Programme: M.E.(PC&I)

Year/Semester: First/Second

Course Name: Advanced Digital Signal Processing

Course Code: 19EIPCPE24

Course Instructor: dr.S.Santhi

Unit-IV

Adaptive filter

Adaptive filters are digital filters whose coefficients change with an objective to make the filter converge to an optimal state. The optimization criterion is a cost function, which is most commonly the mean square of the error signal between the output of the adaptive filter and the desired signal. As the filter adapts its coefficients, the mean square error (MSE) converges to its minimal value. At this state, the filter is adapted and the coefficients have converged to a solution. The filter output, *y(k)*, is then said to match very closely to the desired signal, *d(k)*. When the input data characteristics is changed, sometimes called *filter environment*, the filter adapts to the new environment by generating a new set of coefficients for the new data. Adaptive filter can be implemented as FIR or IIR type. The FIR adaptive filter is represented as shown in Fig.1.

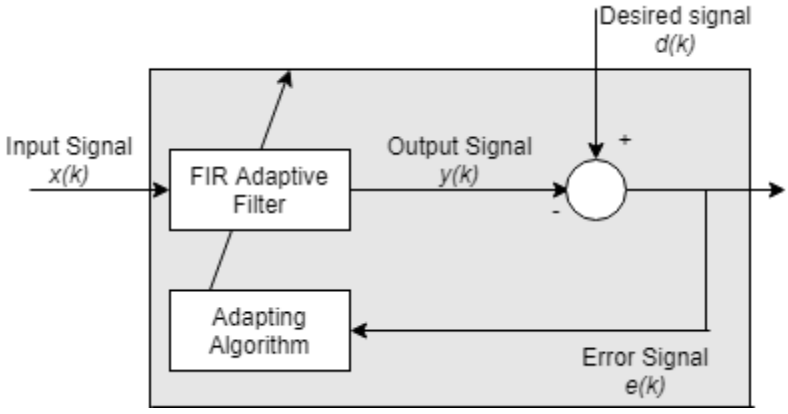


Fig.1. Adaptive Filter representation

Adaptive Filter for System Identification

The adaptive system identification is primarily responsible for determining a discrete estimation of the transfer function for an unknown digital or analog system. The same input x(n) is applied to both the adaptive filter $W_n(z)$and the unknown system (Plant) from which the outputs are compared. The output of the adaptive filter y(n) is subtracted from the output of the unknown system resulting in a desired signal d(n). The resulting difference is an error signal e(n) used to manipulate the filter coefficients of the adaptive system trending towards an error signal of zero. After performing this process for a number of iterations, and if the system is designed correctly, the adaptive filter's transfer function will converge to, or near to, the unknown system's transfer function. For this configuration, the error signal does not have to go to zero, although convergence to zero is the ideal situation, to closely approximate the given system. There will, however, be a difference between adaptive filter transfer function and the unknown system transfer function if the error is nonzero and the magnitude of that difference will be directly related to the magnitude of the error signal. Additionally the order of the adaptive system will affect the smallest error that the system can obtain. If there are insufficient coefficients in the adaptive system to model the unknown system, it is said to be under specified. This condition may cause the error to converge to a nonzero constant instead of zero. In contrast, if the adaptive filter is over specified,

meaning that there are more coefficients than needed to model the unknown system, the error will converge to zero, but it will increase the time it takes for the filter to converge. In this v(n) represents measurement noise.

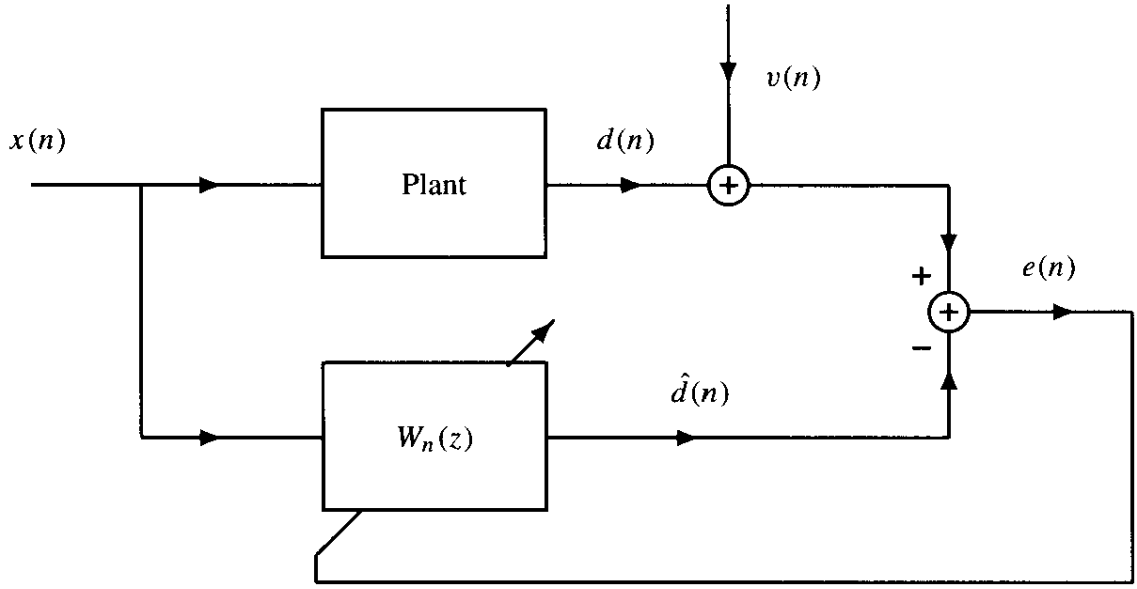


Fig.2. Adaptive Filter for System Identification

Adaptive filter for Noise Cancellation

The second configuration is the adaptive noise cancellation configuration as shown in figure 3. In this configuration the input x(n) is a combination of desired signal d(n) and a noise source v(n), output of adaptive filter y(n) is compared with a signal x(n) until it cancels v(n) that is until the error e(n) becomes zero. The adaptive filter coefficients adapt to cause the error signal to be a noiseless version of the signal x(n). The noise signal for this configuration need to be uncorrelated to the signal d(n).

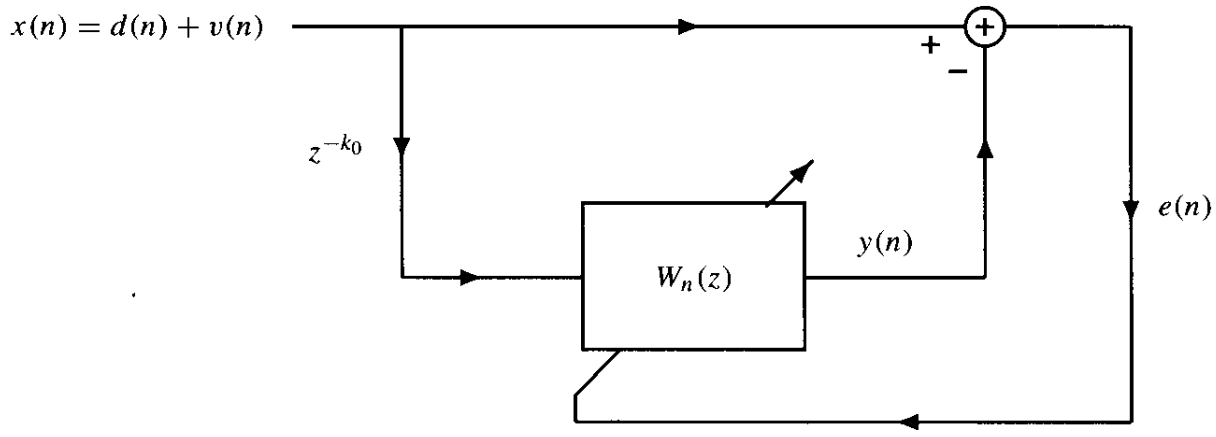


Fig.3. Adaptive filter for Noise cancellation

Adaptive Filter for Linear Prediction

Adaptive linear prediction is the third type of adaptive configuration (see figure 3). This configuration essentially performs two operations. The first operation, if the output is taken from the error signal e(n), is linear prediction. The adaptive filter coefficients are being trained to predict, from the statistics of the input signal x(n), what the next input signal will be. The second operation, if the output is taken from

