ARL-TR-10269 ● JAN 2026

On Using Cubic in Analytical Approach to Continuous-Time Signals

by Vinod K. Mishra

NOTICES

Disclaimers

ARL-TR-10269 ● JAN 2026

On Using Cubic in Analytical Approach to Continuous-Time Signals

Vinod K. Mishra
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
January 2026	Technical Report	10/01/2025	11/15/2025

4. TITLE AND SUBTITLE
On Using Cubic in Analytical Approach to Continuous-Time Signals

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Vinod K. Mishra

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-NC Aberdeen Proving Ground, MD 21005	ARL-TR-10269

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Signals (e.g., communication, radar, electrocardiogram, and so on) are generally analyzed using discrete time methods. Recently, advantages of continuous-time methods for machine learning have been observed. This approach is nonuniform in time and uses sampling based on level-crossing. In this report, we present an analytical model for the latter approach, which is also known as a level-crossing-based digital sampling process.

15. SUBJECT TERMS
Network, Cyber and Computational Sciences, cubic splines, digital signal processing, continuous-time DSP, level-crossing, discrete-time DSP, aliased sine

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	31

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Vinod K. Mishra	(520) 691-5761

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Signals in the tactical arena are of many kinds and vary in their characteristics across a wide range, e.g., communication, sensing, radar signals, and so on. Signal processing approaches applicable to these signals use either discrete or continuous-time (CT) sampling. The approach based on CT is a relatively new one and offers distinct advantages in digital signal processing (DSP) and analysis of physiological signals, such as electrocardiogram (ECG).[1–7] Earlier, an analytical approach to calculate the level crossing of these signals[8] was presented. In the current work, we extend that approach to many cubic curves representing the digital signals.

2. CT DSP

The Discrete Time DSP (DT-DSP) is the standard approach to signal processing. It uses ideas of Nyquist rate, sinc (short for cardinal sine) functions, etc. The other approach is CT-DSP,[9–13] which is a relatively new subfield of signal processing. It has time domain properties like analog signal processing (asynchronous, no clock) with benefits of DSP without the limitations of conventional sampling. It has many advantages over the conventional DT-DSP. CT-DSP advantages include the following:

1) The input CT signal is quantized when it exactly equals a voltage level, resulting in no inherent quantization error. Power savings due to this can be as high as 10 times in some cases.
2) The sampling frequency of the input CT signal is proportional to its slope, e.g., a 2.5-s ECG dataset requires only 225 samples compared to 1250 samples for conventional DSP. Fewer samples result in less data processing and lower energy.
3) It does not use uniform time sampling, so it has no frequency aliasing.
4) It uses adaptive sampling, so an input signal is only captured when it crosses a threshold level. For signals whose important amplitudes are known as a priori, this can lead to significant power savings.
5) The input signal is captured at the exact time and amplitude at which it crosses a threshold level. This accuracy allows the potential for the following:
   a) high signal to interference and noise distortion (SINAD) reconstruction of level-crossing sampled (LCS) signals. LCS is a sequence of Dirac-delta functions exactly representing the time and amplitude of all detected level crossings, and

b) higher SINAD values than conventional methods for level-crossing delta (LCD) signals as it counterbalances their harmonic distortion. LCD is a step function, consisting of adjoining rectangular pulses.

The time accuracy of the sampling benefits control systems and allows accurate detection of the transient signals. This is accomplished by giving information at the time of a transient, instead of waiting for the next clock tick. Both CT-DSP and analog signal processing are asynchronous (no clock), but the former has nonuniform time steps.

3. Level-Crossing and a CT Signal

CT-DSP has discrete amplitude and CT, and it emphasizes accuracy timing over that in amplitude. It is characterized by equivalent times, t_k, when the input signal exactly equals a discrete amplitude and there is no quantization error compared to DSP. It is nonlinear, so the signal reconstruction must be done before any processing. CT sample point is a vector given as follows.

1) The *k*-th time stamp and amplitude level, (t_k, a_k) or
2) a time difference and an amplitude level $(t_k - t_{k-1}, a_k)$. The time steps are fixed for the DT-DSP but are nonuniform for the CT-DSP. Input signal vectors in such a system can be characterized in either of the two ways:
   - $(t_k,\ a_k)$ where t_k= *k*-th level crossing timestamp, or
   - $(\Delta t_k,\ a_k)$ where Δt_k= $t_{k+1} - t_k$.= time difference between the *k+1*-th and *k*-th level crossing timestamps, and $a_k = n_k \Delta V$= the *k*-th amplitude level (n_k = an integer, and ΔV = the distance between voltage levels).
3) For CT, $a_k(t_k)$ = the amplitude level for level crossing at time t_k, $x_{CT}(t_k) = a_k(t_k)$.

   For DT, $t(k)t_k = kT$, $x_{DT}(k) = x(kT)$, k is an integer, and T is the time period.

