Journal of Sensor Technology, **2012, 2, 177-184**
doi:10.4236/jst.2012.24025 Published Online December 2012 (http://www.SciRP.org/journal/jst)

Scientific Research

Lookup Table Optimization for Sensor Linearization in Small Embedded Systems

Lars E. Bengtsson
Physics Department, University of Gothenburg, Gothenburg, Sweden
Email: lars.bengtsson@physics.gu.se

Received October 5, 2012; revised November 6, 2012; accepted December 6, 2012

ABSTRACT

This paper treats the problem of designing an optimal size for a lookup table used for sensor linearization. In small embedded systems the lookup table must be reduced to a minimum in order to reduce the memory footprint and intermediate table values are estimated by linear interpolation. Since interpolation introduces an estimation uncertainty that increases with the sparseness of the lookup table there is a trade-off between lookup table size and estimation precision. This work will present a theory for finding the minimum allowed size of a lookup table that does not affect the overall precision, *i.e*. the overall precision is determined by the lookup table entries' precision, not by the interpolation error.

Keywords: Lookup Table; Sensor Linearization; Embedded Systems; Interpolation

1. Introduction

Look-up tables (LUTs) are used in a wide variety of computer and embedded applications; NASA use it to improve the pointing precision of antennas [1], CERN uses LUTs to calibrate the beam energy acquisition system of the Large Hadron Collider (LHC) [2] and it is one of the most common methods for digital synthesis of arbitrary waveforms [3]. It is used extensively in numeric calculations, for example in division algorithms [4], square root algorithms [5] and even for fast evaluation of general functions [6,7]. In Data Acquisition Systems (DAQ) it is used to correct non-linearities and offset errors in Analog-to-Digital Converters (ADCs) [8-10] or to design non-uniform ADCs [11]. This work will focus mainly on LUT applications in Embedded Measurement Systems (EMS); linearizing sensor signal outputs is one of the major applications of LUTs [12-18].

A sensor converts the physical unit (the *measurand*) into some electrical unit (preferably volt) and the embedded measurement system converts the sensor output into a digital value (typically an integer) [19]. The main errors in most measurement systems are related to the transducer's offset, gain and non-linearities [20], and for that reason the process of sensor linearization is a crucial step in the design of an embedded measurement system [21]. The linearization process must compensate for the non-linear relationship between the sensor's input and the output signals [21], see **Figure 1**.

If the relationship between the sensor's input and its (digitalized) output y, is given by the non-linear function f, *i.e.* $y = f(x)$, then the relationship between the linearizing block's input y and its output z should be f^{-1} [22], *i.e.* $z = f^{-1}(y)$, so that

$$z = f^{-1}(y) = f^{-1}(f(x)) = x \tag{1}$$

Many DAQs and EMSs use floating-point or fixed-point controllers that can handle real-numbers without any significant overhead penalty [23]. However, in smaller, 8-bit integer systems, the main concern is not necessarily to reproduce the input signal exactly; they are dedicated to linearizing the signal and any parameter estimations are performed off-line in the host computer. In these cases, expression (1) changes to

$$z = k \times x \tag{2}$$

where k is in general a real constant transferring the integer x into to a real number estimate of the measurand. Consequently, these systems work with integers only and apart from generating faster computations, they also typically reduce the memory footprint of the LUT since integers are stored in bytes or words (8 or 16 bits) while real-numbers are stored as "floats" or "doubles" (24 or 32 bits). Even if we would use a larger, 32-bit system with a real-number computational engine, and perhaps also lots of non-volatile memory for LUT storage, it is still important to keep the LUT size to a minimum since even the most advanced 32-bit systems have a limited size of cache memory; LUT sizes should be small enough to fit into level 1 cache since a memory fetch from level 1 cache render a maximum penalty of two clock cycles while a fetch from level 2 cache may need

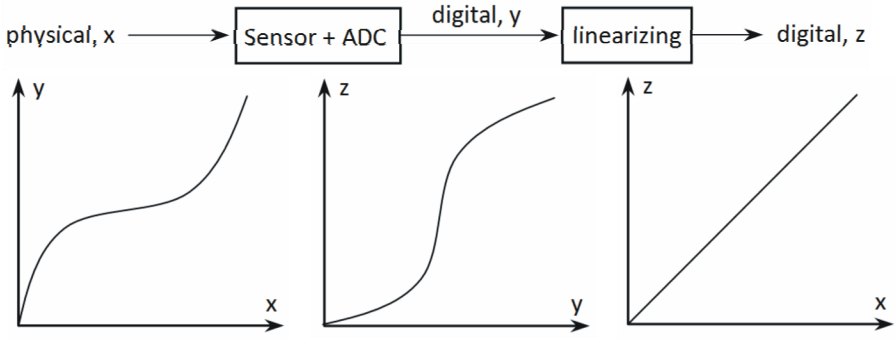

Figure 1. The linearization process.

as much as 15 clock cycles [24].

