5. ERROR CORRECTION IN A/D CONVERTERS

2-D correction LUT
- Table filling techniques

Pipeline converters
- Gain errors in pipeline
- Redundant codes
- Correction kernel for pipeline

Sigma-delta converters
- MASH ΣΔ converter
- Digital error correction in multi-bit sigma-delta ADCs

Others
- Histogram analysis
- Errors in flash ADC
- Parallel quantisers - estim. channel gain, offset & timing offset

2-D LOOK-UP-TABLE CORRECTION

Look-up-table based correction technique is generic and can be applied to any type of ADC - however, the structural information can be employed to speed up the collection of data by modelling larger than just local errors.

If errors can be measured, they can be corrected by using a look-up-table (LUT). To compensate dynamic errors, we need to know how fast the signal is increasing or decreasing. Because of this, the memory is addressed by both the signal and its derivative, or present and previous sample - the latter being the most simple to implement.

The LUT is normally trained by using known sinusoidal 1- or 2-tone test signals. The task is divided to:
- filling the data
- estimating the errors
TRAINING METHODS

Direct INL Mapping
The most simple way to build the correction table is to estimate or measure the INL error and store it directly to the entry addressed by \( x(k), x(k-1) \).

INL update in the neighbourhood
To speed up the training, the measured INL result is usually updated (weighted by a certain spreading function) in some neighbourhood around \( x(k), x(k-1) \).

Base functions for INL
Direct INL mapping is slow, and if any information of the error is available, various states can be grouped together by using base functions that model the error typical in that type of ADC. Base functions can be gaussian bumps or pyramid shapes, for example, and only the amplitude of the base functions is searched.

\[
\text{error}(i,j) = \text{desired}(x,y) - \text{table entry}(i,j)
\]

\[
\text{update}(i,j) = \text{error}(i,j) \times \text{Gaussian mask}(i,j)
\]

SPREADING FUNCTIONS

To speed up training, an error measured in point \( x,y \) is updated also in a larger region in its neighbourhood. This creates problems with entries that already have some non-zero value and here, the following is done:

- the desired value is calculated in the center point of update region
- difference (error) between table entry and the desired value are calculated for all table entries within the update region
- this error plane is smoothed with pyramid or Gaussian mask so that far-away points will be updated only little
- table values are updated with the update value

Thus the nearest points are updated towards the desired value but never past is.
SOME LESSONS

In the example left, a 2-D LUT is trained using a 2-tone test signal and exponential spreading function. An 8-bit pipeline ADC was linearised at various 3-tone signals, and the example left shows a couple of typical errors:

- The training signal does not cover the corners of the table. Left is shown a situation where a high-frequency test signal sweeps outside from the correction area: the circled results are outside the training area and cause lot of spurious components.
- Another feature can be seen in the corrected INL. The INL of a pipeline ADC has stepwise changes (top curve) which are smoothed by the exponential spreading curve. This leaves impulse-like INL at the boundaries as shown with a dashed circle.
- Optimum width of the Gaussian spreading function shown left seems to be $L=3.5-4$ for an 8-bit flash and 4-5 for an 8-bit pipeline ADC.

**USING BASE FUNCTIONS**

The direct filling used above needs the information of state-wise INL. Alternatively, the entire table can be filled using a set of some base functions by optimising the level of harmonic and IM distortion, for example. In this case, much less data is needed.

On the left, an M sample record is taken. $x$ and $y$ are the current and previous sample, respectively, and $g_j()$ is a Gaussian base function in a certain position depending on $j$. Error $e_H$ is formed as a sum of $N$ (e.g. 50) first harmonics from FFT of the measurement, and it is modelled as a sum $C \gamma$ of $L$ base functions $g()$ evaluated in each time point. Now, the amplitude vector $\gamma$ can be solved in LMS sense, and note that the results of several test tones can be combined in the error sums $G$ and $r$.

$$ (Friel, Hummels, Irons) $$

---

(c) 2001- Timo Rahkonen, University of Oulu, Oulu, Finland
BASE FUNC CONT...

The figure left illustrates how \( e_H \) is formed as a sum of harmonic and spurious components. The figure below shows a grid of 3 x 6 gaussian base functions in a x-y plane. By solving \( \gamma \), the amplitudes for these base functions are obtained.

NOTE: TABLES VS. POLYNOMIAL

To linearise ADC converters, also 3rd or 5th order Volterra filters have been experimented. However, Tsimbinos et al. have shown that a low-order post-distoster is not capable of correcting all the nonlinearity errors, as typically the nonlinearity is of very high order.

