improvements Primarily for Programmers in 15JAN94

Coding standards enforced

The subject of coding standards in *AIPS* tends to evoke reactions of either extreme boredom or extreme anger. (For those who have tried to forget, the basic coding standards may be found in *Going AIPS, Volume 1* and the ANSI Fortran standard.) Nonetheless, they are there for a variety of reasons including the need to get through the wide variety of Fortran compilers encountered by *AIPS*. The general items of standards we encountered, and at least began to correct, in this release were:

- **Unused variables:** The enormous number of unused variables in some programs actually caused at least one compiler to fail. In addition, unused variables are reported by compilers to allow us to detect typographical errors and the like. If these messages are taken to be "normal", then the abnormal ones, which we need to detect, will be overlooked.

Level	Meaning
0	Message to file only (user input)
1	Message to terminal only (instructions)
2	Progress messages, input parameter display
3	Nature of computation, lesser informative messages
4	Interesting, useful messages
5	Significant answers, results of computations
6	Minor error messages, warnings (suppressible)
7	Probably serious error messages, warnings (suppressible)
8	Serious errors
9	Major structural failure
10	The building is on fire or under nuclear attack

Table 1: $\mathcal{AIPS}$ Message Level Meanings

- **Routines in wrong subroutine directories:** The Solaris 2.x operating system, which many of us will have on our desks in the next few months, will not run a load module if there are any unresolved external references in the shared libraries. This is a departure from past practice which allowed load modules to be executed so long as the unresolved references were never called — a useful debugging tool among other things. If, for example, a subroutine calls a `Y` routine, then it must be in `$YSUB` or `$YNOT` and all tasks that call it must be in the appropriate `Y` program areas. If this routine were placed instead in, for example, `$APLSUB`, then no program in $\mathcal{AIPS}$ except those in `Y` program areas would run under Solaris, at least with shared libraries. Several such examples were found and corrected. In another case, the subroutine `VERBSC` had to be moved from `$AIPSUB` into the program file `AIPSC.FOR` itself, since it called routines that were only found in that file.

- **Interchanged character and non-character variables:** Fortran `CHARACTER` variables are fundamentally different from all other kinds since they have both a length and a data area. Under VMS, this is implemented as a pointer to a data structure. UNIX systems, however, assume that you might be calling C routines, and expand the `CHARACTER` argument to a routine into two arguments, a length argument and a data argument. Many UNIX systems replace the `CHARACTER` variable with the data and add the length argument at the end of the list of arguments. However, some use other, equally sensible, arrangements. Therefore, even if a `CHARACTER` variable in a particular call will not be used, it is not acceptable to substitute a non-character dummy variable. To do so will change the number of arguments to the called routine from the number expected by it and might even change their position in the call from that expected. The `f2c` used in compiling on PCs was particularly useful in detecting the rather large number of call sequence mismatches involving character variables, and a number of erroneously missing or inserted arguments of other kinds as well.

- **Sloppy precursor comments:** $\mathcal{AIPS}$ is too large to maintain all documentation by hand, and has been for many years. We have programs to generate "shopping lists" of subroutines for programmers. For this release, we also developed user verbs called `APROPOS` and `ABOUT` to make help-file information available to users by keywords and categories. We created tools to build the necessary files, but found that the precursor comments in many of the help files were poorly worded and lacking adequate keywords, or in some cases, plain wrong. The usefulness of these tools to users and programmers is clearly limited by the (lack of) quality of the precursor comments. We are gradually improving them, but it will be a slow process.

- **Incorrect message levels:** The message levels used as the argument to `MSGWRT` have meaning and may be used for `PRTMSG` displays which select just the desired types of messages. When programmers misuse the levels, then, for example, displays of error messages show lots of minor informative things and alarm people running the `DDT` test. We have corrected the tasks used by `DDT` among others to the message level meanings given in Table 1.

