.

1. Introduction

1.1 Basic Digital Signal Processing Algorithms; A Short Overview

This chapter briefly introduces the most common signal processing algorithms which are used and treated in the following chapters. These algorithms are used directly in the most of DSP tasks, or solve as base of more complex algorithms (adaptive and nonlinear filtering, data compression pattern recognition etc. ...). Since the theoretical background of these algorithms is discussed in detail by the most of DSP literature we only briefly summarize them.

1.1.1 Description of signals

Several kinds of transforms and signal description methods are known and used in signal processing. These transforms emphasize special features of the signals, make possible to reduce the redundancy in information or reduce the computational requirement using them.

Because of the importance of the linear, time invariant systems, the real and complex frequency domain has emphasized role in the signal processing. The most simple periodic continuous and discrete signals can be described by Fourier series:

$$x(t) = \sum_{n=-\infty}^{\infty} c_n \, e^{j2\pi n f_0 t}, \tag{1.1}$$

where f_0 is the frequency of the signal and t is the discrete or continuous time.

To put continuous transient signals to the real or complex frequency domain, the Fourier and the Laplace transforms are used. The Fourier transform has different definitions in the literature. We use the following formulas for the Fourier and Laplace transforms:

Fourier transform and its inverse:

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi ft}\, dt \tag{1.2}$$

$$x(t) = \int_{-\infty}^{\infty} X(f)\, e^{j2\pi ft}\, df \tag{1.3}$$

Laplace transform and its inverse:

$$X(s) = \int_{-\infty}^{\infty} x(t)\, e^{-st}\, dt \tag{1.4}$$

$$x(t) = \frac{1}{2\pi j} \int_{c-j-\infty}^{c+j\infty} X(s)\, e^{st}\, ds \tag{1.5}$$

The discrete Fourier transform (DFT) and the z transform play the same role in the analysis of discrete-time signals as the Fourier and Laplace transforms do in the analysis of continuous-time signals.

The direct and inverse z transforms:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} \tag{1.6}$$

$$x(n) = \frac{1}{2\pi j} \oint_C X(z)\, z^{n-1}\, dz; \tag{1.7}$$

where C is any contour that encircles the origin.

The direct and inverse DFT:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi \frac{nk}{N}} \tag{1.8}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j2\pi \frac{nk}{N}} \tag{1.9}$$

DFT and IDFT can be written in matrix product form as well:

$$\underline{X} = \underline{\underline{W}}\, \underline{x} \tag{1.10}$$

$$\underline{x} = \frac{1}{N} \underline{\underline{W}}^{-1} \underline{X}, \tag{1.11}$$

where $\underline{\underline{W}} = \{W_{nk}\} = \{e^{-j2\pi \frac{nk}{N}}\}$ is an N by N quadratic matrix, $\underline{x}$ and $\underline{X}$ are vectors made from the time samples and the result of DFT.

Besides the frequency domain, the so called lag domain is also used for stochastic signals. The correlation function describes the signals in the lag domain. The $R_{x,y}$ cross-correlation of x and y stationary random processes is the following:

$$R_{x,y}(\tau) = E\{x(t)\,y(t+\tau)\} \tag{1.12}$$

where $E\{x\}$ denotes the expected value of x. If x and y are ergodic processes, the correlation function can be calculated in the following way:

$$R_{x,y}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T x(t)\,y(t+\tau)\,dt \tag{1.13}$$

for continuous signals and

$$R_{x,y}(k) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} x(n)\,y(n+k) \tag{1.14}$$

for discrete signals.

Stochastic processes can be described in the frequency domain by its power density spectrum:

$$S_x(f) = \int_{-\infty}^{\infty} R_x(\tau)\,e^{-j2\pi f\tau}\,d\tau \tag{1.15}$$

or

$$S_x(f) = \sum_{n=-\infty}^{\infty} R_x(n)\,e^{-j2\pi nf} \tag{1.16}$$

where R_x is the autocorrelation function of x (the cross correlation of x and x).