y(n), is a noise filter similar to the adaptive noise cancellation outlined in the previous section. As in the previous section, neither the linear prediction output nor the noise cancellation output will converge to an error of zero. This is true for the linear prediction output because if the error signal did converge to zero, this would mean that the input signal x(n) is entirely deterministic, in which case we would not need to transmit any information at all.

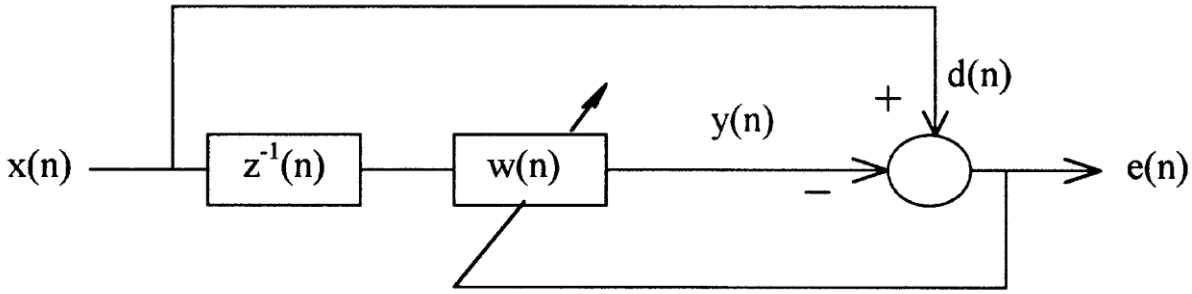


Fig. 4. Adaptive filter for Linear prediction.

Finite Impulse Response (FIR) Algorithms- Least Mean Squares Gradient Approximation Method

Given an adaptive filter with an input $\underline{x}(n)$, an impulse response w(n) and an output y(n) you will get a mathematical relation for the transfer function of the system

$$y(n) = \underline{w}^T(n)\underline{x}(n) \text{ and}$$

$$x(n) = [x(n), x(n-1), x(n-2), \ldots , x(n-(N-1))]$$

where $\mathbf{w}^T(n) = [w_0(n), w_1(n), w_2(n) \ldots w_{N-1}(n)]$ are the time domain coefficients for an N^{th} order FIR filter. Note in the above equation and throughout a boldface letter represents a vector and the super script T represents the transpose of a real valued vector or matrix.

Using an estimate of the ideal cost function the following equation can be derived.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \Delta E[e^2(n)]$$

In the above equation $\underline{w}(n+1)$ represents the new coefficient values for the next time interval, m is a scaling factor, and D $E[e^2(n)]$ is the ideal cost function with respect to the vector $\underline{w}(n)$. From the above formula one can derive the estimate for the ideal cost function

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\, e(n)\mathbf{x}(n) \text{ where}$$

$$e(n) = d(n) - y(n) \text{ and}$$

$$y(n) = \mathbf{x}^T(n)\mathbf{w}(n).$$

In the above equation m is sometimes multiplied by 2, but here we will assume it is absorbed by the m factor. In the Least Mean Squares Gradient Approximation Method, often referred to as the Method of Steepest Descent, a guess based on the current filter coefficients is made, and the gradient vector, the derivative of the MSE with respect to the filter coefficients, is calculated from the guess. Then a second guess is made at the tap-weight vector by making a change in the present guess in a direction opposite to the gradient vector. This process is repeated until the derivative of the MSE is zero.

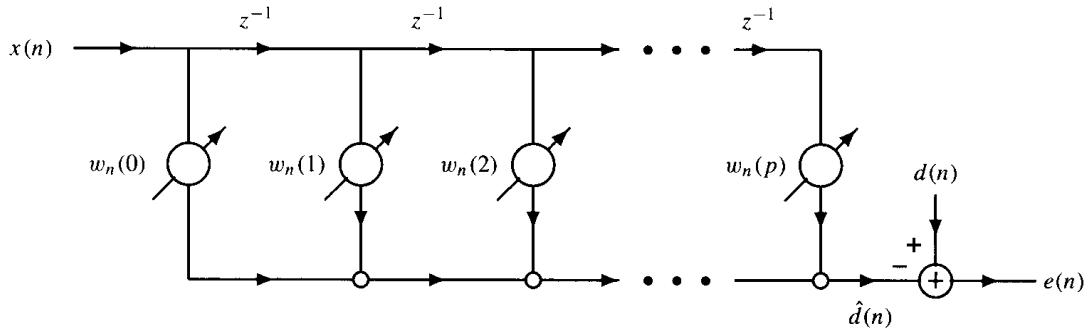


Fig.5 FIR Adaptive filter

Convergence of the LMS Adaptive Filter

The convergence characteristics of the LMS adaptive filter is related to the autocorrelation of the input process as defined by

$$\mathbf{R}x = E[\mathbf{x}(n)\mathbf{x}^T(n)]$$

There are a two conditions that must be satisfied in order for the system to converge. These conditions include:

- The autocorrelation matrix, **R**x, must be positive definite.
- $0 < \mu < 1/\lambda_{max}$., where λ_{max} is the largest eigenvalue of **R**x.

In addition, the rate of convergence is related to the eigenvalue spread. This is defined using the condition number of **R**x, defined as $k = l_{max}/l_{min}$, where l_{min} is the minimum eigenvalue of **R**x. The fastest convergence of this system occurs when k = 1, corresponding to white noise. This states that the fastest way to train a LMS adaptive system is to use white noise as the training input. As the noise becomes more and more colored, the speed of the training will decrease.

Infinite Impulse Response (IIR) Adaptive Filters

The primary advantage of IIR filters is that to produce an equivalent frequency response to an FIR filter, they can have a fewer number of coefficients. This in theory should reduce the number of adds, multiplies and shifts to perform a filtering operation. This theory of using IIR filters to reduce the computational burden is the primary motivation for the use of IIR adaptive filters. There are, however, a number of problems that are introduced with the use of IIR adaptive filters.

- The fundamental concern with IIR adaptive filters is the potential for instability due to poles moving outside the unit circle during the training process. Even if the system is initially stable and the final system is stable, there is still the possibility of the system going unstable during the convergence process. Some suggestion has been made to limit the poles to within the unit circle, however, this method requires that the step sizes be small, which considerably reduces the convergence rate.

