TECHNICAL RESEARCH REPORT

A New Deterministic Codebook Structure for CELP Speech Coding

by Yu-Hung Kao and John S. Baras

T.R. 92-135

A New Deterministic Codebook Structure for CELP Speech Coding

Yu-Hung Kao and John S. Baras
Electrical Engineering Department
and Systems Research Center
University of Maryland
College Park, MD 20742

Abstract

Low bit rate, high quality speech coding is a vital part in voice telecommunication systems. The introduction of CELP (1984) (Codebook Excited Linear Prediction) speech coding provided a feasible way to compress speech data to 4.8 kbps with high quality, but the formidable computational complexity required for real-time processing has prevented its wide application. Using the new deterministic codebook, we reduce the computational complexity of codebook search, which originally accounted for 2/3 of the computational complexity, to negligible. Based on this reduction, we produce an algorithm with complexity about 5 MIPS. It can be implemented in even inexpensive DSP chips, while maintaining the same high quality. In addition to extremely simple encoding and decoding schemes, this codebook also provides optimal error tolerance and it doesn't require codebook storage. We hope that this contribution can finally make CELP speech coding a widely applicable and practical technology.

1 Introduction

It is widely accepted that the speech residuals (after short and long term predictions) are Gaussian distributed, so stochastic codebooks (generated by a Gaussian process) are used to predict speech residuals. However, since the stochastic codebooks are generated randomly, there are no special structures to organize them, so we need to use exhaustive search to find an optimum codebook vector. Although some other researchers (NSA [3]) have proposed overlapped codebook to reduce the complexity of convolution (in doing perceptual weighting) by end-point correction, the computational complexity is still high (8 MIPS). Furthermore, the use of overlapped codebook is an approximation and it degrades quality. It is not exact as in the pitch codebook case. The fundamental question we investigated is : *Is it necessary to use an un-structured stochastic codebook? Is it possible to construct a deterministic codebook and by its regular structure to find some efficient ways to search the codebook instead of exhaustive search?* We found that stochastic codebook is not necessary, and there exists a codebook structure that provides not only tremendous savings in search time but also other nice properties. This paper is organized as follows: Section 2 explains the motivation behind the new codebook structure. Section 3 describes the fast search algorithm. Section 4 introduces a modification of the codebook, which provides even better performance. Based on the same principle, the readers can construct their own

modifications to achieve the best performance under different situations. Section 5 provides performance comparisons and summary of the property of the new codebook. Section 6 gives our conclusion.

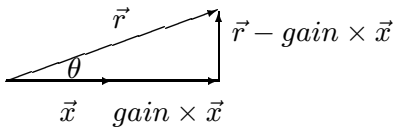
2 The Motivation Behind the Deterministic Codebook

To find the answer of whether a stochastic codebook is necessary, let us discuss the physical meaning of finding an optimum excitation vector in the codebook : In CELP, for a given *speech residual* vector $\vec{r}$, we want to find a codebook vector $\vec{x}$, which, *after scaling*, will produce minimum square error from the speech residual vector $\vec{r}$. Because of the scaling factor, the criterion is *not* the same as *nearest neighbor* in the Euclidean distance sense. To see this, suppose that we have a speech residual vector $\vec{r}$ and a codebook vector $\vec{x}$. The criterion is equivalent to maximizing

$$\frac{|\vec{r}\cdot\vec{x}|^2}{|\vec{x}|^2} = \frac{|\vec{r}|^2|\vec{x}|^2 cos^2\theta}{|\vec{x}|^2} = |\vec{r}|^2 cos^2\theta$$

$$\forall \vec{x} \in codebook$$

Since $\vec{r}$ is fixed in the search, we are actually maximizing $cos^2\theta$. To maximize $cos^2\theta$ is equivalent to minimize $sin^2\theta$, thus minimizing the difference between these two vectors, $|\vec{r} - gain \times \vec{x}|$. Therefore the criterion is to maximize $cos^2\theta$, since scaling can do the rest. To reveal more insight into this problem, suppose we want to maximize $cos\theta$. To maximize $cos\theta$ means to find a codebook vector which is most *parallel* to the speech residual vector. The above discussion is best illustrated by the following diagram:



