AN OVERVIEW OF DIGITAL TECHNIQUES FOR PROCESSING SPEECH SIGNALS

Murat Kunt
Signal Procesing Laboratory
Swiss Federal Institute of Technology
16 Ch. de Bellerive
CH - 1007 Lausanne, Switzerland

and

Heinz Hugli
Mircrotechnique Institute
University of Neuchatel
Rue de la Maladière 71
CH - 2007 Neuchatel, Switzerland

ABSTRACT

This paper discusses major digital signal processing methods used in processing speech signals. Basic tools, such as the discrete Fourier transform, the z transform and linear filter theory are briefly introduced first. A general view of fast transformation algorithms and most widely used particular fast transformations are given. Linear prediction is then described with a particular emphasis on its lattice structure. A brief introduction to homomorphic processing for multiplied and convolved signals and to its applications in speech processing is given. Recalling some fundamentals of the speech signal, various speech analysis and synthesis models are described, showing which kind of processing methods are

NATO ASI Series, Vol. F16
New Systems and Architectures for Automatic Speech Recognition and Synthesis. Edited by R. De Mori and C. Y. Suen
© Springer-Verlag Berlin Heidelberg 1985

involved. Finally, two aspects of speech recognition are presented: feature traction and pattern matching using dynamic time warping.

1. INTRODUCTION

Because of its multidisciplinary character, digital signal processing became increasingly important in a number of scientific and technical areas. Continuous interaction between the methods and the particular applications have led to an avalanche on both sides. Increasingly sophisticated methods are developed to fulfil wider needs of a large number of applications. There is no doubt that one of the major application areas of digital signal processing is speech signals. Over the last two decades, considerable effort has been devoted to analyse, code, model, synthesize and recognize speech signals. A dozen of books are already available, presenting various aspects of digital speech processing.

This paper attempts to give a tutorial review of major digital signal processing methods used in processing speech signals. Because of space limitations and the wide range of the subject, in depth treatments are omitted. Essence of the methods and insight for the interpretation of the results are indicated whenever possible. In section two, basic methods are defined such as the discrete Fourier transform, correlation functions, the z transform, the convolution, and the linear system theory. A general view of fast transformation algorithms is given, showing structures for hardware and software. Commonly used fast transformations are also briefly indicated. The last part of this section presents the linear prediction models and tools for one dimensional signals and introduces its lattice structure, a structure that is modular and hence suitable for various implementations. In section three, homomorphic processing of multiplied and convolved signals is discussed with particular emphasis on its applications to speech signals, particularly for deconvolution. Section four gives an overview of the speech analysis and synthesis methods using previously defined tools. Speech recognition is summarized in section five with a particular emphasis on pattern matching. The objective, in these last two sections, is to point out particular digital signal processing methods used for reaching the goals.

2.0 BASIC METHODS

In this section basic signal processing methods are defined and their use in speech processing are discussed. Analysis and synthesis tools for digital signals, such as the discrete Fourier transform and the correlation function, and for systems, such as the z transform and the convolution are described first. A brief discussion on linear filters and fast transformations is presented next. The section ends with a rather detailed description of linear prediction. For more detail, the reader may consult [1] and [2].

THE DISCRETE FOURIER TRANSFORM

The discrete Fourier transform of a digital signal x(k) is a complex series defined by:

$$X(n) = \sum_{k=k_0}^{k_0+N-1} x(k)\exp(-j2\pi kn/N) \qquad (1)$$

with $n = -N/2, \ldots, N/2-1$

In this definition, only N consecutive samples of the signal are used starting at $k = k0$. The series X(n) is periodical in n with a period of N. The integer variable n represents discrete frequencies. For example $n = 0$ is the DC component and $n = N/2$ is the folding frequency, i.e. half of the sampling rate.

The inverse transform is given by:

$$x(k) = (1/N) \sum_{n=-N/2}^{N/2-1} X(n)\ \exp(j2\pi nk/N) \qquad (2)$$

with $k = k0, \ldots, k0 + N - 1$

Eq. (1) is referred to as the analysis of the signal, whereas eq. (2) is used to synthesize the signal from its Fourier Transform. From the complex numbers X(n) two real sequences are obtained. The magnitude X(n) plotted as a function of n is the magnitude spectrum. The argument arg[X(n)] is the phase spectrum. They inform on the frequency distributions of complex exponential signals composing the analysed signal x(k). If the number of samples N is small compared to the total length of the signal, these spectra are called short term spectra. On a long signal, such as a speech signal, several short term spectra can be computed. Sections of the signal used in these computations may partially overlap or may be apart. If these spectra are plotted in three dimensions as a function of the frequency n and of the time (for example time instants corrresponding to the beginning of each signal section), the resulting surface is called spectrogram. It is usually represented as a black-and-white two level image on the (n,k) plane. Additional grey levels, if available, give more precise and detailed information on the frequency variations of various components of the signal. In section 1.6 fast algorithms for computing spectrograms will be discussed.

2.2 CORRELATION FUNCTIONS

The similarity of two signals x(k) and y(k) is measured by their cross correlation function defined by:

$$\varphi_{xy}(k) = \sum_{l=-\infty}^{+\infty} x(l)\, y(k+l) \qquad (3)$$

For a given delay k of the second signal y(k) with respect to the first signal x(k), the cross correlation function is just the integral of the product of these two signals. It reaches its maximum value for the greatest similarity. If x(k) is identical to y(k), the cross correlation function is called autocorrelation function. It is given by:

$$\varphi_x(k) = \sum_{l=-\infty}^{+\infty} x(l)\ x(k+l) \tag{4}$$

Its maximum is at the origin k = 0. If this function is normalized by dividing it by the variance of the signal x(k), the result is called correlation coefficient. Its values lie between +1 and -1.

An equivalent way of computing correlation functions is obtained by taking the discrete Fourier transform of both side of eq. (3) or eq. (4). One obtains respectively;

$$\Phi_{xy}(n) = X^*(n)\ Y(n) \tag{5}$$

and

$$\Phi_x(n) = X^*(n)\ X(n) = |X(n)|^2 \tag{6}$$

These results can be proved easily. They are left as exercises to the reader.

2.3 THE z TRANSFORM

The discrete Fourier transform is a very powerful tool for analysing and synthesizing signals. It is not, however, suitable for studying signal processing systems. A more general transformation is needed. The z transform fulfils this need and becomes identical to Fourier transform in a particular case. The z transform of a signal is defined by:

$$X(z) = \sum_{k=-\infty}^{+\infty} x(k)\ z^{-k} \tag{7}$$

where z is a complex variable. A power series, such as this one, may not converge for all the possible values of z. The area of the complex plane z containing all the values for which eq. (7) converges is called convergence region.

The inverse transform is a complex integral given by:

$$x(k) = (1/2\pi j) \oint X(z)\, z^{k-1}\, dz \tag{8}$$

The integration contour must lie in the convergence region. Usually, the inverse z transform is computed by using partial fraction decomposition in which the inverse transform of each term is known. Since this transformation is linear, the sum of these partial signals gives the desired result.

It is interesting to note that if the z transform is computed on the unit circle of the z plane, i.e. for $|z| = 1$, the result is the continuous Fourier transform:

$$X(z)\Big|_{|z|=1} = X(f) = \sum_{k=-\infty}^{+\infty} x(k)\, \exp(-j2\pi kf) \tag{9}$$

If now, the continuous variable f is replaced by a discrete variable n with $f_n = n\Delta f = n/N$, or equivalently, if the z transform is computed on equally spaced points of the unit circle, the result is the discrete Fourier transform (1).

2.4 CONVOLUTION

Let us consider a signal processing system S which acts on the input signal x(k) to produce an output signal y(k):

$$y(k) = S[x(k)] \tag{10}$$

If it is required from the system to be linear, the superposition principle is satisfied, i.e.:

$$S[ax_1(k) + bx_2(k)] = aS[x_1(k)] + bS[x_2(k)] \tag{11}$$

An additional constraint may be required to be shift invariant. In this case, if the response to x(k) is y(k), the response to x(k-k0) is y(k-k0). Linear shift invariant systems are completely specified with their response to a unit sample, i.e. by their impulse response g(k). To see this, let us write the input signal in terms of unit samples:

$$x(k) = \sum_{1=-\infty}^{+\infty} x(1)\ d(k-1) \qquad (12)$$

where d(k) is the unit sample having the value 1 at the origin k=0 and 0 elsewhere. Using eq. (12) as the input signal, we have:

$$\begin{aligned} y(k) = S[x(k)] &= \sum_{1=-\infty}^{+\infty} x(1)\ S[d(k-1)] \\ &= \sum_{1=-\infty}^{+\infty} x(1)\ g(k-1] = x(k) * g(k) \qquad (13) \end{aligned}$$

where g(k) is the response of the system to d(k). Eq. 13 is called convolution product or convolution. Notice the similarity between correlation and convolution.

By taking the Discrete Fourier transform of both sides of eq. (13) a simpler form is obtained:

$$Y(n) = X(n)\ G(n) \qquad (14)$$

where G(n) is the frequency response of the system or the discrete Fourier transform of its impulse response.

A similar result is obtained by taking the z transform of both sides of eq. (13):

$$Y(z) = X(z)\ G(z) \qquad (15)$$

where G(z) is the transfer function of the system.

2.5 LINEAR FILTERS

A linear shift invariant system with an impulse response g(k) is a filter. This terminology results directly from eq. (14). Since X(n) and the frequency response G(n) are simply multiplied, the selection of a particular function G(n) may attenuate or amplify some frequency components of the input signal to the detriment of others. For example, if G(n) is zero beyond a certain value of n, the output signal will not have any components at these frequencies. These components are filtered by the system.

Digital filters are divided into two broad classes depending on their impulse response. If the length of the impulse response is finite, i.e. if g(k) has only a finite number of non zero samples, the filter is called Finite Impulse Response (FIR) filter. In contrast, if the length of the impulse response is infinite, the filter is called Infinite Impulse Response (IIR) filter.

There are three equivalent ways of implementing a digital filter: 1) By convolution, using eq. (13) with finite number of samples. 2) By discrete Fourier transform, using eq. (14) and then taking the inverse transform of Y(n). 3) By using a difference equation of the following type

$$\sum_{n=0}^{N} a(n)\ y(k-n) = \sum_{m=0}^{M} b(m)\ x(k-m) \tag{16}$$

Eq. (16) seems to come out of the blue, but it is not. It is simply the inverse z transform of eq. (15) where the transfer function G(z) is a quotient of the polynomials in z, i.e..:

$$G(z) = \frac{\sum_{m=0}^{M} b(m)\ z^{-m}}{\sum_{n=0}^{N} a(n)\ z^{-n}} \tag{17}$$

Almost every transfer function can be written in this form. The rare cases where this is not possible, an approximation in the form of eq. (17) can be established by choosing N and M large enough. FIR filters are usually implemented using the convolution eq. (13). It is a simple form which can be analyzed easily. In contrast IIR filters are more suitable for a difference equation. In this case some cares must be taken. In particular, if the dynamic range of the input signal is bounded, it is desirable for the dynamic range of the output signal to be also bounded. Filters of this type are stable filters. Unstable filters do not produce any useful output, because the output signal diverges continuously.

So far, the filters that are described were shift invariant. Their characteristics (impulse response or frequency response or the coefficients of the difference equation) do not vary in time. These filters are convenient for processing stationary signals, i.e. signals whose statistical characteristics do not vary in time also. For nonstationary signals, however, such as the speech signal, it might be desirable to vary the characteristics of the filters. In this case, these filters are called time varying or adaptive. Rules must be established, of course, to find the new characteristics. Often, these rules are derived from the input signal itself, after detecting changes in its characteristics.

The main problem in filtering is to find samples of the desired impulse response or the coefficients of the differences equation in accordance with the filtering requirements. Methods for designing FIR and IIR filters are quite different. They have been investigated quite extensively over the last two decades. Because of their variety and the complexity of some of them, they will not be discussed in this text. The reader may consult [1] or [3] for detailed discussion of the subject. Presently, research efforts in digital filtering are rather oriented into implementation technologies such as VLSI and into multi-dimensional filtering. A brief discussion is included herein to emphasize basic problems related to the design and to the structures of digital filters.

When filtering is required, the frequency response of the desired filter is often a binary function: all the components within a frequency band should be completely attenuated and others in a different band should not be modified at all. Accordingly, the frequency response is a binary function switching between 0 and 1 on each cutoff frequency. These responses are impossible to match in practice. That is why they are called ideal frequency responses or ideal filters. Implementable filters can only approximate this behaviour according to an error criterion. Fig. 1 indicates major ideal frequency responses. In Fig. 2 tolerances

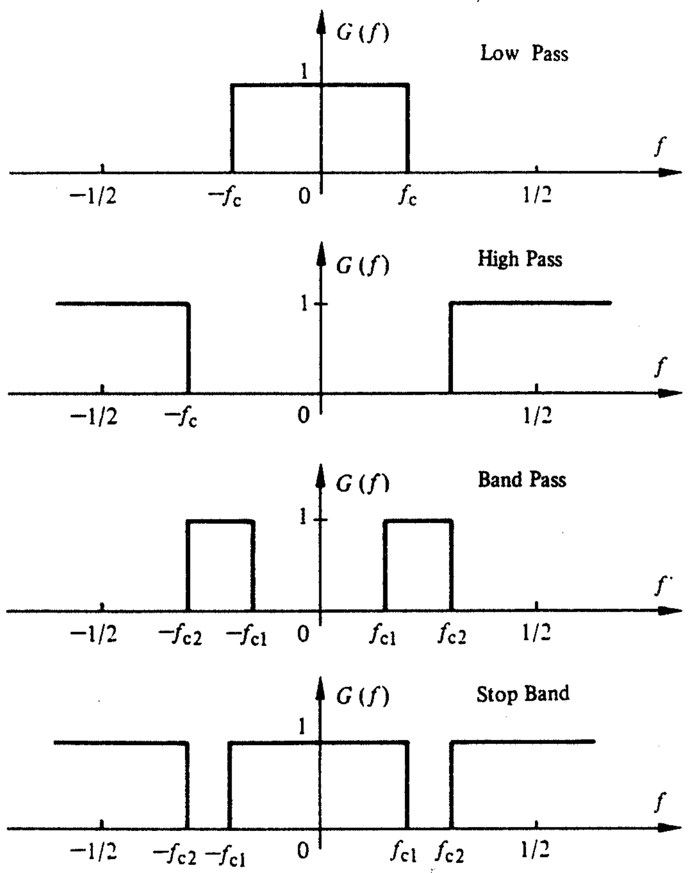


Fig. 1 Ideal frequency responses

are shown for a high pass filter. Instead of an ideal constant 1 in the passing band, oscillations within a band whose width is controlled by the parameter δ_2

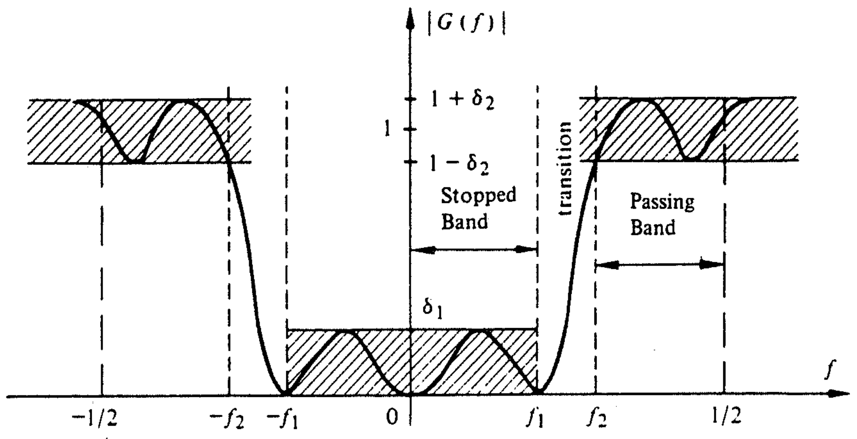


Fig. 2 Tolerance bands for a high pass filter

are tolerated. Similarly, in the stopped band, other oscillations are tolerated as regulated with the parameter δ_1. Finally, the transition between these two bands cannot occur at a unique frequency fc, but over a small frequency interval controlled with f1 and f2. Algorithms are then designed to find the impulse response or the transfer function of an implementable filter whose frequency response falls within this tolerance band.