- Due to the interplay between the movement of the poles and zeros, the convergence of IIR systems tends to be slow [3]. The result is that even though IIR filters have fewer coefficients, therefore few calculations per iteration, the number of iterations may increase cause a net loss in processing time to convergence. This, however, is not a problem with all pole filters.
- In an IIR system, the MSE surface may contain local minimum that can cause a convergence of that system to the local minimum instead of the absolute minimum. More care need to be taken in the initial conditions in IIR adaptive filters than in FIR adaptive filters.
- IIR filters are more susceptible to coefficient quantization error than FIR, due to the feedback.

There have been a number of studies done on the use of IIR adaptive filters, but due to the problems stated above, they are still not widely used in industry today.

Newton's Steepest Descent method of Adaptive filter algorithm

The method pivots on the point that the slope at any point on the surface provides the best direction to move in. It is a feedback approach to finding the minimum of the error performance surface. The steepest descent direction gives the greatest change in elevation of the surface of the cost function for a given step laterally. The steepest descent procedure uses the knowledge of this direction to move to a lower point on the surface and find the bottom of the surface in an iterative manner.

Consider a system identification problem in which the output of a linear FIR filter must match to the desired response signal d(n). The output of this filter is given by

$$d_1(n)= W^T(n)x(n)$$

where $x(n) = [x(n)\ x(n-1)\ \cdots\ x(n-L+1)]^T$ is a vector of input signal samples and $W(n) = [w_1(n)\ w_2(n)\ \cdots\ w_{L-1}(n)]^T$ is a vector containing the coefficients or the weights of the FIR filter at time n.

Now, the coefficient vector W(n) needs to be found out that optimally replicates the input-output relationship of the unknown system such that the cost function of the estimation error given by

$e(n) = d(n)- d_1(n)$

is the smallest among all possible choices of the coefficient vector.

An apt cost function is to be defined to formulate the steepest descent algorithm mathematically. The mean-square-error cost function is given by

$$J(n) = E\{((e(n))^2\} =E\{(d(n)-W^T(n)x(n))^2\}$$

Where J(n) is the non- negative function of the weight vector.

Step 1: in order to implement the steepest descent algorithm, the partial derivatives of the cost function are evaluated with respect to the coefficient values. Since derivatives and expectations are both linear operations, we can change the order in which the two operations are performed on the squared estimation error.

$$\frac{\partial E\{e2(n)\}}{\partial W(n)}=E\{2e(n)\frac{\partial e(n)}{\partial W(n)}\}$$

$$=-2E\{e(n)x(n)\}$$

Step 2: Finding the difference in the coefficient vector of two consecutive time spaced weights which forms the basis for the algorithm.

$$W(n+1) = W(n) + \mu E\{e(n)x(n)\}$$

Where μ is step size of the algorithm and

$$\Delta W= W(n+1) - W(n)$$

Step 3: Determining the expectation from the above equation

$E\{e(n)x(n)\} = E\{x(n)(d(n)-d_1(n))\}$

$= P_{dx}(n) - R_{xx}(n)W(n)$

Where $R_{xx}(n)$ is autocorrelation matrix of input vector and $P_{dx}(n)$ is the cross correlation matrix vector of the desired response signal and the input vector at time n. The optimal coefficient vector is found out when the estimation error or ΔW is zero.

$W_{opt}(n) = R_{xx}^{-1}(n) P_{dx}(n)$

The aim is to iteratively descend to the bottom of the cost function surface, so that W(n) approaches Wopt(n) i.e. the coefficient vector is updated repetitively using a strategy analogous to that of the ball rolling in a bowl.

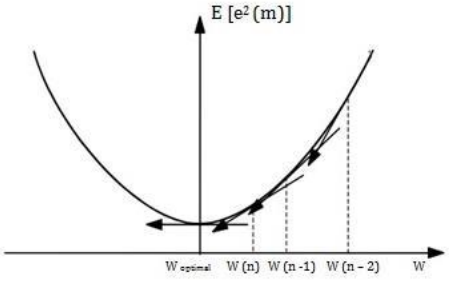


Fig.6 Mean square Error cost function for a single coefficient linear filter

From the figure Fig.5, the following facts become evident:

1) The slope of the function is zero at the optimum value associated with the minimum of the cost function.

2) There is only one global minimum of the bowl- shaped curve and no local minima.

3) The slope of the cost function is always positive at points located to the right of the optimum parameter value.

4) For any given point, the larger the distance from this point to the optimum value, the larger is the magnitude of the slope of the cost function.

The current tap weights are moved in the direction opposite to that of the slope of the cost function at the current parameter value. Additionally, if the magnitude of the change in the parameter value is proportional to the magnitude of the slope of the cost function, the algorithm will make large adjustments of the parameter value when its value is far from the optimum value and will make smaller adjustments when the value is close to the optimum value. This approach is the essence of the steepest descent algorithm. The filter coefficients are successively updated in the downward direction, until the minimum point, at which the gradient is zero, is reached.

Widrow Hoff LMS adaptive algorithm

Optimization Criterion

To minimize the mean square error $E\{e^2(n)\}$

Adaptation Procedure

It is an approximation of the steepest descent method where the expectation operator is ignored, i.e., $\frac{\partial E\{e2(n)\}}{\partial W(n)}$ is replaced by $\frac{\partial\{e2(n)\}}{\partial W(n)}$

The LMS adaptation algorithm is as follows.

$$w(n+1)=w(n)-\mu \frac{\partial\{e2(n)\}}{\partial W(n)}$$

$$=w(n)- \mu \frac{\partial\{e2(n)\}}{\partial e(n)} \cdot \frac{\partial\{e(n)\}}{\partial w(n)}$$

$$=w(n)-2 \mu e(n) \frac{\partial\{d(n)-wT(n)x(n)\}}{\partial w(n)}$$

$$=w(n)+2\mu e(n)x(n)$$

$w_i(n+1)=w_i(n)+ 2\mu e(n)x(n-i)$ where i=1,2,….L-1

Advantages

1.Low computational complexity
2. Simple to implement
3. Allow real-time operation
4. Does not need statistics of signals, i.e., R_{xx} and R_{dx}

Performance Surface

The mean square error function or performance surface is identical to that in the Wiener filtering:

$E\{e^2(n)\}= \varepsilon_{min}+(w(n)-w_{MMSE})^T R_{xx}(w(n)-w_{MMSE})^T$

Where w(n) is the adaptive filter coefficient vector at time .

Recursive Least Square Adaptive Filter

Recursive least squares (RLS) is an adaptive filter algorithm that recursively finds the coefficients that minimize a weighted linear least squares cost function relating to the input signals. This approach is in contrast to other algorithms such as the least mean squares (LMS) that aim to reduce the mean square error. In the derivation of the RLS, the input signals are considered deterministic, while for the LMS and similar algorithm they are considered to be stochastic. Compared to most of its competitors, the RLS exhibits extremely fast convergence. However, this benefit comes at the cost of high computational complexity. In general, the RLS can be used to solve any problem that can be solved by adaptive filters. For example, suppose that a signal d(n) is transmitted over an echoey, noisy channel that causes it to be received as

$$x(n) = v(n) + \sum_{k=0}^{q} (bn(k) + d(n-k))$$

Where v(n) represents additive noise. The intent of RLS filter is to recover the desired signal d(n) using p+1 tap FIR filter

$$d(n) = \sum_{k=0}^{p} (w(k) + x(n-k)) = w^T X_n$$

Where $X_n=[x(n),x(n-1),\ldots x(n-p)]^T$ is a column vector containing p+1 most recent samples of x(n). The estimate of recovered desired signal is

$$\hat{d}(n) = \sum_{k=0}^{p} (wn(k) + x(n-k)) = w_n^T X_n$$

he goal is to estimate the parameters of the filter w, and at each time n we refer to the current estimate as w_n and the adapted least-squares estimate by w_{n+1}. w_n is also a column vector, as shown below, and the transpose, w_n^T, is a row vector. The matrix product $w_n^T X_n$ (which is the dot product of w_n and X_n) is , a scalar. The estimate is *"good"* if $\hat{d}(n)-d(n)$ is small in magnitude in some least squares sense.

