# Efficient Encoding Algorithm for Third-Order Spectral-Null Codes

Vitaly Skachek, Tuvi Etzion, *Member, IEEE*,
and Ron M. Roth, *Senior Member, IEEE*

***Abstract*— An efficient algorithm is presented for encoding unconstrained information sequences into a third-order spectral-null code of length $n$ and redundancy $9\log_2 n + O(\log\log n)$. The encoding can be implemented using $O(n)$ integer additions and $O(n\log n)$ counter increments.**

***Index Terms*—DC-free codes, spectral-null codes.**

## I. Introduction

Let $F$ be the bipolar alphabet $\{+1,-1\}$. A word

$$\underline{x} = (x_1, x_2, \cdots, x_n)$$

in $F^n$ is a *$k$th-order spectral-null word* (at zero frequency) if the respective real polynomial $x_1 z + x_2 z^2 + \cdots + x_n z^n$ is divisible by $(z-1)^k$. We denote by $\mathcal{S}(n,k)$ the set of all $k$th-order spectral-null words in $F^n$. Any subset $\mathcal{C}$ of $\mathcal{S}(n,k)$ is called a *$k$th-order spectral-null code* of length $n$. The concatenation of any $l$ words in $\mathcal{C}$ yields a word in $\mathcal{S}(nl,k)$; so, spectral-null codes can be used as block codes with a *redundancy* of $n - \log_2|\mathcal{C}|$ bits (per block of length $n$).

The set $\mathcal{S}(n,k)$ is equivalently characterized by

$$\mathcal{S}(n,k) = \left\{ \underline{x} \in F^n : \sum_{j=1}^{n} (j+c)^\ell x_j = 0, \quad \ell = 0,1,\ldots,k-1 \right\} \tag{1}$$

where $c$ is any real constant (see [5], and [6, Ch. 9]).

First-order spectral-null codes are also known by the names *balanced codes*, *zero-disparity codes*, or *DC-free codes*. There are known efficient encoding algorithms for these codes due to Knuth [3], Al-Bassam and Bose [1], and Tallini, Capocelli, and Bose [8]. Those algorithms result in codes with redundancy at most $\lceil \log_2 n \rceil$, where $n$ is the code length. By "efficient" we refer to the time and space complexity of the encoding; for example, in one of Knuth's algorithms, the complexity amounts to a lookup table of $\lceil \log_2 n \rceil^2$ bits and $O(n)$ increments/decrements of a $\lceil \log_2 n \rceil$-bit counter (as shown in [3], the space requirement can be eliminated by increasing the redundancy to $\log_2 n + O(\log\log n)$). The redundancy of $\mathcal{S}(n,1)$ is $\frac{1}{2}\log_2 n + O(1)$, and such redundancy can be attained by enumerative coding [6, p. 117]. In terms of complexity, however, enumerative coding is less efficient than Knuth's algorithms or the algorithms in [1] and [8].

Efficient coding algorithms for the second-order spectral-null case were presented in [5] and [7]. Those algorithms have redundancy of $3\log_2 n + O(\log\log n)$ bits and time complexity which amounts to $O(n)$ additions of $O(\log n)$-bit integers. Enumerative coding already turns out to be impractical for this case [5]. The redundancy of $\mathcal{S}(n,2)$ is known to be $2\log_2 n + O(1)$ [7].

Manuscript received February 22, 1997; revised September 22, 1997. This work was supported under Grant 95-522 from the United-States–Israel Binational Science Foundation (BSF), Jerusalem, Israel. The material in this correspondence was presented at the IEEE International Information Theory Workshop, Longyearbyen, Svalbard, Norway, July 1997.

The authors are with the Computer Science Department, Technion—Israel Institute of Technology, Haifa 32000, Israel.

Publisher Item Identifier S 0018-9448(98)00980-8.

For higher orders $k$ of spectral null, Karabed and Siegel presented in [2] a coding method based upon finite-state diagrams (see also Monti and Pierobon [4]). However, since the rate of their construction is strictly less than 1, the resulting redundancy is linear in the code length $n$. It follows that for any fixed $k$ and sufficiently large $n$, this redundancy is significantly larger than the upper bound $O(2^k \cdot \log n)$ on the redundancy of $\mathcal{S}(n,k)$; this bound is proved in [5] by nonconstructive arguments. A recursive construction is presented in [5] whose redundancy is $O(n^{1-\epsilon_k})$, where $0 < \epsilon_k < 1$ and $\lim_{k\to\infty} \epsilon_k = 0$. Yet, this redundancy is still considerably larger than the actual redundancy of $\mathcal{S}(n,k)$.

In this correspondence, we present an efficient algorithm for encoding unconstrained sequences into a third-order spectral-null code whose redundancy is logarithmic in the code length. More specifically, for code length $n$, the redundancy is $9\log_2 n + O(\log\log n)$ bits and the encoding complexity is $O(n)$ additions of $O(\log n)$-bit integers and $O(n\log n)$ increments/decrements of $\lceil \log_2 n \rceil$-bit counters.