Fig. 3 shows the block diagram of an FIR filter. This structure is quite simple and may conveniently be used with a variety of technologies (surface acoustic waves, switched capacitor circuits, charge coupled devices and VLSI). If a similar effort is produced to draw a block diagram for a difference equation, a structure is obtained as shown in Fig. 4a. This structure is not efficient because of the redundant number of delay elements. With a little algebra on the difference equation, it is possible to show that this structure is equivalent to that shown in Fig. 4b in which the number of delay elements is minimized. The big disadvantage of this last structure is its sensitivity to finite arithmetic. In fact, all the coefficients a(n) and b(m) and the signal samples must be quantized at a finite number of levels. Quantization error on one coefficient influence the entire frequency response and may unstabilize a stable filter. For this reason, especially when the number of quantization levels are not too large, this structure is divided

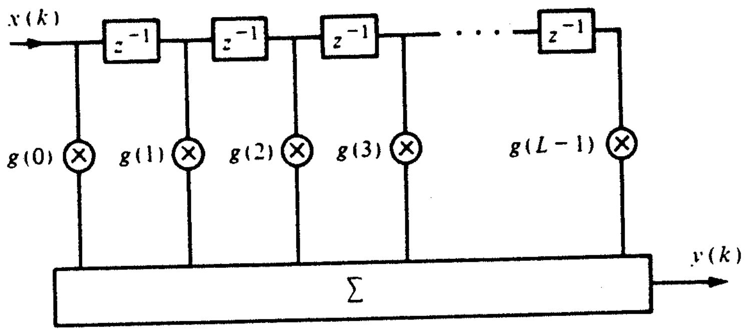


Fig. 3 Implementation structure for a transversal FIR filter

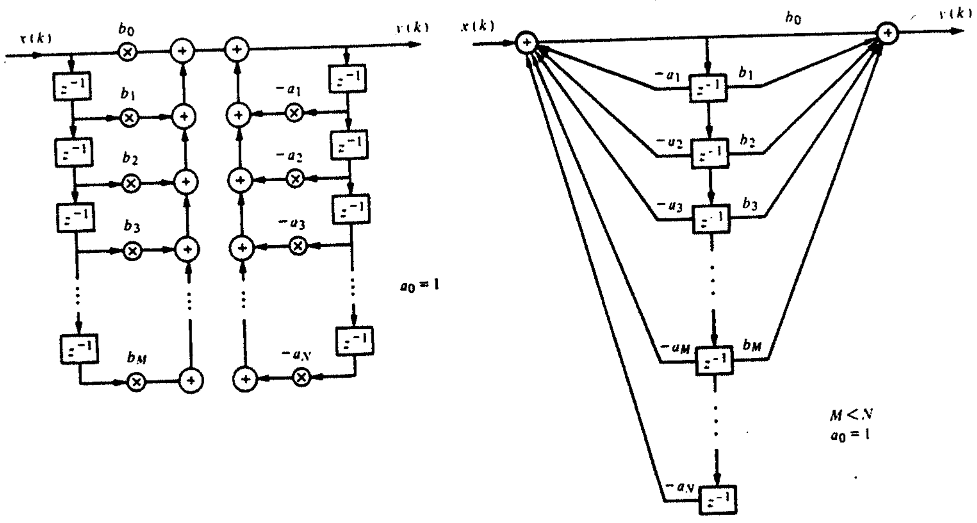


Fig. 4 Implementation structure for a recursive IIR filter. (a) structure as derived directly from a difference equation. (b) one equivalent canonical form.

into the cascade of first and second order sections.

2.6 FAST TRANSFORMATIONS

2.6.1 PRELIMINARY COMMENTS

Fast transformations play an increasingly important role in signal analysis, synthesis and coding. Since the reinvention of the fast Fourier transform in 1965, a number of complex methods that have not been used because of their computational cost, find increasingly wide applications and helped to solve various signal processing problems. With continuous progress in technology, a one-chip fast transform will soon find its socket in commonly used equipments.

What is meant by fast transformation is an algorithm which computes a linear transformation by using a minimum number of multiplications and additions. For example all the so-called fast Fourier transforms (FFT) compute the discrete Fourier transform given by eq. (1). Other linear transformations can also be computed with fast algorithms.

2.6.2 LINEAR TRANSFORMATIONS

A linear transformation of size N, transforms N samples of a signal into a set of N transform coefficients, each of them being a linear combination of the signal samples. From a mathematical point of view, a linear transformation can be expressed as:

$$X(n) = \sum_{k=0}^{N-1} a(n,k)\ x(k) \tag{18}$$

with $n = 0, ..., N-1$

where x(k) are the signal samples, a(n,k) the kernel of the transformation and X(n) the transformed coefficients. By comparing this equation with eq. (1), it can be shown that the discrete Fourier transform is a particular case of eq. (18) with $a(n,k) = \exp(j2\pi nk/N)$.

In order to compute each transformed coefficient independently from the others, the kernel a(n,k) is required to have othogonal or orthonormal rows, i.e.:

$$\sum_{k=0}^{N-1} a(n,k)a^*(m,k) = \delta(n,m) = \begin{cases} 1 \text{ if } n = m \\ 0 \text{ otherwise} \end{cases} \tag{19}$$

where * represents the complex conjugate and $\delta(n,m)$ the Kronecker symbol. If X and x denote the coefficient and sample vectors respectively with:

$$X = (X(0), X(1), ..., X(N-1))^T \tag{20}$$

and

$$x = (x(0), x(1), ..., x(N-1))^T \tag{21}$$

Eq. (18) can be written as matrix vector product:

$$X = A\ x \tag{22}$$

where A is the matrix containing all the a(n,k). To recover the signal samples from the transform coefficients, A is required to be inversible. In this case, the following equation holds:

$$A\ A^{-1} = I \tag{23}$$

where I is the unit matrix. The matrix form of eq. (19) is:

$$A^H A = I \tag{24}$$

where H represents Hermitian transpose. Since A is unique, we have $A^H = A^{-1}$ and hence:

$$A\ A^H = I \tag{25}$$

If this last equation is written in series form, one obtains:

$$\sum_{n=0}^{N-1} a(n,k)\ a^*(n,1) = \hat{o}(k,1) = \begin{cases} 1 \text{ if } n = m \\ 0 \text{ otherwise} \end{cases} \tag{26}$$

This equation implies that the columns of A are orthonormal too. A matrix A for which eqs. (19) and (26) hold, is called unitary. The transformation (18) is then a change in the N dimensional coordinate system in which x and X are expresed as vectors.

2.6.3 FAST ALGORITHMS

If the number of multiplications and additions required by a linear transformation such as (18) is counted, it is found to be $Nc = N.N = N^2$. Then, how this number can be reduced to a more or less theoretical minimum ? In general, this number is the minimum, unless there is some redundancy in the kernel so that some entries can be computed as functions of the others. Fast transform algorithms exist only for those transform matrices which have a structured redundancy. The discrete Fourier transform is one example.

Two classes of fast algorithms can be established based on the factorization of the number of samples N. In the first class, a repetitive basic structure in the algorithm, and hence in the corresponding hardware, can be found if N is an integer power of a small number such as $N = 2^m$, or $N = \alpha^m$. These algorithms are more suitable for VLSI implementations. In the second class, algorithms are designed for N factorized as a product of a series of various integer numbers such as N = N1.N2.N3....Nk. In this case the basic structure exists but changes its size according to the particular Nj in the factorization.

To see how a structured redundancy can be introduced in a transform matrix, let us consider the successive tensorial product of a set of ρ by ρ basic matrices Bi. We have:

$$A_n = \bigotimes_{i=0}^{n-1} Bi \otimes Bn\text{-}2 \otimes \ldots \otimes B0 \qquad (27)$$

where x represents the tensorial product defined by $C = A \otimes B$:

$$A = \begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix} \qquad B = \begin{pmatrix} b11 & b12 \\ b21 & b22 \end{pmatrix}$$

$$C = \begin{pmatrix} a11b11 & a11b12 & a12b11 & a12b12 \\ a11b21 & a11b22 & a12b21 & a12b22 \\ a21b11 & a21b12 & a22b11 & a22b12 \\ a21b21 & a21b22 & a22b21 & a22b22 \end{pmatrix} \qquad (28)$$

The matrix A_n is of size ρ by ρ with only $n\rho$ non redundant entries. It can be shown [1] that A_n obtained with successive tensorial products can be factored into n matrices Cj, each of size ρ by ρ and having only ρ non zero entries per line. This result is known as Good's theorem and is expressed by:

$$A_n = \prod_{j=0}^{n-1} Cj = Cn\text{-}1.Cn\text{-}2\ldots C0 \qquad (29)$$

Eqs. (27) and (29) can be illustrated with the following example with $n = \rho = 2$.

$$A_2 = \begin{pmatrix} b_{1,0,0} & b_{1,0,1} \\ b_{1,1,0} & b_{1,1,1} \end{pmatrix} \otimes \begin{pmatrix} b_{0,0,0} & b_{0,0,1} \\ b_{0,1,1} & b_{0,1,1} \end{pmatrix}$$

$$= \begin{pmatrix} b_{1,0,0} & b_{1,0,1} & 0 & 0 \\ 0 & 0 & b_{1,0,0} & b_{1,0,1} \\ b_{1,1,0} & b_{1,1,1} & 0 & 0 \\ 0 & 0 & b_{1,1,0} & b_{1,1,1} \end{pmatrix} \cdot \begin{pmatrix} b_{0,0,0} & b_{0,0,1} & 0 & 0 \\ 0 & 0 & b_{0,0,0} & b_{0,0,1} \\ b_{0,1,0} & b_{0,1,1} & 0 & 0 \\ 0 & 0 & b_{0,1,0} & b_{0,1,1} \end{pmatrix} \qquad (30)$$

A transformation using A_η as the transform matrix can be written as:

where the products should be carried out from right to left. A flow chart can be derived from eq. (31) which can be used to design various fast algorithms. Fig. 5 shows the chart of eq. (31). A basic structure is used twice. It is

$$\begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{pmatrix} = \begin{pmatrix} b_{1,0,0} & b_{1,0,1} & 0 & 0 \\ 0 & 0 & b_{1,0,0} & b_{1,0,1} \\ b_{1,1,0} & b_{1,1,1} & 0 & 0 \\ 0 & 0 & b_{1,1,0} & b_{1,1,1} \end{pmatrix} \cdot \begin{pmatrix} b_{0,0,0} & b_{0,0,1} & 0 & 0 \\ 0 & 0 & b_{0,0,0} & b_{0,0,1} \\ b_{0,1,0} & b_{0,1,1} & 0 & 0 \\ 0 & 0 & b_{0,1,0} & b_{0,1,1} \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{pmatrix} \quad (31)$$

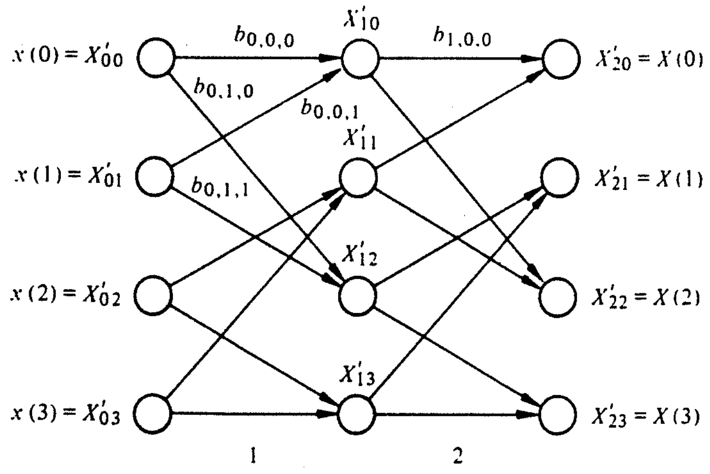


Fig. 5 Flow chart for a fast transform algorithm according to eq. (29).

possible to modify the flow charts provided that the information flow is not altered. For example, if the 2nd and 3rd lines of the rightmost matrix in (31) are swapped, the 2nd and the 3rd columns of the second matrix should also be permuted to keep the result unchanged. The corresponding flow chart is shown in Fig. 6. The basic structure of this flow chart is known as the butterfly operation which allows 'in-place' computation, i.e. the storage for input data is used for intermediary and final results.

In this scheme, the total number of multiplications and additions is reduced to Nf = ρNlog N if N is an integer power of ρ compared to Nc = N. For example, for N = 1024, which a very commonly used transform size, the saving is larger than a factor of 50 !

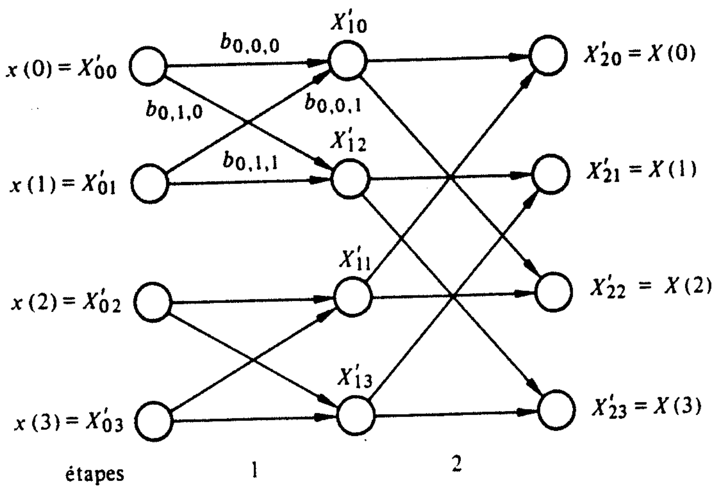


Fig. 6 Modified flow chart for 'in place computation'.

2.6.4 PARTICULAR FAST TRANSFORMS

Elaborating the general principles outlined in the previous paragraph, fast algorithms can be designed for the following particular transforms. Algorithms are not described due to the limited space of this text.

1) The discrete Fourier transform

Several fast algorithms exist for this transformation given by eq. (1). They are too specialized to be cited here. The reader may find details in [1] or [3].

