Cavity BPM ADC Data Processing Ideas

M. Stadler, 10.2012

1 Objective

The output signal from the Cavity BPM RFFE to the LT2209 160MSPS ADC board has the shape of a bipolar, DC-free pulse as seen in Figure 2-1.

The position information is (for example) contained in the value of the pulse peak relative to the baseline (=DC component of ADC samples). The goal of the sample processing on the GPAC platform is to precisely measure the peak values of all the RFFE output signals.

Basically this can be achieved by accurately sampling each pulse at its top. If the sampling clock phase is constant and zero with respect to the arrival time of the IF pulse peak the systems operates optimally.

In reality, the clock phase is never precisely known. It can drift away from its actual value due to internal or external effects. Therefore, from the sample values we must derive the following information:

1. The actual ADC cock phase offset from the pulse peak must be estimated (and corrected in a subsequent step)
2. The pulse amplitude must be estimated with suitable accuracy. The estimation error should remain within the acceptance range independently of the actual ADC clock phase offset.

An optimal interpolation method would give an accurate pulse amplitude estimate regardless of the actual ADC clock phase. In such a case requirement 1 is obsolete.

One interpolation method that comes close to ADC phase independency is the sin(x)/x (sinc) interpolation of the sampling data. It will be shown that even with this method there remains an error which is too high for our purpose (due to bandwidth limitations). In addition, this interpolation scheme requires too much FPGA resources to be useful for bunch-to-bunch calculations at 220ns bunch repetition time with the present digital hardware.
If a longer time lag can be tolerated (as for calibration routines or the like), then the sinc-method works excellently and can be very useful.

The polynomial interpolation represents another interesting possibility and can be numerically computed much easier.

This document contains an analysis of both sinc and polynomial interpolation methods. The polynomial interpolation scheme is intended to be implemented on FPGA for bunch-by-bunch computation. For this reason the analysis will be restricted to 2nd order polynomials only.

Based on the analysis, a basic data handling flow is suggested.

2 Sinc Interpolation of sampled data

From theory it is known that a continuous signal can be reconstructed from its samples by use of the interpolation formula

(1) $$s_{rec}(t) = \sum_{n=0}^{N-1} s(n \cdot T) \cdot \frac{\sin\left(\pi \cdot f_s \cdot (t - n \cdot T)\right)}{\pi \cdot f_s \cdot \left(t - n \cdot T\right)},$$

where f_s is the sampling frequency, T is its inverse (the sampling interval), and N is the number of samples considered.

However, this reconstruction is only exact if the Nyquist theorem is satisfied. If this is not the case (1) does not hold strictly and the reconstruction deviates from the continuous waveform as seen in Figure 2-1.

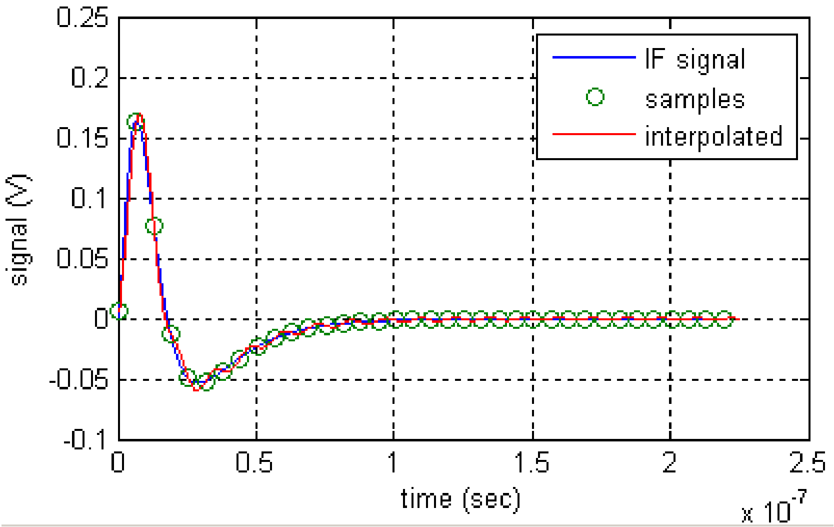


Figure 2-1: IF signal pulse shape and its sinc-interpolation from samples

This error is due to the overlapping of the original signal with its images created by the sampling process.

2.1 Approximate analytical calculation

To estimate the effect of interpolation error due to the signal energy contained in the second Nyquist zone one can relate this energy to the total signal energy at the ADC input.
The downconverted cavity signal has (at the ADC input) a spectral distribution as sketched in Figure 2-2 (top figure). It decays after a 3dB frequency of approximately 25MHz (given a Q of the resonator of 70 at 3.3GHz mid-frequency).A n-pole filter is applied for pulse shaping and to reduce image signal leakage. This filter is assumed to be of Butterworth-type.

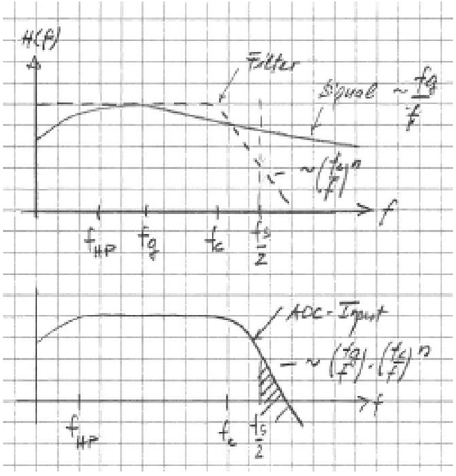


Figure 2-2: Sketch of spectral amplitude densities

If *H(f)* is the Fourier transform of the continuous IF signal, then the energy that leaks from the image signal is estimated as

$$(2)\qquad \Delta E = \int_{f_s/2}^{\infty} \left|H(f)\right|^2 df \approx \left|H_0\right|^2 \cdot \int_{f_s/2}^{\infty} \left(\frac{f_g}{f}\right)^2 \left(\frac{f_c}{f}\right)^{2n} df = \left|H_0\right|^2 \cdot \frac{1}{2n+1} \cdot \frac{f_g{}^2}{f_c} \cdot \left(\frac{f_c}{f_s/2}\right)^{2n+1} \quad (V^2 \cdot \mathrm{s})$$

The total signal Energy is approximately

$$(3)\qquad E = \int_0^{\infty} \left|H(f)\right|^2 df \approx \left|H_0\right|^2 \cdot \left(f_c - f_{HP}\right) \cdot \frac{\pi}{2}$$

The relation of leakage energy to the total signal energy is

$$(4)\qquad \frac{\Delta E}{E} \approx \frac{1}{2n+1} \cdot \frac{2}{\pi} \cdot \left(\frac{f_g{}^2}{f_g - f_{HP}}\right) \cdot \frac{1}{f_c} \cdot \left(\frac{f_c}{f_s/2}\right)^{2n+1} .$$

The amplitude deviation (in fractions of the IF pulse top value) depends on the sampling phase as the Nyquist theorem is not fully provided. The maximum deviation $D_{\max}$ can be estimated by evaluating the square root of the energy ratio:

$$(5)\qquad D_{\max} = \frac{\text{max interpolation peak amplitude error}}{\text{peak amplitude}} \approx \sqrt{\frac{\Delta E}{E}} .$$

This simple expression holds reasonably well, as will be verified by numerical simulation. Above all it gives a good insight into how the different parameters contribute to the interpolation error.