1.1.2 Linear systems

Linear time-invariant systems can be described by their impulse responses (h) or transfer functions (H). The transfer function can be given in the discrete or continuous frequency (f), the complex frequency (s) or the z domain. The following expressions give the relation between the input (x) and output (y) of a linear time-invariant system:

In the time domain:

$$y(t) = \int_{-\infty}^{\infty} x(t-u)\,h(u)\,du \tag{1.17}$$

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)\,x(n-k) \tag{1.18}$$

In the lag domain:

$$R_{x,y}(\tau) = \int_{-\infty}^{\infty} R_x(\tau - u)\, h(u)\, du \tag{1.19}$$

$$R_y(\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R_x(\tau - u + v)\, h(u)\, h(v)\, du\, dv \tag{1.20}$$

$$R_{x,y}(n) = \sum_{k=-\infty}^{\infty} R_x(n-k)\, h(k) \tag{1.21}$$

$$R_y(n) = \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} R_x(n-k+l)\, h(n)\, h(l) \tag{1.22}$$

In the frequency domain:

$$Y = HX \tag{1.23}$$

$$S_y = |H|^2 S_x \tag{1.24}$$

1.1.3 Nonlinear systems

In the early development of signal processing, linear filters were the primary tools. Their mathematical simplicity, the well developed theoretical background and the existence of some desirable properties make them easily design and implement. Moreover, linear filters offered satisfactory performance in many applications. However linear filters have poor performance in the presence of noise that is not additive as well as in problems where system nonlinearities or non-Gaussian statistics are encountered. For such reasons, nonlinear filtering/signal processing methods became more and more important in the last decade.

A filter can be defined in a mathematical sense as an operator L(.), which maps a signal x into an other signal y . When L(.) is a linear operator the filter is linear, for nonlinear filters the operator is not linear. The lack of satisfaction of the superposition or proportionality principles or both does not allow us to obtain the output of a nonlinear filter by convolving the input with its impulse response function. For this reason different families of nonlinear filters have been studied, each of which has a different characterization which makes the filters in this class amenable to some form of mathematical analysis.

A special and the simplest possible nonlinear transformation of the input signal is a memoryless nonlinearity that maps the signal x to y. A memoryless nonlinearity implements a static mapping from x to y. This means that current value of y depends only on the current value of x. In the last ten-twenty years a new field of general nonlinear systems, the neural networks have been developed. Neural networks can be considered as general systems that can implement almost any static nonlinear mapping. In many cases the required transformation of input signal cannot be done using memoryless systems, static mapping is not enough for given tasks. In such cases dynamic systems can be applied where the current value of y depends not only on the current, but the previous values of the input, x, or in many cases the previous values of y itself. Nonlinear dynamic systems or from an other point of view nonlinear filters can be categorized as the extensions of the corresponding linear systems. From the point of view of nonlinear dynamic systems we can say about NFIR, NARX, NOE, NARMAX, NBJ systems. As nonlinear filters there are many families which can be distinguished. One of the most popular families of nonlinear filters for noise removal is the family of order statistic filters. There exist several filters which are members of this family, e.g. median filters, the stack filters, the median hybrid filters, etc. An other important family of nonlinear filters are such nonlinear filters which retain the linear-in-the parameter structure. The most important member of this family is the Volterra filters, but many neural network-based dynamic systems such as RBF filters, FIR-MLPs or TDNNs can be regarded as members of this family as well. When we are thinking about nonlinear systems, nonlinear filters and about their implementations we can find problems which are common with the linear systems, filters, and of course there are some significant differences. Some basic questions concerning nonlinear systems are discussed in chapter 5. The first part of this chapter deals with the most important nonlinear filters structures, while the second larger part with neural networks.

1.2 Computing methods

Most of the expressions written in previous sections cannot be computed numerically, even if we consider only the equations for the discrete signals, because of the unlimited length of signals. Even if we use finite records of signals, the computational requirements are so large, that we have to find more effective algorithms.

For linear systems Fourier transform plays a crucial role. Its importance is emphasized by the fact, that the Discrete Fourier Transform (DFT) can be computed by computationally very effective algoritms the Fast Fourier Transform. Because of its efficiency, several other expressions have computation method that use the FFT. Since the FFT algorithm is usually well known and a lot of sources can be found that describe its several versions in detail, we don't introduce it. Instead of that we introduce, how the previous expressions can be calculated using FFT.