2) The Hadamard transform

The Hadamard transform is perhaps the simplest transformation. Its transform matrix contains only +1's and -1's as entries. It is given by:

$$X(\beta) = (1/\sqrt{2^n}) \sum_{\alpha=0}^{N-1} (-1)^{\sum_{k=0}^{n-1} \alpha(k)\beta(k)} x(\alpha) \qquad (32)$$

with $\alpha \leftrightarrow \alpha(n\text{-}1) \ldots \alpha(1)\alpha(0)$

$\beta \leftrightarrow \beta(n\text{-}1) \ldots \beta(1)\beta(0)$

and $\alpha(k)$, $\beta(k) = 0$ or 1.

The coefficients $X(\beta)$ given by this equation are in the so-called natural order, as naturally given by the definition. They can be ordered in various ways such as bit-reversed order, sequency order or cal-sal order. Fast algorithms exist for each type ordering. A particular order is selected depending on the problem. For example, if even or odd symmetries are searched in the signal, cal-sal order is the appropriate ordering. If a parallel is made to the frequency as in Fourier transform, then sequency order should be used.

3) The R transform

This transformation is almost identical to Hadamard transform except that absolute values are taken after each substraction in the Hadamard transform flow chart. Because of this nonlinearity, this transform has no inverse. An interesting property of this transform is that the transform coefficients remain unchanged under cyclic translation of the input samples.

4) The Haar transform

A closed form analytical expression for this transform is cumbersome to write. The general form (18) remains valid here, where each a(k,n) are replaced by samples from the orthogonal set of Haar functions shown in Fig. 7.

5) The sine transform

The closed form analytical expression of the sine transform is the following:

$$S(n) = \sum_{k=0}^{N-1} x(k) \sin\left[\frac{(k+1)(n+1)}{N+1}\pi\right] \tag{33}$$

A similar form can be used depending on the even or odd numbers involved. This transform, along with the cosine transform as defined below, are used often for speech coding.

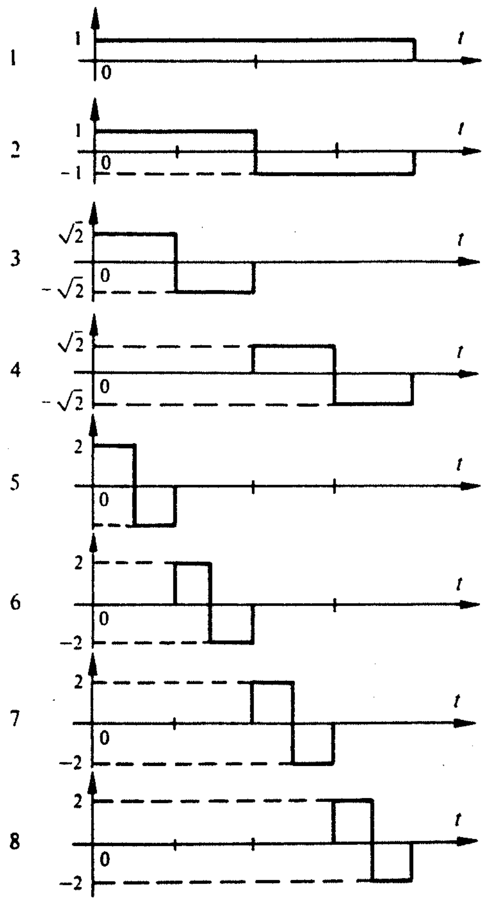


Fig. 7 Orthonormal set of Haar functions.

6) The cosine transform

The cosine transform is defined by:

$$C(n) = \sum_{k=0}^{N-1} x(k) \cos[\frac{\pi}{2N} (2k+1)n] \tag{34}$$

Note that sine and cosine transforms are byproducts of the discrete Fourier transform. Because they require real coefficients, instead of complex exponentials used in Fourier series, they are faster to compute and hence more suitable for real-time applications.

7) Slant transform.

As the Haar transform, the closed form analytical exprression for this transform is cumbersome to write. In this case a(n,k) are replaced by samples from the orthogonal set of slanted functions shown in Fig. 8.

2.6.5 THE KARHUNEN LOEVE TRANSFORM

Although there is no fast algorithm for this transformation, it is useful to describe it briefly because, in many cases, it is considered as the 'best' linear transform; best meaning whatever the reader thinks it means. The Karhunen Loeve transform, by definition, produces statistically uncorrelated coefficients, i.e. :

$$E[X(n)X*(m)] = \lambda_{\eta}\delta(n,m) \tag{35}$$

Substituting the general form (18) in (35), we have :

$$E[X(n)X*(m)] = E[\ \sum_{k} a(n,k)\ x(k)\ \sum_{1} a*(m,1)x*(1)\ ]$$

$$= \sum_{k}\sum_{1} E[x(k)x*(1)]\ a(n,k)\ a*(m,1) = \lambda[\eta]\delta(n,m) \tag{36}$$

The expectation $E[x(k)\ x^{*}(1)]$ is, by definition, the general entry of the correlation matrix $\varphi_{x}(k,1)$ of the signal. Eq. (36) is then :

$$\sum_{k}\sum_{1} \varphi_{x}\ (k,1)\ a(n,k)\ a^{*}(m,1) = \lambda_{\eta}\delta(n,m) \tag{37}$$

Comparing this result to eq. (19), we have :

$$\sum_{k} \varphi_{x}\ (k,1)\ a(n,k) = \lambda_{\eta}a(n,1) \tag{38}$$

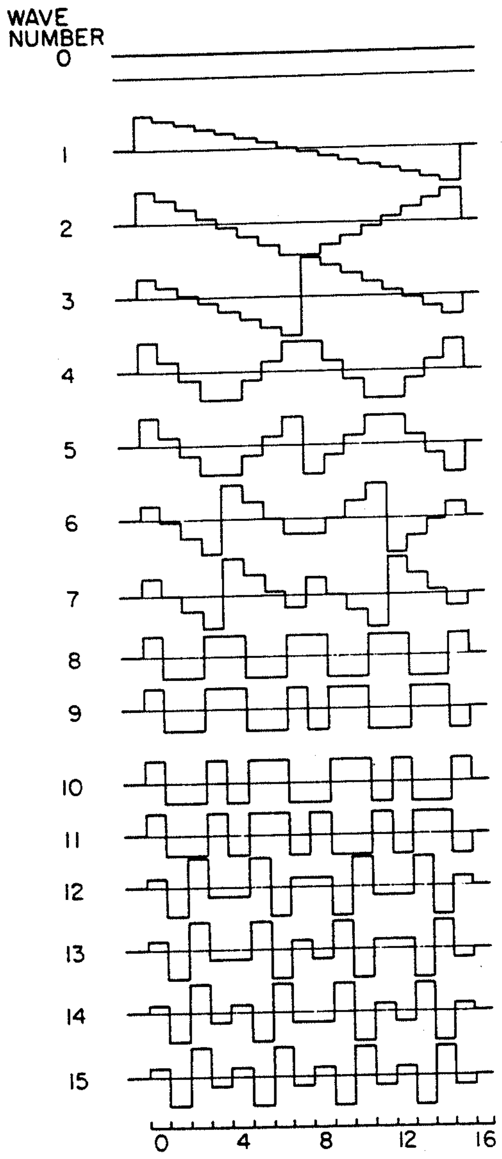


Fig. 8 Othonormal set of functions used in the slant transform.

In matrix notation, the equivalent form is :

$$\varphi_x \; A = \lambda_\eta A \tag{39}$$

which is the classical eigen-vector, eigen-value problem. The solution of eq. (39) gives the transform matrix which guarantees (35). To derive this transformation, the correlation matrix $\varphi_x(k,1)$ of the signal must be estimated first. Then a linear tranform requiring $Nc = N^2$ operations should be computed. Because of this computational load, the Karhunen Loeve transform is not very often used in practice. It gives an indication about the upper bound of what other transform cited earlier, computationally more efficient, should attempt to reach for decorrelating data samples.

2.7 LINEAR PREDICTION

2.7.1 PRELIMINARY REMARKS

There are several ways to introduce the linear prediction. The one which will be discussed here is based on the convolution, the z transform and linear systems described previously. Let us consider a linear system and its associated equations :

$$y(k) = x(k)*g(k) = \sum_{-\infty}^{+\infty} g(1)\; x(k-1) \tag{40}$$

and

$$Y(z) = X(z)\; G(z) \tag{41}$$

along with the convergence region.

The transfer function G(z) is then the ratio Y(z)/X(z). The roots of Y(z) is also the roots of G(z). In contrast, the roots of X(z) are the poles of G(z). The desired transfer function is often so idealized that various approximation forms are the followings :

1) Polynomial approximation.

In this case, the practical transfer function is given by :

$$G(z) = \sum_{i=0}^{M} b(i)\ z^{-i} \tag{42}$$

Substituting this particular form in (41) and taking the inverse z transform of both side of this equation, we have :

$$y(k) = \sum_{i=0}^{M} b(i)\ x(k-i) \tag{43}$$

which is nothing else than a particular form of the convolution (12) for FIR filters or systems. Note that in this case, each output sample is a weighted sum of a finite number of past and present input samples. The system is entirely described by m coefficients b(i). This type of system is also called all zero system since the transfer function has M zeros.

2) Inverse polynomial approximation

In this case, the practical transfer function is given by :

$$G(z) = \frac{1}{\sum_{j=0}^{N} a(j)\ z^{-j}} \tag{44}$$

It is possible to introduce a gain factor GO in (5) to replace the unity numerator. Substituting this particular form in (41) and taking the inverse z transform of both sides of this equation, we have :

$$y(k) = - \sum_{j=1}^{N} a(j)\ y(k-j) + x(k) \tag{45}$$

assuming $a(0) = 1$.

which is a particular form of the difference equation (16) for IIR filters or systems. Note that in this case, each output sample is a weighted sum of a finite number of past output samples and the present input sample. The system is entirely described by N coefficients a(j). These systems are also called all pole systems since the transfer function has N poles.

3) Ratio of polynomials

In the most general case, the practical transfer function is a ratio of two polynomials in z^{-1}. This is the case discussed in section 2.5 leading to the difference equation :

$$y(k) = - \sum_{j=1}^{N} a(j)\ y(k-j) + \sum_{i=0}^{M} b(i)\ x(k-i) \tag{46}$$

assuming again $a(0) = 1$

In this most general case, the output sample is a weighted sum of the past output samples plus a weighted sum of the present and past input samples. It is a mixed, pole-zero system.

In each of the preceeding approximations, the present output sample is expressed as a linear combination of past (and one present) samples. If the signal under investigation is associated to y(k), or equivalently if it can be viewed as the output of a known linear system (coefficients a(j)'s and b(i)'s known) whose input is a known signal x(k), eqs. (5), (6) or (7) can be used to predict the present sample of the output signal. Iterating these equations, the entire signal y(k) can be predicted. Since this prediction is done as a linear combination of other samples, it is called linear prediction.

There are several possible framework to study in detail linear prediction such as covariance method, autocorrelation method, lattice structures, inverse filtering, spectral estimation, maximum likelihood method, dot product method, etc.

Mathematical developpments for each of them are quite simple. But each step of the computation offer a pletore of possible interpretations. This results in a very rich, precise, reliable and robust technique.

2.7.2 LINEAR PREDICTION WITH ALL POLE SYSTEMS

Linear prediction with all pole systems has the most widespread use because of the following general properties.

Let y(k) denote the signal under investigation. If N past samples of this signal are used to predict its present value, we have :

$$\hat{y}(k) = \sum_{i=1}^{N} a(i)\ y(k-i) \tag{47}$$

It is hoped that $\hat{y}(k) \approx y(k)$. The predicted signal $\hat{y}(k)$ is thus produced with a linear system excited by y(k) and whose transfer function is P(z) given by :

$$P(z) = \sum_{i=1}^{N} a(i)\ z^{-i} \tag{48}$$

This result is obtained by just taking the z transform of both sides of eq. (47) and solving for the transfer function.

The prediction error e(k) is simply the difference between y(k) and $\hat{y}(k)$:

$$e(k) = y(k) - \hat{y}(k) = y(k) - \sum_{i=1}^{N} a(i)\ y(k-i) \tag{49}$$

Again, by taking the z transform of both sides of this equation, we obtain the transfer function Q(z) of the linear system that produces the prediction error signal when excited with the signal y(k). We have :

$$Q(z) = 1 - \sum_{i=1}^{N} a(i)\ z^{-i} = 1 - P(z) \tag{50}$$

Let us assume now the hypothesis that the signal y(k) is produced according to a linear prediction model given by :

$$y(k) = \sum_{i=1}^{N} \alpha(i)\, y(k-i) + GO\ x(k) \tag{51}$$

where the coefficients $\alpha(i)$'s are not known. The transfer function H(z) of this production model is obtained by taking the z transform of both sides of eq. (51) :

$$H(z) = \frac{GO}{1 - \sum_{i=1}^{N} \alpha(i)\, z^{-i}} \tag{52}$$

If the signal y(k) is produced with eq. (51) and if a(i) = α(i) with i = 1,...,N, then Q(z) is the inverse filter of H(z) :

$$H(z) = \frac{GO}{Q(z)} \tag{53}$$

In other words, systems characterized by the transfer functions H(z) and Q(z) have opposite effects. The first one produces the signal y(k) from a signal GO x(k), whereas the second produces the prediction error e(k) = GO x(k) from the signal y(k). This discussion is illustrated in Fig. 9. In this context, the prediction problem consists in finding the set coefficients a(i) from y(k) in order to represent this signal in the best possible way — according to a criterion — by eq. (52). Since in this equation Q(z) is a denominator polynomial, we have an all pole system for H(z).

2.7.3 COMPUTING THE PREDICTION COEFFICIENTS

The prediction coefficients are obtained by minimizing the energy of the prediction error e(k). This energy is given by :

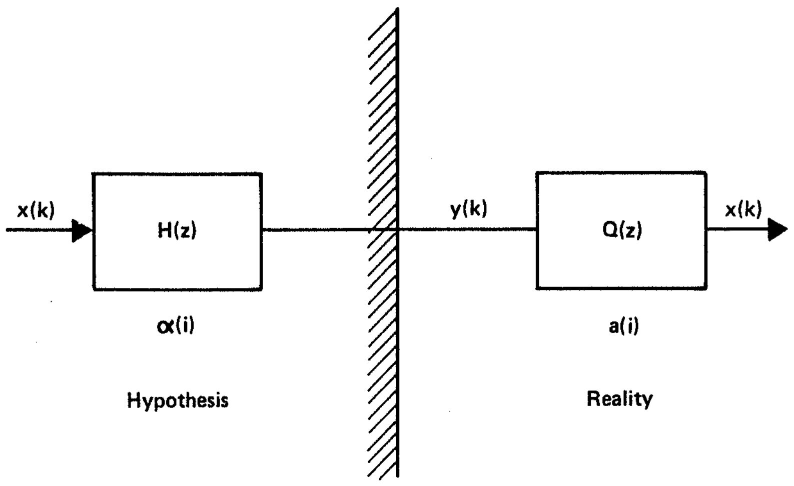


Fig. 9 Illustrating the prediction model.

$$W = \sum_{k=k_1}^{k_2} e^2(k) \tag{54}$$

Eq. (49) can be rewritten as :

$$e(k) = - \sum_{i=0}^{N} a(i)\ y(k-i) \tag{55}$$

with a(0) = -1.

