Blind error estimation and correction in an AD converter

Jonas Elbornsson

Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
WWW: http://www.control.isy.liu.se
Email: jonas@isy.liu.se

November 2, 1999

Report no.: LiTH-ISY-R-2206
Submitted to ICASSP2000

Technical reports from the Automatic Control group in Linköping are available by anonymous ftp at the address ftp.control.isy.liu.se. This report is contained in the pdf file 2206.pdf.
Blind error estimation and correction in an AD converter
Jonas Elbornsson
November 2, 1999

Abstract
For small integrated CMOS digital chips that communicate via analog signals the size and the cost of the AD-converters is a problem. An integrated CMOS AD-converter that could be built into the chip would solve this problem. In a CMOS process, the manufacturers only guarantee a very low accuracy of the resistances. The components of the ADC is therefore very inaccurate. The purpose of this report is to present a method for identification of the errors in the ADC and to perform simulations that evaluate the method. The algorithm should work while the ADC is used and it does not assume any knowledge of the input signal except that the distribution function is smooth. The estimated values can then be used in the chip to compensate for the errors in the ADC.

1 Introduction
In an analog-to-digital converter (ADC) the requirement on good accuracy in the resistors\(^1\) are normally very high since the accuracy on the digital signal depends on the resistors. When an ADC is implemented on a CMOS chip, the parasite resistances of the wires are used as resistors. The resistances in these wires have very low accuracy, it might vary up to 50 percent from the nominal value on different chips and up to 1 percent on one chip. But an ADC can be made very small and cheap in CMOS, therefore it is interesting to use this technique. To avoid the increasing errors in the digital signal we have to estimate the actual values of the resistances, or equivalently, the actual levels in the ADC. In this report a new method for estimation of the parameters is presented. This method assumes no prior knowledge of the distribution of the input signal, other than smoothness. This method also works online and adapts to changes in the resistances caused by temperature changes for example, that earlier methods have not taken care of [5].

1.1 The ADC
The AD conversion is done by comparing the unknown analog signal to known levels in a resistance ladder [2]. To get an accuracy of \(n\) bits in the digital

\(^1\)The resistors are used for generating analog reference levels. In many cases, capacitors are used instead, however, the matching problem and the analysis in this report is exactly the same in that case.
signal $2^n$ resistors is required in the ADC, figure 1. This type of ADC is called SA-ADC (successive-approximation). The digital value is found by the binary search algorithm: First the analog signal is compared to the reference level in the middle of the resistance ladder. If the analog signal is smaller than that reference level it is compared to the reference level in the middle of the lower half of the ladder, and if it is larger than the reference level in the middle it is compared to the reference level in the middle of the upper half of the ladder etc. until the correct value is found. The correct value will then be the closest reference level smaller than the analog value. Such an n-bit ADC will give a digital value with n correct bits as long as the reference levels are correct. If this is the case the transfer function of the ADC, from the analog input to the digital output, will look like the plot to the left in figure 2. If a rectangularly distributed signal is used as input to this ideal ADC, a histogram of the output will look like the plot to the right in figure 2. Here the histogram is flat as it should be with a rectangularly distributed input.

If there are errors in the reference levels the transfer function will be incorrect as in the plot to the left in 3. Now the histogram will not be flat, as seen in the plot to the right in figure 3, since the length of the intervals between the levels are different.

To correct the histogram a decoder that maps the digital levels onto the true levels in the ADC is used, see figure 4. The histogram will now again become flat, but with different lengths of the intervals, see figure 5.
Figure 2: To the left: The transfer function of an ideal ADC. To the right: A histogram of the output when the input is rectangularly distributed

Figure 3: To the left: The transfer function of an ADC with errors. To the right: A histogram of the output when the input is rectangularly distributed

