A SIMPLE LPC AND RELP VOCODERS SIMULATION PROGRAM DEVELOPED ON THE MATLAB® PLATFORM

Fábio A. R. Nascimento[1], Alan M. Marotta[2] and Francisco J. Fraga[3]

Abstract *— The widespread use of mobile telephones has reinforced the importance of coding and transmitting speech at low bit rates. As a natural consequence, the interest of researchers and postgraduate students in the speech coding area has increased significantly over the last years. A special attention has been focused on the topic of parametric LPC-based (LPC: Linear Predictive Coding) Voice Encoders (VOCODERS). This class of Vocoders offers the most promising bandwidth-quality trade-off. This paper presents a simulation program wrote on the Matlab® platform which allows postgraduate students understanding the basic principles of the LPC and RELP (Residual Excited Linear Prediction) Vocoders. The resulting speech signal can be heard and also visually compared with the original signal. By means of this tool, the student can also compare the trade-off between the quality of the synthesized speech and the bit-rate reduction, for both the LPC and RELP Vocoders.*

Index terms — LPC Vocoder, RELP Vocoder, Speaking Machine, Vocoders simulation.

INTRODUCTION

There are several *LPC-based* speech coding techniques with low bit rate already standardized and commercially implemented. An example is the ADPCM system that requires one half of the PCM bit rate, with the same performance. In the case of speech (or voice) transmission, there are even bolder techniques, reducing up to 8 times the PCM rate, paying the price of a slight loss of quality. To do so, a speech production model becomes necessary, where some characteristic properties of this kind of signals are applied. The systems using this approach are called Vocoders [1], a term derived from *voice encoders*.

The strategy used for the construction of this kind of circuits is based on the transmission of the voice parameters, instead of transmitting the voice signal itself, as following described. Studies on the modeling of the human speech production system are lead, officially, since 1780, with the experiments of Von Kempelem and its speaking machine [1]. Nowadays this study has already gone reasonably deep and it is possible to get a very good model of the human speech production system by a discrete-time system. Thus, it is possible to implement at the transmitter a circuit capable to analyze the input speech signal and to provide at the output the codified parameters of this signal. If we implement a speech synthesis sub-system at the receiver with the received parameters, then we have got a system that is able to transmit only the parameters of the original speech signal, but with good intelligibility at the receiver output. This leads to an enormous decrease in the transmission bit rate.

Nevertheless, in order to obtain better speech quality, we have to look for more elaborated Vocoders, where more advanced techniques are used, such as CELP (Code Excited Linear Prediction) and VSELP (Vector Sum Excited Linear Prediction) [2]. As an example, a VSELP Vocoder, which is the TIA (Telecommunications Industry Association) coding scheme for cellular telephony in the United States, is able to codify speech at 8 Kbps with an acceptable level of quality. Such low bit rate signifies almost a 10:1 bandwidth reduction factor, if compared to standard PCM (64 Kbps).

Although commercial Vocoders like VSELP present a high level of complexity, all of them are based on LPC techniques, which have not so complex theoretical principles. With aim of teaching the LPC techniques in an efficient way, it is important to develop some computer-based educational methods to make easier the comprehension of their theoretical concepts. This is the goal of the program we have developed and described here.

THE LPC VOCODER

The LPC VOCODER is a discrete-time speech production system, as illustrated in figures 1 and 2.

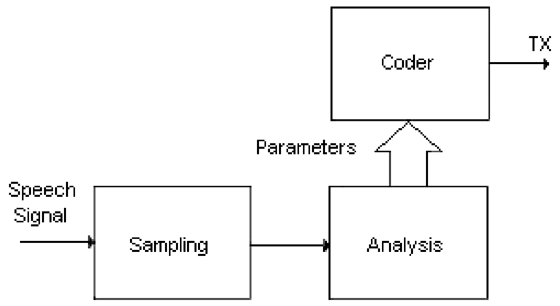

FIGURE. 1
LPC Vocoder Transmitter

[1] Fábio A. R. Nascimento, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí, MG, Brazil, fabio.nascimento@inatel.br
[2] Allan M. Marotta, Instituto Nacional de Telecomunicações, alanm@inatel.br
[3] Francisco J. Fraga, Instituto Nacional de Telecomunicações, fraga@inatel.br