By the above argument, we conclude that the criterion for a *good* codebook is : *it must span the n-dimensional sphere as uniformly as possible*. Given a fixed number of vectors, they will have the best *directional representation ability* if they are uniformly distributed over the n-dimensional sphere. We now see clearly that it is *not* necessary to use an un-structured stochastic codebook; rather, we can construct a codebook which can span the n-dimensional sphere *more uniformly* than a randomly generated stochastic codebook. This means we can even construct a codebook which is actually better than a stochastic codebook. We call such a codebook a *deterministic codebook*. Such codebooks have been proposed for CELP coding in earlier works [1,2]. However, our codebook is substantially different.

The reason why randomly generated stochastic codebooks are widely used is that when we plot the histogram of the speech residuals, we can see that they are approximately Gaussian distributed. So an i.i.d. Gaussian process can be used to generate the codebook, and fortunately, the result comes out reasonably good. When we construct our deterministic codebook, we must take this approximate *Gaussian distribution* property into consideration. We shall see later that this step is also necessary in order to bring down the codebook size from impractically large to somewhat manageable.

We know that a good codebook needs to span the n-dimensional sphere uniformly. To simplify things, we restrict the elements of our codebook vectors to be ternary, i.e. -1, 0, and 1. Note that all the simplifications made here have always been justified by experimental results; and if this is not considered satisfactory, we may argue that in the search process stated above, the *direction* of a vector is used as the matching criterion rather than its *exact location*. So the ternary restriction should be able to retain the *directional* representation ability of each vector.

In the NSA CELP standard (Federal 1016) [3], the vector length is n = 60. This means that even with the ternary restriction, there are $3^{60} - 1$ (*zero vector*) possible vectors in the 60-dimensional space. That is of course too large! To achieve 4.8 kbps encoding (the original speech is quantized to 14 bits/sample, sampling rate 8K, so it is 112 kbps), we only have 9 bits reserved for the codebook index, which means that our codebook size can only be 2^9. To bring down the size, first we consider the approximate Gaussian distribution property of speech residuals. We know that most of the residuals are fairly small, so we can let a fair amount of codebook vectors' components be zero and thus reduce the size of the codebook. But how many? There is no way to derive the cutoff threshold theoretically. Based on experiments, the NSA team used [3] a 77% zero codebook and have reported a fairly good performance. So we follow this suggestion (make it 80% zero, because 77% of 60 is not an integer) and let our vectors have 48 zeros out of the total 60 components, and the remaining 12 components are 1's or -1's. After these simplifications, the codebook size becomes

$$S_w^n = 2^w \times \left(\begin{array}{c} n \\ w \end{array} \right) = \frac{2^w \times n!}{(n-w)! \times w!} = \frac{2^{12} \times 60!}{48! \times 12!}$$

where n is the dimension, w is the weight. As we can see, this size is still too large, much larger than the desired 2^9.

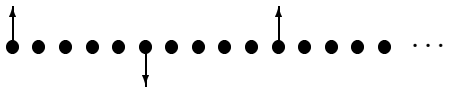
How can we further reduce the size of the codebook? The residuals are time sequences, and we know that human ears are relatively insensitive to phase shifts in the speech waveform. So, the positions of those 12 1's and -1's may not be that important. *Why not choose 12 fixed positions to put those non-zero spikes, and only play with the signs of them!* After doing so, the size is reduced to 2^{12}, quite close to the ideal 2^9. We still need some more restrictions to reduce the codebook size!

By now, it seems that we have run out of intelligent and rationally justified ways to come up with other restrictions. The next step is entirely empirical: let us put some restrictions about the possible combinations of these 1's and -1's.

Now we have two decisions to make : to choose 12 non-zero positions out of the total 60, and set some restrictions on the combinations of those 1's and -1's.

First let us consider how to choose 12 non-zero positions out of the 60 vector length. Intuitively, it seems reasonable to place them uniformly over the 60 positions : only elements with index 5n are non-zero , i.e. *X0000X0000X0000 ...* , each *X* can be either 1 or -1. This configuration has critical significance when we consider the calculation of *perceptual weighting*. We shall see this a little later.