PIPELINE ADC

Redundancy

Most subranging ADCs employ some redundancy that already corrects interstage offset errors and errors in the ADC decision levels.

Gain error correction

As shown in chp 3, interstage gain errors appears as an error proportional to the state of the stages. Thus, an easy correction algorithm exists, provided that the gain errors can be measured.

Sample-to-sample memory

Pipeline and algorithmic converters have a built-in memory of several samples. Depending on the circuit structure, sample-to-sample memory may appear.
REDUNDANCY

In subranging converters, accurate matching of the subconverters is impossible. Thus, each subconverter has certain amount of redundancy to avoid clipping. Typical redundancies are

- No redundancy. In this case offset errors result in clipping of the latter stages. A common solution to this is to use non-integer radix (e.g. interstage gain of 1.95 instead of 2.00)
- 1 extra level. This allows 1/Gain lsb offset error in interstage summation
- 1 extra bit (6 dB range). This is simple to arrange and gives total of 6 dB margin for offset errors.

Redundant outputs are combined as shown on the next slide.

EXAMPLE: 10-B 2-STEP ADC

Suppose a 4+6 bit 2-step ADC, where the 1st stage makes 4-b quantisation and forwards a residue voltage \( V_2 \) (\( \text{abs}(V_2) < 64(V_{r1}/2) \)) to the 2nd stage that makes the fine quantisation.

Now the threshold errors of the 1st ADC are studied. Depending on the level of threshold \( V_{th1} \), the output of the 1st stage is \( X \) or \( X+1 \). Now the input voltage \( V_2 \) to the 2nd stage is either one of (\( G=64 \))

\[
V_2 = 64(V_{in} - (X + 1) \cdot V_{r1})
\]

\[
V_2 = 64(V_{in} - (X \cdot V_{r1}))
\]

that differ by \( 64V_{r1} \) which is a complete full scale of the latter ADC. To allow 0.5 lsb threshold errors in the 1st stage, the latter stage must be able to handle results that may be \( 64 \times 0.5 = 32 \) wrong. This means excess range of one full redundant bit. With the help of this extra bit, the total output is either

\[
\text{out} = 64(X + 1) - (G \cdot ((1 \cdot V_{r1})/V_{r2})) + Y_{res}
\]

\[
\text{out} = 64X + Y_{res}
\]

which are equal if \( GV_{r1}/V_{r2} = 64 \) exactly. (Above \( Y_{res} = G(V_{in} \cdot XV_{r1})/V_{r2} \).)
EXAMPLE CONT ...: COMBINING

It was seen, that excess dynamic range in the latter ADC helps to cancel threshold errors in the 1st ADC, but the 1st DAC and interstage gain need to be very accurate. Next question is how to combine the results. This is simply done by weighting and summing

\[ \text{out} = 64 \cdot X + Y \]

which can be using either signed or unsigned digits. As an example, consider that a correct value for ADC2 is +17 (signed) or 81 (unsigned). In both cases the msb of ADC is used to decide whether X should be decreased/increased or not.

In case of unsigned digits, this results simply in summing the overlapping digital words X and Y. In case of signed digits, a sign extension of negative numbers is needed. With signed digits this is commonly called RSD (Redundant Signed Digit) error correction.

REDUNDANCY IN 1-B PIPELINE ADCS

1-bit/stage pipeline converters can be in both redundant and non-redundant form. The last residue is

\[
V_{\text{res}} = G_0(G_1G_2V_{\text{in}} + c_3V_R) + c_2V_R + c_1V_R
\]

where \(c_i = -2D_i\) for 1-level and \(-2D_i\) for 2-level implementations. Replacing \(V_{\text{res}}\) with \(D_0V_R\) and assuming \(G_i = 2\), we can solve for \(V_{\text{in}}\):

\[
\frac{V_{\text{in}}}{V_R} = \frac{1}{8} \cdot (D_0 + 2D_1 + 4D_2 + 8D_3 - 7)
\]

or \((-7/8..8/8)V_R\) input range for 1-level implementation, and

\[
\frac{V_{\text{in}}}{V_R} = \frac{1}{8} \cdot (D_0 + 2D_1 + 4D_2 + 8D_3)
\]

or \((-15/8..15/8)V_R\) input range for 2-level implementation. Hence, redundant converter has twice as large input range, which is typically used as spare room for variations and overdrive headroom.
AMOUNT OF REDUNDANCY IN PIPELINE ADC

Due to 3-level coding, a 1-b/3-level (1.5 bit) pipeline ADC has $3N$ possible output states, while only $2N$ will be used. This creates lots of redundant states.