Substituting (55) in (54), we have :

$$W = \sum_k \sum_i \sum_j a(i)\ y(k-i)\ y(k-j)\ a(j) \tag{57}$$

Defining the quadratic form

$$C(i,j) = \sum_{k=k_1}^{k_2} y(k-i)\ y(k-j) \tag{58}$$

we have

$$W = \sum_{i} \sum_{j} a(i)\ C(i,j)\ a(j) \tag{59}$$

Setting to zero all the derivaties of W with respect to a(i)'s :

$$C(0,j) = \sum_{i=1}^{N} a(j)\ C(i,j) \tag{60}$$

Particular cases are defined by constraining k_1, k_2 and y(k). For example, if C(i,j) is computed over an infinite interval ($k_1 = -\infty$ and $k_2 = +\infty$) and the signal y(k) is observed over a finite interal from $k = 0$ to $k = N\text{-}1$ (implicit rectangular window), the quadratic form C(i,j) becomes :

$$\begin{aligned} C(i,j) &= \sum_{-\infty}^{+\infty} y(k\text{-}i)\ y(k\text{-}j) = \sum_{-\infty}^{+\infty} y(k)\ y(k+|i\text{-}j|) \\ &= \sum_{-\infty}^{+\infty} y(k)\ y(k+|i\text{-}j|) = \varphi_y(|i\text{-}j|) \end{aligned} \tag{61}$$

In this case C(i,j) is the autocorrelation function and the method used to solve (60) for a(i)'s is called autocorrelation method.

If $k_1 = M$ where M is the order of the predictor to start the prediction with M initial conditions and $k_2 = N\text{-}1$, C(i,j) behaves like a covariance matrix. The method using this case to solve (60) is called covariance method.

Different polynomials Q(z) are obtained with different methods and analysis conditions. The autocorrelation method guarantees in theory the stability of H(z) given by (52) but requires a window on the signal which reduces the frequency resolution. On the other hand, the covariance method does not require any window, but does not guarantee the stability. The lattice method overcomes these problems, provides a quite general solution and guarantees the stability.

2.7.4 LATTICE METHOD [2]

The lattice method is obtained by computing two predictions on the signal, one for the future (which will be denoted by +) and one for the past (which will be denoted by -). Fig. 10 shows a signal x(k) and its samples to illustrate these predictions. Starting with the sample x(k-m) the future prediction is

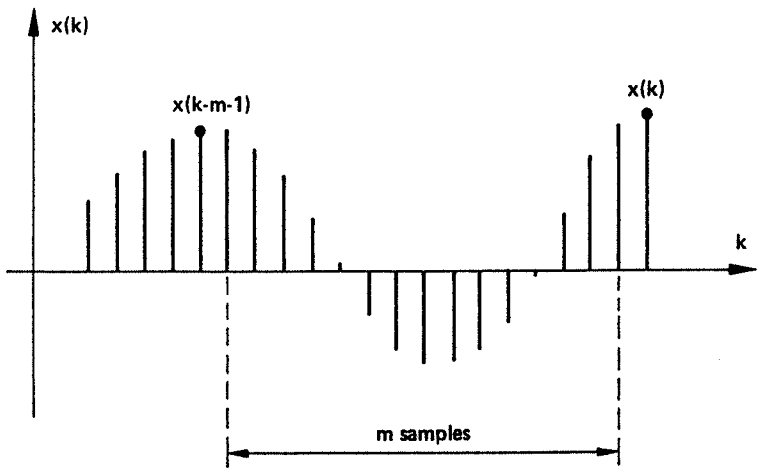


Fig. 10 Illustrating past and future predictions.

computed according to :

$$\hat{x}_{m+}(k) = - \sum_{i=1}^{m} a(m,i)\ x(k-i) \tag{62}$$

where m is a parameter taking the values 1, 2, ..., M. The prediction error is :

$$x_m^+(k) = x(k) - \hat{x}_{m+}(k) = \sum_{i=0}^{m} a(m,i)\ x(k-i) \tag{63}$$

with a(m,0) = 1. The past prediction is computed similarly :

$$\hat{x}_{m-}(k) = - \sum_{i=1}^{m} b(m,i)\ x(k-i) \tag{64}$$

leading to the past prediction error :

$$x^-_m = x(k\text{-}m\text{-}1) - \hat{x}_{m-}(k) = \sum_{i=0}^{m} b(m,i)\ x(k\text{-}i) \tag{65}$$

with $b(m,m+1) = 1$.

Note that in eq. (65) the error is computed at $x(k - m - 1)$ to have causal systems. The criterion used is the simultaneous minimization of the past and future prediction error energies, computed over the interval [k0,k1] :

$$W_+(m) = \sum_{k=k_0}^{k_1} |x^+_m(k)|^2 \quad \text{and} \quad W_-(m) = \sum_{k=k_0}^{k_1} |x^-_m(k)|^2 \tag{66}$$

with $m = 1, \ldots, M$

The transfer functions of the future and past predictors (62) and (64) are given respectively by :

$$A_m(z) = \sum_{i=0}^{m} a(m,i)\ z^{-i} \quad \text{and} \quad B_m(z) = \sum_{i=0}^{m} b(m,i)\ z^{-i} \tag{67}$$

It can be shown [2] that the simultaneous minimization of prediction error energies given by (66) leads the orthogonality of the polynomials $A_m(z)$ and $B_m(z)$ to z^{-j} :

$$\{A_m(z),\ z^{-j}\} = 0 \quad \text{and} \quad \{B_m(z),\ z^{-j}\} = 0 \tag{68}$$

where {.,.} represents the dot product.

The lattice structure is obtained by establishing iteratively the form of the polynomials $A_m(z)$ and $B_m(z)$ for $m = 1,\ldots,M$ using two initial conditions $a(0,i) = b(0,i) = 1$. Iterations are governed by :

$$A_m(z) = A_{m-1}(z) + k_m B_{m-1}(z) \tag{69}$$

and by :

$$B_m(z) = z^{-1}[B_{m-1}(z) + k_m A_{m-1}(z)] \tag{70}$$

where k_m is given by :

$$k_m = (-1/W_{m-1}) \sum_{k=k_0}^{k_1} x^+_{m-1}(k) x^-_{m-1}(k) \tag{71}$$

and can be interpreted as partial correlation coefficient.

Taking the z transform of both sides of (63) and (65), substituting in this result (69) and (70) and taking the inverse z transform, we obtain the equations of the lattice structure :

$$x^+_m(k) = x^+_{m-1}(k) + k_m x^-_{m-1}(k) \text{ and } x^-_m(k) = x^-_{m-1}(k) + k_m x^+_{m-1}(k) \tag{72}$$

Fig. 11 shows the lattice structure of a filter equivalent to the filter characterized by the transfer function Q(z) given by (50) and producing the prediction error e(k). Because of the modularity contained in it, this structure is very suitable for hardware implementations.

2.8 HOMOMORPHIC PROCESSING OF SIGNALS

Homomorphic processing is a convenient tool to be used for processing signals combined by convolution or multiplication (modulation). Let us assume that the observed signal is given by $z(k) = x(k) \cdot y(k)$ or $z(k) = x(k) * y(k)$ where x(k) is the signal to be recovered. Clearly, in these cases linear filtering theory cannot be used, independently from its well known and powerful techniques. Two approaches can be envisioned. The first one consists in developing special methods to solve these problems, whereas the second attempts to transform these problems into other problems already solved, for example by using linear system theory. The second approach seems to be easier to develop and to apply, because it benefits from the large amount of available results in linear system

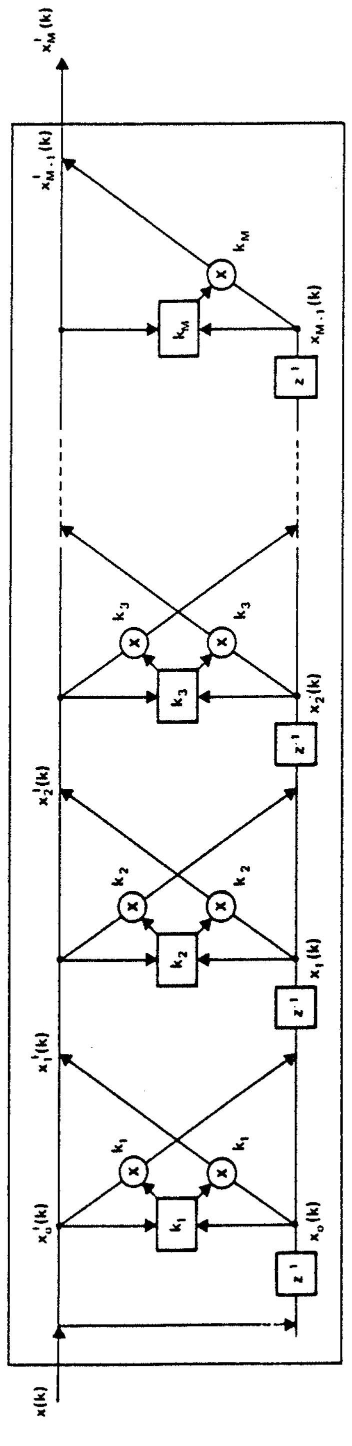


Fig. 11 Lattice structure for linear prediction

theory.

2.8.1 MULTIPLIED SIGNALS

If the observed signal is of the form :

$$x(k) = [x_1^{a_1}(k)]\ [x_{x[2]}^{a_2}(k)] \tag{73}$$

where a1 and a2 are two arbitrary constants, the well known way to transform this signal into a weighted sum is the use of logarithm. The real logarithm may be too restrictive because it can be used only for strictly positive signals or for signals which have been previously rescaled with the addition of a constant value to render them positive. The complex logarithm, however, can be used for bipolar or complex signals. In polar coordinates, a complex signal x(k) can be written as :

$$\begin{aligned} x(k) &= |x(k)| \exp\ [j \arg(x(k)] \\ &= \exp[\ln|x(k)| + j \arg x(k)] \end{aligned} \tag{74}$$

The complex logarithm, denoted by ln[.], is then :

$$\ln[x(k)] = \ln|x(k)| + j \arg x(k) \tag{75}$$

As defined, the complex logarithm has an important disadvantage. It is not a one-to-one transformation. Its argument is defined modul 2π, i.e. adding multiples of 2π to the argument in (75) does not change the result. In other words, the phase is wrapped around the unit circle. To remove this ambiguity from the argument of the complex logarithm, all the arguments to be used need to be defined as continuous functions of x(k). With a continuous argument, the complex logarithm becomes a one-to-one transform. If ever, the former form of the argument is required, it is sufficient to compute it modulo 2π. Usually, what is given in the first place is the argument modulo 2π. There are several algorithms to transform such an argument into a continuous function. This operation is known as phase unwrapping. Fig. 12 shows wrapped and unwrapped versions of an argument.

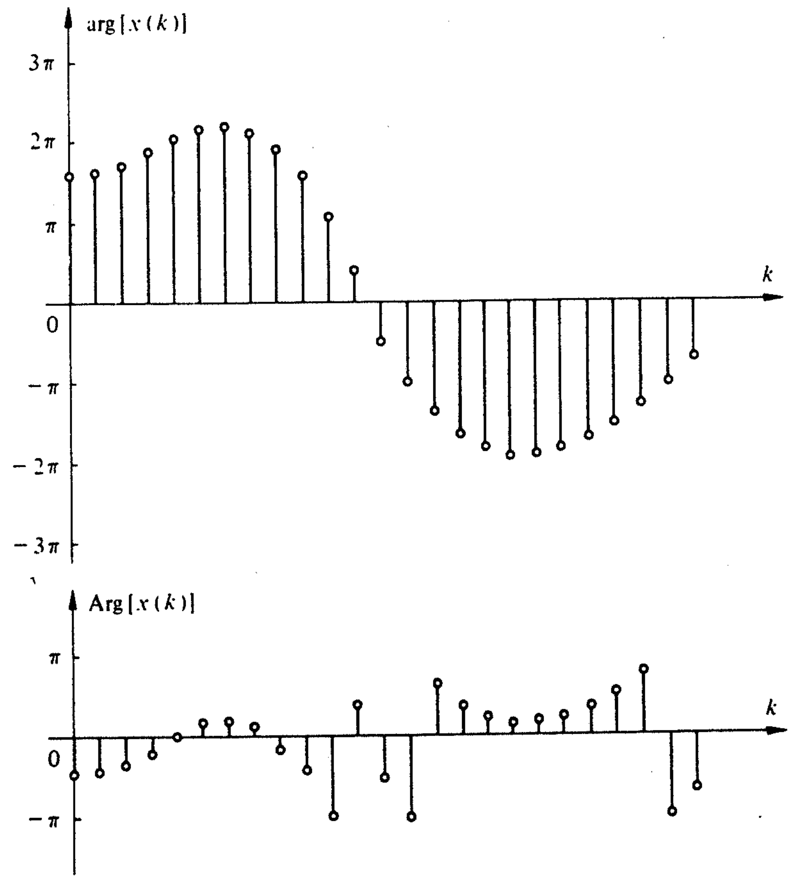


Fig. 12 Wrapped and unwrapped phase curve.

If the complex logarithm is now applied to the input signal x(k) given by eq. (73), the result is :

$$\hat{x}(k) = \ln[x(k)] = a1 \ln[x_1(k)] + a2 \ln[x_2(k)]$$

$$= a1 \hat{x}_1(k) + a2 \hat{x}_2(k) \qquad (76)$$

Notice that, now, the input signal x(k) is transformed into a signal $\hat{x}$ (k) which is a linear combination of other signals. Hence, according to the frequency content of $\hat{x}_1(k)$ and $\hat{x}_2(k)$, linear filtering may be used to recover for example $\hat{x}_1(k)$ or $\hat{x}_2(k)$ to the detriment of the other. Once the transformed version of the signal we are looking for is obtained, the signal itself is simply given by the inverse transform, i.e. :

$$x(k) = \exp[\hat{x}(k)] \tag{77}$$

The results of this section show that with a nonlinear one-to-one transformation, it is possible to transpose a multiplicative combination of signals into an additive combination, a familiar situation in which linear system theory can be used. Fig. 13 shows the block diagram of a homomorphic system for multiplied signals.

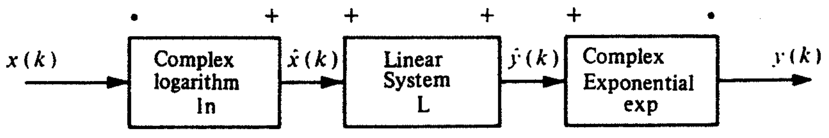


Fig. 13 Block diagram for homomorphic processing of multiplied signals.

As an example related to speech signals, let us consider the dynamic range handling. Whenever a speech signal changes its physical support, dynamic ranges on both supports should be measured and compared. Very often they are not compatible. It is therefore necessary to compress or expend the dynamic range before changing the recording media. The speech signal can be viewed as a low frequency envelope e(k) modulated with a high frequency carrier p(k). Hence, the input signal is :

$$x(k) = e(k) \cdot p(k) \tag{78}$$

Since e(k) is strictly positive, the complex logarithm leads to :

$$\hat{x}(k) = \ln e(k) + \ln|p(k)| + j \arg[p(k)] \tag{79}$$

A linear filter can be used to compress or to expand the dynamic range of e(k) as given by :

$$\hat{y}(k) = a \ln e(k) + b \ln|p(k)| + jb \arg[p(k)] \tag{80}$$

In practice, b = 1 because the carrier does not influence too much the dynamic range of e(k). If a is larger than 1, the dynamic range is expended. In contrast, if a is less than 1, the dynamic range is compressed.

