Digital Signal Processing 23 (2013) 478–487

Contents lists available at SciVerse ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Tracking the best level set in a level-crossing analog-to-digital converter

Suleyman S. Kozat [a,*], Karen M. Guan [b], Andrew C. Singer [b]

[a] Bilkent University, Bilkent, 06800, Ankara, Turkey
[b] University of Illinois at Urbana Champaign, United States

ARTICLE INFO

Article history:
Available online 16 August 2012

Keywords:
Analog-to-digital conversion
Level-crossing
Adaptive level-crossing

ABSTRACT

In this paper, we investigate level-crossing (LC) analog-to-digital converters (ADC)s in a competitive algorithm framework. In particular, we study how the level sets of an LC ADC should be selected in order to track the dynamical changes in the analog signal for effective sampling. We introduce a sequential LC sampling algorithm asymptotically achieving the performance of the best LC sampling method which can choose both its LC sampling levels (from a large class of possible level sets) and the intervals (from the continuum of all possible intervals) that these levels are used based on observing the whole analog signal in hindsight. The results we introduce are guaranteed to hold in an individual signal manner without any stochastic assumptions on the underlying signal.

1. Introduction

In this paper, we investigate the level-crossing (LC) sampling method [1–9] in a competitive algorithm framework [2,10–14]. In particular, we study how the level sets of an LC sampler should be selected in order to track changes in an analog input signal. Here, we introduce a sequential LC sampling algorithm asymptotically achieving the performance of the best LC sampling method that can choose both its LC sampling levels (from a large class of possible level sets) and the intervals of time over which these levels are used based on observing the whole analog signal [14]. The results we introduce are guaranteed to hold in an individual signal manner without any stochastic assumptions [2,13–15].

In conventional and most common (Nyquist–Shannon) analog signal sampling, the underlying signal is uniformly sampled in time (at a frequency at least twice the maximum frequency content of the signal) and analog levels are subsequently quantized [16,17]. Nevertheless, it has been shown that sampling or quantization methods that incorporate information of the underlying signal in their framework may drastically improve efficiency and performance over conventional methods (for certain signals), e.g., using sparsity for compressive sensing [18]. In this sense, the level-crossing (LC) sampling method is akin to opportunistic methods and has been shown to yield more economical and effective sampling for certain signals [3–5,7–9,19–25]. In this approach, analog signals are compared to a set of fixed reference levels and samples are taken on the time axis, indicating the times at which the analog signal exceeded each of the associated reference levels. Hence, unlike the conventional sampling, where the time axis, i.e., the sampling times, are fixed and amplitudes are quantized, here the amplitude thresholds are fixed and crossing times of these thresholds by the analog signal are quantized. This threshold-based sampling is particularly suitable for processing *bursty* signals, where information is delivered in bursts, or temporally sparse regions, rather than in a constant stream [9]. Sampling by LC visibly mimics the behavior of such input signals such that when the input is bursty, LC samples also arrive in bursts. Unlike traditional sampling circuits that consume a constant amount of power even when there is no change in the input signal, the LC sampling offers higher instantaneous bandwidth/precision when sampling is performed [9,26]. However, we also note that LC ADC is indeed not a mainstream implementation, which renders it an exciting alternative to the traditional Nyquist ADCs. Just like the new compressive sensing samplers that take advantage of signal characteristics to lower sampling rate and hence data throughput, LC ADC is an event-driven implementation that also demonstrated significant compression capability when used to collect information on sparse signals. However, the resolution of (time) samples after the ADC step, fundamentally, is driven by some synchronous hardware, be it directly through a clock or indirectly through a token scheme. Resolving these limitations in hardware design will be a part of our future work.

The performance gain due to LC sampling heavily depends on the proper placement of threshold (or reference) levels [26]. The reference levels are in general selected as uniform [3] similar to uniform sampling in time [17]. However, instead of setting uniformly spaced threshold levels, one can choose the corresponding levels based on the statistics of the underlying signal. In such a "static" method, the best set of levels can be selected using different methods, such as the Lloyd–Max quantization [17], yielding a

* Corresponding author.
E-mail addresses: skozat@ku.edu.tr (S.S. Kozat), kguan@uiuc.edu (K.M. Guan), acsinger@illinois.edu (A.C. Singer).

1051-2004/$ – see front matter © 2012 Published by Elsevier Inc.
http://dx.doi.org/10.1016/j.dsp.2012.07.018

non-uniformly spaced level set that minimizes the mean-squared error (MSE) between the reconstructed signal and the original. We call these methods "static" since the levels are optimized based on certain statistics of the underlying signal and are fixed from the beginning, i.e., they do not change with time. We emphasize that such an approach requires certain probabilistic information *a priori*. Although such implementation is optimal in the sense of minimizing MSE (provided the *a priori* information), no quality of service (QoS) is *guaranteed* since good performance is attained on average, i.e., in the expectation sense typical of minimum MSE based algorithms [13,14]. Moreover, such an implementation is not adaptive to the input's statistical variations. Mismatch between the assumed (or estimated) and the actual statistics may lead to severe performance degradation. Although time adaptive extensions are possible [17], the same difficulties due to model mismatch and QoS problems persist [12,27]. Note that after the LC samples are provided, one can use different methods including [17,28,29] to reconstruct the original signal. Our results hold for different reconstruction methods provided that they are sequentially computable.

There exist other level selection methods that are more inline with signal tracking [7,8]. In these methods, signal tracking is accomplished by extensively increasing the sampling bandwidth of the ADC, say $\frac{1}{\tau}$, (which, depending on the actual design of the ADC, either refers to the circuit bandwidth where τ is the total delay of one conversion loop, or to the sampling frequency f_s) [26]. Typically only one LC can be acknowledged every τ seconds due to circuit limitations [7]. When the conversion delay following a LC is long with respect to the signal variation, additional LCs that occurred in this interval will be unaccounted for, eroding the reconstruction fidelity. In order to avoid such a scenario, either a circuit with larger bandwidth is used [7], or a flash-type ADC that can sample with a bank of comparators simultaneously is used [8]. We emphasize that the variable threshold comparators needed for LC sampling is extensively studied in [20], where a design of a time continuous variable voltage (threshold) comparator with low propagation delay dispersion and power consumption is presented for LC sampling. In [7] the input is sampled at full bandwidth to ensure any change between subsequent samples is bounded between adjacent levels, hence no more than one level can be crossed within an interval of τ. In [8] the specification for high circuit bandwidth is alleviated by feeding the input into multiple analog comparators simultaneously. Implementation differences aside, both designs [7] and [8] employ tracking circuits that rely on the assumption that if a signal is sampled fast enough, successive samples do not vary arbitrarily. The interval-by-interval search for the best quantizer dominates the computational complexity, and their good performance comes at the cost of large sampling bandwidth as well as high power consumption, rendering them unattractive in low-cost and low-power applications [2].

In this paper, instead of tracking the analog signal by oversampling or by making statistical assumptions, we employ a competitive algorithm framework and compete against a class of LC ADC algorithms that can select and optimize their level sets even based on observing the whole analog signal ahead of time. In this competition class, the underlying analog signal, say $y(t)$, of duration T seconds, is divided into arbitrary length segments, and for each segment a different level set is selected (from a large class of possible level sets) by minimizing the reconstruction error. Note that such selection can only be performed with a delay of T seconds (i.e., the whole signal must be seen in advance) and the optimization is over the set of all possible partitions of the real line. Here, we introduce an algorithm that sequentially observes the underlying analog signal and immediately selects the LC levels, however, asymptotically achieves the performance of the best algorithm in the competition class with the optimal partition of the real line and the reference levels. We emphasize that in similar lines, a competitive algorithm is introduced in [2] in order to achieve the performance of the best static LC sampling scheme without any statistical assumptions on the underlying analog signals. However, the corresponding algorithm as well as the setup significantly differs in here since in [2] the competition class is "static", i.e., does not change in time, which significantly simplifies the algorithm design and implementation.