Several other linearization methods have been suggested in literature. The most common linearization methods can be classified as follows [21,25,26]: 1) Analog hardware-based; 2) Software-based; 3) Analog hardware-software mixed approach.

Analog hardware-only solutions are frequently used [27,28] but the disadvantage is that the extra analog components necessary increase both cost and power consumption and also, due to inherent variations in the manufacturing process parameters, each sensor is likely to need individual trimming and tailored compensation [29, 30] and that may be complicated (and expensive) in analog hardware-only solutions. Some sensors are also sensitive to secondary parameters (typically temperature) [31-34] and this may be very hard to compensate for in hardware-only solutions. (The dependence on secondary parameters is sometimes referred to as *cross-sensitivity* [22]). However, for non-digital measurement systems these solutions are important and could also be considered for digital measurement systems with limited memory and/or limited computing power [27,28,34,35]. This work is concerned with software-based solutions only, or rather, *firmware*-based solutions, since it focuses on linearizing by using embedded controllers.

The rest of this work is organized as follows; Section 2 presents some basic theory concerning linearzing with LUTs and interpolation. Section 3 presents the hypothesis of which this work is based upon and Section 4 describes the methods used to verify the hypothesis. Section 5 presents some results and they are discussed in Section 6. The work is summarized in Section 7 with some conclusions.

2. Theory

2.1. Linearization

The process of designing the linearization block in **Figure 1** is typically a multi-step process. First of all the non-linear function *f*, relating the sensor input and output, is in general not known and needs to be determined. Typically, this is done by a calibration process where a number of *x*,*y*-pairs are registered experimentally and *f* is determined by off-line curve fitting (using the MATLAB commands *polyfit()* or *nlinfit()*). This is illustrated in **Figure 2**.

Once we have found *f*, we can solve for the inverse function f^{-1} that we need to implement into the linearizing block in figure 1. There are basically two different ways to implement f^{-1}; if your embedded measurement system has access to 32-bit floating-point processing (with hardware multiplication), then $z = f^{-1}(y)$ can be calculated in real-time. In small embedded 8-bit systems with limited computational power, f^{-1} is typically implemented as a LUT in flash memory. In this case we assume that *y* is the *n*-bit integer produced by the ADC and this integer is simply used as a pointer to the memory location where $f^{-1}(y)$ is stored, see **Figure 3**. (Notice that **Figure 3** indicates that the resolution of the *z*-output (= *m*) in general differs from the resolution of the *y*-input (= *n*).)

The disadvantage of the first method, when $f^{-1}(y)$ is calculated in real-time, is that it requires a complex (and expensive) floating-point or fixed-point processor. The advantage is that it does not require much program memory; only the function parameters for the f^{-1} function needs to be stored (= $p + 1$ parameters for a polynomial of order *p*). The advantage of the LUT method is that it can be implemented even in the simplest controller but the disadvantage is that it occupies a lot of program memory. So, the choice between the two methods is a trade-off between the need for signal processing power and memory space occupancy. Since memory space is typically less expensive than a floating-point processing engine, a LUT is the dominating linearity method. However, a combination of the use of a (sparse) LUT and some non-complex integer signal processing may reduce the demand for LUT space and still meet the real-time deadlines. The "non-complex integer signal processing" is typically limited to linear interpolation in order to retrieve the "missing" LUT elements. This work is concerned with the details of this process and the question

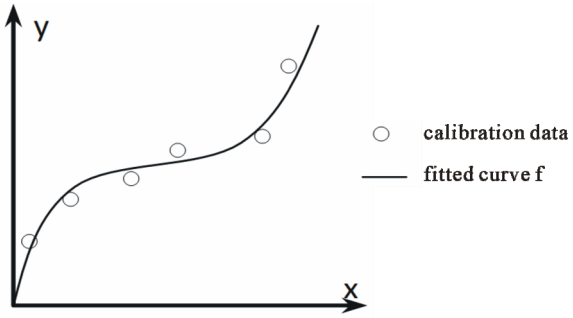

Figure 2. Finding *f* by curve fitting.

