Using a fast RLS adaptive algorithm for efficient speech processing

C. Papaodysseus*, G. Roussopoulos, A. Panagopoulos

Department of Electrical and Computer Engineering, National Technical University of Athens, 9, Irion Politechniou Str., Athens GR 15773, Greece

Received 23 April 2003; received in revised form 23 September 2004; accepted 4 October 2004
Available online 25 December 2004

Abstract

In this paper, a new method is presented that offers efficient computation of Linear Prediction Coefficients (LPC) via a new Recursive Least Squares (RLS) adaptive filtering algorithm. This method can be successfully used in speech coding and processing. The introduced algorithm is numerically robust, fast, parallelizable and has particularly good tracking properties. By means of this scheme, Linear Prediction Coefficients are obtained that offer an improvement in the reconstruction of the speech signal before coding, as compared to the signal obtained by various classical algorithm. An analogous improvement is observed in speech coding experiments too, while a subjective test confirms the improvement of the quality of synthesized speech. The overall processing time of the proposed method of speech coding is a bit greater, but comparable to the time the classical methods need.

© 2004 IMACS. Published by Elsevier B.V. All rights reserved.

Keywords: Speech coding; Speech processing; Adaptive RLS filtering; Forward linear prediction

1. Introduction

As is well known, Linear Prediction Analysis for speech coding is quite powerful and useful. According to this, speech can be modeled as the output of a linear, time-varying system, excited by either a properly chosen train of pulses (for voiced speech) or a white noise (for unvoiced speech). The subject has been
extensively treated in the literature, since the early seventies (e.g. see [3,4]). The Linear Prediction Coefficients (LPC) have been so far calculated, with various methods, which can be classified into three major categories, namely: (1) the covariance method, (2) the autocorrelation one and (3) the lattice method. All these methods solve the Yule–Walker equations by taking advantage of the special structure of the system matrix.

In this paper, the ideas of adaptive filtering are used for obtaining LPC that offer a quality of approximation of the original speakers speech signals that is better than the one offered by the aforementioned classical methods. The LPC are computed by means of a new Kalman-type Recursive Least Squares (RLS) adaptive algorithm that has a number of interesting properties: (a) it is stabilizable, (b) has excellent tracking properties, (c) is fast, and if necessary, can be parallelized. This computational scheme is presented in the Appendix A. Notice that the overall processing time required by the proposed algorithm is a bit greater than the one required by the Levinson–Durbin scheme, but it is essentially comparable to it.

In Section 3 of the present paper, the quality of approximation of English speakers voice signals by the proposed adaptive RLS algorithm is presented, in comparison with the approximation offered by the classical computational methods. In Section 4.1, it is presented a comparison of the quality of synthesis of the English speakers voice signals after coding, by employing the proposed adaptive algorithm on one hand and the classical algorithms on the other. In Section 4.2, a subjective test is reported, which confirms that the quality of synthesis, obtained by means of the introduced methodology, is slightly better than the one obtained by classical methods.

2. The applied method for speech signal processing

Let \( x(n), n = 1, 2, \ldots \), be the obtained samples of a speech signal. It is a standard practice to apply the Forward Linear Prediction technique to this kind of signal as this is described, for example in [1,2]. In this paper, we have adopted the following method, which is a rather straightforward extension of the classical one.

We have divided the speech signal into consecutive frames (windows) of length \( L \) that in contrast with the classical method are not necessarily overlapping. In each such window, we look for \( m \) LPC \( a_i, i = 1, \ldots, m \), such that if \( \hat{x}(n) = \sum_{i=1}^{m} a_i x(n-i) \), then the quantity \( \hat{x}(n) \) is close to the actual sampled signal \( x(n) \) in the classical Least Squares sense. Following the standard method (see [1–3]) the analysis leads to the solution of the set of linear equations \( R_m(n) a_m(n) = -r_m(n) \), which is usually called the Yule–Walker equation. The recursive solution of this set of equations has been the object of extensive study. The first RLS algorithm of \( O(m^2) \) computational complexity that solves this set of equations recursively, has been developed by Kalman. Subsequently, a number of well known fast (i.e. of \( O(m) \) computational complexity) RLS algorithms have been developed of either a growing window type [1,2] or a sliding window one [8].

