Aliaga, RJ. (2017). Real-time estimation of zero crossings of sampled signals for timing using cubic spline interpolation. IEEE Transactions on Nuclear Science. 64(8):2414-2422. https://doi.org/10.1109/TNS.2017.2721103

Real-time estimation of zero crossings of sampled signals for timing using cubic spline interpolation

Ramón J. Aliaga

***Abstract*—A scheme is proposed for hardware estimation of the location of zero crossings of sampled signals with sub-sample resolution for timing applications, that consists in interpolating the signal with a cubic spline near the zero crossing and then finding the root of the resulting polynomial. An iterative algorithm based on the bisection method is presented that obtains one bit of the result per step and admits an efficient FPGA implementation using fixed-point representation. In particular, the root estimation iteration involves only two additions, and the initial values can be obtained from FIR filters with certain symmetry properties. It is shown that this allows online, real-time estimation of timestamps in free-running sampling detector systems with improved accuracy with respect to the more common linear interpolation. The method is evaluated with simulations using ideal and real timing signals and estimates are given for the resource usage and speed of its implementation.**

***Index Terms*—Digital arithmetic, digital circuits, digital timing, FPGA, interpolation, signal processing algorithms, splines, time estimation, time resolution.**

I. Introduction

TIME pick-off, i.e. the process of assigning an univocally determined time mark to detected events, is one of the fundamental functions of electronic detector and data acquisition systems [1]. Time marks usually correspond to captured signals crossing certain predetermined values. The simplest procedure, known as Leading Edge Discrimination [2], consists in determining the time when an input signal $s(t)$, usually a pulse with a fast edge, crosses a certain threshold value v; equivalently, the assigned time mark is the zero crossing of $y(t) = s(t) - v$. A more refined method is Constant Fraction Discrimination (CFD), where the time mark corresponds to the zero crossing of a bipolar signal generated from the original signal $s(t)$ and a delayed copy of it [3].

These methods have been traditionally implemented with analog electronics, using fast comparators to generate digital pulses as the crossings take place. However, the current trend is to digitize detector signals as early as possible in order to preserve signal integrity and implement an increasing part of the required signal processing in digital devices. Established time pick-off methods have been translated to the digital domain, using a discrete signal that is obtained from the input samples and may depend on reconfigurable parameters such as those in digital CFD [4].

Digital timestamping can thus be interpreted as the problem of recovering the zero crossing of a continuous signal $y(t)$ from its sampled values y_n. The location of the zero crossing can be narrowed down to a single sampling interval by looking for sign changes between consecutive samples, but applications such as PET [4] and others based on time-of-flight measurements between detectors in the same array [5] usually demand a precision better than the typical sampling periods of 4–10 ns, and require methods for sub-sample resolution. The simplest option is linear interpolation between the samples at the edges of the selected interval to estimate the zero location [6]. This procedure can theoretically achieve good timing resolution [7] and is simple enough that it can be implemented online on FPGA for real-time processing [8], [9].

A more sophisticated option is to use a higher order polynomial to interpolate the signal in the interval of interest and then estimate the location of the root of the resulting polynomial. This has been shown to achieve better results, especially for low sampling frequencies that result in loss of fine signal details like inflection points [10]. In particular, cubic spline interpolation may be employed; it has been used for image processing for a long time [11] and the interpolation procedure i.e. computation of the polynomial coefficients has been shown to admit efficient hardware implementations [12], [13]. However, the zero search procedure is more complex and is generally implemented on software [5], [14]. Mixed approaches are possible such as [15], where an upsampled spline interpolation is obtained first and the zero crossing is determined by linear interpolation on the upsampled signal.

In this paper, we present an efficient algorithm for estimation of the zero crossing of the cubic interpolation. Each iteration yields one bit of the result and only requires two additions and bit shift operations, making it particularly suitable for real-time implementation in FPGA with very low computational complexity and limited area requirements. In timing applications, this allows fast online computation of fine timestamps of detected events very early in the data acquisition chain, so they can be used for early trigger and filtering decisions. After reviewing the more common method of linear interpolation, the extension to the cubic case is analyzed and its properties are described. The method is then evaluated using simulations with ideal and real timing signals, and area and speed estimates of its possible hardware implementations are obtained. Some theoretical properties of the algorithm are stated without proof; their proofs have been obtained but have not been included in the paper for the sake of brevity.

Manuscript received Feb 14, 2017; revised May 21, 2017 and Jun 20, 2017; accepted Jun 26, 2017.

This work was partially supported by the Generalitat Valenciana, Spain, under Grant PROMETEOII/2014/019.

The author is with the Instituto de Física Corpuscular (CSIC-UV), C/ Catedrático José Beltrán 2, 46980 Paterna, Spain, and with the Instituto Universitario de Matemática Pura y Aplicada, Universitat Politècnica de València, Camino de Vera S/N, 46022 Valencia, Spain.

E-mail: raalva@upvnet.upv.es.

II. Linear Interpolation

We consider a continuous-time signal $y(t)$ and the problem of estimating the time t_0 of its zero crossing from its samples $y_n = y(n)$. We assume that the samples have been obtained with an ADC with F bit precision and uniform period which we normalize to 1 and that, after proper scaling, the samples are quantized and represented digitally as numbers $y_n \in [-1, 1]$, using two's complement fixed-point representation with one sign bit, $F-1$ fractional bits and no integer bits, so that their values are integer multiples of ulp $= 2^{-(F-1)}$, where "ulp" stands for Unit in the Last Place [16]. We also assume that the sampling is dense enough that each interval between samples contains at most one zero crossing of y and so zero crossings can be put into one-to-one correspondence with those intervals where y changes its sign, i.e. where $\operatorname{sign} y_n \neq \operatorname{sign} y_{n+1}$.

The first step of the solution consists in locating the first sign change in the sample stream in order to constrain t_0 to a sampling interval; we will assume that this happens between samples y_0 and y_1. The second, more complex step then involves estimating the location of t_0 within $[0, 1]$. To this end, a model $f(t)$ of the original signal $y(t)$ is constructed that is intended to be a good approximation of it on $[0, 1]$, and the zero crossing $t^\star$ of $f(t)$ is obtained instead as an estimation of the true zero crossing t_0. We restrict ourselves to the case where f interpolates y at the interval end-points, i.e. it is constrained to $f(0) = y_0$ and $f(1) = y_1$.

The simplest option is to consider a linear approximation $f(t)$, which is completely determined by the two known end-points. Expressing $f(t) = \Delta \cdot (t - t^\star)$, where $\Delta = y_1 - y_0$, immediately yields

$$t^\star = -\frac{y_0}{\Delta} = \frac{y_0}{y_0 - y_1}. \tag{1}$$

This value may be obtained online using just an adder and a divider circuit, but here we examine a different approach: estimating $t^\star$ using the well-known bisection method for root finding [17]. This method consists in iteratively evaluating the continuous function f at the midpoint of an interval where f changes its sign, and using signs to determine the half-interval where the root is located. More specifically, let $a_0 = 0$ and $b_0 = 1$, and suppose we begin the n-th iteration with an interval $[a_{n-1}, b_{n-1}]$ such that $\operatorname{sign} f(a_{n-1}) \neq \operatorname{sign} f(b_{n-1})$. Then f is evaluated at

$$\mu_n = \frac{1}{2}(a_{n-1} + b_{n-1}). \tag{2}$$

If $\operatorname{sign} f(\mu_n) = \operatorname{sign} f(b_{n-1})$ then a zero crossing must exist within $[a_{n-1}, \mu_n]$, otherwise the root is known to lie in $[\mu_n, b_{n-1}]$; the appropriate subinterval is selected for the next iteration. This procedure is repeated for M iterations until an interval $[a_M, b_M]$ of length 2^{-M} that contains $t^\star$ is obtained; a_M is then an estimation of $t^\star$ with a precision of 2^{-M}.

Application of the bisection method to a linear function f admits an efficient digital implementation for two reasons. First, evaluation of f at the midpoint of an interval only requires knowledge of the values at the extremes because

$$f(\mu_n) = \frac{1}{2}[f(a_{n-1}) + f(b_{n-1})]. \tag{3}$$

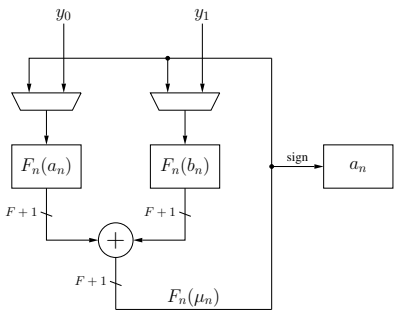


Fig. 1. Implementation of the zero crossing estimator using linear bisection.

Hence, it is only necessary to keep track of the values of $f(a_n)$ and $f(b_n)$ after each iteration, and computation only requires an addition. Second, the comparison of signs and choice of subinterval in the n-th iteration directly yields the n-th fractional bit of $t^\star$: it is a 0 if $t^\star$ is located in the left half $[a_{n-1}, \mu_n]$ or a 1 otherwise. The binary representation of $t^\star$ thus consists of the (possibly inverted) sign bits of successive values of $f(\mu_n)$, and is built one bit at a time until the first M bits are obtained.