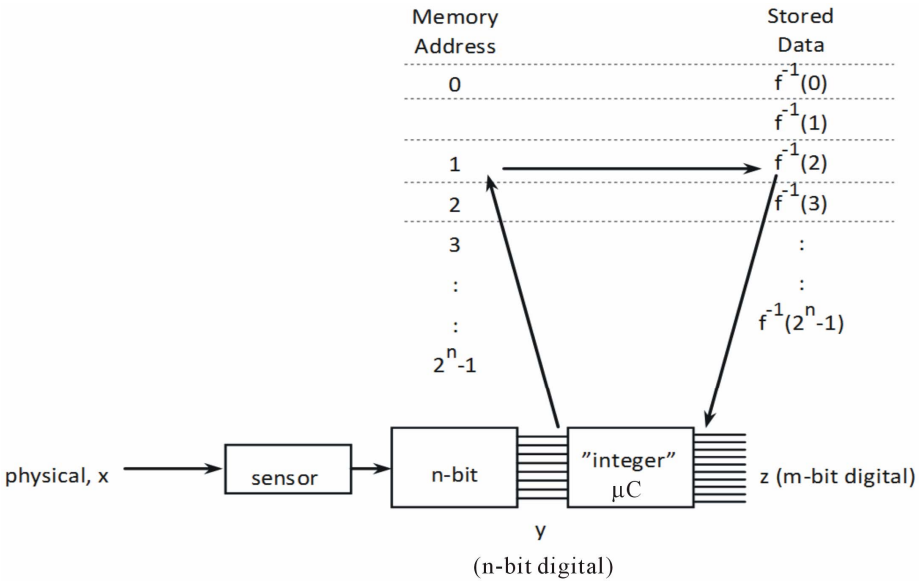

Figure 3. Linearizing by LUT.

of the necessary precision of the LUT elements and/or the maximal sparseness of the LUT entries.

2.2 Interpolation

In the following we will assume that we need to linearize a sensor signal using a small embedded system, *i.e.* the non-linear sensor signal is digitalized by an *n*-bit ADC and we are looking for a firmware algorithm that linearizes the signal according to expression (2). The fact that we use a "small" (and inexpensive) system, indicates that memory is scarce and that the processing power is limited; all signal processing will be on integers.

If the resolution of the ADC is *n* bits, a complete LUT would occupy 2^n memory locations. For example, a 12-bit ADC would need 4 kbyte of program memory if we settle for "byte" resolution and twice as much if we want "word" resolution. This is by no means an insignificant amount of program memory for a general purpose microcontroller. For example, the PIC18F458 RISC controller from Microchip has a flash memory of 32 kbyte [36].

In order to save some program memory, a sparser LUT may be implemented and then we can use interpolation to retrieve intermediate values [20]. Considering our assumed limited computation capability, *linear* interpolation is the obvious choice, see **Figure 4**.

Since we approximate each interval between LUT entries with a different straight line, the method is referred to as *Piece-wise Linear Interpolation* (PwLI) [37] and the combination of a sparse LUT and PwLI is referred to as *polygon interpolation* [32,33].

Assume that *y* is the integer output from an *n*-bit ADC and that we use f^{-1} to map each y_i to a (linearized) z_i; this would require 2^n LUT entries. Since this would occupy too much memory space we need to *decimate* the LUT table and the decimation factor should be an even multiple of 2 (in order to simplify integer division later). If the decimation factor is 2^{n_p} ($n_p < n$) the number of LUT entries is reduced to 2^{n-n_p}, and that leaves us an n_p –bit number that we can use for interpolation.

By decimating the LUT we save precious flash/cache memory. However, the downside is that we also reduce the precision of the LUT estimation. Consider **Figure 5**.

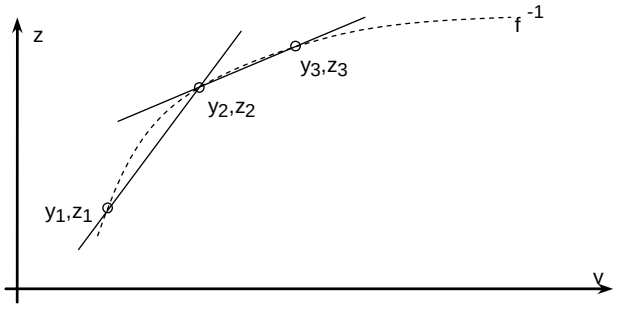

Figure 4. Piece-wise Linear interpolation.

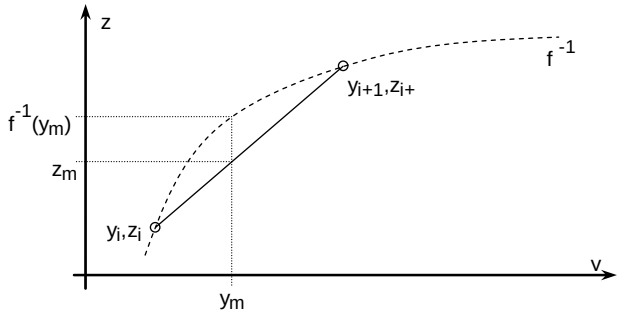

Figure 5. Linear interpolation.