Example:

filter order: n=6
fc=50MHz
f_{HP}=10MHz
f_s=160MHz

This parameter combination result in a Dmax=0.94% (simulation says 0.7%, compare Figure 2-3). Taking a filter order of only 4 results in Dmax=2.8% (simulation says 2.3%, compare Figure 2-3).

Doubling the sampling rate would decrease the error by a factor of 90 (with the parameters above). The simulation returns a factor of about 80.

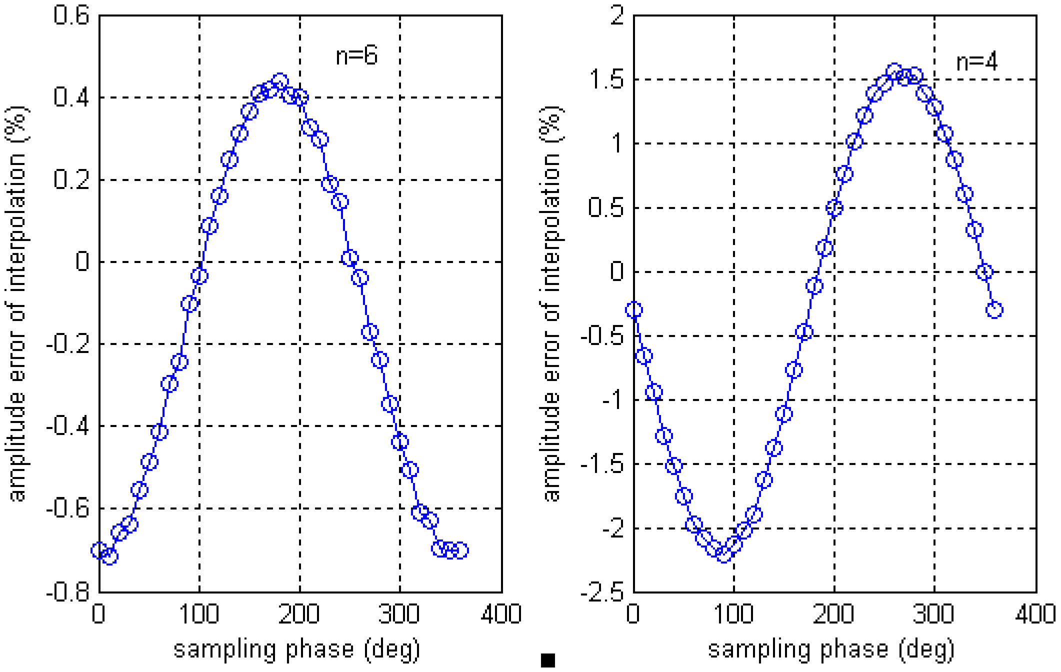


Figure 2-3: Simulated values of Eq. 4 vs. sampling phase for n=4 and n=6.

2.2 Relative interpolated top value error vs. sampling frequency

Figure 2-4 plots the maximum relative interpolation error (as defined in Eg. 5) as a function of sampling frequency with the filter order n as parameter.

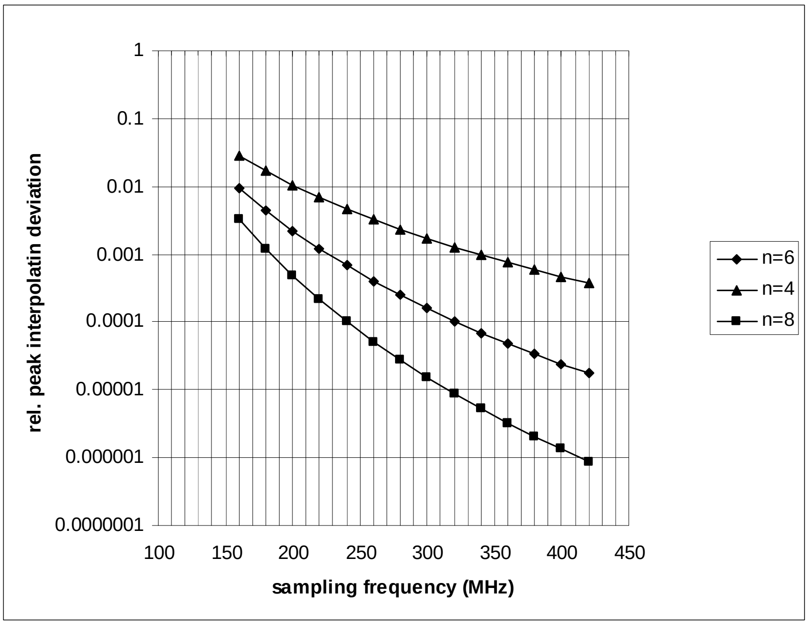


Figure 2-4: relative maximum interpolation error vs. sampling frequency (n=filter order). Other parameters are: fc=50MHz, fg=23MHz, f_{HP}=10MHz.

If an interpolation error shall result in less than 0.1µm of measurement error the relative error should be less than 10^{-4}. With the present filter order (n=6) and realistic signal parameters this implies a sampling frequency higher than 340MHz.

An increased filter order of n=8 with a sampling frequency of 250MHz would theoretically reduce the interpolation error due to image signal leakage below 0.1‰ (which transfers into a measurement deviation in the order of 0.1µm). A 16-bit 250MSPS ADC is available on the market, and a suitable IF filter could be constructed. However, higher filter orders are subject to increased relative pulse delay variation due to component mismatch as well as temperature variations. This is a separate problem that would need extra attention and analysis.

The interpolation used in the Matlab script to generate the plots effectively increased the sampling frequency by a factor of 20 (20 interpolation steps between two adjacent samples). It seems clear that this kind of numerical calculations will not be feasible using the GPAC hardware, at least not for each single bunch. For offline computations the method is however well suited.

3 Interpolation by 2nd Order Polynomial

The sinc interpolation scheme is relatively accurate, but it is so time intensive that it is not suited to be used to perform bunch-by-bunch calculations.

Other schemes are based on a polynomial fit of the sampled data points. But also in this case the computation complexity grows rapidly with the order of the polynomial to be fitted.

Only the simplest case, a polynomial of second order, seems suitable for bunch-by-bunch calculations.

Therefore, it will be used here estimate the actual ADC phase offset, which then can be corrected by reprogramming the DDS. Once the ADC phase is known to be aligned to the real pulse top the top sample is used as a measure for the pulse peak value.

The ADC phase error will be defined here as

(6) $$\Delta\Phi_{ADC} = 360 \cdot \frac{(\text{pulse peak time}) - (\text{peak sample time})}{\text{sampling intervall}}$$ (in degrees).