Now we have a 60-dimensional vector which has 12 spikes uniformly distributed. It looks like this :



The spikes can flip up and down to form the 2^9 vectors needed. We may see this vector this way : since we want to use it to represent the *noise like* speech residual, by flipping these spikes up and down we will have *good shaping ability*, at least better than randomly generated vectors.

Now we have 2^{12} vectors in our codebook. We further need to put some restrictions on the combinations of 1's and -1's. Let us first partition the vector into 3 equal length sub-vectors. The length of each sub-vector is 20 and there are 4 non-zero elements in each of them. We pose the restriction that we allow only even numbers of -1's out of these 4 non-zero elements. So, now we can have 4 1's (1 combination), 4 -1's (1 combination), and 2 1's, 2 -1's (6 combinations) for each sub-vector. Each sub-vector has 8 combinations, that means each vector has $8^3 = 2^9$ combinations. Finally, we get a codebook size of 2^9, that takes 9 bits for the codebook index ; exactly what we need!

It is important to note that because this is a deterministic codebook, *we don't even need to store the codebook,* the codebook index, alone, already specifies each vector exactly.

The arguments above are not easily justified on a theoretical basis. The problem is that we are trying to densely "pack" a 60-dimensional sphere with only 2^9 vectors; and this number is definitely not enough to do a good job. We've already taken the Gaussian distribution into consideration, which is the only characteristic exploited by a randomly generated stochastic codebook. Besides that, we have just tried to make these vectors spread more uniformly. The good behavior of our codebook is heuristically suggested by extensive experiments [4], indicating that the *size* of the excitation signal sphere is of secondary importance compared with the long term predictor.

3 A Fast Algorithm to Compute Inner Products

Recall that we need to calculate the measure:

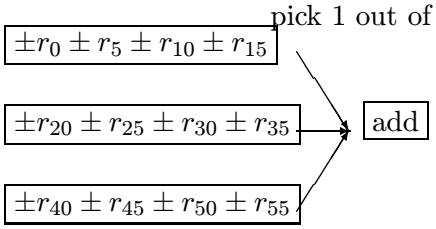
$$\frac{|\vec{r} \cdot \vec{x}|^2}{|\vec{x}|^2} \ , \ \forall \vec{x} \in \textit{codebook}$$

Which means we need to calculate the square of the inner product $|\vec{r} \cdot \vec{x}|^2$ and the energy $|\vec{x}|^2$. To calculate the 2^9 inner products of the speech residual vector with respect to each of our codebook vectors, we note that there are only 1's and -1's in the codebook vectors, so there is actually no need for multiplications. We only need to pick the right components in the speech residual vector, and then add or subtract them. We will come back to perceptual weighting later.

Using appropriate combinations of different terms, we can calculate all the 2^9 inner products with an extremely small amount of operations. Suppose the speech residual is $\vec{r}$, and the codebook vector is $\vec{x}$. Since in our codebook vectors, only elements with index 5n $(x_0, x_5, x_{10}, \dots)$ are non-zero, and they are all 1's and -1's, we only need speech residual elements with index 5n $(r_0, r_5, r_{10}, \dots)$ to calculate all the inner products. For each of the 3 sub-vectors, we calculate 8 sums corresponding to 8 combinations of codebook sub-vectors. The 8 sign combinations for each sub-vector that we use are:

	r_0	r_5	r_{10}	r_{15}
0	+	+	+	+
1	+	+	-	-
2	+	-	+	-
3	+	-	-	+
4	-	+	+	-
5	-	+	-	+
6	-	-	+	+
7	-	-	-	-

For each sub-vector we have 8 sums. If we pick one sum from each subvector we have 3 sums, and by adding these 3 sums we have one inner product. There are 8^3 ways to pick 3 sums from 3 sub-vectors, and these ways provide exactly the 2^9 inner products we need! This is illustrated in the following schematic :



