JIVE Uniboard Correlator Memo 2: On scaling delays

Des Small

September 11, 2012

1 Introduction

A crucial part of VLBI is the *a priori* estimation of geometric and atmospheric delays between antennas. In the JCCS system that is used to run the Mark 4 hardware correlator and the SFXC software correlator, these delays are calculated using the program CALC[1], and it is intended that the Jive Uniboard Correlator should follow their example.

In the course of implementing the support for model generation and distribution in the correlator control system it emerged that there were some places in which the coverage of delay and phase models in the EVN Correlator Design document[12] were unclear or under-specified.

The purpose of this document is to fill in all gaps in the Jive Uniboard Correlator (JUC) team's understanding of delay models and their implementation. In the process, I have consulted documentation from the Mark 4 and SFXC correlators and spoken with Bob Campbell, Sergei Pogrebenko (designer of the software correlator that was later the basis for SFXC), and Mark Kettenis and Aard Keimpema of the SFXC team. I am grateful for their assistance and patience. Any remaining misunderstandings are my own fault, however, and I'd be glad to hear about them.

2 Off-board handling of delay and phase models

The original plan for treatment of delay and phase models in the EVN Uniboard-based correlator was to stay as close as possible to the SFXC software correlator.

The software correlator SFXC uses CALC to calculate delays on a grid on times at 1 s intervals. The delays are then calculated as needed using third-order Akima splines to interpolate to times off this grid.

With this treatment, Aard Keimpema assures me that the delays are identical to those of the model itself down to machine precision with double-precision floating point. There is no need to calculate phase delays separately, since the time delays can be multiplied by frequencies with double-precision floating point accuracy inside the correlator.

Essentially all the differences in delay model handling between SFXC and the JUC arise from the constraint that the latter does not have floating-point arithmetic. Instead we use fixed-point arithmetic (and corresponding registers), and this means that we have to be a lot more careful about the possibility of numerical under- and overflow. In particular, the need to have separate registers for the phase delay arises because the delay coefficients are not stored with enough precision to allow the phase delay to be calculated on board the correlator.

The original intent for the JUC control system was to emulate the SFXC approach as closely as possible. The JUC uses quadratic polynomials with fixed-point coefficients to approximate time and phase delays over $\frac{1}{32}$ s intervals.

- Evaluate CALC at 1 s intervals to provide coarse-grid values;
- Interpolate onto $\frac{1}{32}$ s intervals using Akima splines (fine-grid values);
- Calculate floating point coefficients for quadratic time delay polynomials from successive sets of three fine grid points
- Calculate floating point coefficients for phase polynomials from time delay coeffients
- Discretize time and phase delay polynomial coefficients and send them to the correlator

The plan was that staying close to SFXC's methodology would mean we wouldn't have to reinvent any wheels. However, it turns out that things aren't quite so simple.

Özdemir[2] showed that Akima's method can interpolate model delays to machine precision (for doubles, around 10^{-16}) using a relatively coarse (1 s) grid on which linear and cubic interpolation have worst-case accuracy of only about 10^{-15}.

However, this effectively solves the problem of "which interpolation method is as good as evaluating CALC for each sample, given that we are using double precision?" Note that 10^{-15} corresponds to 2^{-50} so calculating to this accuracy requires 50 bits after the binary point, which is larger than any of the registers discussed in this document.

This is therefore not quite the question we need to answer here. We need to find a treatment of the model that is "good enough" given much more modest resources.

In particular, we do not have any form of floating point. This is the reason that the phase models for the Uniboard correlator have to be calculated off-board - the integer/fixed-point value of the delay doesn't have enough resolution to scale up for the phase delays.

3 JUC design criteria

The architecture of the correlator is designed around Delay registers using fixed binary point, with the integer part used to shift an entire FFT block at once based on the time of (say) the centre sample of the block (rather than individual samples). Pogrebenko argues that shifting the FFT block makes the quadratic component of the delay model more critical, although I cannot currently see why.

The four most-significant bits after the binary point in delay model are used as an index into a lookup table containing phase corrections to apply to the samples. (As the Design Document[12] clearly states on p.20). What had not previously been clear to me, however, is that it is envisaged that the delay registers would have considerably more bits after the binary point than this, in order to avoid degrading accuracy by underflowing small values. (The Design Document doesn't seem to say where the binary point should be placed.)