Miscellaneous changes for programmers

The DDT test has been changed and new master data files computed (except for the HUGE test). The nature of these changes is described in a separate article below. As part of this process, the handling of the pseudo array processor has been changed. The size of the "AP" memories is set, for Fortran compilation, by the new parameter include file(s) $INCS/PAPC.INC. The OS-dependent version of this file is examined by ZDCHIN after it reads the $\mathcal{AIPS}$ Manager set "secondary AP size" parameter from the system parameter file. The lower of the two values for the AP size is then used. This means that a large AP can be compiled into the code, but only a portion of it used. One can also, for example, compile a large AP size into the TST version and a smaller size into the NEW version of $\mathcal{AIPS}$ and the correct AP size will be known for both versions even though they use the same system parameter file. The default AP size is now 5 Mbytes. If you have less than 32 Mbytes of memory, you might consider editing the file to use a smaller AP at your site. Numerous examples are given in the file. We have changed all "array processor" programs to use as much of the AP memory as they can reasonably do. This is rather restricted for the Clark Clean algorithm (under control of a new user adverb MAXPIXEL) and quite large for most other algorithms. Pseudo-AP code must not use floating point variables to store addresses into the AP memory. This becomes inaccurate (in IEEE floating point) at 16 Megawords and quite a lot sooner in other floating formats. We corrected a number of routines for this usage (which was once required by the rolling of true array processors to disk).

The object-based programming package received numerous improvements and corrections. All memory needed for objects is now allocated dynamically (using a C routine) and deallocated when the object is closed. This raises the limit on the number of simultaneous objects to around 1000 rather than the previous 20. A private D-array survey program which can run for weeks at a time has been used to test for "memory leaks" and we believe no more remain. A printer class was added for this release. Routines which will correct polarization in a spatially-dependent fashion across the single-dish beam were added. The UV data access routine UVGET was given an added parameter INITVS to specify the first visibility desired in single-source data sets.

The coordinate location common was made multi-dimensional, so that more than one image may be referenced at a time simply by changing a single index variable. This should speed up the conversions done in HGEOM, OHGEO, and the like. The abort handling routine ZABORS.C in its numerous versions was changed to die cleanly on recursion rather than simply returning. Some infinite looping occurred in bad task deaths otherwise. Two new procedures were added to help programmers find things. PROG is like WHICH to find files by name, but looks only in program directories. GREP does a case-sensitive `grep -l` through whole directory trees. List the procedures (from $SYSUNIX) to see the full range of options and grammar.

Ports to new operating systems

A number of annoying problems with the Solaris 2.x operating system were uncovered during the past six months. Some of them are mentioned in the article on patches to the 15JUL93 release (later in the AIPSLETTER), including minor C errors in ZDAOPN, missing quotes in ZLPCL2, and incorrect units in ZFRE2. In addition, errors in setting buffer sizes in ZVTPX2 and ZVTPX3 and errors in setting the number of adverb values in CNTR, GREYS, PROFL, and MULTI affected all systems, but were serious only on Solaris. Hewlett-Packard loaned us a model 9000/755 for a few weeks, which allowed us to test and correct a variety of things. Its C compiler seemed to have trouble with having two subroutines in one file, causing us to split out ZIGNAL from ZABOR2 in $APLUNIX. We corrected XAS to pick out and use the default visual — HP offers two pseudocolor visuals in the basic window and the second is the default. We also tested out and improved the basic tape handling routine ZTAP2. DEC has loaned us an Alpha 3000/300 which we used to complete a port to the OSF/1 operating system and which we continue to use to support that system. The port was relatively easy and we now use $APLDEC to mean OSF/1, with a subdirectory for ULTRIX below it. Routines that were cleaned up and improved included ZFRE2, ZMOUN2 , and ZTAP2 — the usual three.

The other new system for us is Linux — a public-domain UNIX-like operating system for personal computers (80386 and 80486 architectures). Again, the port turned out to be relatively straightforward. A couple of $APLUNIX routines had minor changes to allow for different spellings of include file names. The Berkeley versions of ZDELA2 and ZTQSP2 are needed in $APLLINUX. The tape Z routines have not been developed since there are a great many tape devices for personal computers, but none has emerged as a usable standard. The remote tape and FITS disk routines work well, however. The port revealed problems in our Fortran usage (see above) and a bad call sequence to XShmDetach in XAS. See also the article on this port below.