The redundancy is removed in the error correction that simply implements

$$\text{out} = \sum_{j=1}^{M} 2^{(j-1)} \cdot \text{out}_j$$

As noted before, the stage outputs can be considered as signed (-1,0,1) or unsigned (0,1,2) integers, and the error correction logic for unsigned stage outputs is shown.

DIGITAL SELF-CALIBRATION

One of the first digital correction systems was presented by Karanicolas (IEEE JSSC December 1993). Starting from the lowest stage corrected, $\text{Vin}$ is forced to zero and $D(i)$ to 0 and 1 to measure levels $S1$ and $S2$ with the remaining stages of the converter. Now the output can simply be reconstructed by adding the difference $D(S1-S2)$ where $D=0$ or 1 to the result of the latter stages.

After calibrating all stages from lowest towards msb, the total output can be formed as

$$Y = X + \sum_{i=0}^{NC} (S_{1i} - S_{2i}) \cdot D_i$$

where $X$ is the uncorrected output, $D_i$ is the output of stage $i$, $NC$ is the number of corrected stages and measure weight $S_{1i}-S_{2i}$ is used instead of assumed power of 2. Now it is also easy to use non-radix-of-2 base, and base 1.93 was employed in 12 first stages to avoid missing codes due to overflowing residues.
Essentially, Karanicholas has used in each calibrated stage the exact weight $\Delta_i = S1,i - S2,i$ instead of the assumed power of two. Alternatively, the same correction can be implemented by using radix-2 weights, which simplifies the combining of the uncorrected data $X_{raw}$, and then summing the errors $\epsilon_i$ to the correct weights. This can be done where-ever (not necessarily on-chip) the comparator data is available, and storing the error instead of the actual weight reduces the word length of the correction term $\epsilon_i$.

**Example**

Actual weight of an output bit is 2.0469 instead of 2.00. This corresponds to a digital value $\Delta$ of 10.0000110. The error $\epsilon_i$ can be stored as a 3-bit number, $\Delta$ as an 9-bit number.

---

**ACCURACY BOOSTING**

Soenen&Geiger extended the technique to multi-bit stages. Here, the height of each step is measured again using the latter stages of the pipeline, and the weight of code $n$ is sum all the steps leading to it. These weights are calculated and stored into local memories.

The calibration is started from the last stage to be calibrated, and usually, a single non-iterated calibration cycle is sufficient. Enough resolution is needed in the latter stages, and to maintain this, several approaches can be used:

- recycle the stages so that the residue of the lsb stage is fed as an input to the msb stage
- use additional lsb stages for calibration purposes only as in Karanicholas
- temporarily increase the gain between the calibrated and following stages

(Soenen, Geiger: An Architecture and an algorithm for fully digital ..., IEEE TCAS-II, March 95, pp.143-153)
CORRECTING AMPLIFIER NONLINEARITY

Plain redundancy corrects the effects of comparator and amplifier offsets. Previous correction techniques cancel the errors caused by capacitor mismatch and finite DC gain. Also in pipeline ADCs, the fact that DC gain varies with output voltage causes distortion, and this can be compensated with similar table based technique.

One technique is based on performing piecewise-linear measurements of the actual gain

(Nagaraj, US pat 6232898)

CORRECTION EXAMPLE

Similar technique can be applied in pipeline converters with redundancy, where gain and Vref errors can be corrected by multiplying the (-1,0,+1) output of each stage by a coefficient \( c_i \) and summing it to the output word:

\[
Y = X_{\text{raw}} + \sum_{i=0}^{NC} c_i \cdot D_i
\]

Now measuring or estimating the step height \( \Delta_i \) at \( V_{\text{in}}=0 \) (point a) may overdrive the rest of the converter, and a bias closer to switching point Vth (point b) may be preferred. However, the calibration procedure is similar: last corrected stage is calibrated first, and its corrected result is used to calibrated by the next stage towards input. The correction \( c_i \) can be calculated as a difference of \( \Delta_i \) and ideal step height.