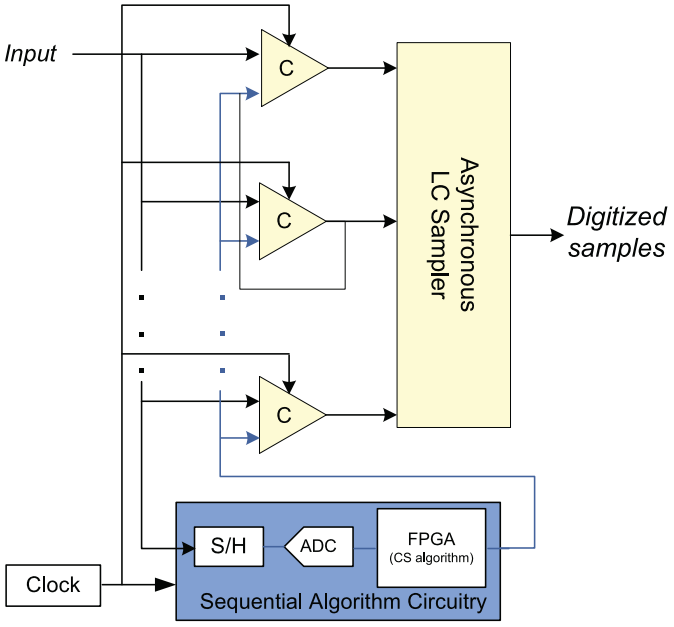

Fig. 1. An example LC ADC sampling framework. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

The organization of the paper is as follows. In Section 2, we provide a basic architecture for a LC ADC. In Section 4, we introduce a sequential algorithm that tracks the best level set in time, and provide a complete algorithmic description with the corresponding performance analysis. The performance of the introduced algorithm is simulated in Section 5 under different scenarios. The paper concludes with certain remarks in Section 6.

2. A basic LC ADC framework

In this section, we first present a basic architecture for LC sampling. We continue to introduce a class of LC sampling algorithms that can arbitrarily switch their threshold levels in time and, then, a sequential algorithm to track the best LC levels in time.

Here, we provide a conceptual LC ADC framework and note that various hardware implementations of asynchronous LC samplers are investigated in [7,8,30]. The conceptual LC ADC architecture studied in this paper is presented in Fig. 1. Consider a b-bit (with 2^b levels) flash-type ADC with an array of $K = 2^b - 1$ analog comparators that compare the analog input with corresponding reference levels. We assume that reference levels are implemented with a voltage divider (or with a digital to analog converter) and designed to be noise resistant so that fluctuations due to noise do not cause any chattering, hence, sampling is exact [26]. Suppose the input signal $y(t)$ is bounded such that $|y(t)| \leqslant A/2$, for a known $A > 0$. Although our derivations are generic with respect to how the reference levels are chosen, for notational simplicity, we assume that the levels of the LC ADC are uniformly spaced in the dynamic range with $\delta = \frac{A}{2^b}$ intervals. Let $\boldsymbol{l} = \{l_1, l_2, \ldots, l_K\}$ represent the set of reference levels used by the comparators with cardinality $|\boldsymbol{l}| = K$. Note that we reserve bold letters to represent sets (or vectors) and given a set (or a vector) $\boldsymbol{l}$, $|\boldsymbol{l}|$ represents the cardinality (or length). We emphasize that only a subset of this

complete set of reference levels is usually active, i.e., used for LC sampling, due to power limitations [7]. Given a subset of levels, the LC ADC compares the input $y(t)$ to this subset every τ seconds and records a level-crossing with l_k if the following comparison holds:

$$\big(y\big((n-1)\tau\big)-l_k\big)\big(y(n\tau)-l_k\big)<0. \tag{1}$$

Although the crossing occurs at a time $t=c_n$ in the interval $c_n\in[(n-1)\tau,n\tau)$, however, only the quantized value of this crossing time is recorded, i.e., $Q(c_n)=(n-1)\tau+\tau/2$ is recorded. We represent the LC sample acquired by the ADC as a quantized time and level pair, $(Q(c_n),\alpha_n)$, such that $t=c_n$, $y(c_n)=\alpha_n=l_k\in\boldsymbol{l}$.

Remark 2.1. Since $\alpha_n=l_k$ is in $\boldsymbol{l}$, it is known with perfect precision. This is the main difference between quantization of LC samples from that of uniform-in-time samples such that uniform-in-time samples are quantized in amplitude, while LC samples are quantized in time.

In the next section, we introduce the class of LC ADC algorithms that can switch their reference level sets in time. We compete against this class of algorithms and try to sequentially achieve the performance of the best in this class.

3. The class of LC ADCs that switch level sets in time

Since the power consumption of an ADC circuitry is dominated by the comparators [7], say, at most m of the K comparators with $m\ll K$ are turned *on* at any moment [26]. The asynchronous digital circuitry processes the output of the analog circuitry and outputs a sequence of bits corresponding to the LCs. How this encoding is carried out to maximize the coding efficiency is discussed in [11,12].

Let $\boldsymbol{L}$ denote the set of all possible m-level sets that can be activated from $\boldsymbol{l}$, i.e., $\boldsymbol{L}=\{\boldsymbol{L}_1,\dots,\boldsymbol{L}_{|\boldsymbol{L}|}\}$ with $|\boldsymbol{L}|=\binom{K}{m}$, where $\boldsymbol{L}_i=\{L_{i,1},\dots,L_{i,m}\}$, $L_{i,j}\in\boldsymbol{l}$. Given an m-level set $\boldsymbol{L}_i$, let $\{(Q(c_k),\alpha_k)\}_{k\in\mathbb{Z}^+}$ represents the LC samples produced by $\boldsymbol{L}_i$. The corresponding reconstructed signal at time t, starting from time 0, is represented by $\hat{y}(t,0,\boldsymbol{L}_i)$. For example, using a piecewise linear modeling (PWL) approximation scheme, given that $\{(Q(c_k),\alpha_k)\}_{k\in\mathbb{Z}^+}$ are the LC samples, the reconstruction is given by

$$\hat{y}(t,0,\boldsymbol{L}_i)=\sum_k\left\{\left[\left(\frac{\alpha_{k+1}-\alpha_k}{Q(c_{k+1})-Q(c_k)}\right)\big(t-Q(c_k)\big)+\alpha_k\right]\times\big[u\big(t-Q(c_{k+1})\big)-u\big(t-Q(c_k)\big)\big]\right\}, \tag{2}$$

for any interval T, where $u(t)$ is a unit step function, i.e., $u(t)=1$ when $t\geqslant 0$ and $u(t)=0$ otherwise. Note that our derivations are generic with respect to how the reconstruction is carried out as long as the reconstruction is causal, i.e., non-anticipating of the future, [17]. Then, the reconstruction error is given by

$$e(T,0,\boldsymbol{L}_i)=\int_{t=0}^{T}\big[y(t)-\hat{y}(t,0,\boldsymbol{L}_i)\big]^2\,dt, \tag{3}$$

over any T.

Instead of using a single level set $\boldsymbol{L}_i$ for all T (where T is arbitrary), we allow switching between m-level sets and define an "$(R,m,\boldsymbol{t}_R)$" level set sequence (or an LC sampling algorithm). In the class of $(R,m,\boldsymbol{t}_R)$ algorithms, the m-level set used by the LC ADC is updated R times over a period of T, at an arbitrary collection of instances represented by $\boldsymbol{t}_R=(t_1,\dots,t_R)$, such that $t_1<t_2<\cdots<t_R$, $\bigcup_{r=0}^{R}[t_r,t_{r+1})=[0,T)$, where we set $t_0=0$ and $t_{R+1}=T$ for notational simplicity. Here, $\boldsymbol{L}_{\boldsymbol{t}_R}$ denotes a sequence of m-level sets used over a period of T, over $R+1$ segments formed by R switches, i.e., $\boldsymbol{L}_{\boldsymbol{t}_R}=\{\boldsymbol{L}(0),\boldsymbol{L}(1),\dots,\boldsymbol{L}(R)\}$ such that each $\boldsymbol{L}(i)\in\boldsymbol{L}$ and $\boldsymbol{L}(i)$ is the m-level set used for the $(i+1)$th segment. One can arbitrarily choose both the switching times $\boldsymbol{t}_R$ and reference level set in each interval. An LC ADC that samples with $(R,m,\boldsymbol{t}_R)$ produces a set of samples, with a slight abuse of notation, $\{(Q(c_k),\alpha_k)\}_{k\in\mathbb{Z}^+}$. Using a PWL approximation scheme, the reconstruction is given by

$$\hat{y}\big(t,0,(R,m,\boldsymbol{t}_R)\big)=\sum_k\left\{\left[\left(\frac{\alpha_{k+1}-\alpha_k}{Q(c_{k+1})-Q(c_k)}\right)\big(t-Q(c_k)\big)+\alpha_k\right]\times\big[u\big(t-Q(c_{k+1})\big)-u\big(t-Q(c_k)\big)\big]\right\}. \tag{4}$$

