NATIONAL RADIO ASTRONOMY OBSERVATORY
VERY LARGE ARRAY PROGRAM
SOCORRO, NM 87801

VLA COMPUTER MEMORANDUM NO. 161

FLOATING POINT FORMAT FOR VISIBILITIES

B. G. Clark

October 1981

It is widely agreed that it is worth getting rid of the annoying gain codes. They are a great nuisance, and in a very few cases, cannot be properly chosen (certainly for the sun, perhaps for large, strong sources, e.g. Sgr A).

The desiderata for a floating point format are that it should cover the possible range of correlation coefficients, and that it not degrade the amplitude accuracy of the instrument.

The minimum correlation coefficient to be represented should be chosen such that it causes no appreciable degradation of signal-to-noise ratio after a moderate (say 40 seconds) on-line integration at 50 MHz bandwidth. The noise on this integration of 4×10^9 bits with a 54% duty cycle is 1.16×10^{-5}; a minimum representable correlation of 10^{-6} will cause no degradation of signal-to-noise ratio (the current system has a least significant bit (lsb) of 1.2×10^{-7} at gain code 0; gain code 3 has a lsb of 10^{-6} correlation).

The maximum correlation to be represented is, suprisingly, not one. Consider the spectral line case, with normalization by the zero lag channel only (see VLA Test Memorandum No. 131). When fed by two

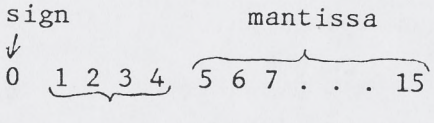
perfectly correlated broadband signals, the system will report unit correlation on each channel. In a lag space, though, there will be a single non-zero correlation coefficient, unity at zero lag. Consider what happens, then, if the system is fed with perfectly correlated narrow band signals. Each lag channel will then have unit correlation, and the amplitude of a output Fourier transform of length n will be n times greater. The VLA, therefore, in a bizzare case, might need to represent a "correlation" of 512.

It would be very convenient to be able to represent fluxes in the same way we represent correlations, especially if we get on-line calibration organized. A convenient way of doing so would be to merely assume the flux scale differs from the correlation scale by 2**8 = 256. A given number then will represent both flux and correlation if the system temperature is about 25 K. In the practical case, then, we convert from correlations to fluxes by multiplying by numbers somewhat greater than one, compressing the high end fluxes which can be represented (these have other, special problems anyway) but taking no chances on loosing to truncation noise.

There are three floating point formats which are fairly reasonable.

1. Simple Binary Floating Point

   Format



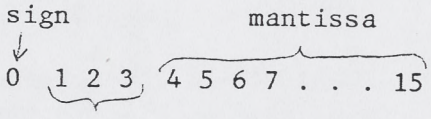
Correlation: with zero exponent, bit 15 corresponds to 1.0×10^{-6}. Largest number representable: 64 (probably large enough for practical cases). Bit 15 represents .05% to 0.1% of the number.

Fluxes: with zero exponent, 1sb corresponds to 0.24 mJy. Largest number representable is 16 kJy.

Comment: the simplest form.

2. Base Four Floating Point

Format



exponent = power of 4

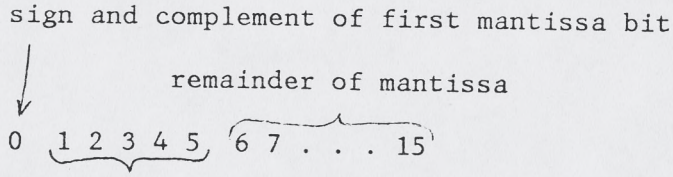
Correlation: with zero exponent, 1sb corresponds to 1.0×10^{-6}. Largest number representable: 64. Bit 15 represents 0.025% to 0.1% of the number.

Fluxes: with zero exponent, 1sb corresponds to 0.24 mJy. Largest number representable is 16 kJy.

Comment: most readable form, because the exponent lies in a single hexidecimal or octal digit.

3. Binary Floating Point With Phantom Bit.

Format



exponent = power of 2

Correlation: Minimum representable number: 2**-23 = 1.2×10^{-7}. Maximum representable number: 511.75. Bit 15 represents 0.05% to 0.1% of the number.

Fluxes: Minimum representable number: 30 μJy. Maximum representable: 131 kJy.

Comment: the most powerful, though most unreadable, form.

I have placed the exponent bits on the left so that a simple integer comparison (determination of <, =, >) of, say, a visibility with a limit, will give the correct answer. The same constraint dictates that the negative visibilities be represented by the ones or twos complement of the representation of the absolute value, including complementation of the exponent.

I have also briefly considered a form which is basically base four with phantom bits, which extends the range of base four representation by a factor of eight, but reject it as too messy: a given bit is sometimes exponent, sometimes mantissa.

Conversion back to fixed point with gain code requires the Modcomp about 22 μs per word, about 15 minutes per day for continuum (I have a draft program). I expect a similar impact on the DEC-10.

I believe the third form of floating point above - binary floating point with phantom bit - is sufficiently more powerful to be worth implementing.

BGC/ap