2.8.2 CONVOLVED SIGNALS

In this case, the observed signal x(k) is the convolution of two signals x (k) and x (k), the former being the signal of interest.

$$x(k) = \sum_{l=-\infty}^{+\infty} x_1(l)\, x_2(k-l) = x_1(k) * x_2(k) \tag{81}$$

The problem is now to recover $x_1(k)$ from x(k). Remembering eq. (14), if the discrete Fourier transform is applied to both sides of eq. (81), the result is a simple product :

$$X(n) = X_1(n) \cdot X_2(n) \tag{82}$$

Notice that the use of the discrete Fourier transform for convolved signals, transform this problem into a problem that has just been solved, ie. the problem of multiplied signals. Since the discrete Fourier transform of a signal can be viewed as a complex signal, all that has to be done is to take the complex logarithm of both sides of eq. (82). We have :

$$\ln [X(n)] = \ln [X_1(n)] + \ln [X_2(n)] \tag{83}$$

Now, the convolved input signal is transformed into a linearly combined signal for which the linear system theory may again be applied. Notice also that, in constrast with eq. (76), in this case signals are function of the frequency and not of the time. If a linear filter is to be used for eq. (83), it should be a filter whose impulse response is expressed in the frequency domain and its frequency response in the time domain. This is rather an uncomfortable situation in which it is not difficult to mix up time and frequency. To overcome this difficulty, the inverse Fourier transform of both sides of eq. (83) can be taken, leading to :

$$\begin{aligned}\hat{x}(k) &= \mathcal{F}^{-1} \ln [X(n)] \\ &= \hat{x}_1(k) + \hat{x}_2(k) \qquad (84)\end{aligned}$$

which is now a time domain equation for linearly combined signals. Linear filtering can be applied to this equation to recover $\hat{x}_1(k)$ or $\hat{x}_2(k)$. Then, to transform everything back to the original representation, a discrete Fourier transform should be computed first. Its result is then exponentiated and inverse Fourier transformed. Fig. 14 shows the block diagram of a homomorphic system for convolved signal. Notice that solving eq. (81) for $x_1(k)$ is the inverse of convolution. This operation is commonly referred to as deconvolution. Homomorphic processing is one of the possible ways to deconvolve a signal. The inverse Fourier transform of the complex logarithm of the Fourier transform of a signal is called cepstrum; a word obtained by scrambling the letters of the word spectrum. It finds a rather large number of applications in speech processing such as echo removal, pitch detection, speech parameters estimation and restoration.

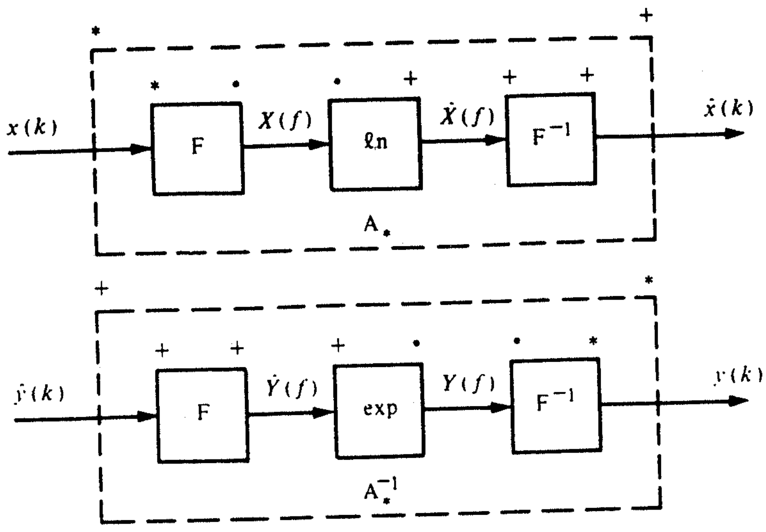


Fig. 14 Block diagram for homomorphic processing of convolved signals.

2.8.3 MAJOR APPLICATIONS OF THE CEPSTRUM

In this section three major applications of the cepstrum are given.

As a first case, let us consider a recorded signal x(k) resulting from several echos of the same initial signal $x_1(k)$:

$$x(k) = x_1(k) + \sum_{i=1}^{M} \alpha(i)\ x_1(k\text{-}ki) \tag{85}$$

where α(i)'s are different attenuation factors and ki's the corresponding delays. Eq. (85) can be viewed as the convolution of the signal $x_1(k)$ given by :

$$x_2(k) = d(k) + \sum_{i=1}^{M} \alpha(i)\ d(k\text{-}ki) \tag{86}$$

where d(k) is the unit sample. For simplicity, let us consider one echo only, i.e. M = 1. The discrete Fourier transform of (86) is :

$$X_2(n) = 1 + \alpha(1)\exp[-j2\pi k_1 n/N] \tag{87}$$

Taking the complex logarithm, we have :

$$X_2(n) = \ln[1 + \alpha(1)\exp(-j2\pi k_1 n/N)] \tag{88}$$

This function is periodical of period $1/k_1$. Accordingly, the cepstrum $\hat{x}_2(k)$ will be nonzero only for integer multiples of k_1. In the cepstrum of the recorded signal, the non zero values of $\hat{x}_2(k)$ will appear as peaks. To surpress the echo, these peaks should be eliminated. This can be done with a comb filter applied to the cepstrum $\hat{x}(k)$ which recovers only the contribution of $\hat{x}_1(k)$. The inverse transform gives the original signal without any echo. Fig. 15 shows various functions involved in filtering the cepstrum for echo removal.

The second application of the cepstrum is its use in pitch detection and speech parametres estimation. A voiced portion of a speech signal can be modeled as shown in Fig. 16. The excitation signal e(k) containing a pulse train is filtered with a time-varying filter whose impulse response is h(k). A prefilter is also introduced to better represent vocal cords. Its impulse response is denoted by g(k). Accordingly, the produced speech s(k) is given by :

$$s(k) = e(k) * g(k) * h(k) \tag{89}$$

over a short period of time (10-30 ms). To avoid discontinuities at the ends of each speech segment, a window function w(k) is introduced, producing the windowed speech x(k) :

$$\begin{aligned} x(k) &= s(k) \cdot w(k) \\ &= [e(k)*g(k)*h(k)]w(k) \end{aligned} \tag{90}$$

Assuming that the window w(k) is smooth enough over the variation of g(k) * h(k), we have :

$$x(k) = e_w(k)*g(k)*h(k)$$

with

$$e_w(k) = e(k) \cdot w(k) \tag{91}$$

If the pitch period is k0, then :

$$e_w(k) = \sum_{m=0}^{M-1} w(mk0)\ d(k-mk0) \tag{92}$$

where M is the number of pulses seen through the window. The discrete Fourier transform of eq. (92) is given by :

$$E_w(n) = \sum_{m=0}^{M-1} w(mk0)\ \exp(-j2\pi nmk0/N) \tag{93}$$

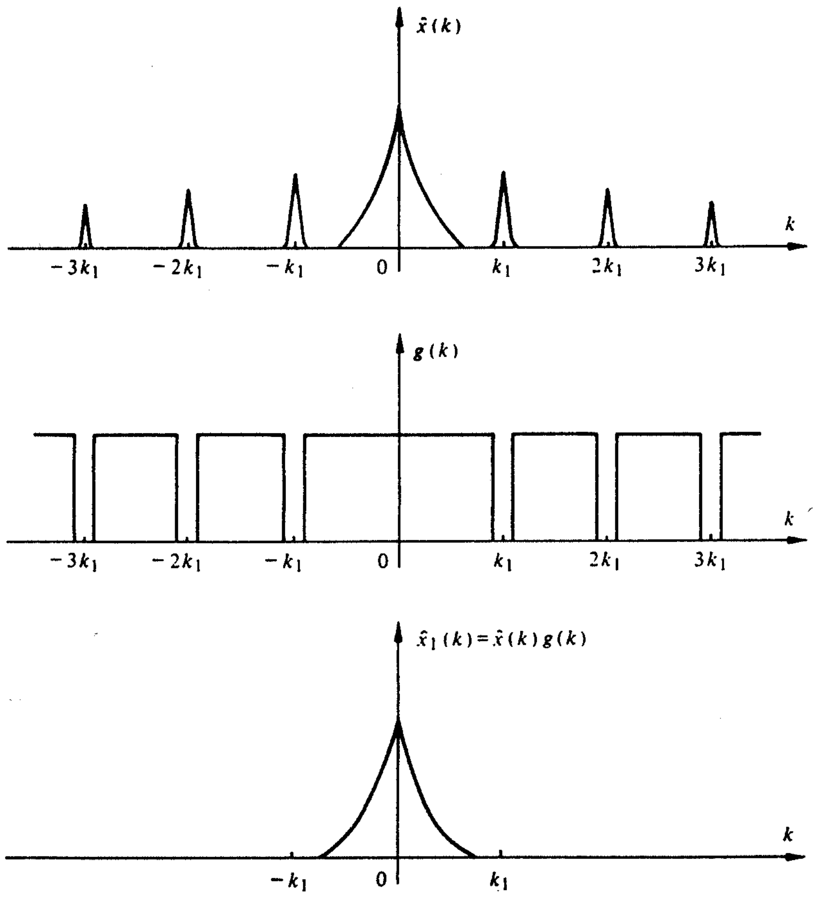


Fig. 15 Echo removal by deconvolution

This function is periodical with a period of 1/k0. The Fourier transform of the cepstrum ê(k) will also be periodical with the same period, since the logarithm is a memoryless monotonic function. Accordingly, the cepstrum ê(k) will be non zero only at integer multiples of k0. On the other hand, the duration of the composite impulse response g(k) * h(k) does not exceed a few dozens of milli seconds. In the cepstral domain, the contribution of this composite impulse response is non negligible only over a short time interval around the origin. A high-pass time filter applied to the cepstrum will isolate e(k), whereas a low-pass time filter recovers the composite impulse response g(k) * h(k). Fig. 17 illustrates, on a voiced speech section, various steps of this procedure.

The last application which will be described here concerns the restoration of old recordings. If s(k) denotes a speech signal or a song produced by an artist, its recorded version can be modeled as a convolution :

$$x(k) = s(k)*g(k) \tag{94}$$

where g(k) is the impulse response of the recording system. The only assumption of this method is that the length of the impulse response g(k) is much shorter than that of the signal s(k). If the recording is done with old equipment and if there is no way to repeat the recording with modern equipment, it is necessary to recover s(k) to avoid the effects of the recording system contained in g(k). In general s(k) is a nonstationary signal which excludes the use of classical filtering techniques. Homomorphic deconvolution can be applied in this case also. Taking the complex logarithm of the Fourier transform of both sides of (94) leads to :

$$\ln[X(n)] = \ln[S(n)] + \ln[G(n)] \tag{95}$$

The difficulty of this problem is that not only s(k) is not known but also g(k), and hence G(n). A possible way to estimate G(n) is the use of eq. (95) for several, uncorrelated recordings, so that after averaging the right side of eq. (95) will converge to G(n). The problem is that it is difficult, if not impossible, to find a large number of old recordings recorded with the same equipment under

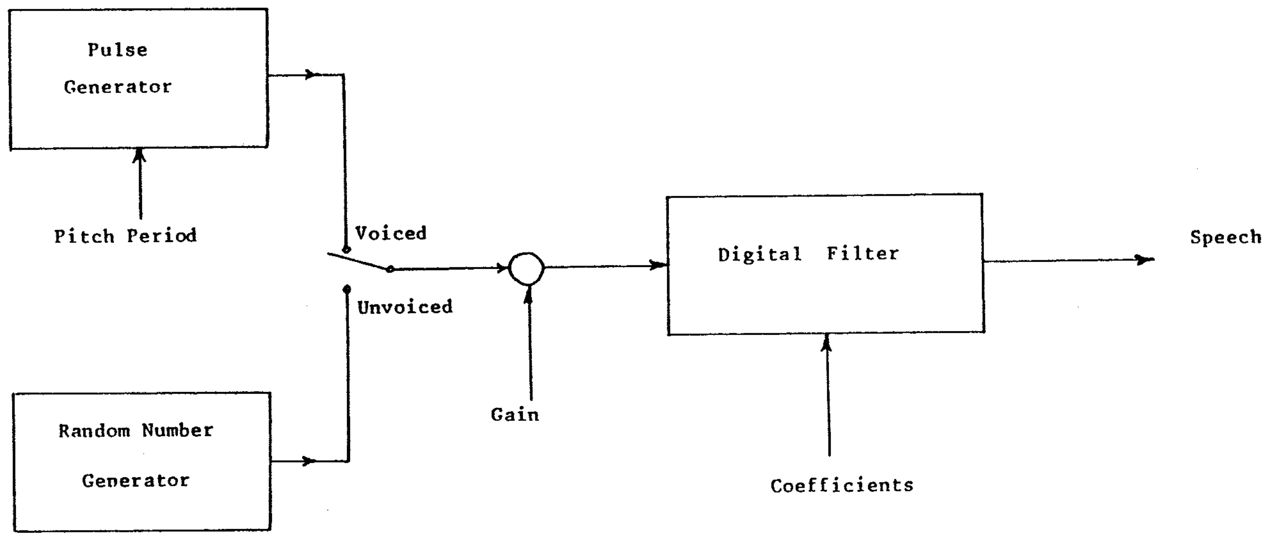


Fig. 16 Voiced speech production model

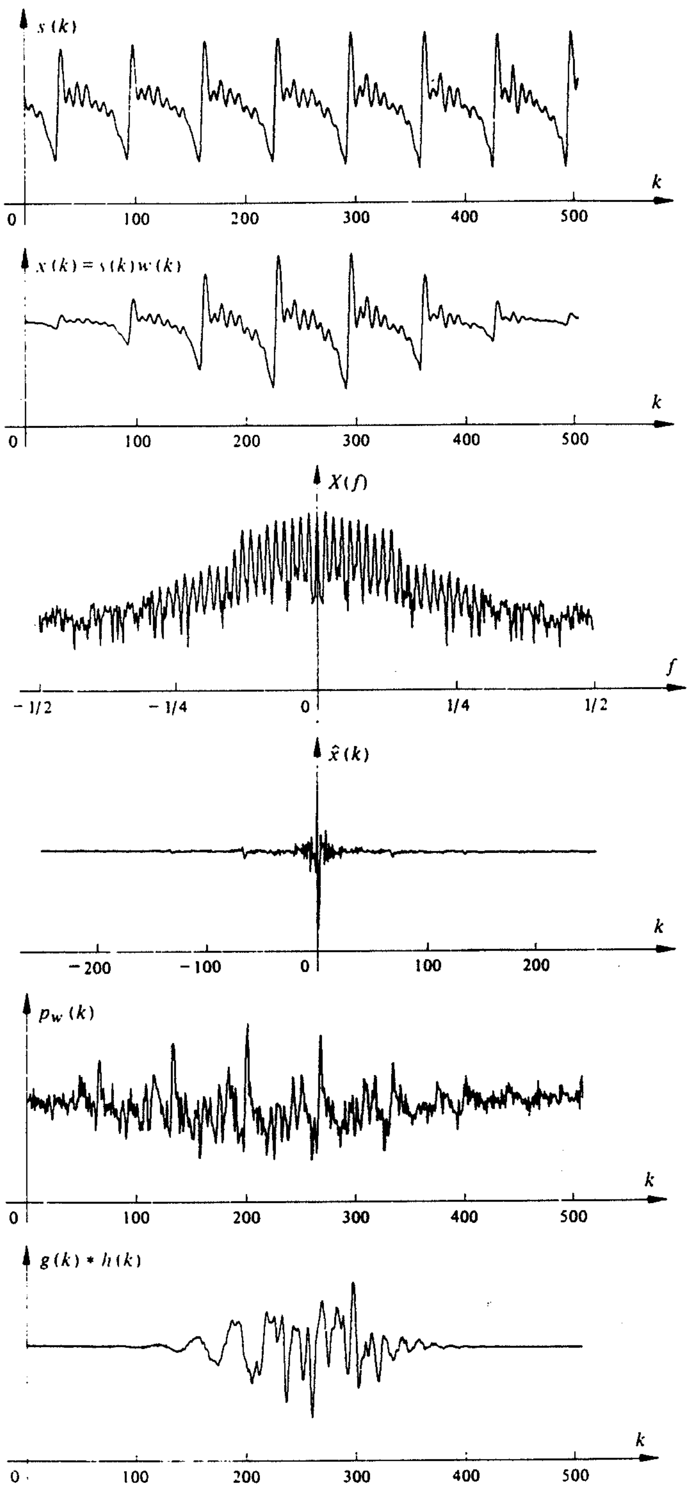


Fig. 17 Pitch period and parameter estimation

comparable conditions. This problem can be overcome if the recording x(k) is divided into several segments $x_i(k)$ of moderate length, a few seconds for example, longer than that of the impulse response g(k). Since each of these sections contains different s(k)'s, statistical filtering (averaging) may be applied. A given section $x_i(k)$ can be viewed as a windowed part of the signal x(k), i.e. :