The reconstruction error over an interval of T is given by

$$e\big(T,0,(R,m,\boldsymbol{t}_R)\big)=\int_{t=0}^{T}\big[y(t)-\hat{y}\big(t,0,(R,m,\boldsymbol{t}_R)\big)\big]^2\,dt \tag{5}$$

$$=\sum_{r=0}^{R}\int_{t_r}^{t_{r+1}}\big[y(t)-\hat{y}\big(t,t_r,\boldsymbol{L}(r)\big)\big]^2\,dt, \tag{6}$$

where $\hat{y}(t,t_r,\boldsymbol{L}(r))$ is the reconstructed signal using $\boldsymbol{L}(r)$ in the $(r+1)$th region, starting from time $t=t_r$. The switching algorithm with the best performance is one that minimizes the error function (5) by updating and selecting the most appropriate level sets at the right times tuned to $y(t)$. This $(R,m,\boldsymbol{t}_R)$ algorithm can be found by evaluating the following optimization problem:

$$\big(R^*,\boldsymbol{t}_R^*,(R,m,\boldsymbol{t}_R)^*\big)=\arg\inf_{R,\boldsymbol{t}_R,(R,m,\boldsymbol{t}_R)}e\big(T,\boldsymbol{t}_R,(R,m,\boldsymbol{t}_R)\big), \tag{7}$$

where $\boldsymbol{t}_R$ is any partition of $[0,T]$, for any $R\leqslant\lfloor\frac{T}{v}\rfloor$ for some small constant v, where $\lfloor\frac{T}{v}\rfloor$ is the largest integer smaller or equal to $\frac{T}{v}$. We limit the number of switches by $\lfloor\frac{T}{v}\rfloor$ for tractability and v is an algorithmic parameter explained in Section 4. This optimization provides for the best R^*, the best sequence of update times $\boldsymbol{t}_R{}^*$ over all possible partitions of $[0,T]$, and the most suitable $\boldsymbol{L}^*_{(R,m,\boldsymbol{t}_R)}$ from a set of $|\boldsymbol{L}|(|\boldsymbol{L}|-1)^R$, i.e., choose for the first segment the best $\boldsymbol{L}_i$ from the set $\boldsymbol{L}$ and continue selection for other segments subsequently.

Remark 3.1. Evaluating (7) however requires a delay of T seconds. In other words, the best $(R,m,\boldsymbol{t}_R)$ level sequence cannot be known *a priori* (at the start). Our goal is to achieve this "batch" performance sequentially.

In the next section, we construct a sequential algorithm, which has no knowledge of R, the switching times $\boldsymbol{t}_R$ or the selected m-level sets $\boldsymbol{L}_{\boldsymbol{t}_R}$ in each segment, but asymptotically performs as well as the algorithm that can tune all of these parameters by observing $y(t)$ in $[0,T]$ *a priori*.

4. Achieving the best batch performance sequentially

In this section, we introduce a randomized sequential algorithm that asymptotically achieves the performance of the best $(R,m,\boldsymbol{t}_R)$ scheme that could only be chosen in hindsight (after observing the whole signal $y(t)$ ahead of time). At fixed intervals, our algorithm randomly selects a level set and uses it to sample $y(t)$ until this level set is replaced by the next selection. The level set is chosen

from a class of possible level sets according to a probability mass function (PMF) generated by the cumulative performance of each level set in this class on $y(t)$ until that time.

4.1. A competitive sequential algorithm

The competitive sequential (CS) algorithm is a sequential algorithm that updates its m-level set at every v seconds based on certain PMF assigned to level sets, but has no knowledge of the optimal R, $\boldsymbol{t}_R$ or $(R, m, \boldsymbol{t}_R)$. We provide the pseudo-code of the CS algorithm

A Pseudo-Code of the Competitive Sequential Algorithm:

Inputs: η: learning rate, ϵ: algorithmic parameter, v: update interval for level sets, $y(t)$: analog signal to be sampled.
Outputs: $\hat{y}(t)$: reconstructed analog signal, $\boldsymbol{L}(n)$: level set used during $[nv, (n+1)v)$, $w(n,q,k)$: assigned weight to $\boldsymbol{L}_k$ at time $t = nv$.
Set of inputs η, ϵ, v, where $\eta > 0$, $\epsilon > 0$ and $v > 0$.

Initialize the reconstructed signal $\hat{y}(0) = 0$.
Initialize weights $w_0(1,1,k) = 1/|\boldsymbol{L}|$, $k = 1, \ldots, |\boldsymbol{L}|$.
Initialize $\hat{y}(0,0,\boldsymbol{L}_k) = 0$, $k = 1, \ldots, |\boldsymbol{L}|$.

for $n = 1 : \ldots,$
  for $k = 1 : |\boldsymbol{L}|$,
    % Update the weights associated with each level set $\boldsymbol{L}_k$ based on
    % performance.
    for $q = 1, \ldots, n-1$,

$$w(n,q,k) = \left(\frac{1/n^{1+\epsilon}}{\sum_{j=1}^{\infty}(1/j^{1+\epsilon}) - \sum_{j=1}^{n-1}(1/j^{1+\epsilon})}\right) w(n-1,q,k) \qquad \text{(line A)}$$
$$\times \exp\left(-\eta \int_{(n-1)v}^{nv} \left[y(t) - \hat{y}(t, qv, \boldsymbol{L}_k)\right]^2 dt\right),$$

    endfor

$$w(n,n,k) = \frac{1}{|\boldsymbol{L}|-1} \sum_{q=1}^{n-1} \left(\frac{\sum_{j=1}^{\infty}(1/j^{1+\epsilon}) - \sum_{j=1}^{n}(1/j^{1+\epsilon})}{\sum_{j=1}^{\infty}(1/j^{1+\epsilon}) - \sum_{j=1}^{n-1}(1/j^{1+\epsilon})}\right) w(n-1,q,k) \qquad \text{(line B)}$$
$$\times \sum_{j \neq k}^{|\boldsymbol{L}|} \exp\left(-\eta \int_{(n-1)v}^{nv} \left[x(t) - \hat{y}(t, (n-1)v, \boldsymbol{L}_j)\right]^2 dt\right).$$

  endfor
  % Select one $\hat{y}(t, qv, \boldsymbol{L}_k)$ according to the PMF given as follows.
  $\Pr[\hat{y}(t, qv, \boldsymbol{L}_k) \text{ is selected}] = \frac{w(n,q,k)}{\sum_{k=1}^{|L|}\sum_{h=1}^{n} w(n,h,k)}$, $k = 1, \ldots, |\boldsymbol{L}|$, $q = 1, \ldots, n$.
  % Use the selected set $\boldsymbol{L}(n) = \boldsymbol{L}_k$ to sample $x(t)$ in the interval $[nv, (n+1)v)$.
  % Form the reconstruction $\hat{y}(t)$ as: $\hat{y}(t) = \hat{y}(t, \boldsymbol{L}_{\mathrm{CS}}) = \hat{y}(t, qv, \boldsymbol{L}_k)$,
  % $t \in [nv, (n+1)v)$.
endfor

	Number of additions	Number of multiplications
The sequential algorithm	$2\|\boldsymbol{L}\|n$	$3\|\boldsymbol{L}\|n + \|\boldsymbol{L}\|$

Fig. 2. Number of additions and multiplications required for the introduced sequential algorithm at each decision time, i.e., $n = \lfloor \frac{T}{v} \rfloor$.

For the algorithmic parameters, we have $\tau \ll v$, i.e., the update period is much larger than the sampling period. At every update, the CS algorithm calculates certain "weights" which are subsequently used to construct a PMF over the class of all possible level sets. For each level set $\boldsymbol{L}_k$, $k = 1, \ldots, |\boldsymbol{L}|$, $w(n,q,k)$ in the pseudo-code denotes a weight generated at update $t = nv$, computed using level set $\boldsymbol{L}_k$ over a time interval $[qv, nv)$, where $q < n$. At each selection time, one of the level sets, $\boldsymbol{L}_k$, $k = 1, \ldots, |\boldsymbol{L}|$, is selected based on the weight $w(n,q,k)$ assigned to it as shown in the pseudo-code of the competitive sequential algorithm. Number of additions and multiplications required for the introduced sequential algorithm at each decision time is given in Fig. 2.