In other words, the ADC phase error is the difference between the time of occurrence of the actual pulse peak and the maximum sample, scaled by the sampling interval (inverse of sampling frequency). A negative phase error means that the pulse peak precedes the sample and that the DDS phase must be reduced in order to align the sample to the peak.

3.1 Single IF pulse peak interpolation by 2nd order polynomial function

One can also try to fit the IF pulse peak part by a simple second order polynomial. In order to do so on takes the top sample and its left and right neighbor samples.

The 2nd order polynomial is:

(7) $$s(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2$$

For simplicity, we redefine the sampling time such that the top sample location is x=0. Then the samples to be considered are the top sample s_0 and its left and right neighbors (s_{-1} and s_1, respectively).

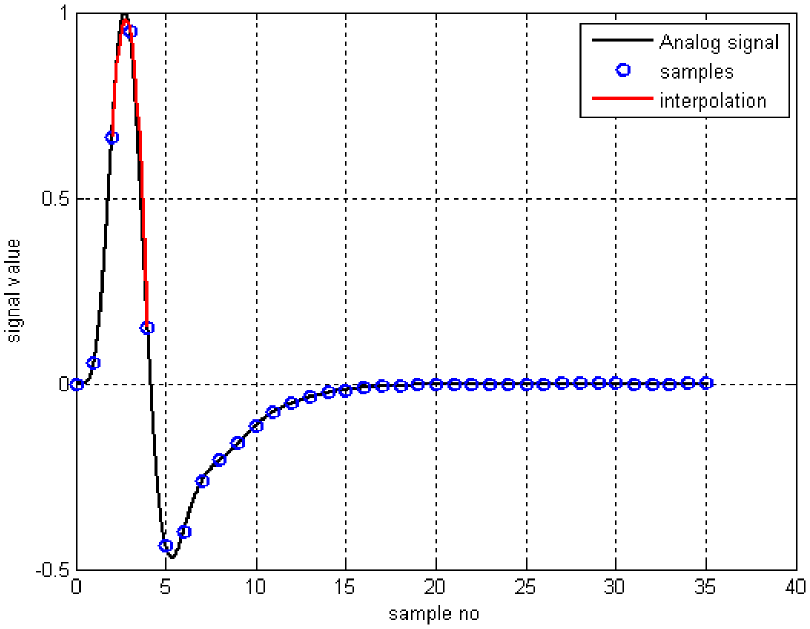


Figure 3-1: 2nd order polynomial interpolation example

The parameters a_0, a_1 and a_2 are then calculated from the sample values s_x by

(8) $$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} s_{-1} \\ s_0 \\ s_1 \end{pmatrix}$$

The so interpolated top value is calculated as

(9) $$s_{top} = a_0 - \frac{1}{4} \frac{a_1^2}{a_2}$$

This top value occurs at the normalized sampling time

(10) $$t_{top} = -\frac{a_1}{2 \cdot a_2}.$$

Because the sampling time has been normalized to the real sampling intervals (samples occur at t=0, 1, 2, …) the estimated sampling phase error is

(11) $$\Delta\Phi_{ADC,est} = t_{top} \cdot 360 = -180 \cdot \frac{a_1}{a_2} \quad \text{(in degrees).}$$

This estimated ADC clock phase deviation is not the actual sampling phase relative to the IF pulse top position. If the ADC clock is adjusted such that t_{top}=0 the top sample is still off by a few degrees relative to the true pulse top value. This is simply because the pulse is not symmetric around its peak. Using realistic system parameters the simulation with Matlab generates Figure 3-2(left).
The zoomed view shows that this remaining phase error is in the order of 10 degrees (Figure 3-2, right).

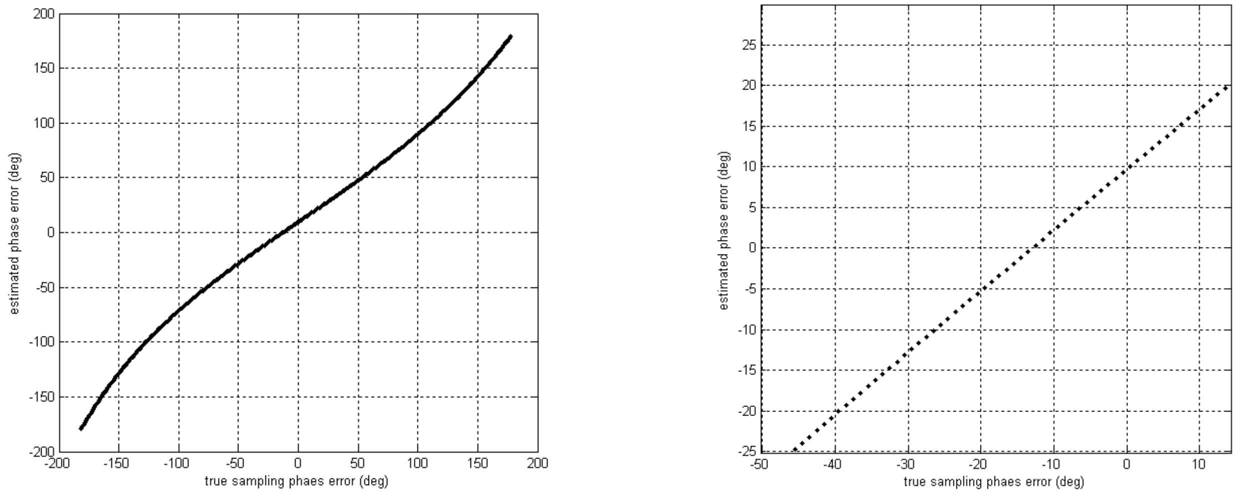


Figure 3-2: Deviation from estimated and true sampling clock phase error (right: detail view)

The effect that a sampling phase change has on the relative deviations from the pulse top value is seen in the simulation results in Figure 3-3. If the sampling phase is zero the top sample coincides with the pulse top value (the red curve peaks at the graph origin).

