

Optimization of Window and LSF Interpolation Factor for the ITU-T G.729 Speech Coding Standard

Wai C. Chu and Toshio Miki

Mobile Media Laboratory
DoCoMo USA Labs
181 Metro Drive, Suite 300, San Jose, CA 95110, U.S.A.
wai,miki@docomolabs-usa.com

Abstract

A gradient-descent based optimization procedure is applied to the window sequence used for linear prediction (LP) analysis of the ITU-T G.729 CS-ACELP coder. By replacing the original window of the standard by the optimized versions, similar subjective quality is obtainable at reduced computational cost and / or lowered coding delay. In addition, an optimization strategy is described to find the line spectral frequency (LSF) interpolation factor.

1. Introduction

Recent studies have shown that many popular windows incorporated in the linear prediction (LP) analysis procedure of speech coders are suboptimal [1] [2]. In this paper, the optimization results as applied to the ITU-T G.729 [3] coder are reported. This coder employs a 240-sample window consisting of two parts: the first part is half a Hamming window and the second part is a quarter of a cosine function cycle. The LP coefficients (LPC) are extracted at every 80-sample frame, subdivided into two 40-sample subframes. The steps leading to two sets of coefficients (one per subframe) are:

- Highpass filter input speech.
- Data windowing.
- Autocorrelation computation and modification.
- Levinson-Durbin algorithm.
- LPC to LSF conversion.
- LSF quantization and interpolation.
- LSF to LPC conversion.

Note that quantization is not performed if the LPCs are used for perceptual weighting. Assuming that the current frame is located at $n \in [0, 79]$, the corresponding windowing interval is $[-120, 119]$, which requires the buffering of 40 samples from the future frame before starting the LP analysis procedure.

In this work we adapt the optimization technique developed in [2] to work with the G.729 coder, which has shown to be successful in elevating the subjective quality of the G.723.1 coder with no increase in computational cost. The rest of the paper is organized as follows: the window optimization procedure is described in Section 2; Section 3 contains results related to windows of various lengths and positions; Section 4 describes the optimization procedure for the LSF interpolation factor; some conclusion is included in Section 5.

2. Window Optimization Procedure

The technique used in the present work follows the same gradient-descent approach described in [2], with modifications reflecting the underlying structure of the G.729 coder. An outline is given as follows: for a 240-sample window sequence $w[k]$, $k = 0$ to 239, two LP analysis computations are performed, one for the past frame (located at $n \in [-80, -1]$ with past window at $[-200, 39]$) and another for the current frame (located at $n \in [0, 79]$ with current window located at $[-120, 119]$); the two sets of LPCs are interpolated and applied to the corresponding subframes (synthesis intervals) to obtain two values of prediction-error energy, denoted by J_0 and J_1 ; the window is then perturbed with

$$w_o[n, n_o] = \begin{cases} w[n] + \Delta w; & \text{if } n = n_o \\ w[n]; & \text{otherwise} \end{cases}$$

for n and n_o ranging from 0 to 239. The constant Δw is a small positive quantity, and a value of 10^{-4} is used in the present study. For each n_o , new values of prediction-error energy are calculated and are denoted by $J_0'[n_o]$ and $J_1'[n_o]$. Based on these values of prediction-error energy, an error gradient is calculated with

$$\frac{\partial J}{\partial w[n_o]} = \frac{J_0'[n_o] - J_0}{\Delta w} + \frac{J_1'[n_o] - J_1}{\Delta w}.$$

Once the gradient is obtained, the window can be tuned in the direction negative to the gradient so as to reduce the prediction-error energy, this is performed with

$$w_{new}[n] = w[n] - \mu \frac{\partial J}{\partial w[n]}$$

where μ is a positive constant called the step size parameter, a value of 10^{-10} is used throughout the experiment.