Our sequential algorithm works conceptually as follows. Instead of trying to estimate the best switching times and the best m-level sets for each segment, we implicitly implement all possible $(R, m, \boldsymbol{t}_R)$ algorithms, for all R, $\boldsymbol{t}_R$ and $\boldsymbol{L}_{\boldsymbol{t}_R}$. Clearly, this is a conceptual implementation since there exist a continuum of $\boldsymbol{t}_R$ and corresponding selection of m-level sets given R and $\boldsymbol{t}_R$. At each time $t = nv$, we compare the performance of each such $(R, m, \boldsymbol{t}_R)$ algorithm, for all R, $\boldsymbol{t}_R$ and $\boldsymbol{L}_{\boldsymbol{t}_R}$, and prefer the one with the best performance until that time (by assigning a larger probability mass to it). The weights $w(n,q,k)$ quantify the performance of these $(R, m, \boldsymbol{t}_R)$ algorithms. However, instead of storing a different weight for each possible $(R, m, \boldsymbol{t}_R)$ algorithm, we merge the weights of certain $(R, m, \boldsymbol{t}_R)$ algorithms and store cumulative weights $w(n,q,k)$. For example, $w(n,q,k)$ is the single weight assigned to all $(R, m, \boldsymbol{t}_R)$ algorithms that use the m-level set $\boldsymbol{L}_k$ in certain parts of the interval $t \in [qv, nv)$ to sample $y(t)$, for $q < n$, as explained in more detail in Section 4.2. Accordingly $w(n,n,k)$ is the initial weight assigned to the kth quantizer that is switched to at time $t = nv$ and will be used after $t = nv$. A flowchart of our sequential algorithm is given in Fig. 3.

Let $\boldsymbol{L}_{\mathrm{CS}}$ be a sequence of levels chosen by the CS algorithm up to time T and $\hat{y}(t, 0, \boldsymbol{L}_{\mathrm{CS}})$ be the reconstructed signal obtained by sampling $y(t)$ with the sequence of levels chosen by the CS algorithm. Then, we have the following theorem.

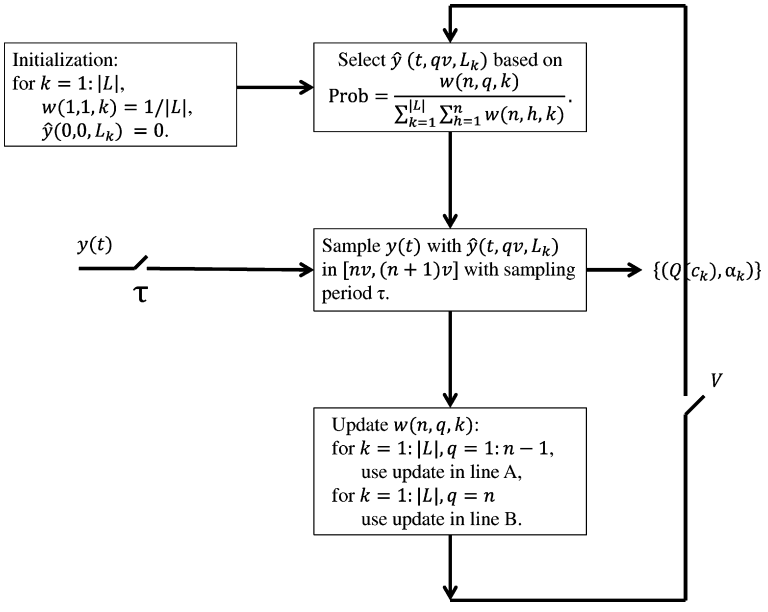

Fig. 3. A flowchart of our competitive sequential algorithm with the pseudo-code.

Theorem. *When applied to any bounded input $y(t)$, $|y(t)| \leqslant \frac{A}{2}$, and for any parameters $\eta > 0$, $\epsilon > 0$ and $v > 0$, a consistent reconstruction of the input signal using the CS algorithm given in the pseudo-code incurs*

$$\frac{1}{T}E\big[e(T,0,\boldsymbol{L}_{\mathrm{CS}})\big] \leqslant \frac{1}{T}e\big(T,0,(R,m,\boldsymbol{t}_R)^*\big) + \frac{R\ln T + (R+1)\ln|\boldsymbol{L}|}{\eta T} + \frac{\eta v A^4}{8} + O\left(\frac{R+1}{T}\right) + O\left(\frac{1}{\epsilon T}\right), \tag{8}$$

over any T and selecting η to minimize the upper bound yields

$$\frac{1}{T}E\big[e(T,0,\boldsymbol{L}_{\mathrm{CS}})\big] \leqslant \frac{1}{T}e\big(T,0,(R,m,\boldsymbol{t}_R)^*\big) + \times\sqrt{\frac{vA^4(R\ln T + (R+1)\ln|\boldsymbol{L}|)}{2T}} + O\left(\frac{R+1}{T}\right) + O\left(\frac{1}{\epsilon T}\right), \tag{9}$$

for any $R \leqslant \lfloor \frac{T}{v} \rfloor$, $\boldsymbol{t}_R$ or $\boldsymbol{L}_{\boldsymbol{t}_R}$ without using R, T, $\boldsymbol{t}_R$ or $(R,m,\boldsymbol{t}_R)$.

We point out that the expectations in the theorem are with respect to the PMF generated by the randomization in the pseudo-code such that there are no stochastic assumptions on $y(t)$, i.e., our results hold for any possible bounded analog signal $y(t)$. The theorem states that the normalized performance of the CS algorithm is asymptotically as good as the normalized performance of the best LC quantizer that can only be chosen in hindsight. We observe that the difference between the best algorithm and the sequential algorithm diminishes as T increases. We point out that for the optimized value of η in (9), one needs to know T and R ahead of time. This can be easily surpassed using a "doubling" method as in [31] by selecting the optimal algorithmic parameters over exponentially growing time intervals. We emphasize that the introduced algorithm should store $O(\lfloor t/v \rfloor)$ variables, i.e., weights shown in line A and line B of the pseudo-code, at each time t and should update these variables with $O(\lfloor t/v \rfloor)$ computational complexity over an ordinary LC quantizer as shown in line A and line B. Furthermore, the parameter η in line A and line B can be considered as the learning rate and can be selected accordingly to trade of between increased convergence speed versus reduced final MSE [32]. Note that if η is small, then the algorithm favors less the past performance. The parameter ϵ is used so that series $\sum_{j=1}^{\infty}(1/j^{1+\epsilon})$ is absolutely summable.

Remark 4.1. We emphasize that the CS algorithm does not depend on R or $\boldsymbol{t}_R$ and the performance bounds in the theorem hold for any R, $\boldsymbol{t}_R$ or $(R,m,\boldsymbol{t}_R)$. Clearly, the optimal $(R,m,\boldsymbol{t}_R)$ algorithm (that can only be chosen in hindsight) that minimizes the accumulated error should use maximum number of switchings, i.e., $R^* = \lfloor \frac{T}{v} \rfloor$. The regret bound in (9) reflects this such that when the competition class is more powerful, i.e., R is large, then the regret bound is large. When the competing algorithm is less powerful such that R is small, the bound is tighter. For both cases, the sequential algorithm is the same and does not depend on R.

Remark 4.2. While constructing the CS algorithm, the corresponding weights, $w(n,q,k)$, are calculated based on the analog signal $y(t)$ using analog integration. Naturally, $y(t)$ is not available (which is the reason for sampling) and analog integration could be carried out using a digital circuit [26]. To circumvent this problem, we approximate analog integrations with discrete summations as in [2]. The competitive sequential algorithm proposed by this work can easily be incorporated into a flash-type LC ADC, and its implementation is shown by the blue-colored box labeled "sequential algorithm circuitry" in Fig. 1. The sequential algorithm circuitry is a separate circuit element that works in parallel to the LC ADC. In order to generate a recommended level set, input is sampled with a low-rate ADC at full bandwidth (the sample and hold circuitry has full bandwidth of the input), followed by an FPGA that performs the needed digital computation to generate the PMF from which the next level set is selected and fed into the LC ADC. The sequential algorithm circuit is synchronized with rest of the circuitry by feeding off the same clock supplied to the rest of the circuitry. Since algorithm implementation relies only on a low-rate ADC and similarly low-rate FPGA, such circuitry would not require significant additional power. According to the technology trend [33], low-rate-medium-resolution ADC consumes only a few milliwatts of power. Power consumption of FPGA is driven by algorithm complexity but is nominally under one watt [33]. Future work on this subject will report hardware simulation results (performed in MATLAB SIMULINK with SPICE models to account for electronic device level characteristics), along with device part list and system power summary. Furthermore, it has been shown in [2] that the algorithm using the approximated sums with the proposed approaches can sufficiently approximate the competitive sequential algorithm such that the bounds in (9) hold with additional terms that diminish as T grows.

Remark 4.3. We emphasize that the algorithm in the pseudo-code requires storing the weights $w(n,k,q)$ with perfect precision. It has been shown that the corresponding weights can be adequately represented without increasing the corresponding bit rate in [11,12,27].