As time evolves, it is desired to avoid completely redoing the least squares algorithm to find the new estimate for w_{n+1}, in terms of w_n.

The benefit of the RLS algorithm is that there is no need to invert matrices, thereby saving computational cost. Another advantage is that it provides intuition behind such results as the Kalman filter.

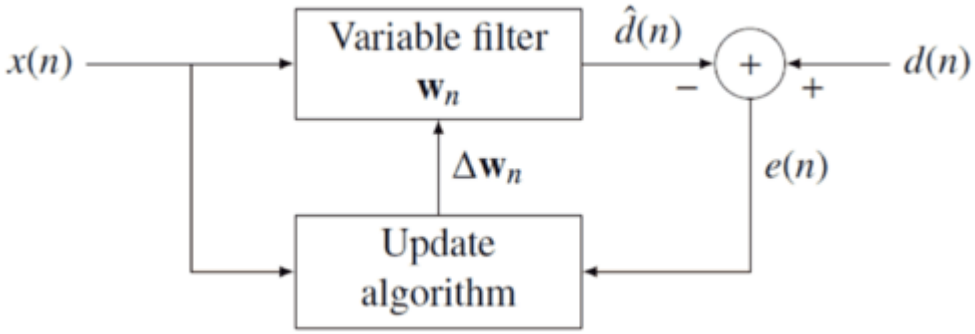


Fig. 7 RLS Adaptive filter

The error implicitly depends on the filter coefficients through the estimate $d^{\wedge}(n)$

$E(n)=d(n)-d^{\wedge}(n)$

The weighted least squares error function $C(w(n))$ is the cost function we desire to minimize and being a function of $e(n)$ is therefore also dependent on the filter coefficients:

$$C(wn) = \sum_{n=0}^{\infty} (\lambda)^{n-i}\, e2(i)$$

Where $0<\lambda\leq 1$ is the forgetting factor which gives exponentially less weight to older error samples.

The cost function is minimized by taking the partial derivatives for all entries k of the coefficient vector w_n and setting the results to zero

$$\frac{\partial C(wn)\}}{\partial Wn(k)}=\sum_{i=0}^{n} 2(\lambda)^{n-i}\, e(i)\, \frac{\partial e(i)\}}{\partial Wn(k)}=-\sum_{i=0}^{n} 2(\lambda)^{n-i}\, e(i)\, x(i-k)=0$$

Next, replace e(n) with the definition of the error signal

$$\sum_{i=0}^{n}(\lambda)^{n-i}\, e(i)[d(i)-\sum_{l=0}^{p} wn(l)\, x(i-l)]\, x(i-k)=0 \qquad k=0, 1,\ldots p$$

Rearranging the equation yields

$$\sum_{l=0}^{p} wn(l) \sum_{i=0}^{n}(\lambda)^{n-i}\, x(i-l)\; x(i-k)= \sum_{i=0}^{n}(\lambda)^{n-i}\, d(i)\; x(i-k) \qquad k=0, 1,\ldots p$$

This form can be expressed in terms of matrices as below.

$R_x(n)w_n=r_{dx}(n)$

where $R_x(n)$ is the weighted sample covariance matrix for x(n), and $r_{dx}(n)$ is the equivalent estimate for the cross-covariance between d(n) and x(n). Based on this expression we find the coefficients which minimize the cost function as

$w_n=R_x^{-1}(n)r_{dx}(n)$

This is the result of our main discussion.

Choosing λ

The smaller λ is, the smaller is the contribution of previous samples to the covariance matrix. This makes the filter *more* sensitive to recent samples, which means more fluctuations in the filter coefficients. The $\lambda=1$ case is referred to as the *growing window RLS algorithm*. In practice, λ is usually chosen between 0.98 and 1. By using type-II maximum likelihood estimation the optimal λ can be estimated from a set of data.

Recursive algorithm

The discussion resulted in a single equation to determine a coefficient vector which minimizes the cost function. In this section we want to derive a recursive solution of the form

$w_n=w_{n-1}+\Delta w_{n-1}$

where Δw_{n-1} is a correction factor at time n-1

We start the derivation of the recursive algorithm by expressing the cross covariance $r_{dx}(n)$ in terms of $r_{dx}(n-1)$

