EFFICIENT LINEAR PREDICTION FOR DIGITAL AUDIO EFFECTS

Florian Keiler[1], *Daniel Arfib*[2], *and Udo Zölzer*[1]

{florian.keiler, udo.zoelzer}@unibw-hamburg.de, arfib@lma.cnrs-mrs.fr
[1]University of the Federal Armed Forces, Holstenhofweg 85, D-22043 Hamburg, Germany
[2]CNRS-LMA, 31 Chemin Joseph Aiguier, 13402 Marseille Cedex 20, France

ABSTRACT

In many audio applications an appropriate spectral estimation from a signal sequence is required. A common approach for this task is the *linear prediction* [1] where the signal spectrum is modelled by an all-pole (purely recursive) IIR (infinite impulse response) filter. Linear prediction is commonly used for coding of audio signals leading to *linear predictive coding* (LPC). But also some audio effects can be created using the spectral estimation of LPC.

In this paper we consider the use of LPC in a real-time system. We investigate several methods of calculating the prediction coefficients to have an almost fixed workload each sample. We present modifications of the autocorrelation method and of the Burg algorithm for a sample-based calculation of the filter coefficients as alternative for the gradient adaptive lattice (GAL) method. We discuss the obtained prediction gain when using these methods regarding the required complexity each sample. The desired constant workload leads to a fast update of the spectral model which is of great benefit for both coding and audio effects.

1. INTRODUCTION

In LPC the current input sample $x(n)$ is approximated by a linear combination of past samples of the input signal. The prediction of $x(n)$ is computed using an FIR filter by

$$\hat{x}(n) = \sum_{k=1}^{p} a_k x(n-k) \tag{1}$$

where p is the *prediction order* and a_k are the prediction coefficients. With the z-transform of the prediction filter

$$P(z) = \sum_{k=1}^{p} a_k z^{-k} \tag{2}$$

the difference between the original input signal $x(n)$ and its prediction $\hat{x}(n)$ is evaluated in the z-domain by

$$E(z) = X(z) - \hat{X}(z) = X(z)[1 - P(z)] = X(z)A(z). \tag{3}$$

The difference signal $e(n)$ is called *residual* or *prediction error* and its calculation is depicted in Figure 1(a). Here the feed forward prediction is considered where the prediction is calculated in forward direction from the input signal.

Using the *excitation* $\tilde{e}(n)$ as input to the all-pole filter

$$H(z) = \frac{1}{1 - P(z)} \tag{4}$$

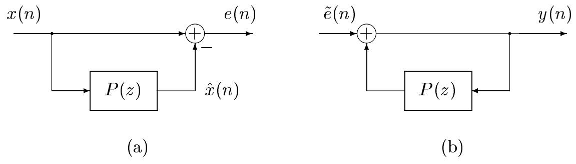

Figure 1: LPC structure with feed forward prediction. (a) Analysis, (b) Synthesis.

produces the output signal

$$Y(z) = \tilde{E}(z) \cdot H(z) \tag{5}$$

where $H(z)$ can be realized with the FIR filter $P(z)$ in a feedback loop as shown in Figure 1(b). If the residual $e(n)$ calculated in the analysis stage is fed directly into the synthesis filter, the input signal $x(n)$ will be ideally recovered.

The IIR filter $H(z)$ is termed *synthesis filter* or *LPC filter* and it models – except for a gain factor – the input signal $X(z)$. For speech coding this filter models the time-varying vocal tract. The filter $A(z) = 1/H(z)$ for calculating the residual from the input signal (see Eq. (3)) is called the *inverse filter*.

With optimal filter coefficients, the residual energy is minimized. This can be exploited for efficient coding of the input signal where the quantized residual $\tilde{e}(n) = Q\{e(n)\}$ is used as excitation to the LPC filter.

Other applications for linear prediction are audio effects such as the cross-synthesis of two sounds or the pitch shifting of one sound with formant preservation [2, 3, 4, 5].