## II. A Third-Order Spectral-Null Encoder

It was shown in [5] that the length $n$ of a third-order spectral-null word is divisible by 4, so we can write $n = 2h$ for some even integer $h$. We will use the definition of $\mathcal{S}(2h,3)$ that is obtained from (1) by substituting $k = 3$ and $c = -h - 1$. It will also be convenient hereafter to index the entries of a real word $\underline{x}$ of length $2h$ by $(x_{-h}, x_{-h+1}, \cdots, x_{h-1})$. We define the *moments* of such a word $\underline{x}$ by

$$\sigma_\ell(\underline{x}) \stackrel{\text{def}}{=} \sum_{j=-h}^{h-1} j^\ell \cdot x_j, \qquad \ell = 0,1,2,\cdots.$$

Clearly, a word $\underline{x} \in F^n$ is in $\mathcal{S}(2h,3)$ if and only if

$$\sigma_0(\underline{x}) = \sigma_1(\underline{x}) = \sigma_2(\underline{x}) = 0.$$

The following is an outline of our encoding algorithm. Let $n = 2h$ where $h$ is even and let $m$ be the integer $\lceil \log_2 n \rceil = 1 + \lceil \log_2 h \rceil$. The input to the algorithm is a balanced word $\underline{y}$ over $F$ of length $2h - 6m + 2$; namely, $\underline{y}$ is a word in $\mathcal{S}(2h - 6m + 2, 1)$ that is generated from the raw data by any known DC-free encoder (e.g., [1], [3], or [8]). Our algorithm regards $\underline{y}$ as a subword of a word $\underline{x}$ of length $n$ over $F \cup \{0\}$, where the remaining entries of $\underline{x}$ are initially set to zero; hence, $\sigma_0(\underline{x}) = 0$. Next, the algorithm reduces to zero the absolute values of $\sigma_2(\underline{x})$ and $\sigma_1(\underline{x})$ (in that order), by a sequence of bit shifts and bit swaps, and by assigning values of $F$ to the zero entries of $\underline{x}$. At this point, $\underline{x}$ becomes a word in $\mathcal{S}(2h,3)$. The encoding ends by coding recursively certain counters that were computed in the course of the algorithm, resulting in a word $\underline{x}' \in \mathcal{S}(2m + O(\log m), 3)$. The concatenation of $\underline{x}$ and $\underline{x}'$, in turn, will form the output third-order spectral-null word.

The algorithm makes use of the following index sets, all being subsets of $S = \{-h, -h+1, \cdots, h-1\}$:

- $S_{\mathrm{B2}} = \{d_i\}_{i=0}^{2m-8} \cup \{e_i\}_{i=0}^{2m-8}$, where

$$(d_i, e_i) = \begin{cases} (-10 \cdot 2^{i/2}, -6 \cdot 2^{i/2}), & \text{if } i \text{ is even} \\ (-9 \cdot 2^{(i+1)/2}, -7 \cdot 2^{(i+1)/2}), & \text{if } i \text{ is odd,} \end{cases} \qquad 0 \le i \le 2m-10 \tag{2}$$

$$(d_{2m-9}, e_{2m-9}) = (\tau_1, \tau_2)$$

and

$$(d_{2m-8}, e_{2m-8}) = (-\tau_1, 7)$$

where $\tau_1$ is the smallest odd integer in $S$ that is at least

**Step A: Initialization of $\underline{x}$**

Let $\langle \underline{x} \rangle_{S \setminus S_0} \leftarrow$ balanced $\underline{y}$. Let $\langle \underline{x} \rangle_{S_0} \leftarrow \underline{0}$.

**Step B: Reduction of $|\sigma_2(\underline{x})|$**

**Step B1:** Shift cyclically the entries of $\langle \underline{x} \rangle_{S \setminus S_0}$, until the resulting $\underline{x}$ is such that $|\sigma_2(\underline{x})| \le h^2$. Let $j_{\mathrm{B}}$ be the smallest number of shifts applied until this condition is met.

**Step B2:** For decreasing values of $i = 2m-8, 2m-9, \ldots, 0$, reduce the value of $|\sigma_2(\underline{x})|$ by assigning $x_{d_i} = -x_{e_i} = -1$ if $\sigma_2(\underline{x}) \ge 0$ and $x_{d_i} = -x_{e_i} = 1$ otherwise.

**Step B3:** Let $\langle \underline{x} \rangle_{S_{\mathrm{B3}}} \leftarrow$ the row in Table 1 that corresponds to $|\sigma_2(\underline{x})|$. If $\sigma_2(\underline{x}) \ge 0$ then let $\langle \underline{x} \rangle_{S_{\mathrm{B3}}} \leftarrow -\langle \underline{x} \rangle_{S_{\mathrm{B3}}}$ (i.e., negate $\langle \underline{x} \rangle_{S_{\mathrm{B3}}}$).

**Step C: Reduction of $|\sigma_1(\underline{x})|$**

**Step C1:** For increasing values of indexes $j = 1, 2, \ldots,$ swap $x_j$ with $x_{-j}$ until $|\sigma_1(\underline{x})| \le 2(h-1)$, and let $j_{\mathrm{C}}$ denote the number of swaps made until this condition is met.

**Step C2:** For decreasing values of $i = m-2, m-3, \ldots, 0$, reduce the value of $|\sigma_1(\underline{x})|$ by assigning $x_{2^i} = -x_{-2^i} = -1$ if $\sigma_1(\underline{x}) \ge 0$ and $x_{2^i} = -x_{-2^i} = 1$ otherwise.

**Step D: Recursive encoding**

Apply Step A–C recursively to the binary representation of $(j_{\mathrm{B}}, j_{\mathrm{C}})$. Concatenate the resulting word, $\underline{x}'$, with $\underline{x}$ to generate the final output of the encoder.