Implementing linear interpolation, we approximate $f^{-1}(y_m)$ with z_m:

$$f^{-1}(y_m) \approx z_m = z_i + \frac{z_{i+1} - z_i}{y_{i+1} - y_i} \times (y_m - y_i) \tag{3}$$

y_m represents the measured (digital) value, and z_m is the point estimator of our measurement. From **Figure 5** we can see that the error of this estimation (the *remainder* term) is

$$R(y_m) = f^{-1}(y_m) - z_m \tag{4}$$

In general, the remainder term of a first order approximation is given by [38]

$$R(y_m) = \frac{(f^{-1})''}{2!}(y_m - y_i) \times (y_m - y_{i+1}) \tag{5}$$

3. Hypothesis

There will basically be two contributions to the uncertainty of the final estimation of x: the interpolation error represented by expression (5) and the precision t_e of the LUT entries themselves (truncation or rounding errors). The total uncertainty of the estimation will be proportional to the square root of the sum of the squares:

$$\text{Uncertainty} \propto \sqrt{R(y_m)^2 + t_e^2} \tag{6}$$

We will assume that we work with integers only and that LUT entries are stored either as 8-bit bytes or as 16-bit words. That means that $|t_e|$ is

$$2^{-8} = 3.906 \times 10^{-3} \tag{7}$$

$$2^{-16} = 1.52 \times 10^{-5} \tag{8}$$

for the 8- and 16-bit cases, respectively.

Our goal here is to reduce the LUT size to a minimum (by decimation) without reducing the overall accuracy of the estimation. According to expression (6) that means that we have to monitor the size of the remainder term and increase the decimation factor for as long as $|R(y_m)| \ll |t_e|$ [39].

Hence, we need to know the maximal value of the remainder term in (5). Consider expressions (4) and (5). According to expression (4) the remainder term is the difference between the linear interpolation z_m and "true" function value $f^{-1}(y_m)$. If we assume f^{-1} to be a smooth continuous function, the maximum error must occur where the function's curviness is maximum and this agrees with expression (5); the second derivative of f^{-1} represents the curviness. Hence, we need to find the maximal value of $(y_m - y_i) \times (y_m - y_{i+1})$ in the segment of maximal curviness.

First of all we approximate the second derivative:

$$\left(f^{-1}(y)\right)'' = \frac{\mathrm{d}^2 f^{-1}}{\mathrm{d}y^2} \approx \frac{\Delta^2 f^{-1}}{(\Delta y)^2} \tag{9}$$

We find the maximum of the second part in (5) by setting the first derivative equal to zero:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}y_m}(y_m - y_i)(y_m - y_{i+1}) \\ &= \frac{\mathrm{d}}{\mathrm{d}y_m}\left(y_m^2 - y_m(y_i + y_{i+1}) + y_i y_{i+1}\right) \\ &= 2y_m - (y_i + y_{i+1}) = 0 \\ &\Rightarrow \quad y_m = \frac{1}{2}(y_i + y_{i+1}) \end{aligned} \tag{10}$$

Hence, the maximum value of $(y_m - y_i) \times (y_m - y_{i+1})$ is

$$\begin{aligned} &\left|\left(\frac{1}{2}y_i + \frac{1}{2}y_{i+1} - y_i\right) \times \left(\frac{1}{2}y_i + \frac{1}{2}y_{i+1} - y_{i+1}\right)\right| \\ &= \frac{1}{2}\left|(y_{i+1} - y_i) \times \frac{1}{2}(y_i - y_{i+1})\right| \\ &= \frac{1}{4}(y_{i+1} - y_i)^2 = \frac{1}{4}(\Delta y)^2 \end{aligned} \tag{11}$$

Inserting (11) and (9) into (5) gives us

$$|R(y_m)|_{\max} \approx \frac{1}{8}|\Delta^2 f^{-1}| \tag{12}$$

Expression (12) should be calculated in the segment of the function f^{-1} that has the greatest curviness.

We would also like to be able to calculate the remain-

der term from data. Suppose we have implemented the LUT in **Table 1**.

We can see that

$$\begin{aligned}\Delta^2 f_i^{-1} &= \Delta f_i^{-1} - \Delta f_{i-1}^{-1} \\ &= (z_{i+1} - z_i) - (z_i - z_{i-1}) \\ &= z_{i+1} - 2z_i + z_{i-1}\end{aligned} \quad (13)$$

Hence, we can estimate the maximum of the remainder term from the LUT entries by the following expression:

$$\left|R(y_m)\right|_{\max} = \frac{1}{8}\left|z_{i+1} - 2z_i + z_{i-1}\right|_{\max} \quad (14)$$

4. Methods

We will illustrate the ideas suggested above with a case study. We will linearize a thermistor whose resistance depends on the temperature as follows [40]:

$$R_T = R_{25} \times \exp\left(\alpha\left(\frac{1}{T} - \frac{1}{298}\right)\right) \quad (15)$$

where the temperature T is in Kelvin [K]. In order to get a positive temperature coefficient and to get a voltage signal, we use the simple signal conditioning circuit in **Figure 6**.

