LOW COMPLEXITY FACTORIAL PULSE CODING OF MDCT COEFFICIENTS USING APPROXIMATION OF COMBINATORIAL FUNCTIONS

Udar Mittal, *James P. Ashley*, and *Edgardo M. Cruz-Zeno*
mittal, ashley, cruz@labs.mot.com
Speech Processing Research Labs, Motorola Labs,
Schaumburg, IL, 60196, USA

ABSTRACT

Factorial pulse coding, a method which is known to efficiently code an information signal using unit magnitude pulses, involves computation of combinatorial functions. These computations are highly complex as they require many multiply and divide operations on multi-precision numbers, especially when the length of a signal is large or many unit magnitude pulses are used for coding. In this paper, we propose a very low complexity method for approximation of these combinatorial functions. The approximate functions satisfy a property which preserves unique decode-ability of the factorial packing encoding/decoding algorithm. The low complexity computation enables use of factorial packing in encoding/decoding of 144 MDCT coefficients using 28 unit magnitude pulses for the audio coding mode of the EVRC-WB speech coding standard without affecting the number of bits required for coding.

Index Terms—factorial packing, factorial pulse coding, enumeration, MDCT, audio coding

1. INTRODUCTION

Factorial pulse coding (FPC) [1] is an enumerative method for coding a vector $\boldsymbol{x} = \{x_0, x_1, .., x_{n-1}\}$ satisfying

$$m = \sum_{i=0}^{n-1} |x_i|, \tag{1}$$

and all values of vector x_i are integral valued such that $-m \le x_i \le m$, where m is the total number of unit amplitude pulses, and n is the vector length. The $M = \lceil \log_2(N) \rceil$ bits are used to code N combinations in a maximally efficient manner [1,2], such that the following expression, which describes the theoretical minimum number of combinations, holds true:

$$N = \sum_{d=1}^{\min(m,n)} F(n,d)D(m,d)2^d \le 2^M \tag{2}$$

For this equation, $F(n, d)$ is given by:

$$F(n,d) = \frac{n!}{d!(n-d)!}, \tag{3}$$

$D(m,d)$ are the number of combinations of d non-zero vector elements given m total unit pulses given by:

$$D(m,d) = F(m-1, d-1), \tag{4}$$

and 2^d represents the combinations required to describe the polarity (sign) of the d non-zero vector elements. In the FPC method, the information content is divided into four constituents: 1) *number of non-zero pulse positions* (ν); 2) *position of the non-zero pulses* (π); 3) *magnitude of the non-zero pulses* (μ); and 4) *signs of the non-zero pulses* (σ). The encoding of pulse positions in FPC is given by

$$C = \sum_{k=1}^{\nu} F(p_k, k) \qquad 0 \le p_k < n \tag{5}$$

where $\pi = \{p_1, p_2, ..., p_n\}$.

The magnitude of non-zero pulses is coded using (5) iteratively. Details of the encoding can be found in [1,3]. Decoding π from the code C requires finding the smallest value of s such that $F(s,k)$ is greater than a given constant. This procedure is repeated v times. Efficient encoding and decoding requires low complexity computation of $F(s,k)$. $F(s,k)$ is generally computed left to right using the following expression:

$$F(s,k) = s \cdot (s-1) \cdot \left(\frac{1}{2}\right) \cdot (s-2) \cdot \left(\frac{1}{3}\right) \cdot \cdot (s-k+2) \cdot \left(\frac{1}{k-1}\right) \cdot (s-k+1) \cdot \left(\frac{1}{k}\right) \tag{6}$$

This requires k-1 multiplications and k-1 divisions. Smaller values of n and m, such as n = 54 and m $\le$ 7 which is used in coding of fixed codebook excitation in the EVRC-WB codec, do not cause any unreasonable complexity burden. However, larger values can quickly cause problems as these operations are then performed over a multi-precision number. For example, use of this coding method for audio coding with $n = 144$ and $m = 28$ requires 28 multiply and divide operations over a 99-bit number. This complexity burden can be reduced by storing these functions in memory using methods proposed in [3]. However, the memory requirement for large values of n and m may be prohibitive, especially for use in handheld devices, such as a cellular phone.