Fig. 1. Third-order spectral-null encoder.

$\sqrt{(h^2/2) + 49}$, and $\tau_2$ is the largest odd integer in $S$ that is at most $h/2$. We remove $\{d_i, e_i\}$ from $S_{\mathrm{B2}}$ if $d_i < -h$.[1]

- $S_{\mathrm{B3}} = \{0, -3, 3, -5, 5, 6, -7, -9, 9, 10, -11, 12, -13, 14\}$.
- $S_{\mathrm{C}} = \{\pm 2^i\}_{i=0}^{m-2}$.

We will assume hereafter that $h$ is large enough, in which case the sets $S_{\mathrm{B2}}$, $S_{\mathrm{B3}}$, and $S_{\mathrm{C}}$ are pairwise disjoint.[2] We let $S_0$ be the union $S_{\mathrm{B2}} \cup S_{\mathrm{B3}} \cup S_{\mathrm{C}}$. Note that

$$|S_0| \le 2(2m-7) + 14 + 2(m-1) = 6m - 2.$$

For a word $\underline{x}$ of length $n$ and a subset $Y$ of $S$, we will use the notation $\langle \underline{x} \rangle_Y$ for the subword of $\underline{x}$ that is indexed by $Y$.

The algorithm is summarized in Fig. 1. The input $\underline{y}$ is of length $|S \setminus S_0| \ge 2h - 6m + 2$.

## III. Analysis of the Algorithm

### A. Validity

We verify step by step that the algorithm indeed terminates with a third-order spectral-null word.

Step A ends with a word $\underline{x}$ with $\sigma_0(\underline{x}) = 0$. We turn to Step B and first verify that the shift counter $j_{\mathrm{B}}$ is well-defined.

*Lemma 3.1:* There is always a cyclic shift of $\langle \underline{x} \rangle_{S \setminus S_0}$ in Step B1 for which $|\sigma_2(\underline{x})| \le h^2$.

[1]This can happen only for $i = 2m-10, 2m-11$. Nevertheless, in those cases where only $\{d_{2m-10}, e_{2m-10}\}$ can be removed, then $\{d_{2m-9}, e_{2m-9}\}$ is redundant as well. In fact, it turns out that we will need all the $2(2m-7)$ elements of $S_{\mathrm{B2}}$ only when $h$ is close in value to a power of 2.

[2]As we show in the example of Section IV and as pointed out in the previous footnote, some elements in $S_{\mathrm{B2}}$ may sometimes be excluded. This allows to have $h$ as small as 18.

*Proof:* Let $\underline{x}^{(0)}$ denote the value of $\underline{x}$ at the beginning of Step B1 and let

$$\underline{x}^{(s)} = \left( x_{-h}^{(s)}, x_{-h+1}^{(s)}, \cdots, x_{h-1}^{(s)} \right)$$

be the word obtained from $\underline{x}^{(0)}$ by $s$ right cyclic shifts of $\langle \underline{x}^{(0)} \rangle_{S \setminus S_0}$ (note that $\langle \underline{x}^{(s)} \rangle_{S_0}$ remains zero for all $s$).

First, we show that

$$\left| \sigma_2(\underline{x}^{(s+1)}) - \sigma_2(\underline{x}^{(s)}) \right| \le 2h^2$$

for every $s \ge 0$. We say that location $j$ in $\underline{x}^{(s)}$ contains a *sign change* if $x_j^{(s)} \ne x_j^{(s+1)}$. Let $j_1 < j_2 < \cdots < j_t$ be the locations of the sign changes in $\underline{x}^{(s)}$. It is easy to verify that

$$\left| \sigma_2(\underline{x}^{(s+1)}) - \sigma_2(\underline{x}^{(s)}) \right| = \left| 2 \sum_{i=1}^{t} (-1)^i \cdot j_i^2 \right|. \tag{3}$$

Let $r$ be the smallest index $i$ such that $j_i \ge 0$. Define

$$\beta^- = \sum_{i=1}^{r-1} (-1)^i \cdot j_i^2$$

and

$$\beta^+ = \sum_{i=r}^{t} (-1)^i \cdot j_i^2.$$

Now, $\beta^-$ is a sum of integers with alternating signs and decreasing absolute values, where the first integer in the sum (if any) is negative. Hence

$$-h^2 \le -j_1^2 \le \beta^- \le 0. \tag{4}$$

On the other hand, $\beta^+$ is a sum of integers with alternating signs and increasing absolute values. Furthermore, since $t$ is even, the last integer in the sum is positive. Hence