One possible hardware realization of the algorithm may be based on storing $f(a_n)$ and $f(b_n)$ in registers with fixed-point representation, using more than the initial F bits, and then iteratively computing $f(\mu_n)$ and updating the $f(a_n)$ and $f(b_n)$ accordingly. However, this is inefficient due to the variation over time in the dynamic range of the values of f; specifically, $f(a_n)$ and $f(b_n)$ are $O(2^{-n})$. This implies that the least significant fractional bits in these registers would be unused initially, i.e. equal to zero, then start being taken into account progressively as the iteration number increases at a rate of one bit per iteration. As a_n and b_n converge to $t^\star$, the most significant bits would then become unused eventually.

Instead, it makes more sense to store the value of $F_n = 2^n f$, which is equivalent to a "moving-point" representation where the radix point is shifted one position to the left at a time. The iteration step (3) then takes the form

$$F_n(\mu_n) = F_{n-1}(a_{n-1}) + F_{n-1}(b_{n-1}). \tag{4}$$

In particular, this involves no division or shift and no truncation takes place; hence, the operation introduces no finite precision errors. Moreover, if x_n denotes one of the abscissae a_n, b_n or μ_n, then $|x_n - t^\star| \leq 2^{-n}$ and the bound

$$|F_n(x_n)| = 2^n\,|f(x_n)| = 2^n\,|\Delta|\,|x_n - t^\star| \leq |\Delta| \leq 2 \tag{5}$$

is obtained. Hence, only one additional integer bit is needed per register in order to store $F_n(a_n)$ and $F_n(b_n)$ without risk of overflow. This representation with minimal register width has the benefits of reduced area and power consumption and increased operating frequency by shortening the adder size and the critical data path [18].

The hardware implementation of this procedure is outlined in Fig. 1. It only requires an $(F+1)$-bit adder for the computation of $F_n(\mu_n)$ and two $(F+1)$-bit registers for storage of $F_n(a_n)$, $F_n(b_n)$, plus an M-bit register to store the estimation a_n. Note that the adder output is $F+2$ bits

wide, but one of them can be safely discarded as guaranteed by bound (5). All registers in Fig. 1 are shift registers: a_n shifts every cycle, while the F_n either shift or load a parallel value depending on the sign bit of $F_n(\mu_n)$. Multiplexers are included that allow loading the initial values. The control circuitry and signals are very simple and are omitted for clarity.

III. Cubic Spline Interpolation

A. Cubic Splines

Cubic approximations of the form

$$f(t) = c_3 t^3 + c_2 t^2 + c_1 t + c_0 \tag{6}$$

are considered now. Since $f(0)$ and $f(1)$ are fixed, there are two degrees of freedom for their selection. The most obvious choice is the cubic polynomial interpolating y at the four points y_{-1}, y_0, y_1 and y_2, which has been used previously e.g. in [10]; however, this is prone to introducing unwanted oscillations [19]. An alternative approach is to use cubic spline approximations [17], i.e. piecewise polynomial functions f that are defined as a different cubic polynomial in each interval between interpolation nodes in such a way that f, f' and f'' remain continuous. Splines are the smoothest possible approximations of third degree in a certain sense [11], [12] and provide uniform approximations not only of y but also of its derivatives [20]. The case will be considered where the spline interpolation is obtained for a set of $2N$ consecutive samples y_k, $-(N-1) \le k \le N$ centered at the zero crossing interval, for $N \ge 2$. A different cubic polynomial is then defined for each interval, but only the polynomial (6) corresponding to $[0, 1]$ is of interest to the root-finding procedure.

The definition of cubic splines given so far does not determine them completely, as there are still two free parameters. The choice of different additional conditions defines different types of cubic spline. Two of them that do not involve additional data are considered here: *natural splines*, for which $f'' = 0$ is imposed at the two extreme nodes, and *parabolically terminated splines*, such that the polynomials at the first and last interval are quadratic instead of cubic.

For both types of splines, the coefficients c_i of f may be obtained as fixed linear combinations of the set of samples y_k under consideration. More specifically, if the notation

$$\mathbf{c} = \begin{bmatrix} c_3 & c_2 & c_1 & c_0 \end{bmatrix}^T \tag{7}$$

$$\mathbf{y} = \begin{bmatrix} y_{-(N-1)} & y_{-(N-2)} & \cdots & y_N \end{bmatrix}^T \tag{8}$$

is used, then there is a $4 \times 2N$ matrix $\mathbf{M}$, which depends only on N, such that $\mathbf{c} = \mathbf{M}\mathbf{y}$. This matrix can be expressed as $\mathbf{M} = \mathbf{U} + \mathbf{V}\mathbf{A}^{-1}\mathbf{B}$ where

$$\mathbf{A} = \begin{bmatrix} \omega & 1 & 0 & \cdots & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 \\ 0 & 0 & 0 & \cdots & 1 & \omega \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -\zeta & \zeta & 0 & \cdots & 0 & 0 \\ -3 & 0 & 3 & \cdots & 0 & 0 \\ 0 & -3 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 3 \\ 0 & 0 & 0 & \cdots & -\zeta & \zeta \end{bmatrix} \tag{9}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{z} & 2 & -2 & \mathbf{z} \\ \mathbf{z} & -3 & 3 & \mathbf{z} \\ \mathbf{z} & 0 & 0 & \mathbf{z} \\ \mathbf{z} & 1 & 0 & \mathbf{z} \end{bmatrix}, \mathbf{V} = \begin{bmatrix} \mathbf{z} & 1 & 1 & \mathbf{z} \\ \mathbf{z} & -2 & -1 & \mathbf{z} \\ \mathbf{z} & 1 & 0 & \mathbf{z} \\ \mathbf{z} & 0 & 0 & \mathbf{z} \end{bmatrix} \tag{10}$$

and $\mathbf{z} = \begin{bmatrix} 0 & \ldots & 0 \end{bmatrix}$ is a vector of $N-1$ zeros, with parameters $\omega = 2$, $\zeta = 3$ for natural splines and $\omega = 1$, $\zeta = 2$ for parabolically terminated splines. This implies that if N is fixed beforehand then the coefficients $c_i = \mathbf{m}_i \cdot \mathbf{y}$ can be computed by applying appropriate FIR filters with constant coefficients, given by the rows $\mathbf{m}_i$ of $\mathbf{M}$, to the stream of input samples. Moreover, there is an integer $D = \det \mathbf{A}$ such that $D\mathbf{M}$ consists of integer elements; hence, if the coefficients Dc_i of the polynomial $g(t) = Df(t)$ are computed instead, then the resulting FIR filters have only integer coefficients. The root finding procedure can then be applied to g instead of f, as they have the same roots.

B. Bisection Based Implementation

We now show that efficient computation of $f(\mu_n)$ is possible when f is cubic so the bisection method can still be applied. To this end, we apply the expansion formula

$$\begin{aligned} f(a+\xi) = f(a) + 3c_3\xi\, a^2 + (3c_3\xi^2 + 2c_2\xi)a \\ + (c_3\xi^3 + c_2\xi^2 + c_1\xi) \end{aligned} \tag{11}$$

to $\mu_n = a_{n-1} + 1/2^n$ and $b_{n-1} = a_{n-1} + 1/2^{n-1}$, in order to evaluate the error incurred when approximating $f(\mu_n)$ as (3). After substitution and simplification, the recursive formula

$$f(\mu_n) = \frac{f(a_{n-1}) + f(b_{n-1})}{2} - \frac{1}{4^n}(c_2 + 3c_3\mu_n) \tag{12}$$

is obtained.