Figure 4: The transfer function of the decoder
2 Method

2.1 Mapping from analog input to digital output

In this report we assume that the ADC spans the signal interval from zero to one, any other range is just a rescaling. The analog input is $u_t$ and the digital output is $y_t$. The mapping from analog input to digital output in an ideal ADC is described by

$$y_t = \left\lfloor \frac{2^n u_t}{2^n} \right\rfloor$$

where $\lfloor \cdot \rfloor$ denotes the integer part. In a real ADC there are errors in $2^n - 1$ of the levels (the first level is ground). We use a parameter vector $\Theta = [\theta_1 \theta_2 \ldots \theta_{2^n-1}]^T$ to describe the errors in the levels. Each $\theta_i$ is the relative error in the distance between level $i - 1$ and level $i$, so the actual reference levels are: 0, $\frac{1+\theta_1}{2^n}$, ..., $\frac{k+\sum_{i=1}^{k} \theta_i}{2^n}$, ..., $\frac{2^n-1+\sum_{i=1}^{2^n-1} \theta_i}{2^n}$. The analog signal is then mapped onto the nearest lower level.

2.2 Parameter estimation algorithm

The estimation algorithm should work online and adapt to possible changes of the parameters. The algorithm therefore has to work with the actual signals that are converted in the ADC. The analog input signal is not measurable (if it was we would not need the ADC). The only assumption that is made about the input signal is that the distribution is smooth. The parameter estimation is based on statistical properties of the signal. A histogram is calculated from the measured digital values. To estimate the parameters the measured histogram should be compared to a theoretical histogram parameterized with respect to the parameter vector $\Theta$ [3]. Since the true distribution of the signal is not known we can not calculate the theoretical histogram. Instead we have to estimate it from the measured data. The estimated histogram will be used as the nominal histogram for $\Theta = 0$. The estimation of the true histogram is based on the assumption that the distribution function of the input signal is smooth and that the errors in the levels are random. This means that most of the energy of the true histogram (not the energy of the signal that generated the histogram)
lies in a quite narrow low frequency band, while the energy due to the errors in the measured histogram is spread out over the whole frequency band. In figure 6 the DFT’s (Discrete Fourier Transform) of an ideal and a measured histogram from a Gaussian distributed signal are shown. We see that the two histograms coincide for low frequencies while for higher frequencies there is a large difference due to high frequency errors.

This means that the nominal histogram can be estimated by low pass filtering the measured histogram. The measured histogram is filtered both in the forward and backward direction. This gives a zero-phase filter and by that phase drift is avoided. In the frequency domain the zero-phase filter is given by

$$H_0(e^{j\omega}) = \|G(e^{j\omega})\|^2 H_m(e^{j\omega})$$

where $G(z)$ is the transfer function for the filter. Here we see that the stop band rejection is doubled, measured in dB, compared to just forward filtering. The cut-off frequency in the filter must be chosen for each kind of signal since the frequency content of the true histogram depends on the signal. This choice can be made by studying the DFT of the measured histogram where it is usually easy to find the bandwidth of the true histogram. To keep the pass band as flat as possible the butterworth low pass filter has been chosen. The order of the filter was chosen to two. When the parameters are estimated we have to be able to compare the measured histogram with a theoretical histogram for any value of $\Theta$. This is calculated by interpolation between adjacent histogram bars

$$h_0(k, \Theta) = h_1(k, 0) + \frac{h_0(k + 1, 0) + h_0(k, 0)}{2} \sum_{i=1}^{k} \theta_i$$

$$- \frac{h_0(k, 0) + h_0(k - 1, 0)}{2} \sum_{i=1}^{k-1} \theta_i$$

(3)
where \( h_0(j, 0) \) is the low pass filtered measured histogram. To find an estimate of the parameter vector \( \Theta \) the criterion function

\[
V(\Theta) = \sum_{k=1}^{2^n} (h_m(k) - h_0(k, \Theta))^2
\]

should be minimized. The search for minimum can be done in several different ways such as Newton’s method or Gauss-Newton, see for example [4] or [1]. These methods give fast convergence in number of iterations but are numerically sensitive and computationally demanding for large systems since both the first and second derivatives have to be calculated. For hardware implementation it is therefore better to use a steepest descent method, where only the first derivative is calculated. In the steepest descent method the parameters are updated in the negative gradient direction at each iteration

\[
\hat{\Theta}_{t+1} = \hat{\Theta}_t - \eta \nabla V(\Theta)
\]

where \( \eta \) is a constant step length. The gradient can not be calculated analytically since the histogram is not analytically available. Instead a finite difference approximation has to be used. Both the measured and the estimated histograms are updated with new data at certain time intervals. To make the algorithm adaptable to changes in the parameters an exponential forgetting of the old histograms is introduced.

