CANADIAN CONFERENCE ON ELECTRICAL AND COMPUTER ENGINEERING
CONGRES CANADIEN EN GENIE ELECTRIQUE ET INFORMATIQUE
Vancouver, B.C., Canada, November 3-4 novembre 1988

DESIGN OF A 16 kbit/s VECTOR ADPCM SPEECH CODING ALGORITHM

Lloyd Watts and Vladimir Cuperman

Department of Engineering Science
Simon Fraser University, Burnaby, B.C. V5A 1S6

Abstract - A Vector ADPCM speech coding algorithm for low complexity, low delay (0 - 5 ms), 16 kbit/s applications is described. An Analysis-by-Synthesis configuration is used to allow a Vector Quantizer to operate with a scalar adaptive linear predictor. The use of a polarity crosscorrelation update term in the predictor update equation is found to introduce a significant performance degradation as compared to the full gradient algorithm when used with a Vector Quantizer. In addition to the usual all-pole stability checks, constraints must be applied to the zeroes of the predictor to ensure stability of the prediction residual. Gain-Adaptive Vector Quantization and Postfiltering are used to improve the speech quality.

1. INTRODUCTION

This paper describes Vector ADPCM, a low delay (< 1 ms) speech coding algorithm for 16 kbit/s applications based on the introduction of a Vector Quantizer into an ADPCM configuration. Previous attempts to combine Vector Quantization and Linear Prediction have resulted in Vector Predictive Coding [1,2], a vector generalization of DPCM, in which the prediction residual of a Vector Predictor is Vector Quantized. The disadvantage of this scheme is the low prediction gain of the Vector Predictor, due to the decreasing speech autocorrelation for increasing lag.

The Vector ADPCM algorithm [3], employs a Vector Quantizer combined with an adaptive scalar predictor similar to that found in the CCITT ADPCM standard [4] in an Analysis-by-Synthesis configuration [5,6]. The present paper describes design aspects of the the VADPCM algorithm, including codebook design and design of the predictor adaptation algorithm to account for the presence of the Vector Quantizer in the feedback loop. In particular, the use of a polarity crosscorrelation update term in the predictor update equation is found to introduce a significant performance degradation as compared with the full gradient algorithm when used with a Vector Quantizer. In addition to the usual all-pole stability checks, constraints must be applied to the zeroes of the predictor to ensure stability of the prediction residual.

Pole-zero Postfiltering [7] and Gain-Adaptive Vector Quantization [10] based on fixed lognorm prediction are used to improve the subjective performance.

2. BASIC CONFIGURATION

Figure 1(a) shows a block diagram of the Analysis-by-Synthesis (A-S) configuration, containing a Vector Quantizer and backward-adaptive pole-zero Scalar Predictor. The

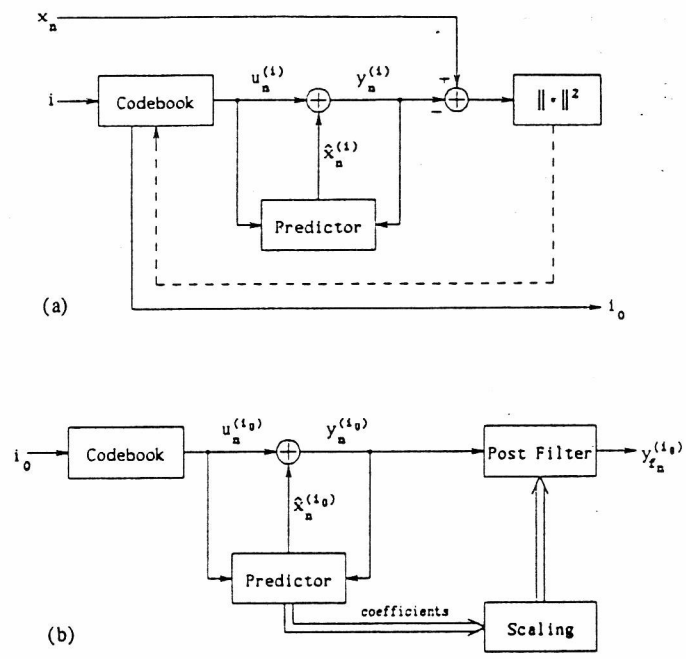

FIGURE 1. (a) Analysis-by-Synthesis Configuration. (b) Receiver with Postfiltering.

receiver configuration including Postfiltering is shown in Figure 1(b).