$$x_i(k) = w_i(k)\ x(k) = w_i(k)[s(k)*g(k)] \tag{96}$$

Assuming that the window is smooth enough over g(k), we have :

$$\begin{aligned} x_i(k) &= [w_i(k)s(k)] * g(k) \\ &= s_i(k)*g(k) \end{aligned} \tag{97}$$

The complex logarithm of the Fourier transform of both sides of eq. (97) leads to :

$$\ln[X_i(n)] = \ln[S_i(n)] + \ln[G(n)] \tag{98}$$

If M denotes the number of section $x_i(k)$ in x(k), the average of both sides of eq. (98) gives :

$$(1/M) \sum_{i=1}^{M} \ln[X_i(n)] = (1/M) \sum_{i=1}^{M} [\ln[S_i(n)] + \ln[G(n)]] \tag{99}$$

The first term in the right hand side of eq. (99) is the estimate of the logarithmic power spectrum of s(k). It can be estimated by using a contemporary recording of the same song or speech recorded with modern equipment. For these systems G(n) is flat over a wide frequency range and equal to 1. So, if eq. (99) is used for a new recording, ln[G(n)] is zero and hence :

$$\frac{1}{M} \sum_{i=1}^{M} \ln[s_i(n)] = \ln[s(n)] \tag{100}$$

Substituting this in eq. (99) and solving with respect to G(n), we have :

$$\ln[G(n)] = (1/M) \sum_{i=1}^{M} \ln[X_i(n)] - \ln[S(n)] \tag{101}$$

Since the logarithm used is complex, eq. (101) should be written for the magnitude and for the phase. Neglecting the phase as a first approximation, we have :

$$\ln|G(n)| = (1/M) \sum_{i=1}^{M} \ln|X_i(n)| - \ln|S(n)| \tag{102}$$

The deconvolution is then obtained with a filter whose frequency response is the inverse of G(n), i.e. :

$$|G_r(n)| = 1/|G(n)| \tag{103}$$

The phase of $G_r(n)$ can either be obtained with the Hilbert transform or be neglected. Stokham used this method to restore old recodings from E. Caruso done in 1907 and obtained spectacularly good results [4].

To conclude this paragraph, notice that all the applications of the cepstrum are related to the deconvolution problem. The same ideas are also used for two dimensional signals, i.e. digital images.

3. SPEECH ANALYSIS-SYNTHESIS

In this section, we present principles and models of speech analysis and synthesis and show the use of signal processing methods in this context. After recalling the fundamentals of the speech signal, the principle of speech analysis-synthesis and speech short-time processing, we treat speech analysis and synthesis in two main sections. The first of them, spectral analysis, explores spectral envelope analysis and synthesis in the various speech synthesis models. The second is dedicated to pitch detection and pitch period measurement.

3.1 SPEECH SIGNAL

The speech signal is modeled after the speech production mechanism.

3.1.1 SPEECH PRODUCTION

Major elements of the vocal apparatus are: the lung, the larynx and the vocal tract:

a) The lung is the source of energy in form of air under pressure.

b) The larynx consists of three cartilages supporting the vocal cords which is an opening of variable size through which air from the lung flows. Voiced sounds are produced by adjusting the vocal cords in a way that they vibrate and modulate the air flow. Unvoiced sounds are produced by keeping the vocal cords open.

c) The vocal tract consists of: 1) the pharynx and oral tract which together from a tube, 17 to 20 cm in length, whose cross section varies with jaw, tongue and lips position; and 2) the nasal tract, 12 cm in length and of fixed cross section, which lies in parallel with the oral tract and is made active or inactive by displacing the velum.

Speech is the product of an original excitation, later modified by the vocal tract.

A first type of excitation, responsible for producing voiced sounds, is the airflow modulation produced by vibrating vocal cords. The excitation signal is periodic (pitch) and its spectrum displays harmonics whose power spectrum decreases with an average of 12 dB/octave.

A second type of excitation, responsible for producing unvoiced sounds, is the air turbulence provoked by a constriction somewhere in the vocal tract. The signal has noise characteristics.

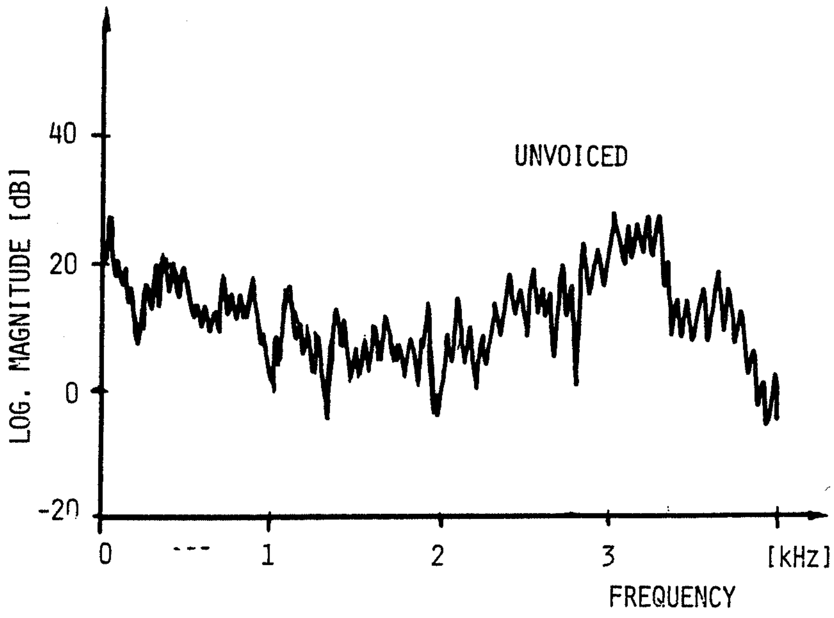
The vocal tract acts as a resonator and modifies the excitation signal. For voiced sounds, the vocal tract usually shows 4 resonances which are called formants and which characterize the sound. Fig. 10 shows a typical spectrum of a voiced sound where both the periodic excitation spectrum and the formants are visible. When the nasal tract is also active then the overall transfer function also includes zeros call antiformants.

3.1.2 SPEECH MODEL

All speech production models have in common the separation of excitation features, which are accounted for by two pulse train generators and resonator features, which are accounted for by a time-varying linear system. Thus speech production is modeled after Fig. 18.

3.1.3 SPEECH ANALYSIS AND SYNTHESIS

The basic schema of a vocoder is given in Fig. 19. There are four main functions. The analysis functions are: detection and measurement of the pitch of fundamental frequency F_o, and analysis of the spectral envelope. The two synthesis functions are: generation of the excitation and restitution of the spectral envelope. Next sections will describe this function more in detail. However, compression of the data flow between analyser and synthesiser is not described further.



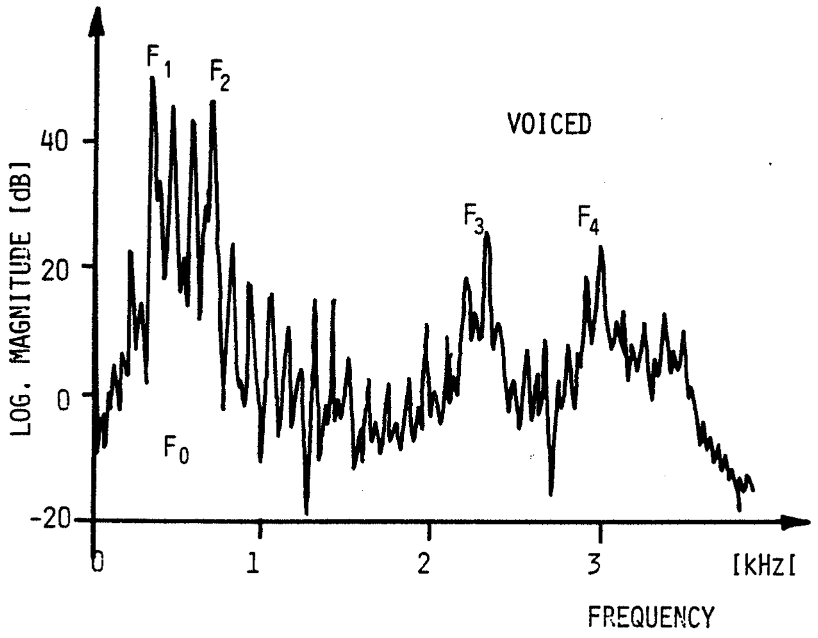


Fig. 18 Spectrum of a voiced sound

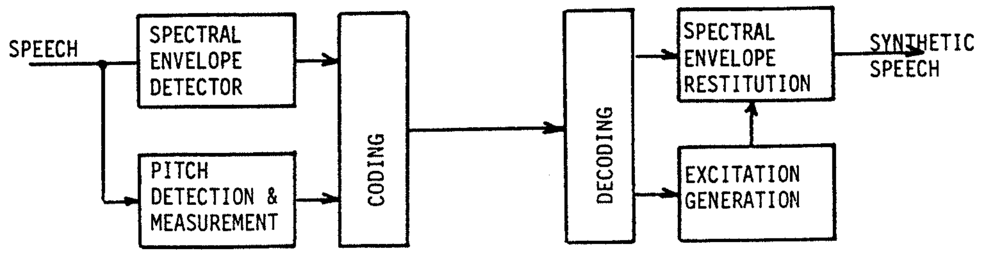

SPEECH
SPECTRAL ENVELOPE DETECTOR
PITCH DETECTION & MEASUREMENT
CODING
DECODING
SPECTRAL ENVELOPE RESTITUTION
EXCITATION GENERATION
SYNTHETIC SPEECH

Fig. 19 Main functions of speech analysis and synthesis

3.1.4 SHORT-TIME SPEECH ANALYSIS

The properties of the speech signal change relatively slowly with time. This leads to short-time processing methods in which speech segments called frames are isolated and processed as if they had fixed properties. Such segments can be isolàted by multiplication with a window sequence w(k) positioned at a location k = ℓ. Windowing the speech signal x(k) using the time-domain window w(k) leads to the analysis sequence:

$$x_\ell(k) = x(k) \cdot w(\ell-k) \tag{104}$$

Two examples of short-time parameters are given. The short-time average magnitude is defiend by :

$$M(1) = \sum_{k=\ell}^{\ell+N-1} x(k) \cdot w(\ell-k) \tag{105}$$

The short-time average zero-crossing rate is defined by :

$$Z(k) = \sum_{k=\ell}^{\ell+N-1} c(k) \cdot w(\ell-k) \qquad (106)$$

where : $c(k) = \begin{cases} 1 & \text{if } \operatorname{sign}(x(k)) \neq \operatorname{sign}(x(k-1)) \\ 0 & \text{else} \end{cases}$

3.2 SPECTRAL ANALYSIS-SYNTHESIS

We describe here the analysis and restitution of the spectral envelope. The various interpretations of short-time Fourier analysis are given and three vocoders are described.

3.2.1 SHORT-TIME FOURIER ANALYSIS

Three interpretations of the short-time Fourier transform are given. First it is defined as the normal Fourier transform of a sequence which is limited in time. Using the window sequence w(k) which is 0 outside the significant range $k = 1..N$, we find, using the definition of eq. (1), the short-time Fourier transform in its discrete form:

$$X_\ell(n) = \sum_{k=\ell}^{\ell+N-1} w(\ell-k) \cdot x(k) \cdot \exp(-j2\pi kn/N) \qquad (107)$$

with : $n = -N/2, ..., N/2-1$

where N is the maximum width of the window sequence w(k). Note that it is a function of both the discrete frequency n and the position ℓ of the window. A first interpretation of X (n) is as follows. Eq. (107) can be seen as the DFT of a sequence $w(\ell-k) \cdot x(k)$ which itself, according to eq. (104) is the x(k) sequence observed through a window with impulse response w(k) displaced at position ℓ.

The second interpretation derives directly from eq. (107) by considering it as the convolution defined in eq. (3).

$$X_\ell(n) = [x(k) \cdot \exp(-j2\pi kn/N)] * w(k) \tag{108}$$

We thus obtain $X_\ell(n)$ by cascading a modulation of x(k) by exp(-j2πkn/N) and a filter with impulse response w(k).

To catch the third meaning of $X_\ell(n)$, we transform the equation above into the equivalent form:

$$X_\ell(n) = \exp(-j2\pi kn/N) \sum_{k=\ell}^{\ell+N-1} x(\ell-k) \cdot w(k) \cdot \exp(j2\pi kn/N) \tag{109}$$

which interprets as the result of modulating (exp(-j2πkn/N)) the output of a complex bandpass filter whose impulse response is w(k) exp(j2πkn/N). The practical consequence of this interpretation is that $X_\ell(n)$ can be obtained by filtering x(k) with a bandpass filter followed by a modulator. Moreover, in vocoders we are satisfied with the amplitude of the complex value $X_\ell(n)$. Using equation above, we find :

$$|X_\ell(n)| = |\sum_{k=\ell}^{\ell+N-1} x(1-k) \cdot w(k) \cdot \exp(j2\pi kn/N)| \tag{110}$$

which now has the following interpretation: $|X_\ell(n)|$ is obtained by cascading a bandpass filter with impulse response w(k)exp(j2πkn/N) and a RMS circuit. Also, it can be shown that an equivalent result is obtained if an envelope detector is used instead of the RMS circuit.