In this paper, modifications of two commonly used linear prediction methods (autocorrelation method, Burg algorithm) are presented to get methods which are suited for real-time computation, i.e. to get a similar workload each sample. Furthermore the quality of the spectral model computed by different linear prediction methods is compared regarding the required computation time. We consider here linear prediction methods for a computation of the residual with zero delay. Thus, the prediction coefficients are computed from past samples of the input signal and the methods are suited for audio coding using the ADPCM structure where no transmission of the filter coefficients is required. The fast filter update coming from the similar workload each sample leads to better spectral models than block-based approaches where the coefficients are held constant for the duration of one block. With a fast update of the spectral model no interpolation of the filter coefficients between frames is required as usually performed in audio effects based on a frame-by-frame LPC analysis [3, 4].

2. CALCULATION OF THE PREDICTION ERROR

Regarding to Equations (1) and (3), the prediction error can be calculated by a standard FIR structure which requires the direct FIR coefficients a_k. Some methods for calculating the prediction coefficients are based on the lattice structure [6], as shown in Figure 2. In the lattice structure the signals $f_m(n)$ and $b_m(n)$ are used which are the forward and backward prediction errors of an m-th order predictor. Forward prediction means the prediction from past samples while in backward prediction a sample is predicted from future samples.

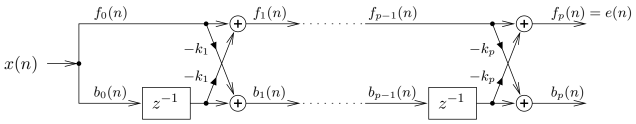

Figure 2: Lattice structure.

In the lattice methods the lattice or PARCOR (partial correlation) coefficients k_i are calculated instead of the direct FIR coefficients. Although it is possible to calculate the direct coefficients from the lattice ones, sometimes it is useful to perform the filter operation in the lattice structure. The main advantage of the lattice coefficients is that the stability of the LPC filter is guaranteed for $|k_i| < 1$. Furthermore, if a predictor of order $m-1$ is already known, for a predictor of order m only the coefficient k_m has to be calculated. For the direct FIR coefficients normally the complete coefficient set has to be changed in this case.

In case of using the lattice structure, the lattice states $b_i(n)$ have to be recalculated if the lattice coefficients k_i are changed. This problem does not occur in the direct FIR structure since the filter states are equal to past samples of the input signal and they are independent of the used coefficient set.

3. LINEAR PREDICTION METHODS

In this section we describe some methods to calculate the prediction coefficients for minimizing the residual energy. First the standard block-based approaches of the Burg Algorithm [6, 7] and the autocorrelation method [1, 7] are summarized. Then the sample-based gradient adaptive lattice (GAL) method [7, 8] is described. Finally we present modifications of the block-based methods for a sample-based calculation in a real-time system.

3.1. Burg Algorithm

The Burg algorithm is based on the lattice structure and it minimizes for a predictor of order m in a block of length N the sum of the energies of the forward prediction error $f_m(n)$ and of the backward prediction error $b_m(n)$.

The initialization of the forward and backward prediction errors of order zero for the considered block is obtained by

$$f_0(n) = x(n) \quad , n = 0, \ldots, N-1 \tag{6}$$
$$b_0(n) = x(n) \quad , n = 0, \ldots, N-1 \tag{7}$$

where n denotes the time index in the considered block. For $m = 1, \ldots, p$ the following operations are performed:

- Calculation of the m-th lattice coefficient

$$k_m = \frac{2 \sum\limits_{n=m}^{N-1} \left[f_{m-1}(n) b_{m-1}(n-1) \right]}{\sum\limits_{n=m}^{N-1} \left[f_{m-1}^2(n) + b_{m-1}^2(n-1) \right]} \tag{8}$$

- Recursively calculation of the forward and backward prediction errors of order m (see Figure 2)

$$\begin{aligned} f_m(n) &= f_{m-1}(n) - k_m b_{m-1}(n-1), \\ &\qquad n = m+1, \ldots, N-1 \end{aligned} \tag{9}$$
$$\begin{aligned} b_m(n) &= b_{m-1}(n-1) - k_m f_{m-1}(n), \\ &\qquad n = m, \ldots, N-1 \end{aligned} \tag{10}$$