This will produce a voltage $U(T)$ equal to

$$U(T) = \frac{1}{1 + \exp\left(\alpha\left(\frac{1}{T} - \frac{1}{298}\right)\right)} \times U_{ref} \quad (16)$$

and the ADC will produce an integer N according to

$$\begin{aligned}N &= \frac{U(T)}{U_{ref}} \times 2^n \\ &= \frac{1}{1 + \exp\left(\alpha\left(\frac{1}{T} - \frac{1}{298}\right)\right)} \times 2^n \\ &= f(T)\end{aligned} \quad (17)$$

We get the inverse, linearizing function by solving for T:

$$T = \frac{1}{\frac{1}{\alpha} \times \ln\left(\frac{2^n}{N} - 1\right) + \frac{1}{298}} = f^{-1}(N) \ [\mathrm{K}] \quad (18)$$

In order to get some numbers to work with, we will assign α a typical value of 3000 K **[40]**. We will also assume that we use a 16-bit ADC. Hence, expression (18) is

$$T = \frac{1}{\frac{1}{3000} \times \ln\left(\frac{65536}{N} - 1\right) + \frac{1}{298}} = f^{-1}(N) \quad (19)$$

Table 1. Implemented LUT (*y*,*z*).

y	$z = f^{-1}(y)$	Δf^{-1}	$\Delta^2 f^{-1}$
y_0	z_0	$\Delta^2 f_0^{-1} = \Delta f_1^{-1} - \Delta f_0^{-1}$	
y_1	z_1	$z_2 - z_1$	$\Delta^2 f_0^{-1} = \Delta f_1^{-1} - \Delta f_0^{-1}$
y_2	z_2	$z_3 - z_2$	$\Delta f_2^{-1} - \Delta f_1^{-1}$
y_3	z_3	$z_4 - z_3$	$\Delta f_3^{-1} - \Delta f_2^{-1}$
			$\Delta f_4^{-1} - \Delta f_3^{-1}$

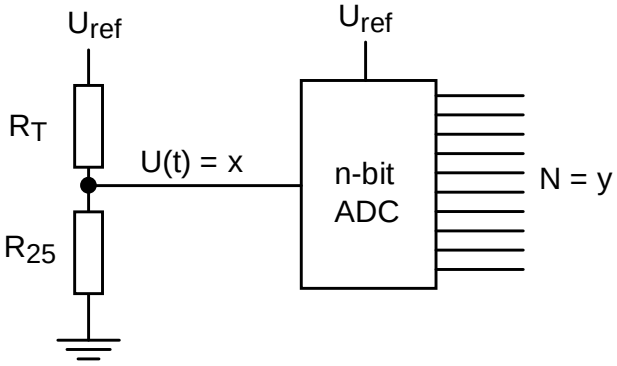

Figure 6. Signal conditioning for non-linear thermistor.

Ideally, we should store one LUT value for each value in the domain of N (0.65535), but we want to save memory space and decimate this table; we will estimate intermediate values by linear interpolation. According to (5), the interpolation error is proportional to the second derivative of expression (19). The second derivative corresponds to the "curviness" of the function and since we are looking for the size of the maximum interpolation error, we will focus on the segments of f^{-1} that has the greatest curviness. Also, we can estimate the second derivative from data by using expression (13). In **Figure 7** we have plotted both expression (19) and the second derivative from expression (13). As expected, the maximum of the second derivative occurs at the points of maximum curviness.

In **Figure 7** we have only plotted the temperature for N ranging from 13,602 to 57,889; this corresponds to an assumed temperature range of –10˚C to +100˚C.

5. Empiri/Results

From **Figure 7** we can see that the maximal curviness, and hence the maximal interpolation errors, occurs at the upper end of the f^{-1} function. Hence, we only need to calculate expression (12) for the last N values in order to find the maximal interpolation error for any decimation factor.

Table 2 lists the values of the maximal remainder term for different degrees of decimation. The remainder term