For optimization, a training data set is created from the TIMIT database (downsampled to 8kHz) using 54 files, total duration is approximately three minutes. During training, the window is updated after each presentation of the frame. A complete pass through the whole training data set is called an epoch. The training is carried out until the window converges and no additional gain is observed. To evaluate the generalization capability of the optimized windows for signals outside the training data set, a testing data set is formed using

6 files not included in the training data set; total duration is 18 seconds.

Result of optimization is shown in Fig. 1, which is obtained after roughly 1000 epochs of training. Note that the shape of the optimized window (denoted by w_1) is substantially different from the original. Justification of this approach for optimization is that by lowering the average prediction-error energy, the synthesis error is expected to drop, since most speech coders rely on some form of quantization for the excitation signal.

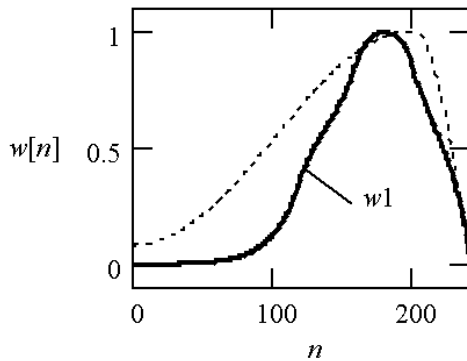


Figure 1. The 240-sample original window (dot trace) and optimized window (solid trace).

With respect to the original window, an increase of 1.7% in segmental prediction gain (SPG [1] [2]) is obtained for the training data set. In terms of subjective quality, we used the ITU-T P.862 perceptual evaluation of speech quality (PESQ) standard [4]. It is observed that the optimized window outperforms the original one by 0.51% and 1.29% in PESQ score for the training data set and testing data set, respectively.

3. Windows of Alternative Lengths and Positions

From Fig. 1 it is tempting to jump to the conclusion that the first one third of the window (80 samples) does not contribute at all, since that portion of the optimized window has very low amplitude samples. In fact, increase in SPG and PESQ score for the optimized window might be a direct consequence of lowering the amplitude of the window in the mentioned region.

A shorter window is beneficial since computation can be reduced. On the other hand it is unclear whether the original positioning of the window with respect to the frame is necessary, since a coding delay equivalent to 40 samples (5 ms) is added; by re-positioning the window it might be possible to reduce the buffering requirement, leading to a more desirable coder. Thus, we apply the same optimization strategy to windows having different lengths and positions, with the positions reflected by the amount of samples buffered from the future frame. Table 1 contains a summary of the cases considered; note that w_1 is the optimized window in Fig. 1.

Fig. 2 contains plots of the optimized windows, where the relative positions with respect to the original window are shown. Performance comparison is given in Table 2. All optimized windows have higher SPG with respect to the original. Not quite for the PESQ scores, where only w_1 and w_2 have significant improvements; w_3 and w_4 have deteriorated slightly, and change for w_5 is slim.

Even though the changes in subjective quality seem to be minor, it is important to point out that they are achieved at same or lower computational cost and coding delay; this is because a shorter window requires less computation for the autocorrelation values, and re-positioning the window toward the left reduces the future buffering requirements. Moreover, modification to the encoder requires minimum effort: a window exchange will do. For the case of w_4 , for instance, there is no need to buffer future samples and the window is half as long as the original; deterioration in PESQ score is only 0.51%.