Two changes are now applied with the digital implementation in mind. First, we perform bisection on $g = Df$ instead of f, so that we can use integer filters. Second, the issue discussed in Section II regarding storage of values of g of decreasing magnitude still applies in this case, and suggests basing the algorithm on $G_n = 2^n g$ instead. We thus multiply (12) with $2^n D$ to obtain the iteration formula

$$G_n(\mu_n) = G_{n-1}(a_{n-1}) + G_{n-1}(b_{n-1}) + K_{n-1} \tag{13}$$

where K_n is the cubic correction term

$$K_n = -\frac{1}{2^{n+1}}(Dc_2 + 3Dc_3\mu_{n+1}). \tag{14}$$

This term can be computed iteratively, too, by noticing that

$$\mu_{n+1} = \mu_n \pm \frac{1}{2^{n+1}} \tag{15}$$

where the sign is $-$ or $+$ if the bit of $t^\star$ obtained in the n-th iteration is a 0 or a 1, respectively. Hence

$$\begin{aligned} K_n &= -\frac{1}{2^{n+1}}\left[Dc_2 + 3Dc_3\left(\mu_n \pm \frac{1}{2^{n+1}}\right)\right] \\ &= \frac{1}{2}\left[-\frac{1}{2^n}(Dc_2 + 3Dc_3\mu_n) \mp \frac{1}{2^n}\frac{3Dc_3}{2^{n+1}}\right] \\ &= \frac{1}{2}(K_{n-1} \pm L_{n-1}) \end{aligned} \tag{16}$$

Algorithm 1 Iterative cubic interpolation

Fixed data: Number of iterations/bits M; number of nodes $2N$; coefficients D, $\mathbf{k}$ and $\mathbf{l}$
Input: Stream y_n of signal samples
Detect a zero crossing as $\operatorname{sign} y_0 \neq \operatorname{sign} y_1$
Initialize $K_0 \leftarrow \mathbf{k} \cdot \mathbf{y}$, $L_0 \leftarrow \mathbf{l} \cdot \mathbf{y}$
Initialize $a_0 \leftarrow 0$, $G_0(a_0) \leftarrow Dy_0$, $G_0(b_0) \leftarrow Dy_1$
for $n = 1$ to M **do**
  $G_n(\mu_n) \leftarrow G_{n-1}(a_{n-1}) + G_{n-1}(b_{n-1}) + K_{n-1}$
  if $\operatorname{sign} G_n(\mu_n) = \operatorname{sign} G_{n-1}(a_{n-1})$ **then**
    $a_n \leftarrow \{a_{n-1}, 1\}$
    $K_n \leftarrow (K_{n-1} + L_{n-1})/2$
    $G_n(a_n) \leftarrow G_n(\mu_n)$
    $G_n(b_n) \leftarrow 2G_{n-1}(b_{n-1})$
  else
    $a_n \leftarrow \{a_{n-1}, 0\}$
    $K_n \leftarrow (K_{n-1} - L_{n-1})/2$
    $G_n(a_n) \leftarrow 2G_{n-1}(a_{n-1})$
    $G_n(b_n) \leftarrow G_n(\mu_n)$
  end if
  $L_n \leftarrow L_{n-1}/4$
end for
Output: a_M, estimation of $t^\star$

where

$$L_n = \frac{-3Dc_3}{2^{2n+3}}. \tag{17}$$

This results in the procedure described in Algorithm 1, where the notation $\{a, \beta\}$ stands for concatenation of bit $\beta \in \{0, 1\}$ to the right of a, and the circuit depicted in Fig. 2, which is an extension of the one used for linear bisection. It requires registers for $G_n(a_n)$, $G_n(b_n)$, K_n and L_n plus the result a_n; except for K_n, all of them are shift registers (L_n shifts two bits per iteration). Their optimal widths are shown; they will be obtained in Section III-D. The circuit also requires a three input adder for $G_n(\mu_n)$ using (13) and a two input adder for K_n using (16). Note that the second addition is dependent on the first one, because the sign bit of $G_n(\mu_n)$ determines whether L_n is added or subtracted in (16). Since both adders have comparable size, an adequately pipelined version of the circuit is obtained by registering the partial result $G_n(\mu_n)$ (indicated by a dashed box). This increases its operating frequency, but also reduces its throughput from one estimation every M clock cycles to one every $2M$ cycles.

C. Computation of Initial Values

Function values are initialized as $G_0(a_0) = g(a_0) = Dy_0$ and $G_0(b_0) = g(b_0) = Dy_1$, and coefficients are initialized as $K_0 = \mathbf{k} \cdot \mathbf{y}$ and $L_0 = \mathbf{l} \cdot \mathbf{y}$ where

$$\mathbf{k} = -\tfrac{1}{2} D\mathbf{m}_2 - \tfrac{3}{4} D\mathbf{m}_3 \tag{18}$$

$$\mathbf{l} = -\tfrac{3}{8} D\mathbf{m}_3 \tag{19}$$

according to (14) and (17), since $\mu_1 = \frac{1}{2}$; recall that $\mathbf{m}_3$ and $\mathbf{m}_2$ are the first two rows of $\mathbf{M}$. It is thus only necessary to implement FIR filters with the coefficients in $\mathbf{k}$ and $\mathbf{l}$ instead of all the $\mathbf{m}_i$. These filters turn out to have certain symmetry properties that allow efficient implementations. In order to state them, we fix some notation: if $\mathbf{v}$ is a vector, $\mathbf{v}^\times$ denotes the vector obtained by reversing the order of elements of $\mathbf{v}$, and $\mathbf{v}$ is said to be even (or symmetrical) if $\mathbf{v} = \mathbf{v}^\times$, and odd (or antisymmetrical) if $\mathbf{v} = -\mathbf{v}^\times$. Then it is possible to prove that $\mathbf{k}$ is even and $\mathbf{l}$ is odd. Moreover, their coefficients are related and they can be expressed as

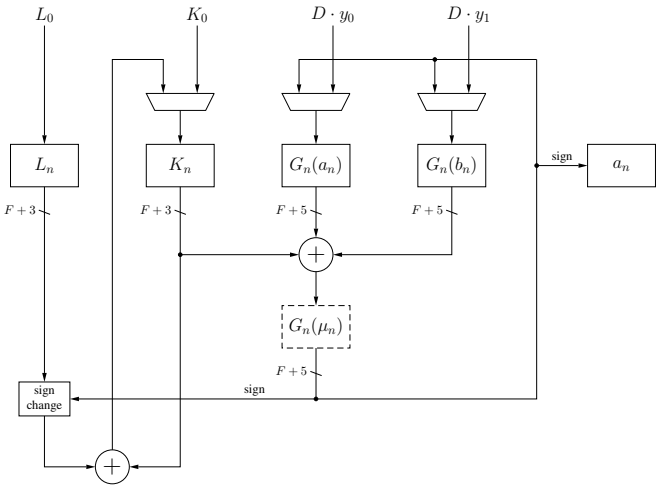


Fig. 2. Implementation of the zero crossing estimator using cubic bisection. The circuit can be optionally pipelined by registering $G_n(\mu_n)$.

$$\mathbf{k} = A_K \cdot \begin{bmatrix} \boldsymbol{\alpha} & \beta & \beta & \boldsymbol{\alpha}^\times \end{bmatrix} \tag{20}$$

$$\mathbf{l} = A_L \cdot \begin{bmatrix} \boldsymbol{\alpha} & \gamma & -\gamma & -\boldsymbol{\alpha}^\times \end{bmatrix} \tag{21}$$

where $\boldsymbol{\alpha}$ is an integer vector, β and γ are integers, and A_K and A_L are integers divided by powers of two.

Table I shows the values of D, $\mathbf{k}$ and $\mathbf{l}$ in the form (20), (21) for $N \leq 5$ and both types of splines. In each case, the smallest integer value of D is given that results in integer vectors except for a negative power of two, which has no impact on hardware implementation. Only half of $\mathbf{k}$ and $\mathbf{l}$ are shown; they are meant to be extended evenly and oddly, respectively. Note that coefficient values grow with N and are impractical for $N > 5$.

Different implementations of the FIR filters $\mathbf{k}$, $\mathbf{l}$ have been considered. The most obvious choice is the so-called direct or tapped delay line form shown in Fig. 3, where samples are stored in a shift register and the filter outputs are computed directly as weighted sums [21]. Filter symmetry allows a reduction in the amount of constant coefficient multipliers by adding or subtracting pairs of samples before weighting. However, the fact that $\mathbf{k}$ and $\mathbf{l}$ exhibit different kinds of symmetry prevents us from taking advantage of their common coefficients $\boldsymbol{\alpha}$. Moreover, this structure includes a very long combinational path featuring adder trees with N inputs, whose delay increases with N.