The correction logic per stage is very simple: one addition of constant multiplied by -1/0/+1. A divide-by-2 may used to keep same order of magnitude for all coefficients \( c_i \).
Example

On the left, measured results from a 14-bit, 300 kS/s pipeline ADC are corrected. Top figures show the non-corrected spectrum and INL of the ADC, and in the figures below, gain errors in 4 msb stages are corrected. In this case, SFDR improved by 16 dB (from 63 to 79 dB) and the largest remaining tones are 2nd and 3rd harmonics produced in the sampling switches.

See also Lee: A 12-bit 600 kS/s .... j SSC, April 1994. Here capacitor mismatches in each stage are calibrated by measuring separately the offset for zero input, and output for Vref input and Di forced to 1. The difference of these is directly $\alpha V_{\text{ref}}$, where $\alpha$ is the capacitor mismatch ratio.

\[ \text{SNR} / \text{SFDR} / \text{ENOB} \]

\[ 55.2/63.1 \text{ dB} / 8.9 \text{ bit} \]

Original

\[ 60.8/79.1 \text{ dB} / 9.8 \text{ bit} \]

Corrected

(c) 2001- Timo Rahkonen, University of Oulu, Oulu, Finland

Direct fitting to measured data

\[ \begin{bmatrix} D_0(1) & \ldots & D_n(1) \\ \vdots & \ddots & \vdots \\ D_0(N) & \ldots & D_n(N) \end{bmatrix} \]

\[ c = \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \]

\[ y = \begin{bmatrix} y(1) - y_{\text{est}}(1) \\ \vdots \\ y(N) - y_{\text{est}}(N) \end{bmatrix} \]

\[ c = (D^T D)^{-1} (D^T y) \]

FITTING THE COEFFICIENTS

Besides by DC measuring the level heights, the correction coefficients can also be fitted. Two methods are shown on this page. In both methods, ideal sine yest is fitted to measured signal y, and correction coefficients $c_0$-$c_n$ are searched so that instantaneous $\text{INL}=y(i)-y_{\text{est}}(i)$ is modelled by comparator outputs $d_0$-$d_n$ from first stages. The routine below supports on-line adaptation of a double sampling AD where even and odd samples have different coefficients. Direct fitting on the left above can be used for a measured data sequence, for example.

\[ \text{NT} = \text{length}(y) \; ; \; \mu = 5 \cdot 10^{-3} \; ; \; \text{modul}=2; \]

\[ \text{yest} = \text{fitsine}(y, \text{freq}, 1/\text{clk}); \; \text{INL} = y - \text{yest}; \]

for $i=1:\text{NT}$

\[ \text{px} = 1 + \text{mod}(i, \text{modul}) \; ; \; \% \; 1,2,1,2,1,2,... \]

\[ \text{corr} = c_0(\text{px}) \cdot d_0(i) + c_1(\text{px}) \cdot d_1(i); \]

\[ \text{err} = \text{INL}(i) - \text{corr}; \]

\[ c_0(\text{px}) = c_0(\text{px}) + \mu \cdot d_0(i) \cdot \text{err}; \]

\[ c_1(\text{px}) = c_1(\text{px}) + \mu \cdot d_1(i) \cdot \text{err}; \]

end
SLOW SHADOW ADC

The use of parallel and accurate but slow shadow ADC has been proposed in many papers. At every Nth sample (depending on the speed of the shadow ADC) the results of the shadow and real ADC are compared and a corrector is adapted. In the example left, the stage i (AD(i)) is adapted by comparing the results of the shadow AD (SHAD) and back-end stages (AD-BE). LMS algorithm can be used to adapt the linear corrector:

\[
\alpha(i) = \alpha(i-1) + \mu \cdot \varepsilon \cdot D \\
\beta(i) = \beta(i-1) + \mu \cdot \varepsilon
\]

which also converges (although slower) without the actual multiplications:

\[
\alpha(i) = \alpha(i-1) + \mu \cdot \text{sign}(\varepsilon) \cdot \text{sign}(D) \\
\beta(i) = \beta(i-1) + \mu \cdot \text{sign}(\varepsilon)
\]

see also US pat 4903026

PIPELINE: PN TEST SIGNAL

One way to measure the interstage gain is to use a known pseudo-random (PN) test signal. Left, it is injected in the residue voltage of the first stage in a multi-stage sub-ranging converter. This +/- 0.25 injection increases the range of the residue, and more bits are needed in the 1st DAC.