4.2. Proof of the theorem and construction of the CS algorithm

Proof of the theorem uses averaging ideas from [34] used in [14] for prediction, however, requires different formulation due to continuous time processing and the particular application. We mainly concentrate on these differences here. Given R and for an interval $[0,T)$, an infinite number of possible partitions of $[0,T)$ into $R+1$ segments can be defined, each represented by $(t_1,\ldots,t_R)$, $t_1 < t_2 < \cdots < t_R$, $t_i \in [0,T]$. However, we only allow the sequential algorithm to switch between m-level sets every v seconds, where for the notational simplicity, we take $v = M\tau$ for some integer M and τ is the sampling period of the LC ADC. Hence, given $y(t)$ and the corresponding interval $[0,T)$, there exist only $U \triangleq \lfloor T/v \rfloor$ possible switching points for our sequential algorithm. Furthermore, for presentation purposes, at the start of the proof, we assume $T = Uv$, for some integer U. At the end of the proof, we demonstrate that the additional error in the segment $[Uv,T)$, i.e., if T is not an integer multiple of v, will only contribute $O(1)$ to the total error. Additionally, at the start of the proof, we only allow $(R,m,\boldsymbol{t}_R)$ sequences to switch at only integer multiple of v and represent the switching points as $\tilde{\boldsymbol{t}}_R = (\tilde{t}_1,\ldots,\tilde{t}_R)$. We note that for a possible switching pattern $(t_1,\ldots,t_R)$, we can find the closest point for each t_i on $[0,T)$ and construct a $(\tilde{t}_1,\ldots,\tilde{t}_R)$ on $[0,T)$. We demonstrate that the error of choosing path $(\tilde{t}_1,\ldots,\tilde{t}_R)$, instead of $(t_1,\ldots,t_R)$, will be negligible for large T and sufficiently small switching intervals v.

At each time, that is an integer multiple of v, say $t = Uv$, there exists $\binom{U-1}{R}$ possible ways of choosing switching pattern $\tilde{\boldsymbol{t}}_R$, a total of 2^{U-1} possible switching patterns for all $R = 1,\ldots,U-1$ and corresponding $(R,m,\tilde{\boldsymbol{t}}_R)$ algorithms after selecting LC levels for each region. To each $(R,m,\tilde{\boldsymbol{t}}_R)$ algorithm, we assign a weight

$$\beta\big((R,m,\tilde{\boldsymbol{t}}_R)\big) \triangleq W\big(\tilde{\boldsymbol{t}}_R,(R,m,\tilde{\boldsymbol{t}}_R)\big)\exp\big(-\eta e\big(Uv,\tilde{\boldsymbol{t}}_R,(R,m,\tilde{\boldsymbol{t}}_R)\big)\big) \tag{10}$$

based on its performance on $y(t)$ up to $t = Uv$, where $W(\tilde{\boldsymbol{t}}_R, (R, m, \tilde{\boldsymbol{t}}_R))$'s are certain weights from [34] only used for proof purposes and are not required for the construction of the CS algorithm. We next construct a randomized algorithm, such that at time $t = Uv$ this algorithm selects any of the switching algorithms with probability $\beta((R, m, \tilde{\boldsymbol{t}}_R))/(\sum_{R,\boldsymbol{L}_{\tilde{t}_R}} \beta((R, m, \tilde{\boldsymbol{t}}_R)))$ and uses it in the interval $[Uv, (U+1)v)$ for sampling and reconstructing, say, $\hat{y}(t)$. Then, it can be shown that this brute force algorithm achieves [31]

$$\begin{aligned}
&\eta E\left[\int_0^{Uv} \big(y(t) - \hat{y}(t)\big)^2 dt\right] \\
&\leqslant \eta e\big(Uv, 0, (R, m, \boldsymbol{t}_R)\big) + \big(R + (R+1)\epsilon\big)\ln(U/R) \\
&\quad + \left((R+1)\ln\frac{1+\epsilon}{\epsilon} + \ln\epsilon\right) + (R+1)\ln\big(|\boldsymbol{L}|\big) + \frac{\eta^2 B^2}{8},
\end{aligned} \tag{11}$$

where B is an upper bound on the error in one segment, i.e., $B \triangleq \max_{i,a} \int_0^v (y(t) - \hat{y}(t, a, \boldsymbol{L}_i))^2 dt \leqslant A^2 v$, for any $\tilde{\boldsymbol{t}}_R$ and $(R, m, \boldsymbol{t}_R)$, $r = 1, \ldots, U-1$, using Hoeffding's inequality [35]. Hence, repeating this construction of $\hat{y}(t)$ at each time t yields the required universal algorithm. We next show that $\hat{y}(t, \boldsymbol{L}_{\mathrm{CS}})$ in the pseudo-code is equal to $\hat{y}(t)$ by construction. Note that at each time $t = Uv$, $\hat{y}(t)$ needs to construct all $(R, m, \boldsymbol{t}_R)$ algorithms and calculate the corresponding weights. However, instead of storing a different weight for each possible $(R, m, \boldsymbol{t}_R)$ algorithm, we merge the weights of $(R, m, \boldsymbol{t}_R)$ algorithms that use the same m-level set, say, $\boldsymbol{L}_k$ in $t \in [qv, nv)$ to sample $y(t)$, and store cumulative weights $w(n, q, k)$. It can also be shown as in [14] that the randomization over $w(n, q, k)$ and $\beta((R, m, \tilde{\boldsymbol{t}}_R))$ yields the same bound in (11) and the cumulative weights $w(n, q, k)$ can be calculated recursively using the updates in line A and line B of the pseudo-code. Hence, $\hat{y}(t, \boldsymbol{L}_{\mathrm{CS}})$ provided the same performance, i.e., competitive with respect to the switching class, as $\hat{y}(t)$.

The weight $w(n, q, k)$ is the summation of all weights $\beta(\cdot)$ assigned to all $(R, m, \boldsymbol{t}_R)$ algorithms that have the last switching time qv and use the kth algorithm in the last segment for sampling. This assigned weight is updated as follows [36]. We assign to each time point, $t = nv$, a distribution over the discrete time, z, $z \geqslant n$, for the probability that the next transition occurs just before time z, i.e., the probability that the next switching time will occur just before zv is

$$\pi(z) = \frac{1}{z^{1+\epsilon}}. \tag{12}$$

From (12), we observe that as time progresses, it is less likely to have a switching. The form (12) is shown to yield the bound in the theorem in [34]. Hence, the assigned probability of transition at time $n+1$ is (12) normalized by the infinite summation of $\pi(z)$, $z \geqslant n$, i.e.,

$$\left(\frac{1/n^{1+\epsilon}}{\sum_{j=1}^{\infty}(1/j^{1+\epsilon}) - \sum_{j=1}^{n-1}(1/j^{1+\epsilon})}\right). \tag{13}$$

Note that in (12), we have $\epsilon > 0$ in the exponent so that series $\sum_{j=1}^{\infty}(1/j^{1+\epsilon})$ is absolutely summable.

Up to this point, we proved that, the universal algorithm can approximate the reconstruction error of any switching LC quantizer when the transition instants are integer multiples of v. We next demonstrate that we can approximate any $e(T, 0, (R, m, \boldsymbol{t}_R))$ sufficiently well enough with $e(Uv, \tilde{\boldsymbol{t}}_R, (R, m, \tilde{\boldsymbol{t}}_R))$.

Given any $\boldsymbol{t}_R$, for each t_i, we can choose the closest $\tilde{t}_i$ and then construct the corresponding $\tilde{\boldsymbol{t}}_R$. The difference between any t_i and $\tilde{t}_i$ can be at most $\frac{v}{2}$. Hence, the difference between the error of using $\tilde{\boldsymbol{t}}_R$ with respect to $\boldsymbol{t}_R$ for each t_i to $\tilde{t}_i$ can be at most

$$\max_{i,j,a,b}\left|\int_0^{\frac{v}{2}} \big(y(t) - \hat{y}(t, a, \boldsymbol{L}_i)\big)^2 dt - \int_0^{\frac{v}{2}} \big(y(t) - \hat{y}(t, b, \boldsymbol{L}_j)\big)^2 dt\right|, \tag{14}$$

which is bounded by $A^2 v$. Since there are R transitions,

$$e\big(Uv, 0, (R, m, \boldsymbol{t}_R)\big) \leqslant e\big(Uv, 0, (R, m, \tilde{\boldsymbol{t}}_R)\big) + RA^2 v. \tag{15}$$

Since this is also true for the last segment of $[0, T)$, i.e., the interval $[Uv, T)$, we have

$$e\big(T, 0, (R, m, \boldsymbol{t}_R)\big) \leqslant e\big(Uv, 0, (R, m, \tilde{\boldsymbol{t}}_R)\big) + (R+1)A^2 v. \tag{16}$$

Combining these equations yields

$$\begin{aligned}
&E\left[\int_0^T \big[y(t) - \hat{y}(t, \boldsymbol{L}_{\mathrm{CS}})\big]^2 dt\right] \\
&\leqslant e\big(T, \boldsymbol{t}_R, (R, m, \boldsymbol{t}_R)\big) + (R+\epsilon)\frac{\ln(U)}{\eta} \\
&\quad + \frac{1}{\eta}\left(\ln(1+\epsilon) + R\ln\frac{1}{\epsilon}\right) + \frac{U\eta v^2 A^4}{8} + (R+1)A^2 v \qquad (17) \\
&= e\big(T, \boldsymbol{t}_R, (R, m, \boldsymbol{t}_R)\big) + \frac{(R\ln(T) + (R+1)\ln(|\boldsymbol{L}|))}{\eta} \\
&\quad + \frac{\eta T v A^4}{8} + O(R+1) + O\left(\frac{1}{\epsilon}\right). \qquad (18)
\end{aligned}$$

Selecting the η parameter to minimize the left hand side of (17) yields the bound in (9). This concludes the proof of the theorem. □

5. Numerical examples

In this section, we demonstrate the performance of the introduced algorithm under different scenarios. We note that the experiments carried out here are algorithmic simulations performed on MATLAB to emulate the LC sampling framework, rather than a simulation of hardware performance.

The first set of experiments include processing of speech signals, where the analog signals and analog integrations are mimicked using uniform sampled versions with sufficiently large sampling rate. Note that speech signals are selected for these experiments since voiced and unvoiced parts of speech signals clearly require different level sets for efficient sampling due to different characteristics in amplitude and shape. Here, the analog speech signal is already processed by an ADC such that the signal is already uniformly sampled (with a sampling rate above the Nyquist sampling rates) and converted to digital format. The sampling rate is $f_s = 8000$ Hz with a 16-bit quantization. The amplitude range of the speech signals are normalized to 1 after mean removal. We next emulate the corresponding LC quantization framework using this setup. Note that such uniform sampling setup is analogous to the framework discussed in Remark 4.2 that approximates analog integrations with discrete summations.

For this LC quantization framework, the level sets are checked every 5 samples, i.e., $\tau = \frac{5}{f_s} = 6.25 \times 10^{-4}$ seconds. After the LC samples are taken, the signal is reconstructed using PWL, however, our results and implementation hold for any casual reconstruction scheme [17,28,29]. We point out that since we work with sampled speech signals, the integrations in the pseudo-code are replaced with summations. Such a scheme is suggested in [2] to

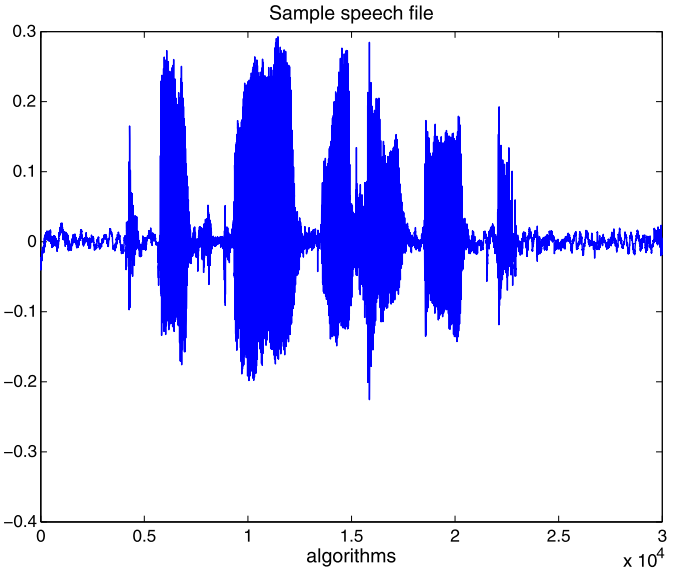

Fig. 4. A sample speech file used in the experiments.

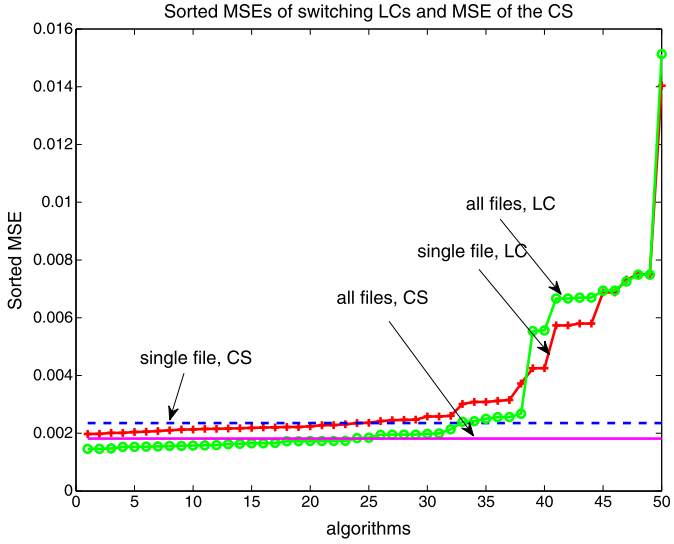

Fig. 5. MSEs corresponding to the introduced algorithm and the LC quantizers in $\boldsymbol{L}$ for a randomly selected sample speech file and averaged over all speech files.

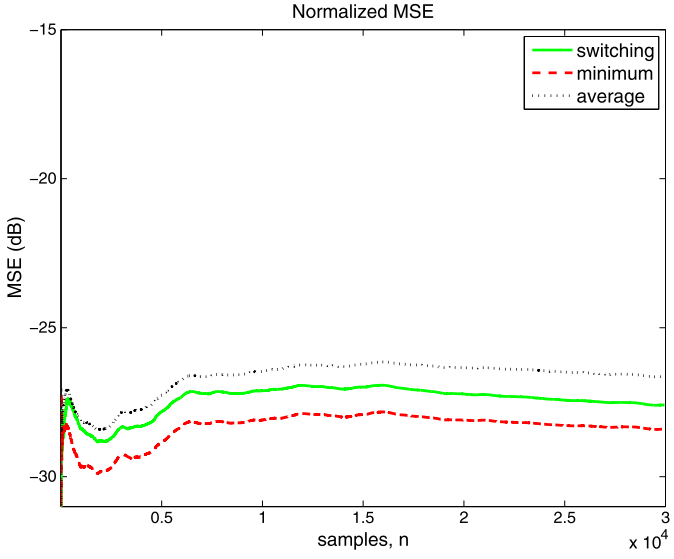

Fig. 6. Accumulated MSEs normalized with time corresponding to the introduced algorithm and the LC quantizers in $\boldsymbol{L}$ averaged over all the speech files.

replace analog integrals. Furthermore, replacing the analog integrations with discrete summations does not affect the corresponding theorems since derivations from (11) towards (17) still hold when integrations are replaced by discrete summations. In the first set of experiments, we divide the amplitude range using 16 uniformly spaced levels, i.e., $|\boldsymbol{l}| = 16$. We then construct all possible 10-level LC quantizers using these 16 levels, i.e., $\binom{16}{10}$ 10-level LC quantizers, and we randomly choose 50 10-level quantizers, i.e., $|\boldsymbol{L}| = 50$, from this set. In Fig. 4, we plot a sample speech file used for LC quantization. For such a randomly selected speech file of length approximately 4 seconds, we plot the MSE, i.e., the average of the error, corresponding to the CS algorithm, labeled "single file, CS", and MSEs corresponding to the 50 LC quantizers in $\boldsymbol{L}$, labeled "single file, LC" in Fig. 5. The MSEs of the LC quantizers are sorted in the figure. We note that the CS algorithm nearly achieves the performance of the best LC quantization scheme that can only be chosen in hindsight. To avoid any bias, we next perform the experiments over nearly 20 different speech files and plot in Fig. 5 the averaged MSEs corresponding to the switching algorithm, labeled "all files, CS", and LC quantizers in $\boldsymbol{L}$, labeled "all files, LC". To see the time evolution of the accumulated error, we next plot the accumulated MSE normalized with time, i.e., at each T, the accumulated MSE up to that time is normalized by T. In Fig. 6, we plot the normalized MSE for the introduced algorithm, labeled "CS", for the optimal batch algorithm, labeled "minimum", and for the average MSEs of all the LC quantization methods, labeled "average". We observe that the CS algorithm nearly achieves the performance of the optimal in hindsight LC quantizer and it has a better normalized MSE than the average normalized MSEs.

We next simulate the corresponding algorithm over a signal constructed as a summation of sinusoids with different frequencies and phases. Here, the underlying signal is generated as $y(t) = 0.4\sin(t+\pi/3) - 0.8\sin(0.8t) + \sin(2t+\pi/4) + w(t)$, where $w(t)$ is the additive noise. We simulate the algorithms under different SNRs, including SNR = 20 dB, 10 dB, 0 dB. In the first set of algorithms, the additive noise is zero mean and i.i.d., i.e., white noise, with Gaussian statistics, where the variance is set to yield the corresponding SNR. In the second set of experiments, the additive noise is generated from a first order auto-regressive model such that $w(t+1) = -0.9w(t) + bn(t)$, where $n(t)$ is zero mean i.i.d. Gaussian random process. Here, b is selected to yield the corresponding SNR. As the LC-quantizers, we choose $|\boldsymbol{l}| = 16$ and randomly select 80 8-level LC quantizers. We plot in Fig. 7 averaged MSEs over 20 independent trials under different SNRs and noise statistics. In Fig. 7, we have MSEs corresponding to 80 8-level LC quantizers, labeled "all-sorted", the competitive algorithm, labeled "CS". To test the performance of the CS algorithm with respect to ordinary uniform quantizers, we also plot the MSEs corresponding to uniform quantizers using 3 bits, labeled "quant, 3" and using 2 bits, labeled "quant, 2", respectively. We emphasize that an ordinary uniform quantizer, in our setup, produces samples with four times higher rate than the LC quantizers since $\tau = \frac{4}{f_s}$ in our simulations. Furthermore, for the LC quantizer we use only 8 levels, i.e., 3 bits, and for the CS algorithm we choose $v = 5\tau$. In this sense, we compare the performance of the CS algorithm with a uniform quantizer using 2 bits. We observe in these simulations that although the performance of the LC quantizers are affected from SNR, such affects are less visible compared to uniform quantizers. We emphasize that as the SNR drops, the performance of the CS algorithm degrades since as SNR drops the signal is dominantly noise and a single LC quantizer is sufficient, i.e., switching does not contribute to performance. We also observe that the CS algorithm is fairly robust to noise statistics used in our simulations.

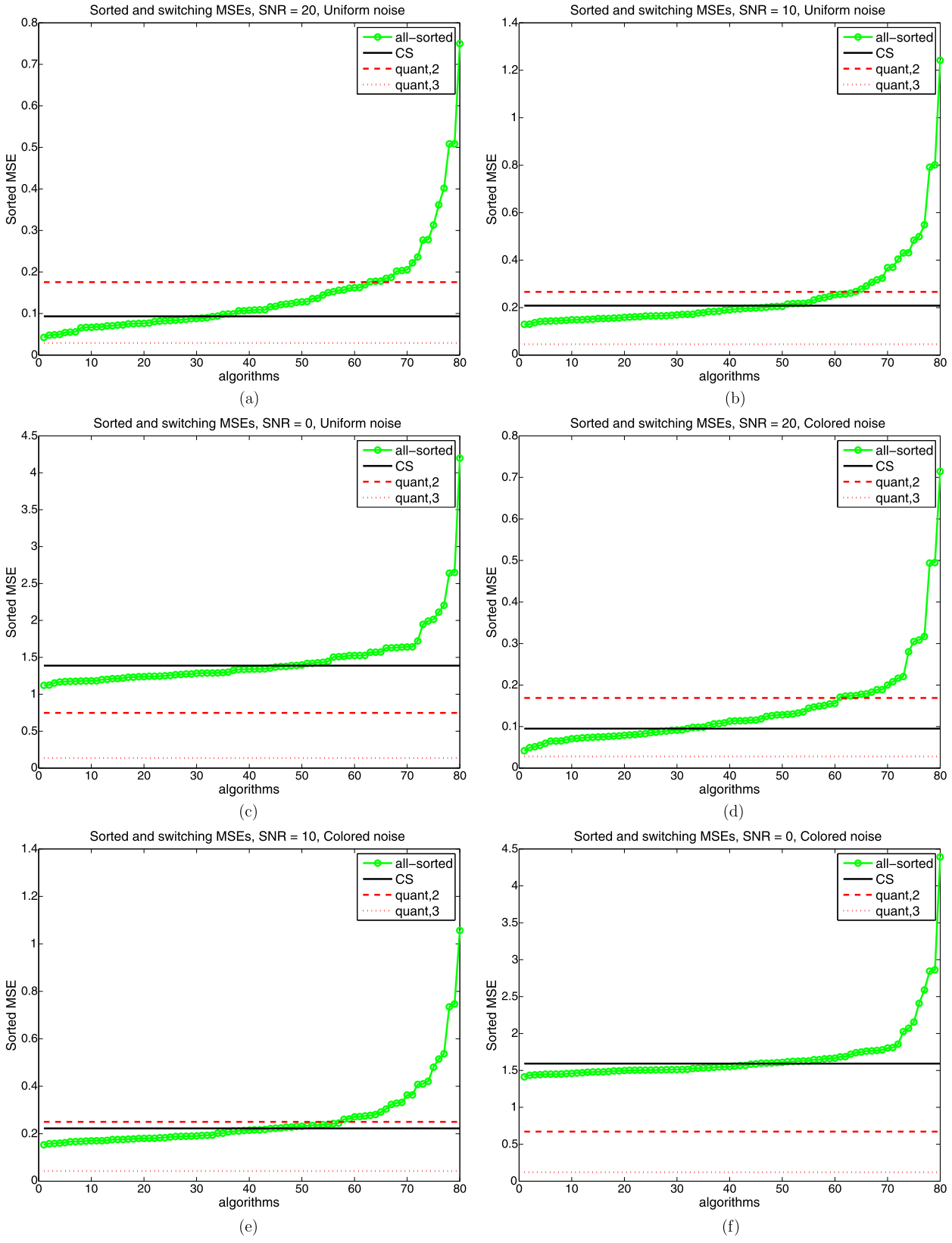

Fig. 7. Accumulated MSEs normalized with time corresponding to the introduced algorithm and the LC quantizers in $\boldsymbol{L}$ over sinusoids. (a) SNR = 20 dB uniform noise. (b) SNR = 10 dB uniform noise. (c) SNR = 0 dB uniform noise. (d) SNR = 20 dB colored noise. (e) SNR = 10 dB colored noise. (f) SNR = 0 dB colored noise.

6. Conclusion

In this paper, we introduce a novel LC sampling scheme that can track the best LC threshold levels (from a large class of possible threshold levels) uniformly for any deterministic and bounded analog signal without any stochastic assumptions on the underlying analog signal. The algorithm asymptotically achieves the performance of the best algorithm that can select both the partition

of the real line as well as the best LC levels in each interval by observing the whole analog signal in hindsight. We achieve this using a randomized sequential algorithm and demonstrate its performance on speech signals.