$$0 \le \beta^+ \le j_t^2 \le (h-1)^2. \tag{5}$$

TABLE I
GENERATING ODD INTEGERS UP TO 63 BY BALANCED ASSIGNMENTS

| $\lvert\sigma_2(\underline{x})\rvert$ \ index | 0 | −3 | 3 | −5 | 5 | 6 | −7 | −9 | 9 | 10 | −11 | 12 | −13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | + | + | − | − | + | − | + | − | − | + | − | + | − | + |
| 3 | + | + | − | + | + | − | − | − | − | + | − | − | + | + |
| 5 | − | + | − | − | − | + | + | + | + | + | − | − | + | − |
| 7 | − | − | − | + | + | − | + | + | + | − | + | + | − | − |
| 9 | + | − | − | − | + | + | + | − | − | + | + | − | − | + |
| 11 | + | + | − | + | − | − | − | + | − | + | − | + | + | − |
| 13 | + | + | − | − | + | − | + | − | + | − | − | − | + | + |
| 15 | − | − | − | + | + | − | + | + | + | + | − | − | + | − |
| 17 | + | − | − | − | − | + | + | + | − | + | + | + | − | − |
| 19 | − | − | − | + | + | + | − | + | − | + | + | + | − | − |
| 21 | + | + | − | − | − | − | + | + | + | − | − | + | + | − |
| 23 | + | + | − | − | − | + | + | − | − | + | − | + | − | + |
| 25 | + | + | − | − | + | + | − | − | − | + | − | − | + | + |
| 27 | + | + | − | − | − | − | − | + | + | + | + | + | − | − |
| 29 | − | − | − | + | − | + | + | + | + | − | + | + | − | − |
| 31 | − | − | − | + | + | + | − | + | + | − | + | − | + | − |
| 33 | + | − | − | + | + | − | + | − | − | + | − | + | − | + |
| 35 | + | + | − | − | − | + | + | − | + | − | − | − | + | + |
| 37 | − | + | − | + | + | + | − | − | + | − | − | − | + | + |
| 39 | − | + | − | + | + | − | + | − | − | − | + | + | + | − |
| 41 | + | + | − | − | − | + | − | − | + | + | + | − | − | + |
| 43 | + | + | − | − | + | − | + | − | − | − | + | + | − | + |
| 45 | + | + | − | + | + | − | − | − | − | − | + | − | + | + |
| 47 | − | + | − | − | − | + | + | + | + | − | + | − | + | − |
| 49 | + | − | − | + | + | + | − | − | − | − | + | + | − | + |
| 51 | + | + | − | − | + | − | + | − | − | + | − | − | + | + |
| 53 | + | − | − | + | − | − | + | + | + | − | − | + | + | − |
| 55 | + | − | − | − | + | + | + | − | − | + | − | + | − | + |
| 57 | + | − | − | + | + | + | − | − | − | + | − | − | + | + |
| 59 | + | − | − | + | − | − | − | + | + | + | + | + | − | − |
| 61 | + | − | − | − | + | − | + | + | + | − | + | − | − | + |
| 63 | − | + | − | + | + | − | + | − | + | − | − | − | + | + |

Combining (3)–(5), we obtain

$$|\sigma_2(\underline{x}^{(s+1)}) - \sigma_2(\underline{x}^{(s)})| = 2 \cdot |\beta^- + \beta^+| \le 2h^2. \tag{6}$$

Next, we observe that

$$\sum_{s=0}^{|S\setminus S_0|-1} \sigma_2(\underline{x}^{(s)}) = 0.$$

Indeed, since $\underline{x}^{(0)}$ is balanced, it follows that

$$\sum_{s=0}^{|S\setminus S_0|-1} x_j^{(s)} = 0$$

for every $j \in S$. Hence

$$\sum_{s=0}^{|S\setminus S_0|-1} \sigma_2(\underline{x}^{(s)}) = \sum_{s=0}^{|S\setminus S_0|-1} \sum_{j\in S} j^2 \cdot x_j^{(s)} = \sum_{j\in S} j^2 \cdot \sum_{s=0}^{|S\setminus S_0|-1} x_j^{(s)} = 0.$$

Therefore, there is a "zero-crossing" value of $s$ for which

$$\sigma_2(\underline{x}^{(s)}) \cdot \sigma_2(\underline{x}^{(s+1)}) \le 0.$$

By (6), for such an $s$ we must have either $|\sigma_2(\underline{x}^{(s)})| \le h^2$ or $|\sigma_2(\underline{x}^{(s+1)})| \le h^2$. □

Each iteration in Step B2 changes the value of $\sigma_2(\underline{x})$ by an additive term $\pm(d_i^2 - e_i^2)$, where the negative sign is chosen when $\sigma_2(\underline{x}) \ge 0$. This further reduces the absolute value of $\sigma_2(\underline{x})$ as follows.

*Lemma 3.2:* The value of $\sigma_2(\underline{x})$ after Step B2 is an odd integer between $-63$ and 63.

*Proof:* First note that

$$2\left(d_{i-1}^2 - e_{i-1}^2\right) \ge d_i^2 - e_i^2, \qquad i = 2m-8, 2m-9, \cdots, 1 \tag{7}$$

and $2(d_{2m-8}^2 - e_{2m-8}^2) \ge h^2$. Specifically, for the values in (2) we have $d_i^2 - e_i^2 = 2^{i+6}$ for $i \le 2m-10$, and a simple check reveals that (7) holds also for $i \in 2m-8, 2m-9$ (if $d_{i-1} < -h$, then $\{d_{i-1}, e_{i-1}\}$ is removed from $S_{\mathrm{B2}}$; nevertheless, it can be verified that (7) still holds if we replace $(d_{i-1}, e_{i-1})$ by the pair $(d_r, e_r)$ of elements in $S_{\mathrm{B2}}$ with the largest index $r < i$). It follows that after iteration $i$ in Step B2, the resulting absolute value of $\sigma_2(\underline{x})$ is bounded from above by $d_i^2 - e_i^2$. In particular, for $i = 0$, the value of $\sigma_2(\underline{x})$ is an integer between $-64$ and 64. Furthermore, at this stage, the only zero entries of $\underline{x}$ are those that are indexed by $S_{\mathrm{B3}} \cup S_{\mathrm{C}}$. Since $S\setminus(S_{\mathrm{B3}} \cup S_{\mathrm{C}})$ contains an odd number of odd indexes, it follows that $\sigma_2(\underline{x})$ must be odd. □

The final reduction of $|\sigma_2(\underline{x})|$ to zero is done is Step B3, using Table I. It can be readily checked that for $r = 1, 3, \cdots, 63$, the values in row $r$ in the table contribute $r$ to $\sigma_2(\underline{x})$ (we negate those values in Step B3 if the contribution needs to be $-r$). Note that neither of the changes made in Step B affects the value of $\sigma_0(\underline{x})$, which still remains zero.