These disadvantages are typically overcome by implementing the transposed form of the filters [21]. In that case, the current sample is weighted by each coefficient separately and a shift register with partial filter sums is employed instead. This allows sharing the multipliers between both filters, and also reduces the critical path; moreover, the resulting structure

TABLE I
INTEGER VECTORS FOR INITIAL CUBIC BISECTION COEFFICIENTS

N	Natural splines	Parabolically terminated splines
2	$D = 15 \quad , \quad S = 15$ $\mathbf{k} = \frac{1}{4} \times (-9) \times \begin{bmatrix} 1 & -1 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{8} \times 15 \times \begin{bmatrix} 1 & -3 & \dots \end{bmatrix}$	$D = 1 \quad , \quad S = 0.75$ $\mathbf{k} = \frac{1}{8} \times (-1) \times \begin{bmatrix} 1 & -1 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{32} \times 3 \times \begin{bmatrix} 1 & -3 & \dots \end{bmatrix}$
3	$D = 209 \quad , \quad S = 285$ $\mathbf{k} = \frac{1}{4} \times 33 \times \begin{bmatrix} 1 & -6 & 5 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{8} \times (-57) \times \begin{bmatrix} 1 & -6 & 13 & \dots \end{bmatrix}$	$D = 7 \quad , \quad S = 9$ $\mathbf{k} = \frac{1}{32} \times 7 \times \begin{bmatrix} 1 & -7 & 6 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{16} \times (-3) \times \begin{bmatrix} 1 & -7 & 16 & \dots \end{bmatrix}$
4	$D = 2911 \quad , \quad S = 4260$ $\mathbf{k} = \frac{1}{4} \times (-123) \times \begin{bmatrix} 1 & -6 & 24 & -19 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{8} \times 213 \times \begin{bmatrix} 1 & -6 & 24 & -49 & \dots \end{bmatrix}$	$D = 195 \quad , \quad S = 281.25$ $\mathbf{k} = \frac{1}{8} \times (-13) \times \begin{bmatrix} 1 & -7 & 30 & -24 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{32} \times 45 \times \begin{bmatrix} 1 & -7 & 30 & -62 & \dots \end{bmatrix}$
5	$D = 40545 \quad , \quad S = 60420$ $\mathbf{k} = \frac{1}{4} \times 459 \times \begin{bmatrix} 1 & -6 & 24 & -90 & 71 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{8} \times (-795) \times \begin{bmatrix} 1 & -6 & 24 & -90 & 183 & \dots \end{bmatrix}$	$D = 679 \quad , \quad S = 1008$ $\mathbf{k} = \frac{1}{64} \times 97 \times \begin{bmatrix} 1 & -7 & 30 & -114 & 90 & \dots \end{bmatrix}$ $\mathbf{l} = \frac{1}{16} \times (-21) \times \begin{bmatrix} 1 & -7 & 30 & -114 & 232 & \dots \end{bmatrix}$

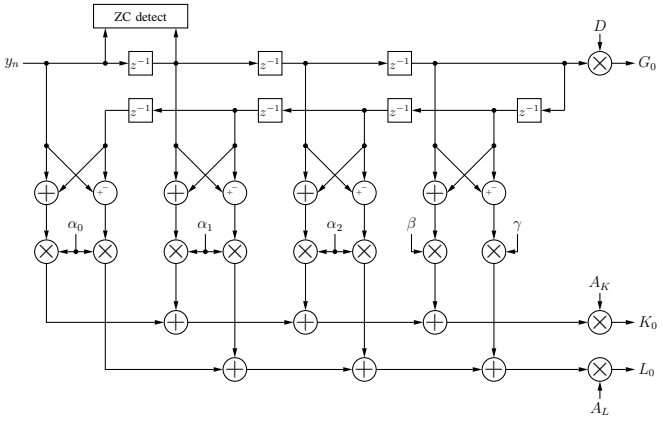


Fig. 3. Direct (tapped delay line) FIR filter structure for simultaneous computation of the initial bisection values, for $N = 4$.

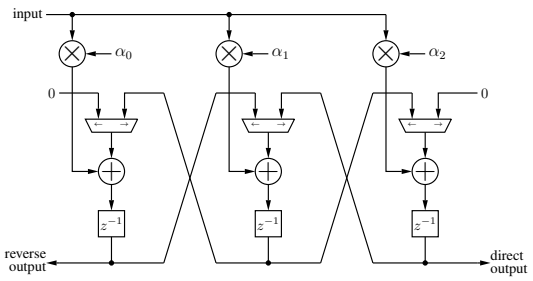


Fig. 4. Bidirectional transposed FIR filter structure, for $N = 4$.

featuring cascaded adder-register elements is particularly efficient for FPGA implementations. The drawback in that case is that separate register chains are needed for $\mathbf{k}$ and $\mathbf{l}$, and thus more registers are needed, which moreover need to be wider as they store multiplication-accumulation results instead of the raw samples as in the direct case.

This problem can be partially solved by using a custom structure that takes advantage of the common coefficients $\boldsymbol{\alpha}$

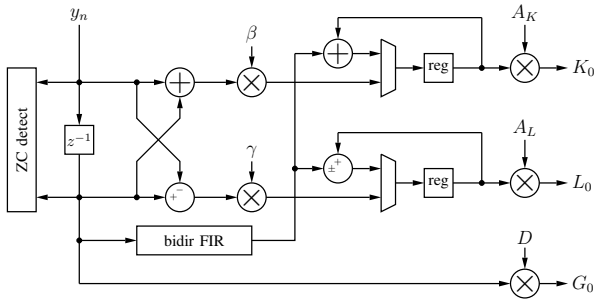


Fig. 5. Custom filter structure for simultaneous computation of the initial bisection values.

and the symmetry simultaneously as follows: express the initial values K_0, L_0 as

$$K_0 = \mathbf{k} \cdot \mathbf{y} = A_K \cdot [d + \beta(y_0 + y_1) + r] \tag{22}$$

$$L_0 = \mathbf{l} \cdot \mathbf{y} = A_L \cdot [d + \gamma(y_0 - y_1) - r] \tag{23}$$

where

$$\begin{aligned} d &= \boldsymbol{\alpha} \cdot \begin{bmatrix} y_{-(N-1)} & y_{-(N-2)} & \cdots & y_{-1} \end{bmatrix}^T \\ &= \alpha_0 y_{-(N-1)} + \alpha_1 y_{-(N-2)} + \ldots + \alpha_{N-2} y_{-1}, \end{aligned} \tag{24}$$

$$\begin{aligned} r &= \boldsymbol{\alpha}^{\times} \cdot \begin{bmatrix} y_2 & y_3 & \cdots & y_N \end{bmatrix}^T \\ &= \alpha_{N-2} y_2 + \alpha_{N-3} y_3 + \ldots + \alpha_0 y_N \end{aligned} \tag{25}$$

are the result of applying the same filter $\boldsymbol{\alpha}$ to the first and last $N - 1$ samples in $\mathbf{y}$, first in direct order (d) and then in reverse order (r). The transposed FIR filter structure can be modified to allow for bidirectional operation as shown in Fig. 4, where a common signal controls all multiplexers and determines whether the data flow is from left to right (direct) or vice versa (reverse); the same circuit can then be used to compute d and r, as they correspond to non-overlapping windows of the input signal. The full, custom filter structure is depicted in Fig. 5; control signals are omitted for clarity.

D. Error Analysis and Dimensioning

Unlike the linear case, values stored in the proposed cubic method converge to 0 at different speeds, so truncation errors appear eventually. An analysis of finite precision effects is thus necessary for proper circuit dimensioning. It seems reasonable to store the values $G_n(a_n)$, $G_n(b_n)$, K_n and L_n in registers with the same precision, as they are added with each other directly in (13) and (16). Suppose that they use ulp $= 2^{-Q}$, where Q is allowed to be smaller than $F-1$ or even negative, with the meaning that the lowest $-Q$ integer bits are discarded in the latter case. Then it is possible to prove that the absolute error in the evaluation of g is bounded by $1/2^{Q-1}$, and that the algorithm always converges to some $\hat{t}^\star$ such that $\left|g(\hat{t}^\star)\right| \leq 1/2^{Q-1}$. An appropriate design choice is then to choose Q such that the evaluation error can be neglected by forcing it to be smaller than the quantization error in g, which is bounded by $D/2^{F-1}$. This criterion yields a minimum value

$$Q = F - \lfloor \log_2 D \rfloor \tag{26}$$

where $\lfloor x \rfloor$ denotes the greatest integer not exceeding x. Notice that Q decreases as N and D increase: since the samples are amplified by a factor of D before bisection, the ulp may be increased while maintaining the same relative precision.

Maximum values of K_n, L_n and $G_n(x_n)$ also need to be estimated for register sizing. It is easy to see that K_n and L_n are both bounded by $S = \max\{\|\mathbf{k}\|_1, \|\mathbf{l}\|_1\}$, where $\|\mathbf{v}\|_1$ denotes the sum of the absolute values of the elements of vector $\mathbf{v}$; the values of S are included in Table I, and it can be observed that they are always bounded by $2D$ for all values of N under consideration. Regarding G_n, it can be proven rigorously that its value is always bounded by $8D$ if one assumes that g is monotonic in the approximation interval; we conjecture that the stronger bound $3\sqrt{3}D \approx 5.2D$ actually applies and that the hypothesis of monotonicity is not necessary, but we were not able to prove it. We remark that the assumption that f is monotonic is reasonable in timing applications: linearization around the zero crossing shows that time resolution can be estimated as [2]

$$\sigma_t \approx \frac{\sigma_y}{|y'(t_0)|} \approx \frac{\sigma_y}{|f'(t^\star)|} \tag{27}$$

where σ_y is the noise in the input signal. Hence, proper design will try to force the timing signals to cross the zero level with the steepest possible slope. If the sampling is dense enough, the signal will therefore be monotonic on the zero crossing interval; in fact, its derivative should not vary much and may be estimated as Δ.

It follows from these bounds that the amount of integer bits for the registers holding G_n and K_n/L_n need not be higher than $3 + \lceil \log_2 D \rceil$ and $1 + \lceil \log_2 D \rceil$, respectively, to avoid overflow; here, $\lceil x \rceil$ denotes the smallest integer which is not smaller than x. Combining this with (26), the optimal register widths are $F+5$ and $F+3$ bits for G_n and K_n/L_n, respectively. Hence the size of the zero crossing estimator circuit in Fig. 2 is completely independent of N: it only depends on the bit resolution F of the original signal y_n.