Interstage gain G0 is evaluated by correlating the result of ADC1 with the PN sequence, taking long-term average (2^{23} samples, this de-correlates the input signals) and scaling. This gives an estimate for G0, and its inverse is used to scale the results of ADC1

(Siragusa&Galton, Elect.Letters March2000)
SIGMA-DELTA ADCS

ΣΔ modulators have a large amount of spectrally shaped quantization noise. Some architectures are based on measuring and cancelling this noise with another ADC. On the other hand, multi-bit structures are sensitive to mixing problems: DAC non-linearities mix this quantization noise back to baseband, thus reducing SNR.

Quantisation error cancelling techniques

- Leslie-Singh architecture
- MASH architecture

Reducing non-linear effects in multi-bit sigma-delta converters

- In multibit ΣΔ DA converters, dynamic element matching (DEM) or trimming are the only ways to reduce non-linear effects.
- In multibit ΣΔ AD converters, either DEM, trimming, or digital post-correction can be used.

FLOWGRAPH ANALYSIS

Especially in sigma-delta converters, the transfer functions easily get complicated. To aid symbolic analysis, the structure can be expressed as a group of equations that can be symbolically solved using Maple, for example.

Another usual trick is to avoid recursive functions, in which case the system can be expressed as a sum of state variables multiplied by FIR transfer functions. In the example below, the situation in the input of block H₁ can be expressed as

\[
\frac{W}{H_1 H_2} = U - b_1 V + \frac{b_2}{H_1} V
\]

which after substitution and multiplying by \(z^2\) gets into non-recursive form shown bottom left.
Example

Below is a Maple code for 2nd order SD modulator:

```maple
with(linalg):
H1 := 1/(z-1); H2 := 1/(z-1); # integrator functions
Y := matrix(
[ [   1,0, 0, -b1], # node 1
  [ -H1, 1, 0, -b2], # node 2
  [   0, -H2, 1, 0], # quantizer input
  [   0, 0, -K, 1] ] ); # output
Z := evalm(inverse(Y)); # solve system
sig := vector([a1,a2,a3,0]); # input branches
noi := vector([0,0,0,1]); # quant noise
Hsig := collect(simplify(evalm(Z &* sig)[4]),z);
Hnoi := collect(evalm(Z &* noi)[4],z);
```

LESLIE-SINGH ARCHITECTURE

One of the first multi-bit SDMs was the L-S architecture, where multibit ADC but still a single-bit feedback is used. Here, N-bit quantization is modelled as noise source un1, and its truncation to 1-bit feedback as another noise source un2.

Now, un2 has a transfer function $H_{n2} = \text{out}/\text{un2}$ to output, and it must be modelled by a similar transfer function $H_{n2}'(z)$ in the digital domain:

- ADC msb is used as feedback
- all bits except msb (i.e. the second quantization un2) is shaped by transfer function $H_{n2}'(z)$
- the original output out and the shaped 2nd quantization noise $H_{n2}*(\text{out}-\text{msb})$ are summed together

This cancels the effects of the 2nd quantization. The total noise transfer function is

$$N_{out} = \frac{(z-1)^2}{z^2 - z + 0.5} \cdot u_{n1} + \left( \frac{-z + 0.5}{z^2 - z + 0.5} \cdot H_{n2}'(z) \right) \cdot u_{n2}$$

(US pat 4987416, Jan1991)
Example.

Left is simulated results of a 2nd order L-S modulator with 6 bit A/D converter and 1-bit feedback. As seen, the ADC output is corrupted by high noise floor. The msb output already shows noise shaping, but after removing the error caused by truncating to 1 bit, the total output shows clear improvement. In this example, error is calculated as sign(out)-out and it is shaped with transfer function

\[
H_{n2} = \frac{a_2 K (b_2 z + (a_1 b_1 - b_2))}{z^2 - (a_2 K b_2 + 2) z + (a_2 K (b_2 - a_1 b_1) + 1)}
\]

where a1,a2 are the integrator and b1,b2 are feedback weights, was used to cancel the quantization error. This can be broken as two FIR filters, as shown on the next slide.