$$rdx(n) = \sum_{i=0}^{n} (\lambda)^{n-i}\, d(i)x(i)$$

$$= \sum_{i=0}^{n-1} (\lambda)^{n-i}\, d(i)x(i) + (\lambda)^0\, d(n)x(n)$$

$$= \lambda r dx(n-1) + d(n)x(n)$$

Where x(i) is p+1 dimensional data vector.
$X(i)=[x(i),x(i-1),\ldots x(i-p)]^T$

Similarly we express $R_x(n)$ in terms of $R_x(n-1)$ by
$R_x(n)= \sum_{i=0}^{n} (\lambda)^{n-i}\, x(i)xT(i)$
$R_x(n)=\lambda R_x(n-1)+x(n)x^T(n)$
In order to generate the coefficient vector we are interested in the inverse of the deterministic auto-covariance matrix. For that task the Woodbury matrix identity comes in handy. With
$A=\lambda R_x(n-1)$ is p+1 by p+1
$U=x(n)$ is p+1 by 1 column vector
$V=x^T(n)$ is 1 by p+1 row vector
$C=I_1$ is 1 by 1 Identity matrix
The woodbury identity matrix follows as
$R_x^{-1}(n)=[\lambda R_x(n-1)+x(n)x^T(n)]^{-1}$
$=\lambda^{-1}R_x^{-1}(n-1)- \lambda^{-1}R_x^{-1}(n-1)x(n)$
$=\{1+x^T(n)\, \lambda^{-1}R_x^{-1}(n-1)x(n)\}^{-1}x^T(n)\, \lambda^{-1}R_x^{-1}(n-1)$
To come in line with the standard literature, we define
$P(n)=R_x^{-1}(n)$
$=\lambda^{-1}P(n-1)-g(n)x^T(n)\, \lambda^{-1}P(n-1)$
Where the gain vector g(n) is given as
$g(n)= \lambda^{-1}P(n-1)x(n)\{1+x^T(n)\, \lambda^{-1}P(n-1)x(n)\}^{-1}$
$=P(n-1)x(n)\{\lambda+x^T(n)P(n-1)x(n)\}^{-1}$
Before we proceed we bring g(n) in another form
$g(n)\{1+x^T(n)\, \lambda^{-1}P(n-1)x(n)\}= \lambda^{-1}P(n-1)x(n)$
$g(n)+g(n)x^T(n)\, \lambda^{-1}P(n-1)x(n)= \lambda^{-1}P(n-1)x(n)$
With the recursive definition of P(n) the desired form follows
$g(n)=P(n)x(n)$
Now we are ready to complete the recursion. As discussed
$W_n=P(n)r_{dx}(n)$
$=\lambda P(n)r_{dx}(n-1)+d(n)P(n)x(n)$
The second step follows from the recursive definition of $r_{dx}(n)$. Next we incorporate the recursive definition of P(n) together with alternate form of g(n) and get w(n) as
$w_n=\lambda[\lambda^{-1}P(n-1)-g(n)x^T(n)\, \lambda^{-1}P(n-1]r_{dx}(n-1)+d(n)g(n)$
$=P(n-1)r_{dx}(n-1)-g(n)x^T(n)P)n-1)r_{dx}(n-1)+d(n)g(n)$
$= P(n-1)r_{dx}(n-1)+g(n)[d(n)-x^T(n)P(n-1)r_{dx}(n-1)]$
With w_{n-1} as $P(n-1)r_{dx}(n-1)$ we arrive at the update equation
$w_n=w_{n-1}+g(n)[d(n)-x^T(n)w_{n-1}]$
$= w_{n-1}+g(n)\alpha(n)$
Where $\alpha(n)= [d(n)-x^T(n)w_{n-1}]$ is the priori error. Compare this with the *a posteriori* error; the error calculated *after* the filter is updated:
$e(n)= d(n)-x^T(n)w_n$
It means that we found the correction factor as
$\Delta w_{n-1} =g(n)\alpha(n)$
This intuitively satisfying result indicates that the correction factor is directly proportional to both the error and the gain vector, which controls how much sensitivity is desired, through the weighting factor λ.
The RLS algorithm for p^{th} order RLS filter can be summarized as follows.
Parameters:
p=Filter order
λ=Forgetting factor
δ=Value to initialize P(0)
Initialization:
w(n)=0;

x(k)=0 for k=-p, ….-1
d(k)=0 for k=-p, ….-1
P(0)= δI where I is the identity matrix of rank p+1
Computation for n=1,2,….
$x(n)=[x(n),x(n-1),\ldots x(n-p)]^T$

$\alpha(n)=d(n)-x^T(n)w(n-1)$

$P(n)=P(n-1)x(n)\{\lambda+x^T(n)P(n-1)x(n)\}^{-1}$

$P(n)= \lambda^{-1}P(n-1)-g(n)x^T(n)\ \lambda^{-1}P(n-1)$

$w(n)=w(n-1)+\alpha(n)g(n)$

The recursion for P(n) follows algebraic Riccati equation and thus draws parallel to Kalman filter.

Unit-V

Multirate and Wavelet transform

Basics of Multirate

Multirate simply means "multiple sampling rates". A multirate DSP system uses multiple sampling rates within the system. Whenever a signal at one rate has to be used by a system that expects a different rate, the rate has to be increased or decreased, and some processing is required to do so. Therefore "Multirate DSP" really refers to the art or science of changing sampling rates. The most immediate reason is when you need to pass data between two systems which use incompatible sampling rates. For example, professional audio systems use 48 kHz rate, but consumer CD players use 44.1 kHz; when audio professionals transfer their recorded music to CDs, they need to do a rate conversion. But the most common reason is that multirate DSP can greatly increase processing efficiency (even by orders of magnitude!), which reduces DSP system cost. This makes the subject of multirate DSP vital to all professional DSP practitioners.

Multirate consists of:

1. **Decimation:** To decrease the sampling rate,
2. **Interpolation:** To increase the sampling rate, or,
3. **Resampling:** To combine decimation and interpolation in order to change the sampling rate by a fractional value that can be expressed as a ratio. For example, to resample by a factor of 1.5, you just interpolate by a factor of 3 then decimate by a factor of 2 (to change the sampling rate by a factor of 3/2=1.5.)

Multirate Digital Signal Processing

1.Introduction

Multirate systems have gained popularity since the early 1980s and they are commonly used for audio and video processing, communications systems, and transform analysis to name but a few. In most applications multirate systems are used to improve the performance, or for increased computational efficiency. The two basic operations in a multirate system are decreasing (*decimation*) and increasing (*interpolation*) the sampling-rate of a signal. Multirate systems are sometimes used for *sampling-rate conversion*, which involves both decimation and interpolation.

2.Decimation

Decimation can be regarded as the discrete-time counterpart of sampling. Whereas in sampling we start with a continuous-time signal $x(t)$ and convert it into a sequence of samples $x[n]$, in decimation we start with a discrete-time signal $x[n]$ and convert it into another discrete-time signal $y[n]$, which consists of *sub-samples* of $x[n]$. Thus, the formal definition of M-fold decimation, or down-sampling, is defined by Equation 1 In decimation, the sampling rate is reduced from F_s to F_s/M by discarding $M - 1$ samples for every M samples in the original sequence.

$$y[n] = v[nM] = \sum_{k=-\infty}^{\infty} h[k]x[nM - k] \tag{1}$$

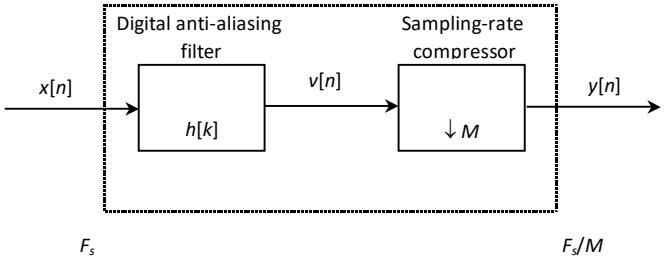


Figure 1. Block diagram notation of decimation, by a factor of M.

The block diagram notation of the decimation process is depicted in Figure.1. An anti-aliasing digital filter precedes the down-sampler to prevent aliasing from occurring, due to the lower sampling rate. The subject of aliasing in decimated signals is covered in more detail in Section 4. In Figure .2 below, it illustrates the concept of 3-fold decimation i.e. $M = 3$. Here, the samples of $x[n]$ corresponding to $n = \ldots, -2, 1, 4, \ldots$ and $n = \ldots, -1, 2, 5, \ldots$ are lost in the decimation process. In general, the samples of $x[n]$ corresponding to $n \neq kM$, where k is an integer, are discarded in M-fold decimation. In Figure.2 (b), it shows samples of the decimated signal $y[n]$ spaced three times wider than the samples of $x[n]$. This is not a coincidence. In real time, the decimated signal appears at a slower rate than that of the original signal by a factor of M. If the sampling frequency of $x[n]$ is F_s, then that of $y[n]$ is F_s/M.

3. Interpolation

Interpolation is the exact opposite of decimation. It is an information preserving operation, in that all samples of $x[n]$ are present in the expanded signal $y[n]$. The mathematical definition of L-fold interpolation is defined by Equation 2. and the block diagram notation is depicted in Figure .3. Interpolation works by inserting $(L-1)$ zero-valued samples for each input sample. The sampling rate therefore increases from F_s to LF_s. With reference to Figure .3, the expansion process is followed by a unique digital low-pass filter called an *anti-imaging filter*. Although the expansion process does not cause aliasing in the interpolated signal, it does however yield undesirable replicas in the signal's frequency spectrum. We shall see how this special filter, in Section 9.4, is necessary to remove these replicas from the frequency spectrum.

$$y[n] = L\sum_{k=-\infty}^{\infty} h[k]w[n - k] \tag{2}$$

Where w(n)=x(n/L) if L is an integer

=0 if L is non integer

Figure .4 below, it depicts 3-fold interpolation of the signal $x[n]$ i.e. $L = 3$. The insertion of zeros effectively attenuates the signal by L, so the output of the anti-imaging filter must be multiplied by L, to maintain the same signal magnitude.