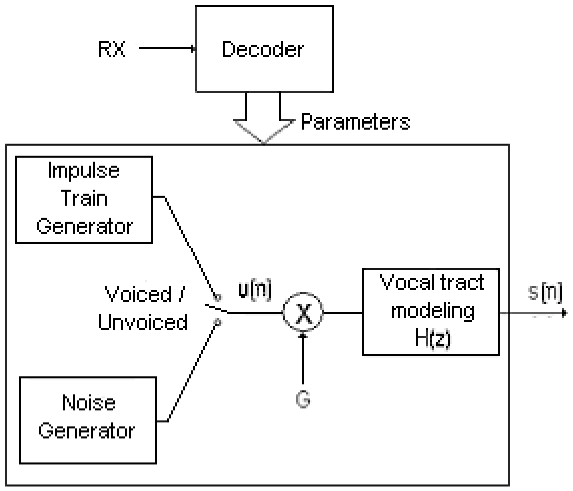

FIGURE. 2
LPC Vocoder Receiver

One can find strange the use of an impulse train generator and a noise generator at the receiver in order to reproduce the speech. But an analysis of the speech signal leads us to agree with the idea that the voiced sounds have a fundamental frequency, known as pitch frequency, and the unvoiced sounds are very similar to a noisy sound. Thus, we can admit that the excitation signal can be reasonably substituted by an impulse train for voiced sounds and by a random noise signal for unvoiced sounds [3]. In figure 3 we can see an illustration containing the speech signal generated from the pronunciation of the word “vocoder”.

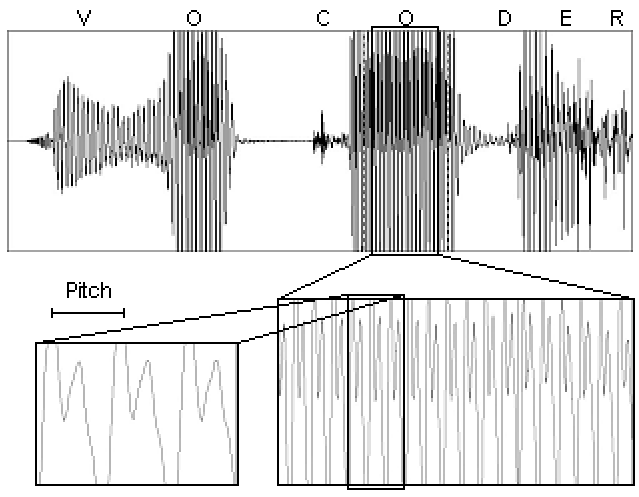

FIGURE. 3
SPEECH SIGNAL IN THE TIME DOMAIN

Another relevant point is the implementation of a vocal tract modeling filter, with transfer function H(z). Using linear prediction concepts applied to the speech signal, we are able to determine the parameters that describe such a filter. The parameters that are essential to an intelligible speech reproduction at the receiver are, as we can foresee by analyzing the figures 1, 2 and 3, the impulse train frequency, which is determined from the *pitch*; the vocal tract *modeling filter coefficients* and the *gain* of the excitation signal.

The mathematical modeling representing the output speech signal of the decoder circuit is shown below [4]:

$$s(n)=\sum_{k=1}^{p}a_k s(n-k)+Gu(n) \tag{1}$$

The expression that represents the speech modeling filter transfer function is [4]

$$H(z)=\frac{G}{1-\sum_{k=1}^{p}a_k z^{-k}}, \tag{2}$$

where a_k are filter coefficients corresponding to the transfer function poles. As we notice, this function just contains poles; although it is able to perform a good approximation of the vocal tract, with small damage to the nasal and fricative sounds, which would require a function containing also zeros, if we desire a more accurate speech production model.