Computer		$A_m^{(93)}$	$AL_m^{(93)}$	R	T_{LARGE}	T_{MEDIUM}	T_{SMALL}
IBM RS/6000 (580)		3.62	3.94		1104	323	184
IBM RS/6000 (560)		3.29	3.15	4.6	1215	357	193
HP 9000/755		3.06		7.1	1306	337	205
Sun 10/512MP (2.2)	*	2.45		4.0	1630	448	179
DEC Alpha 3000/300		2.10	4.83	4.4	1908	501	267
HP 9000/735	*	1.82	2.71	7.1	2199	546	200
IBM RS/6000 (530)		1.50	1.56		2674	683	328
Sun IPX (4.1.2)		1.01	1.00	1.0	3975	918	355
Convex C-1	*	0.99	0.49		4039	1319	770
Sun LX Solaris 2.3		0.95	1.27	1.0	4225	1018	405
Sun LX Solaris 2.2		0.91	1.32	1.0	4376	1135	453
Sun IPC Solaris 2.1		0.56		0.5	7135	1670	595
Sun IPC (4.1.2)		0.54	0.52	0.5	7375	1616	564
Gateway 486DX2-66V		0.51	0.75		7780	1579	508
DECStation 3100		0.38	1.01		10626	2399	849

Table 2: New `DDT` AIPSMark$^{(93)}$ Results

AIPS Verification Package Changed

The set of $\mathcal{AIPS}$ procedures and data files known as `DDT` (for Dirty-Dozen Test) has become a standard for verifying that basic $\mathcal{AIPS}$ tasks continue to function correctly and for measuring the performance of computer systems. We run it, and its younger siblings `VLAC` and `VLAL`, routinely every few weeks in Charlottesville as a continuing quality check on the developing software. Over time, however, the tests had become a bit "shop-worn." The worst problem was that `DDT` depended heavily on `ASCAL`, a task we no longer recommend to users. Also, as the software has changed, the "right" output data have also changed slightly. The accumulation of these changes made some of the current output images only agree to 9 bits or so with the previous "master" images. Because of this, minor programming errors were able to slip by undetected.

Initially it was thought that a couple of days in early June would be enough to effect these changes. However, de Sade's principle ("no good deed goes unpunished") struck. Eight months later, the full saga of changes is documented in $\mathcal{AIPS}$ Memo 85. `DDT` itself was changed to use `CALIB` rather than `ASCAL`, to use `FITLD` rather than `IMLOD` and `UVLOD`, to use IEEE floating-point in its FITS files, to support FITS disk files fully, and to use better imaging windows and cell sizes. The imaging tasks were revised to put correct shift parameters in the image headers, to limit the portion of the *uv* plane into which data are gridded, to allow the user control over the number of cells searched in each minor Clean iteration, to improve fits to the dirty beam, and to use the full pseudo-array processor memory wherever possible, including when that memory exceeds 16777216 words. Even the POPS language and task initiation got some improvements. In doing timing tests, numerous revisions were made to the ports to individual architectures as well.

The revised `DDT` runs faster than before. The new "$\mathcal{AIPS}$ Mark", called AIPSMark$^{(93)}$, is defined as

$$A_m^{(93)} \equiv \frac{4000}{T_{LARGE}} ,$$

where T_{LARGE} is the total run time in seconds of the LARGE test. (In the older test, including only 40% of the `ASCAL` real time, run times were around 5000 seconds for the IPX and Convex. The constant 4000 was chosen to keep the AIPSMarks for the Sun IPX about 1.0, as in the previous test.) The results for a number of architectures and operating systems are shown in Table 2. The rightmost columns give the total run times in seconds for the LARGE, MEDIUM, and SMALL `DDT` tests. The column labeled R is the ratio of the SPECfp92 measured for the computer to that of the Sun IPX. This and the column labeled $AL_m^{(93)}$ (defined in $\mathcal{AIPS}$ Memo 85) tend to measure cpu performance (relative to the IPX) rather than the total system performance measured by $A_m^{(93)}$. All machines were tested with all of the algorithmic improvements described above except those marked with an asterisk. The results for those three machines should be regarded as representative, but not precisely correct. (The HP 9000/755 was no longer available when `RTIME`, the task that measures $AL_m^{(93)}$, was written, but it should give about the same $AL_m^{(93)}$ as the HP 9000/735.)

It is interesting to note how, for some of the machines, general performance on a mix of normal $\mathcal{AIPS}$ tasks (*i.e.*, `DDT`) is well predicted by their SPECfp92 ratings (or AIPSLoopMarks$^{(93)}$) and, for others, it is not. Some of the machines were affected by poor I/O performance in the `DDT`, particularly the HP 9000/735 and the DECStation 3100. But, even for these machines, poor I/O does not explain all of the differences. Please do not base any procurement decisions on this summary article. Get and read the full details in $\mathcal{AIPS}$ Memo 85 first.