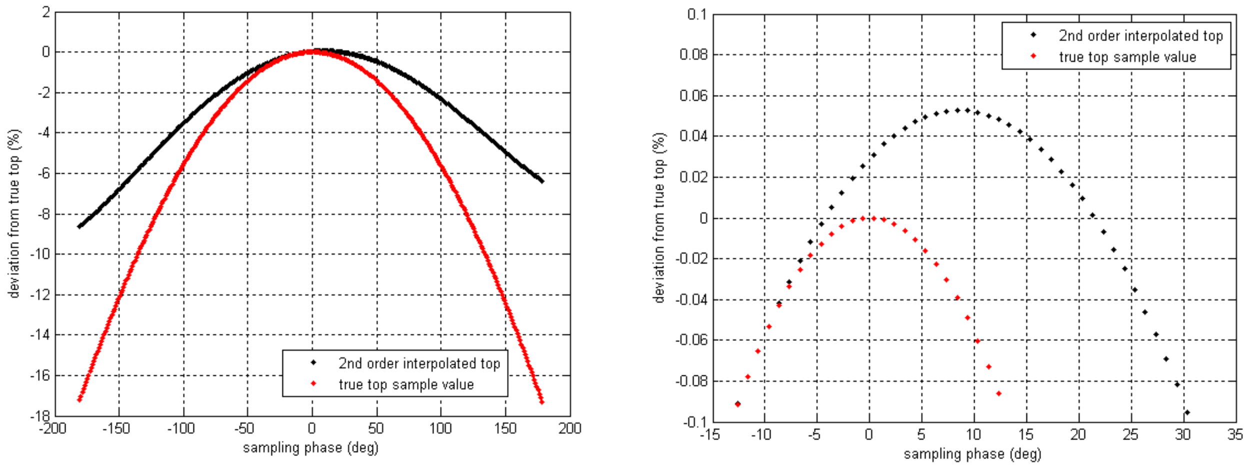


Figure 3-3: Relative deviation from pulse top value versus sampling phase offset for the top sample (no interpolation) and the interpolated top. Right: full phase range, Left: zoomed view

If the ADC clock phase is unknown (as is the case when measuring the first bunch after startup) the interpolated pulse peak is within 10% of its true value. When sinc-interpolation is used this relative error would be about 1% (see Figure 2-3). The direct top sampling method (no interpolation) results in a relative error of up to 20%.

3.2 IF I/Q pulse interpolation

In section 3.1 the 2nd order polynomial interpolation of as single IF pulse (I- or Q-channel) has been discussed. However, the interpolation is only reliable if a certain signal to noise ratio is guaranteed with the input samples. This is practically only so with the data from the reference channel, where the S/N ratio is best. The position channels do have very limited S/N ratios when the beam is near the beam pipe center (as will be the case under normal operation), resulting in a large interpolation error. In addition, the S/N-ration of a single output channel is small when the I-Q phase is located at near multiples of 90 degrees. In this case either I- or Q channel signal RFFE output signal is small, as is the S/N ratio. But the I-Q-vector magnitude remains constant with a high S/N ratio. Therefore it is more robust to interpolate the vector magnitude information in order to derive information about the actual ADC clock phase.

The I-Q-vector squared magnitude is calculated as

(12) $$s(n) = R_I^2(n) + R_Q^2(n).$$

A more easy to implement function with comparable time-domain characteristic is the sum of absolute values given by

(13) $$\tilde{s}(n) = \left|R_I(n)\right| + \left|R_Q(n)\right|.$$

Eq. (12) and (13) will be used in combination as basis for signal interpolation and estimation.

Figure 3-4shows interpolated signals:

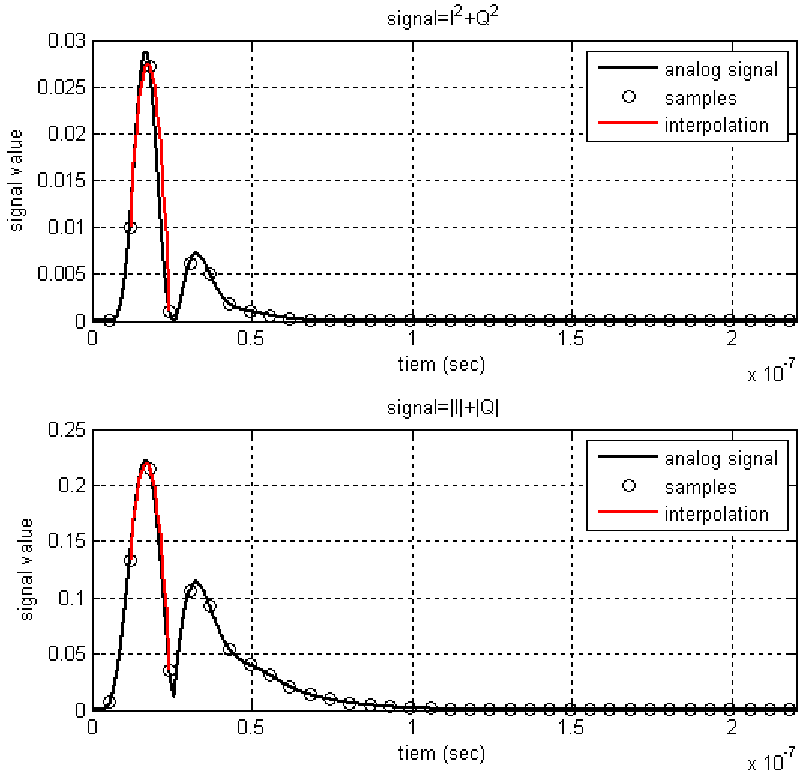


Figure 3-4: Signal interpolation (top: s=I^2+Q^2, bottom: s=|I|+|Q|

Sweeping the signal sampling phase in the Matlab simulation and comparing the interpolated sampling phase to the true sampling phase results in the plot Figure.

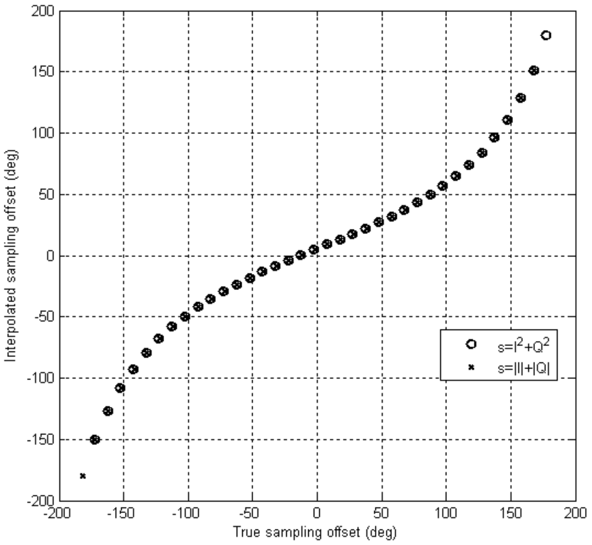


Figure 3-5: True sampling phase vs. interpolated sampling phase

This plot relates the interpolated (estimated) phase to the real phase deviation from the pulse peak and can be used as the basis for a look-up table (see Figure 3-6).

The general flow for ADC clock offset estimation and correction is seen in Figure 3-6.