3.2. Autocorrelation Method

The autocorrelation method minimizes the prediction error $e(n)$, or in terms of the lattice structure, the forward prediction error. For a block of length N an approximation of the autocorrelation sequence is calculated by

$$R(i) = \frac{1}{N} \sum_{n=i}^{N-1} u(n) u(n-i) \tag{11}$$

where $u(n) = x(n) \cdot w(n)$ is a windowed version of the considered block $x(n)$, $n = 0, \ldots, N-1$. Normally a Hamming window is used [9]. For a predictor of order p the filter coefficients a_i for $i = 1, \ldots, p$ are obtained by solving the normal equations

$$\sum_{k=1}^{p} a_k R(i-k) = R(i) \quad , i = 1, \ldots, p. \tag{12}$$

An efficient solution of the normal equations is performed by the Levinson-Durbin recursion [1]. First the energy of the predictor of order zero is initialized to $E_0 = R(0)$. Afterwards the following operations are performed for $m = 1, \ldots, p$, where $a_k^{(m)}$ denotes the k-th coefficient of an m-th order predictor.

$$k_m = \frac{R(m) - \sum_{k=1}^{m-1} a_k^{(m-1)} R(m-k)}{E_{m-1}} \tag{13}$$
$$a_m^{(m)} = k_m \tag{14}$$
$$a_k^{(m)} = a_k^{(m-1)} - k_m a_{m-k}^{(m-1)} \quad , k = 1, \ldots, m-1 \tag{15}$$
$$E_m = (1 - k_m^2) E_{m-1} \tag{16}$$

3.3. Gradient Adaptive Lattice (GAL)

As in the block-based Burg algorithm, in the gradient adaptive lattice method the lattice coefficients are used and the sum of the forward and backward prediction errors is minimized [7, 8]. Using the approximation of the error energy of the m-th order predictor

$$\hat{J}_m(n) = f_m^2(n) + b_m^2(n) \tag{17}$$

yields with the steepest decent approach the coefficient update

$$k_m(n+1) = k_m(n) - \mu_m \frac{\partial \hat{J}_m(n)}{\partial k_m(n)} \tag{18}$$

with the gradient weights μ_m. Applying the recursions after Equations (9) and (10) for the current time index n leads to

$$\frac{\partial \hat{J}_m(n)}{\partial k_m(n)} = -2\left[f_m(n)b_{m-1}(n-1) + b_m(n)f_{m-1}(n)\right]. \quad (19)$$

Thus putting (19) into (18) gives a formula for a sample-by-sample update of the lattice coefficients.

The simplest approach is to choose the μ_m values constant. Simulations have shown that the optimum value of μ (equal for all orders for simplicity) depends highly on the used signals, the optimum value varies approximately in the range from 1 to 10. Better results are expected for gradient weights which are adaptively dependent on the expectation value of the sum of the forward and backward prediction error energies. An approximation of this expectation value can be recursively calculated by [7]

$$D_m(n) = \lambda D_m(n-1) + (1-\lambda)\left[f_{m-1}^2(n) + b_{m-1}^2(n-1)\right] \quad (20)$$

where $0 < \lambda < 1$ influences the weight of older samples. The gradient weights are obtained by

$$2\mu_m = \frac{\alpha}{D_m(n)} \quad (21)$$

with a constant value α which is normally chosen to $\alpha = 1 - \lambda$ for a recursive formulation of the Burg algorithm [7]. In our simulations using different sounds the optimum value was $\lambda \approx 0.995$ independent of the used sound.

3.4. Modification of Block-based Methods for Real-Time Computation

Both the autocorrelation method and the Burg algorithm require first an initialization process before the prediction coefficients are computed recursively. The real-time computation of a coefficient set of order p is spread over $p+1$ samples; one sample for the initialization and one sample each for the p coefficients. Thus, with the counter $i = 0, 1, \ldots, p, 0, 1, \ldots$ (changing every sample) we get the following procedure:

- For $i == 0$ perform the initialization.
- For $i \in \{1, p\}$ calculate the coefficient with index i.