IV. Algorithm Evaluation

The validity of the proposed algorithms was evaluated by applying them to CFD signals offline using commercial software (MATLAB 2014b, The Mathworks, Inc., Natick, MA). Ideal, simulated detector pulses and digitized signals from a real detector setup were both tested. A database of captured timing pulses was used which had been acquired using the experimental setup and DAQ system described in detail in [9]. The setup consisted of a pair of continuous scintillating crystals coupled to position-sensitive photomultiplier tubes working in coincidence, with a 511 keV point source placed close to one of them. The last dynode signals from both PMTs were shaped with a pole-zero cancellation filter and sampled at 156.25 MHz with 12-bit ADCs. A set of 28000 pairs of timing signals corresponding to position-, energy- and coincidence-filtered events was used for the tests.

Simulated noiseless detector pulses with similar parameters as the real ones were used first, for two reasons. First, this provided us with the whole waveform instead of just the captured samples, and thus the exact zero crossing for algorithm error evaluation. Second, it allowed us to generate a larger amount of simulated signals. Pulses were modeled as

$$s(t) = \begin{cases} 0 & , t < 0 \\ A \cdot t^2 e^{-t/\tau_{sh}} & , t \geq 0 \end{cases} \tag{28}$$

as proposed in [22]. The shaping constant τ_{sh} was considered to be uniformly distributed in $[1, 1.5]$, which corresponds approximately to the filter used for acquisition of the real signals. The CFD signals

$$y(t) = s(t-\tau) - \alpha \cdot s(t) \tag{29}$$

were generated with fixed parameters $\tau = 4$ and $\alpha = 0.5$; amplitudes A were chosen so that $\max|y(t)|$ would be distributed uniformly on $[0.2, 0.95]$. The signals were then sampled as $y_n = y(n-\delta)$ with a delay δ that was uniformly distributed on $[0, 1]$ in order to model arbitrary sampling phase, and quantized with $F = 12$ bit precision (i.e. with ulp $= 2^{-11}$).

We simulated 10^7 different pulses. For each one, the true zero crossing t_0 of the signal was obtained from the continuous signal $y(t)$. All 9 proposed bisection methods (linear, and both types of spline for N between 2 and 5) were then applied to the samples y_n to get the estimated zero crossings $t^\star$ with a precision of $M = 10$ bits, using the fixed point precisions Q given in Table II; they correspond to the optimum values (26) for each case. An example of the simulated pulses is pictured in Fig. 6 (a), together with the 6-node parabolically terminated spline interpolation around the zero-crossing interval.

The estimation error $|t_0 - t^\star|$ was evaluated, and the mean and maximum errors obtained for each method are shown in Table II; for comparison, the precision is around 10^{-3}. The mean error is reduced by roughly one third when using cubic instead of linear interpolation, with both types of spline yielding equivalent results, and no significative change is observed when increasing N beyond 3. Fig. 6 (a) includes a close-up of the actual zero crossing and the linear, 4-node and 6-node interpolation, and shows how the interpolated zero crossings get closer to the real one as N increases.

TABLE II
ALGORITHM PERFORMANCE ON SIMULATED SIGNALS

Type	N	Q	Mean error	Max. error	Reg. bound
Linear			$4.08 \cdot 10^{-2}$	$1.82 \cdot 10^{-1}$	43.5 %
Natural	2	9	$3.01 \cdot 10^{-2}$	$1.29 \cdot 10^{-1}$	11.8 %
	3	5	$2.65 \cdot 10^{-2}$	$1.10 \cdot 10^{-1}$	11.7 %
	4	1	$2.67 \cdot 10^{-2}$	$1.09 \cdot 10^{-1}$	11.7 %
	5	−3	$2.65 \cdot 10^{-2}$	$1.08 \cdot 10^{-1}$	11.7 %
Parabolically terminated	2	13	$3.03 \cdot 10^{-2}$	$1.37 \cdot 10^{-1}$	11.6 %
	3	10	$2.72 \cdot 10^{-2}$	$1.14 \cdot 10^{-1}$	11.7 %
	4	5	$2.66 \cdot 10^{-2}$	$1.09 \cdot 10^{-1}$	11.7 %
	5	3	$2.66 \cdot 10^{-2}$	$1.08 \cdot 10^{-1}$	11.7 %

TABLE III
ALGORITHM PERFORMANCE ON CAPTURED SIGNALS

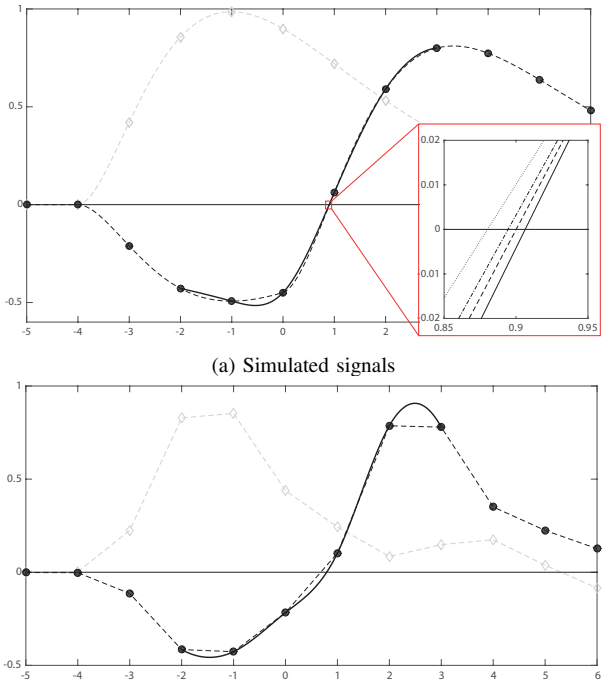
Type	N	FWHM coincidence resolution	Reg. bound
Linear		3.46 ns	39.8 %
Natural	2	3.25 ns	10.3 %
	3	3.17 ns	10.1 %
	4	3.17 ns	10.1 %
	5	3.18 ns	10.1 %
Parabolically terminated	2	3.28 ns	10.3 %
	3	3.19 ns	10.2 %
	4	3.17 ns	10.1 %
	5	3.18 ns	10.1 %

For each real stored signal, a digital CFD signal was obtained using (29) with the same parameters $\tau = 4$ and $\alpha = 0.5$; an example is pictured in Fig. 6 (b) that shows a similar behaviour to the simulated pulses. For each of the proposed algorithms, the zero crossing timestamps were estimated for each timing signal, their pairwise differences were histogrammed, and a gaussian fit was used to estimate the coincidence resolution. The results are summarized in Table III and reveal a similar pattern as observed in the simulated signals, although the performance improvement is lower due to the presence of noise and distortion in the samples.

The maximum values stored in registers are also shown in Tables II and III as a fraction of the theoretical bound, i.e. $\max |F_n(x_n)| / 2$ for linear and $\max |G_n(x_n)| / 8D$ for cubic interpolation. No overflow occurred, and in fact the observed extrema are significantly lower than the theoretical bounds, so that register widths may safely be shortened.

V. HARDWARE IMPLEMENTATION

We synthesized all of the proposed variations of the cubic interpolation circuit on FPGA technology, i.e. for natural and parabolically terminated splines, $2 \leq N \leq 5$, and using both FIR filter structures (direct and customized), in order to analyze the trade-offs involved in varying the amount of interpolation nodes. The linear bisection circuit was also synthesized for reference, as well as the standalone zero crossing estimators. The HDL code for each circuit was simulated first in order to check its correctness of operation. A Stratix IV EP4SGX110 device was chosen for implementation, the same one that is present in the DAQ system employed for acquisition



(a) Simulated signals

(b) Digitized signals

Fig. 6. Illustration of the method on simulated and real timing signals. Each plot includes the original detector signal (gray, dashed), the CFD signal (black, dashed), and the 6-node spline interpolation (black, solid). For the simulated signals, a close-up of the zero crossing is shown that includes, from left to right: linear interpolation (dotted), 4-node spline (dot-dashed), 6-node spline (dashed) and CFD signal (solid). All splines are parabolically terminated.

of the waveforms used in Section IV [9]. In all cases, we collected the area occupation and the delay of the critical register to register path in the placed and routed circuit. The results are summarized in Table IV; the area occupation is specified in terms of Adaptive Logic Modules (ALMs), the basic building blocks of the Stratix IV that consist of two LUTs, two registers, and additional adder and chain logic. All ALMs involved in the circuits are considered, even if they are only partially populated.