In this paper, we propose using approximations of the combinatorial functions instead of exact combinatorial functions in FPC. These approximate functions can be computed at very low complexity using operations on finite precision numbers (< 32 bits) and with very low memory requirements. We will show that if these functions satisfy a certain property, then the resulting codeword is uniquely decodable using the encoding/decoding algorithm of FPC.

2. FACTORIAL PULSE CODING USING APPROXIMATE COMBINATORIAL FUNCTIONS

As mentioned in Section 1, FPC uses (5) for encoding the pulse positions and magnitudes. The decoding process requires finding the smallest value of s such that $F(s,k)$ is greater than a given constant. We want to construct an approximate function $F'(s,k)$ such that using these functions, the encoding method of (5), and the same FPC decoding algorithm can be used. First, we state and prove a property which is a sufficient condition for this replacement, and then present an algorithm to compute such functions.

Uniqueness Theorem: *If* $F'(s,k)$ *satisfy*

$$\begin{aligned}&F'(s,k) \ge F'(s-1,k) + F'(s-1,k-1),\ n \ge s \ge k \ge 0, m \ge k\\&F'(s,k) = 0, s < k \text{ or } k < 0,\\&F'(0,0) = 1,\end{aligned} \tag{7}$$

then $F'(s,k)$ *can be substituted in the encoding/decoding algorithms of FPC to form a codeword which is uniquely decodable.*

Proof: We assume that $F'(s,k)$ satisfies the property given in (7). Since the decoding process of FPC finds the location of the k^{th} pulse by finding the smallest value of $s \ge k$ such that $F'(s,k)$ is greater than a given constant. Thus, if the maximum codeword for k pulses occupying position 0 to s-1 is less than $F'(s,k)$ then s-1 can be decoded as the occupied position. The maximum such codeword, using (5) and the fact that for a given j, $F'(r,j)$ is a non-decreasing function of r (from (7)), is given by:

$$C_{\max}(s,k) = \sum_{j=1}^{k} F'(s-j,k-j+1) \tag{8}$$

Thus for correct decoding,

$$F'(s,k) > \sum_{j=1}^{k} F'(s-j,k-j+1) \tag{9}$$

Since $F'(s,k)$ is a non-negative integer, the above expression is equivalent to:

$$F'(s,k) \ge \sum_{j=1}^{k} F'(s-j,k-j+1) + 1 \tag{10}$$

Consider first the case where $s = k$. In (10), replacing $F'(k-j,k-j+1) = 0$, we get:

$$F'(k,k) \ge 1 \tag{11}$$

Repeatedly applying the inequality in (7) for $s = k$, we get:

$$F'(k,k) \ge F'(k-1,k-1) \ge F'(k-2,k-2) \ldots \ge F'(0,0) = 1 \tag{12}$$

From (12) we can see that (7) implies (9) for $s = k$. Now let us look at $s > k$, again repeatedly applying inequality in (7), we get

$$\begin{aligned}F'(s,k) &\ge F'(s-1,k) + F'(s-1,k-1)\\&\ge F'(s-1,k) + F'(s-2,k-1) + F'(s-2,k-2)\\&\ge \sum_{j=1}^{r} F'(s-j,k-j+1) + F'(s-r,k-r)\\&\ge \sum_{j=1}^{k} F'(s-j,k-j+1) + F'(s-k,0)\end{aligned} \tag{13}$$

Similarly as in (12), for $s > k$, we can show that $F'(s-k,0) \ge 1$. This combined with (13) shows that (10) is also true if (7) is satisfied.