We now consider perceptual weighting. Let the impulse response $h_0, h_1, h_2, \ldots$ characterize the partical filter, i.e. FIR filter. We need to complete convolutions of the impulse response $\vec{h}$ with each of our codebook vectors. Since all of our codebook vectors have 4 zeros between two nonzero elements, if we reduce the impulse response sequence length to 5, keep only $h_0, h_1, h_2, h_3, and\ h_4$, the codebook vector after perceptual weighting should look like :

$$\underbrace{h_0 h_1 h_2 h_3 h_4}_{+or-}\ \underbrace{h_0 h_1 h_2 h_3 h_4}_{+or-}\ \underbrace{h_0 h_1 h_2 h_3 h_4}_{+or-}\ \ldots\ (60\ h_i's)$$

With different sign combinations, but each group of $(h_0, h_1, h_2, h_3, h_4)$ should be of the same sign. By inspecting $\vec{h}$, we see that over 80% of its energy is contained in the first 5 components, thus this approximation should not cause too much distortion.

Keeping the same structure as before, we perform the following substitutions :

$$r_0 = r_0 h_0 + r_1 h_1 + r_2 h_2 + r_3 h_3 + r_4 h_4$$

$$r_5 = r_5 h_0 + r_6 h_1 + r_7 h_2 + r_8 h_3 + r_9 h_4$$

$$\vdots$$

$$r_{55} = r_{55} h_0 + r_{56} h_1 + r_{57} h_2 + r_{58} h_3 + r_{59} h_4$$

We see that we can still get all 2^9 inner products, with pereptual weights, using an extremely small number of operations.

We also need to calculate the energy for each codebook vector after perceptual weighting, and fortunately, as the codebook vectors after perceptual weighting all look alike, as described above, their energies are all the same and equal to:

$$12 \times \sum_{i=0}^{4} h_i^2$$

Because they are all the same, we don't even have to calculate them.

The spirit of this algorithm is : *Since all the codebook vectors are just different combinations of signs, the "components" in all inner products are the same : it is not necessary to re-compute these components, it is enough to play with the combinations of signs and we can get all the inner products.*

We now compute the exact number of operations needed. First, we need $5 \times 12 = 60$ MUL and $4 \times 12 = 48$ ADD to compute the above $r_0, r_5, \ldots, r_{55}$. Then we need $4 \times 8 \times 3 = 96$ ADD or SUB to compute 8(combinations) $\times$ 3(sub-vectors) = 24 terms. Finally we need $2 \times 512 = 1024$ ADD to compute the 512 inner products. *The total is only 1228 operations to get 512 inner products!* And only 60 of them are MUL, others are ADD or SUB. $1228/7.5ms = 0.16$ MIPS. Compared with the brute force search requiring 80 MIPS (512×60 codebook), this represents *an improvement of 500.* Compared with overlapped codebook complexity of 8 MIPS, this represents *an improvement of 50.* Originally the codebook search dominated the complexity of CELP analysis, now the computations needed for codebook search is negligible compared to pitch search.

Actually the codebook doesn't have to be so restrictive (as stated above) to benefit from the above algorithm. As long as the non-zero positions in all codebook vectors are fixed, and their absolute values are the same, which means *the only difference among all codebook vectors is different sign combination, then we can use the above algorithm.*

4 An Improvement

Based on the discussions in the previous sections, we can interpret the meaning of those 12 spikes in each codebook vector $\vec{x}$ as dividing the speech residual $\vec{r}$ into 12 "clusters", namely, $r_0 \cdots r_4$ is the first cluster, $r_5 \cdots r_9$ is the second cluster, $\ldots$ *etc.* And the decision of +1 or -1 of each spike simply determines whether the cluster is a "positive" cluster or a "negative" cluster. Because we have only 9 bits allocated for the codebook index and there are 12 spikes in our codebook vectors, we have to pose the restriction that only even numbers of -1's are allowed in each sub-vector in order to restrict the number of possible sign combinations to be 2^9 instead of 2^{12}. Obviously this restriction will limit the ability for each of those 12 spikes to determine the signs of those 12 clusters, and we suspect that this limitation will increase the encoding error. It would be preferrable if we can encode the sign of each cluster freely. Recall that we made the decision to use 12 spikes because NSA used a 77% zero stochastic codebook, and 12 spikes out of 60 vector elements is 80%, which is close to 77%. What if we use 9 spikes? Then there are 9 clusters and we can decide the sign of each cluster freely.