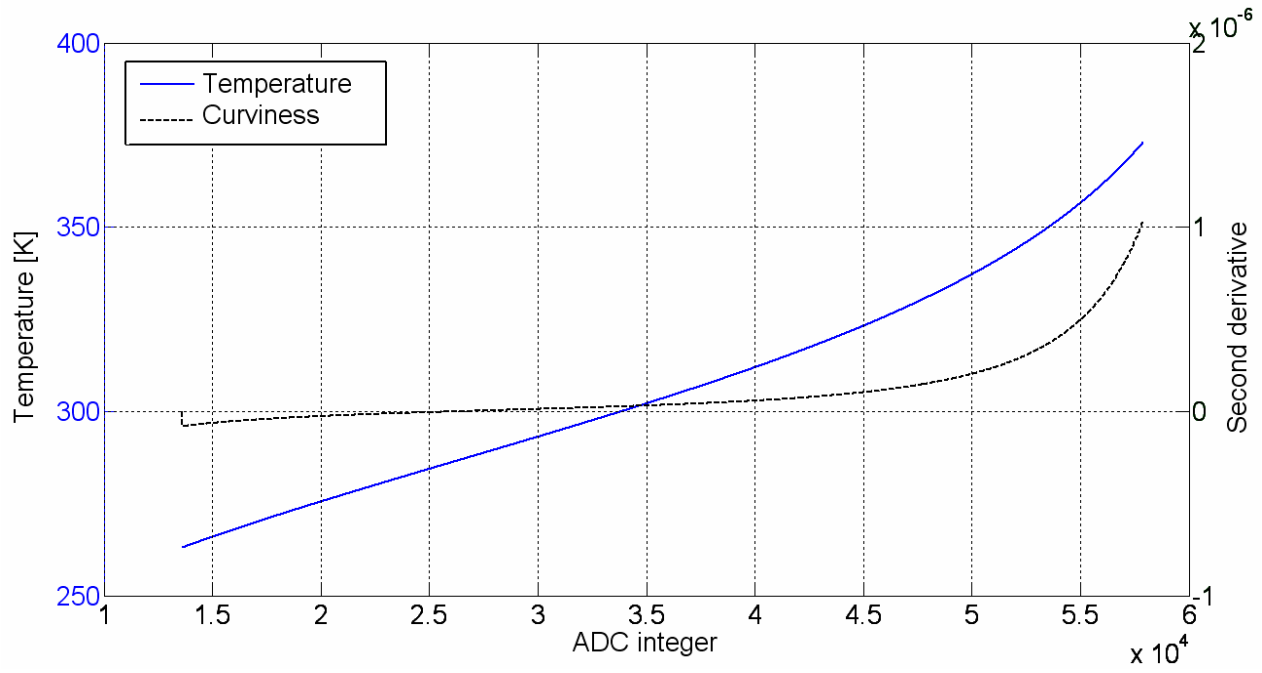

Figure 7. The function f^{-1} and its second derivative.

Table 2. Maximal remainder term vs LUT decimation factor.

Decimation factor	Maximal remainder
1	1.2896×10^{-7}
2	5.1553×10^{-7}
4	2.0597×10^{-6}
8	8.2191×10^{-6}
16	3.2720×10^{-5}
32	1.2965×10^{-4}
64	5.0810×10^{-4}
128	2.0021×10^{-3}
256	7.3008×10^{-3}

was calculated from expressions (12) and (13).

6. Discussion

The values in **Table 2** should be compared to the byte and word precisions in expressions (7) and (8).

If we use the restraint that $|R(y_m)| \ll |t_e|$, we can see from **Table 2** and expression (7) that if we store LUT entries in 8-bit byte format and use a 16-bit ADC, we can decimate the LUT by a factor of 64 without reducing the overall precision; the precision of the measurement estimate is still determined by the precision of the stored LUT entries, not by the errors introduced by the linear interpolation. If we store the LUT values as 16-bit words, we can see from **Table 2** and expression (8) that a decimation factor of 8 will not affect the overall precision.

For the 8-bit case, we reduce the size of the LUT from 65 kbyte to only 1 kbyte which is a significant (and absolutely necessary) reduction of the LUT memory footprint in a small embedded system. Of the 16-bit integer produced by the ADC, we use 10 bits for LUT indexing and 6 bits for (piece-wise) linear interpolation. This is illustrated in **Figure 8**.

7. Conclusions

LUTs are used in a vast variety of applications [1-18] and the precision of LUT entries and the size of the LUT are crucial design factors in every digital system that relies on LUTs. Wrong or inaccurate LUT entries may cause serious malfunctions. As a matter of fact, it was a LUT error that caused the infamous malfunction of Intel's Pentium processor in the mid 90's [41].

This work has suggested a method for finding an optimal LUT size when linearizing sensor outputs in small embedded systems. The idea is to use some of the (most significant) bits from the quantizer for LUT index ing and the rest of the (least significant) bits for linear interpolation. The allocation of bits for indexing and interpolation, respectively, depends on the precision of the stored LUT values. Since interpolation inherently introduces an error due to the curviness of the mapped function, it is important to keep the LUT size large enough to make the interpolation errors insignificant. On the other hand, the LUT size must be reduced to a minimum in order to

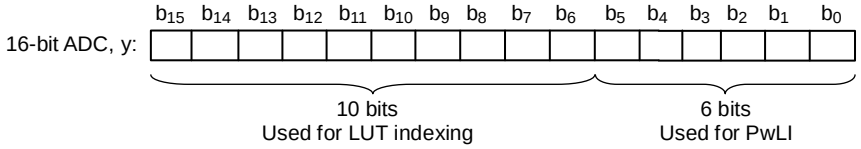

Figure 8. LUT indexing and Piece-wise Linear Interpolation.

minimize the LUT memory footprint, which is absolutely crucial in small embedded systems. This work has presented a theory for finding the optimal LUT size and demonstrated its use by a case study.