AIPS Joins the World Wide Web

Recently there has been a virtual explosion in the use of a new protocol on the Internet — that of the *World Wide Web* or *WWW* for short. It is basically a method for sending hypertext over the network. Clients such as *NCSA Mosaic* and *Lynx* have certainly helped this new method of information access. The astronomy community, as usual, is particularly well connected and has been quick to offer many "home pages," and NRAO is among the institutions working towards this. The current NRAO Newsletter has details on the NRAO home page, and the $\mathcal{AIPS}$ project also has one of these. The Universal Resource Locator (URL) is:

`ftp://baboon.cv.nrao.edu/pub/aips/aips-home.html`

If your system cannot resolve baboon's address, substitute `192.33.115.103` for `baboon.cv.nrao.edu` in the URL above. The information accessible here is mostly what was already accessible via anonymous ftp to baboon, but access to items such as the `15JUL93` patches and to $\mathcal{AIPS}$ documentation is considerably improved. Comments and suggestions for improvement are welcome; send these to `pmurphy@nrao.edu`.

Latest $\mathcal{AIPS}$ Memos

Below is a list of the latest $\mathcal{AIPS}$ Memos. Memos 84 and 85 are new in this AIPSLETTER.

MEMO	DATE	TITLE and AUTHOR
83	92/12/14	Dual Libraries and Binaries in AIPS Patrick P. Murphy, NRAO
84	93/11/12	A Proposed Package to Support the Use of the X Window System in AIPS Tasks Chris Flatters, NRAO
85	94/02/01	DDT Revised and AIPSMark$^{(93)}$ Measurements Eric W. Greisen, NRAO

Note that the version of $\mathcal{AIPS}$ Memo 83 now available has been modified slightly since it was distributed in December 1992. In addition, a heavily revised edition of $\mathcal{AIPS}$ Memo 78 (on Object-Oriented Programming in $\mathcal{AIPS}$) is available. It appears as file `AIPSOOF.TEX` and, in postscript form, as `AIPSOOF.PS`.

Many of the $\mathcal{AIPS}$ Memos and AIPSLETTERs may be obtained through the facilities of the World-Wide Web or by anonymous `ftp`. Since some Memos are not available electronically and others do not yet have computer readable figures, you may wish to write for a paper copy of these. To do so, use an $\mathcal{AIPS}$ order form or e-mail your request to aipsmail@nrao.edu.

To use `ftp` to retrieve the memos:

1. `ftp baboon.cv.nrao.edu` or `192.33.115.103`
2. Login under user name anonymous and use your e-mail address as a password.
3. `cd pub/aips/TEXT/PUBL`
4. Read `AAAREADME` for more information.
5. Read `AIPSMEMO.LIST` for a full list of $\mathcal{AIPS}$ Memos.

AIPS Memos from Number 65 through 85 are present in this area as are Numbers 27, 33, 35, 39, 46, 51, 54, 61, and 62. We have been filling in this list gradually, by finding and fixing old files in other areas of the authors' disks, by scanning in text and figures, or by retyping text and redrawing the figures. The AIPSLETTERs from 1991 through the present are also available in this area. Many of the Memos are in both TEX and PostScript forms, with the TEX ones stored in a subdirectory called `TEX`. Note that many, if not all of these may be found on your home *AIPS* system in an area called `$AIPSPUBL`. All Memos are available in paper form from Ernie Allen at the addresses in the masthead.

Of particular interest to users, the *AIPS* `COOKBOOK` will soon be available (in the form of PostScript files) in this area as well. Initially the chapters from the 1990 version of the `COOKBOOK` will be available; whenever one of these chapters is updated, the latest version will be available immediately in this area. We note that the chapter on VLBI processing in *AIPS* is slated for update soon, and updates for others are under consideration. All files in this area are, of course, immediately available via `ftp` and WWW.

Patch Distribution

Since *AIPS* is now released only semi-annually (or even less frequently), we make selected, important bug fixes and improvements available via *anonymous* `ftp` on the NRAO Cpu `baboon` (`192.33.115.103`). Documentation about patches to a release is placed in the anonymous-ftp area `pub/aips/`*release-name* and the code is placed in suitable subdirectories below this. Reports of significant bugs in `15JUL93` *AIPS* have been relatively few; however, the documentation file `pub/aips/15JUL93/README.15JUL93` mentions the following items:

MX `$QNOT/ALGSUB`, and `$QNOT/ALGSTB` used only the old and small part of the AP memory to decide how big to make the interpolation kernel for gridded modeling. For images of 4096 rows, this was too small and caused inaccurate modeling — namely some of the source did not get subtracted even though it was represented in the Clean components. `$QYPGNOT/MX` had some faulty logic affecting restarts which would cause it not to recompute images that required recomputation.