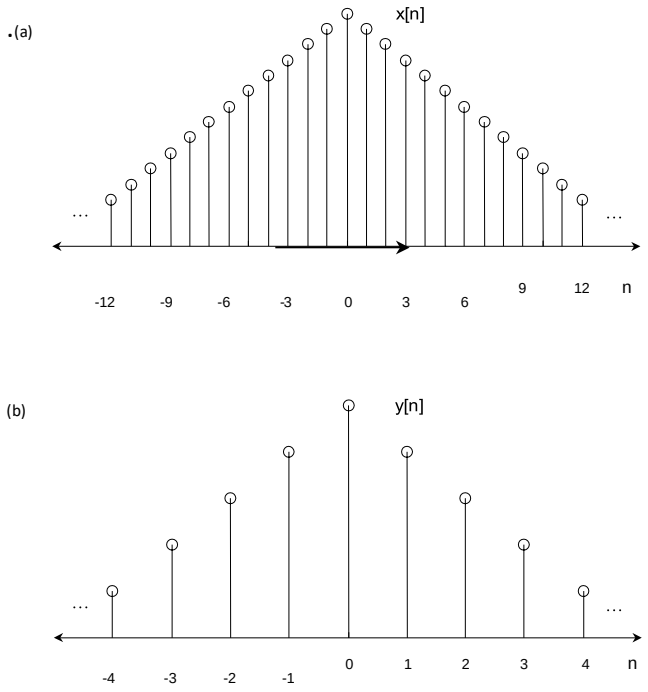


Figure 2. Decimation of a discrete-time signal by a factor of 3.

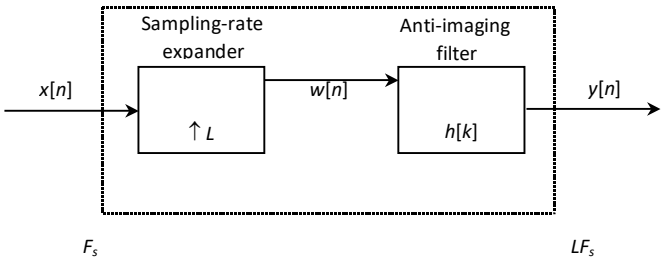


Figure 3. Block diagram notation of interpolation, by a factor of L.

4. Frequency Transforms of Decimated and Expanded Sequences

The analysis of decimation and expansion is better understood by assessing their respective frequency spectrums using the Fourier transform.

i. Decimation

The implications of aliasing caused by decimation are very similar to those in the case of sampling a continuous-time signal. In general, if the Fourier transform of a signal, $X(\Box)$, occupies the entire bandwidth from $[-\Box, \Box]$, then the Fourier transform of the decimated signal, $X_{(\Box M)}(\Box)$, will be aliased. This is due to the superposition of the M shifted and frequency-scaled transforms. This is illustrated in Figure .5 below, which shows the aliasing phenomenon for $M = 3$.

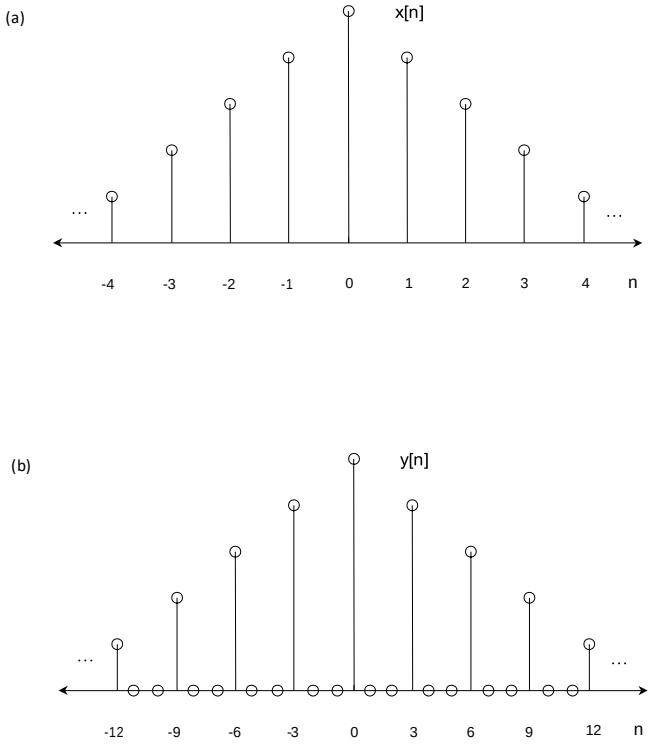


Figure 4. Interpolation of a discrete-time signal by a factor of 3.

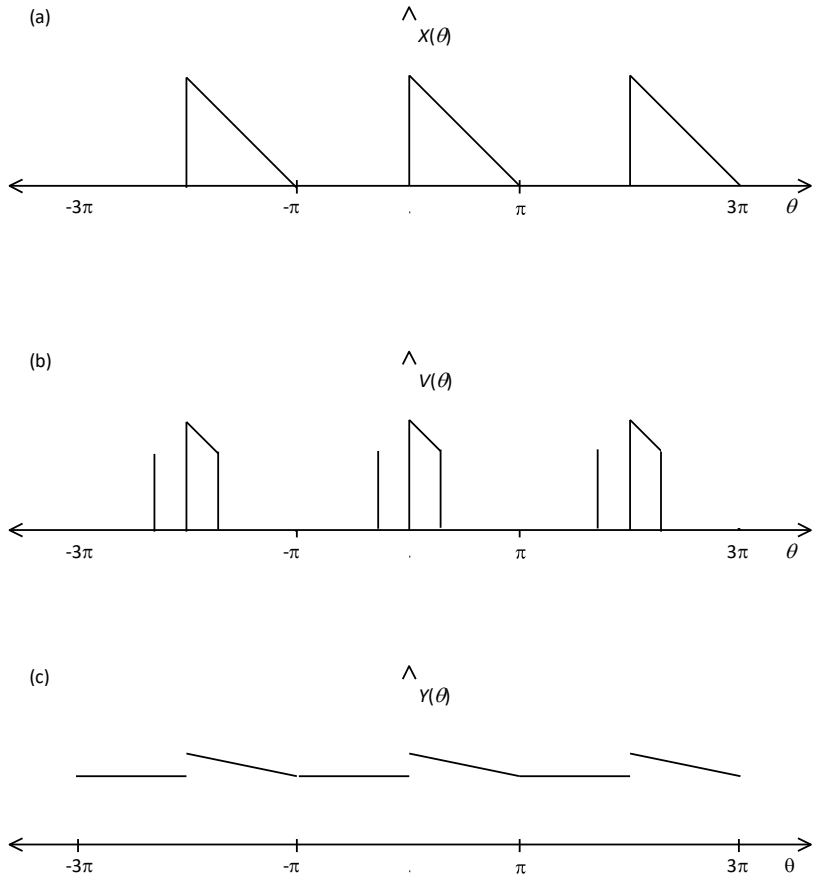
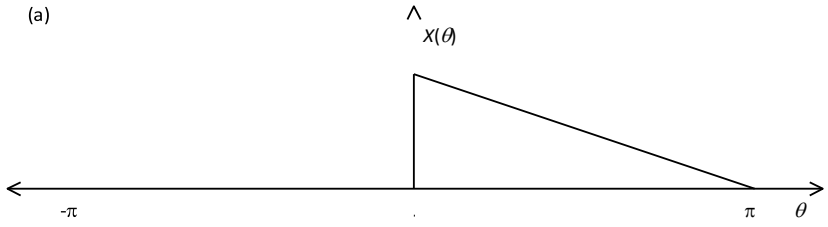


Figure .5: Aliasing caused by decimation; (a) Fourier transform of the original signal; (b) After decimation filtering; (c) Fourier transform of the decimated signal.