We now turn to Step C. This step is very similar to "Phase A" of the second-order spectral-null encoder in [5, Sec. IV]). We show next that the swap counter $j_{\mathrm{C}}$ is well-defined.

*Lemma 3.3:* There is always a word $\underline{x}$ obtained by less than $h$ swaps in Step C1 for which $|\sigma_1(\underline{x})| \le 2(h-1)$.

*Proof:* Let $\underline{x}^{[0]}$ denote the value of $\underline{x}$ at the beginning of Step C1 and let $\underline{x}^{[j]}$ be the word after the $j$th swap. First, it is easy to check that

$$|\sigma_1(\underline{x}^{[j+1]}) - \sigma_1(\underline{x}^{[j]})| \leq 4(h-1)$$

for all $j \geq 0$. Suppose we continue the swaps until $j = h-1$, and let $\underline{x}^{[h]}$ be the word obtained from $\underline{x}^{[h-1]}$ by negating the first entry (indexed by $-h$). In that case we will have

$$|\sigma_1(\underline{x}^{[h]}) - \sigma_1(\underline{x}^{[h-1]})| = 2h$$

and

$$\sigma_1(\underline{x}^{[h]}) = -\sigma_1(\underline{x}^{[0]}).$$

Hence, there must be a "zero-crossing" index $j < h$ for which $\sigma_1(\underline{x}^{[j]}) \cdot \sigma_1(\underline{x}^{[j+1]}) \leq 0$. For such a $j$ we must have either

$$|\sigma_1(\underline{x}^{[j]})| \leq 2(h-1)$$

or

$$|\sigma_1(\underline{x}^{[j+1]})| \leq 2(h-1).$$

Furthermore, if the zero-crossing index is $j = h-1$, we have

$$|\sigma_1(\underline{x}^{[h-1]})| \leq h$$

or

$$|\sigma_1(\underline{x}^{[h]})| = |\sigma_1(\underline{x}^{[0]})| \leq h. \qquad \square$$

Turning to Step C2, it can be easily verified that after iteration $i$ in that step, the resulting value of $|\sigma_1(\underline{x})|$ is bounded from above by $2^{i+1}$. In particular, for $i = 0$, the value of $\sigma_1(\underline{x})$ is an integer between $-2$ and 2. The following lemma implies that $\sigma_1(\underline{x})$ is actually zero at this point.

*Lemma 3.4:* For $n$ divisible by 4 and every $\underline{w} \in F^n$

$$\sigma_1(\underline{w}) \equiv \sigma_2(\underline{w}) \pmod 4.$$

*Proof:* Let $n = 2h$ and write $\underline{w} = (w_{-h}, w_{-h+1}, \cdots, w_{h-1})$. Then

$$\begin{aligned}\sigma_2(\underline{w}) - \sigma_1(\underline{w}) &= \sum_{j=-h}^{h-1} j(j-1) \cdot w_j \\ &= \sum_{l=-h/2}^{(h/2)-1} ((2l)(2l-1) \cdot w_{2l} + (2l+1)(2l) \cdot w_{2l+1}) \\ &= \sum_{l=-h/2}^{(h/2)-1} (2l)((2l-1) \cdot w_{2l} + (2l+1) \cdot w_{2l+1}).\end{aligned}$$

The result follows by observing that $(2l)((2l-1) \cdot w_{2l} + (2l+1) \cdot w_{2l+1})$ is divisible by 4 for every $l$. $\square$

Neither of the changes made in Step C affects the values of $\sigma_0(\underline{x})$ or $\sigma_2(\underline{x})$, which still remain zero. Hence, at the end of Step C we will have $\sigma_1(\underline{x}) \equiv 0 \pmod 4$. And since $-2 \leq \sigma_1(\underline{x}) \leq 2$, it follows that $\sigma_1(\underline{x})$ is zero.

Finally, Step D is rather straightforward and is based on the fact that the concatenation of two $k$th-order spectral-null words yields a $k$th-order spectral-null word.

Decoding of $\underline{y}$ is done by first reconstructing the values $j_{\mathrm{B}}$ and $j_{\mathrm{C}}$ from $\underline{x}'$. Once we have those two counters, we can reconstruct the values of $\underline{x}$ at the beginning of Steps C and B (in that order).

### B. Redundancy

We now compute the redundancy of the code which is defined by the words generated by the algorithm for any given length.

Using the algorithms in [1], [3], or [8], the redundancy in Step A due to the balancing of $\underline{y}$ is at most $m$ bits.

Steps B and C require $|S_0| \leq 6m-2$ bits to reduce $|\sigma_2(\underline{x})|$ and $|\sigma_1(\underline{x})|$ to zero. We also need $m$ bits to represent the shift counter $j_{\mathrm{B}}$ and $m-1$ bits to represent the swap counter $j_{\mathrm{C}}$.