The following relation connects a DT signal $x_d[n]$ to a corresponding CT signal $x_d(t)$.

$$x_d(t) = \sum_n x_d[n] p(\tau_0 + n\Delta t), t = \tau_0 + n\Delta t$$

Here τ_0 is a constant, *n* is an integer, and Δt is the time step. The *p*-function is an analytical function and depends on the nature and properties of the signal. For example, raised cosine function is used for band-limited signals. Regular level-crossing values in a CT signal correspond to nonuniform time values in $x_d(t)$.

4. An Analytical Approach to the Level-Crossing-Based Sampling

4.1 Signal Represented as Two Cubic Functions with Equal Slopes

Earlier in the above approach,[8] the slopes of the segments were not considered separately. Now we impose the conditions that slopes of the segments are equal at the intersection.

4.1.1 First and Second Segments

The first cubic and its slope are given as:

$$s = a_1 t^3 + b_1 t^2 + c_1 t + d_1$$

Let (t_1, s_1), (t_2, s_2), (t_3, s_3), and (t_4, s_4) be the four points on the first curve. The four cubic equations are solved to obtain four unknowns (a_1, b_1, c_1, d_1), as shown earlier.

The slope of the first cubic is given as

$$\frac{ds}{dt} = s' = 3a_1 t^2 + 2b_1 t + c_1 = k_1$$

Similarly, the second cubic and its slope are

$$s = a_2 t^3 + b_2 t^2 + c_2 t + d_2$$

$$\frac{ds}{dt} = s' = 3a_2 t^2 + 2b_2 t + c_2$$

The point (s_4, t_4) is common to both curves.

The continuity condition is captured by requiring that at the meeting point t_4:

a) The values of the two cubic segments are equal.

$$s_4 = a_2 t_4^3 + b_2 t_4^2 + c_2 t_4 + d_2$$

b) The slopes of the two cubic segments are equal.

$$k_1 = 3a_2 t_4^2 + 2b_2 t_4 + c_2$$

So, equating the value and the slope at the meeting point reduces the number of unknowns to two for the second cubic. The unknown coefficients $(a_2, b_2$ of the second cubic) are found by solving the following matrix equation applied to the subsequent two points (s_4, t_4) and (s_4, t_4).

$$\begin{bmatrix} t_5 + 2t_4 & 1 \\ t_6 + 2t_4 & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} \{(s_5 - s_4) - k_1(t_5 - t_4)\}/(t_5 - t_4)^2 \\ \{(s_6 - s_4) - k_1(t_6 - t_4)\}/(t_6 - t_4)^2 \end{bmatrix}$$

The other two coefficients are found from:

$$d_2 = s_4 - (a_2 t_4^3 + b_2 t_4^2 + c_2 t_4)$$

$$c_2 = k_1 - 3a_2 t_4^2 - 2b_2 t_4$$

The four coefficients (a_2, b_2, c_2, d_2) define the second cubic curve. Appendix A gives details about the solutions of the cubic equations.

4.1.2 An Example

For the first cubic curve, we start with four points with coordinates given as:

(t_1, s_1) = (2.4,12), (t_2, s_2) = (2.6,13), (t_3, s_3) = (2.8,14), and (t_4, s_4) = (3,12.5).

This leads to the coefficients of the first cubic curve:

$$(a_1, b_1, c_1, d_1) = (-52.08,\ 406.25,\ -1049.17,\ 910)$$

with the slope as:

$$k_1 = 3a_1 t_4^2 + 2b_1 t_4 + c_1 = -17.92$$

For the second cubic curve, we start with three points with coordinates given as:

(t_4, s_4) = (3,12.5), (t_5, s_5) = (3.2,12), and (t_6, s_6) = (3.4,10).

The value and the slope at the meeting point (t_4, s_4) are equated, leading to the (a_2, b_2) coefficients of the second cubic curve. The other two coefficients (c_2, d_2) are derived using them. So finally, the coefficients are:

$$(a_2, b_2, c_2, d_2) = (-239.58,\ 2281.25,\ -7236.67,\ 7660)$$

Figure 1 shows the two blue and orange color curves with a common point at (t_4, s_4) = (3,12.5).

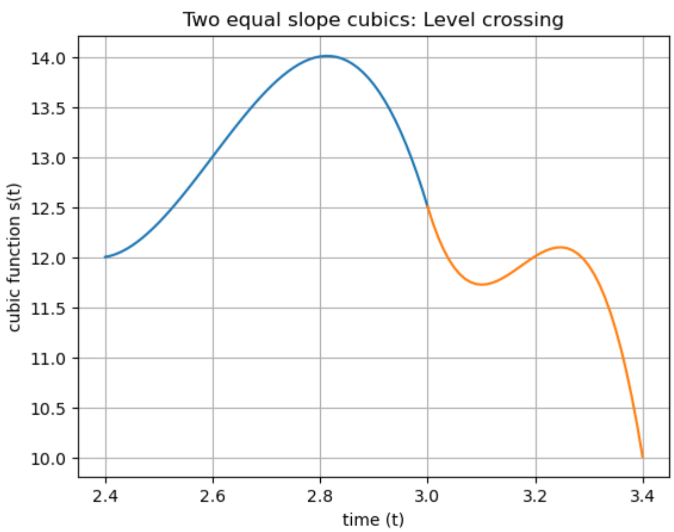

Figure 1. Two cubic curves with continuous value and slope at the meeting point.