3.4.1. Successive Burg Algorithm

In the initialization process ($i = 0$) the operations of Equations (6) and (7) are performed for setting both the forward and backward prediction error of order zero to the input samples $x(n)$ in the considered block. For $i = 1, \ldots, p$ one coefficient k_i is calculated according to Equation (8), and applying the recursions of order $m = i$ after Equations (9) and (10), the forward and backward prediction errors of i-th order are computed which are required in the following sample for computing k_{i+1}. The new k_i replaces the previously used k_i. Since one coefficient is changed, a recalculation of the lattice states prior to the filter operation is required.

3.4.2. Successive Autocorrelation Method

In the Levinson-Durbin recursion the autocorrelation sequence $R(i)$ for $i = 0, \ldots, p$ is required, see Equation (11). In the initialization process ($i = 0$) first the input data block is windowed, where normally a Hamming window is used; then $R(0)$ is computed. For $i \in \{1, p\}$ the value $R(i)$ is computed followed by the Durbin recursion of order $m = i$ as given in Equations (13) to (16). Thus, the calculation of the complete set of the direct FIR coefficients $a_k^{(p)}$ of order p requires $p+1$ samples. If the standard FIR structure is used, the use of the coefficients in the filter operation has to be delayed by $p+1$ samples.

3.5. DSP Workload of the Sample-Based Methods

The required instructions per sample on the Motorola DSP 56307 (DSP=Digital Signal Processor) for the described sample-based methods including the filter operations are summarized in Table 1. For the GAL (with adaptive gradient weights) only an estimation is given since it is not implemented yet on the DSP. Furthermore, the required update of the lattice states is only estimated to be of the order of $3p^2$, as in the Burg algorithm. If not using the states update, during the GAL coefficient update the prediction error using the states corresponding to the old coefficient set is calculated.

Method	Computation	DSP instructions
suc.	FIR	$p + 15$
autoc.	coeff calc. ($i = 0$)	$4N + 55$
	coeff calc. ($i = 1, \ldots, p$)	$3N + 5i + 172$
suc.	lattice + states update	$3p^2 + 19p - 30$
Burg	coeff calc. ($i = 0$)	$3N + 14$
	coeff calc. ($i = 1, \ldots, p$)	$9(N - i) + 81$
GAL	lattice states update	$O(3p^2)$
	coeff calc. + std. lattice	$90p + 1$

Table 1: Required DSP instructions per sample for real-time prediction methods and filter operations dependent on block length N and prediction order p.

Table 2 shows the maximum workloads per sample for calculating the filter coefficients, i.e. the filter operations to calculate the prediction are not considered. In the GAL the prediction order p has the greatest influence on the complexity. In this method for each sample p divisions are required which are very expensive on a DSP. Notice that in the Burg algorithm the maximum workload is only influenced by the block length N which is also the case in the autocorrelation method for long blocks.

Method	DSP instructions
suc. autocorrelation	$\max\{4N + 55, 3N + 5p + 172\}$
suc. Burg	$9N + 72$
GAL	$90p + 1$

Table 2: Maximum workload per sample for coefficient calculation.

4. SIMULATION RESULTS

In this section we present some simulation results to compare the performance of the sample-based linear prediction methods regarding the required DSP workload. For the simulations we use two short sequences from a piano and a triangle sound, respectively. The duration is approximately 0.8 seconds each and the sequences are taken from tracks 39 and 32 of the EBU CD [10]. The following parameters are used:

- Successive autocorrelation method:
  $p = 4, \dots, 20$ with $N = 24, \dots, 256$;
  $p = 25, \dots, 40$ with $N = 50, \dots, 400$
- Successive Burg algorithm:
  $p = 4, \dots, 10$ with $N = 12, \dots, 160$;
  $p = 12, \dots, 20$ with $N = 24, \dots, 160$;
  $p = 25, \dots, 40$ with $N = 50, \dots, 200$
- GAL with $\lambda = 0.995$: $p = 1, \dots, 40$