In Figure .5 (a) it shows the Fourier transform of the original signal. Part (b) shows the signal after lowpass filtering. In Figure .5 (c), it depicts the expanded spectrum after decimation.

Expansion

The effect of expansion on a signal in the frequency domain is illustrated in Figure .6 below. Part (a) shows the Fourier transform of the original signal; part (b) illustrates the Fourier transform of the signal with zeros added $W(\square)$; and part
(c) shows the Fourier transform of the signal after the interpolation filter. It is clearly visible that the shape of the Fourier transform is compressed by a factor L in the frequency axis and is also repeated L times in the range of $[-\square, \square]$. Despite the compression of the signal in the frequency axis, the shape of the Fourier transform is still preserved, confirming that expansion does not lead to aliasing. These replicas are removed by a digital low-pass filter called an *anti-imaging* filter, as indicated in Figure .3.



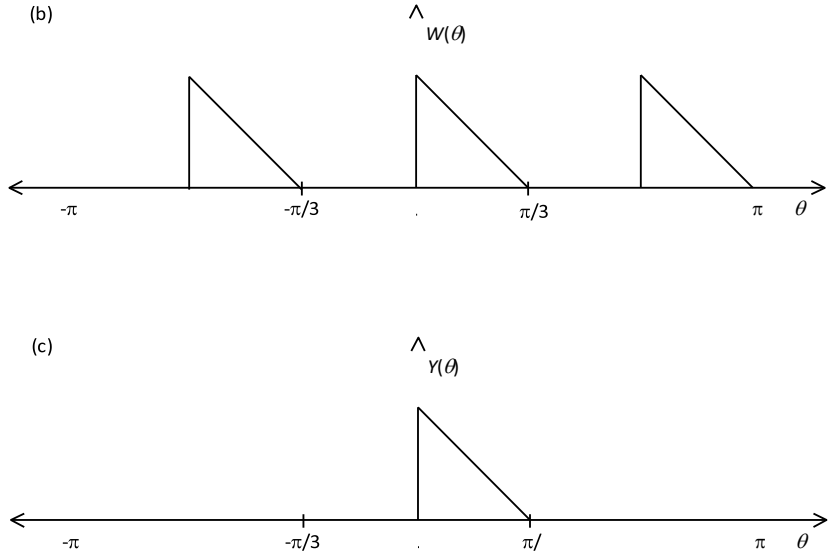


Figure 6. Expansion in the frequency domain of the original signal (a) and the expanded signal (b).

b. Sampling-rate Conversion

A common use of multirate signal processing is for sampling-rate conversion. Suppose a digital signal $x[n]$ is sampled at an interval T_1, and we wish to obtain a signal $y[n]$ sampled at an interval T_2. Then the techniques of decimation and interpolation enable this operation, providing the ratio T_1/T_2 is a rational number i.e. L/M.

Sampling-rate conversion can be accomplished by L-fold expansion, followed by low-pass filtering and then M-fold decimation, as depicted in Figure .7. It is important to emphasis that the interpolation should be performed first and decimation second, to preserve the desired spectral characteristics of $x[n]$. Furthermore by cascading the two in this manner, both of the filters can be combined into one single low-pass filter.

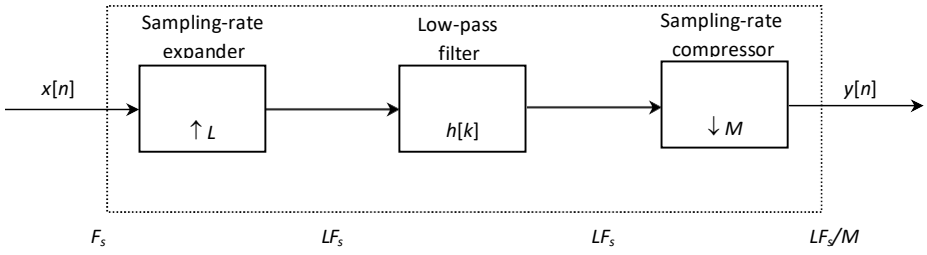


Figure 7: Sampling-rate conversion by expansion, filtering, and decimation.

An example of sampling-rate conversion would take place when data from a CD is transferred onto a DAT. Here the sampling-rate is increased from 44.1 kHz to 48 kHz. To enable this process the non-integer factor has to be approximated by a rational number:

$$\frac{L}{M} = \frac{48}{44.1} = \frac{160}{147} = 1.08844$$

Hence, the sampling-rate conversion is achieved by interpolating by L i.e. from 44.1 kHz to [44.1x160] = 7056 kHz. Then decimating by M i.e. from 7056 kHz to [7056/147] = 48 kHz.

c. Multistage Approach

When the sampling-rate changes are large, it is often better to perform the operation in multiple stages, where $M_i(L_i)$, an integer, is the factor for the stage i.

$$M = M_1 M_2 \ldots M_I \text{ or } L = L_1 L_2 \ldots L_I$$

An example of the multistage approach for decimation is shown in Figure .8. The multistage approach allows a significant relaxation of the anti-alias and anti-imaging filters, with a consequent reduction in the filter complexity. The optimum number of stages is one that leads to the least computational effort in terms of either the multiplications per second (MPS), or the total storage requirement (TSR).

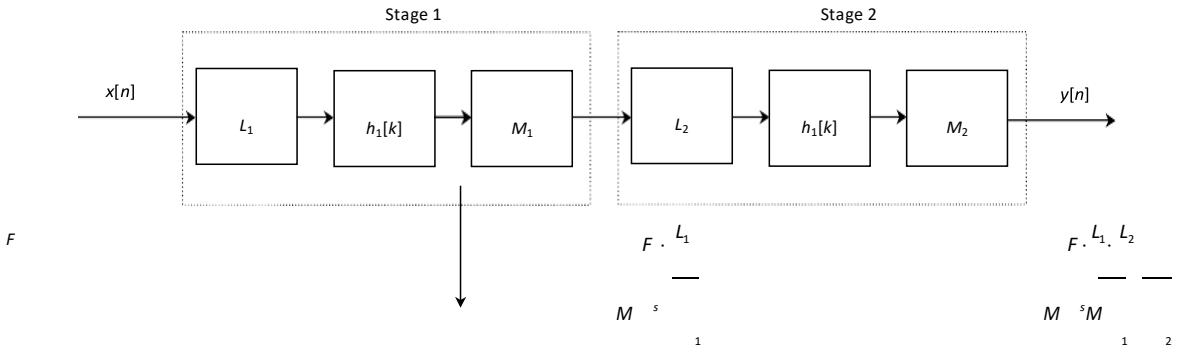


Figure.8: Multistage approach for the decimation process.