In Step D, the encoding procedure is applied recursively to the $2m-1$ bits that represent $(j_{\mathrm{B}}, j_{\mathrm{C}})$, thus generating a word $\underline{x}' \in \mathcal{S}(m', 3)$ of length $m' = 2m + O(\log m)$. Since $m = \lceil \log_2 n \rceil$, it follows that the total redundancy of the encoding scheme is $9 \log_2 n + O(\log \log n)$ bits. This expression will be an upper bound on the redundancy also if we replace $n$ by the overall length, $n + m'$, of the output word.

### C. Time and Space Complexity

Step A can be implemented by $O(n)$ increments/decrements of a $\lceil \log_2 n \rceil$-bit counter, and a lookup table of size $\lceil \log_2 n \rceil^2$ bits.

As for Step B, we need to have the value of $\sigma_2(\underline{x})$ for each cyclic shift in Step B1. Assuming that the squares of the elements between 1 and $h$ are precomputed in a table, the initial value of $\sigma_2(\underline{x})$ in this step can be found in $O(n)$ additions of $O(\log n)$-bit integers. Now, let $\hat{\underline{x}}$ denote the word obtained from $\underline{x}$ by one right cyclic shift of $\langle \underline{x} \rangle_{S \setminus S_0}$, and let $\tilde{\underline{x}}$ be the word obtained from $\underline{x}$ by one right cyclic shift of the *whole* word $\underline{x}$. We describe next how $\sigma_\ell(\hat{\underline{x}})$ can be computed efficiently from $\sigma_\ell(\underline{x})$, $\ell = 1, 2$. Step B1 will then proceed iteratively by making $\hat{\underline{x}}$ the new value of $\underline{x}$.

Noting that $\sigma_0(\underline{x}) = 0$, it is easy to verify that

$$\sigma_1(\tilde{\underline{x}}) = \sigma_1(\underline{x}) - 2h \cdot x_{h-1} \quad \text{and} \quad \sigma_2(\tilde{\underline{x}}) = \sigma_2(\underline{x}) + 2\sigma_1(\underline{x}).$$

Therefore, once we have $\sigma_1(\underline{x})$ and $\sigma_2(\underline{x})$, it is straightforward to compute $\sigma_1(\tilde{\underline{x}})$ and $\sigma_2(\tilde{\underline{x}})$.

Let $S_1$ denote the set of all indexes $j \in S_0$ such that $j-1 \in S \setminus S_0$ (when $j = -h$, the index $j-1$ should read $h-1$). For an index $j \in S_1$, let $\hat{j}$ denote the smallest index in $S \setminus S_0$ that is larger than $j$ (if no such index exists, then $\hat{j}$ is defined as the smallest index in $S \setminus S_0$). For $\ell = 1, 2$, define

$$\alpha_\ell(\underline{x}) = \sum_{j \in S_1} (\hat{j}^\ell - j^\ell) \cdot x_{j-1}.$$

It can be readily verified that

$$\sigma_\ell(\hat{\underline{x}}) = \sigma_\ell(\tilde{\underline{x}}) + \alpha_\ell(\underline{x}), \qquad \ell = 1, 2.$$

The expressions $\alpha_\ell(\underline{x})$ can be computed using $O(\log n)$ additions of $O(\log n)$-bit integers. The following discussion outlines how the computation of $\alpha_\ell(\underline{x})$ can be accelerated further through the use of small lookup tables.

Let $S_1 = \cup_t S_1(t)$ be a partition of $S_1$ into $O(1)$ subsets $S_1(t)$, each of size less than $m$. For each subset $S_1(t)$, construct a lookup table for computing the expression

$$\alpha_\ell(\langle \underline{x} \rangle_{S_1(t)}) = \sum_{j \in S_1(t)} (\hat{j}^\ell - j^\ell) \cdot x_{j-1}$$

as a function of the entries $x_{j-1}, j \in S_1(t)$. Each lookup table consists of less than $n$ entries and each entry contains an $O(\log n)$-bit integer. Note that these lookup tables can be computed in time $O(n)$ and that they depend on $n$, but not on the encoded word. In order to access the bits $x_{j-1}, j \in S_1(t)$, within $\langle \underline{x} \rangle_{S \setminus S_0}$, we will use $|S_1(t)|$ pointers (counters) that will be decremented after each shift. (In hardware implementations, we can instead store $\langle \underline{x} \rangle_{S \setminus S_0}$ in a shift register.) Once we have computed the $O(1)$ expressions $\alpha_\ell(\langle \underline{x} \rangle_{S_1(t)})$,

$$----++---+++-+-+++--++-+--+ \tag{8}$$

$$----++-0--0+0+0+0000000000000\overset{\downarrow}{0}0000000000+0-0+0-+++--++-+--+ \tag{9}$$

$$+--+---0-+0+0-0-0000000000000\overset{\downarrow}{0}0000000000-0+0+0++-+-+++--++- \tag{10}$$

$$+--+---+-+++--0-+0-0+000-0000000\overset{\downarrow}{0}000000-000-0+0+0++-+-+++--++- \tag{11}$$

$$+--+---+-+++--0-+----+-0--+0-00\overset{\downarrow}{+}00+0+--0-+-++++0++-+-+++--++-. \tag{12}$$

$$+--+---+-+++--0++++-+-0--+0+00\overset{\downarrow}{+}00-0+--0-+----+-0++-+-+++--++- \tag{13}$$

$$+--+---+-+++--+++++-+----++-+\overset{\downarrow}{+}-+--+--+-+---+--++-+-+++--++- \tag{14}$$

we obtain $\alpha_\ell(\underline{x})$ as their sum. Note that this computation of $\sigma_2(\underline{x})$ allows us to find $j_{\rm B}$ without actually shifting $\langle \underline{x} \rangle_{S \setminus S_0}$. This makes the computation efficient also in software implementations.