As we shall see, it is not practical to have all of

- 32-bit delay registers;

- The ability to handle "worst-case" space VLBI;
- Enough fractional bits to avoid underflow; and
- The same number of fractional bits for delay, delay rate and delay acceleration registers

The last requirement is apparently not that firm, and we will certainly want to violate it.

The use of the four most significant bits of the fractional delay for the phase-correction lookup is driven by the fact that the phase-correction vector is discretized at a comparable precision.

Clearly, all of the phase registers are after the binary point. The choice of a 48-bit phase register was made so that phase rate is resolved to a precision of 0.01 samples/s over a full 24-hour observation. $(32\times10^6\ \text{ticks/s}\times(24\times60\times60)\ \text{s}\times2^{-48}\ \text{ticks/tick}^2 s\times\approx 0.01\text{tickstick})$.

4 Space constraints

The worst case space delay and delay rates are those associated with the Radio Astron space telescope; these are 2 s (according to Sergei Pogrebenko) and 50 μs/s (according to Dmitry Duev).

A 2 s delay corresponds to 64×10^{32} ticks, which takes up 26 bits of the delay register for the integer delay, leaving 6 bits for the fractional part. This isn't a lot of scope to handle underflow, and certainly isn't adequate to handle the delay rate with the same fixed point placement (see Equation 2 below).

In Section 5.3 Hargreaves raises the possibility of using different scales for delay itself and delay rate, which would solve this problem.

A 50 $\mu s/s$ delay rate corresponds to 1600 ticks, or 11 bits. This is also not compatible with Hargreaves' proposal to use effectively 28 bits of the 32-bit delay rate register after the binary point.

In short, if the worst-case space requirements really are this bad, and I haven't made a mistake with the calculations, then it is not practical to use the proposed register sizes for space applications.

5 Terrestrial Delays

5.1 A priori calculations

We are going to use quadratic polynomials to approximate station delays relative to the centre of the earth. In order to get bounds for the coefficients of the polynomials we consider a Taylor's series expansion of the diurnal delay due to rotation of the earth.

$$c_0^{\text{delay}} = \text{delay} = \frac{R}{c}\cos\Omega t \tag{1a}$$

$$c_1^{\text{delay}} = \frac{d}{dt}\,\text{delay} = -\frac{R}{c}\Omega\sin\Omega t \tag{1b}$$

$$c_2^{\text{delay}} = \frac{1}{2}\frac{d^2}{dt^2}\,\text{delay} = -\frac{1}{2}\frac{R}{c}\Omega^2\cos\Omega t \tag{1c}$$

Plugging in the numbers ($\Omega = 2\pi/(24 \times 60 \times 60) = 7.3 \times 10^{-5}$radians/s, $R = 6.4 \times 10^{6} m$) we get values for the coefficients (in second-based units) of

$$c_0^{\text{delay}} \approx 0.02\,\text{s} \tag{2a}$$
$$c_1^{\text{delay}} \approx 1.6 \times 10^{-6}\,\text{s/s} \tag{2b}$$
$$c_2^{\text{delay}} \approx 5.6 \times 10^{-11}\,\text{s/s}^2. \tag{2c}$$

Suppose we want to calculate the derivatives in delay and t but evaluate the polynomial in variables Delay and T, with scaling Delay $= L$delay and $T = Mt$. We get

$$\text{Delay} = L\,\text{delay}(t_0) + \frac{L}{M}\frac{d}{dt}\text{delay}(t_0)\delta T + \frac{1}{2}\frac{L}{M^2}\frac{d^2}{dt^2}\text{delay}(t_0)\delta T^2 \tag{3}$$