The A-S configuration is necessary to allow the Vector Quantizer (VQ) and Scalar Predictor to operate together, since the VQ introduces a block delay in the encoding process, while the Scalar Predictor requires sample-by-sample update of the quantized prediction residual. The nearest-neighbor codebook search proceeds as follows (see Figure 1(a)): for a trial codevector i, the codevector elements $u_n^{(i)}$ are processed through the predictor filter to produce the predicted samples $\hat{x}_n^{(i)}$. The predictor equation is

$$\hat{x}_n^{(i)} = \sum_{j=1}^{p} a_j^{(i)} y_{n-j}^{(i)} + \sum_{j=1}^{z} b_j^{(i)} u_{n-j}^{(i)} \tag{1}$$

where p is the number of poles, z is the number of zeroes. $y_{n-j}^{(i)}$ and $u_{n-j}^{(i)}$ may not depend on i if the index $n-j$ refers to samples of a previous vector. This fact can be exploited to reduce computational load by precomputing the component due only to the previous vectors (the Zero-Input-Response) [3].

The reconstructed samples are generated by adding the predicted samples to the codevector elements:

$$y_n^{(i)} = \hat{x}_n^{(i)} + u_n^{(i)} \tag{2}$$

and the squared reconstruction error for the codevector is

$$D^{(i)} = \sum_{n=n_0}^{n_0+k-1} (x_n - y_n^{(i)})^2 \tag{3}$$

where k is the vector dimension and n_0 is the sample number of the first sample in the vector. This procedure is repeated for $i=1,2,...,N$, where N is the number of codevectors in the codebook, and the codevector which minimizes the squared reconstruction error is selected:

$$i_0 = ARGMIN_i\ [D^{(i)}]. \tag{4}$$

3. PREDICTOR ADAPTATION

The predictor adaptation algorithm is based on the gradient algorithm [11], resulting in the following adaptation equations:

$$h_i(n+1) = \lambda_{h_i} h_i(n) + \frac{\alpha_{h_i}}{\sigma_y^2} u(n) y(n-i) \tag{5}$$

$$g_j(n+1) = \lambda_{g_j} g_j(n) + \frac{\alpha_{g_j}}{\sigma_y^2} u(n) u(n-j) \tag{6}$$

for $i = 1,2,...,p$ and $j = 1,2,...,z$. Dividing by the variance of the reconstructed signal results in a predictor update term which is independent of the variance of the input signal. In order to track the local statistics of the input signal, an exponentially weighted time average of the variance is used:

$$\sigma_y^2(n) = \delta\sigma_y^2(n-1) + (1-\delta)y^2(n). \tag{7}$$

The predictor "freezes" when the variance $\sigma_y^2(n)$ drops below a threshold ε.

A common complexity reduction technique in the scalar quantizer environment is to use the "sign" algorithm, which uses polarity crosscorrelations in the predictor update term:

$$h_i(n+1) = \lambda_{h_i} h_i(n) + \alpha_{h_i}\ sgn\ u(n)\ sgn\ y(n-i) \tag{8}$$

$$g_j(n+1) = \lambda_{g_j} g_j(n) + \alpha_{g_j}\ sgn\ u(n)\ sgn\ u(n-j) \tag{9}$$

for $i = 1,2,...,p$ and $j = 1,2,...,z$. This algorithm works well in the scalar quantizer environment, since $sgn\ u(n) = sgn\ e(n)$, and thus, in the absence of transmission errors, the predictor is guaranteed to move in the right direction (i.e. the same direction that would be indicated if the unquantized values were used to generate the update term). However, in the Vector Quantizer environment, $sgn\ u(n)$ may not be the same as $sgn\ e(n)$, particularly when $e(n)$ is small, and thus the predictor may make a large step in the wrong direction, resulting in a significant performance degradation with the "sign" algorithm. The gradient algorithm in equations (5) and (6) ensures that when the predictor moves in the wrong direction due to quantization noise in a high-variance waveform segment, it will at least take small steps.