CNTR `$YPGM/CNTR`, `$YPGM/GREYS`, `$YPGM/PROFL`, and `$APGNOT/MULTI` give incorrect numbers of parameters to `GTPARM`, causing problems especially on Solaris.

System `15OCT92` (yes, the older version) `$SYSLOCAL` and `$SYSUNIX` files conflict with those of `15JUL93` if they are installed with the same $AIPS_ROOT area.

Solaris `ZDAOPN.C` had a C error which caused compilation to fail on Solaris systems. `ZVTPX2.C` and `ZVTPX3.C` had errors which caused the send and receive buffer sizes to be smaller than intended on all systems and zero on new Solaris systems. All of these are in `$APLUNIX`.

Solaris `$APLSOL/ZFRE2.C` used incorrect units, producing confusing results.

Solaris `$SYSSOL/LIBR` (and `$SYSLOCAL/LIBR` via `INSTEP1`) should have been marked as executable in the distribution tar file, but was not.

Solaris `$SYSSOL/ZLPCL2` or `$SYSLOCAL/ZLPCL2` on Solaris systems have a missing quote on line 31.

HP `$APLHP/ZTAP2.C` corrected to do an advance file after a BAKF operation; Affects HP systems only.

Note that we did not revise the original `15JUL93` tape or `tar` files for these patches. No matter when you received your `15JUL93` "tape," you must fetch and install these patches if you require them. See the `15APR92` AIPSLETTER for an example of how to fetch and apply a patch. Information on patches and how to fetch and apply them is also available through the World-Wide Web pages for *AIPS*.

As bugs in `15JAN94` are found, the patches will be placed in the `ftp` area for `15JAN94`. As usual, we will not revise the original `15JAN94` tape or `tar` files for any such patches. No matter when you receive your `15JAN94` "tape," you must fetch and install these patches if you require them.

AIPS On a Personal Computer!

AIPS is now available to people who cannot afford (or have limited access to) the other types of systems on which *AIPS* currently runs, but who do have access to a modern personal computer. The only PC operating system *AIPS* understands currently is Linux, which is a free (public domain) UNIX-like operating system available by anonymous `ftp` from **sunsite.unc.edu** (preferred) and **tsx-11.mit.edu**. See below for more details on versions of the bits and pieces you will need.

Initial benchmarks in November 1993 put *AIPS* on a 486DX2/66V at roughly the same speed as a Sun IPC (see the article on `DDT` earlier in this AIPSLETTER). A 486DX/33 runs the small `DDT` just over half as fast as a DX2/66, and a 386DX/33 (even with a 80387 installed) is so slow as to be nearly unusable in any serious applications (about 45 minutes for the Small `DDT` versus about 10 minutes on the 486DX2/66V).

The minimal and "comfortable" system requirements are as follows:

- A 386, 486 or pentium processor. A 486DX at 33 MHz is probably the lowest "usable" system. On 486SX or any 386 systems, make sure you have the math coprocessor.
- 8 Mbytes of RAM (more if you plan on running X windows; our benchmarks on the 486 were with 16 and 32 Mbytes). 16 Mbytes of swap space is also recommended; you can have multiple swap partitions/files and this will improve performance for larger problems.
- If you want to run X windows, a color monitor is recommended; the Linux X server supports many SVGA card/monitor combinations. The NRAO system we run has an ATI Mach_32 chipset accelerated SVGA card.
- For a full Linux/X11/*AIPS* installation, you will need about 100 Mbytes of disk for the Linux/X11 part, and about 170–200 Mbytes for all of *AIPS*, assuming you strip the `*.EXE` files. Add to that whatever you want for local user data. A 400-Mbyte disk would give you 100+ Mbytes for *AIPS* data after you are done with everything else. Note that this is a *real* 400-Mbyte disk; there are no 'disk doublers" of interest to Linux and *AIPS*. Our test system actually has two 400-Mbyte disks.
- All software needed is in the public domain or under a GNU CopyLeft (except *AIPS* which is of course freely available under the usual user agreement to academic/education/research sites).

Some of the details of the port follow.