As expected, six points are needed for this scheme to derive all the coefficients of the two cubic curves. Given any level-crossing values, the corresponding times can be obtained by using the cubic root expressions.

4.2 Continuation of the Scheme to the Third Cubic

Now we want to extend this approach to the third cubic after the first two.

4.2.1 Third Segment

Let us recall that the second cubic and its slope:

$$s = a_2 t^3 + b_2 t^2 + c_2 t + d_2$$

$$\frac{ds}{dt} = s' = 3a_2 t^2 + 2b_2 t + c_2$$

The previous process is now extended to the third cubic with the common point at (t_6, s_6). The continuity condition is captured by requiring that at the common point:

a) The values of the two cubic segments are equal.

$$s_6 = a_3 t_6^3 + b_3 t_6^2 + c_3 t_6 + d_3$$

b) The slopes of the two cubic segments are equal.

$$k_2 = 3a_3 t_6^2 + 2b_3 t_6 + c_3$$

The unknown coefficients (a_3, b_3 of the third cubic) are found by solving the following matrix equation applied to the subsequent two points (t_7, s_7) and (t_8, s_8).

$$\begin{bmatrix} t_7 + 2t_6 & 1 \\ t_8 + 2t_6 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} = \begin{bmatrix} \{(s_7 - s_6) - k_2(t_7 - t_6)\}/(t_7 - t_6)^2 \\ \{(s_8 - s_6) - k_2(t_8 - t_6)\}/(t_8 - t_6)^2 \end{bmatrix}$$

The other two coefficients are found from:

$$d_3 = s_6 - (a_3 t_6^3 + b_3 t_6^2 + c_3 t_6)$$

$$c_3 = k_2 - 3a_3 t_6^2 - 2b_3 t_6$$

The four coefficients (a_3, b_3, c_3, d_3) define the third cubic curve.

4.2.2 An Example

For the third cubic curve, we start with three points with coordinates given as:

(t_6, s_6) = (3.4,10), (t_7, s_7) = (3.6, 9), and (t_8, s_8) = (3.8, 11).

The value and the slope at the meeting point (t_6, s_6) are equated, leading to the (a_3, b_3) coefficients of the third cubic curve. The other two coefficients (c_3, d_3) are derived using them. So finally, the coefficients are:

(a_3, b_3, c_3, d_3) = (–255.208, 2793.75, –10179.792, 12356.25)

Table 1 shows the above results in a concise form.

Table 1. Parameters of the three cubic curves.

Cubic no.	(*t*,*s*) – values	Equal slope point	Cubic coefficients
First	(t_1, s_1) = (2.4,12), (t_2, s_2) = (2.6,13), (t_3, s_3) = (2.8,14), (t_4, s_4) = (3,12.5).	First cubic: no constraint	(a_1, b_1, c_1, d_1) = (–52.08, 406.25, –1049.17, 910)
Second	(t_4, s_4) = (3,12.5), (t_5, s_5) = (3.2,12), (t_6, s_6) = (3.4,10).	(t_4, s_4) = (3,12.5)	(a_2, b_2, c_2, d_2) = (−239.58, 2281.25, −7236.67, 7660)
Third	(t_6, s_6) = (3.4,10), (t_7, s_7) = (3.6, 9), (t_8, s_8) = (3.8, 11).	(t_6, s_6) = (3.4,10).	(a_3, b_3, c_3, d_3) = (−255.208, 2793.75, −10179.792, 12356.25)

Figure 2 shows the two orange and green color curves with a common point at (t_6, s_6) = (3.4,10).

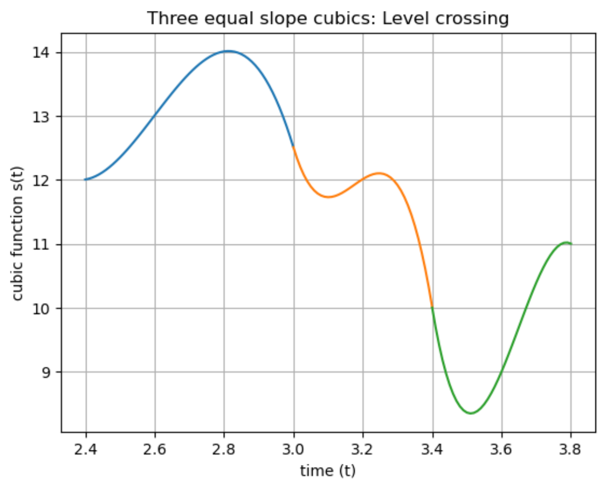

Figure 2. Three cubic curves with continuous value and slope at the meeting points.

This scheme can be repeated as many times as needed to find a CT envelope of a discrete time signal. The signal can be periodic or nonperiodic in time because of the analytic nature of the cubic curve. The final expression can be used to calculate the time value corresponding to any desired level crossing by finding the roots of a cubic equation.