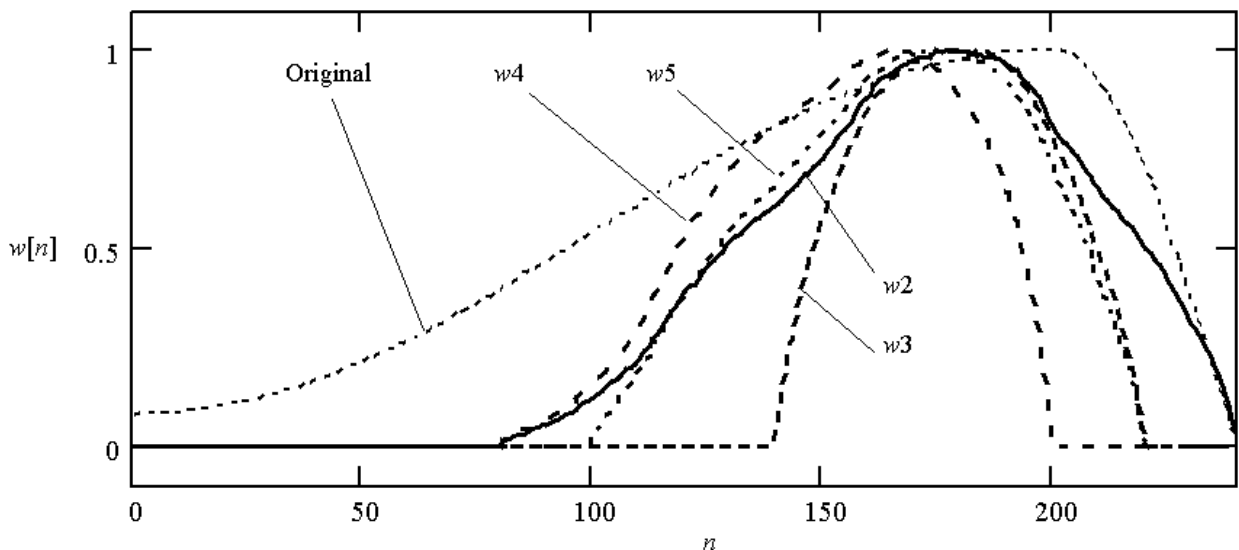


Figure 2. Original window and four optimized windows.

Table 1. Features of various windows subject to optimization.

Window	Length h	Future buffering requirement (samples)
w1	240	40
w2	160	40
w3	80	20
w4	120	0
w5	120	20

Table 2. Performance comparison between the original window and various optimized windows.

Window	SPG <dB>	PESQ (Training)	PESQ (Testing)
Original	9.42	3.92	3.87
w1	9.58(+1.70%)	3.94(+0.51%)	3.92(+1.29%)
w2	9.58(+1.70%)	3.95(+0.77%)	3.89(+0.52%)
w3	9.50(+0.85%)	3.90(-0.51%)	3.85(-0.52%)
w4	9.47(+0.53%)	3.90(-0.51%)	3.85(-0.52%)
w5	9.58(+1.70%)	3.92	3.90(+0.78%)

4. Optimization of LSF Interpolation Factor

Another heuristic aspect of many LP-based coders is the rule for LSF interpolation. For the G.729 coder, with \mathbf{u}_0 and \mathbf{u}_1 denoting the LSF vectors of the first and second subframe, respectively, we have

$$\mathbf{u}_0 = 0.5\mathbf{u}_{\text{past}} + 0.5\mathbf{u}, \mathbf{u}_1 = \mathbf{u}$$

where \mathbf{u} is the LSF vector of the current frame, and \mathbf{u}_{past} is the LSF vector of the past frame. Consider the general interpolation rule:

$$\mathbf{u}_0 = (1-\alpha)\mathbf{u}_{\text{past}} + \alpha\mathbf{u}$$

with $0 < \alpha < 1$. We propose a procedure to jointly optimize the window as well as the interpolation factor α . Flow chart is shown in Fig. 3. Window optimization is based on the gradient-descent technique described before. Flow chart of the interpolation factor adjustment procedure is shown in Fig. 4. Assume that the variable $sign$ is initialized with 1, the procedure tests the performance of the new interpolation factor $\alpha+0.01sign$, if the factor increases the performance (in terms of an elevated SPG), then it exits with the new interpolation factor; otherwise $sign$ is reversed and α returned to its original value. Thus, the next time that the procedure is entered, the opposite direction will be evaluated. The technique is a fixed step-size search approach [5] where the interpolation factor is changed by a maximum of 0.01 at each pass.