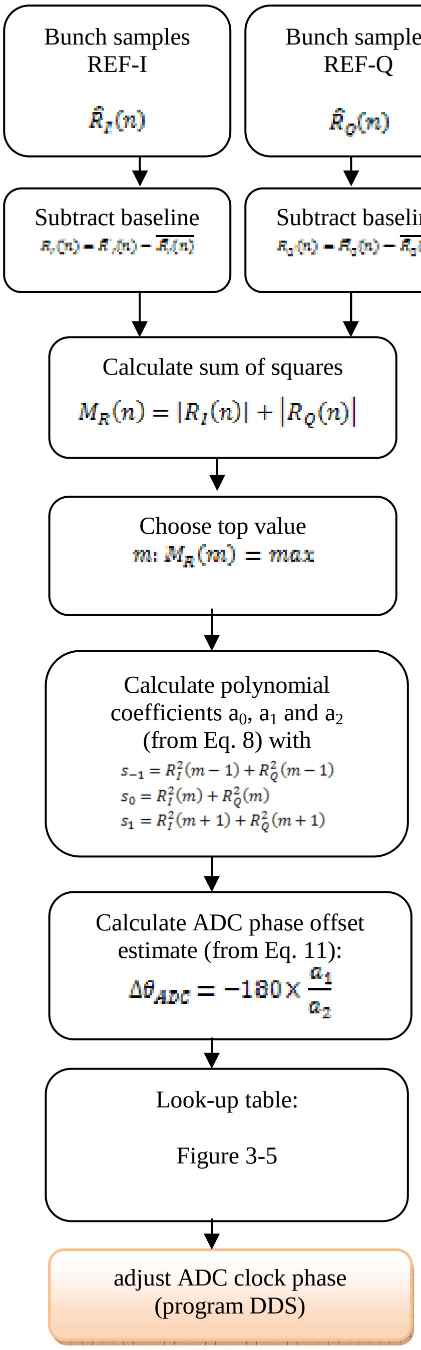
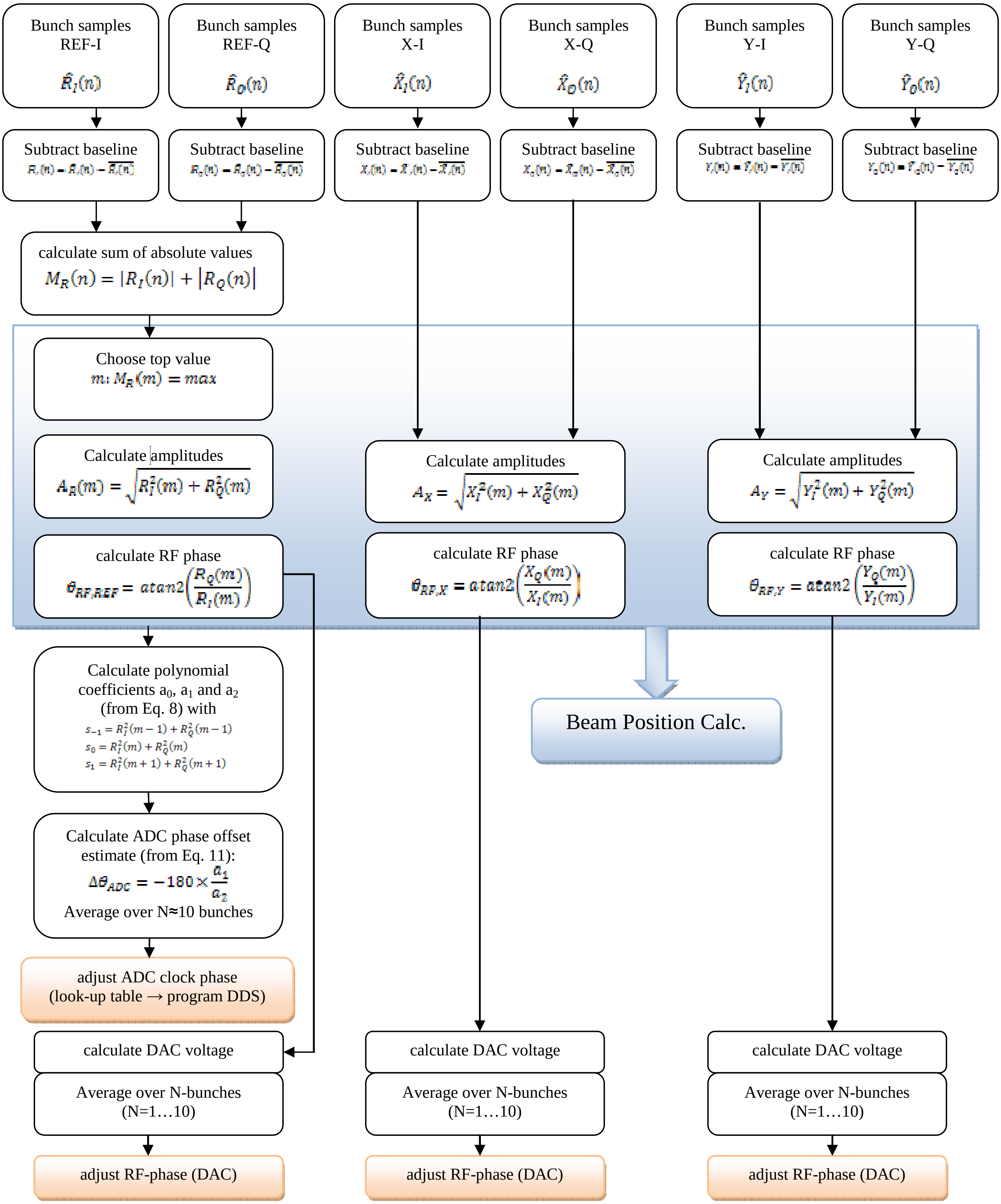

Bunch samples
REF-I
$\hat{R}_I(n)$
Bunch samples
REF-Q
$\hat{R}_Q(n)$
Subtract baseline
$R_I(n) = \hat{R}_I(n) - \overline{\hat{R}_I(n)}$
Subtract baseline
$R_Q(n) = \hat{R}_Q(n) - \overline{\hat{R}_Q(n)}$
Calculate sum of squares
$M_R(n) = |R_I(n)| + |R_Q(n)|$
Choose top value
$m{:}\ M_R(m) = max$
Calculate polynomial coefficients a_0, a_1 and a_2 (from Eq. 8) with
$s_{-1} = R_I^2(m-1) + R_Q^2(m-1)$
$s_0 = R_I^2(m) + R_Q^2(m)$
$s_1 = R_I^2(m+1) + R_Q^2(m+1)$
Calculate ADC phase offset estimate (from Eq. 11):
$\Delta\theta_{ADC} = -180 \times \frac{a_1}{a_2}$
Look-up table:
Figure 3-5
adjust ADC clock phase
(program DDS)