When we have the freedom to determine the sign of each cluster freely, we don't even have to compute inner products as in the 12 spikes case (this will become clear shortly). Not only the computation is reduced, but the resulting codebook has improved error tolerance as well. Here is the full description of this improved codebook and its encoding and decoding schemes:

Since 9 is not a factor of 60, we cannot have equal size clusters as in the case of 12 spikes. However, it doesn't matter if each cluster has the same size, we can still spread these 9 spikes as uniformly as possible in the 60 positions. Naturally, we choose the cluster sizes to be 7, 7, 6, 7, 7, 6, 7, 7, 6; these 9 clusters add up to 60.

The codebook vectors after perceptual weighting look like:

$$\underbrace{h_0h_1h_2h_3h_4h_5h_6}_{+or-}\underbrace{h_0h_1h_2h_3h_4h_5h_6}_{+or-}$$

$$\underbrace{h_0h_1h_2h_3h_4h_5}_{+or-}\ \ldots (60\ h_i's)$$

Note that in the 12 spikes case we use $h_0 \cdots h_4$ as perceptual weighting impulse response, here we use either $h_0 \cdots h_6$ or $h_0 \cdots h_5$ depending on the size of each cluster.

Recall that we need to calculate the measures:

$$\frac{|\vec{r} \cdot \vec{x}|^2}{|\vec{x}|^2}\ ,\ \forall \vec{x} \in codebook$$

where $\vec{r}$ is the perceptually weighed speech residual, $\vec{x}$ is the perceptually weighed codebook vector. And we want to choose the $\vec{x}$ that results in the largest value. Note that all $\vec{x}'s$ have the same energy:

$$6 \times \sum_{i=0}^{6} h_i^2 + 3 \times \sum_{i=0}^{5} h_i^2$$

so we don't have to compute it. Regarding the inner products of $\vec{r}$ and $\vec{x}$, we only have to compute the following 9 terms:

$$r_0h_0 + r_1h_1 + r_2h_2 + r_3h_3 + r_4h_4 + r_5h_5 + r_6h_6$$

$$r_7h_0 + r_8h_1 + r_9h_2 + r_{10}h_3 + r_{11}h_4 + r_{12}h_5 + r_{13}h_6$$

$$r_{14}h_0 + r_{15}h_1 + r_{16}h_2 + r_{17}h_3 + r_{18}h_4 + r_{19}h_5$$

$$\vdots$$

$$r_{54}h_0 + r_{55}h_1 + r_{56}h_2 + r_{57}h_3 + r_{58}h_4 + r_{59}h_5$$

We can add (or subtract, depending on the signs of those 9 spikes) these 9 terms to form all the 2^9 inner products. But it is not necessary. If we choose the signs of the 9 spikes in the codebook vector to be the same as the 9 terms computed above, then the inner product will be the largest. So we can then determine the 9 bits in the codebook index according to the signs of these 9 terms. Set the bit to 1 if the corresponding term is positive, set to 0 if negative. This completes the encoding of the codebook index. In the decoding, it is even easier: set the spike to 1 if the corresponding bit (in the codebook index) is 1, set to -1 if 0.

To compute the exact number of operations needed, we only need to calculate the 9 terms mentioned above: 60 MUL and 51 ADD. That is only 111 operations or $111/7.5ms = 0.015$ MIPS. With a complexity this low, it really doesn't make sense to compute the MIPS. The real benefit of this improved codebook is its error tolerance property. Because each bit in the codebook index encodes 1/9 of the codebook vector, if 1 bit is flipped, only 1/9 of the codebook vector will be decoded incorrectly. However in the stochastic codebook case, if 1 bit is flipped, the whole vector will be different.