In general, the circuits become larger and slower when N increases as the amount and size of the filter coefficients grow, and are always smaller for parabolically terminated splines. The customized filter structure yields vastly improved performance in the FPGA implementations compared to the direct structure. The main reason is that the FPGA architecture is optimized for two-input adders such as those found in the custom filters, rather than the combinational adder trees that are necessary for multiple operand addition in the direct filter. Pipelining the zero crossing estimator decreases its critical path from 6.30 ns to 4.36 ns but doubles its latency so it results in an overall slower circuit that is only needed if a particularly high operating frequency is mandatory. The critical path is located in the FIR filter in most cases; hence, the figures corresponding to the pipelined version have been omitted.

Like most modern FPGAs, the Stratix IV hosts a number of DSP blocks containing 18×18 bit hardware multipliers. Circuits were generated with and without allowing the use of these blocks in order to evaluate their impact. Using the

TABLE IV
IMPLEMENTATION RESULTS ON FPGA

Type	N	w/ mults Area (ALMs)	w/ mults Mults (DSPs)	w/ mults Delay (ns)	w/o mults Area (ALMs)	w/o mults Delay (ns)
Linear (ZC only)		27	0	3.13		
Linear (full)		32	0	4.45		
Cubic (ZC only)		55	0	6.30		
Direct FIR filter						
Natural	2	160	0	9.45		
	3	236	1	13.22	295	13.76
	4	344	2	15.02	500	14.95
	5	452	2	15.13	729	14.95
Parabolically terminated	2	121	0	7.99		
	3	216	0	11.99		
	4	305	2	15.15	445	15.03
	5	471	2	15.32	580	15.29
Custom FIR filter						
Natural	2	161	0	6.65		
	3	202	1	6.86	227	7.36
	4	254	2	8.96	369	8.91
	5	297	2	8.08	510	9.94
Parabolically terminated	2	133	0	6.81		
	3	186	0	5.73		
	4	201	2	7.52	341	8.01
	5	276	2	9.75	431	8.45

dedicated hardware multipliers results in large area gains but does not have a significant impact on circuit speed. However, we remark that no attempt was made to optimize the FIR filters by fine tuning the DSP block configuration options.

VI. CONCLUSIONS AND DISCUSSION

A method has been presented for real-time estimation of the zero crossing locations of sampled signals consisting in two steps: interpolation by a cubic spline, and iterative estimation of its root. The interpolation step requires only two small FIR filters, and a custom filter structure has been proposed for it that benefits from symmetry properties. The zero location algorithm only requires two additions per iteration and lends itself to a very efficient hardware implementation. Optimal number representations have been determined that make the precision error negligible. The exposition has focused on FPGA implementation using fixed-point representation, but the algorithm can also improve the speed of zero estimation in other settings such as DSP or software since it does not require divisions or multiplications by variable factors.

Different algorithm configurations have been studied. A preliminary evaluation on simulated and real data suggests that the choice between natural and parabolically terminated splines does not have a significant impact and that there is no real gain from using more than 6 nodes for interpolation, so this amount should be used since the circuits become less efficient as the number of nodes increases. Note that this value has also been used without justification in previous work such as [5]. The results also suggest that this method might improve time resolution by a setting-dependent factor between 10 % and 30 % with respect to linear interpolation. A more comprehensive evaluation of this method and other tuning options, such as the variation of CFD parameters, in an experimental setup is currently planned.

APPENDIX
PROOFS OF THE THEORETICAL RESULTS

Theorem 1: *For natural and parabolically terminated splines, there is a $4 \times 2N$ matrix $\mathbf{M}$, which depends only on N, such that $\mathbf{c} = \mathbf{M}\mathbf{y}$. Moreover, there is an integer D such that $D\mathbf{M}$ consists of integer elements.*

Proof: Let the spline s consist of the polynomials $p_k : [0,1] \to \mathbb{R}$ that interpolate y at the interval $I_k = [k, k+1]$, i.e. $s(t) = p_k(t-k)$, for $-(N-1) \le k \le N-1$. Denote $y'_k = s'(k)$ and

$$\mathbf{y}' = \begin{bmatrix} y'_{-(N-1)} & y'_{-(N-2)} & \cdots & y'_N \end{bmatrix}^T. \tag{30}$$

Notice that p_k can be expressed in terms of its values and those of its first derivative at its end points as

$$\begin{aligned} p_k(t) = {} & (2y_k - 2y_{k+1} + y'_k + y'_{k+1})t^3 \\ & + (-3y_k + 3y_{k+1} - 2y'_k - y'_{k+1})t^2 + y'_k t + y_k. \end{aligned} \tag{31}$$

In particular, for $k = 0$ the relationship $\mathbf{c} = \mathbf{U}\mathbf{y} + \mathbf{V}\mathbf{y}'$ is obtained with the $4 \times 2N$ matrices $\mathbf{U}$ and $\mathbf{V}$ given by (10). Thus, it suffices to show that $\mathbf{y}' = \mathbf{E}\mathbf{y}$ for a $2N \times 2N$ matrix $\mathbf{E}$.

The continuity of s'' implies coincidence of the second derivatives at all interpolation nodes except the first and last ones, i.e. that $p''_{k-1}(1) = p''_k(0)$ and therefore

$$y'_{k+1} + 4y'_k + y'_{k-1} = 3y_{k+1} - 3y_{k-1} \tag{32}$$

for $-(N-2) \le k \le N-1$, by substitution into (31). Two conditions remain to be imposed. In the case of natural splines, $s'' = 0$ at the extreme nodes, resulting in

$$y'_{-(N-2)} + 2y'_{-(N-1)} = 3y_{-(N-2)} - 3y_{-(N-1)} \tag{33}$$

$$2y'_N + y'_{N-1} = 3y_N - 3y_{N-1}. \tag{34}$$

These two equations, together with (32), can be written in matrix form as $\mathbf{A}\mathbf{y}' = \mathbf{B}\mathbf{y}$, where $\mathbf{A}$ and $\mathbf{B}$ are the $2N \times 2N$ matrices (9) with parameters $\omega = 2$ and $\zeta = 3$. Hence we finally obtain

$$\mathbf{M} = \mathbf{U} + \mathbf{V}\mathbf{E} = \mathbf{U} + \mathbf{V}\mathbf{A}^{-1}\mathbf{B} \tag{35}$$

where $\mathbf{E} = \mathbf{A}^{-1}\mathbf{B}$ and the expressions of matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{U}$ and $\mathbf{V}$ are given in (9) and (10).

Similarly, for parabolically terminated splines, the cubic coefficient in (31) is equated to zero at the extreme intervals, resulting in

$$y'_{-(N-1)} + y'_{-(N-2)} = 2y_{-(N-2)} - 2y_{-(N-1)} \tag{36}$$

$$y'_{N-1} + y'_N = 2y_N - 2y_{N-1} \tag{37}$$

which result in the same equations $\mathbf{A}\mathbf{y}' = \mathbf{B}\mathbf{y}$ but with parameters $\omega = 1$ and $\zeta = 2$. The same formula (35) is thus obtained for $\mathbf{M}$.

Finally, notice that taking $D = \det \mathbf{A} \in \mathbb{Z}$ results in $D\mathbf{M}$ having only integer elements because it is obtained as addition and product of integer matrices, since the elements of $D\mathbf{A}^{-1}$ are minors of $\mathbf{A}$ which is itself integer. □

Theorem 2: *For natural and parabolically terminated splines,* $\mathbf{k}$ *is even and* $\mathbf{l}$ *is odd and they have the form* (20) *and* (21)*, where* $\boldsymbol{\alpha}$ *is an integer vector,* β *and* γ *are integers, and* A_K *and* A_L *are integers divided by powers of two.*

Proof: Let us generalize the notation in the text as follows: for a $m \times n$ matrix $\mathbf{X}$, denote by $\mathbf{X}^\times$ the $m \times n$ matrix that is obtained by reversing the order of its rows and columns, i.e. such that its (i,j)-th element is $x^\times_{ij} = x_{m+1-i,n+1-j}$, and let us call a matrix $\mathbf{X}$ $\times$*-symmetric* or $\times$*-antisymmetric* if $\mathbf{X}^\times$ is equal to $\mathbf{X}$ or $-\mathbf{X}$, respectively. Notice that both types of symmetry are closed under addition and multiplication by scalars. Notice also that if $\mathbf{X}$ is $m \times n$, $\mathbf{Y}$ is $n \times p$ and $\mathbf{Z} = \mathbf{XY}$, then

$$\begin{aligned} z^\times_{ij} = z_{m+1-i,p+1-j} &= \sum_{k=1}^{n} x_{m+1-i,k} \cdot y_{k,p+1-j} \\ &= \sum_{k=1}^{n} x_{m+1-i,n+1-k} \cdot y_{n+1-k,p+1-j} = \sum_{k=1}^{n} x^\times_{ik} y^\times_{kj} \end{aligned} \quad (38)$$

so that $(\mathbf{XY})^\times = \mathbf{X}^\times \mathbf{Y}^\times$. In particular, if $\mathbf{X}$ is square and invertible then $\mathbf{I} = \mathbf{I}^\times = (\mathbf{X}\mathbf{X}^{-1})^\times = \mathbf{X}^\times (\mathbf{X}^{-1})^\times$ and so $(\mathbf{X}^{-1})^\times = (\mathbf{X}^\times)^{-1}$; hence, both types of symmetry are also preserved by matrix inversion.