- The `/usr/lib/libbsd.a` library was used for the socket library, because of BSD-style `ioctl()` calls used by *AIPS*. This should be transparent to both the installer and the end user.
- `INSTEP2` may hang when compiling large libraries (those containing a large number of object files) due to the passing of a greater number of command line parameters to COMRPL than the environment can handle. A simple `CONTROL-C` and re-execution of `INSTEP2` will restart it where it left off. Likewise for `INSTEP4`.
- Local tape access has not been implemented. Most PC's have either no tape drive or have a QIC (quarter-inch cartridge) or similar drive. The latter is incapable of backspace operations (other than rewind) so is of little interest to *AIPS*. Remote tape access and FITS disk file access both work.
- Two tasks, `TVFLG` and `SPFLG`, need special attention to get around a subtle "feature" of `gcc` and the `f2c` converter. This is handled automatically in `INSTEP1` as long as you have the `patch` program on your system. Otherwise you will have to patch the files by hand.

AIPS has been ported and is known to work on the following configurations:

- 0.99.12 kernel with libc–4.4.4, libm–4.4.4 and gcc–2.4.5
- 0.99.14 kernel with libc–4.5.8, libm–4.5.8 and gcc–2.5.7

- 0.99.13 kernel with same config as 0.99.12 with one *notable* exception: The file locking in 0.99.13 does not seem to work, so in order to use the 0.99.13 kernel, the `/usr/src/linux/fs/locks.c` kernel source file from the 0.99.12 kernel must be substituted for the `locks.c` file found in the 0.99.13 kernel (and the kernel recompiled).
- 0.99.15 kernel still to be tested (probably will work by the time you read this).

For all configurations, the FORTRAN code was compiled with `f2c` (which outputs C code that is fed to `gcc`). The version of f2c used was "*VERSION 21 October 1993 13:46:10*". The `f2c` shared library used was `libf2c.so.0.9`.

The `f77` shell script front-end to `f2c` was modified locally to perform some *AIPS*-specific functions, and the modified version is placed in your `$SYSLOCAL` by the installation procedure `INSTEP1`.

All kernel sources, libraries, and compilers are in the public domain, covered by the GNU public license. They are available on the sites mentioned above. There are also several Linux "packages" available on these archives that can be used to build a complete, working system. These are not necessarily needed for *AIPS* but you may want them for other purposes.

On the Future of AIPS (Contributed by Richard Simon)

There seems to be some confusion in the community regarding the future of *AIPS* and *AIPS* support. The driver for this confusion, of course, is the AIPS++ project. AIPS++ is the project to create a new, object-oriented system for processing radio astronomical data; NRAO is one of seven members of the AIPS++ Consortium created for the project. The ultimate expectation is that AIPS++ will replace *AIPS* completely, with full support for all of the current functionality within *AIPS*, as well as new capabilities (improved programability, full support of other interferometers, single dish support, and many other ambitious goals).

NRAO is fully committed to the AIPS++ project, but also recognizes the vital role that *AIPS* currently plays, and the necessity for maintaining *AIPS* until AIPS++ is a truly viable alternative.

The key questions are:

- When is AIPS++ likely to be delivered?
- When is AIPS++ likely to be fully functional and relatively bug-free, from a user's perspective?
- How does long does NRAO plan to continue *AIPS* support?

The first question can be answered fairly simply: the AIPS++ project plans to release a beta version of AIPS++ in December of this year, followed by a full release in early 1995. The beta version will have some interesting capabilities, including some not currently available within *AIPS*. It will not be a full replacement for *AIPS*. This schedule means that a functional AIPS++ should be available by the 2$^{\text{nd}}$ quarter of 1995. Full functionality for AIPS++ will be achieved in a subsequent release, perhaps later in 1995. This schedule is admittedly ambitious.

The second question above is harder to answer. Users who are currently happy within *AIPS* may or may not see reasons to switch during 1995. It is difficult if not impossible to predict the amount of time between the release of a large package and its general acceptance by the community. If fully functional is defined as encompassing all of the current capability of *AIPS*, the end of 1995 may be a reasonable target for AIPS++ to aim at.

As far as the last question, it is clear that *AIPS* will be needed and supported through the end of 1995, and very likely to the end of 1996. In other words, the current lifetime of *AIPS* is about two to three years, with a large margin of error, depending on how optimistic one is about AIPS++. It will be important that *AIPS* and AIPS++ have some overlap, with both packages fully operational, so that new or translated algorithms can be thoroughly tested and compared between the two packages.

AIPS is a very difficult act to top, and the AIPS++ Consortium has its work cut out for it. It is fair to say that one of the successes of AIPS++ to date has been to make the radio astronomy community realize how good *AIPS* really is!