A further performance improvement is possible by using a scaling factor with a better match to the crosscorrelation product:

$$h_i(n+1) = \lambda_{h_i} h_i(n) + \frac{\alpha_{h_i}}{\sigma_y \sigma_u} u(n) y(n-i) \tag{10}$$

$$g_j(n+1) = \lambda_{g_j} g_j(n) + \frac{\alpha_{g_j}}{\sigma_u^2} u(n) u(n-j) \tag{11}$$

for $i = 1,2,...,p$ and $j = 1,2,...,z$.

4. STABILITY CONDITIONS.

In order to ensure stability from $u(n)$ to $\hat{x}(n)$, it is common to impose stability constraints on the poles of the predictor. However, it is also neccesary to keep the zeroes of the predictor inside the unit circle to ensure stability from $y(n)$ to $\hat{x}(n)$, as can be seen from the transfer function:

$$\frac{\hat{X}(z)}{Y(z)} = \frac{H(z) + G(z)}{1 + G(z)}. \tag{12}$$

An example of zero-related prediction residual instability is shown in Figure 2, in the 2-pole 3-zero open-loop case (no quantizer), with stability checks on the poles but not on the zeroes of the predictor. The large prediction error burst in Figure 2(c) coincides with a period in which the zeroes are outside the unit circle.

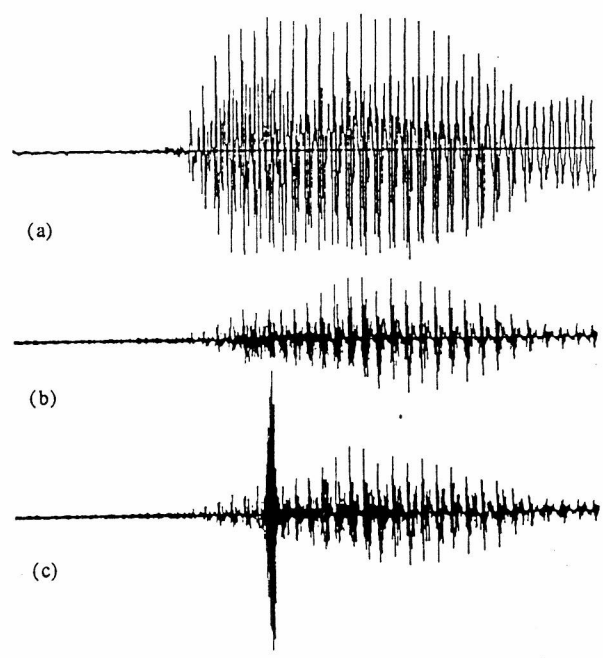

FIGURE 2. Example of unquantized prediction residual instability arising from zeroes outside the unit circle. (a) Original speech. (b) Residual from 2-pole 3-zero predictor with stability checks on poles and zeroes. (c) Residual from 2-pole 3-zero predictor with stability checks only on poles.

It should be noted that the leak factors in the predictor update equation have an effect on this zero-related instability; in the example in Figure 2, λ_{g_j} is set to unity for $j = 1,2,3$. However, when $\lambda_{g_j} = \frac{255}{256}$ for $j = 1,2,3$, there were no observed occurrences of zero-related instability. This is because the leak factors limit the growth rate of the all-zero predictor coefficients. However, while such a leak factor seems to help in practice, it is no guarantee of stability.

In our system, stability checks on both the poles and the zeroes of the predictor are used. The inequalities which describe the constraints on the all-pole coefficients for up to order 3 are [12]:

$$\begin{aligned} a_1 + a_2 + a_3 &< 1 \\ -a_1 + a_2 - a_3 &< 1 \\ a_3(a_3 - a_1) - a_2 &< 1 \\ |a_3| &< 1 \end{aligned} \tag{13}$$

Constraints on the all-zero coefficients follow by substituting $-b_j$ for a_j for $j = 1,2,3$ in (13).