Since $\mathbf{A}$ is $\times$-symmetric and $\mathbf{B}$ is $\times$-antisymmetric, $\mathbf{E} = \mathbf{A}^{-1}\mathbf{B}$ is $\times$-antisymmetric. Hence, if $\mathbf{e}_N$ and $\mathbf{e}_{N+1}$ are the N-th and $(N+1)$-th rows of $\mathbf{E}$ then $\mathbf{e}_{N+1} = -\mathbf{e}^\times_N$ so that $\mathbf{e}_N + \mathbf{e}_{N+1}$ is odd and $\mathbf{e}_N - \mathbf{e}_{N+1}$ is even. Now, (10) and (35) imply that

$$\mathbf{k} = -\tfrac{1}{2} D\mathbf{m}_2 - \tfrac{3}{4} D\mathbf{m}_3 = \tfrac{1}{4} D(\mathbf{e}_N - \mathbf{e}_{N+1}) \quad (39)$$

is even, whereas

$$\begin{aligned} \mathbf{l} &= -\tfrac{3}{8} D\mathbf{m}_3 \\ &= \begin{bmatrix} \mathbf{z} & -\tfrac{3}{4}D & \tfrac{3}{4}D & \mathbf{z} \end{bmatrix} - \tfrac{3}{8} D(\mathbf{e}_N + \mathbf{e}_{N+1}) \end{aligned} \quad (40)$$

is a sum of odd vectors and therefore odd itself.

For the second part of the theorem, we will use the following elementary fact: *If* $\mathbf{X}$ *and* $\mathbf{Y}$ *are square matrices then*

$$\det \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \cdot & \mathbf{Y} \end{bmatrix} = \det \mathbf{X} \cdot \det \mathbf{Y} \quad (41)$$

where the dot represents arbitrary elements. This follows immediately from

$$\begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Z} & \mathbf{Y} \end{bmatrix} = \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Z} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y} \end{bmatrix} \quad (42)$$

where $\mathbf{I}$ are identity matrices of appropriate size. We will also use the auxiliary $n \times n$ matrices

$$\mathbf{G}_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 4 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 4 & 1 \end{bmatrix}, \mathbf{H}_n = \begin{bmatrix} 4 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 \\ 0 & 0 & 0 & \cdots & 1 & \omega \end{bmatrix} \quad (43)$$

and denote $r_n = \det \mathbf{H}_{n-1} / \det \mathbf{H}_n$; note that $\det \mathbf{G}_n = 1$.

Let $\mathbf{Q} = \mathbf{A}^{-1}$, and express the first N elements in its two central rows in terms of minors of $\mathbf{A}$. They take the form

$$\begin{aligned} q_{N,m} &= \frac{(-1)^{N+m}}{\det \mathbf{A}} \det \begin{bmatrix} \mathbf{H}^\times_{m-1} & \mathbf{0} & \mathbf{0} \\ \cdot & \mathbf{G}_{N-m} & \mathbf{0} \\ \cdot & \cdot & \mathbf{H}_N \end{bmatrix} \\ &= (-1)^{N+m} \frac{\det \mathbf{H}^\times_{m-1} \cdot \det \mathbf{H}_N}{\det \mathbf{A}} \end{aligned} \quad (44)$$

and

$$\begin{aligned} q_{N+1,m} &= \frac{(-1)^{N+m+1}}{\det \mathbf{A}} \det \begin{bmatrix} \mathbf{H}^\times_{m-1} & \mathbf{0} & \mathbf{0} \\ \cdot & \mathbf{G}_{N-m+1} & \mathbf{0} \\ \cdot & \cdot & \mathbf{H}_{N-1} \end{bmatrix} \\ &= (-1)^{N+m+1} \frac{\det \mathbf{H}^\times_{m-1} \cdot \det \mathbf{H}_{N-1}}{\det \mathbf{A}} \end{aligned} \quad (45)$$

for $m \le N$, where (41) has been used. It follows that $q_{N+1,m}/q_{N,m} = -r_N$ does not depend on m, and thus the left halves of rows N and $N+1$ of $\mathbf{A}^{-1}$ are proportional. Since $\mathbf{B}$ is tridiagonal, the last N elements in its first $N-1$ columns are all zero, and it follows that $e_{N+1,m}/e_{N,m} = -r_N$ for $m < N$, i.e. the first $N-1$ elements in $\mathbf{e}_N$ and $\mathbf{e}_{N+1}$ are proportional. By (40) and (39), so are the first $N-1$ elements in $\mathbf{k}$ and $\mathbf{l}$, as was to be proved. The fact that the elements are integers except possibly for a negative power of two follows immediately from (18), (19) and Theorem 1. □

Theorem 3: *The absolute error in the evaluation of* g *is bounded by* $1/2^{Q-1}$*, and the algorithm converges to* $\hat{t}^\star$ *such that* $\left|g(\hat{t}^\star)\right| \le 1/2^{Q-1}$.

Proof: Let $\hat{G}_n$, $\hat{K}_n$, $\hat{L}_n$ denote the values of G_n, K_n, L_n estimated by the algorithm implementation. Then $\hat{L}_n$ is L_n truncated to a multiple of 2^{-Q} and thus $\left|\hat{L}_n - L_n\right| \le 1/2^Q$ for all n. Similarly, $\left|\hat{K}_0 - K_0\right| \le 1/2^Q$. We now prove that $\left|\hat{K}_n - K_n\right| \le 1/2^{Q-1}$ by induction: assume it to be true for $n-1$. Notice that (16) implies

$$\hat{K}_n = \frac{1}{2}\hat{K}_{n-1} \pm \frac{1}{2}\hat{L}_{n-1} - \epsilon_n \quad (46)$$

where ϵ_n represents the LSB of the sum which is discarded when shifting one bit to the right to divide by two; thus, it is either 0 or $1/2^{Q+1}$. Subtracting (46) and (16) yields

$$\begin{aligned} \left|\hat{K}_n - K_n\right| &\le \frac{1}{2}\left|\hat{K}_{n-1} - K_{n-1}\right| + \frac{1}{2}\left|\hat{L}_{n-1} - L_{n-1}\right| + |\epsilon_n| \\ &\le \frac{1}{2} \cdot \frac{1}{2^{Q-1}} + \frac{1}{2} \cdot \frac{1}{2^Q} + \frac{1}{2^{Q+1}} = \frac{1}{2^{Q-1}} \end{aligned} \quad (47)$$

as was to be shown.

Let ε_n denote the maximum evaluation error of G_n i.e.

$$\begin{aligned} \varepsilon_n = \max \Big\{ &\left|\hat{G}_n(a_n) - G_n(a_n)\right|, \\ &\left|\hat{G}_n(b_n) - G_n(b_n)\right| \Big\}. \end{aligned} \quad (48)$$

For $n = 0$, G_0 is just a truncated value of g so $\varepsilon_0 \le 1/2^Q$. For higher n, one of a_n, b_n is equal to μ_n and the other one retains its value from the previous iteration, so

$$\varepsilon_n \le \max \left\{ 2\varepsilon_{n-1}, \left|\hat{G}_n(\mu_n) - G_n(\mu_n)\right| \right\} \quad (49)$$

while (13) implies

$$\begin{aligned}\left|\hat{G}_n(\mu_n) - G_n(\mu_n)\right| &\le \left|\hat{G}_{n-1}(a_{n-1}) - G_{n-1}(a_{n-1})\right| \\ &+ \left|\hat{G}_{n-1}(b_{n-1}) - G_{n-1}(b_{n-1})\right| \\ &+ \left|\hat{K}_{n-1} - K_{n-1}\right|. \end{aligned} \tag{50}$$

In particular, for $n = 1$

$$\left|\hat{G}_1(\mu_1) - G_1(\mu_1)\right| \le \varepsilon_0 + \varepsilon_0 + \left|\hat{K}_0 - K_0\right| \le \frac{3}{2^Q} \tag{51}$$

whereas $2\varepsilon_0 \le 1/2^{Q-1}$, so (49) implies $\varepsilon_1 \le 3/2^Q$.

