NATIONAL RADIO ASTRONOMY OBSERVATORY
SOCORRO, NEW MEXICO
VERY LARGE ARRAY PROGRAM

VLA COMPUTER MEMORANDUM NO. 146

USE OF THE HERMITIAN PROPERTY IN THREE-DIMENSIONAL MAP PROCESSING

B. G. Clark

May 1978

To minimize storage and data thrashing in making three-dimensional maps, we would like the transposing memory (the place the data is stored between the row transform and the column transform) to be the size of a single map (say 2 k x 2 k real numbers). After the transforms are completed in the first two dimensions ($u,v \to x,y$) and the outputs are stored on disk, a simple DFT would take care of the third (short) dimension.

If we were not using the Hermitian property, we would have to make the transposing memory 2 k x 2 k complex numbers, twice as much. The procedure for saving this memory is well-known in one dimension and readily generalized to three.

The procedure described below was generated by reading the IBM RHARM documentation from bottom to top. For simplicity I neglect the edge effects.

Suppose we have a half-solid, with the other half understood to be the Hermitian conjugate.

$$X_{mnr} \begin{cases} m = -N+1, -N+2, ----, 0, 1, ---, N-1 \\ n = 0, 1, 2, ----, N-1 \\ r = -M+1, -----, M-1 \end{cases}$$

We generate a new solid Y of the same dimensionality by

$$Z_{mnr} = \begin{cases} iX_{m,n+\frac{N}{2},r}\, e^{i\pi\frac{n}{N}} & n<\frac{N}{2} \\ -iX_{m,n-\frac{N}{2},r}\, e^{i\pi\frac{n}{N}} & n\geq \frac{N}{2} \end{cases}$$

$$Y_{mnr} = X_{mnr} + X_{-m,N-n,-r} + i(Z_{mnr} + Z_{-m,N-n,-r})$$

We then may FFT Y in any order we please. If we do the m and n directions first, the transposing memory required is N*(2N-1) complex words, or $(2N)^2$ real words, as required. Define B to be the Fourier transform of Y.

$$B_{jk\ell} = \sum_{M=-N+1}^{N-1} \sum_{n=0}^{N-1} \sum_{r=-M+1}^{M-1} Y_{mnr}\, e^{2\pi i(\frac{mj}{2N} + \frac{nk}{N} + \frac{r\ell}{2M})}$$

$$= \sum_{m=-N}^{N-1} \sum_{n=0}^{N-1} \sum_{r=-M+1}^{M-1} X_{mnr}\, e^{2\pi i(\frac{mj}{2N} + \frac{nk}{N} + \frac{r\ell}{2M})}$$

$$+ \Sigma\ \Sigma\ \Sigma\ X^*_{mnr}\, e^{-2\pi i(\frac{mj}{2N} + \frac{nk}{N} + \frac{r\ell}{2M})}$$

$$+ i[\Sigma\ \Sigma\ \Sigma\ Z_{mnr}\, e^{2\pi i(\frac{mj}{2N} + \frac{nk}{N} + \frac{r\ell}{2M})}$$

$$+ \Sigma\ \Sigma\ \Sigma\ Z^*_{mnr}\, e^{-2\pi i(\frac{mj}{2N} + \frac{nk}{N} + \frac{r\ell}{2M})}]$$

$$\mathrm{Re}\{B_{jk\ell}\} = \mathrm{Re}\{\Sigma\ \Sigma\ \Sigma\ X_{mnr}\, e^{2\pi i(\frac{mj}{2N} + \frac{2kn}{2N} + \frac{r\ell}{2M})}\}$$

which is just what we want for the even (note the 2kn) map points.

(N.B. - having tired of writing indices and ranges on the sums, I omit them in the cases where they are relatively obvious.)

$$\mathrm{Im}\{B_{jk\ell}\} = \mathrm{Re}\{\Sigma\ \Sigma\ \Sigma\ Z_{mnr} e^{2\pi i\left(\frac{mj}{2N} + \frac{nk}{N} + \frac{r\ell}{2M}\right)}\}$$

$$= \mathrm{Re}\{\sum_m \sum_r \Big[\sum_{n=0}^{\frac{N}{2}-1} iX_{m,n+\frac{N}{2},r}\, e^{i\pi\frac{n}{N}} e^{2\pi i\frac{nk}{N}}$$

$$+ \sum_{n=\frac{N}{2}}^{N-1} (-i)X_{m,n-\frac{N}{2},r}\, e^{i\pi\frac{n}{N}} e^{2\pi i\frac{nk}{N}} \Big]$$

$$e^{2\pi i\left(\frac{mj}{2N} + \frac{r\ell}{2M}\right)}\}$$

$$= \mathrm{Re}\{\sum_m \sum_r \Big[\sum_{n=\frac{N}{2}}^{N-1} iX_{mnr}\, e^{2\pi i\left(\frac{2h+1}{2N}\right)n} (i)^{-2k-1}$$

$$+ \sum_{n=0}^{\frac{N}{2}-1} (-i)X_{mnr}\, e^{2\pi i\left(\frac{2h+1}{2N}\right)n} (i)^{2k+1} \Big]$$

$$e^{2\pi i\left(\frac{mj}{2N} + \frac{r\ell}{2M}\right)}\}$$

$$= \mathrm{Re}\{\sum_m \sum_r \sum_{n=0}^{N-1} (-1)^k X_{mnr}\, e^{2\pi i\left(\frac{2k+1}{2N}n + \frac{mj}{2N} + \frac{r\ell}{2M}\right)}\}$$

So

$$(-1)^k\ \mathrm{Im}\ \{B_{jk\ell}\}$$

is just what we want for the odd-numbered (note the 2k+1) map points.

Therefore, to halve the size of the transposing memory, we have to increase the work of the input convolution, as the same data must be either convolved twice or the result of the first convolution must be saved for later use.