5 Results and Advantages of the Proposed Codebook

With all the nice properties mentioned above for the proposed codebook, it is important to know how the quality of the encoded speech using this codebook compared to that using the traditional stochastic codebook. For subjective speech quality tests, we submitted the 12 spikes codebook to NSA, and the score reported by NSA is shown below (we compare with the NSA standard because we follow the NSA standard except for the codebook) :

	DAM	MOS
NSA Quiet	62.9 ± 1.0	$3.25 \pm .11$
Office	54.4 ± 1.0	$2.71 \pm .07$
E4B	53.3 ± 0.9	$2.59 \pm .06$
12 spikes Quiet	60.3 ± 1.0	$3.14 \pm .09$
Office	52.1 ± 1.0	$2.56 \pm .07$
E4B	51.7 ± 0.9	$2.52 \pm .06$

As shown in the scores, the 12 spikes codebook scores a little lower than NSA stochastic codebook. However, we believe the 9 spikes codebook will score better than the stochastic codebook for the reasons mentioned throughout this paper and also based on the S/N ratios we obtained from processing 12 sentences including male, female and children.

	NSA	12 spikes	9 spikes
child	18.30	19.81	21.01
female	17.29	14.21	21.65
male	11.66	8.74	9.36
female	10.93	10.25	9.90
male	11.95	11.63	12.16
male	10.75	9.18	11.27
male	10.36	9.06	11.47
child	17.62	16.26	18.36
male	13.73	13.21	13.79
female	19.12	18.31	20.53
female	12.21	12.29	12.17
female	19.30	18.73	23.36
average	14.43	13.47	15.42

As shown in the above table, the 9 spikes codebook provides the best averaged S/N ratio.

For objective properties, i.e. speed, memory requirement, and error tolerance, our codebook is consistently much better (here we refer to the 9 spikes codebook).

1. Search speed is 5000 times faster than stochastic codebook, and 500 times faster than overlapped codebook. Reducing the computational complexity of CELP from 15 MIPS to 5 MIPS, which can be handled by even low end DSP chips.

2. No need for codebook storage. The codebook vector can be completely decided by the codebook index. For a 512-size codebook, overlapped codebook needs 1082 words, and

stochastic codebook needs $60 \times 512 = 30720$ words. Considering the implementation of a stand-alone coding unit, this saving can be a crucial edge.

3. Improved error tolerance. For a stochastic codebook, if one bit of codebook index is flipped, it represents a totally different codebook vector; that means the quality degrades miserably. But in our codebook, if one bit of codebook index is flipped, *only* 1/9 of the codebook vector components are different, we still have the other 8/9 components correct; that means the quality degrades gracefully.

6 Conclusions

We found that although speech residuals are approximately Gaussian distributed, a stochastic codebook is not necessary for higher quality CELP coding. A *ternary, fixed nonzero position* deterministic codebook can encode speech residuals as well. It provides not only tremendous savings in search time but also improved error tolerance. Moreover, it doesn't require codebook storage. This structure is very easy to be modified to fit with different speech frame sizes and different encoding bit allocations. We expect to see this structure applied to other cases and provide the same good performance results.

7 Acknowledgement

We would like to thank T. Tremain and V. Welch of NSA for their help and suggestions throughout this work. The University of Maryland has applied for a patent for this algorithm. For further information please contact Mr. G. Gillespie at (301) 405-7507.

8 References

1. M. A. Ireton and C. S. Xydeas, "On improving vector excitation coders through the use of spherical lattice codebooks (SLC's)" ICASSP 1989, 57 - 60.

2. C. Lamblin, J. P. Adoul, and S. Morissette, "Fast CELP coding based on the Barnes-Wall lattice in 16 dimensions" ICASSP 1989, 61 - 64.

3. Thomas E. Tremain, Joseph P. Campbell, JR, and Vanoy C. Welch, "A 4.8 K bps Code Excited Linear Predictive Coder" U.S. DoD.

4. R. C. Rose and T. P. Barnwell, III, "Design and Performance of an Analysis-by-Synthesis class of predictive speech coders" IEEE Trans. on ASSP, Vol. ASSP-38, NO. 9, Sept. 1990, 1489 - 1503.