Figure 3-6: ADC clock phase error estimation

4 Basic Flow for Sampling Data Processing

Figure 4-1: General sample processing flow


Bunch samples REF-I $\hat{R}_I(n)$
Bunch samples REF-Q $\hat{R}_Q(n)$
Bunch samples X-I $\hat{X}_I(n)$
Bunch samples X-Q $\hat{X}_Q(n)$
Bunch samples Y-I $\hat{Y}_I(n)$
Bunch samples Y-Q $\hat{Y}_Q(n)$
Subtract baseline $R_I(n) = \hat{R}_I(n) - \overline{\hat{R}_I(n)}$
Subtract baseline $R_Q(n) = \hat{R}_Q(n) - \overline{\hat{R}_Q(n)}$
Subtract baseline $X_I(n) = \hat{X}_I(n) - \overline{\hat{X}_I(n)}$
Subtract baseline $X_Q(n) = \hat{X}_Q(n) - \overline{\hat{X}_Q(n)}$
Subtract baseline $Y_I(n) = \hat{Y}_I(n) - \overline{Y_I(n)}$
Subtract baseline $Y_Q(n) = \hat{Y}_Q(n) - \overline{Y_Q(n)}$
calculate sum of absolute values $M_R(n) = |R_I(n)| + |R_Q(n)|$
Choose top value $m: M_R(m) = max$
Calculate amplitudes $A_R(m) = \sqrt{R_I^2(m) + R_Q^2(m)}$
Calculate amplitudes $A_X = \sqrt{X_I^2(m) + X_Q^2(m)}$
Calculate amplitudes $A_Y = \sqrt{Y_I^2(m) + Y_Q^2(m)}$
calculate RF phase $\theta_{RF,REF} = atan2\left(\frac{R_Q(m)}{R_I(m)}\right)$
calculate RF phase $\theta_{RF,X} = atan2\left(\frac{X_Q(m)}{X_I(m)}\right)$
calculate RF phase $\theta_{RF,Y} = atan2\left(\frac{Y_Q(m)}{Y_I(m)}\right)$
Beam Position Calc.
Calculate polynomial coefficients a_0, a_1 and a_2 (from Eq. 8) with
$s_{-1} = R_I^2(m-1) + R_Q^2(m-1)$
$s_0 = R_I^2(m) + R_Q^2(m)$
$s_1 = R_I^2(m+1) + R_Q^2(m+1)$
Calculate ADC phase offset estimate (from Eq. 11): $\Delta\theta_{ADC} = -180 \times \frac{a_1}{a_2}$
Average over N≈10 bunches
adjust ADC clock phase (look-up table → program DDS)
calculate DAC voltage
Average over N-bunches (N=1…10)
adjust RF-phase (DAC)
calculate DAC voltage
Average over N-bunches (N=1…10)
adjust RF-phase (DAC)
calculate DAC voltage
Average over N-bunches (N=1…10)
adjust RF-phase (DAC)

5 ADC Clock Phase Correction Feedback Loop

Each bunch within the bunch train is processed as has been graphically defined in Figure 4-1.
This means that each bunch will generate an estimated ADC clock offset value that is calculated from equation (11). In Figure 5-1 those values are shown as black dots. After the last bunch has been received and processed individually an average over all the estimated ADC phase offsets over the bunch train is calculated (blue dots).

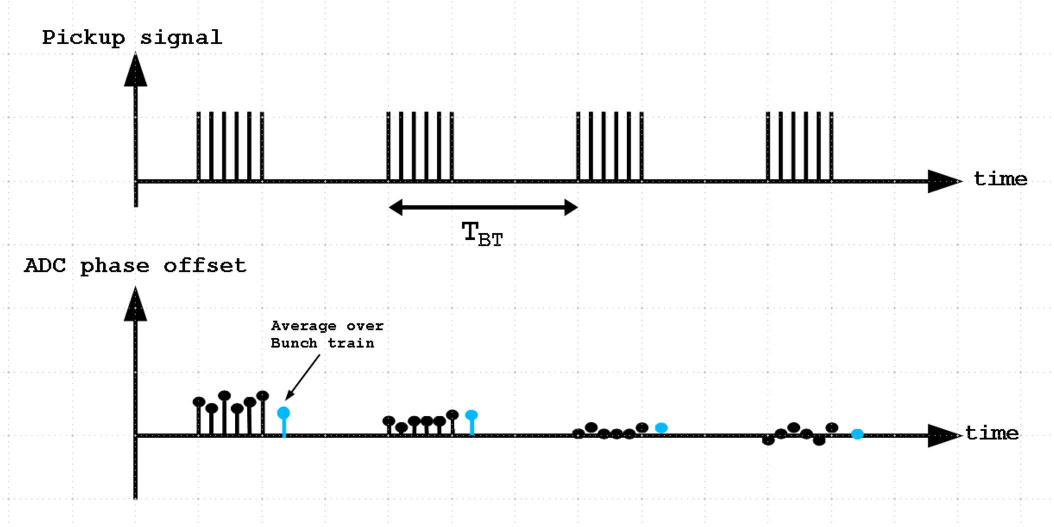


Figure 5-1: ADC offset value processing

This means that we take a sample of the averaged phase actually every T_{BT} seconds, where T_{BT} is the bunch train repetition rate.

Then after a further averaging over N bunch-trains the DDS phase is reprogrammed as can be seen in Figure 5-2.

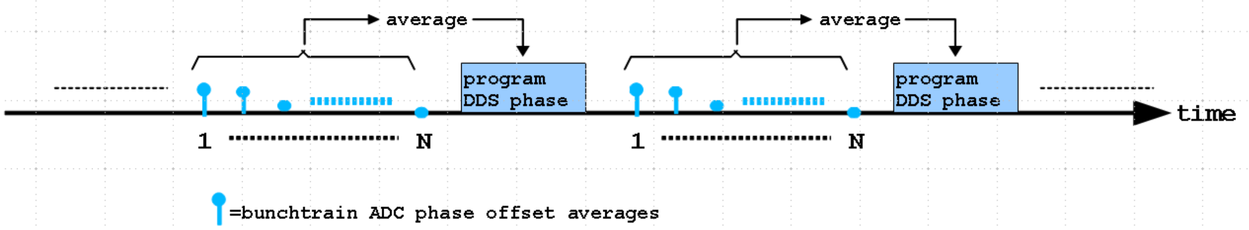


Figure 5-2: Bunch phase average processing

The convergence behavior of this processing scheme has been simulated, and result is shown in Figure 5-3. The initial ADC phase has been chosen randomly between 0 and 360 degrees. Additionally, excessive ADC phase noise and additive sampling noise is also considered in order to demonstrate robustness.
The ADC clock phase has been updated by directly correcting for the offset (no look-up table).