The work is limited to the case were LUT entries are stored as integers since this is typically the case in small (8-bit) microcontrollers; the controller linearizes the sensor signal only, it does not estimate the original sample value. We have argued that the interpolation error should be much less than the precision of the LUT entries. If the LUT entries are stored with m bits resolution we can estimate their precision to be 2^{-m} and if we use expressions (12) and (13) to estimate the interpolation error, we get the following general condition for determining the LUT size:

$$\frac{1}{8}\left|z_{i+1}-2z_i+z_{i-1}\right|_{\max} \ll 2^{-m} \qquad (20)$$

REFERENCES

[1] W. Gawronski, F. Baher and E. Gama, "Track- Level-Compensation Look-Up Table Improves Antenna Poiting Precision," 2006. http://ipnpr.jpl.nasa.gov/progress_report/42-164/164E.pdf

[2] R. A. Barlow, P. Bobbio, E. Carlier, G. Gräwer, N. Voumard and R. Gjelsvik, "The Beam Energy Tracking System of the LHC Beam Dumping System", *The* 10*th International Conferences on Accelerators* & *Large Experiment Physics Control Systems*, Geneva, 10-14 October, 2005, p. 02.056-4.

[3] P. Gaydecki, "New Real-Time Algorithms for Arbitrary, High Precision Function Generation with Applications to Acoustic Transducer Excitation," *Journal of Physics*: *Conference Series*, Vol. 178, 2009, pp. 12-15. doi:10.1088/1742-6596/178/1/012015

[4] P. Hung, H. Fahmy, O. Mencer and M. J. Flynn, "Fast Division with a Small Lookup Table," 2002. http://arith.stanford.edu/~hung/papers/asilomar.pdf

[5] M. D. Ercegovac, T. Lang, J.-M. Muller and A. Tisserand, "Reciprocation, Square Root, Inverse Square Root and Some Elementary Functions Using Small Multipliers," *IEEE Transactions on Computers*, Vol. 49, No. 7, 2000, pp. 628-637.

[6] H. Hassler and N. Takagi, "Functions Evaluation by Table Look-up and Addition." *Proceedings of the* 12*th IEEE Symposium Computer Arithmetic*, Bath, 19-21 July 1995, pp. 10-16. doi:10.1109/ARITH.1995.465382

[7] W. F. Wong and E. Goto, "Fast Evaluation of the Elementary Functions in Single Precision," *IEEE Transactions on Computers*, Vol. 44, No. 3, 1995, pp. 453-457. doi:10.1109/12.372037

[8] E. Balestrieri, P. Daponte and S. Rapuano, "A State of the Art on ADC Error Compensation Methods," *IEEE Transactions on Instrumentation and Measurement*, Vol. 54, No. 4, 2005, pp. 1388-1394. doi:10.1109/TIM.2005.851083

[9] A. C. Dent and C. F. N. Cowan, "Linearization of Analog-to-Digital Converters," *IEEE Transactions on Circuits and Systems*, Vol. 37, No. 6, 1990, pp. 729-737. doi:10.1109/31.55031

[10] M. Frey and H.-A. Loeliger, "On Flash A/D Converters with Low-Precision Comparators," *Proceedings of IEEE International Symposium on Circuits and Systems*, Greece, 21-24 May 2006, pp. 3926-3929.

[11] S. A. Jawed, "Analog-to-Digital Converter Design for Non-Uniform Quantization", Master Thesis, University of Linköping, Linköping, 2004. http://liu.diva-portal.org/smash/get/diva2:19990/FULLTEXT01

[12] M. Pascale, "Microcontrollers CORDIC Methods", 2004. http://www.drdobbs.com/184404244

[13] S. L. Gaverick, K. Fujino, D. T. McGrath and R. D. Baertsch, "A Programmable Mixed-Signal ASIC for Power Metering," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 12, 1991, pp 2008-2016. doi:10.1109/4.104195

[14] E. Laulainen, L. Koskinen, M. Kosunen and K. Halonen, "Compass Tilt Compensation Algorithm Using CORDIC," *Proceedings of the* 2008 *IEEE International Symposium on Circuits and Systems*, Vol. 1-10, 2008, pp 1188-1191.

[15] M. Beckman and L. Chioye, "Precision Thermocouple Measurement with the ADS1118," Texas Instruments, 2011. http://www.ti.com/lit/an/sbaa189/ sbaa189.pdf

[16] J. Julicher, "Simplified Thermocoupld Interfaces and PIC micro MCUs," *Microchip Technology*, 2002.

[17] Mathworks, "Look-up Tables and Polynomials," 2000. http://radio.feld.cvut.cz/matlab/toolbox/rtw/rtw_ug/opt_mod4.html

[18] B. C. Baker, "Precision Temperature-Sensing with RTD Circuits", 2008. http://ww1.microchip.com/downloads/en/appnotes/00687c.pdf

[19] J. Day and S. Bible, "Piecewise Linear Interpolation on PIC12/14/16 Series Microcontrollers," 2004. http://ww1.microchip.com/downloads/en/AppNotes/00942A.pdf

[20] J. M. D. Pereira, P. M. B. S. Girão and O. Postolache, "Fitting Transducer Characteristics to Measured Data,"

IEEE Instrumentation & Measurement Magazine, Vol. 4, No. 4, 2001, pp. 26-39.