Correlation function

The definition of the correlation function (1.14) is not suitable for calculation even if the length of the record is finite, because of the high computational requirement. If the length of x and y is N ($x(n) = y(n) = 0$ if $n < 0$ or $n \geq N$), the required operation number for calculation of R_{xy} (which length is $2N-1$) is proportional to N^2:

$$R_{x,y}(k) = \frac{1}{N-k} \sum_{n=0}^{N-1} x(n)y(n+k) \qquad k = (1-N)..(N-1) \quad (1.25)$$

Let consider, how can we use the FFT for calculating the correlation function. Let denote X and Y the FFT of x and y. The expression of the correlation function estimation is the following:

$$\hat{R}_{x,y}(l) = \frac{1}{N-k} \mathrm{IFFT}(X^*Y) \quad (1.26)$$

where X^* is the complex conjugate of X.

$$\hat{R}_{x,y}(k) = \frac{1}{N(N-k)} \sum_{l=0}^{N-1} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(n) e^{j2\pi \frac{nl}{N}} y(m) e^{-j2\pi \frac{ml}{N}} e^{j2\pi \frac{lk}{N}} \quad (1.27)$$

$$\hat{R}_{x,y}(k) = \frac{1}{N(N-k)} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(n)y(m) \sum_{l=0}^{N-1} e^{-j2\pi \frac{(m-n-k)l}{N}} \quad (1.28)$$

$$\hat{R}_{x,y}(k) = \frac{1}{N-k} \sum_{n=0}^{N-1} x(n)y[(n+k)|N] \qquad k = 0..(N-1) \quad (1.29)$$

where $m|N$ is the m modulo N.

The (1.29) expression of $\hat{R}_{xy}$ is called circural correlation. This circular correlation is the biased estimator of (1.25). To avoid the error in $\hat{R}_{x,y}$ caused by the circularity, we have to clear the second half of the $2N$ long x and y arrays for FFT, and then we get the following result:

$$\hat{R}_{x,y}(k) = \frac{1}{N-k} \sum_{n=0}^{2N-1} x(n)y[(n+k)|2N] \qquad k = 0..(2N-1) \quad (1.30)$$

Since $x(n) = y(n) = 0$ if $n \geq N$, the connection between $R_{x,y}$ and $\hat{R}_{x,y}$ is the following:

$$\hat{R}_{x,y}(k) = \begin{cases} R_{x,y}(k) & \text{if } 0 \leq k < N \\ \\ R_{x,y}(k-2N) & \text{if } N \leq k < 2N \end{cases} \tag{1.31}$$

Convolution

Very similarly to the calculation of the correlation function, the convolution of two finite length signal also can be calculated by using FFT. Let x and y two, N sample long signals and X and Y are their DFT. The IDFT of XY similarly to the circular correlation yields circular convolution. To avoid the circularity we have to use again $2N$ long arrays for FFT with 0-s in the second half. This method is efficient only if we have large, but finite arrays. If we wanted to use it for filtering, we should calculate the FFT of x and IFFT of XY step by step and it needs more calculation than the original formula.

1.3 Digital Filters

The linear digital filters can be described as all linear systems by their transfer function or by the impulse response. Since both the input and the output of a digital filter are discrete in time, the transfer function can be describe in z domain.

The two types of digital filters, FIR (finite-duration impulse response) and IIR (infinite duration impulse response), are treated in this section. They have very different characteristics, yet they can often meet the same specifications. The FIR filter has a transfer function that is a polynomial in z^{-1} and is an all-zero filter in the sense that the zeros in the z plane determine the frequency-response magnitude characteristic. Although a length N FIR filter has a pole order $N-1$ at the origin of the z plane, a pole at the origin does not affect the magnitude of the frequency response of the filter. An FIR filter can have an impulse response that is symmetric around the point $(N-1)/2$ and can therefore exactly linear phase.

The IIR filter has both poles and zeros in the z^{-1} plane and in the z plane. The combination of a pole near the pass-band edge with a zero near the stop-band edge can give an IIR filter very short transition region between the pass band and the stop band. Generally, an IIR filter can give a sharper cutoff than an FIR filter of the same order because both poles and zeros are present. However, a casual IIR filter cannot achieve exactly linear phase but the FIR filter can. The phase and group delay characteristics of conventional IIR filters are generally not as good as those of FIR filters.

Which filter is better for a particular application depends on the hardware used for the implementation of the filter. All DSPs have special instructions (MAC, circular buffer handle addressing modes, hardware loop instructions etc.) to facilitate the implementation of an FIR filter. On the other hand, the IIR filters require less memory than an FIR filter that meets about the same frequency-domain specifications. The IIR filter, when implemented in fixed-point arithmetic, may have instabilities (limit cycles) and may have large quantization noise, depending on the word length of the processor. The FIR filter, on the other hand, is usually implemented in non-recursive way, which guarantees a stable filter.

For narrow-band, sharp cutoff filters where phase is not important, IIR filters are likely to superior to FIR filters. For applications where wave shape is important, the FIR filter with its good phase characteristics is usually a good choice.