As a way to compare the interpretations given, we emphasize the respective role of the window in each form. It appears as a simple window function w(1-k) first, then as (low-pass) impulse response w(k) and finally as (bandpass) impulse response w(k)exp(j2πkn/N).

3.2.2 ANALYSIS AND SYNTHESIS METHODS

Spectral analysis is the transform of eq. (107). As shown right above, it can be realized either by linear filtering of x(k) according to eq. (110) or by FFT of w(l-k) · x(k).

Spectral synthesis is given by the inverse DFT of X (n) according to eq. 2. Three synthesis methods are now shown. To each corresponds a particular vocoder structure.

3.2.3 CHANNEL VOCODER

Principally, the speech spectrum is divided by a number of continguous bandpasses at the input and reconstructed by addition at the output. A channel is affected to each bandpass.

Practically, either a filter bank to implement digital filtering or FFT to implement DFT is being used both for analysis and synthesis. An other practical aspect is the division of the spectrum in several bands. Both uniform and non-uniform spectrum division is used. In the case of non-uniform spectrum division, the channel bandwidth increases with frequency (exponential law for instance) to account for the characteristics of the human ear.

Fig. 20 shows the principle of a channel vocoder using filter banks both for analysis and synthesis.

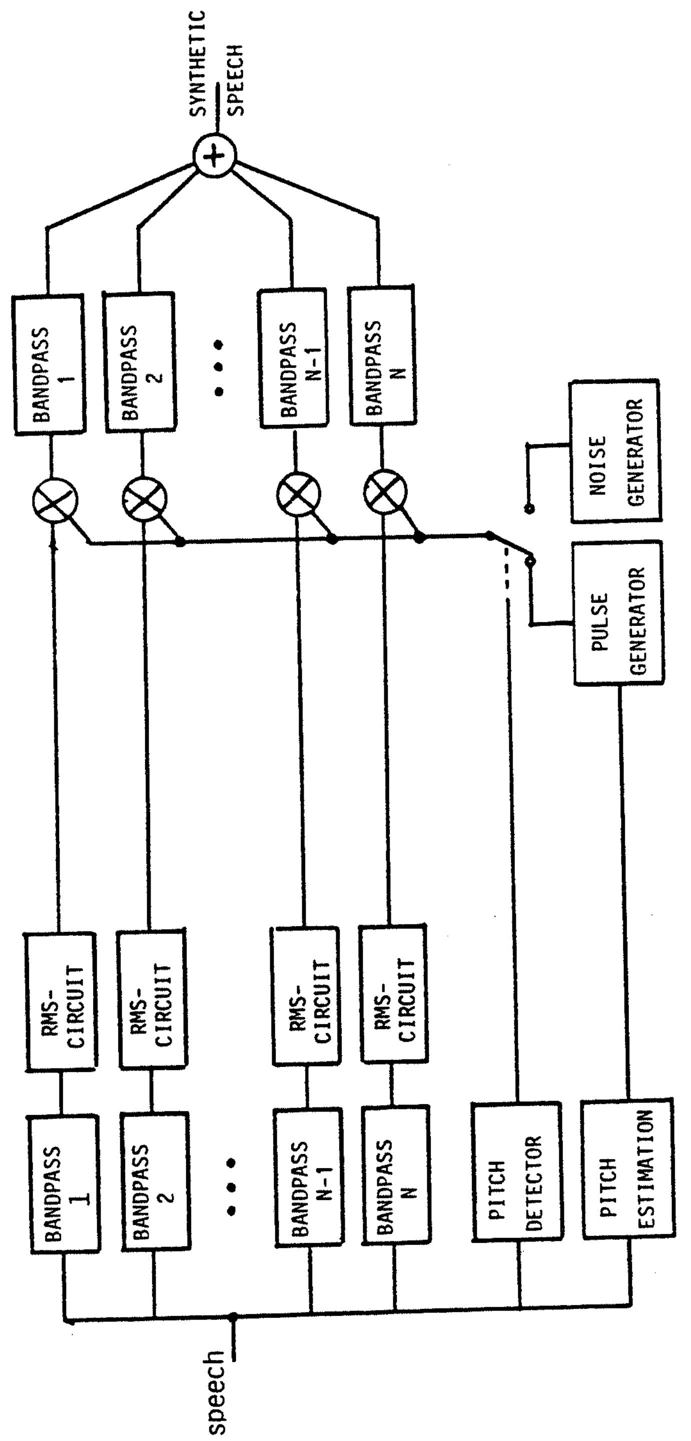


Fig. 20 Channel vocoder

3.2.4 FORMANT VOCODER

Rather than dividing the spectrum envelope in fixed contiguous bandpasses, this approach characterizes the spectral envelope by more specific information like the position, the amplitude and width of the formants. A better speech quality or better compresion factor is expected by this more specific approach.

Starting from a speech synthesis model which uses a cascade of digital resonators as a time-varying digital filter, the basic step is to find the model parameters fulfilling the following criteria: best match of the spectral envelope found by analysis, with the DFT of the system impulse response. For example, a system can use a 5 pole digital filter to account for voiced sounds and a 1 pole-1 zero digital filter for unvoiced sounds. The variable filter parameters are the pole positions F_1, F_2, F_3 for the voiced component and both the pole and zero position Fp and Fz for the unvoiced component.

A first analysis method directly finds the first three maxima of the short-time spectrum by simple peak detection. Practically, a rather large number of channels (30 to 50) is required in order to obtain a high enough spectral resolution to make those maxima detectable. Note that some kind of smoothing must be applied to the spectrum in order to remove the periodic pitch structure. By one method call cepstral smoothing, the pitch signal is filtered out from the cepstrum x (k).

A second, more simple analysis method divides the spectrum in three analysis channels by bandpass-filtering. The bandpasses are designed such as to capture each, one of the first formants F_1, F_2, F_3 and also, they are selected broad-banded enough to allow a sufficiently large variation of formant position. Then, the exact formant position and power is obtained by measurement of both the average zero-crossing rate eq. (106) as well as the average power in each channel.

A third method, called analysis-by-synthesis, proceeds by successive approximation. The model parameter are varied as long as to get, at the output, a good approximation of the input function.

Fig. 21 shows also an example of pole-zero locations for the two time-varying digital filters of the formant synthesizer.

3.2.5 LINEAR PREDICTION VOCODER

To remedy certain shortcomings encountered with the formant synthesizer, the filter model is now changed in that the cascade of second-order filters is replaced by a higher order linear system. The purpose of this system is to model together the excitation pulse shape, the vocal tract and other effects as well. The transfer function of the filter is of the form:

$$H(z) = \frac{1}{1 - \sum_{i=1}^{M} a_i \cdot z^{-i}} \tag{111}$$

where M is the filter order, typical values being 10 or 12; and $\{a_i, i = 1 \cdot M\}$ are the model parameter called the predictor coefficients. Their determination was shown in section 2.7.3.

Fig. 22 illustrates the corresponding linear predictive (LPC) synthesizer.

3.3 PITCH DETECTION AND PITCH PERIOD ESTIMATION

The purpose of pitch analysis is to provide the two pitch dependant parameters of the model: excitation mode (voiced/unvoiced) and pitch period value. The corresponding analysis procedures are pitch detection respectively pitch period estimation.

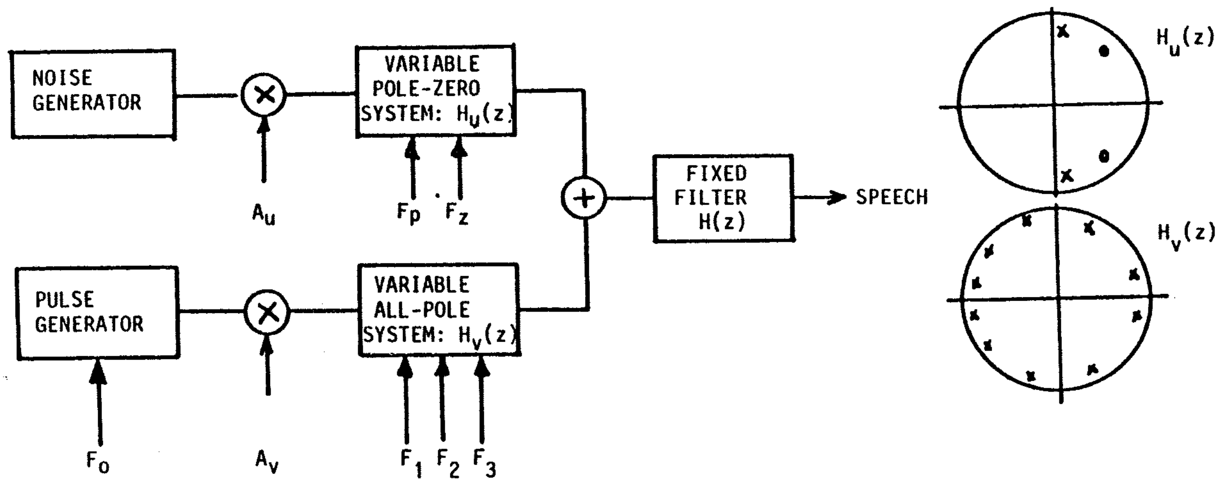


Fig. 21 Formant vocoder and typical pole-zero diagrams

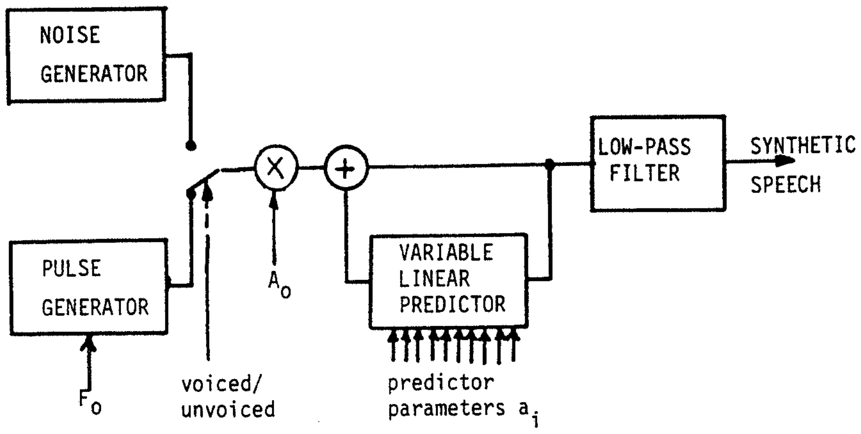


Fig. 22 Linear prediction synthesizer

Both because the human ear is very sensitive to the pitch and because the pitch contributes much to the naturalness of speech, correct pitch analysis and restitution is decisive for the quality of synthesized speech.

Pitch analysis methods are numerous and various and practical algorithms often combine seveal of them. For a comparison see for example [6] or [7]. We restrict this presentation to the basic methods. Also, note that for most practical methods, the effective processing is preceded by preprocesing like normalization, low-pass filtering or clipping and followed by postprocessing like smoothing, correction etc.

Pitch detection method can be divided in three categories: time-domain, frequency domain and hybrid methods.

3.3.1 TIME DOMAIN METHODS

The basic idea is to preprocess the quasiperiodic speech signal such as to reduce sufficiently the formant structure and then to use simple time domain methods to estimate the pitch.

3.3.1.1 ZERO CROSSING

The principle is: low-pass filtering followed by zero-crossing (eq. 106) measurements. A particularity however is that the useful pitch and the filter cut-off frequency are related, with the practical consequence of a limited pitch frquency range for a given filter configuration. To remedy that, adaptive filtering as a function of F can be performed.,

3.3.1.2 PEAK AND VALLEY MEASUREMENTS

The principel is: low-pass filtering followed by peak and valley detection and a final decision for the choice of the pitch. Practically, there are 6 parallel detectors: peak, valley, peak-to-valley, valley-to-peak, peak-to-previous-peak and valley-to-previous-valley detectors. The decision is made to guarantee correct pitch measurement in the case of a signal with both fundamental and second harmonic.

3.3.1.3 AUTOCORRELATION FUNCTION

This method uses the property of autocorrelation: the autocorrelation function of a periodic signal is periodic with the same period, i.e. it will show a peak at a lag value equal to the signal period. Further, the peak has same amplitude as the peak at the origin. For quasiperiodic signals, the peak is slightly reduced.

Practically, a window (eq. 104) is applied to the speech signal and the autocorrelation function from eq. 4 is computed:

$$\varphi_\ell(m) = \sum_{k=\ell}^{\ell+N-1} x(k) \cdot w(\ell-k) \cdot x(k+m) \cdot w(\ell-k-m) \tag{112}$$

which can be computed directly or by FFT (see eq. 6). The lag value m corresponding to the first maximum of $\varphi_\ell(m)$ indicates the pitch period value.

The performance of this method is significantly improved by a non-linear preprocessing called central clipping which sets small relative values to 0.

Computation of the autocorrelation function can be simplified by quantizing the signal to the three levels +1, 0, -1. This eliminates all multiplications encountered with the normal computation of eq. 4 or 6.

3.3.1.4 AVERAGE MAGNITUDE DIFFERENCE FUNCTION

As the one just described, this method is a simple to implement approach to pitch detection. Compared to the autocorrelation function, it uses subtractions instead of multiplications. It takes advantage of the fact that for a periodic signal, $x(k)-x(k+p) = 0$ when p is a delay equal to one or several periods. The average magnitude difference function (AMDF) thus writes:

$$\mathrm{AVDF}(m) = \sum_{k=\ell}^{\ell+N-1} [x(k) \cdot w(\ell-k) - x(k+m) \cdot w(\ell-k-m)] \tag{113}$$

The detected pitch period is the lag value m for the first minimum of AVDF(m).

3.3.2 SPECTRAL METHODS

Frequency-domain pitch detectors use the property that, if the signal is periodic in time, then the frequency spectrum of the signal will consist of a series of impulses at the fundamental frequency and its harmonics. This harmonic structure, only present with voiced sounds, must be detected and the frequency interval between two lines must be measured.

3.3.2.1 CEPSTRUM METHOD

As we have seen, a basic step of the cepstrum computation is the logarithmic transform of the amplitudes in the spectral domain. The speech spectrum X(n) which is the product of the excitation spectrum E(n) and the vocal tract spectrum R(n) is thus transformed in the sum of two signals:

$$\ln|E(n) \cdot R(m)| = \ln|E(n)| + \ln|R(n)| \tag{114}$$

After inverse DFT we get the cepstrum $\hat{x}(k)$ which now, is the sum of the cepstra of e(k) resp. r(k): $\hat{x}(k) = \hat{e}(k) + \hat{r}(k)$. As the excitation e(k) is periodic, its cepstrum $\hat{e}(k)$ will display a strong peak, indicating the position of the pitch period, whereas the slow and aperiodic oscillation r(k) give raise to a much flatter curve.

Practical cepstral computation requires a good resolution in the spectral domain. Therefore difficulties arise for small values of F_o where the spectral lines are close to each other.