4.2.3 Generalization of the Procedure

The steps of the process applicable to this approach are as follows:

- ***Step 1***: Start with the first four level-crossing points given as (t_i, s_i), i = 1,…,4. The time steps can be either uniform or arbitrary.
- ***Step 2***: Obtain the coefficients (a_1, b_1, c_1, d_1) of the cubic curve passing through these points by solving the four cubic equations passing through the four points given earlier.
- ***Step 3***: Calculate the point and slope values of the first cubic at the common point (t_4, s_4) using its coefficients.
- ***Step 4***: Identify the three points through which the second cubic will pass. The first such point (t_4, s_4) is the last point used for the first cubic. The next two points are (t_5, s_5) and (t_6, s_6).
- ***Step 5***: Use the results of Step 3 to reduce the number of unknowns for the second cubic coefficients to (a_2, b_2). The other coefficients (c_2, d_2) can be obtained from using the results of Step 3. Solve the equations to get (a_2, b_2) and afterwards (c_2, d_2).
- ***Step 6***: Repeat Steps 3–5 to get the third and other cubic curves as needed.

4.3 Signal Represented as Piecewise Aliased Sine Cardinal (*asinc*) Functions

The ***asinc*** functions are used in the reconstruction of the CT-signals from DT-signals. They are the basis of the digital-to-analog conversion of the signals. They are also known as LCS signals.

Given the fundamental frequency ω_0, the waveform period is

$$T_p = (2\,\pi/\omega_0)q,$$

and sample spacing is

$$T_s = \left(\frac{2\pi}{\omega_0}\right)\left(\frac{q}{N}\right) = \frac{T_p}{N}$$

such that $T_s < (\pi/\omega_{max})$, and q is an integer (usually 1).

The signal is represented as:

$$s(t) = \sum_{n=0}^{N-1} s_n h(t - nT_s).$$

Let $\alpha = \frac{\pi}{T_s}$, then for even N:

$$h(t) = \left(\frac{1}{N}\right)\frac{cos\,(\pi t/T_p)}{sin\,(\pi t/T_p)} sin\,(\pi t/T_s) = \left(\frac{1}{N}\right)\frac{cos\,(\alpha t/N)}{sin\,(\alpha t/N)} sin\,(\alpha t)$$

So, the signal becomes:

$$s(t) = \sum_{n=0}^{N-1} s_n \left(\frac{1}{N}\right) cos\,((\alpha t - n\pi)/N)\,\frac{1}{sin\,((\alpha t - n\pi)/N)}\, sin\,(\alpha t - n\pi),$$

Figure 3 presents the full *asinc* signal approximated by a cubic.

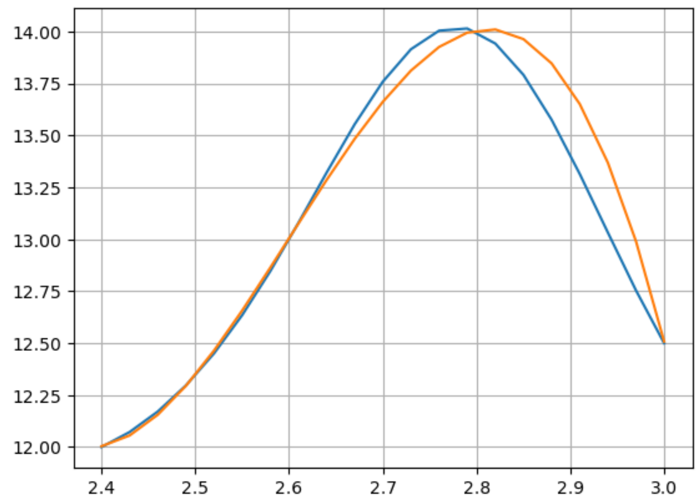

Figure 3. **The signal expressed in terms of *asinc* functions (N = 4, T_s = 0.2, orange curve) compared with the first cubic of Figure 1 (blue curve).**

The two curves follow one another for the initial one-third of the total time, diverge afterwards, and meet again at the end. This shows that the cubic is a good approximation to the theoretical curve.

Improving the Fitting Procedure

A better match can be found by refitting a new cubic to the *asinc* curve in the region where they differ. This becomes possible due to the analytic nature of this approach. Figure 4 presents the result of this approach.

The *asinc* curve with the t-range from $t = 2.4$ to $t = 3.4$ is divided into five segments. The cubics fitting them are found such that their slopes are equal at their meeting points. The algorithm presented earlier is used repeatedly for this purpose. Table 2 shows the results in a concise form.

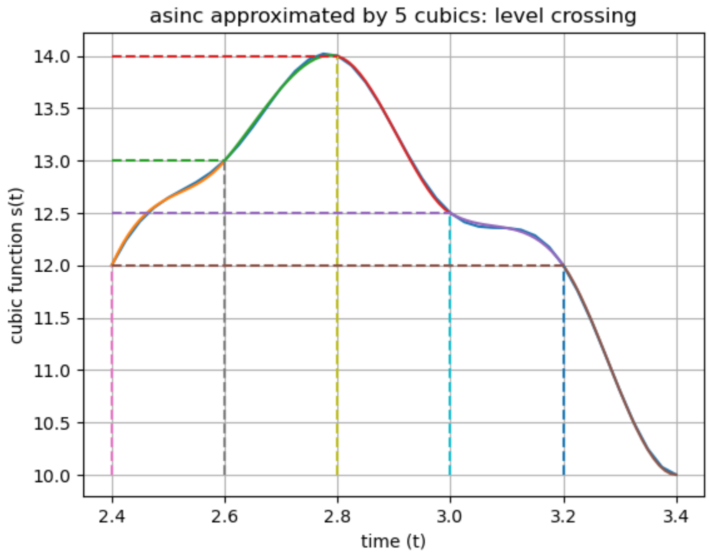

Figure 4. The larger range signal expressed in terms of *asinc* functions and its approximation by five cubic curves.