We now prove that

$$\varepsilon_n \le \frac{1}{2^Q}(2^{n+1} - 1) \tag{52}$$

for all n by induction. This has already been shown for $n = 0$ and $n = 1$. Let $n \ge 2$ and assume that (52) holds for $n-1$ and $n-2$. Notice that one of a_{n-1}, b_{n-1} is equal to μ_{n-1} while the other one holds its value from the $(n-2)$-th iteration; for the former, the evaluation error is bounded by ε_{n-1}, but for the latter, the stronger bound $2\varepsilon_{n-2}$ applies. Then (50) yields

$$\begin{aligned}\left|\hat{G}_n(\mu_n) - G_n(\mu_n)\right| &\le \varepsilon_{n-1} + 2\varepsilon_{n-2} + \frac{1}{2^{Q-1}} \\ &\le \frac{1}{2^Q}[(2^n - 1) + 2(2^{n-1} - 1) + 2] \\ &= \frac{1}{2^Q}(2^{n+1} - 1). \end{aligned} \tag{53}$$

On the other hand

$$2\varepsilon_{n-1} \le 2 \cdot \frac{1}{2^Q}(2^n - 1) < \frac{1}{2^Q}(2^{n+1} - 1). \tag{54}$$

Equations (49), (53) and (54) show that (52) holds for n and complete its proof. Finally, since $g(\mu_n) = 2^{-n}G_n(\mu_n)$, the absolute error in its evaluation is bounded by $2^{-n}\varepsilon_n < 1/2^{Q-1}$ for all n, and the first part of the theorem is proved.

For the second part, notice that the algorithm always yields intervals $[a_n, b_n] \subset [a_{n-1}, b_{n-1}]$ such that $\operatorname{sign}\hat{g}(a_n) \ne \operatorname{sign}\hat{g}(b_n)$. If evaluation errors never cause $\hat{g}(\mu_n)$ to have the wrong sign, then subintervals are always selected correctly and the algorithm converges to $t^\star$. Otherwise, $\operatorname{sign}\hat{g}(\mu_m) \ne \operatorname{sign} g(\mu_m)$ for some iteration m and $t^\star$ lies outside of $[a_n, b_n]$ for all $n \ge m$. Even in this case, the algorithm still converges to some $\hat{t}^\star = \lim a_n = \lim b_n$ because a_n is non-decreasing, b_n is non-increasing, and $b_n - a_n = 2^{-n} \to 0$. Since $t^\star$ is the only root of g in $[0, 1]$, $\operatorname{sign} g(a_n) = \operatorname{sign} g(b_n)$ must hold for all $n \ge m$, so one of $g(a_n)$, $g(b_n)$ undergoes a change of sign when evaluated; hence, its absolute value cannot be higher than $1/2^{Q-1}$. The continuity of g then implies that $\left|g(\hat{t}^\star)\right| \le 1/2^{Q-1}$. □

Theorem 4: *If g is monotonic in $[0, 1]$, then $|G_n(a_n)|$, $|G_n(b_n)|$ and $|G_n(\mu_n)|$ are all bounded by $8D$ for all n.*

Proof: The proof is based on the following classical result by Bernstein (Theorem 15.5.3 in [23]): *Let p be a polynomial of odd degree n that is monotonic on $[-1, 1]$. Then*

$$|p'(t)| \le \left(\frac{n+1}{2}\right)^2 \cdot \max_{x\in[-1,1]} |p(x)| \tag{55}$$

for all $t \in [-1, 1]$. In our case, $n = 3$ and the polynomial is rescaled to $[0, 1]$ so the derivative is doubled and $g'(t) \le 8\max|g| \le 8D$ for all $t \in [0, 1]$. Now, if x_n is one of a_n, b_n or μ_n, then $|x_n - t^\star| \le 2^{-n}$ and so

$$\begin{aligned}|G_n(x_n)| &= 2^n\,|g(x_n)| = 2^n\,|g(x_n) - g(t^\star)| \\ &= 2^n \left|\int_{t^\star}^{x_n} g'(t)\,dt\right| \\ &\le 2^n\,|x_n - t^\star| \cdot \max|g'| \le 8D. \quad \square \end{aligned} \tag{56}$$

REFERENCES

[1] G. F. Knoll, *Radiation detection and measurement*, 3rd ed. John Wiley & Sons, 2000, pp. 659–679.

[2] T. J. Paulus, "Timing electronics and fast timing methods with scintillation detectors," *IEEE Trans. Nucl. Sci.*, vol. 32, no. 3, pp. 1242–1249, Jun. 1985.

[3] D. A. Gedcke and W. J. McDonald, "A constant fraction of pulse height trigger for optimum time resolution," *Nucl. Instr. Meth. A*, vol. 55, pp. 377–380, 1967.

[4] J. M. Monzó, R. Esteve, C. W. Lerche, N. Ferrando, J. Toledo, R. J. Aliaga, V. Herrero, and F. J. Mora, "Digital signal processing techniques to improve time resolution in Positron Emission Tomography," *IEEE Trans. Nucl. Sci.*, vol. 58, no. 4, pp. 1613–1620, Aug. 2011.

[5] V. Modamio, J. J. Valiente-Dobón, G. Jaworski, T. Hüyük, A. Triossi, J. Egea, A. Di Nitto, P.-A. Söderström, J. Agramunt Ros, G. de Angelis, G. de France, M. N. Erduran, S. Ertürk, A. Gadea, V. González, J. Kownacki, M. Moszynski, J. Nyberg, M. Palacz, E. Sanchís, and R.Wadsworth, "Digital pulse-timing technique for the neutron detector array NEDA," *Nucl. Instr. Meth. A*, vol. 775, pp. 71–76, Mar. 2015.

[6] M. A. Nelson, B. D. Rooney, D. R. Dinwiddie, and G. S. Brunson, "Analysis of digital timing methods with BaF_2 scintillators," *Nucl. Instr. Meth. A*, vol. 505, pp. 324–327, 2003.

[7] E. Delagnes, "What is the theoretical time precision achievable using a dCFD algorithm?" arXiv:1606.05541, 2016.

[8] P. Guerra, J. E. Ortuño, G. Kontaxakis, M. J. Ledesma-Carbayo, J. J. Vaquero, M. Desco, and A. Santos, "Real-time digital timing in positron emission tomography," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 5, pp. 2531–2540, Oct. 2008.

[9] R. J. Aliaga, V. Herrero-Bosch, J. M. Monzo, A. Ros, R. Gadea-Girones, and R. J. Colom, "Evaluation of a modular PET system architecture with synchronization over data links," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 1, pp. 88–98, Feb. 2014.

[10] L. Bardelli, G. Poggi, M. Bini, G. Pasquali, and N. Taccetti, "Time measurements by means of digital sampling techniques: a study case of 100 ps FWHM time resolution with a 100 MSample/s, 12 bit digitizer," *Nucl. Instr. Meth. A*, vol. 521, pp. 480–492, 2004.

[11] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 26, no. 6, pp. 508–517, Dec. 1978.

[12] M. Unser, "Splines: a perfect fit for signal and image processing," *IEEE Signal Process. Mag.*, vol. 16, no. 6, pp. 22–38, Nov. 1999.

[13] Y. Huang and Y. Ding, "Cubic B-spline approximation in scalers: efficient design and implementation," in *11th IEEE Singapore International Conference on Communication Systems (ICCS)*, Nov. 2008, pp. 342–346.

[14] N. Brekke, D. Röhrich, K. Ullaland, and R. Gruner, "Trigger performance simulation of a high speed ADC-based TOF-PET read-out system," *IEEE Trans. Nucl. Sci.*, vol. 59, no. 5, pp. 1910–1914, Oct. 2012.

[15] H. Semmaoui, M.-A. Tétrault, R. Lecomte, and R. Fontaine, "Signal deconvolution concept combined with cubic spline interpolation to improve timing with phoswitch PET detectors," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 3, pp. 581–587, Jun. 2009.

[16] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. A. K. Peters, 2002.

[17] R. L. Burden and J. D. Faires, *Numerical analysis*, 8th ed. Thomson, 2005, pp. 46–52 and 137–164.

[18] D. Lee, R. C. C. Cheung, W. Luk, and J. D. Villasenor, "Hardware implementation trade-offs of polynomial approximations and interpolations," *IEEE Trans. Comput.*, vol. 57, no. 5, pp. 686–701, May 2008.

[19] J. F. Epperson, "On the Runge example," *American Mathematical Monthly*, vol. 94, no. 4, pp. 329–341, Apr. 1987.

[20] C. A. Hall and W. W. Meyer, "Optimal error bounds for cubic spline interpolation," *Journal of Approximation Theory*, vol. 16, no. 2, pp. 105–122, Feb. 1976.
[21] J. G. Proakis and D. G. Manolakis, *Digital signal processing: principles, algorithms and applications*, 4th ed. Prentice Hall, 2007, pp. 565–601.
[22] J. M. Monzó, R. J. Aliaga, V. Herrero, J. D. Martínez, F. Mateo, A. Sebastiá, F. J. Mora, J. M. Benlloch, and N. Pavón, "Accurate simulation testbench for nuclear imaging systems," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 1, pp. 421–428, Feb. 2008.
[23] Q. I. Rahman and G. Schmeisser, *Analytic theory of polynomials*. Oxford University Press, 2002.

Final version available at http://dx.doi.org/10.1109/TNS.2017.2721103

This preprint includes an appendix with the proofs of all theoretical results that are stated throughout the text. Please note that the appendix is not included in the final version of the paper.