For each parameter set the prediction error is calculated by using the described algorithms. Afterwards the segmental prediction gain (segG_p) is calculated which is the average value of the prediction gains of small blocks of 100 samples duration. The prediction gain of a length L block is computed by

$$G_p = 10 \log_{10} \left(\frac{\sum_{n=1}^{L} x^2(n)}{\sum_{n=1}^{L} e^2(n)} \right) \text{ dB.} \tag{22}$$

The prediction gain is a measure for the quality of the spectral model. The inverse values of the averaged spectral flatness measures are appr. 53 dB for the piano and 23 dB for the triangle, which are upper bounds for the achievable prediction gains [11]. Figure 3 shows the obtained seg$G_p$values dependent on the maximum DSP workload per sample given in Table 2. For the block-based methods the plots show lines for constant values of the prediction order p. In the plots for the Burg algorithm (on the right) additionally the results of the GAL (dashed) are shown.

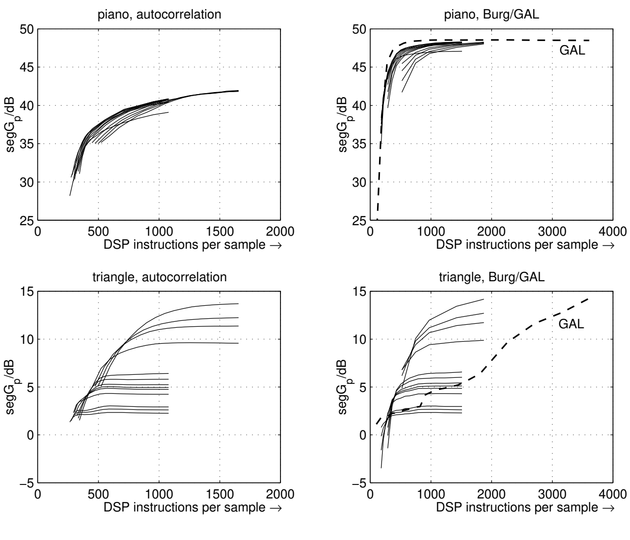

Figure 3: Segmental prediction gain over maximum DSP instructions per sample for sample-based LP methods; lines with constant pred. order p for the block-based methods.

From these results it can be seen that the performance of the methods depend highly on the input signals. It is not possible to say in general which method gives a better prediction gain for a fixed DSP workload. The GAL and Burg methods perform better for the piano sequence while the Burg algorithm does not work satisfying at very small block lengths for the triangle sequence (seg$G_p < 0$ in some cases). The prediction gain decreases for an increasing prediction order in case of using the Burg algorithm at short blocks. It seems that the GAL performs for a given prediction order like the Burg algorithm with a long block length. For the triangle sequence none of the methods is able to reach the maximum prediction gain with the used parameters. Notice that in these results the complexity for the filter operations is not included which is much higher for the lattice-based Burg and GAL methods.

5. MUSICAL APPLICATIONS OF LPC

LPC has been very active in the very early years of computer music, technically as well as musically. Some publications regarding to LPC based audio effects and musical applications are those from Dodge [5], Lansky [4], and Moorer [3]. Impressive pieces of music have been achieved at a time where real-time was only a dream.

Nowadays some powerful software implementations of musical processing offer LPC as a way to modify, transform, mutate, hybridize sounds. Csound is a classical one and it has functions to perform LPC filter operations. SoundSculpt follows its earlier version, named SVP, with a cross-synthesis between two sounds using LPC.

First the general structure used to produce LPC based audio effects is presented and then some special applications are described.

5.1. General Approach

Producing a sound is mainly based on the LPC analysis/synthesis structure of Figure 1. As mentioned at the beginning, if using the residual coming out of the analysis stage as excitation to the synthesis filter, the original sound is recovered. Now either the excitation or the synthesis filter or both of them are changed to generate a sound different from the input. Figure 4 shows the LPC synthesis with excitation $x_s(n)$ and synthesis filter $H_s(z)$. The subscript s indicates "synthesis". The excitation $x_s(n)$ may be a processed version of the residual or a sound not related to the signal which is used for processing the spectral model. As explained in the following, a gain factor $g(n)$ is required for the amplitude control of the synthesized sound $y(n)$.