Table 2. Parameters of the cubic curves.

Segment no.	Range	(*t*,*s*) – values	Equal slope point	Cubic coefficients
First	(2.4,12)–(2.6,13)	(t_1, s_1) = (2.4,12), (t_2, s_2) = (2.467,12.527), (t_3, s_3) = (2.534,12.733), (t_4, s_4) = (2.6,13).	First cubic slope: no constraint	(a_1, b_1, c_1, d_1) = (215.267, 1628.949, 4111.328, 3448.296)
Second	(2.6,13)–(2.8,14)	(t_4, s_4) = (2.6,13), (t_5, s_5) = (2.7,13.299), (t_6, s_6) = (2.8,14).	(t_4, s_4) = (2.6,13)	(a_2, b_2, c_2, d_2) = (122.902, 976.176, -2577.256, 2275.039)
Third	(2.8,14)–(3,12.5)	(t_6, s_6) = (2.8,14), (t_7, s_7) = (2.9,13.299), (t_8, s_8) = (3,12.5).	(t_6, s_6) = (2.8,14)	(a_3, b_3, c_3, d_3) = (258.728, -2255.934, 6546.62, -6309.614)
Fourth	(3,12.5)–(3.2,12)	(t_8, s_8) = (3,12.5), (t_9, s_9) = (3.1,12.356), (t_{10}, s_{10}) = (3.2,12).	(t_8, s_8) = (3,12.5)	(a_4, b_4, c_4, d_4) = (147.4, 1360.22, 4184.848, 4304.864)
Fifth	(3.2,12)–(3.4,10)	(t_{10}, s_{10}) = (3.2,12), (t_{11}, s_{11}) = (3.3,10.807), (t_{12}, s_{12}) = (3.4,10).	(t_{10}, s_{10}) = (3.2,12)	(a_5, b_5, c_5, d_5) = (314.6, 3095.24, 10137.456, 11041.414)

The cubics provide a good fit overall. The algorithm can be changed if better fits are needed for smaller segments. Appendix B gives the Python programs (used in the Jupyter segment of Anaconda) for the fitting procedures.

5. Conclusion and Future Directions

In this work, the approach presented by Mishra[8] has been improved to give an analytical method to obtain a CT envelope to discrete signals. The examples focus on the time-periodic discrete signals, but the method can be very easily extended to the nonperiodic case. The method needs less computational resources than standard such algorithms.

In the future, we will include noise and distortion and extend the analytical approach presented here. Comparisons will be made with the traditional methods which are difficult to use analytically. We will also explore the computational cost savings from this approach in edge computing compared to the current ones.

6. References

1. Antony A, Paulson SR, Moni DJ. Asynchronous adaptive threshold level crossing ADC for wearable ECG sensors. J Med Syst. 2019;43(3):78. https://doi.org/10.1007/s10916-019-1186-8

2. Pae K, Mahbub I. A low-power asynchronous level crossing ADC designed in 180nm CMOS process for electrophysiological signal recording applications. In: 2022 IEEE 15th Dallas Circuit and System Conference (DCAS); 2022 June 17–19; Richardson, TX. IEEE; c2022. p. 1–5. https://doi:10.1109/DCAS53974.2022.9845498

3. Zanoli S, Ansaloni G, Teijeiro T, Atiena D. Mathematical model of parameters relevance in adaptive level-crossing sampling for electrocardiogram signals. arXiv; 2025 Jan 18. arXiv: arXiv:2501.10829. https://doi:10.48550/arXiv.2501.10829

4. Marisa T et al. Pseudo asynchronous level crossing ADC for ECG signal acquisition. IEEE Transactions on Biomedical Circuits and Systems. 2017;11(2):267–278. https://doi:10.1109/TBCAS.2016.2619858

5. Rincón F, Recas J, Khaled N, Atienza D. Development and evaluation of multilead wavelet-based ECG delineation algorithms for embedded wireless sensor nodes. IEEE Transactions on Information Technology in Biomedicine. 2011;15(6):854–863. https://doi:10.1109/TITB.2011.2163943

6. Qaisar SM, Hussain SF. Arrhythmia diagnosis by using level-crossing ECG sampling and sub-bands features extraction for mobile healthcare. Sensors. 2020;20(8):2252. https://doi.org/10.3390/s20082252

7. Goldberger A et al. PhysioBank, physiotoolkit, and physioNet: components of a new research resource for complex physiologic signals. Circulation [online]. 2000;101(23):e215–e220. https://doi.org/10.1161/01.cir.101.23.e215

8. Mishra VK. An analytical approach to level-crossing-based sampling. DEVCOM Army Research Laboratory (US): 2025 Aug. Report No.: ARL-TR-10197.