■

In order to describe the generation of $F'(s,k)$, let us proceed by first deriving a function $F'(s,k)$ that is a suitable approximation of $F(s, k)$. The first step is to take the logarithm of an arbitrary base a of (3), and taking the inverse log base a of the rearranged terms:

$$F(s,k) = \exp_a\left(\sum_{i=s-k+1}^{s} \log_a(i) - \sum_{j=1}^{k} \log_a(j)\right), \tag{14}$$

where the function $\exp_a(t) = a^t$. Next, define functions $P(i)$, $Q(k)$, and $R(t)$, and substitute into (14) such that:

$$F(s,k) = R\left(\sum_{i=s-k+1}^{s} P(i) - Q(k)\right), \tag{15}$$

where $P(i) = \log_a(i)$, $Q(k) = \sum_{j=1}^{k} \log_a(j)$, and $R(t) = \exp_a(t) = a^t$. Clearly, $F(s,k)$ can be precisely computed using (14) and (15) provided the logarithm in (14) has multi-precision output and the exponential function in (15) has both multi-precision input and output. Thus, such an approach may not be conducive for exact computation of combinatorial function.

We will now investigate using a similar approach for computing approximations of combinatorial functions without using multi-precision logarithm and exponential functions. Besides having low complexity computation, these functions should also satisfy Uniqueness Theorem (7).

Referring back to (15), we now wish to generate $F'(s, k)$ by creating the functions $P'(i)$, $Q'(k)$, and $R'(t)$, with low complexity approximations of the original functions, $P(i)$, $Q(k)$, and $R(t)$, respectively, such that:

$$F'(s,k) = R'\left(\sum_{i=s-k+1}^{s} P'(i) - Q'(k)\right). \tag{16}$$

Considering $P(i)$, we may wish to approximate the function such that $P'(i) \geq \log_a(i), i \in [1, 2, \ldots, n]$. If we choose $a = 2$ and then restrict $P'(i)$ to 32 bits of precision, the resulting operations are easy to implement on a handheld mobile device since most DSPs support efficient 32 bit additions. Therefore, we define:

$$P'(i) = 2^{-l(i)} \left\lfloor 2^{l(i)} \log_2(i) + 1 \right\rfloor, \quad i \in [1, 2, \ldots, n], \tag{17}$$

Using this methodology, the function $P'(i) \geq \log_2(i)$ for all $i \geq 1$. To avoid the complexity of computing these values in real-time, they can be pre-computed and stored in a table using only 144 x 4 bytes of memory for the $F(144, 28)$ example. Using a similar methodology for approximating $Q(k)$, we get:

$$Q'(k) = \begin{cases} 0, & k = 1 \\ \sum_{j=2}^{k} 2^{-l(j)} \left\lfloor 2^{l(j)} \log_2(j) - 1 \right\rfloor, & k \in [2, \ldots, m] \end{cases}, \tag{18}$$

This guarantees that $Q'(k) \leq \sum_{j=1}^{k} \log_2(j)$ so that the contribution of $Q'(k)$ will guarantee $F'(s,k) \geq F(s,k)$. Like $P'(i)$, $Q'(k)$ can be pre-computed and stored in a table using only 28 x 4 bytes of memory for the $F(144, 28)$ example. For defining $R'(t)$, we need to first define t as:

$$t(s,k) = \sum_{i=s-k+1}^{s} P'(i) - Q'(k). \tag{19}$$

With $P'(i)$ and $Q'(k)$ defined above, t is preferably a 32 bit number with an 8 bit unsigned integer component t_i and a 24 bit fractional component t_f. Using this, we can derive $R'(t) \approx \exp_2(t) = 2^t$ by letting $t = t_i + t_f$ and then taking the inverse logarithm base 2 to yield $2^t = 2^{t_i} 2^{t_f}$. We then use a Taylor series expansion to estimate the fractional component to the desired precision, represented by $T_f = 2^{t_f}$, truncating the result using a shift and the floor function, and then appropriately shifting the result to form a multi-precision result (with only l significant bits), such that:

$$R'(t) = \left\lfloor 2^{t_i - l} \left\lfloor 2^l T_f \right\rfloor \right\rfloor. \tag{20}$$