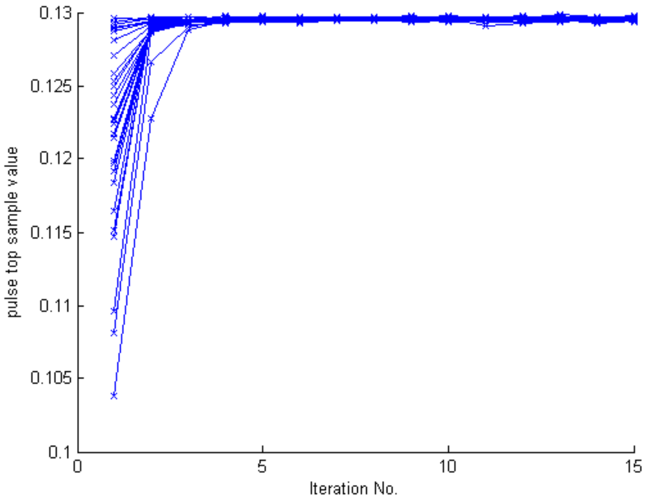


Figure 5-3: Convergence simulation with N=2

It can be concluded that convergence is reached after a maximum of 4 iterations with an averaging buffer size N of only 2.
The matlab code segment that calculated the ADC phase update is shown below for illustration.

```
    sumofsquares=(Qsamples.^2+Isamples.^2);

    %------------ estimate sampling phase ------
    [maxval,maxpos]=max(abs(Qsamples)+abs(Isamples));
    a=[0 1 0;-0.5 0 0.5;0.5 -1 0.5]*...
        [sumofsquares(maxpos-1) sumofsquares(maxpos) sumofsquares(maxpos+1)]';

    figure(1);subplot(311);
    plot(sampleoutput.sampling_times,sumofsquares,'o',...
        output.time,(output.if_I).^2+(output.if_Q).^2);

    figure(1);subplot(312);
    plot(sampleoutput.sampling_times,sqrt(sumofsquares),'o',...
        output.time,sqrt((output.if_I).^2+(output.if_Q).^2));

    phase=-180*a(2)/a(3);
        phaseerror(n)=phase;
            subplot(313);plot(phaseerror);

    %------------ update circular buffer ---------
    buffer_pointer=mod(n,circular_buffer_size)+1;
    circular_buffer(buffer_pointer)=phase;

    %----------- calculate phase correction ------
    delta_phase=sum(circular_buffer)/circular_buffer_size/1;

    if mod(n,circular_buffer_size)==0;
    sampleinput.sampling_delay=sampleinput.sampling_delay+...
        sampleinput.samplinginterval/360*delta_phase+...
        sampling_time_variance*randn;

    m=m+1;
    A(m)=sqrt(Isamples(maxpos)^2+Qsamples(maxpos)^2);
    circular_buffer=zeros(1,circular_buffer_size);
    end
```

6 I/Q Imbalance Correction

Any analog implementation of an I/Q downconverter suffers from imbalance both in amplitude and phase. If the RF and LO signals are sine waves having a difference frequency Δω, then the baseband signals out of an ideal downconverter are

(14)
$$\begin{aligned} I(t) &= \cos(\Delta\omega \cdot t) \\ Q(t) &= \sin(\Delta\omega \cdot t) \end{aligned} \quad .$$

Including the phase-imbalance (Δø) and amplitude-imbalance (A) of the Q-channel:

(15)
$$\begin{aligned} I^{'}(t) &= \cos(\Delta\omega \cdot t) \\ Q^{'}(t) &= A \cdot \sin(\Delta\omega \cdot t + \Delta\phi) = A \cdot \sin(\Delta\omega \cdot t) \cdot \cos(\Delta\phi) + A \cdot \cos(\Delta\omega \cdot t) \cdot \sin(\Delta\phi) \end{aligned}$$

Therefore, I’ and Q’ are linearly dependent on I and Q (ideal components) and can be written as

(16)
$$\begin{pmatrix} I' \\ Q' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ A \cdot \sin(\Delta\phi) & A \cdot \cos(\Delta\phi) \end{pmatrix} \cdot \begin{pmatrix} I \\ Q \end{pmatrix} .$$

If we want to go from the I’/Q’ values including imbalance to the corrected values I and Q one has to multiply with the inverse matrix:

(17)
$$\begin{pmatrix} I \\ Q \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\tan(\Delta\phi) & \dfrac{1}{A \cdot \cos(\Delta\phi)} \end{pmatrix} \cdot \begin{pmatrix} I' \\ Q' \end{pmatrix}$$

The numbers A and Δø are constant parameters of the RFFE. They are furthermore expected to be independent of the current attenuator setting in the RF part of the front-end (version 3).
Since one expects no more than ±2dB of amplitude imbalance and ±5 deg of phase imbalance the matrix elements are bounded: -0.09<tan(Δø)<+0.09 and 0.8<1/(A·cos(Δø))<1.26.

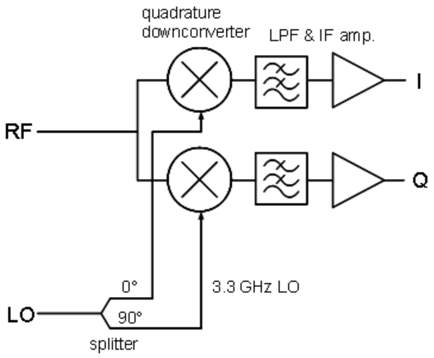


Figure 6-1: I/Q downconverter

The imbalance terms can also be attributed to the Q channel only. In this case the transform matrix in (17) is simply flipped antidiagonally.
Figure shows an example of I/Q imbalance correction. The original (blue) I/Q samples are processed by as in (17). The resulting I/Q samples show no imbalance any more.

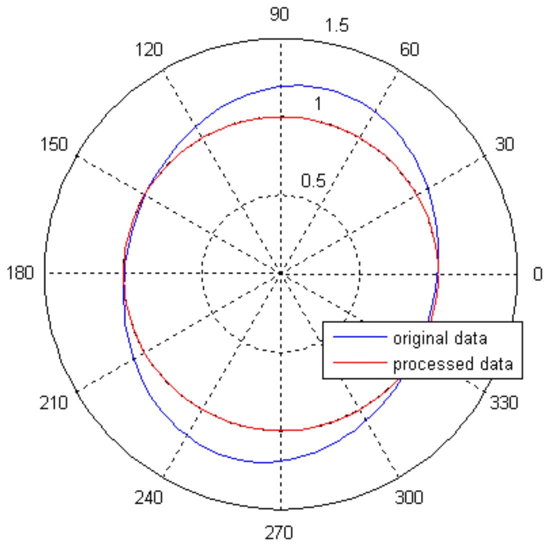


Figure 6-2: Data before and after I/Q balancing (original imbalance: 2dB, 5deg)

7 Sensitivity Improvement using Matched Digital Filtering

The calculated beam position is proportional to the ratio of the IF signal vector magnitudes of position and reference channel. The vector magnitude can be simply calculated using the top samples of I and Q (see Figure 7-1)

This approach is simple and fast. But since only one sample per waveform is considered one expects that this is not the optimum in terms of signal-to-noise ratio. One could, for example, average the three samples around the top and use this averaged value as the measure of signal amplitude. It can be shown that this increases the S/N-ratio by about 15%.

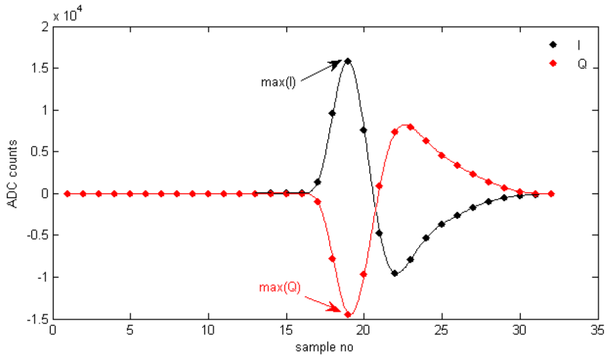


Figure 7-1: I and Q ADC signals

The maximum S/N-ratio possible using linear filtering is achieved with a matched FIR filter. This matched filter is located in the digital domain after the I/Q-balancer, as shown in Figure 7-2. Its output sequence contains a peak sample where the signal-to-noise ratio is maximized. The matched filter output sequence can be processed in exactly the same manner as has been done previously for the direct signal samples in Figure 7-1.

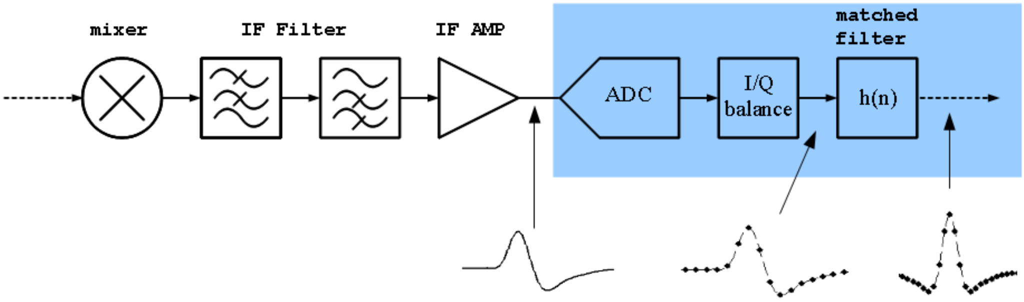


Figure 7-2: System overview

Assuming that the ADC input noise samples are uncorrelated[1] the matched FIR filter impulse response h(n) is given by a time-inverted version of the signal that is to be filtered:

(18) $$h(n) = a \cdot s_p(-n)$$

The constant factor a may be chosen arbitrarily. $s_p(n)$ is a prototype signal that can, for example, be obtained by simulation or (preferably) form averaged measurement data taken under real beam conditions.

The matched filter output signal y(n) is calculated by folding the input signal s(n) with the filter impulse response h(n). Since the latter is, apart from a multiplicative constant a, identical to the time-inverted signal, the peak value of the output signal y(n) is given by

(19) $$y_{peak} = \sum_{\forall n} s(n) \cdot h(-n).$$

Since the multiplicative constant *a* can be chosen arbitrarily we select it such that the sum of squares $\Sigma h(n)^2 = 1$. Then the output rms noise δ_y is calculated by

(20) $$\delta_y{}^2 = \delta_i^2 \cdot \sum_{\forall n} h(n)^2 = \delta_i^2 .$$

The signal-to-noise ratio improvement with respect to the simple top-sampling method then becomes simply

(21) $$\frac{\left.\frac{S}{N}\right|_{\text{with matched filter}}}{\left.\frac{S}{N}\right|_{\text{no matched filter}}} = \left(\frac{\sum_{\forall n} s(n) \cdot h(-n)}{s_{\text{top}}} \right)^2 .$$

s_{top} is the input signal peak sample (which can be positive or negative). The square of the ratio on the left side in (21) is taken only because the S/N ratio is normally defined as a ratio of powers.

[1] This assumption is practically met in the current RFFE versions with an IF bandwidth of ≈40MHz.

An example using measured waveform data is seen in Figure 7-3.

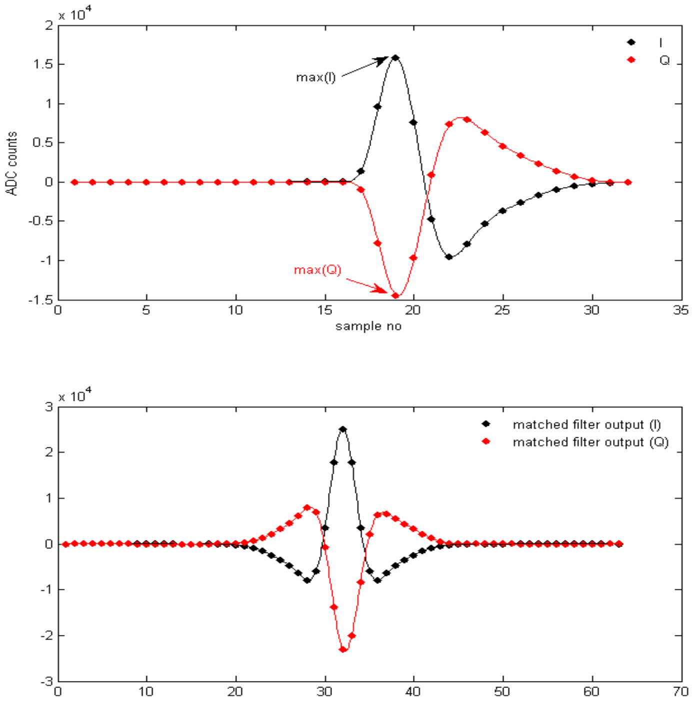


Figure 7-3: Input signals (top) and matched filter output (bottom)

The measured signal samples of the I channel (black samples in Figure 7-3, top) have been chosen as the prototype signal $s_p(n)$.

Then, the matched filter impulse response was calculated:

$$h(-n)=\frac{s_p(n)}{\sqrt{\sum_{\text{all n}} s_p(n)^2}}\ . \tag{22}$$

This enforces that the condition $\Sigma h(n)^2=1$ is met.

Figure 7-3 (bottom) show the output signals obtained by FIR-filtering the input I and Q signals with the matched filter that has the just calculated impulse response h(n).
The maximum signal-to-noise ratio occurs, again, at the output pulse peaks.

The evaluation of (21) with the values obtained from measured signal samples shows that the S/N ration can be improved by 4dB with respect to the simple top-sampling method.

This represents the maximum improvement that can possibly be achieved using linear digital filtering. All other linear filters, as for example averaging over a number of samples, results in an improvement below 4dB.

This result also means that the resolution can be improved by a factor of 1.6 when using matched digital filters.

A1 Document History

03-05-2011	SK84	v1	initial
05-05-2011	SK84	v1r1	- Corrections on Figure 4-1 in Box “Calculate polynomial coefficients…” - Corrections in Figure 3-6 in Box “Calculate polynomial coefficients…”
05-05-2011	SK84	v1r2	- Corrected Eq. 11 - Page 7: sign reversed in definition of phase error. Now in accordance with 11, Figure 3-6 and Figure 4-1
23-01-2012	SK84	v1r3	- changed calculation of coefficients s-1, s0 and s1 - added ADC phase feedback loop description and simulation
10-10-2012	SK84	v1r4	- added chapter 6: I/Q Imbalance Correction
30-10-2012	SK84	v1r4	- added chapter 7: Sensitivity Improvement using Matched Digital Filtering