**LESLIE-SINGH CONT..**

Cauwenberghs98 used a slightly different partitioning for Leslie-Singh with lossy integrators. Now, the quantization noise un2 of the 1-bit feedback can be completely cancelled by combining the multibit and 1-bit output by two FIR filters:

\[
F_1 = \frac{z + (a_1 - p_1)}{a_1 z^2} = \left( a - \frac{p_1}{a_1} \right) z^{-2} + \frac{1}{a_1} z^{-1}
\]

\[
F_2 = \frac{(z - p_1)(z - p_2)}{a_1 a_2 z^2} = \frac{1 - (p_1 + p_2)z^{-1} + z^{-2}}{a_1 a_2}
\]

This results nicely in

\[
H_{\text{sig}} = H_{sa} \cdot F_2 + H_{sb} \cdot F_1 = z^{-2}
\]

\[
H_{\text{noi}} = H_{sa} \cdot F_2 + H_{nb} \cdot F_1 = 0
\]

Note: the transformation left below can be used to break a recursive equalizer function. Now also the latter 1/B function can be left away as plain B nicely equalizes the signal arriving in port 1.
In MASH ΣΔM modulators, low order modulators are cascaded to create a stable high-order modulator. In the figure left, plain quantization noise $n_1$ is extracted by subtraction $g_3(x+n_1)-g_2x$. Hence, only the quantization noise $n_1$ is forwarded to the 2nd stage where it is quantized and subtracted from the output of the 1st ADC (Actually, some linear term can also be added by varying the ratio of $g_2$ and $g_3$, and this results in a tradeoff between dynamic range, maximum SNR, and spurious signals generated in the 2nd ADC. In general, $g_2=-g_3$ gives largest dynamic range).

In converter 2 noise $n_1$ sees the signal transfer function $STF_2$. Now to cancel $n_1$ from output $o_1$, a corrector $H_d(z)$ needs to be used. Hence, only $STF_2$ is further differentiated $H_d$. This gives a differentiating $H_d$, which cancels $n_1$ completely and causes further noise-shaping to $n_2$. As a net result, the system behaves as a higher order modulator.

Example

Supposing 2nd order modulators similar to L-S example ($b_1=b_2=-1$ and $b_3=b_4=-1$), $k_1=k_2=1$, $g_3=1$ and $g_2=-1$, the transfer functions on the left can be drawn.

Here, the nominator of $O_2/S$ is roughly 0 and nominator of $O_2/N_1$ is roughly $k_2^2Den1$. Now

$$H_d = \frac{NTF_1}{STF_2} = \frac{2(z-1)^2}{Den1} \cdot \frac{Den2}{k_2} = 2(z-1)^2$$

which cancels $n_1$ in $OUT$ and leaves

$$\frac{OUT}{N_2} = H_d \cdot \frac{2(z-1)^2}{Den2} = \frac{4(z-1)^4}{Den2}$$

However, the cancellation of $N_1$ is sensitive to the values of $g_2,g_3,k_1$, and $k_2$, as well as lossy integrators in $H_1$ and $H_2$. To correct these errors, $H_d(z)$ needs to be an adaptable FIR filter, and some adaptation metrics are studied next.
ADAPTING THE COMBINER

To adapt the equalizer $H_d(z)$ in a MASH ADC, the following methods can be used:

- **Off-line calibration.** A pseudorandom (PN) 2-level in-band test signal can be used to adapt $H_d(z)$.
- **On-line PN signal can be added in to input and used for**
- **Blind adaptation.** Here, the total amount of out-of-band noise is minimised by adapting $H_d(z)$. This works if input signal is bandlimited so that any out-of-band deterministic components are below noise floor (Cauwenberghs).
- **Out-of-band pilot signal.** A pilot is summed to the $n_1$ quantization noise e.g. by modulating its threshold, and the residual (non-cancelled) pilot in the output is used as an error to adapt $H_d$.

Temes et al.: TCAS-II

ADAPTING MASH ...

Blind adaptation based on out-of-band noise sounds tempting, but it suffers from

- correlation between the quantization noise and the input signal.
- out-of-band blockers must be filtered, so the selectivity of the digital filters can not be fully employed.
- as signal itself is not measured, the gain can not be adapted.

Left is the implementation presented in Part II paper below. Analog errors are a modelled as a leakage path $H_{eq}(z)$, that is adapted so that a PN test signal $ts$ disappears from the output. Hence, $H_{eq}$ is a normal LMS filter which is adapted by correlating out with $ts$.

MULTIBIT ΣΔ ADC

Especially at low oversampling ratios, a multibit feedback in a ΣΔ ADC is often desired, as with it:

- Total SNR improves 6 dB/bit
- Quantizer gain is more stable, resulting in stable NTF and SNR
- As stability is better, more aggressive loop filters can be used

But:
- INL in the feedback DAC causes distortion and mixes shaped quantisation noise back down.
- Linearity of the DAC must match the total requirements.