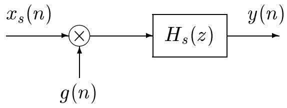

Figure 4: LPC synthesis structure for audio effects.

5.1.1. Estimation of the Gain Factor

Using as excitation the residual $e(n)$ computed by the inverse filter operation

$$E(z) = X(z)/H_s(z) \tag{23}$$

will result exactly in the input signal $x(n)$. Thus if changing the excitation, the gain factor $g(n)$ is used to scale the amplitude of the used excitation $x_s(n)$ in that way that the excitation's energy is similar to the energy of the residual $e(n)$. This can be done recursively to have a permanent update of the gain factor. The energies of the residual and of the excitation can be calculated recursively (see Figure 5) by the first order IIR filter

$$E_e(n) = \lambda E_e(n-1) + (1-\lambda)e^2(n) \tag{24}$$

$$E_{x_s}(n) = \lambda E_{x_s}(n-1) + (1-\lambda)x_s^2(n). \tag{25}$$

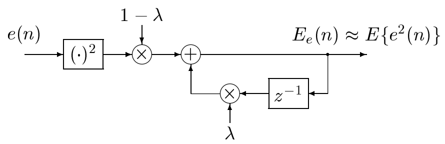

Figure 5: Recursive calculation of the signal energy.

This works similar as the error energy calculation in the adaptive gradient lattice, see Equation (20). Now the gain factor may be computed by

$$g(n) = \sqrt{\frac{E_e(n)}{E_{x_s}(n)}}. \tag{26}$$

But this expression gives sometimes very rapidly changing values. To smooth the gain factor it can be calculated recursively as well by

$$g(n) = \begin{cases} \lambda g(n-1) + (1-\lambda)\sqrt{\frac{E_e(n)}{E_{x_s}(n)}} & , E_{x_s}(n) \neq 0 \\ g(n-1) & \text{, otherwise} \end{cases} \tag{27}$$

Experiments have shown that a value of $\lambda = 0.99$ gives satisfactory results. For greater values of λ more weight is given to the energy of past samples which results in slower varying estimates. The parameter λ is comparable to a block length if a block of samples is used to estimate the signal energy. Since the residual is assumed to be spectrally flat, the gain factor has to be chosen smaller than given in Eq. (27) if the excitation $x_s(n)$ has other spectral properties to avoid clipping of the output signal.

Moorer [3] reported a scaling of the output signal by using the energies of the original signal and of the output signal where a piecewise linear gain factor is used.

Figure 6 shows some time plots for clarifying the behaviour of the gain factor. From top to bottom are shown the input signal (used to determine the LPC model), the energies of the residual and of the excitation, and the gain factor. The signal $x(n)$ for calculating the synthesis filter is the speech utterance "sun", the excitation $x_s(n)$ is a guitar sound. It can clearly be seen from the energy of the residual $e(n)$ that in the unvoiced part at the beginning of the sequence the predictor does not work very well. The gain factor adapts to the changing energies of residual and excitation. In Figure 7 the spectra of the speech input and the calculated LPC filters for an filter order $p = 40$ are shown at times 0.1 s and 0.32 s, respectively. This figure demonstrates well the offset by the gain factor between the original and the LPC model spectra.

5.1.2. Audio Effects based on Speech Models

Many reported LPC based audio effects are using a synthesis filter calculated from a speech input signal [3, 4, 5]. The LPC filter represents the time-varying properties of the vocal tract by modelling its formant frequencies. For a proper vocal tract model the prediction order should be slightly higher than the sampling rate in kHz [4] which is equivalent to modelling one resonance frequency each kHz since each resonance frequency requires two conjugate complex poles. Zeros in the spectral model may be approximated by a small number of poles [9]. Thus for a good model the prediction order should be a small amount higher than the sampling rate in kHz.