In this paper, for the solution of the system of linear equations we have chosen the computational scheme presented in the Appendix A, since (a) it is fast; (b) it is stabilizable by means of the methodology introduced in [5,7,9]; (c) it has excellent tracking properties and (d) is parallelizable. In particular, this algorithm can treat speech signals robustly as well as various non-stationary ones, while most other RLS schemes, when applied to such type of signals, usually demonstrate serious numerical problems due to the finite precision—quantization error. In fact, the proposed algorithm uses a novel method for calculating
the exact number of erroneous digits with which all critical quantities are computed at each algorithm step. Using this information and a proper feedback scheme the correct value of the critical quantities is restored whenever necessary [6,9]. Notice that for all classical adaptive RLS algorithms there is a trade-off between the algorithm tracking capabilities and its robustness. The stabilization method used in the proposed algorithm allows for very good tracking properties while substantial robustness is ensured.

The basic structure of this algorithm is that of a finite window one, i.e. two pairs of Kalman-type gains \((u^*_m(n), u_m(n))\) and \((v^*_m(n), v_m(n))\) of vectors are used as intermediate mathematical quantities of the algorithm, with the purpose of drastically reducing the overall computational complexity (see Appendix A). The updating of the various quantities is made by means of formulae (A.10), (A.11), (A.14), (A.15) and (A.17), which update the system matrix at time instant \(n+1\) by means of the system matrix at time \(n\), using the vector \(x\) that includes the newcoming information and the vector \(y\) that includes the dropped information. The updating formula is for the first time stated and used in the RLS filtering [6]. The algorithm starts at the first sample of the first window either by directly solving the Yule–Walker equations or by choosing an initial diagonal matrix with properly large numbers as entries and by letting the system converge. At the subsequent frames beginning, the previously calculated values are used to initialize the system.

The whole development of the algorithm is made in such a way as to guarantee that the quantities that correspond to the incoming information on one hand and the dropped information on the other hand are computed in parallel, so that the resulting adaptive algorithm is immediately parallelizable. It must be pointed out that although the algorithm computes the LP Coefficients \(a_i\), \(i = 1, 2, \ldots\), iteratively at each time instant \(n\), the actually used coefficients for obtaining the approximation \(\hat{x}(n)\) at each one of the frames in which the speech signal is divided are those computed at the last time instant of the window in hand. In this way, we ensure that minimization is achieved for the whole frame of interest.

3. Quality of approximation of speech signals

In all the experiments performed and referred to in this publication we have used data from the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus, or for abbreviation TIMIT database that contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from eight major dialect regions of the United States.

For each phoneme or sequence of phonemes taken from TIMIT, we have applied the methodology described in the previous section, namely:

1. We have divided the speech signal into consecutive non-overlapping frames (windows) of length \(L\). The sampling frequency has always been 16,000 Hz.

2. We used the algorithm referred to in the previous section and analytically presented in the Appendix A, for computing the LP Coefficients in all windows or frames of interest.

We have chosen the window length \(L = 200\) and the system order \(m = 20\), although we should stress that one can obtain very good results with other combinations of \(L\) and \(m\), too. Notice that the smaller the value of the ratio \(L/m\) the better the obtained approximation of the original speech signal. However, at the same time, a small value of this ratio results in degraded coding efficiency. Therefore, there is a trade-off of window length and order in connection to quality of approximation of the original speech signal and coding efficiency. In addition, we must point out that there is no memory built up from block to block.
Improvement of the proposed algorithm tracking performance

<table>
<thead>
<tr>
<th></th>
<th>Over L–D or covariance (dB)</th>
<th>Over burg (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male speakers</td>
<td>0.63</td>
<td>0.51</td>
</tr>
<tr>
<td>Female speakers</td>
<td>0.6</td>
<td>0.49</td>
</tr>
<tr>
<td>Total</td>
<td>0.62</td>
<td>0.5</td>
</tr>
</tbody>
</table>

We have applied the aforementioned method to speech signals corresponding to various phrases in English, articulated by eighteen speakers of both genders and of different accent. In order to compare the before coding performance of the proposed algorithm with the one of the previous computational schemes (e.g. the extensively used Levinson–Durbin (L–D), the Burg one and the LU decomposition covariance method), we have calculated the total residual error $e = \sum_{l=M}^{M+L-1} e_i = \sum_{l=M}^{M+L-1} (x(i) - \hat{x}(i))^2$, in each window of analysis \([M, M+L-1]\), for all compared algorithms. Clearly, the smaller $e$ obtained from one algorithm, the better the performance of this scheme.