Three different positions are selected for joint optimization and are summarized in Table 3. The length is equal to 120. Fig. 5 shows some experimental outcomes for w_6 , where SPG and α are plotted as a function of the training epoch. The initial window is rectangular and the initial LSF interpolation factor α is equal to 0.5. In the present implementation, one adjustment to the interpolation factor is performed every ten training epochs for window optimization.

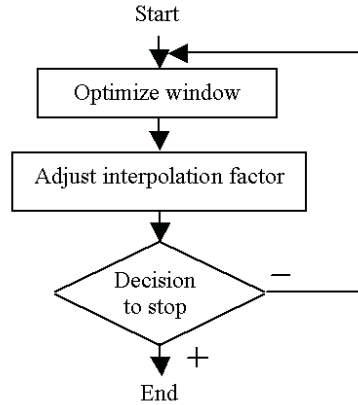


Figure 3. Flow chart of the joint optimization procedure for window and interpolation factor.

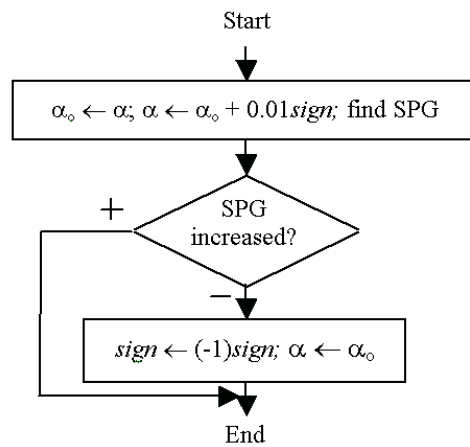


Figure 4. Flow chart of the interpolation factor adjustment procedure.

The optimized windows are plotted in Fig. 6, note how their shapes are adapted according to their positions. Also note that the interpolation factor upon convergence (Table 3) reflects the position of the window; w_6 , for instance, has a lower interpolation factor than w_7 , since the position associated with w_7 covers more the interval where the frame is located. Performance comparison appears in Table 4. As we can see, even though increase in SPG is evident, there is no clear advantage in terms of PESQ score; in fact there is a slight decrease in almost all cases.

Table 3. Features of three windows selected for joint optimization, the length is always 120. The interpolation factors are found through optimization.

Window	Future buffering requirement (samples)	Interpolation factor
w_6	20	0.88
w_7	10	0.96
w_8	0	1.03

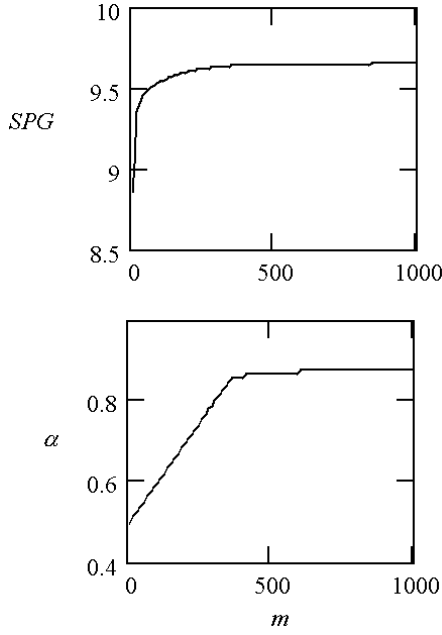


Figure 5. SPG (top) and LSF interpolation factor (bottom) as a function of the training epoch (m) for w_6 .

Table 4. Performance comparison between the original window and three optimized windows.