To maintain low complexity, provide good performance, and ensure stability of $\hat{x}(n)$ and $y(n)$, our work focused on a predictor with 3 poles and 3 zeroes.

5. CODEBOOK DESIGN.

In the codebook training phase, the prediction residuals

$$e_n^{(i_0)} = x_n - \hat{x}_n^{(i_0)}. \tag{5}$$

are grouped into vectors of the form $\left[e_n^{(i_0)}\right]_{n=n_0}^{n_0+k-1}$ and clustered using the LBG algorithm [8].

Training is performed in two phases [1]. Phase I is the "open-loop" training phase, in which the predictor operates on unquantized inputs x_n and e_n, rather than the quantized inputs y_n and u_n, as in equation (1). The codebook is grown using the usual splitting techniques [13], with 5 iterations following each new codebook split, and 20 iterations following the final codebook split. Phase II is the "closed-loop" training phase, in which the predictor operates as in equation (1). The codebook is trained in this fashion for 20 iterations.

The fact that the prediction residuals are clustered and used in the next training iteration dramatically increases the importance of predictor stability in the Vector Quantizer environment (section 4), since a predictor which goes unstable in one iteration can result in divergent codebook training.

In order to ensure that periods of silence are faithfully reconstructed, it is possible to force the elements of one codeword to exactly zero. This results in a negligible SNR reduction but a significant subjective improvement.

6. PERFORMANCE ENHANCEMENTS.

Postfiltering is an effective method of improving the subjective quality of the coded speech [7]. The postfilter is derived simply by scaling the coefficients of the Scalar Predictor [7,9].

A further performance improvement is possible with Gain-Adaptive Vector Quantization (GAVQ) [10], in which a gain-normalized codebook is used. The lognorm of the vector (logarithm of the vector norm) under consideration is predicted using a linear combination of the previous M reconstructed vector lognorms, as shown in Figure 3. Lognorms are used for better dynamic range control and results in a linear predictor configuration. The optimal lognorm predictor coefficients are determined from the autocorrelation of the clean prediction residual lognorm sequence for a given scalar predictor and vector dimension.

7. SIMULATION RESULTS.

Simulation results presented here are based on a speech file consisting of 16 seconds (128,000 samples) of low-pass filtered speech, containing phonetically balanced sentences spoken by two female and four male talkers.

Table 1 shows in-sequence simulation results (same speech file used for coding and training). We have focused on vector dimensions 3 and 4 in order to provide good

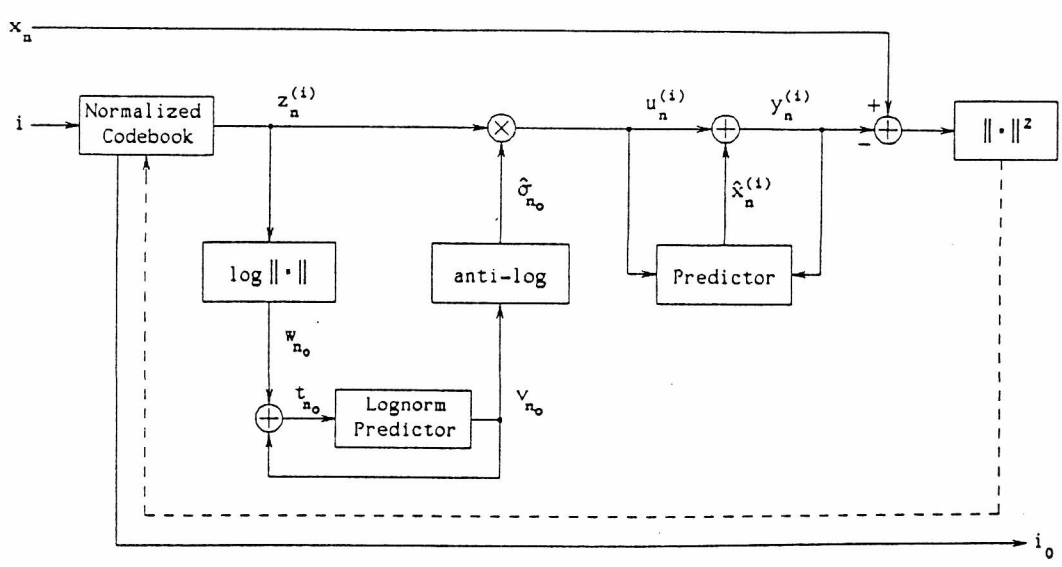

FIGURE 3. Analysis-by-Synthesis Configuration with Gain-Adaptive Vector Quantizer.