Notice that both classical LPC algorithms and adaptive RLS algorithms, like the one introduced in this paper, solve a similar minimization problem that leads to the solution of the linear system of equations and all provide stable LPC filters. But in the case of all autocorrelation algorithms it is absolutely necessary that the whole signal needs to be windowed (e.g. using a Hamming window) in order to avoid prediction of the first signal samples from zero-valued samples. So, in the formula of the error to be minimized $x(i)$ is the original signal multiplied by the Hamming window. On the contrary, in the LPC philosophy introduced here no multiplication by such a window is necessary. This is the first main factor that allows for a better approximation of the original signal via LPC analysis.

But even in the case one decides that use of a window is absolutely necessary, in which case for $\lambda = 1$ one gets identical results, one can still obtain smaller overall residual error, by tuning the value of the forgetting factor $\lambda$ and by making small successive window overlapping. The experiments performed show that, in all cases, the LP Coefficients offered by the introduced RLS adaptive algorithm, give an essentially improved approximation of the actual speech signal, than the one offered by the LPC computed by the L–D algorithm, Burg and covariance algorithm.

4. Quality of synthesis of speech signals

After spotting the LP Coefficients with the RLS adaptive method described in the previous sections, we have applied some existing methods of coding and compared the efficiency of coding
Table 2
Comparison of the quality of approximation realized by the LPC obtained from the proposed algorithm and the L–D one

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of samples</th>
<th>L–D or covariance (dB)</th>
<th>Burg (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affricatives</td>
<td>12</td>
<td>0.53</td>
<td>0.43</td>
</tr>
<tr>
<td>Fricatives</td>
<td>68</td>
<td>0.36</td>
<td>0.3</td>
</tr>
<tr>
<td>Nasals</td>
<td>49</td>
<td>0.79</td>
<td>0.69</td>
</tr>
<tr>
<td>Semivowels–glides</td>
<td>83</td>
<td>0.93</td>
<td>0.78</td>
</tr>
<tr>
<td>Vowels</td>
<td>215</td>
<td>0.72</td>
<td>0.6</td>
</tr>
</tbody>
</table>

and the quality of speech signal reconstruction made by the technique employed here, to the classical ones.

4.1. In connection to the multipulse Linear Prediction Coefficients methods

First, we have applied the very popular multipulse LPC method for coding of the speech signal, employing the coefficients $a_i, i = 1, 2, \ldots$, obtained by our algorithm. In particular, following the procedure described in [10,11], we have computed a set of pulses that are chosen in such a way, so as to minimize the error of the speech signal reconstruction. In order to get good quality of synthetic speech at a satisfactory bit rate, one must choose a proper number of pulses in each window. These pulses are, in fact, calculated recursively, so as to make the difference $\zeta(n) = x(n) - \hat{x}(n)$ minimum in each frame. Further reduction of the bit rate can be achieved with the popular CELP method [11], which is an object of an extensive study and research. In order to test the quality of speech synthesis offered by the RLS method presented here, we have:

1. Applied our methodology in a considerable number of voiced American English phonemes (see Tables) using various numbers of pulses in the excitation.
2. Applied a classical method in the same phonemes, using the L–D algorithm and the Burg algorithm with exactly the same system order and window length, as well as the same with the RLS method number of pulses in the excitation each time. Therefore, we ensured that all tests were made with the same bit rate. In this way, the signal reconstructed by means of the methodology introduced in this paper is better than the one obtained by the classical methods by at least 0.1 dB and by 0.14 dB in the mean case, concerning improvement in the signal-to-noise ratio (SNR). Tables 3 and 4 summarize the results of these experiments.

Notice that the overall processing time required by the proposed algorithm is a bit greater than the one required by the classical schemes, but it is essentially comparable to them. Further research for adaptive
Table 4
Improvement of quality of reconstruction after multipulse coding of speech signals corresponding to most frequent American English phonemes

<table>
<thead>
<tr>
<th>Category</th>
<th>Samples</th>
<th>$\text{dB improvement}$</th>
<th>Over L–D</th>
<th>Over Burg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nasals</td>
<td>54</td>
<td>0.029</td>
<td>0.025</td>
<td></td>
</tr>
<tr>
<td>Semivowels and glides</td>
<td>77</td>
<td>0.183</td>
<td>0.171</td>
<td></td>
</tr>
<tr>
<td>Vowels</td>
<td>216</td>
<td>0.142</td>
<td>0.131</td>
<td></td>
</tr>
</tbody>
</table>

oriented methods of coding, which will fully exploit the drastic improvement of tracking of the speech signal by the introduced algorithm, is now carried on.