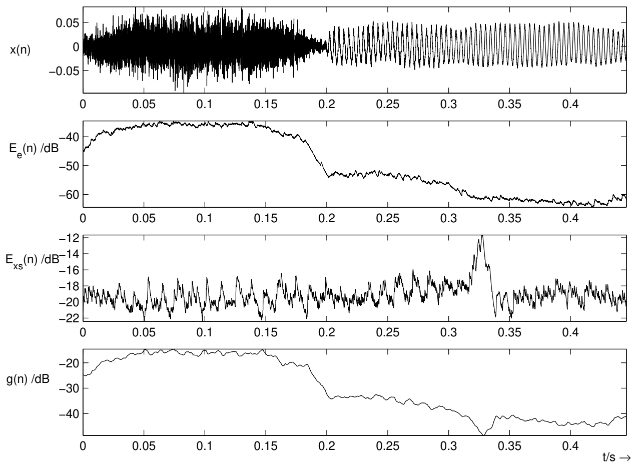

Figure 6: Energies and gain factor for cross-synthesis.

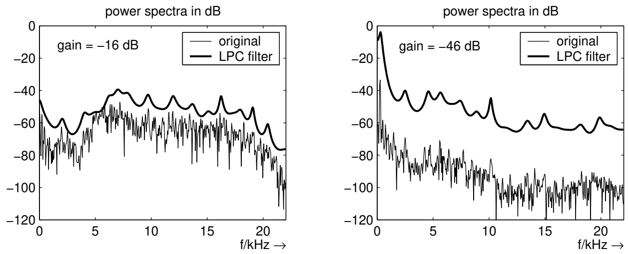

Figure 7: Power spectra of original and LPC filter with an prediction order of $p = 40$, unvoiced (left) and voiced (right).

5.2. Changing the Filter or the Excitation Using one Sound

In this section we explain some effects based on the LPC of one sound. Thus the standard LPC analysis/synthesis is used, but modifications are applied either to the excitation or to the synthesis filter or to both of them.

5.2.1. Modelling the Excitation

The excitation for ideal signal recovery can be modelled as a signal consisting of pulses plus noise, as used in the speech production model [1]. For voiced sounds the periodicity of the pulses determine the pitch of the sound while for unvoiced sounds the excitation is noise-like. The modelling of the excitation requires an algorithm for voiced/unvoiced separation, which can crudely be a switch between pulses and random noise. Such decisions are not so easy to make automatically and a good pitch detector should be used. But then all manipulations are possible.

5.2.2. Frequency Warping

The synthesis filter can be modified by taking a warped version of the initial filter $H_s(z) = H(\tilde{z})$ which moves the formants and will give a "donald duck" voice without altering its pitch. In [12] the frequency warping is performed by using the allpass function

$$\tilde{z} = \frac{d + z^{-1}}{1 + dz^{-1}} \quad \text{with} -1 < d < 1. \tag{28}$$

5.2.3. Time Scaling

The time duration on a sound can be modified by time-stretching the excitation and updating the synthesis filter at a different rate [4]. This preserves the formant structure. If the time duration is expanded even for a sample-based update of the filter an interpolation of the coefficients may be required.

5.2.4. Pitch Shifting

To modify the pitch of the resulting signal, for voiced parts the pitch of the modelled excitation can be changed. This effect can also be combined with the frequency warping to independently change the formants induced by the synthesis filter. It is also possible to apply chorus or flanging or whatever effect over the ideal excitation before it is fed into the synthesis filter.

5.3. Cross-Synthesis between two Sounds

A very classical algorithm is to use two different sounds (x_1 and x_2) and take the residual of one (x_2) as the excitation to the LPC filter of the other (x_1). The main structure is shown in Figure 8. This effect is very reminiscent of the "vocoder effect" that comes from professional vocoder (channel vocoder) or phase vocoder. The synthesis filter is computed from signal $x_1(n)$ by using LPC analysis with a high filter order for representing the harmonic structure of this sound. To compute the excitation an LPC inverse filtering with a low prediction order is performed to whiten the input $x_2(n)$. As reported by Moorer [3] an order of 4 to 6 works well for this whitening process. The cross-synthesis gives good results if a speech signal is used to compute the synthesis filter which results for example in the "talking guitar".