The Uniboard-based correlator uses an internal unit of ticks for time, based on the 32 MHz sample rate, and the delay input is given in units of ticks. If the output is also on this scale we have $L = M = 3.2 \times 10^7 \text{ticks}/s$. Scaling the coefficients c^{delay} to tick units we get new coefficients C^{delay}, with values

$$C_0^{\text{delay}} = L\,\text{delay} \approx 6.4 \times 10^5 \text{ ticks} \tag{4a}$$
$$C_1^{\text{delay}} = \frac{L}{M}\frac{d}{dt}\text{delay} \approx 1.6 \times 10^{-6} \text{ ticks/tick} \tag{4b}$$
$$C_2^{\text{delay}} = \frac{1}{2}\frac{L}{M^2}\frac{d^2}{dt^2}\text{delay} \approx 1.8 \times 10^{-18} \text{ ticks/tick}^2 \tag{4c}$$

The quadratic coefficient C_2^{delay} is very small; it corresponds to $2^{-58.9}$, so that at least 59 places after the fixed binary point would be needed to store it. And the JUC uses only the four most significant bits of the fractional delay (in ticks) to calculate the fractional delay correction, so that the quadratic term becomes relevant at a time t_{quad} given by

$$1.8 \times 10^{-18} \text{ ticks} \cdot t_{\text{quad}}^2 = 2^{-4} \text{ ticks}, \tag{5}$$

which works out at 5.8 s.

The quadratic term, then, is impractical to store and of no practical use. There is no reason to include it in our time delay model.

5.2 Comparison with Mark 4 correlator

The Mark 4 hardware correlator designed in the 1990s and commissioned in 1998 faced essentially the same issues of model generation as the EVN uniboard-based correlator faces now. Indeed, the CALC program used to calculate the model is a common feature of both as well as of SFXC.

The Mk4 correlator relies on station units (SUs) to keep the data streams synchronised to the nearest sample, so the 32-bit delay register only handles *fractional* delays of ± 0.5 samples. The 18-bit delay rate register used to increment the 18 least significant bits of the delay register. This is effectively a hardware optimisation of a 32-bit register with the most-significant bits zeroed off on the grounds that they will not be needed. As Whitney remarks in Mark 4 Memo 131 [5]:

> The maximum delay-rate supported by the 18-bit delay rate register is 1 delay-shift per $2^{32-18} = 16384$ samples, which corresponds to ≈ 60 microsec/sec and is quite adequate for even worst-case space VLBI.

This would also accommodate the anticipated worst-case scenario for Radio Astron. What is also clear is that the Mk4 correlator's delay resolution of 32 bits for just the fractional part of the delay is considerably better than is currently specified for the EVN uniboard-based correlator.

Finally, we note that there is no delay acceleration register on the Mark 4 correlator.

5.3 Possible adjustment to Uniboard correlator

In response to a previous draft of this document, Jonathan Hargreaves proposed changing the delay registers from integer values to fixed-point with 8 bits after the binary point, and additionally scaling the delay-rate up by a "binary million" (2^{20} or 1048576). As he says[11] of this case:

> If we represent your worst case numbers as 32-bit hexadecimals with two digits after the point we get for the constant coefficient Equations 2 0x09c400.00 and then we scale the linear coefficient up by a binary million (ie 2^{20} not 1000000) we get 0x000001.ad

We saw above that the Mark 4 correlator can manipulate delays at a resolution of 2^{-32} samples; this proposal only resolves down to 2^{-8} for the main delay register. But this corresponds to $\frac{1}{32\times10^6} \cdot \frac{1}{2^8}\,\mathrm{s} = 1.22 \times 10^{-10}$ s, and it is generally understood that delay resolution of less than a nanosecond is good enough for VLBI.

5.4 Recommendations for delay model

I recommend incorporating Hargreave's fixed-point model for delay, with linear coefficient scaled up by 20 bits, and discarding the quadratic term for delay.

6 Phase

6.1 JUC requirements

How accurate does the phase model need to be? Hargreaves and Verkouter[12] (p. 20) say that the 9 most-significant bits of the phase model are added to the phase input.

6.2 Comparison with SFXC

SFXC uses floating point arithmetic for fringe phase correction, which it calculates internally based on the model delay. The model is therefore a third-order Akima spline over an interval of 1 s interval. Since SFXC works, we know that this is at least good enough, but the possibility remains that it is overspecified.