3.3.2.2 COMB FILTERING IN THE SPECTRAL DOMAIN

The principle is to filter the short-time speech spectrum $X_\ell(n)$ with a variable comb filter in the spectral domain. The comb filter is a function $C(n,n_o)$ where n_o is the interval between two teeth of the comb. The method looks for the maximum of the cross-correlation when the teeth interval n_o is varied. When this maximum is reached, the comb lies exactly on the harmonic lines of $X_\ell(n)$, and thus: $F_o = n_o$. This writes formally:

$$F_o = \underset{n_o}{\operatorname{argmax}} \left[\sum_{n=0}^{N/2-1} X_\ell(n) \cdot C(n,n_o) \right] \tag{115}$$

3.3.3 HYBRID METHOD

Hybrid methods use features of both the time- and the spectral domain.

3.3.3.1 SIMPLIFIED INVRESE FILTERING TECHNIQUE

The basic idea here is to first eliminate or reduce the effect on x(k) of the spectral envelope (which is a characteristic of the vocal tract, not the pitch) and then to detect the pitch by the autocorrelation method. Because the second of these two steps is described above, we concentrate on step one only.

Here then, the way adopted eliminates the effect of the spectral envelope by flattening it, which is also called signal whitening. Practically, this can be done in the time-domain by inverse filtering, i.e. by a filter which whitens the spectrum of x(k).

3.3.4 CONCLUSION

Basic pitch detection methods have been presented to give the reader an idea of principles and processing methods involved. To the subject, we keep in mind that pitch detection remains a difficult task.

4. SPEECH RECOGNITION

This section presents two aspects found in most speech recognition application: feature extraction and pattern matching.

4.1 FEATURE EXTRACTION

The most versatile methods can and have been used to extract features from the speech signal. Because most of them have already been encountered in this text, we give here just a list of feature extraction methods.

Time-domain methods:

— Average energy or amplitude
— Average zero-crossing rate
— Autocorrelation function

— Parcor or LPC coefficients

— Voiced/unvoiced, pitch value

— Amplitude distribution function

Transform methods:

— Fourier spectrum

— Hadamard transform

— Cepstrum coefficient

— Formants

4.2 PATTERN MATCHING

In either speech or speaker recognition, the basic principle of recognition used is pattern matching: the unknown speech pattern is compared to already available speech reference patterns, then, the best match is detected. If we denote D(T,Ri) the pattern distance measure for the unknown test pattern T and a reference pattern Ri, i = 1...I, then the recognition decision is:

$$i* = \underset{i}{\operatorname{argmin}} [D(T,Ri)] \tag{116}$$

Considering speech patterns as time sequence of feature vectors, i.e.:

$$T = \underline{t}(1), \underline{t}(2), ..., \underline{t}(m), ..., \underline{t}(M) \tag{117}$$

$$Ri = \underline{r}_i(1), \underline{r}_i(2), ..., \underline{r}_i(n), ..., \underline{r}_i(N)$$

and defining the distance measure of two feature vectors:

$$d_i(m,n) = \delta[\ \underline{t}(m),\underline{r}_i(n)\ ] \tag{118}$$

where δ is some adequate distance function, we will present the basic methods for matching the sequences.

The most obvious method establishes a one-to-one correspondence between both sequences (which thus must be of equal length M = N) so that the overall distance is :

$$D(T,Ri) = \sum_{i=1}^{M} d_i(m,m) \tag{119}$$

4.3 DYNAMIC TIME WARPING

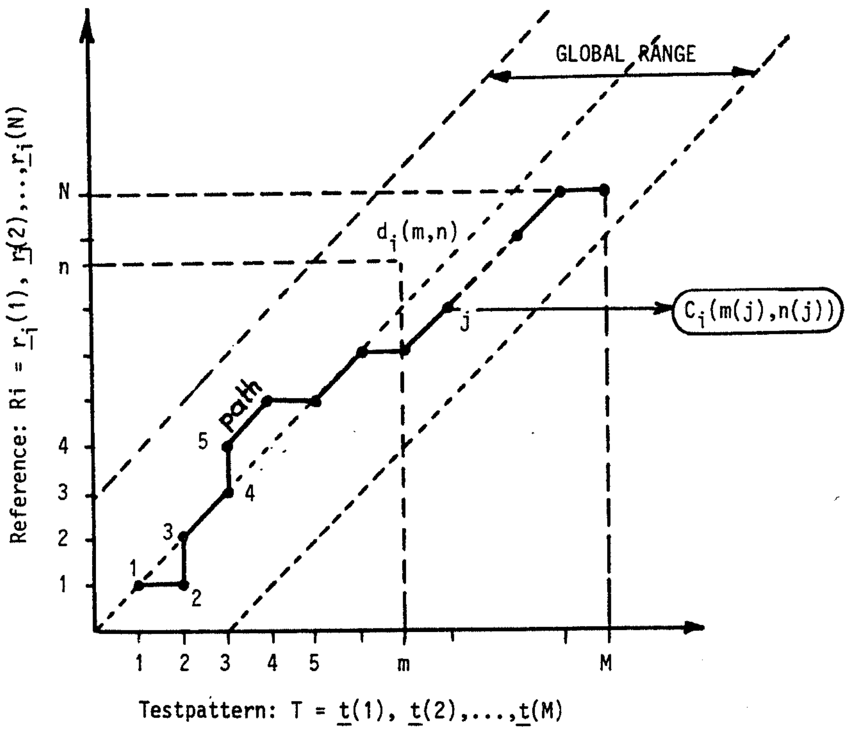
The Dynamic time Warping (DTW) algorithm provides a procedure to align optimally in time the test and reference sequence and to find the optimal distance D (T,Ri) associated to the optimal warping path. The algorithm uses the principle of Dynamic Programming which is the theory telling how to find optimal paths in graphs. It operates in the two-dimensional field of $d_i(m,n)$ distances shown is Fig. 23, and finds under given constraints the optimal path leading from $d_i(1,1)$ to $d_i(M,N)$. If we define the cumulated distance at a given point of the path as :

$$C_i(m,n) = \sum_{j=1}^{J} d_i(m(j),\ n(j)) \cdot w(j) \tag{120}$$

where the pairs [k(j),1(j)], j = 1...J, describe a given path and w(j) is a given associated weighting function, then the optimal path is the one which minimizes $C_i(M,N)$, the cumulated distance at $d_i(M,N)$. Formally,

$$D_1(T,Ri) = \min_{\text{path}} C_i(M,N)] \tag{121}$$

The constraints used are various and fulfill two basic purposes: locally, limiting the range of the path slope; globally, limiting the path domain as shown for example in Fig. 23.



GLOBAL RANGE
Reference: Ri = $\underline{r}_i(1), \underline{r}_i(2), \ldots, \underline{r}_i(N)$
N
n
4
3
2
1
$d_i(m,n)$
path
j
$C_i(m(j),n(j))$
1 2 3 4 5
1 2 3 4 5
m
M
Testpattern: T = $\underline{t}(1), \underline{t}(2), \ldots, \underline{t}(M)$

Fig. 23 Principle of DTW

4.4 CONNECTED PATTERN MATCHING

Connected pattern matching is useful to recognize connected speech patterns like connected phonemes or connected words. Again we consider the set of refrences 'Ri" and the unknown pattern T which, now, stands for a single pattern made of other patterns connected together. T must now be compared with all compound reference patterns made of connected references Ri. We call superrefernce R* such a compound reference. Using the notation and definition of :

$$R_i \otimes R_j = r_i(1),r_i(2),\ldots,r_i(N_i),r_j(1),\ldots,r_j(N_j) \qquad (122)$$

for connected time sequences, we write the genral superreference as :

$$R*(\underline{q}) = R_{q(1)} \otimes R_{q(2)} \otimes \ldots \otimes R_{q(L)} \qquad (123)$$

where $\underline{q} = [q(1),q(2),q(3),q(4),\ldots q(L)]$ is the vector of reference indices.

With the brute force matching approach, the test pattern is compared by DTW with all possible superrefrences built from the reference set {Ri, i = 1...I}, resulting in $I + I^2 + I^L$ single matching operations.

4.5 TWO-LEVEL DTW

Starting from the DTW schema between T and Ri explained above, two levels are built by normalizing all references to a constant length $N_1 = N_2 = \ldots = N_0$ and thus considering fixed boundaries in the superreference. We give the indices 1 = 1,2,...,L to the boundaries. They appear as horizontal lines in Fig. 24. We call lower level processing, the one performed in-between the lines and upper level processing, the one which applies to operations performed on the boundary lines.

The principle of the algorithm is as in the case of simple DTW. The two-dimensional distance field can now be written as $d_i(m,n,1)$, m = 1,...,M; n = 1...N_0; 1 = 1...L. We search the optimal path leading from $d_i(1,1,1)$ to $d_i(M,N,1)$ where 1 = 1...L.

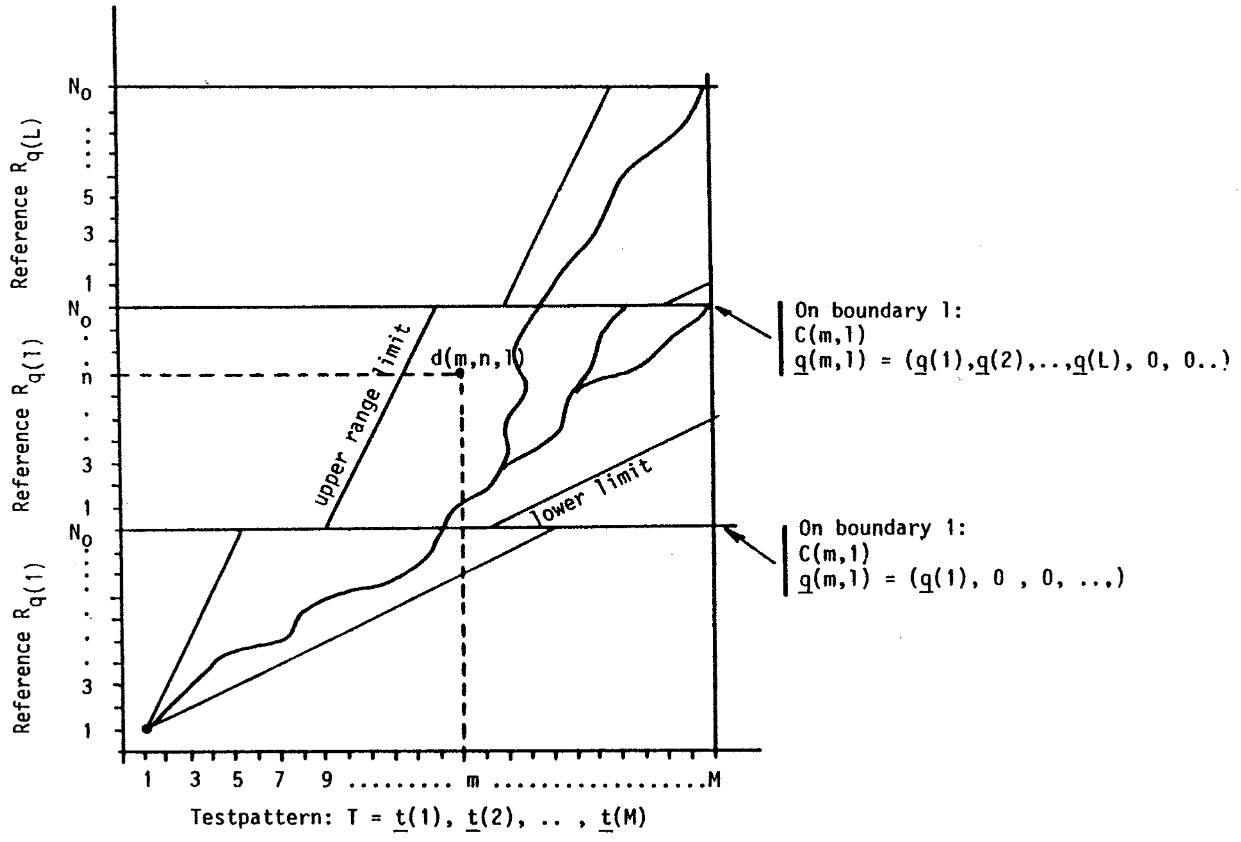

Reference $R_{q(L)}$
Reference $R_{q(1)}$
Reference $R_{q(1)}$
N_o
1
3
5
n
upper range limit
lower limit
d(m,n,1)
1 3 5 7 9 .... m ....M
Testpattern: T = $\underline{t}(1)$, $\underline{t}(2)$, .. , $\underline{t}(M)$
On boundary 1:
C(m,1)
$\underline{q}(m,1)$ = ($\underline{q}(1)$,$\underline{q}(2)$,..,$\underline{q}(L)$, 0, 0..)
On boundary 1:
C(m,1)
$\underline{q}(m,1)$ = ($\underline{q}(1)$, 0 , 0, ...)

Fig. 24 Principle of two-level DTW

The upper level keeps track, at each boundary line 1 and for each position m of the test sequence, of the optimal path reaching that position in the form of the optimal cumulated distance C (m,1) as well as a path descriptor which can be the partial reference index vector $\underline{q}$ (m,1), by which we mean that, at the boundary 1, the vector components are defined up to component number 1.

The lower level performs normal DTW between T and {Ri} in the field bounded by two boundaries 1 and 1-1 with given boundary constraints. It finds, among all paths starting from the boundary 1-1 the best path reaching each position m of the boundary 1. Each lower level is repeated for each reference {Ri, i = 1...I} which ensures the optimality of the path found.

Upper and lower level processing once performed for each boundary 1 = 1...L, the final optimal path is the one given by :

$$D[T,R*(\underline{q})] = \min[C\ (M,1)] \tag{124}$$

D [T,R*($\underline{q}$)] is the optimal distance for the two-level DTW and $\underline{q}$ is the index vector giving the optimal matching.

The advantage of Two-Level DTW compared to the brute force method is to drastically reducing the number of computations and to making thus connected pattern recognition possible. It requires computing I basic DTW whereas this number is $I+I^2+...+I^L$ in the case of brute force.

5. CONCLUSION

In this paper a short tutorial review of major digital signal processing methods used in processing speech signals is given. In depth treatments are omitted to the detriment of the essence and insight for interpretation. Following more than two decades of efforts in designing methods and algorithms, the present trend is in transposing these methods and algorithms into VLSI implementable architectures for better, faster and more reliable practical systems.

It seems reasonable to expect that now, algorithm designers and circuit designers will collaborate increasingly to design algorithms more suitable for VLSI and to develop technological tools and architectures more convenient for digital signal processing and hence speech processing.

REFERENCES

1. M. Kunt, "Traitement numérique des signaux" Presses Polytechniques Romandes, Lausanne, Switzerland, 1980.
2. J. D. Markel and A. H. Gray, "Linear prediction of speech" Springer-Verlag, New York, 1976.
3. L. R. Rabiner and B. Gold, "Theory and application of digital signal procesing", Prentice Hall, Englewood Cliffs, 1975.
4. T. G. Stockham, T. M. Cannon and R. B. Ingebretsen, "Blind deconvolution through digital signal processing" Proc. IEEE, VolL. 63, No. 4, April 1975, pp. 678-692.
5. L. R. Rabiner and R. W. Schafer, "Digital processing of speech signals", Prentice-Hall, N. J., 1978.
6. L. R. Rabiner et al., "A comparative study of several pitch detection algorithms", IEEE Trans. ASSP-24, pp. 399-413.
7. W. Hess, "Algorithmes et methodes pour la determination du fondamental", 12eme Journees d'Etude sur la Parole, Montreal, May 25-27, 1981.