\[
\begin{align*}
(h_m)_{j+1} &= \lambda (h_m)_j + (1 - \lambda) h_{m\text{new}}, \quad \lambda \in (0, 1) \\
(h_0)_{j+1} &= \mu (h_0)_j + (1 - \mu) h_{0\text{new}}, \quad \mu \in (0, 1)
\end{align*}
\]

This can be summarized into the algorithm:

**Algorithm 1**

**Initialization**

- Choose a cut-off frequency, \( f_{co} \) for the low pass filter.
- Choose the number of data, \( N \), for each histogram update.
- Choose the number of iterations, \( M \), between histogram updates.
- Choose the forgetting factors, \( \lambda \) and \( \mu \), for the measured and estimated histogram.

**Data collection and histogram update**

- Collect a batch of data of length \( N \) from the ADC and calculate a histogram, the histogram is normalized so that \( \sum_{i=1}^{2^n} h_{m\text{new}}(i) = 1 \).
- Calculate an estimate of the theoretical histogram, \( h_{0\text{new}} \), by low pass filtering the measured histogram.
- Update the the old histograms, \( h_0 \) and \( h_m \), with new data:
  - \( h_0 = \mu h_0 + (1 - \mu) h_{0\text{new}} \)
\[- h_m = \lambda h_m + (1 - \lambda)h_m^{\text{new}}\]

**Iteration**

The following steps are done \( M \) times during the time new data are collected.

1. Calculate an estimate of the histogram for the parameters \( \hat{\Theta} \) by interpolation between adjacent histogram bars.
   
   \[ h_0(k; \hat{\Theta}) = h_l(k, 0) + \frac{h_0(k+1,0) + h_0(k,0)}{2} \sum_{i=1}^{k} \hat{\Theta}_i - \frac{h_0(k,0) + h_0(k-1,0)}{2} \sum_{i=1}^{k-1} \hat{\Theta}_i \]

2. Calculate the criterion function
   
   \[ V(\hat{\Theta}) = \sum_{i=1}^{2^n} (h_m(i) - h_0(i, \hat{\Theta}))^2 \]

3. Calculate a gradient approximation with finite differences
   
   \[ (\nabla V(\hat{\Theta}))_i = \frac{V(\hat{\Theta} + \varepsilon e_i) - V(\hat{\Theta})}{\varepsilon} \]

4. Update the parameter estimate
   
   \[ \hat{\Theta}_{j+1} = \hat{\Theta}_j - \eta \nabla V(\hat{\Theta}_j) \]

Figure 7 shows a block diagram of the AD converter with estimation of the histogram, estimation of the parameters and correction of the output.

![Block diagram of the whole correction model](image)

3 Simulations

All simulations in this report have been done for an 8-bit AD converter with \( 2^8 = 256 \) levels. The algorithm has been evaluated for two types of signals, a Gaussian signal and a sinusoid signal. We will first present the result for the Gaussian signal and in the end of this section the results for the sinusoid signal is presented.
### 3.1 Evaluation of the histogram estimate

#### 3.1.1 Gaussian input

The histogram estimation is here evaluated for a Gaussian input signal. The Gaussian distribution is chosen so that the mean of the signal lies in the middle of the histogram. The standard deviation is chosen so that the interval $[-5\sigma, 5\sigma]$ falls inside the range of the ADC. In figure 8 the theoretical gaussian histogram is shown. To evaluate the histogram estimation we make a plot of the relative error between the estimated histogram and the theoretical histogram based on the known prior distribution. The initial relative error in each level is uniformly distributed in the interval $[-0.1, 0.1]$, i.e. the error in the distance between two consecutive levels is at most 10 percent. In figure 9 the relative error for the estimated Gaussian histogram is shown. The measured histogram is generated from $10^8$ samples. From this plot we can see that the histogram estimation works well for the middle levels, but not so well near the edges. The bad performance near the edges of the ADC is caused by low excitation.