k	Poles, Zeroes	GAVQ	SNR (dB)	SEGSNR (dB)
3	2p6z	no	20.1	16.9
3	3p3z	no	20.2	16.9
4	3p3z	no	21.0	17.8
3	3p3z	yes	20.3	18.4
4	3p3z	yes	21.1	19.3

TABLE 1. Performance for various algorithms.

speech quality with low complexity. Postfiltering is found to provide a substantial improvement in subjective speech quality. All Gain-Adaptive Vector Quantization results are based on a fixed 10th-order lognorm predictor. The best results are obtained using Gain-Adaptive Vector Quantization, with the 3-pole, 3-zero predictor, in conjunction with pole-zero postfiltering. Informal tests indicate very good subjective quality.

8. CONCLUSIONS.

A low complexity low delay speech coding algorithm with good speech quality at 16 kbit/s has been presented, based on the introduction of a Vector Quantizer into the standard ADPCM algorithm, in an Analysis-by-Synthesis configuration. Pole-Zero Postfiltering and Gain-Adaptive Vector Quantization are found to improve the speech quality significantly. Recordings of the processed speech will be presented at the conference.

8. REFERENCES

[1] V. Cuperman, A. Gersho, "Vector Predictive Coding of Speech at 16 kbit/s", IEEE Trans. Comm., vol. COM-33, pp. 685-696, July 1985.

[2] P. C. Chang, R. M. Gray, "Gradient Algorithms for Designing Predictive Vector Quantizers", IEEE Trans. ASSP, VOL. ASSP-34, No. 4, pp. 679-690, August 1986.

[3] L. Watts, V. Cuperman, "A Vector ADPCM Analysis-by-Synthesis Configuration for 16 kbit/s Speech Coding", To be published in Proc. IEEE Global Telecommunications Conference, Hollywood, Florida, November 1988.

[4] CCITT Recommendation G.721, "32 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)," CCITT Red Book, Fascicle III.3, pp. 125-159, October 1984.

[5] M. R. Schroeder, B. S. Atal, "Code-Excited Linear Prediction (CELP); High-Quality Speech at Very Low Bit Rates", Proc. IEEE Intl. Conf. on Acous., Speech, and Signal Proc., pp. 25.1.1-25.1.4, Tampa, March 1985.

[6] J.-H. Chen, A. Gersho, "Vector Adaptive Predictive Coding of Speech at 9.6 kbit/s", Proc. IEEE Intl. Conf. on Acous., Speech, and Signal Proc., Tokyo, Japan, April 1986.

[7] N. S. Jayant, "ADPCM Coding of Speech with Backward-Adaptive Algorithms for Noise Feedback and Postfiltering", Proc. IEEE Intl. Conf. on Acous., Speech, and Signal Proc., Dallas, Texas, April 1987.

[8] Y. Linde, A. Buzo, R. M. Gray, "An Algorithm for Vector Quantizer Design", IEEE Trans. Comm., Vol. COM-28, pp. 84-95, Jan. 1980.

[9] N. S. Jayant and V. Ramamoorthy, "Adaptive Postfiltering of ADPCM Speech," Proc. ICASSP, pp. 16.4.1-16.4.4, Tokyo, April 1986.

[10] J.-H. Chen, A. Gersho, "Gain-Adaptive Vector Quantization with Application to Speech Coding", IEEE Trans. Comm., Vol. COM-35, No. 9, pp. 918-930, September, 1987.

[11] N. S. Jayant, P. Noll, *Digital Coding of Waveforms*, Prentice Hall, 1984.

[12] G. E. Box, G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, San Francisco, California, Holden-Day, 1970, p. 114.

[13] R. M. Gray, "Vector Quantization," ASSP Mag., Vol. 1, Apr. 1984.