Window	SPG <dB>	PESQ (Training)	PESQ (Testing)
Original	9.42	3.92	3.87
w_6	9.65(+2.44%)	3.90(-0.51%)	3.92(+1.29%)
w_7	9.65(+2.44%)	3.90(-0.51%)	3.84(-0.78%)
w_8	9.61(+2.02%)	3.88(-1.02%)	3.80(-1.81%)

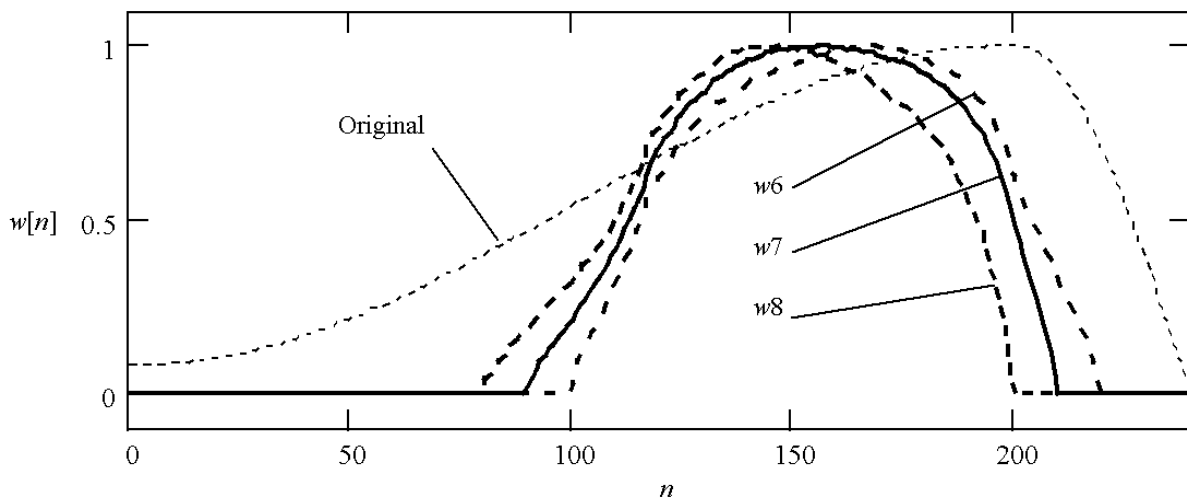


Figure 6. Original window and three optimized windows, obtained by jointly optimizing the LSF interpolation factor.

5. Conclusions

Window optimization is applied to the G.729 coder, where it is shown that increase in SPG is possible, which might not be linked to an elevation in PESQ score. For the case of w_1 having the same length and position as the original window, increase in PESQ is 0.51% and 1.29% for the training and testing data set, respectively. The shorter window w_2 with only 160 samples is slightly superior in performance when compared to w_1 , and is deployable at reduced computational cost. Another interesting outcome is w_5 , which has half the length of the original and requires only 20 samples of future buffering with slightly superior subjective quality; this is a case where similar quality is achievable at reduced computation and lowered coding delay. Joint optimization of LSF interpolation factor is effective in elevating the SPG; however, as is demonstrated before, an increase in SPG might not necessarily imply an improvement in PESQ.

Regardless of whether the PESQ score has improved or not, we can see that the average difference is less than 1% in most cases. Thus, we conclude that the G.729 coder can be deployed at reduced computational cost and lowered coding delay by utilizing alternative windows, which can be found through the training techniques described in the present work.

6. References

- [1] W. C. Chu, "Gradient-Descent Based Window Optimization for Linear Prediction Analysis", *IEEE ICASSP*, Hong Kong, April 2003.
- [2] W. C. Chu, "Window Optimization for the ITU-T G.723.1 Speech Coding Standard", proceeding of the *GTC ISPC*, Dallas, April 2003.
- [3] ITU-T, *CS-ACELP – Recommendation G.729*, 1996.
- [4] ITU-T, *PESQ – Recommendation P.862*, 2001.
- [5] S. S. Rao, *Engineering Optimization – Theory and Practice*, 3rd edition, John Wiley & Sons, 1996.