Empirically it has been found for $n = 144$, and $m = 28$, using $l(i) = 21$ in (17), $l(j) = 14$ in (18), and $l = 19$ in (20) results in $F'(s,k)$ which satisfy (7). The number of bits M needed to encode is now given by

$$N = \sum_{d=1}^{\min(m,n)} F'(n,d) D(m,d) 2^d \leq 2^M, \tag{21}$$

where $D(m,d)$ is identical to the one defined in (4), i.e., the actual combinatorial function F instead of the approximate F' are used for coding of magnitudes. For $m = 28$, $D(m,d)$ is typically less than 31 bits and hence does not require multi-precision operations. For large m, when $D(m,d)$ is greater than 31 bits, F' can be used for generating $D(m,d)$.

It turns out when the above approximate combinatorial functions are used for the encoding purpose, the number of bits for $n = 144$, $m = 28$ is 131, and for $n = 144$, $m = 23$ is 114, which are same numbers of bits needed when precise combinatorial functions are used in FPC.

To preserve a property of $F(s,k)$, define $F'(s, s-k) = F'(s,k)$. With this definition, it can be verified that if $F'(s,k)$ satisfies (7) then $F'(s,u)$, where $u = s - k$, also satisfies (7), i.e., $F'(s,u) \geq F'(s-1,u) + F'(s-1,u-1)$.

The functions $P'(n)$ and $Q'(k)$ are pre-computed and stored in the memory. Here we state few properties as lemmas which can be used to compute $P'(i)$ for any given function $R'(t)$ and predefined $Q'(k)$.

Lemma 1: *If* $2^t \geq R'(t) \geq (1 - \varepsilon(t_i)) \cdot 2^t$, *where* $\varepsilon(t_i) < 1$, *then* $F'(s,k)$ *defined using*

$$P'(s) = \max_{1 \leq j \leq \min(m,s-1)} \begin{pmatrix} P'(s-j) + \log_2(1 + 2^{t(s-1,j-1) - t(s-1,j)}) \\ -\log_2(1 - \varepsilon(t_i)) \end{pmatrix}$$

satisfy the property defined in (7).

Lemma 2: *The function* $R'(t)$ *defined in (20) satisfies* $2^t \geq R'(t) \geq (1 - \varepsilon(t_i)) \cdot 2^t$, *where* $\varepsilon(t_i) = 2^{-l} + 2^{-t_i}$.

2. ENCODING OF MDCT COEFFICIENTS

The EVRC-WB [4], which is a split band codec, uses code excited linear prediction (CELP) for coding the low band speech (0-4 kHz), and a low bit rate bandwidth extension method for the high band (4-7 kHz). To improve the performance of the codec for music-on-hold and other non-speech audio signal, a low band audio coding mode using the modified discrete cosine transform (MDCT) [4] of the linear predictive residual has been included in the standard. The block diagram of the audio coding mode is shown in Figure 1. Quantizing the residual MDCT coefficients is performed using the proposed FPC method. The 144 residual spectrum lines (only 144 coefficients of 160 are quantized) are quantized using 28 or 23 pulses depending on whether the coding is done for narrow band or wide band input, respectively.

The energy scaled version of residual lines X_k is coded such that:

$$m = \sum_{k=0}^{143} \left| \text{round}\{\gamma_m X_k\} \right|, \tag{22}$$

where γ_m is a global scale factor, m is the number of unit magnitude pulses, and the range 0 to 143 corresponds to the frequency range 0 to 3600 Hz. The value of γ_m used to achieve the above is determined iteratively by increasing or decreasing γ_m depending on whether the summation in (22) is less than or more than m.