CALCULATING THE FILTER COEFFICIENTS

A proper approach to the calculus of the *filter coefficients* is related to minimum mean-square error. Following this idea, the expression of the prediction mean-square error is

$$J=E\left[e^2(n)\right]$$

$$J=E\left[\left(s(n)-\sum_{i=1}^{p}a_i s(n-i)\right)^2\right], \tag{3}$$

where $s(n)$ is the speech signal (windowed in frames with a typical duration of 20 to 30 ms) and a_i are *filter coefficients*.

Expanding the expression above, we have

$$J=E\left[s^2(n)\right]-2\sum_{i=1}^{p}a_i E\left[s(n)s(n-i)\right]+ \sum_{k=1}^{p}\sum_{i=1}^{p}a_k a_i E\left[s(n-k)s(n-i)\right] \tag{4}$$

Assuming that $s(n)$ is a sample function of a stationary process, we have

$$J = R(0) - 2\sum_{i=1}^{p} a_i R(i) + \sum_{k=1}^{p}\sum_{i=1}^{p} a_k a_i R(i-k), \qquad (5)$$

where $R(i)$ are samples of the auto-correlation function of the speech signal, which can be obtained by the auto-correlation of the speech signal in each frame [4]:

$$R(k) = \sum_{n=1}^{N} s(n)s(n+k) \qquad 1 \le k \le p, \qquad (6)$$

where p is the prediction filter order, normally chosen between 8 and 12. The idea is to minimize the mean-square error, and this approach will lead us to obtain the proper coefficients. To do so, we derive J in relation to the coefficient of order i and make the resulting expression equal to zero. Thus, we obtain the following expression [5]:

$$\sum_{k=1}^{p} a_k R(i-k) = R(i) \qquad i = 1,2,...,p \qquad (7)$$

In the matrix form we have

$$\mathbf{a} \cdot \mathbf{R} = \mathbf{r} \qquad (8)$$

where

$$\mathbf{a} = [a_1, a_2, ..., a_p]^T$$

$$\mathbf{R} = \left[\begin{array}{c:c:c:c:c} R[0] & R[1] & ... & ... & R[p-1] \\ R[1] & R[0] & ... & ... & R[p-2] \\ : & : & ... & ... & : \\ : & : & ... & ... & : \\ R[p-1] & R[p-2] & ... & ... & R[0] \end{array}\right]$$

and

$$\mathbf{r} = \left[R[1], R[2], ..., R[p]\right]^T$$

The set of equations (7) or (8) are called the Wiener-Hopf equations for linear prediction and their solution lead us to

$$\mathbf{a} = \mathbf{R}^{-1} \cdot \mathbf{r} \qquad (9)$$

$$J_{\min} = R(0) - \mathbf{r}^{\mathbf{T}} \cdot \mathbf{R}^{-1} \cdot \mathbf{r} \qquad (10)$$

CALCULATING THE PITCH

The algorithm used to detect the pitch requires the definition of new parameters: the auto-correlation of **a** (*filter coefficients auto-correlation function*):

$$R_a(i) = \sum_{k=1}^{p} a_k a_{k+i} \qquad 1 \le i \le p \qquad (11)$$

where a_{k+i} is zero if $k + i > p$

The convolution of $R_a(i)$ with $R(l)$:

$$R_e(l) = \sum_{i=1}^{p} R_a(i) R(l-i) \qquad (12)$$

where $R(l)$ can be calculated by equation (6) with $0 \le l \le N-1$. Normalizing $R_e(l)$, we have

$$R_{eN}(l) = R_e(l) / R_e(0) \qquad (13)$$

If the maximum value of $R_{eN}(l)$ is equal or superior to a threshold γ (the most used value for γ = 0.25), then we may consider we have a voiced frame. On the contrary, if $R_{eN}(l)$ is inferior to γ, then we may consider we have an unvoiced frame and, consequently, there is no pitch. If the frame is considered as voiced, the value of the pitch period is set by the position (sample index) where the maximum value of $R_{eN}(l)$ occurs.