9. Tsividis Y. Continuous-time digital signal processing. Electron Lett. 2003;39(21):1551. https://doi.org/10.1049/el:20031015

10. Tsividis Y. Event-driven data acquisition and digital signal processing–a tutorial. IEEE Transactions on Circuits and Systems II: Express Briefs. 2010;57(8):577–581. https://doi.org/10.1109/TCSII.2010.2056012

11. Vezyrtzis C, Tsividis Y. Processing of signals using level-crossing sampling. In: 2009 IEEE International Symposium on Circuits and Systems (ISCAS); 2009 May 24–27; Taipei, Taiwan. IEEE; c2009. p. 2293–2296. https://doi:10.1109/ISCAS.2009.5118257

12. Kurchuk M. Signal encoding and digital signal processing in continuous time [dissertation]. Columbia University; 2011. https://doi.org/10.7916/D85T3SFB

13. Jungwirth P, Crowe WM. Continuous time digital signal processing and signal reconstruction. In: 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC); 2023 Mar 8–11; Las Vegas, NV. IEEE; c2023. p. 1205–1211. https://doi.org/10.1109/CCWC57344.2023.10099213

Appendix A. Roots of Cubic Equation

Let s_i denote the levels at time t_i and be represented as a cubic. Then the first few levels are

$$s_1 = at_1^3 + bt_1^2 + ct_1 + d$$

$$s_2 = at_2^3 + bt_2^2 + ct_2 + d$$

$$\ldots..$$

$$s_n = at_n^3 + bt_n^2 + ct_n + d$$

The first equation can be rewritten as

$$t_1^3 + \frac{b}{a}t_1^2 + \frac{c}{a}t_1 - \left(\frac{s_1 - d}{a}\right) = 0$$

It can be converted to a *depressed cubic* as

$$\left(t_1 + \frac{b}{3a}\right)^3 + p\left(t_1 + \frac{b}{3a}\right) - \left(\frac{s_1}{a} - q\right) = 0\,,$$

where

$$p = \frac{c}{a} - 3\left(\frac{b}{3a}\right)^2, q = \frac{d}{a} - \left(\frac{c}{a}\right)\left(\frac{b}{3a}\right) + 2\left(\frac{b}{3a}\right)^3.$$

The solution for t_1 corresponding to the first level-crossing becomes

$$t_1 = \frac{-b}{3a} + \left[\frac{1}{2}\left(\frac{s_1}{a} - q\right) + \Delta_1\right]^{\frac{1}{3}} + \left[\frac{1}{2}\left(\frac{s_1}{a} - q\right) - \Delta_1\right]^{\frac{1}{3}}, \Delta_1 = \sqrt{\frac{1}{4}\left(\frac{s_1}{a} - q\right)^2 + \left(\frac{p}{3}\right)^3}$$

It can be generalized to the t_n-value corresponding to the *n*-th level crossing.

$$t_n = \frac{-b}{3a} + \left[\frac{1}{2}\left(\frac{s_n}{a} - q\right) + \Delta_n\right]^{\frac{1}{3}} + \left[\frac{1}{2}\left(\frac{s_n}{a} - q\right) - \Delta_n\right]^{\frac{1}{3}}$$

$$\Delta_n = \sqrt{\frac{1}{4}\left(\frac{s_n}{a} - q\right)^2 + \left(\frac{p}{3}\right)^3}$$

The level-crossing amplitudes can have arbitrary values. For the equi-distant case we have:

$$s_n = s_1 + (n-1)\delta$$

Given a, b, c, d, s_1, and δ, all the level crossing time-values can be found.

For signals represented as cubic, the time-value t_n corresponding to a given level s_n is given by:

$$t_n = \frac{-b}{3a} + \left[\left(\frac{s_n}{2a} - \frac{q}{2}\right) + \Delta_n\right]^{\frac{1}{3}} + \left[\left(\frac{s_n}{2a} - \frac{q}{2}\right) - \Delta_n\right]^{\frac{1}{3}},$$

where

$$\Delta_n = \sqrt{\left(\frac{s_n}{2a} - \frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3},$$

For equi-distant levels: $s_n = s_1 + (n-1)\delta$

$$p = \left(\frac{c}{a}\right) - 3\left(\frac{b}{3a}\right)^2, q = \left(\frac{d}{a}\right) - \left(\frac{c}{a}\right)\left(\frac{b}{3a}\right) + 2\left(\frac{b}{3a}\right)^3$$

Appendix B. Jupyter Programs

This appendix appears in its original form, without editorial change.