Steps B2, B3, and C are rather straightforward and can be implemented using $O(n)$ integer additions. Hence, the overall time and space complexity of our encoding algorithm is as follows:

- $O(n)$ additions of $O(\log n)$-bit integers,
- $O(n)$ accesses to $O(1)$ tables, each of size $< n$, and —
- $O(n)$ increments/decrements of $O(\log n)$ counters, each $\lceil \log_2 n \rceil$ bits long.

## IV. Example

We consider here the case $n = 60$ (for such a small value of $n$ the redundancy is relatively big, so this example is given only for the purpose of illustrating the encoding steps). In this case $h = 30$ and $m = 6$, and the set $S_{\rm B2}$ is given by

$$S_{\rm B2} = \{-10, -18, -20, -23\} \cup \{-6, -14, -12, 7\}$$

where $\tau_1 = 23$. Note that we have excluded the elements $\{d_3, e_3\} = \{\tau_1, \tau_2\} = \{23, 15\}$ from $S_{\rm B2}$ since they will not be required in Step B2: The value of $d_2^2 - e_2^2 = (-20)^2 - (-12)^2 = 256$ is already greater than half the value of $d_4^2 - e_4^2 = (-23)^2 - 7^2 = 480$. The set $S_{\rm B3}$ is of size 14 and $S_{\rm C}$ is given by $\{\pm 2^i\}_{i=0}^4$. Hence, $|S_0| = 32$.

Suppose that the input balanced word $\underline{y}$ of length $n - |S_0| = 28$ is given by (8) at the top of the page. After embedding $\underline{y}$ in $\underline{x}$ in Step A, we obtain the word (9) (the arrow points at the entry indexed by 0). For this word we have $\sigma_0(\underline{x}) = 0$ and $\sigma_2(\underline{x}) = -2047$, and when applying the cyclic shifts in Step B1 we produce words $\underline{x}$ with $\sigma_2(\underline{x}) = -1853, -1755, -1357$, and $-625$. The last value corresponds to (10) which is the first word in this step with $|\sigma_2(\underline{x})| \leq h^2 = 900$; so, $j_{\rm B} = 4$. The assignment of values to the entries indexed by $S_{\rm B2}$ in Step B2 results in (11) with $\sigma_2(\underline{x}) = 47$. In Step B3, we fill in the entries indexed by $S_{\rm B3}$ with the negated entries of the row that corresponds to 47 in Table I. This produces the word (12). Step C1 starts with $\sigma_1(\underline{x}) = 174$ and then continues with iterated swaps that generate words $\underline{x}$ with $\sigma_1(\underline{x}) = 194, 194, 182$ (nine iterations), $134, 82, 82,$ and 22. The last value corresponds to (13) and this word is the first to occur in this step with $|\sigma_1(\underline{x})| \leq 2(h-1) = 58$, and so $j_{\rm C} = 15$. Step C2 fills in the entries indexed by $S_{\rm C}$ to produce the word (14) for which we have $\sigma_0(\underline{x}) = \sigma_1(\underline{x}) = \sigma_2(\underline{x}) = 0$.

Note that we can make the counting of the swaps in Step C more economical by skipping index pairs $(-j, j)$ with $x_{-j} = x_j$ (in which case the swaps become in effect negations of $x_{-j}$ and $x_j$ whenever $x_{-j} \neq x_j$).

Finally, the counters $(j_{\rm B}, j_{\rm C})$ are coded into up to $2 \cdot 6 - 1 = 11$ bits and undergo a recursive encoding in Step D.

## Acknowledgment

The authors wish to thank the reviewers for their helpful comments and suggestions.

## References

[1] S. Al-Bassam and B. Bose, "On balanced codes," *IEEE Trans. Inform. Theory*, vol. 36, pp. 406–408, 1990.

[2] R. Karabed and P. H. Siegel, "Matched spectral-null codes for partial-response channels," *IEEE Trans. Inform. Theory*, vol. 37, pp. 818–855, 1991.
[3] D. E. Knuth, "Efficient balanced codes," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 51–53, 1986.
[4] C. M. Monti and G. L. Pierobon, "Codes with a multiple spectral null at zero frequency," *IEEE Trans. Inform. Theory*, vol. 35, pp. 463–472, 1989.
[5] R. M. Roth, P. H. Siegel, and A. Vardy, "High-order spectral-null codes: Constructions and bounds," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1826–1840, 1994.
[6] K. A. Schouhamer Immink, *Coding Techniques for Digital Recorders*. London., U.K.: Prentice-Hall, 1991.
[7] L. G. Tallini, S. Al-Bassam, and B. Bose, "On efficient high-order spectral-null codes," in *Proc. IEEE Int. Symp. Information Theory* (Whistler, BC, Canada, 1995), p. 144.
[8] L. G. Tallini, R. M. Capocelli, and B. Bose, "Design of some new balanced codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 790–802, 1996.