CALCULATING THE GAIN

The gain of the signal is easily deducted from the following concept:

$$Gu(n) = e(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k). \qquad (14)$$

THE RELP VOCODER

Unlike the LPC Vocoder, the RELP (Residual Excited Linear Prediction) Vocoder [3], instead of considering the excitation signal at the decoder as an impulse train or a random noise, it codifies and transmits the prediction error (or *residual*) signal *e(n)*, which is the proper excitation signal. This procedure requires a higher bit rate, but, on the other hand, it does a better job by capturing true characteristics of the original signal through residual signal. It is obvious that it does not make any sense to transmit a signal of such complexity like the error signal as it is. So, we use the artifice of *decimating* it (after low pass filtering) by some decimate factor of at least four (reducing four times the number of samples), before transmitting it. This requires that the decoder perform the inverse process, over-sampling the error signal received, in order to recover the original amount of samples. Then, the decimated samples at the encoder are interpreted as zero at the decoder. Even with this loss introduced for the decimate process, the subjective result is considered much better than the one obtained with the LPC Vocoder.

Thus, the parameters to be transmitted are the *filter coefficients* and the *decimated error signal* or *decimated prediction residue.*

MATLAB IMPLEMENTATION

Bearing in mind that one of our goals is to evaluate the Vocoders transmission rate, comparing it to other systems, a good approach is to use the PCM standard system as our reference. So we may consider speech signals sampled with the same sampling rate and the same number of quantization levels that those in the PCM Standard system, where the sampling frequency f_s = 8 kHz and the quantization levels are 256 (8 bits), leading us to a 64 Kbps bit rate.

In order to derive the filter coefficients we assumed that the speech signal $s(n)$ was a stationary signal. This is not true, due to modifications of the vocal tract characteristics along the time, which allow us to pronounce different phonemes. Fortunately, these modifications have a small variation rate, permitting us to consider that the voice signal is stationary in a short time interval T_q (20 to 30 ms) [1]. Considering the above concept, we notice that the speech signal must be recorded and sub-divided in N_q frames, where N_q is equal to the duration time T_s of the complete speech signal divided by the frame duration time T_q

$$N_q = \frac{T_s}{T_q}\,. \tag{15}$$

The number of samples for each frame is equal to its duration time T_q multiplied by the sample frequency f_s

$$N = T_q f_s\ . \tag{16}$$

The parameters are extracted and transmitted for each frame. The *filter coefficients* for each frame are derived from equation (9), after calculating the auto-correlation of the speech signal in each frame.

It is important to notice that the higher is the prediction filter order, the more precise is the prediction signal and, consequently, smaller the error. However, the error does not decrease significantly for a filter order above 12. Values higher than this order set more parameters to be transmitted, but with almost any improvement of the subjective performance. Therefore, the developer must find the better relation between subjective quality and the required transmission rate, when setting the filter order. The standardized LPC-10 Vocoder, as its name suggests, uses p = 10 prediction coefficients.

Then, let us consider p coefficients for each frame, being each of them satisfactorily encoded with 3 decimal points of precision, what sets 10 bits for encoding and transmitting each coefficient. The necessary transmission rate for the parameter *filter coefficients* is calculated by multiplying the amount of required bits for each coefficient by the number of coefficients in each frame and by the number of frames contained in one second:

$$R_a = 10\frac{p}{T_q} \tag{17}$$

The parameter *filter coefficients* is necessary for both LPC and RELP Vocoders.

We will deal now with implementation issues related with the *pitch* and *gain* parameters, which are necessary only for the LPC Vocoder. Following, we will consider the *prediction error* parameter, which is required only for the RELP Vocoder.

It is possible to prove [4] that we can directly obtain the parameter *gain* using the equation

$$G = \sqrt{R(0) + \sum_{k=1}^{p} a_k R(k)}\ . \tag{18}$$

Typically 5 bits are needed for binary representation of the parameter *gain* for one frame [4]. Its transmission rate is calculated by multiplying the necessary amount of bits for representing the parameter for one frame by the number of frames in one second:

$$R_g = \frac{5}{T_q}\,. \tag{19}$$

The extraction of the parameter *pitch* is done directly through the equations (11), (12) and (13), associated with a routine that looks for the sample index of the peak of the normalized cross-correlation signal:

$$T_o = \max_l \left[R_{eN}(l)\right]. \tag{20}$$

Usually we consider that 6 bits are enough for the binary representation of the parameter *pitch* (T_0) for each frame [4]. Its transmission rate is calculated by multiplying the required amount of bits to represent it for one frame by the number of frames in one second:

$$R_p = \frac{6}{T_q}. \qquad (21)$$

In case of Vocoder RELP, as mentioned above, the *filter coefficients* are obtained exactly the same way as in the LPC case. The extraction of the parameter *decimated prediction error* is made by applying the definition

$$e(n) = s(n) - \hat{s}(n), \qquad (22)$$

where $s(n)$ is the original speech signal and $\hat{s}(n)$ is the predicted signal, obtained from a digital filter with unitary gain and denominator coefficients a_k.

The decimation is done by low pass filtering the error signal and by considering one sample among each group of *d* samples, setting as zero the other ones, where *d* is the decimation factor. Let us represent the decimation function and its decimation factor *d* by Dec_d. So, the *decimated prediction error* parameter is

$$e_d(n) = Dec_d\left[e(n)\right]. \qquad (23)$$

The parameter *decimated prediction residue* requires a lot of bits, because every d^{th} sample of the prediction error signal $e(n)$ must be encoded and transmitted. A similar coding concept occurs in the case of the DPCM system. It is known that the DPCM system represents a reduction of two bits per sample in relation to the PCM system [5]. Thus, 6 bits are enough for coding this parameter properly.

The bit rate due to this parameter is evaluated by multiplying the number of bits needed for representing one sample by the number of samples in one second, divided by the decimation factor

$$R_e = \frac{6 f_s}{d}. \qquad (24)$$

The LPC synthesis is obtained through the implementation of a digital filter, obtained with the received and decoded a_k, G and T_0 parameters. The input signal of the filter defined right above may be of two types: an impulse train, with period defined by the *pitch* parameter T_0, for voiced sounds; or by a random noise signal for unvoiced sounds. The total bit rate for the LPC-10 vocoder (f_s = 8 KHz, p =10 and T_q = 25 ms) will be then equal to

$$R_{LPC} = R_a + R_g + R_p$$
$$R_{LPC} = 4{,}44 \text{ Kbps}$$

The RELP synthesis is obtained through the implementation of a digital filter, with the received and decoded a_k parameters and unity gain. The input signal for the filter mentioned right above is the received, decoded and over-sampled parameter $e_d(n)$. The total bit rate for the RELP vocoder (f_s = 8 kHz, p =10, d = 4, T_q = 25 ms) is

$$R_{RELP} = R_a + R_e$$
$$R_{RELP} = 16 \text{ Kbps}$$

Conclusions

The Matlab implementation of these two Vocoders is relatively easy due to its facility of implementing routines that work with matrices and vectors, beside its variety of built-in functions that can be used. The bit rate tests and also the final subjective results show to anyone who uses the software that there is a quality enhance of the decoded signal obtained by the RELP Vocoder, but with the trade-off of a significant increase of the bit rate if compared with the LPC Vocoder.

Finally, we want to remark that our experience with the master students that helped in the development of this software and with the others that used it for learning the LPC coding techniques demonstrated to be very profitable for our educational purposes.

Acknowledgment

We sincerely thank for the help of Prof. Carlos A. Ynoguti and for the funding of FINATEL and ERICSSON .

References

[1] Rabiner L. R., Schafer R.W. *Digital Processing of Speech Signals*, 1st edition, New Jersey: Prentice-Hall, 1978. 512 p.

[2] Gold B., Morgan N. *Speech and Audio Signal Processing*, New York: John Wiley & Sons, 2000. 537 p.

[3] Deller J. R., Hansen J. H. L., Brakis J. G. *Discrete-Time Processing of Speech Signals*, New York: Institute of Electrical and Electronics Engineers, 2000. 908 p.

[4] Ingle V. K., Proakis J.G. *Digital Signal Processing Using Matlab® V.4*, Boston: PWS Publishing Company, 1997. 421 p.

[5] Haykin S. *Communication Systems*, 4th edition, New York: John Wiley & Sons, 2001. 816 p.