 # Program (1)

 # linear equation solver: 4 equations for 4 coefficients of cubic eqns.: applied to first cubic

```
import numpy as np
# given t1, t2, t3, t4 values
#t1, t2, t3, t4 = 2.4, 2.6, 2.8, 3
t1, t2, t3, t4 = 2.4, 2.467, 2.534, 2.6
# Coefficient matrix (A)
t1sq, t2sq, t3sq, t4sq = t1*t1, t2*t2, t3*t3, t4*t4
A = np.array([[t1sq*t1, t1sq, t1, 1],
              [t2sq*t2, t2sq, t2, 1],
              [t3sq*t3, t3sq, t3, 1],
              [t4sq*t4, t4sq, t4, 1]])
# given s1, s2, s3, s4 values
#s1, s2, s3, s4 = 12, 13, 14, 12.5
s1, s2, s3, s4 = 12, 12.527, 12.733, 13
# Constant matrix (B) for RHS col[B] = Col[s1,s2,s3,s4]
B = np.array([s1, s2, s3, s4])
# Solve the system of equations: [A].Col[a1,b1,c1,d1] = Col[B]. Col[a1,b1,c1,d1] is unknown.
try:
    solution = np.linalg.solve(A, B)
    print("Solution:")
    print(f"a1 = {solution[0]}, b1 = {solution[1]}, c1 = {solution[2]}, d1 = {solution[3]}")
except np.linalg.LinAlgError:
    print("The system of equations has no unique solution.")
```

 # Program (2)

 # linear equation solver: coefficients of second cubic, slopes of two cubics equal at t4.

```
import numpy as np
import matplotlib.pyplot as plt
#---Input for first segment---
af, bf, cf, df = -122.902, 976.176, -2577.256, 2275.039
#----Input for the segement meeting the first---
ts1, ts2, ts3 = 2.6, 2.7, 2.8
ss1, ss2, ss3 = 13,13.693,14
# Coefficient matrix (A) for second cubic.
A = np.array([[ts2+2*ts1, 1],
            [ts3+2*ts1, 1]])
# k1 = slope of 1st cubic at the meeting point t4
ts1sq = ts1*ts1
kf = 3*af*ts1sq+2*bf*ts1+cf
print("slope =", kf)
# matrix (B) for RHS
trm1 = ((ss2-ss1)-kf*(ts2-ts1))/(ts2-ts1)**2
trm2 = ((ss3-ss1)-kf*(ts3-ts1))/(ts3-ts1)**2
B = np.array([trm1, trm2])

# Solve the system of equations: [A].Col[a,b] = Col[B]. a,b unknown.
try:
   solution = np.linalg.solve(A, B)
   print("Solutions for unknowns (a, b):")
   print(f"a = {solution[0]}, b = {solution[1]}")
except np.linalg.LinAlgError:
   print("The system of equations has no unique solution.")
cs = kf-2*solution[1]*ts1-3*solution[0]*ts1sq  #c2=k1-2*b2*t4-3*a2*t4sq
ds = ss1-kf*ts1+solution[1]*ts1sq+2*solution[0]*ts1sq*ts1  #d2=s4-
```

```
k1*t4+b2*t4sq+2*a2*t4sq*t4
print("Coefficients (cs, ds) derived from solutions:")
print("cs=",cs, "ds=", ds)
```

 # Program (3)

 # plots of two cubics with continuous slopes at their meeting points

```
import numpy as np
import matplotlib.pyplot as plt
# Naming the x-axis, y-axis and the whole graph
plt.title("Two equal slope cubics: Level crossing")
plt.xlabel('time (t)')
plt.ylabel('cubic function s(t)')
# case 1 1st cubic
# a1, b1, c1, d1 = -52.083, 406.25, -1049.167, 910
# tc1 = np.linspace(2.4, 3, 61)
# case 1 2nd cubic
# a2, b2, c2, d2 = -239.583, 2281.25, -7236.667, 7660
# tc2 = np.linspace(3, 3.4, 41)
# case 2 1st cubic
a1, b1, c1, d1 = 215.268, -1628.95, 4111.328, -3448.296
tc1 = np.linspace(2.4, 2.6, 41)
# case 2 2nd cubic
a2, b2, c2, d2 = -122.902, 976.176, -2577.256, 2275.039
tc2 = np.linspace(2.6, 2.8, 21)
# plots
tc1sq = tc1*tc1
sc1 = a1*tc1sq*tc1 + b1*tc1sq + c1*tc1 + d1
#print(tc1,sc1)
plt.plot(tc1, sc1)
tc2sq = tc2*tc2
sc2 = a2*tc2sq*tc2 + b2*tc2sq + c2*tc2 + d2
```

```
#print(tc2,sc2)
plt.plot(tc2, sc2)
# use plt.plot((x1, x2), (y1, y2), 'k-') to draw a line from the point (x1, y1) to the
point (x2, y2) in #color k
#plt.plot((2.4, 2.6), (13, 13), 'r-')
#plt.plot((2.6, 2.6), (10, 13), 'r-')
#-----
plt.grid(True)
plt.show()
```