## Exact Recovery of Higher Order Moments

L. Cheded, *Member, IEEE*

***Abstract*—This correspondence addresses the problem of exact recovery of higher order moments of unquantized signals from those of their quantized counterparts, in the context of nonsubtractive dithered quantization. It introduces a new statistical characterization of the dithered quantizer in the form of a $p$th-order moment-sense input/ouput function $h_p\ (x)$. A class of signals for which the solution to the exact moment recovery problem is guaranteed is defined, and some of its key properties are stated and proved. Two approaches to this problem are discussed and the practical gains accruing from the 1-bit implementation of the second approach are highlighted. Finally, a fruitful extension of this work to the exact recovery of cumulants is briefly pointed out.**

### I. Introduction

Statistical moments are useful in a variety of scientific and engineering fields, as witnessed by the wide applicability of correlation functions [1], and by the increasing volume of research into the study and application of higher order moments, cumulants and higher-order spectra in various areas (see [2] and [3] for two large bibliographies on these and other applications). The key motivation behind using cumulants (which are basically linear combinations of higher order moments) and higher order statistics is that these tend to contain more information (e.g., phase) about the signal under study, and also offer processing domains of higher signal-to-noise ratios (SNR's) than their second-order counterparts. The increase in SNR is only true though whenever the contaminating signals are Gaussian, since their cumulants (order greater than 2) and higher order statistics vanish. Other motivating reasons for using higher order statistics can be found in [3]. In practice and for well-known reasons, moments are computed digitally and this involves an amplitude information loss that increases with decreasing quantization resolution. It is therefore of practical interest to seek a characterization of all those quantizer inputs whose finite-order unquantized higher order moments can be exactly recovered from their quantized brethren, regardless of the quantization resolution used. This problem will henceforth be referred to as the exact moment recovery problem. The objective of this correspondence is to seek a solution to this problem which is insensitive to the quantization-induced amplitude information loss, regardless of how coarsely (or finely) coded the quantizer input is. Our study of the exact moment recovery problem was inspired from the classical work of Widrow [4] which, in turn, drew its inspiration from the seminal work of Bennett [5]. The work in [4] popularized the additive model of quantization, i.e., $X_Q \triangleq Q[X] = X + n_Q$, where $Q$ represents the classical (nonlinear) quantizer and $X, X_Q$, and $n_Q$ are the associated input, output, and quantization noise, respectively. As shown in [5], this model holds well under the conditions of no quantizer overload, fine quantization, and smooth input probability density function, $p_X(x)$. In earlier work [6]–[8], we presented a multidimensional framework for the study of the exact moment recovery problem and used it to rediscover the arcsine (Van Vleck) law. Also, based on the following parametrization of the whole class of uniform quantizers:

$$X \mapsto X_Q = Q[X] = (a + n + \tfrac{1}{2})q \quad \text{if } (a+n)q \le x < (a+n+1)q \tag{1}$$

where $a \in [-\frac{1}{2}, \frac{1}{2})$ is the shift factor, $q$ the quantization step, and $n \in \mathcal{Z}$ which is the set of integers, we showed that the $p$th-order quantized moment $\mu_{Q_p} \triangleq E[X_Q^p]$ is related to the unquantized moments, $\mu_r \triangleq E[X^r],\ 0 \le r \le p$ by

$$\mu_{Q_p} = A_p + B_p \tag{2}$$

where $A_p$ and $B_p$ are called the principal (or wanted) term and the bias (or unwanted) term, respectively, and are given by

$$A_p = \frac{1}{p+1}\sum_{r=0}^{p} C_r^{p+1}\left(\frac{q}{2}\right)^{p-r}\mu_r\ [p \oplus r \oplus 1] \tag{3}$$

$$B_p = \sum_{n\neq 0} e^{-i2\pi a n}\sum_{r=0}^{p-1}\left(\frac{q}{2}\right)^{p-r} i^{-r}\sum_{\lambda=0}^{(p-1)-r} \cdot \frac{p!\,i^{\lambda-1}}{r!(p-r-\lambda)!}\frac{p\oplus r\oplus\lambda}{(n\pi)^{\lambda+1}}W^{(r)}\left(\frac{2n\pi}{q}\right) \tag{4}$$

where

$$C_r^{p+1} = \binom{p+1}{r} = (p+1)!/r!(p+1-r)!$$

and $\oplus$ denotes modulo-2 addition.

In an attempt to automatically cancel $B_p$, which depends on the generally unknown unquantized characteristic function $W(u)$, the following theorem (which is an extension of its original counterpart given in [4]) was derived.

*Generalized Quantizing Theorem:* Given a general uniform quantizer with shift factor $a$, input $X$, step $q$, and output $X_Q$, if

a) the input charateristic function, $W(u)$, is bandlimited, i.e.,

$$W(u) = 0 \quad \text{for } |u| \ge u_{\max} \tag{5}$$

b) the quantization fineness, defined by $u_Q \triangleq 2\pi/q$, satisfies

$$u_Q \ge 2u_{\max} \tag{6}$$

Manuscript received December 7, 1995; revised June 1, 1997. This work was supported by the King Fahd University of Petroleum and Minerals.

The author is with the Department of Systems Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia.

Publisher Item Identifier S 0018-9448(98)01306-6.