There are three ways to correct this:
- employ DEM in the feedback DAC
- employ trimming in the feedback DAC (requires e.g. 12 bit linearity from a 3-bit DAC)
- employ digital correction after conversion

SPECTRAL EFFECTS OF DAC NONLINEARITY

Nonlinear feedback causes distortion in the output. Even though the distortion itself were not harmful, it spreads the quantisation noise spectra.

Even order INL (a) is worst, as it both widens the quantisation noise spectra and rectifies it to DC, thus filling the baseband and causing DC offset. It also creates even harmonics.

Odd order nonlinearity does not rectify, but the widening of quantisation noise spectra is enough to fill the passband with excess noise. It also creates odd harmonics, but not DC offset.
SHAPING OF NONLINEARITIES

The most usual way to tackle D/A nonlinearity is to shape the nonlinearity away from signal band using e.g. a DWA algorithm. In the enclosed code and the next figure, 1st order quantization error noise shaping is used to clean up the baseband spectrum. It is seen that high order ($L > 2$) modulators would also require a high order shaping for the errors: at high OSR, the performance is limited by inadequate shaping of the errors. Higher order shaping is possible but also more complex to implement.

Matlab code for the next plot:

```matlab
OSR = 10.^(0:0.05:2);

figure(900)
chs = ['rgbkcy'];
maxs = 120;
v = [1 100 0 maxs];
err = 0.01;

for L=1:6
tmp = sqrt(1.5*(2*L+1))*OSR.^(L+0.5)/(pi^L);
ch = chs(L);
for N=1:5
    M = 2^N;
snr = (2^N - 1)*tmp;
    SNR = 20*log10(snr);
    mask = SNR > maxs;
    SNR = SNR.*(1-mask)+maxs*mask;
    snrdem = sqrt(3*M*OSR.^3)/(pi*err*(1-1/M));
    SNRDEM = 6.02*log(snrdem)/log(2);
    subplot(3,2,L)
    semilogx(OSR,SNR,OSR,SNRDEM,:)
    hold on
end
title(['N=' int2str(L)])
hold off
axis(v)
grid
ylabel('SNR dB'), xlabel('OSR')
end
```

(c) 1999- Timo Rahkonen, University of Oulu, Oulu, Finland
LUT CORRECTION OF MULTIBIT SIGMA-DELTA ADCs

Instead of shaping the DAC errors they can also be completely cancelled. This can be done by storing the actual and measured weights of the feedback DAC in a look-up-table (LUT) and using either

- a trim DAC parallel to the main DAC. Here, the corrected linearity should match the required overall performance (12 bits in the figure).

or

- a digital corrector, where error propagation $H_n$ from DAC to output is modelled with a similar transfer function $H_n'$. Here, $H_n'$ is very important, if $H_n$ is narrowband, but in some cases it can be replaced by simple delaying. Temes et al. have used for $H_n'$ a digital replica of the SDM, which imitates $H_n$ and also noise-shapes quantization errors, in which case the output word length can be reduced.

CALIBRATING MULTIBIT DAC

The simplest form to calibrate the errors in the DAC is to configure (portions of) the modulator to a 1-bit $\Sigma\Delta$ ADC and use the feedback DAC as a signal source. Now the DC levels generated by the m-bit DAC are measured using heavy averaging and the errors compared to ideal levels are stored into the LUT.

After calibration, the test feedback is disconnected and the LUT is connected as shown in the previous slide.

For on-line calibration, a separate 1-bit SDM can be used to measure the feedback unit elements one at a time. To maintain full output range, one redundant element is needed, and for the correction, the sum of unit element errors must be built.
BACKGROUND CALIBRATION USING HISTOGRAMS

Previous adaptation requires a known test signal. As an alternative, a background calibration based on analyzing amplitude histograms is studied. The basic idea is:

- assume continuous and smooth amplitude distributions
- estimate (filter) the ideal histogram
- calculate the difference between the measured and estimated histogram
- recognise discontinuities caused by ADC nonlinearities

Left is shown the amplitude pdf for a 1-tone sinusoid. In a case of multitone test signal, the pdf approaches Gaussian pdf.