6.3 Comparison with Mark III correlator

The original plan for the Mark 4 correlator was to use a system "basically identical"[4] to that of the Mark III correlator: phase delay and phase-rate would both be stored in 32-bit registers, and the phase would be incremented by the value of the phase-rate register at every sample. The phase rate and the initial phase would be loaded at the beginning of each integration period of 20,000 samples. With a 32 MHz sample rate, this implies a model duration of 1/1600 s..

Whitney states[4] that this linear scheme "is quite acceptable for all anticipated ranges of phase-rate and phase-acceleration, including even the worst-case space VLBI scenario".

6.4 Comparison with Mark 4 correlator

The Mark 4 correlator subsequently adopted a scheme with phase corrections modeled by second-order polynomials over a longer interval. Whitney writes[4] that the primary motivation for this was "to allow significant lengthening of the basic chip integration period, with a corresponding reduction in the DSP horsepower necessary to support the chip".

The software for Mark5B data recorder includes a support for emulation of a Mark 4 SU; the code is the same as that in the SU itself (according to Bob Eldering). The SU gets the model as a quintic polynomial valid for 2 minutes; it then interpolates down to quadratic polynomials each valid for a single correlator frame, as with the delay model. The conversion to a quadratic with floating point coefficients (q_0, q_1, q_2) is followed by a discretision step that converts the coefficients into integer values $(\phi_{d,0}, \phi_{d,1}, \phi_{d,2})$ for three 32-bit registers. All the subtleties occur in this last step; Alan Whitney's document [4] is indispensible in understanding the implementation.

Since the phase is stored in units of periods it wraps at 1, and the resolution is therefore $\frac{1}{2^{32}}$ periods.

The phase register is updated every k Mark 4 system clicks ("sysclicks") by incrementing it with the contents of the phase-rate register; the phase-rate register is updated every n sysclicks, by incrementing it with the value of the phase-acceleration register.

The SU code includes the following definitions for the coefficients of the phase polynomial:

$$\phi_{d,0} = 2^{32}\text{frac}(q_0) + \frac{1}{4}\text{delay}[0]\frac{f_{sb}}{f_{os}} \tag{6}$$

$$\phi_{d,1} = 2^{32}\left(\frac{k}{f_{sys}}q_1 + \frac{nk}{f_{sys}^2}\frac{q_2}{2}\right) + k\frac{1}{4}delay[1]\frac{f_{sb}}{f_{os}} \tag{7}$$

$$\phi_{d,2} = 2^{32}\frac{nk}{f_{sys}^2}q_2 \tag{8}$$

where the first two coefficients include a correction for oversampling (f_{os}) and fractional bit correction, depending on the net sideband ($f_{sb} = \pm 1$); these are described in [9]. f_{sys} is simply the time conversion rate of sysclicks$/s$.

The $\phi_{d,1}$ term is the most surprising. Alan Whitney explains[4] that "The second contribution to phase error is due to unmodelled phase-acceleration over the n samples over which the phase-rate is held constant. This is simply given by $\phi = \frac{1}{2}a\left(\frac{n}{f}\right)^2$." (Where a is the value of phase-acceleration over the interval.) Whitney's correction factor is for phase; the code adds a term to the phase-rate which is used to increment the phase value itself, so that the correction is added linearly. The correction is of course rescaled by k/n to accommodate the different update rates of the two registers.

An important point to note is that the second order coefficient in the phase correlator is not that important in the Mark 4 correlator: Bob Campbell tells me that it was in fact zeroed out in Albert Bos's code, with no noticeable ill effects.

Roger Cappallo [3] remarked in 2003 – five years after the Mark 4 correlator was commissioned – that "the old model chose values of n so small that for many baselines the appropriate value of the acceleration register was less than 0.5, and was rounded down to zero.” The algorithm was then changed to reduce round-off errors, with the result that “n is typically a 4 or 5 digit number”.

This is for a correlator frame that is “nominally 500 ms” long, so approximately 10^{10} systicks long. But the EVN Mark 4 correlator, as mentioned above, does without it altogether.

6.5 A priori calculation

Checking the JIVE experiment database for all experiments since its introduction, we find a frequency range from 312 MHz up to 22,392.49 MHz.