4.2. Subjective test

In addition to the above comparisons, we have applied a subjective test as follows. We have chosen five phrases from different speakers each coming from a different area. We divided each phrase in frames and when the frame referred to a voiced phoneme, we have used the two methodologies described in Section 4.1 (the first time the methodology introduced here and subsequently the classical one). When the frame referred to an unvoiced phoneme we used a white noise input or a one-pulse input, and one time the LPC obtained by the L–D algorithm and the other time the LPC obtained by the algorithm introduced in this paper. The result was two signals of synthesized speech for each phrase. Next, we selected 12 persons, members of staff and postgraduate students from the National Technical University of Athens and the Democritus University of Thrace, with very good to excellent knowledge of English language. Two of those persons have English as native language.

Each person listened to the three versions of each phrase, namely the original one, the one reconstructed by our method and the one reconstructed by the classical method and he/she was asked to express his/her opinion about the quality of the synthesized speech. The 12 persons should choose one out of five distinct characterizations concerning the quality of the reproduced phrase with the introduced methodology (named version A) versus the quality of the reproduced phrase with the classical method (named version B). Notice that each listener listened to phrases randomly chosen from classes A or B.

Table 5
Results of experiments described in Section 4.2

<table>
<thead>
<tr>
<th>Characterization of the two versions</th>
<th>Worse</th>
<th>Slightly worse</th>
<th>Equivalent</th>
<th>Slightly better</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase 1</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Phrase 2</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Phrase 3</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Phrase 4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Phrase 5</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>1</td>
<td>24</td>
<td>35</td>
<td>0</td>
</tr>
</tbody>
</table>
without knowing to which class the phrase belonged to. The results of these experiments are summarized in Table 5.

5. Conclusion

This paper concerns an alternative method for efficient LPC computation via a new Recursive Least Squares adaptive filtering algorithm, which is numerically robust, fast, parallelizable and has particularly good tracking properties. This scheme offers a better reconstruction of the speech signal before coding as well as improvement in the reconstruction of voiced phonemes after coding. Therefore, the proposed method can be used as an alternative for the computation of LP Coefficients employed for speech processing, although its computational complexity is a bit greater than the classical methods.

Appendix A. The employed Recursive Least Squares fast algorithm

1. Quantities known at the time instant $n$: $u^*_m(n)$, $v^*_m(n)$, $a_m(n)$, $b_m(n)$, $x_m(n)$, $x_m(M-1)$, $\Delta_1(n)$, $\Delta_2(n)$, $\Delta_f(n)$, $\Delta_{xx}(n)$, $\Delta_{xy}(n)$, $\Delta_{yy}(n)$, $x(M)$, $L = $ window length, $M = n - L + 1 = $ the beginning of the frame.

2. New information: $x(n+1), z(n+1)$

3. Main computational body:

(a) Updating of $u_m(n+1)$, $v_m(n+1)$:

(i) Useful quantities $f_{u_m}^f(n + 1) = x(n + 1) + a_m^T(n)x_m(n)$,

\[ g_{u_m}(n + 1) = x(M) + a_m^T(n)x_m(M - 1); \quad u_{m+1}(n + 1) \]

\[ = \begin{cases} 0 \\ u_m(n) \end{cases} - \frac{1}{\lambda} \begin{cases} 1 \\ a_m(n) \end{cases} \frac{f_{u_m}^f(n + 1)}{\Delta_f(n)} \]  

(A.1)

(ii) Partition of $u_{m+1}(n+1)$:

\[ u_{m+1}(n + 1) = \begin{bmatrix} a_m(n + 1) \\ u_{m+1}(n + 1) \end{bmatrix} \]  

(A.2)

Therefore,

\[ f_{u_m}^b(n + 1) = -\lambda \Delta_f(n) \times u_{m+1}(n + 1) \]  

(A.3)

and eventually

\[ u_m(n + 1) = \bar{u}_m(n + 1) - b_m(n) \times u_{m+1}(n + 1) \]  

(A.4)