MSB-1 (see Elbornsson: Blind Estimation and Error Correction in a CMOS ADC. IEEE 2000)

(c) 2001- Timo Rahkonen, University of Oulu, Oulu, Finland
MISSING OR NARROW CODE

Positive \( h \) in INL function causes missing or narrower-than-usual codes. INL step size can be estimated by

\[
h = \frac{N_1 - N_2}{N_1}
\]

If \( N_2 \) goes to zero or faster collecting of the histogram is desired, several adjacent bins can be combined. If e.g. two bins on both sides of the step are collected, then

\[
h = 2 \cdot \frac{(N_1 + N_3) - (N_2 + N_4)}{(N_1 + N_3)}
\]

Example:
\( N_1 = N_3 = 100, N_2 = 0, \) and \( N_4 = 50 \) gives \( h = 1.5 \). The situation shown left suggests \( h = 0.85 \).

NON-MONOTONICITY OR WIDE CODE

Negative \( h \) in INL function causes missing or narrower-than-usual codes. INL step size can be estimated by

\[
h = \frac{N_2 - N_1}{N_2}
\]

where \( N_1 < 2N_2 \). If the expected non-monotonicity is larger than 1 lsb \( h < -1 \) or faster collecting of the histogram is desired, several adjacent bins can be combined. If e.g. two bins on both sides of the step are collected, then

\[
h = 2 \cdot \frac{(N_2 + N_4) - (N_1 + N_3)}{(N_2 + N_4)}
\]

Example:
\( N_1 = 200, N_3 = 150, N_2 = N_4 = 100 \) gives \( h = -1.5 \). The situation shown left suggests \( h = -0.7 \).
HISTOGRAM CORRECTION IN PIPELINE ADC

It is not necessary to measure the entire histogram. Instead hits in the closest bins around transitions of a stage output can be measured, and based on these number of hits, a correction coefficient can be calculated.

In pipeline converters, two succeeding stage change their state in the same bin. Thus the total error is a sum of errors in two succeeding stages, and e.g. the correction coefficient for MSB stage must be calculated from the errors measured both at MSB and MSB-1 transition points:

$$C_{\text{msb}} = h_{\text{msb}} + 2h_{\text{msb-1}}$$

ERRORS IN FLASH CONVERTERS

Structure
Resistor string, comparators, thermometer coding, lin-to-bin coding

Bubbles in thermometer coding
- Minimize clock skew
- Employ hi-priority coding to find only the highest set flip-flop. Usually not complete, but width=3
- Employ Gray-coded output to minimise the effect of bubbles

Metastability of comparators
During sampling, the comparator is connected to a flip-flop with positive feedback whose output is

$$v_o(t) = v_{i0} \cdot \exp(t/\tau)$$

Thus, for small enough $v_{i0}$ the output may not reach full logic levels during the sampling period and the state of the comparator is indefinite. This can be cured by

- Multiple synchronizer stages
- Force buffers to make always a decision

Time constants of the resistor string
Different signal amplitudes are delayed by different amount. This causes distortion.

In single-ended structure input feedthrough to reference string causes signal-dependent thresholds and hence mixing.

See papers by Steyaert
RC TIME CONSTANTS IN FLASH ADC

Left is a structure of a flash ADC: constant bias current sources and resistor strings are used to create level-shifted versions of differential input signals, and cross-connected comparators are used to sense the amplitude level.

Now parasitic capacitive loading of the resistor string causes displacement currents that cause hysteretic errors in the comparator levels. This causes odd order harmonics.

This phenomena has been analysed using a distributed RC model for the resistor string in Boni, Morandi, Harmonic Distortion in High-Speed Differential A/D Converters, IEEE TCAS-II March 98

BOOTSTRAPPING THE R-STRING

One way to minimise the capacitive loading of the resistor string is to bootstrap the capacitance: a poly1 shield under the poly2 resistor follows the voltage variations in the resistor string. Thus, no displacement currents are flowing in the capacitors and the R-string any more, and distortion may be reduced by 10 - 15 dB.
CORRECTION OF INTERLEAVED AD CONVERTERS

As noted before, both offset, gain and timing mismatches in interleaved A/D converters cause spurious signals. Blind correction methods for these have been studied e.g. by Elbornsson, Eklund, and Gustafsson.

Offset cancellation
This is easily done by collecting DC average values of each channel, calculating the differences of these, and substracting these from output results. A PN chopper can be used to randomize the input and speed up the averaging.

Time skew cancellation
Provided that input bandwidth is limited to fs/6, timing errors can be estimated by collecting the squared differences of adjacent samples. Wideband noise affects the estimated results and it too must be estimated somehow.