References

[1] M. Kafashan, S. Beygi, F. Marvasti, Asynchronous analog-to-digital converter based on level-crossing sampling scheme, EURASIP J. Adv. Signal Process. 2011 (2011) 109–117.
[2] K.M. Guan, S.S. Kozat, A.C. Singer, Adaptive reference levels in a level-crossing analog-to-digital converter, EURASIP J. Adv. Signal Process. 2008 (2008) 1–11.
[3] I.F. Blake, W.C. Lindsey, Level-crossing problems for random processes, IEEE Trans. Inform. Theory IT-19 (1973) 295–315.
[4] H.J. Landau, Necessary density conditions for sampling and interpolation of certain entire functions, Acta Math. 117 (1967) 37–52.
[5] B. Logan, Information in the zero crossings of bandpass signals, Bell Syst. Tech. J. 56 (4) (1977) 487–510.
[6] J.W. Mark, T.D. Todd, A nonuniform sampling approach to data compression, IEEE Trans. Commun. 29 (1) (1981) 24–32.
[7] F. Aeschlimann, E. Allier, L. Fesquet, M. Renaudin, Asynchronous FIR filters: towards a new digital processing chain, in: Proc. 10th IEEE Int. Symp. on Asynchronous Circuits and Systems, April 2004, pp. 198–206.
[8] F. Akopyan, R. Manohar, A.B. Apsel, A level-crossing flash asynchronous analog-to-digital converter, in: Proc. 12th IEEE Int. Symp. on Asynchronous Circuits and Systems, March 2006, pp. 286–294.
[9] K.M. Guan, A.C. Singer, A level-crossing sampling scheme for non-bandlimited signals, in: Proc. IEEE Conf. on Acoustics, Speech, and Signal Proc., May 2006, pp. 381–385.
[10] S.S. Kozat, A.C. Singer, Switching strategies for sequential decision problems with multiplicative loss with application to portfolios, IEEE Trans. Signal Process. 57 (2009) 2192–2208.
[11] A. Gyorgy, T. Linder, G. Lugosi, Efficient adaptive algorithms and minimax bounds for zero-delay lossy source coding, IEEE Trans. Signal Process. 52 (8) (2004) 2210–2217.
[12] A. Gyorgy, T. Linder, G. Lugosi, Tracking the best quantizer, IEEE Trans. Inform. Theory 54 (April 2008) 1604–1625.
[13] S.S. Kozat, A.C. Singer, Universal switching linear least squares prediction, IEEE Trans. Signal Process. 56 (January 2008) 189–204.
[14] S.S. Kozat, A.C. Singer, Universal randomized switching, IEEE Trans. Signal Process. 58 (3) (March 2010) 1922–1927.
[15] S.S. Kozat, A.C. Singer, Switching strategies for sequential decision problems with multiplicative loss with application to portfolios, IEEE Trans. Signal Process. 57 (2009) 2192–2208.
[16] A.V. Oppenheim, R.W. Schafer, Discrete-Time Signal Processing, Prentice–Hall, 2009.
[17] A. Gersho, R. Gray, Vector Quantization and Signal Compression, Kluwer Academic, 1992.
[18] D. Donoho, Compressed sensing, IEEE Trans. Inform. Theory 52 (4) (2006) 1289–1306.
[19] A. Zakhor, A.V. Oppenheim, Reconstruction of two-dimensional signals from level crossings, Proc. IEEE 78 (1) (January 1990) 31–55.
[20] K. Kozmin, J. Johansson, J. Kostamovaara, A low propagation delay dispersion comparator for level-crossing AD converter, Analog Integr. Circuits Signal Process. 62 (1) (2010) 51–61.
[21] K. Kozmin, J. Johansson, J. Delsing, Level-crossing ADC performance evaluation toward ultrasound application, IEEE Trans. Circuits Syst. I. Regul. Pap. 56 (8) (2010) 1708–1719.
[22] R.L. Grimaldi, S. Rodriguez, A. Rusu, 10-bit 5 kHz level-crossing ADC, in: 20th European Conference on Circuit Theory and Design, 2011, pp. 564–567.
[23] M. Trakimas, S. Sonkusale, A 0.8 V asynchronous ADC for energy constrained sensing applications, in: Proceedings of IEEE Custom Int. Circ. Conference, 2008, pp. 173–176.
[24] B. Schell, Y. Tsividis, A continuous-time programmable digital fir filter, IEEE J. Solid-State Circuits 41 (11) (2006) 2512–2520.
[25] B. Schell, Y. Tsividis, A continuous-time ADC/DSP/DAC system with no clock and with activity-dependent power dissipation, IEEE J. Solid-State Circuits 43 (11) (2008) 2472–2481.
[26] K. Guan, Opportunistic sampling by level-crossing, Ph.D. thesis, University of Illinois at Urbana-Champaign, Urban, IL, USA, 2008.
[27] T. Linder, G. Lugosi, A zero-delay sequential scheme for lossy coding of individual sequences, IEEE Trans. Inform. Theory 46 (6) (2001) 190–207.
[28] E. Masry, The application of random reference sequences to the reconstruction of clipped differentiable signals, IEEE Trans. Acoust. Speech Signal Process. ASSP-30 (1982) 953–963.
[29] A.F. Karr, R.J. Serfling, Error bounds for reconstruction of a function f from a finite sequence, SIAM J. Appl. Math. 43 (3) (1983) 476–490.
[30] Y. Tsividis, Digital signal processing in continuous time: a possibility for avoiding aliasing and reducing quantization error, in: Proc. IEEE Conf. on Acoustics, Speech, and Signal Proc., May 2004, pp. 589–592.
[31] M. Herbster, M.K. Warmuth, Tracking the best linear predictor, J. Mach. Learn. Res. (2001) 281–309.
[32] S. Haykin, Adaptive Filter Theory, Prentice–Hall, 1996.
[33] T. Sundstrom, B. Murmann, C. Svensson, Power dissipation bounds for high-speed Nyquist analog-to-digital converters, IEEE Trans. Circuits Syst. I. Regul. Pap. 56 (3) (2009) 509–518.
[34] F.M.J. Willems, Coding for a binary independent piecewise-identically-distributed source, IEEE Trans. Inform. Theory 42 (1996) 2210–2217.
[35] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Amer. Statist. Assoc. 58 (1963) 13–30.
[36] G.I. Shamir, N. Merhav, Low-complexity sequential lossless coding for piecewise-stationary memoryless sources, IEEE Trans. Inform. Theory 45 (5) (1999) 1498–1519.

Suleyman Serdar Kozat received the B.S. degree with full scholarship and high honors from Bilkent University, Turkey. He received the M.S. and Ph.D. degrees in electrical and computer engineering from Coordinated Science Laboratory, University of Illinois at Urbana Champaign, Urbana, IL, in 2001 and 2004, respectively.

After graduation, Dr. Kozat joined IBM Research T.J. Watson Research Center, Yorktown, NY as a Research Staff Member in Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. While doing his Ph.D., he was also working as a Research Associate at Microsoft Research, Redmond, WA, in Cryptography and Anti-Piracy Group. He holds several patent applications for his works performed in IBM Research and Microsoft Research. Currently, Dr. Kozat is an Assistant Professor at the Electrical and Electronics Engineering Department, Koc University, Turkey. Dr. Kozat coauthored more than 50 papers in refereed high impact journals and conference proceedings and has several patent applications. Overall, his research interests include intelligent systems, adaptive filtering for smart data analytics, online learning and machine learning algorithms for signal processing.

Dr. Kozat has been serving as an Associate Editor for the IEEE Transactions on Signal Processing and has been a reviewer for several IEEE Transactions and Conferences. He has been awarded IBM Faculty Award by IBM Research in 2011, Outstanding Young Researcher Award by the Turkish National Academy of Sciences in 2010, and holds Career Award by The Scientific Research Council of Turkey, 2009. Dr. Kozat has won several scholarships and medals in international and national science and math competitions. Dr. Kozat is a Senior Member of the IEEE.

Karen M. Guan received the B.Eng. and M.Eng. in electrical engineering and computer science from Massachusetts Institute of Technology in 2001 and 2002, and a Ph.D. in electrical and computer engineering from University of Illinois at Urbana-Champaign in 2012. Her research interests include signal processing, non-uniform sampling, level-crossing analog-to-digital converters, and universal sequential algorithms. Since 2008 she has been a senior communication systems engineer at Northrop Grumman Aerospace Systems at Redondo Beach, CA.

Andrew C. Singer received the S.B., S.M., and Ph.D. degrees, all in electrical engineering and computer science, from the Massachusetts Institute of Technology. Since 1998, he has been on the faculty of the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, where he is currently a Professor in the ECE department, a Research Professor in the Coordinated Science Laboratory, and a Willett Faculty Scholar. During the academic year 1996, he was a Postdoctoral Research Affiliate in the Research Laboratory of Electronics at MIT. From 1996 to 1998, he was a Research Scientist at Sanders, A Lockheed Martin Company in Manchester, New Hampshire, where he designed algorithms, architectures and systems for a variety of DOD applications. His research interests include statistical signal processing, communication systems and machine learning.

He was a Hughes Aircraft Masters Fellow, and was the recipient of the Harold L. Hazen Memorial Award for excellence in teaching in 1991. In 2000, he received the National Science Foundation CAREER Award, in 2001 he received the Xerox Faculty Research Award, and in 2002 was named a Willett Faculty Scholar. Dr. Singer serves as an Associate Editor for the IEEE Transactions on Signal Processing and is a member of the MIT Educational Council, and of Eta Kappa Nu and Tau Beta Pi. In 2005, Professor Singer was appointed as the Director of the Technology Entrepreneur Center (TEC) in the College of Engineering. He also

co-founded Intersymbol Communications, Inc., a venture-funded fabless semiconductor IC company, based in Champaign Illinois. A developer of signal processing enhanced chips for ultra-high speed optical communications systems, Intersymbol was acquired by Finisar Corporation (NASD:FNSR) in 2007. Professor Singer enjoys many forms of outdoor sports, including competing in triathlons and marathons. He has competed at all distances from olympic distance triathlons to marathons and the Ironman triathlon.