(i) Similarly,

\[ v_{m+1}(n + 1) = \begin{cases} 0 \\ v_m(n) \end{cases} - \frac{1}{\lambda} \begin{cases} 1 \\ a_m(n) \end{cases} \frac{g_{v_m}(n + 1)}{\Delta_f(n)} \]  

(A.5)
(ii) Partition of $v_m(n+1)$:

$$v_m(n+1) = \begin{bmatrix} v_m(n+1) \\ v_{m+1}(n+1) \end{bmatrix}$$  \hspace{1cm} (A.6)

Therefore,

$$v_m^{(n+1)} = -\lambda \Delta b_m(n+1)$$ \hspace{1cm} (A.7)

and eventually,

$$v_m(n+1) = \tilde{v}_m(n+1) - b_m(n+1)$$ \hspace{1cm} (A.8)

(b) Updating of $u_m^*(n+1), v_m^*(n+1)$:

(i) Useful quantities:

$$\delta_{xx} = u_m^T(n+1)x_m(n+1), \quad \delta_{xy} = u_m^T(n+1)x_m(M), \quad \delta_{yy} = v_m^T(n+1)x_m(1), \quad \delta_x = 1 - \delta_{xx}, \quad \delta_y = 1 + \mu \delta_{yy}, \quad \Delta(n+1) = \delta_x \delta_y + \mu \delta_{xy}^2$$  \hspace{1cm} (A.9)

(ii) Relation to the alternative gains:

$$u_m^*(n+1) = \frac{\delta_y}{\Delta(n+1)} u_m(n+1) - \mu \frac{\delta_{xy}}{\Delta(n+1)} v_m^*(n+1)$$

$$v_m^*(n+1) = \frac{\delta_{xy}}{\Delta(n+1)} u_m(n+1) + \frac{\delta_x}{\Delta(n+1)} v_m(n+1)$$

(c) Updating of the Forward Linear Prediction (FLP) solution:

(i) Updating of $a_m(n+1)$

$$a_m(n+1) = a_m(n) + f_m^*(n+1) - g_m^*(n+1)$$  \hspace{1cm} (A.10)

(ii) Useful quantities:

$$\delta_m^2(n+1) = \frac{\delta_{xx} - \delta_{yy}^2}{\delta_{xx}}, \quad \delta_m(n+1) = \delta_{xx}, \quad \delta_m^*(n+1) = \frac{\delta_{xx} + \delta_{yy}^2}{\delta_{xx}}$$

(iii) Updating of $\Delta_f(n+1)$:

$$\Delta_f(n+1) = \lambda \Delta_f(n) + \left( f_m^2(n+1) \right)^2 (1 - \delta_m^2(n+1)) - \mu g_m^2(n+1) (1 - \delta_m^2(n+1))$$

$$\Delta_f(n+1) = \sigma^2 + \Delta_f(n+1)$$ \hspace{1cm} (A.11)

Useful and recommended substitution:

$$\Delta_f(n+1) = \sigma^2 + \Delta_f(n+1)$$ \hspace{1cm} (A.12)

where $\sigma$ is a properly chosen small quantity.

Or, if $\Delta_f(n+1)$ is known:

$$\Delta_f(n+1) = \lambda \Delta_f(n+1)$$ \hspace{1cm} (A.13)

(d) Updating of the Backward Linear Prediction (BLP) solution:

(i) Of the BLP filter coefficients:

$$b_m(n+1) = b_m(n) + f_m^*(n+1) - g_m^*(n+1)$$  \hspace{1cm} (A.14)
(ii) Recursive computation of $\Delta b(n+1)$
\[
\Delta b(n+1) = \lambda \Delta b(n) + (f b_m(n+1))^2 (1 + \delta^2(n)) - \mu g(n+1) \delta^2(n+1)
\]
Useful and recommended substitution: $\Delta b(n+1) = \lambda \sigma^2 + \Delta b(n+1)$, where $\sigma$ is a properly chosen small quantity.

(e) Updating of the filter coefficients:

(i) Computation of the “leading” and the “trailing” prediction error
\[
f_m(n+1) = z(n+1) + c_T M(n)x_m(n+1), \quad g(n+1) = z(M) + c_T M(n)x_m(M)
\]

(ii) Final updating of $c(n+1)$
\[
c_m(n+1) = c_m(n) + f_m(n+1)u^*(n+1) - \mu g_m(n+1)v^*(n+1)
\]

References