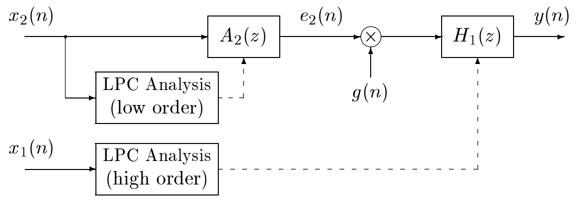

Figure 8: Cross-synthesis structure.

For musical satisfactory results the two used sounds have to be synchronized. Thus, for the case of mixing speech and music, the played instrument must fit to the rhythm of the syllables of the speech. The performance is improved if either speech or music is coming from a prerecorded source and the other sound is produced to match to the recording [3].

5.3.1. Using Spectrally flat Signals as Excitation

If a sound is spectrally almost flat, this sound can be used directly as excitation, thus using $e_2(n) = x_2(n)$ in Figure 8. For example the sound of the ocean (or another noise-like unpitched signal) can be used as excitation. This effect is very different from the preceding ones because now it is the pitch of the sound x_1 which is given to the resulting sound if the prediction order for the spectral model of x_1 is high enough to capture its harmonic structure. This effect can also work for some music signals like the sound of a distorted guitar, but the pitch structure is then a mixture of the two pitches. In this case the resulting sound will have some loss of high frequencies.

5.3.2. Using Synthetic Excitations

The excitation may also be processed directly without any modelling. For example a signal like a sawtooth with the desired pitch frequency can be used which sounds like a vocoder.

6. CONCLUSIONS

We presented algorithms for the sample-based calculation of linear prediction (LP) coefficients. Apart from the well-known gradient adaptive lattice (GAL) we presented modifications of the commonly used block-based approaches like autocorrelation method and Burg algorithm. The performance of the considered methods depends highly on the input signal. The lattice-based methods GAL and Burg algorithm have the disadvantage that in the lattice filter structure an update of the filter states is required for every coefficient change. An improvement on the lattice states update is under further investigation. For the GAL a modification may be developed which reduces the very high complexity.

Apart from possible coding applications (e.g. ADPCM with low delay) the sample-by-sample update of the spectral model is beneficial in LP-based audio effects. Some audio effects have been described including a suggestion for an efficient way to calculate the permanently changing gain factor required in the realization of the effects.

7. REFERENCES

[1] John Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, Apr. 1975.

[2] James A. Moorer, "Audio in the New Millennium," *Journal of the AES*, vol. 48, no. 5, pp. 490–498, May 2000.

[3] James A. Moorer, "The Use of Linear Prediction of Speech in Computer Music Applications," *Journal of the AES*, vol. 27, no. 3, pp. 134–140, Mar. 1979.

[4] P. Lansky, "Compositional Applications of Linear Predictive Coding," in *Current Directions in Computer Music Research*, Max V. Mathews and John Pierce, Eds., pp. 5–8. MIT Press, Cambridge, Mass., 1989.

[5] C. Dodge, "On *Speech Songs*," in *Current Directions in Computer Music Research*, Max V. Mathews and John Pierce, Eds., pp. 9–17. MIT Press, Cambridge, Mass., 1989.

[6] John Makhoul, "Stable and Efficient Lattice Methods for Linear Prediction," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, no. 5, pp. 423–428, Oct. 1977.

[7] Sophocles J. Orfanidis, *Optimum Signal Processing, An Introduction*, McGraw-Hill, Singapore, second edition, 1990.

[8] Peter M. Clarkson, *Optimum and Adaptive Signal Processing*, CRC Press, Boca Raton, Florida, 1993.

[9] D. O'Shaugnessy, *Speech Communication. Human and Machine*, Addison-Wesley, New York, 2nd edition, 2000.

[10] European Broadcasting Union (EBU) Technical Centre, Brussels, *Sound Quality Assessment Material. Recordings for Subjective Tests, CD and User's Handbook for the EBU-SQAM Compact Disc, Tech. 3253-E*, Apr. 1988.

[11] N.S. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[12] P. Lansky and K. Steiglitz, "Synthesis of Timbral Families by Warped Linear Prediction," in *Proc. of the IEEE ICASSP*, Atlanta, Georgia, 1981.