Table 1: Subjective Performance of EVRC for Narrowband Music Signals at 8.5 kbps [5]

Codec	Source PCM	Speech Mode	Audio Mode
MOS	3.67	2.39	3.22

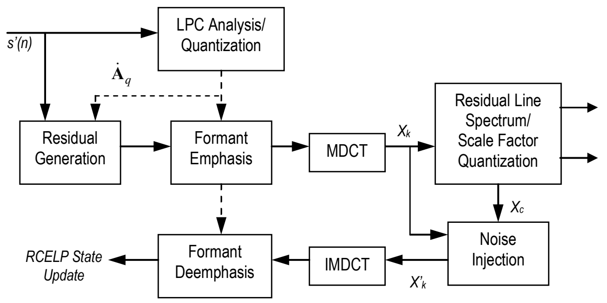

Figure 1: Block diagram of the Generic Audio coding mode of the EVRC-WB standard

3. RESULTS

Table 1 shows the improvement in subjective performance of the narrowband generic audio coding mode of EVRC using the proposed MDCT quantization technique for coding music signals relative to the speech coding mode at an equivalent bit rate (i.e., 8.5 kbps). These listening tests were conducted for the performance characterization of EVRC-WB codec for 3GPP2 [5].

Table 2 shows the complexity reduction associated with the proposed approximate function computation as compared to the precise function computation when these function are used in FPC. For different values of m and n, the associated number of bits M is given. For these examples, the frame length interval is 20 ms, which corresponds to a rate of 50 frames per second. The unit of measure for the complexity comparison is weighted millions of operations per second, or WMOPS. A computer simulation was used to produce an estimate of the complexity as it would be executed on a limited precision fixed point DSP. For these examples, each primitive instruction was assigned an appropriate weighting. For example, multiplies and additions, were given a weight of one operation, while primitive divide and transcendental (e.g., 2^x) operations were given a weight of 25 operations. From the table, it is easy to see that using $F'(s, k)$ provides significant complexity reduction over using $F(s, k)$, and that the proportional reduction in complexity increases and n and m increase. This complexity reduction is shown to be as high as two orders of magnitude for the $F(144, 60)$ case, but would continue to grow as n and m increase further. This is primarily due to the growth in precision of the operands that is required to carry out exact combinatorial expressions for $F(s, k)$. These operations prove to result in an excessive complexity burden and virtually eliminate factorial pulse coding as a method for coding vectors having the potential for large m and n. The proposed method solves these problems by requiring only single cycle low precision operations coupled with a small amount of memory storage to produce approximations of the complex combinatorial expressions required for this type of coding.

Table 2: Complexity comparison of *F'*(*s*,*k*) and *F(s,k)*

n	m	Bits	Using $F(s, k)$		Using $F'(s, k)$	
			Peak WMOP S	Avg WMOPS	Peak WMOPS	Avg WMOPS
54	7	35	0.44	0.32	0.09	0.07
144	28	131	24.50	16.45	0.51	0.37
144	44	180	76.45	46.65	0.96	0.64
144	60	220	150.00	83.25	1.50	0.90

4. CONCLUSIONS

A very low complexity method for approximation of combinatorial functions has been proposed. It has been shown that the approximate functions satisfy a property which preserves unique decode-ability of factorial packing encoding/decoding algorithm when these functions are substituted in place of standard combinatorial functions in FPC encoding/decoding algorithms. The low complexity computation of these functions enables use of FPC in encoding of 144 MDCT coefficients using 28 unit magnitude pulses for the audio coding mode of an EVRC-WB speech coding standard.

5. REFERENCES

[1] J.P. Ashley, E.M. Cruz-Zeno, U. Mittal, W. Peng, "Wideband coding of speech using a scalable pulse codebook," *IEEE Workshop on Speech Coding*, pp.148-150, Sept. 2000.

[2] A.C. Hung, E.K. Tsern, T.H. Meng, "Error-Resilient pyramid vector quantization for image compression," *IEEE Trans. on Image Processing*, pp.1373-1386, Oct. 1998.

[3] U. Mittal, J. P. Ashley, E. M. Cruz-Zeno, "Coding unconstrained FCB excitation using combinatorial and Huffman codes," *IEEE Workshop on Speech Coding,* pp. 129-131, Oct. 2002.

[4] "Enhanced Variable Rate Codec, Speech Service Options 3, 68, and 70 for Wideband Spread Spectrum Digital Systems," Document 3GPP2 C.P0014-C, Version 0.4, Sept. 2006.

[5] http://www.3gpp2.org