![Theoretical Gaussian Histogram](image1)

**Figure 8:** The theoretical gaussian histogram

![Relative Error for Gaussian Histogram](image2)

**Figure 9:** Relative error for the estimation of the Gaussian histogram.

#### 3.1.2 Sinusoid input

The histogram estimation is here evaluated for a sinusoid input signal. The amplitude is chosen so that the signal fits into the range of the ADC. In fig-
ure 10 the theoretical sinusoid histogram is shown. In figure 11 a plot of the
relative error between true and estimated sinusoid histogram is shown. As for

Figure 10: The theoretical gaussian histogram

the Gaussian histogram the estimation works well near the middle of the ADC. The reason for the bad performance near the edges here is, however, caused by the sharp knee in the true histogram 10 that is destroyed by the low pass filter.

3.2 Performance for Gaussian input

For evaluation of the performance of the parameter estimation the same Gaussian signal as described in section 3.1 have been used. The parameter estimation has been done as described in section 2 with $10^8$ samples of data and $10^4$ iterations and no forgetting.

3.2.1 Histogram for compensated ADC

One interesting performance measure is how close the measured histogram compensated by the estimated parameters comes to the theoretical histogram. The compensated histogram is normalized with the width of each histogram bar so that it is comparable with the true histogram. The histograms for the Gaussian input is shown in figure 12. In these plots we can see that the estimation works
Figure 12: Histograms for gaussian input.
quite well. To get a better view of how large the errors are the relative error between the true and the compensated histogram is shown in figure 13. Here we can see that the estimation works well in the middle of the histogram. If we compare this plot with the plot in figure 9 we can see that they look quite similar. That shows that the parameter estimation works well and to improve the estimation, the main effort should be on the histogram estimation.

3.2.2 High frequency error rejection

In a plot showing the DFT of the different histograms we can see how much the high frequency components in the measured histogram caused by the level errors are rejected by the parameter estimation. Figure 14 shows the DFT of four histograms: the theoretical histogram, the measured histogram, the estimated reference histogram, the estimated histogram compensated with the estimated parameters. This plot show that the estimation works well for the Gaussian histogram. The true Gaussian histogram has almost all of its energy
in a narrow low pass band while the energy in the measured histogram is spread over the whole frequency band. Therefore a low pass filter rejects almost all errors.

3.3 Performance for Sinusoid input

The sinusoid signal that has been used for estimation in this section is the same signal as described in section 3.1.2. $10^8$ samples of the signal have been used for the histogram generation.

3.3.1 Histogram for compensated ADC

In this section the histograms for sinusoid input signal are evaluated. Figure 15 shows the theoretical histogram, the measured histogram, the estimated true histogram and the compensated histogram. In these plots we can see that

![Figure 15: Histograms for sinusoid input.](image)

the performance is worse than for the Gaussian case, section 3.2.1. The main reason for the worse performance is the low pass filter that destroys some of the information in the sinusoid histogram. In figure 16 where the relative error between true and compensated histogram is plotted we can see that the errors are larger than in the Gaussian histogram.
3.3.2 High frequency error rejection

Figure 17 shows the DFT of the four histograms: the theoretical histogram, the measured histogram, the estimated reference histogram, the estimated histogram compensated with the estimated parameters. Here we see that the ideal histogram contains quite much high frequency components for sinusoid input. That explains why the histogram estimation does not work so well in the sinusoid case.