 # Program (4)

 # *asinc* kernels expansions for signal

```
import numpy as np
import matplotlib.pyplot as plt
Ts, npi = 0.2, np.pi # Ts = sample spacing
alph = npi/Ts
N = 6
s1, s2, s3, s4, s5 ,s6 = 12, 13, 14, 12.5, 12, 10
# above s-values correspond to t1, t2, t3, t4, t5, t6 = 2.4, 2.6, 2.8, 3, 3.2, 3.4
#tk = np.linspace(2.4, 3.4, 41)
tk = np.linspace(2.4, 2.8, 21)

at = alph*tk  #t is time parameter from 1st cubic, at=alpha*t
hk0 = (1/N)*np.cos(at/N)*np.sin(at)/np.sin(at/N)
hk1 = (1/N)*np.cos((at-npi)/N)*np.sin(at-npi)/np.sin((at-npi)/N)
hk2 = (1/N)*np.cos((at-2*npi)/N)*np.sin(at-2*npi)/np.sin((at-2*npi)/N)
hk3 = (1/N)*np.cos((at-3*npi)/N)*np.sin(at-3*npi)/np.sin((at-3*npi)/N)
hk4 = (1/N)*np.cos((at-4*npi)/N)*np.sin(at-4*npi)/np.sin((at-4*npi)/N)
hk5 = (1/N)*np.cos((at-5*npi)/N)*np.sin(at-5*npi)/np.sin((at-5*npi)/N)
sk = s1*hk0+s2*hk1+s3*hk2+s4*hk3+s5*hk4+s6*hk5
print(tk,sk)
```

```
plt.plot(tk, sk)
#-----
# 1st cubic = a1*t1**3+b1*t1**2+c1*t1+d1=s
a1, b1, c1, d1 = 215.267, -1628.949, 4111.328, -3448.296
# above (a,b,c,d)-values correspond to t1, t2, t3, t4 = 2.4, 2.467, 2.534, 2.6,    s1,
s2, s3, s4 = 12, 12.527, 12.733, 13
t1 = np.linspace(2.4, 2.6, 21)
t1sq = t1*t1
sc1 = a1*t1sq*t1 + b1*t1sq + c1*t1 + d1
#print(tc1,sc1)
plt.plot(t1, sc1)
# 2nd cubic = a2*t2**3+b2*t2**2+c2*t2+d2=s
a2, b2, c2, d2 = -122.902, 976.1762, -2577.2557, 2275.0393
# above (a,b,c,d)-values correspond to t4, t5, t6 = 2.6, 2.7, 2.8     s4, s5, s6 =
13,13.693,14
# the slopes of 1st and 2nd cubic equal at (t4,s4) =(2.6,13)
t2 = np.linspace(2.6, 2.8, 21)
t2sq = t2*t2
sc2 = a2*t2sq*t2 + b2*t2sq + c2*t2 + d2
#print(tc2,sc2)
plt.plot(t2, sc2)
# 3rd cubic = a3*t3**3+b3*t3**2+c3*t3+d3=s
#t3 = np.linspace(2.8, 3, 21)
a3, b3, c3, d3 =
# above (a,b,c,d)-values correspond to t6, t7, t8 = 2.8, 2.9, 3     s6, s7, s8 = 14,
13.299,12.5
# the slopes of 2nd and 3rd cubic equal at (t6,s6) =(2.8,14)
t3 = np.linspace(2.8, 3, 21)
t3sq = t3*t3
sc3 = a3*t3sq*t2 + b3*t3sq + c3*t3 + d3
#print(tc3,sc3)
```

```
plt.plot(t3, sc3)
# 4th cubic = a4*t4**3+b4*t4**2+c4*t4+d4=s
#t4 = np.linspace(3, 3.2, 21)
a4, b4, c4, d4 =
# above (a,b,c,d)-values correspond to t8, t9, t10 = 3, 3.1, 3.2    s8, s9, s10 = 12.5,
12.356,12
# the slopes of 3rd and 4th cubic equal at (t8,s8) =(3,12.5)
t4 = np.linspace(3, 3.2, 21)
t4sq = t4*t4
sc4 = a4*t4sq*t4 + b4*t4sq + c4*t4 + d4
#print(tc4,sc4)
plt.plot(t4, sc4)
# 5th cubic = a5*t5**3+b5*t5**2+c5*t5+d5=s
#t5 = np.linspace(3.2, 3.4, 21)
a5, b5, c5, d5 =
# above (a,b,c,d)-values correspond to t10, t11, t12 = 3.2, 3.3, 3.4    s10, s11, s12 =
12, 10.807,10
# the slopes of 4th and 5th cubic equal at (t10,s10) =(3.2,12)
t5 = np.linspace(3.2, 3.4, 21)
t5sq = t5*t5
sc5 = a5*t5sq*t5 + b5*t5sq + c5*t5 + d5
#print(tc5,sc5)
plt.plot(t5, sc5)
#-----
plt.grid(True)
plt.show()
```

List of Symbols, Abbreviations, and Acronyms

asinc	Aliased Sine Cardinal
CT	Continuous Time
DSP	Digital Signal Processing
DT	Discrete Time
DT-DSP	Discrete Time DSP
ECG	Electrocardiogram
LCD	Level-Crossing Delta
LC-DSP	Level-Crossing-Based Digital Sampling Process
LCS	Level-Crossing Sampled
SINAD	Signal to Interference and Noise Distortion
sinc	Cardinal Sine

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB