SKA – TDT Group

On Phase Computation during Folding

C.Baffa, E.Giani January 2014

The last part of pulsar search algorithm, as in the SKA TDT group approach, is the verification and optimization of pulsar parameters candidates identified in the previous phases. This part can be divided in two portions: folding of data and optimization.
The folding process is in reality a procedure to *integrate* the pulsar signal over the local integration time (typically 600sec) taking into account at least two physical effects: earth-source relative velocity changes (acceleration) and interstellar medium refractive index (de-dispersion).
To compute both folding and correction for physical effects, it is necessary to operate on the exact time stamps of the signal, normally quantized in time step of 50μs, which, in turn, must be converted to a phase measure, to accumulate the pulsar signal data in a coherent way. The computation thus involves up to 10^7 time-steps on one local integration time.

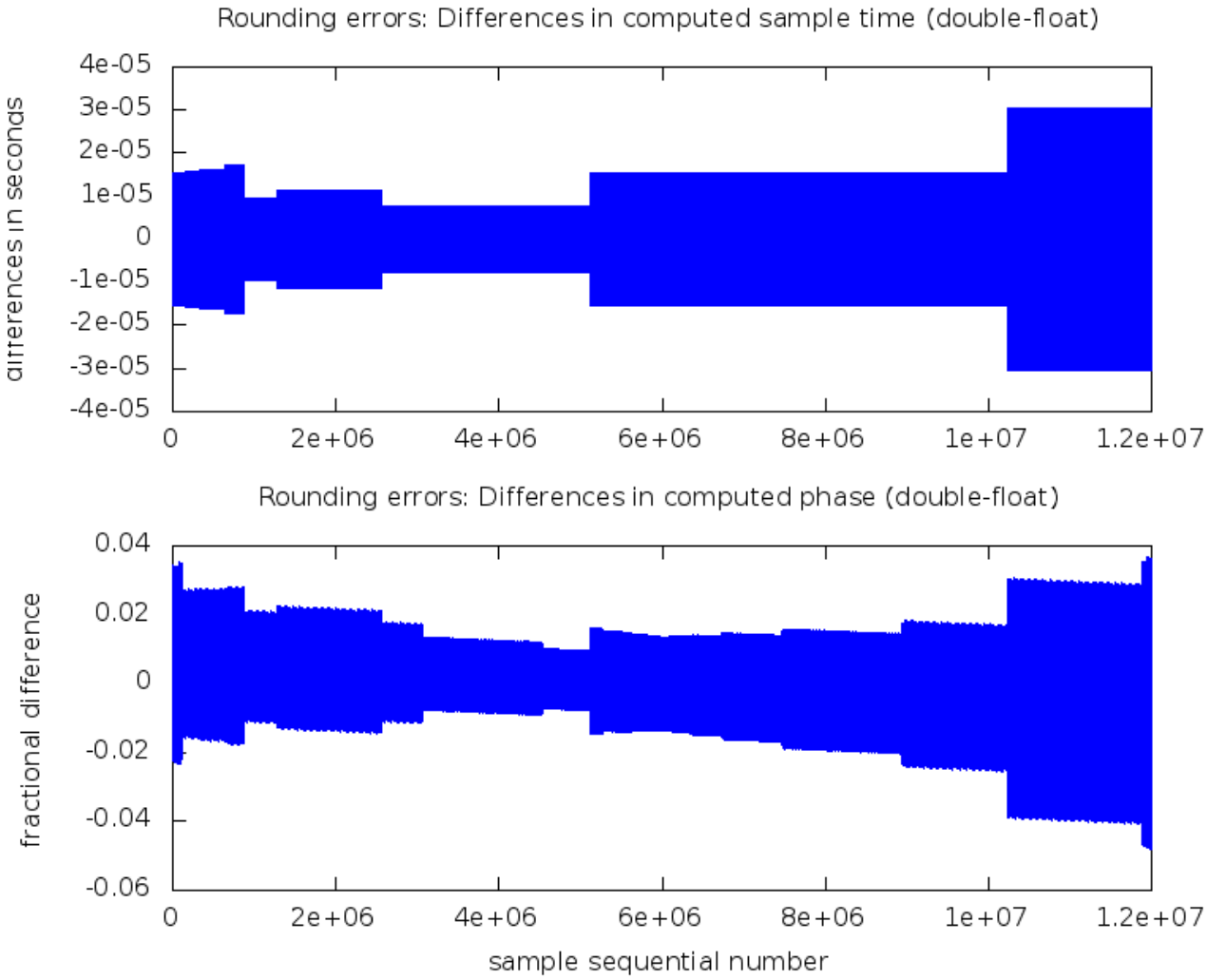

Figure 1: Float versus Double differences in time and phase computation. Total time 600s.

If the folding is performed on a X86 CPU the phase computation gives no numerical problems, as the computations can be performed as a double float number without destructive performance loss. On GPU, instead, there is a significant performance hit using double-type computations. Therefore the folding computation is normally performed as single precision float.
The floating point standard adopted on both X86 and GPU is the IEEE 754, which prescribe a mantissa of 24 bit on single precision (sign plus 6.9 digits). As a consequence a single precision number cannot precisely locate the more than 10^7 time-steps of an elementary integration within its 6.9 digits. We performed a simulation on Octave/Matlab on a x86 CPU to assess the error deriving from the single precision computation.
The errors, both in time and in phase computation, are plotted in the Figure 1. Abscissa is the sample sequential number, which directly correlate to the sample measure time. We note that SKA algorithm sets the zero time at the center of data stream, thus mitigating the precision problem.
On the upper panel it is shown the differences in time determination of the sample between single and double float computation. As errors runs from minimum value to maximum during every period, the graph appear as a filled area, on our full integration scale, revealing its line nature only at the largest enlargement.
The minimum error is near the center of the plot, as a consequence of the zero location choice. The maximum time error is around 30μs, which is acceptable for physical effect computations. For phase computation this error is important.
On the lower panel it is shown the differences in phase computation. As in previous case, the minimum error is near the center of the plot, and the more noisy appearance is probably due to a larger number of rounding operations.
As the phase errors are expressed as fractions of the full period (in our case 44 measurement of 50μs), it is evident that the maximum integration time compatible with single precision phase computation is very short.
It is also to be considered that on x86 the IEEE standard is fully implemented, while on GPU the standard is implemented with a less conservative (and faster) approach, and the weakest part is the rounding phase.
We therefore assess the necessity to perform the time and phase computations on double precision format, to avoid integrations errors and the resulting S/N degradation, due to smearing of the accumulated signal.