Scaling Equations 2 by these frequencies in units of system ticks, and using ϕ for phase we have

$$1.5 \times 10^{-5}\,\text{ticks}^{-1} < \frac{d}{dt}\phi < 1.1 \times 10^{-3}\,\text{ticks}^{-1} \tag{9a}$$

$$3.4 \times 10^{-17}\,\text{ticks}^{-2} < \frac{d^2}{dt^2}\phi < 2.5 \times 10^{-15}\,\text{ticks}^{-2} \tag{9b}$$

With a 48-bit register all of which is used for the fractional part we can store values down to $\frac{1}{2^{48}} = 3.6 \times 10^{-15}$, with the result that we can store $\frac{d\phi}{dt}$ coefficients unchanged, but that $\frac{d^2\phi}{dt^2}$ coefficients underflow even a 48-bit register.

If we really want to keep the second-order term in the polynomial and have it be used, we would again have to resort to a kind of fixed point arithmetic in the spirit of the Mark 4 correlator’s parameter n.

Over a time range of 1×10^6 ticks, the value of $\frac{d^2\phi}{dt^2}$ ranges from 3.4×10^{-5} to 2.5×10^{-3}, which is less than the minimum precision required of $2^{-9} \approx 2 \times 10^{-3}$.

6.6 Discussion and tentative conclusions on phase

After contemplating the Mark 4 correlator algorithms, it is good to remind ourselves that SFXC simply evaluates a cubic Akima spline for the model for every sample. Simply not having the equivalent of the Mark 4 correlator’s k and n parameters should eliminate most of the quirks of the Mark 4 system.

However, the increase in register size for phase polynomial coefficients from 32 to 48 bits apparently isn’t enough to compensate for the loss of k and n.

Given that the EVN Mark 4 correlator has in practice ignored the quadratic phase correction for much if not all of its working life, I would be inclined to recommend using a linear phase correction model for the EVN uniboard-based correlator over its originally specified $\frac{1}{32}$ second interval.

7 Conclusions

It seems that both the delay and phase models can actually be handled by linear interpolation over the original time-range of $\frac{1}{32}$ s.

For delay, fixed point arithmetic should be used, with 8 bits after the point, and the delay rate should be scaled up by 2^{20}. For phase calculations, the registers are already fixed point with *all* the digits after the point, so no such adjustments are necessary.

But all these calculations should be reviewed carefully before they are committed to FPGA. All the calculations are available in a spreadsheet that accompanies this document.

Since this document was first written, the question has been studied further in JIVE Uniboard Memo 6, and it has been concluded that in fact the JUC could refresh its coefficients only once a second, and that the phase polynomial should be linear (at least for terrestrial applications) and the coefficients stored in 48-bit registers. Consult that document for further details.

References

[1] http://gemini.gsfc.nasa.gov/solve/

[2] Hüseyin Özdemir *Comparison of linear, cubic spline and akima interpolation methods* August 30, 2007

[3] Roger Cappallo *Recently Discovered Model Problems in the Mark 4 Correlator*

[4] Alan R Whitney *Mark 4 Memo 101: Addition of acceleration to Mark 4 on-chip phase-rotator* 27 October, 1992

[5] Alan R Whitney *Mark 4 Memo 123: Implementation of Delay/Phase Tracking in the Mark 4 correlator* 20 November, 1992.

[6] B. Anderson *Mark 4 Memo 141: Straw Man SU Design* (Revision C 930302) `http://www.haystack.mit.edu/geo/mark4/memos/141.pdf`

[7] *Mark 4 Memo 140: The EVN/Mark 4 Station Unit Requirements* `http://www.haystack.mit.edu/geo/mark4/memos/140.pdf` 1 March, 1993

[8] MIT Haystack Observatory *Mark 5B System User's Manual* 8 August 2006 `http://www.haystack.edu/tech/vlbi/mark5/docs/Mark%205B%20users%20manual.pdf`

[9] A R Whitney *et al. Mark 4 VLBI correlator: Architecture and algorithms* Radio Science, Vol.39, 27 January 2004

[10] Jonathan Hargreaves, email `<4F74966C020000F500004BD1@jive.nl>`

[11] Jonathan Hargreaves, email `<4F75AACD020000F500004C10@jive.nl>`

[12] Jonathan Hargreaves and Harro Verkouter, *EVN Correlator Design* Version 2.0, 31 March, 2011