VLA COMPUTER MEMORANDUM NO. 142

NOTE ON MINICOMPUTER TIMES FOR VLA POST-PROCESSING

W. R. Burns and F. R. Schwab

July 26, 1977

General Comments on Minicomputer Versus Big Computer

For purposes of this discussion I define a minicomputer as a computer system which in total cost is less than $250K. Such a system might well have a 500 ηsec cycle time, 1 megabyte of memory, a 1600 bpi tape drive, 200 m byte of disk, and an interactive grey scale display device. It is clear that such a system would be adequate for solving many of the data processing requirements of the VLA continuum system. Some specifics in this regard are outlined in the following section. A statement of caution, however, should be given. Most of the cost of the multi-million dollar large systems is in the software, not the hardware. The conveniences that this more highly developed software brings one are real, but are not always apparent in the present type of comparison. This is largely because the better software buys flexibility primarily and when specific tasks are compared, flexibility normally has little merit. In the real computer world, however, flexibility is a very significant factor which should not be underestimated. Additionally, the larger computers in general have much more rugged hardware, but reliability has little relevance in the kind of comparison made here. In brief, if a mini system is used by only one or two people at a time, and if it is used to do rather specific, well-defined and preset things, it can be very helpful. If it is pushed to do rather more than these tasks, it will probably do nothing well.

Timing Comparisons

What is the real time required to run a particular program on a minicomputer of the type one would buy in the next few years? This naturally depends on the type of program, the amount of money and disk available, the kind of hardware, etc. An assumption which is probably no worse than most others is that the real time in question would correspond well to the real time used by the program if it were run on the IBM 360/65 in small partition on a stand alone basis. As such we have tried to estimate the real time for a number of continuum interferometer programs under these conditions.

The program selected are the FFT, contour map drawing, map linear interpolation and scaling for the Dicomed and the clean algorithm. For most of these programs the run time could be either increased or decreased by making changes in the algorithm. The run time of a contour package, for example, depends very heavily on the type of contour algorithm used. As such the times here are given without regard to whether or not probable changes would be made in the algorithm. This is not unreasonable since the particular algorithms timed have evolved over many years of use.

1. Fast Fourier Transform - FFT

   (a) 256 x 256 complex - single precision.
   Estimated ratio of real/CPU time = 2.5
   CPU time = 40 sec
   Real time = 100 sec

   (b) 1024 x 1024 complex - single precision
   CPU time = 13 minutes
   Real time = 32 minutes

2. Contour Map

GPCP algorithm - dirty map of fairly complex source

512 x 512 points

15 levels - 10% contours

Estimated ratio of real/CPU time = 4

CPU time = 9-1/2 minutes

Real time = 38 minutes

3. Prepare Dicomed Tape

a) Output grid = multiple of input grid

Input array = 128 x 128

Output array = 850 x 850

Estimated ratio of real/CPU time = 3

CPU time = 1.5 minutes

Real time = 4.5 minutes

b) Input grid does not match output grid

Input array arbitrary

Output array = 820 x 820

Estimated ratio of real/CPU time = 2.5

CPU time = 2 minutes

Real time = 5 minutes

4. Clean Algorithm

a) Map size = 128 x 128 (resident in core)

Beam is read in 4 rows at a time

Estimated ratio of real/CPU time = 2

CPU time = .8 sec per iteration

Real time = 1.6 sec per iteration

Time changes as square of linear dimension, i.e., for 500 iterations of a 1024 x 1024 map we get $500 \times 8^2 \times 1.6 = 14$ hours. (requires ≈ 4 megabyte of core)

These times could probably be reduced by a factor of 25 with the addition of an array processor on the minicomputer at an additional cost of $50K, i.e., a nominal total system cost of $300K.

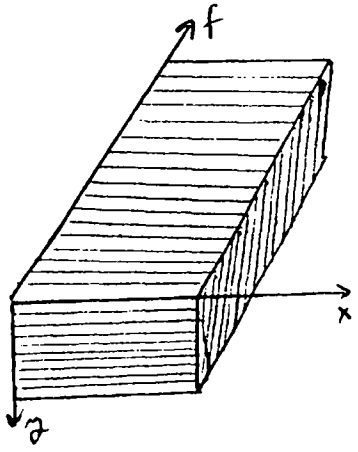
b) Same as (a) but with map stored on disc instead of core.

Increase ratio of real time/CPU time from 2 to 6.