Two parts to the filter design process are the approximation problem and the implementation problem. The approximation part of the problem deals with the choice of parameters or coefficients in the filter's transfer function to approximate an ideal or desired response. The implementation part of the filter design problem deals with choosing a structure to implement the transfer function. This structure may be in the form of a circuit diagram if the filter is to be built of components, or it may be a program to be used on a DSP. In the next the implementation part of the filter design problem will be discussed.

1.3.1 FIR filter implementation problem

The FIR filters can be described by their h impulse response function:

$$y(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k) \tag{1.32}$$

where x is the input and y is the output of the filter.

The (1.32) expression shows the simplest implementation method of FIR filters: $x(n-k)$ for $k = 0..N-1$ must be produced for example using circular buffer, and with the $h(k)$ coefficients a weighted sum must be calculated. This is the direct non-recursive method. Though several other structures exist, this direct method is the most popular, because all DSPs support it by special instructions and addressing modes.

1.3.2 IIR filter implementation problem

The expression of IIR filters is nearly as simple as eq. (1.32):

$$y(n) = \sum_{m=0}^{M} b(m)\, x(n-m) - \sum_{m=1}^{N} a(m)\, y(n-m) \tag{1.33}$$

In frequency domain:

$$H(z) = \frac{\sum_{m=0}^{M} b(m)\, z^{-m}}{1 + \sum_{m=1}^{N} a(m)\, z^{-m}} = \frac{B(z)}{A(z)} \tag{1.34}$$

Expression (1.33) can be implemented likewise FIR filters. When a recursive digital filter is implemented directly, as FIR filters, errors introduced by quantization of coefficients can cause significant variation from the desired frequency response. A filter designed to be stable can become unstable after coefficients are quantized. The coefficient sensitivity of transfer function increases increasing the N order of filter. While a second order IIR filter is usually can be implemented in direct form, direct implementation of a 10th or higher order IIR filter is nearly impossible. This fact yields the next two structures.

In the cascade structures by factoring, we can write the (1.34) transfer function as

$$H(z) = K \prod_{k=1}^{[N/2]} \frac{1 + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 + a_{1k} z^{-1} + a_{2k} z^{-2}} \tag{1.35}$$

where [N/2] is the smallest integer $\geq N/2$.

The parallel structures use the partial fraction expansion of $H(z)$:

$$H(z) = \sum_{k=1}^{[N/2]} \frac{c_0 k - c_1 k z^{-1}}{1 - a_1 k z^{-1} - a_2 k z^{-2}} + \sum_{k=0}^{M-N} p_k z^{-k} \tag{1.36}$$

Each of the second-order factors in (1.35) and (1.36) can be implemented with direct structures. If the filter has an odd order, then a first- or third-order section is necessary. Though these structures have much better properties than the direct structure has, the stability of the filter cannot be warranted in the case of coefficient quantization.

Several other structures are developed for IIR filter implementation. Lots of them have much better coefficient sensitivity than that of the previous ones, and can warrant the filter stability as well. One of them is the resonator-based digital filter structure. It can be described by the following state equations:

$$\underline{q}(n+1) = [\mathrm{diag}(\underline{z}) - \underline{g}\underline{c}^T]\underline{q}(n) + \underline{g}\,x(n) \tag{1.37}$$

$$y(n) = (\underline{w}^T - \underline{c}^T d)\underline{q}(n) + d\,x(n) \tag{1.38}$$

where x is the input, y is the output and $\underline{q}$ is the state vector, $\underline{z}$, $\underline{g}$, $\underline{w}$ are the parameter vectors of the filter, $\underline{c}^T = [1, 1, ..., 1]$.

The transfer function of the filter is the following:

$$H(z) = \frac{d + \sum_{n=0}^{N-1} w_n\, H_n(z)}{1 + \sum_{n=0}^{N-1} H_n(z)} \tag{1.39}$$

where

$$H_n(z) = \frac{g_n\, z^{-1}}{1 - z_n z^{-1}} \tag{1.40}$$

$\underline{z}$, $\underline{g}$ and $\underline{w}$ parameter vectors can be calculated from the original $H(z)$ transfer function considering equations (1.39) and (1.40).

The resonator-based digital filters are structurally passive and can suppress all zero-input limit cycles, and if rounding is applied provides minimum roundoff noise. These properties are due to the fact that this structure generates its output as a linear combination of orthogonal signal components, thus internally implementing an orthogonal realization of a lossless transfer function.