3.4 Evaluation of the parameter estimates

3.4.1 Distortion measures

In this section we will evaluate two different measures of the distortion caused by the errors in the ADC [6], the SNDR (signal to noise plus distortion ratio) 8
and SFDR 9 (spurious free dynamic range).

\[ SNDR = 10 \log \frac{E(signal)}{E(noise + distortion)} dB \] (8)

\[ SFDR = 10 \log \frac{E(signal)}{E(largest \ distortion \ component)} dB \] (9)

where \( E(\cdot) \) is the energy. The second column in table 1 shows the SNDR values for the output signal corrected with different estimates of \( \Theta \). The SNDR for the output without correction is also included for comparison. The SNDR is measured with a sinusoid input signal. These values show that for a Gaussian

<table>
<thead>
<tr>
<th>( \Theta ) estimate</th>
<th>SNDR</th>
<th>SFDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>no correction</td>
<td>40.7 dB</td>
<td>51 dB</td>
</tr>
<tr>
<td>correction with true ( \Theta )</td>
<td>48.5 dB</td>
<td>72 dB</td>
</tr>
<tr>
<td>correction with ( \Theta ) estimated from estimated Gaussian histogram</td>
<td>48.3 dB</td>
<td>61 dB</td>
</tr>
<tr>
<td>correction with ( \Theta ) estimated from known Gaussian histogram</td>
<td>48.7 dB</td>
<td>65 dB</td>
</tr>
<tr>
<td>correction with ( \Theta ) estimated from estimated sinusoid histogram</td>
<td>42.1 dB</td>
<td>55 dB</td>
</tr>
<tr>
<td>correction with ( \Theta ) estimated from known sinusoid histogram</td>
<td>45.9 dB</td>
<td>55 dB</td>
</tr>
</tbody>
</table>

Table 1: SNDR and SFDR with sinusoid input for different estimates of \( \Theta \)

input signal we reach optimal performance measured in SNDR even if we do not know the distribution. For a sinusoid input, however, the performance is much better if we assume that the input signal distribution is known. The reason why we do not reach the theoretical SNDR limit of 49.9\(^{\text{dB}}\) [6] in any of these cases is the actual errors in the ADC. Even if we compensate for these errors the performance will not be as good as in an ADC with equal distance between the levels.

The SNDR gives a measure of the mean performance. If there are just a few distortion frequencies the SNDR can still be quite good. Therefore we will also look at the SFDR, that gives a worst case performance measure. The third column in table 1 shows the SFDR values for different estimates of \( \Theta \). Figure 18 shows the spectrum for the output signal without correction and corrected with estimated \( \Theta \) from estimated histogram.
3.4.2 RMS error

The RMS (root mean square) error measures the mean error of the levels in the ADC:

$$RMS = \sqrt{\frac{1}{2^n - 1} \sum_{i=1}^{2^n-1} \left( \sum_{j=1}^{i} (\theta_0 - \hat{\theta})^2 \right)}$$

The RMS values for the four different estimates described in section 3.4.1 are shown in table 2. From this table we can see that the RMS error is lower for all estimates of $\Theta$ but not always very much better. The reason for this is that the estimates near the edges of the ADC are quite bad and therefore they give a large contribution to the rms, the estimates in the middle of the ADC is much better.

<table>
<thead>
<tr>
<th>$\Theta$ estimate</th>
<th>RMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>no correction</td>
<td>1.01</td>
</tr>
<tr>
<td>correction with $\Theta$ estimated from estimated Gaussian histogram</td>
<td>0.12</td>
</tr>
<tr>
<td>correction with $\Theta$ estimated from known Gaussian histogram</td>
<td>0.10</td>
</tr>
<tr>
<td>correction with $\Theta$ estimated from estimated sinusoid histogram</td>
<td>0.87</td>
</tr>
<tr>
<td>correction with $\Theta$ estimated from known sinusoid histogram</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 2: RMS values with sinusoid input for the different estimates of $\Theta$. 

Figure 18: To the left: spectrum for output without correction. To the right: spectrum for corrected output.
4 Conclusions

We have presented a method for estimation and correction of the errors in an AD converter. The method does not assume any prior knowledge of the signal except that the distribution should be smooth. The method have been evaluated for Gaussian and sinusoid input signal and the estimation works well especially for Gaussian input. The reason why it does not work so well for a sinusoid input is that the distribution contains a quite sharp knee that is removed in the estimation. The future work will concentrate on improving the estimation of the distribution function of the signal to make it possible to use a wider class of input signals for estimation.

References