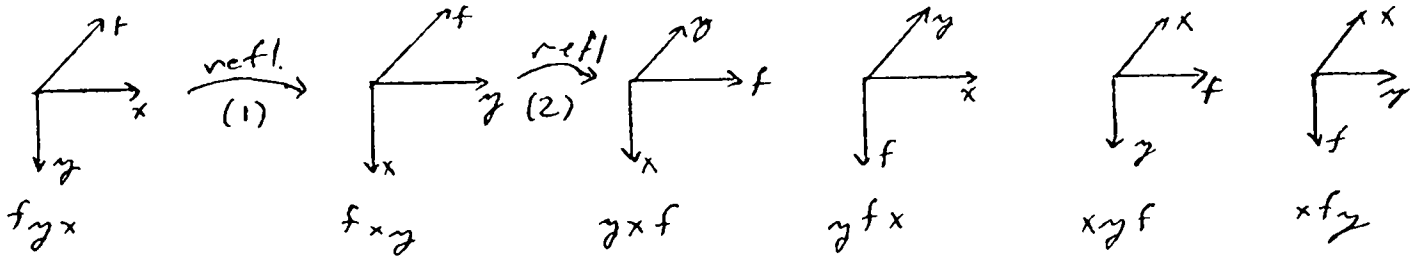
5. Transposition of Spectral Line Maps

In spectral-line aperture synthesis the output of the Fourier transform and map correction step is a series of spatial maps (on RA-Dec or x-y axes), one for each desired velocity. In computer terms, the map values are in the order {fyx} (when the right-most coordinate varies most rapidly). For purposes of spectral analysis one needs to rearrange the data to obtain the spectrum at each map point (i.e., to place the map data in {yfx} order). Such a transposition is also useful in generating maps, familiar to users of the 300-foot telescope, in which the axes are right ascension-velocity at fixed values of declination. The computer operations required to perform this transposition are described below.

A timing run was performed on the CV 360/65 for transposing (reflecting) simulated spectral line maps held on disks in model 3330 disk drives. The data that were operated on can be thought of as cubes and other right-parallelepipeds of numbers (16 bit integers), stored sequentially as rows from successive planes lying parallel to one of the faces of the parallelepiped.



The data sets used in this test were of a type in which map rows could be accessed either sequentially or in a random-access mode. The input/output operations performed in the test achieved reflections of the axes, as drawn above, so that sequential access of a new data set would produce rows parallel to a different axis than in the prior data set, but so that one axis would remain fixed.



The timing information tabulated below comprises the following operations:

(a) Generating a file of test data in fyx order.

(b) Performing the reflection (1) to generate a fxy data set.

(c) Performing the reflection (2) to write a yxf data set.

(d) Reading chunks of test data from each file to verify that the program performed correctly.

Operations (a) and (d) required negligible time in comparison to the total.

The reflections were achieved by reading single entire planes (matrices) of data into core, transposing in core, and writing the data out again. Note that step (b) requires only sequential access of the fyx data set, while step (c) requires random access of the fxy data set. Unbuffered, unblocked Regional (1) data sets were used, with record

lengths sufficient to hold only one row of data. Data access was always to two different disk units. Chained scheduling was used in all cases to obtain priority in disk access.

Map Dimensions	Real Time (min.)	CPU Time (min.)	Real/CPU	Core (bytes)
255x128x128	60.08	13.07	4.6	64K

During this run, no other jobs were allowed to run in the computer.

5/7/76: Instruction from Dr. Hvatum: Two months after a memo number is issued, the memorandum must be in circulation. If it is not, the number will be withdrawn from the person holding it, and reassigned.

12/27/76: This numbering is controlled by Doris Gill, VLA Site. Do not use a number here without clearing it with Doris Gill.

VLA COMPUTER MEMORANDA

NO.	TITLE	AUTHOR	DATE
139	VLA Post-Processing: An Initial Discussion and Proposal	W. R. Burns	7/15/76
140	VLA Post-Processing: Phase I	W. R. Burns E. W. Greisen	3/25/77
141	VLA Post-Processing: Phase I Continued	E. W. Greisen	9/21/77
142	Note on Minicomputer Times for VLA Post-Processing	W. R. Burns F. R. Schwab	7/26/77
143	Report of the VLA Off-Site Data Processing Ad Hoc Study Group	M. S. Roberts	10/21/77