compress.doc

DATA COMPRESSION FOR ARCHIVAL OF VISIBILITIES

crs/91Jly28

The specified data recording rate for GMRT correlator is 128 kB/s in a normal IEEE floating point representation. In this case, a single session (12-hour) requires two tapes with the Exabyte 8200 or one tape with the Exabyte 8500 drive. However, sustaining these data rates in real time is often difficult and depends critically on the quality of the tape being used. In a normal recording, an Exabyte drive does a read after every write to verify integrity of writing; if it detects an error, it simply marks it as a bad record and re-writes the record in the subsequent portion of the tape reel. A good data compression algorithm is therefore very valuable since at these rates, i/o is often more difficult to sustain than the cpu overhead of conversion between compressed and uncompressed forms.

I give below four possible schemes for compressing visibilities worth considering for the online recording. These schemes reduce the net data rate from 128 kB/s to anywhere between 48 kB/s and 96 kB/s depending on the scheme used.

Mode A (Real/Imaginary) : This is the simplest form and involves least overheads of data conversion. In this representation, we simply retain the three most significant bytes in the standard IEEE floating point representation of real/imaginary parts of the number. Thus, each complex number will be represented in 48 bits.

Mode B (Real/Imaginary) : In this, a pseudo-floating point representation will be used for the real/imaginary parts. The representation is essentially a scaled integer, in sign-magnitude form. The scale factor is represented by the 4 msb's as a radix-2 exponent. The remaining 12 bits contain the sign and magnitude of the scaled number. The numbers will be de-normalised, with a possible a bias for the exponent and an offset for the entire number which could be declared in the header/trailer records. In this representation, each complex number will be represented in 32 bits. The sign-bit may be located in the msb of the word, which makes comparisons between two numbers easy as simple integer comparisons will be valid. However, conversion to normal floating point will be faster if the sign bit is shifted after the exponent, and the exponent is contained wholly within the most significant nibble.

Mode C (Amplitude/phase): This is a 32-bit representation with a very high dynamic range capable of accommodating the strongest and weakest meaningful signals without any further need for re-scaling. In this representation, 10 bits are used for phase, 6 bits are used for a scale factor as a radix-2 exponent, and a 16-bit unsigned integer has the scaled amplitude. The number of bits allocated for scaled amplitude (16) is definitely overkill, and if necessary, one may prune it (e.g. to 14 bits) and using the remaining bits for online flagging.

Mode D (Amplitude/phase): This is a 24-bit representation of each complex number, with the 9 most significant bits representating phase, and the 15 remaining bits representing the amplitude. The amplitude is represented the pseudo-floating point format as mode B, where the sign bit is not required. We may use 5 bits for the radix-2 exponent and 10 bits for the mantissa. In this representation, each complex number is represented by 24 bits. If needed, one may include a bias for the exponent in the amplitude by declaring it in the header (it does not make sense to allow an offset other than zero for the amplitude.)

In this representation, the smallest representable phase is 0.7 deg. Ratio of largest to smallest nonzero amplitude represented in this

fashion is 4.26 x 10^9. The relative accuracy of amplitude is about 0.05% to 0.1%. If the dynamic range of amplitudes representable in this fashion is overkill, we may consider 10 bits for phase and 9 bits for the mantissa of amplitude. This gives similar weightage to both phase errors and gain errors. The representation is good upto a signal to noise ratio of about 512-1000.

This is clearly a very powerful notation with a high degree of data compression and without any need for rescaling depending on fluxes. It is somewhat clumsy for conversion to IEEE floating point, but is worth implementing because of the efficient data compression and hence savings in archival/retrieval times.