[21] H. Erdem, "Implementation of Software-Based Sensor Linearization Algorithms on Low-Cost Microcontrollers," *ISA Transactions*, Vol. 49, No. 4, 2010, pp. 552-558. doi:10.1016/j.isatra.2010.04.004

[22] P. Hille, R. Höhler and H. Strack, "A Linearisation and Compensation Method for Integrated Sensors," *Sensors and Actuators A*, Vol. 44, No. 2, 1994, pp. 95-102. doi:10.1016/0924-4247(94)00795-0

[23] "PIC32MX1XX/2XX Data Sheet," Microchip Technology Inc., 2011. http://ww1.microchip.com/downloads/en/DeviceDoc/61168D.pdf

[24] K. Post, "Interpolated Table Lookups Using SSE2 [1/2]," 2010. http://rawstudio.org/blog/?p=457

[25] G. Bucci, M. Faccio and C. Landi "New ADC with Piecewise Linear Characteristic: Case Study—Implementation of a Smart Humidity Sensor". *IEEE Transactions on Instrumentation and Measurements*, Vol. 49, No. 6, 2000, pp. 1154-1166.

[26] S. Khan, A. H. M. Z. Alam, S. M. Ahmmad, I. B. Tijani, M. A. Hasan, L. W. Adetunji, S. F. Abdulazeez, S. H. M. Zaini, S. A. Othman and S. S. Khan, "On the Issue of Linearizing a Sensor Characteristic over a Wider Response Range", *Proceedings of the International Conference on Computer and Communication Engineering*, Kuala Lumpur, 13-15 May 2008, pp. 72-76.

[27] Microchip, "Temperature Sensor Design Guide", Microchip Inc., 2009. http://ww1.microchip.com/downloads/en/DeviceDoc/21895d.pdf

[28] B. Trump "Analog Linearization of Resistance Temperature Detectors," *Analog Application Journal*, Vol. 4Q, 2011, pp. 21-24.

[29] J. E. Brignell, "Software Techniques for Sensor Compensation," *Sensors and Actuators A*, Vol. 25, No. 1-3, 1991, pp. 29-35.

[30] B. Stringham, J. Leonard and S. Yakimchuk, "A Universal Sensor Linearizing Circuit," *Computers and Electronics in Agriculture*, Vol. 4, No. 1, 1989, pp. 81-84. doi:10.1016/0168-1699(89)90016-1

[31] D. K. Anvekar and B. S. Sonde, "Transducer Output Signal Processing Using Dual and Triple Microprocessor Systems," *IEEE Transactions on Instrumentation and Measurement*, Vol. 38, No. 3, 1989, pp.834-836. doi:10.1109/19.32204

[32] A. Flammini, D. Marioli and A. Taroni, "Transducer Output Signal Processing Using an Optimal Look-up Table in Microcontroller-Based Systems," *Electronics Letters*, Vol. 33, No. 14, 1997, pp. 1197-1198. doi:10.1049/el:19970809

[33] A. Flammini, D. Marioli and A. Taroni, "Application of an Optimal Look-up Table to Sensor Data Processing," *IEEE Transactions on Instrumentation and Measurement*, Vol. 48, No. 4, 1999, pp. 813-816. doi:10.1109/19.779179

[34] P. N. Mahana and F. N. Trofimenkoff, "Transducer Output Signal Processing Using an 8-Bit Microcontroller," *IEEE Transactions on Instrumentation and Measurement*, Vol. 35, No. 2, 1986, pp. 182-186.

[35] "Sensors, Excitation and Linearization", 2007. http://media.wiley.com/product_data/excerpt/33/07803601/0780360133-2.pdf

[36] Microchip, "PIC18FXX8 Data Sheet," Microchip Inc., 2003. http://www.micrchip.com/wwwproducts/Devices.aspx?dDocName=en010301

[37] S. Y. C. Catunda, O. R. Saavedra, J. V. FonsecaNeto and R. A. Morais, "Look-up Table and Breakpoints Determination for Piecwise Linear Approximation Functions Using Evolutionary Computation," *IMTC* 2003, *Instrumentation and Measurement Technology Conference*, Vail, 20-22 May 2003, pp. 435-440.

[38] L. Råde and B. Westergren, "Mathematics Handbook for Science and Engineering", 3rd Edition, Studentlitteratur, Lund, 1995.

[39] P. Pohl, G. Eriksson and G. Dahlquist,"Numeriska Metoder", 5th Edition, ILiber Tryck, Stockholm, 1982.

[40] E. O. Doeblin, "Measurement Systems—Application and Design," 4th Editio, McGraw-Hill, Singapore, 1990.

[41] B. Cipra, "How Number-Theory Got the Best of the Pentium Chip", *Science*, Vol. 267, No. 5295, 1995, p. 175. doi:10.1126/science.267.5195.175