d. Polyphase Filters

Potential computational savings can be made within the process of decimation, interpolation, and sampling-rate conversion. Polyphase filters is the name given to certain realizations of multirate filtering operations, which facilitate computational savings in both hardware and software. As an example, the combined low-pass filter in the sampling-rate converter, as illustrated in Figure .7, can be re-drawn as a realization structure. In principle, the simplest realization of the low-pass filter is the direct-form FIR structure, as depicted in Figure .9. However, this type of structure is very inefficient owing to the interpolation process, which introduces (L–1) zeros between consecutive points in the signal. If L is large, then the majority of the signal components fed into the FIR filter are zero. As a result, most of the multiplications and additions are zero i.e. many pointless calculations. Furthermore, the decimation process itself implies that only one out of every M output samples is required at the output of the sampling-rate converter. Consequently, only one out of every M possible values at the output of the filter needs to be computed. This type of structure therefore, leads to much inefficiency during the process of sampling-rate conversion. A more efficient realization structure of the sampling-rate converter uses Polyphase filters, as illustrated in Figure 10. It takes into account that after the interpolation process the signal consists of (L–1) zero coefficients, and the decimation process implies that only one out of every M samples is required at the output of the

converter. To make the scheme more efficient, the low-pass filter in Figure .9 is replaced by a bank of filters arranged in parallel, as illustrated in the efficient realization. The sampling-rate conversion process is undertaken by the multiplexer at the output by selecting every MT/L samples. In this example, the efficient realization is illustrated for a signal which is interpolated by L = 3 and decimated by M = 2 samples.

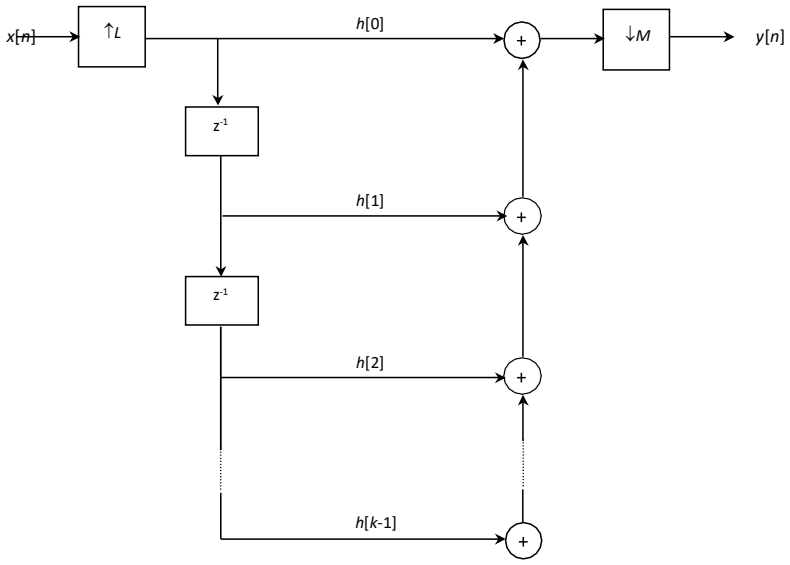


Figure 9. Realisation structure of sampling-rate conversion.

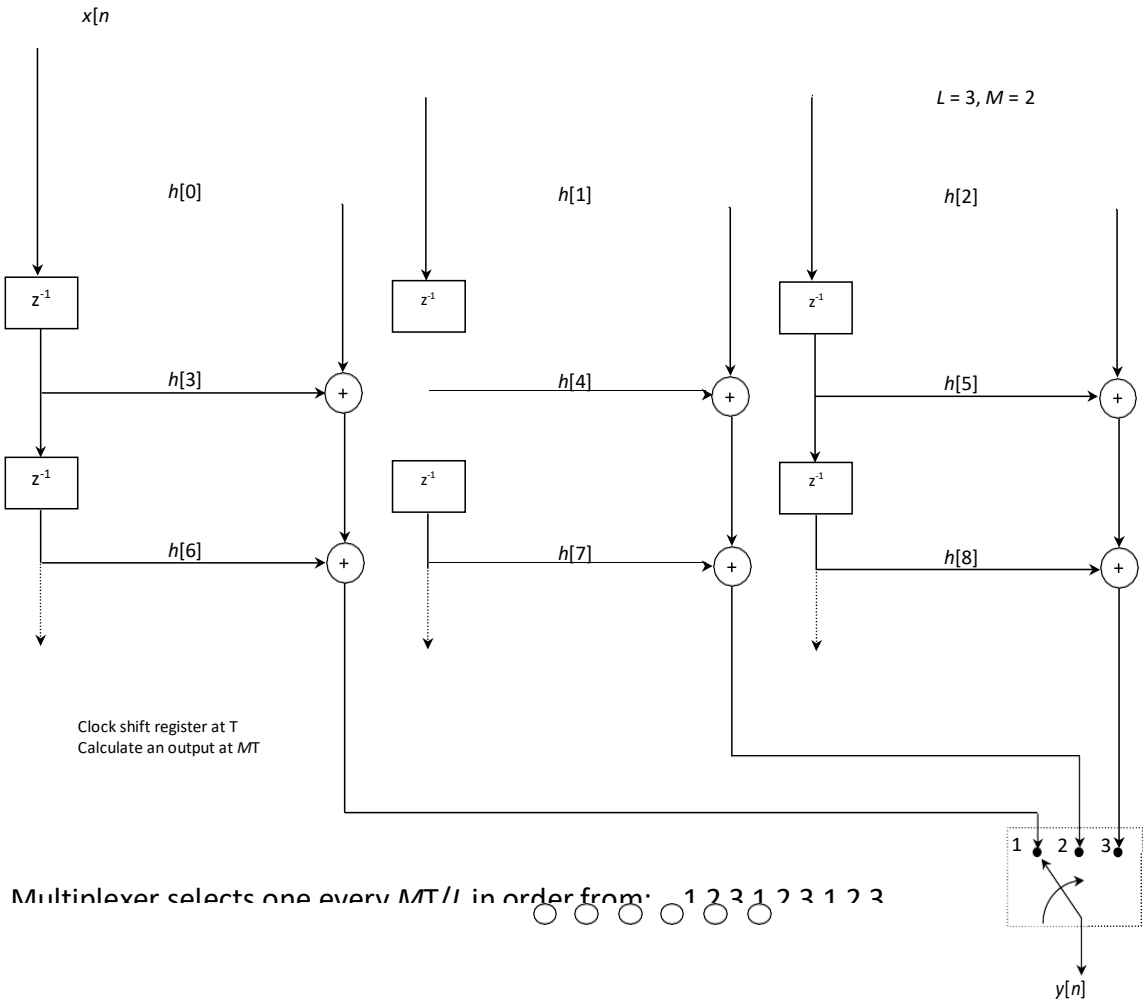


Figure 10. Efficient realisation structure for sampling-rate conversion.

Sub band coding

In signal processing, **sub-band coding** (**SBC**) is any form of transform coding that breaks a signal into a number of different frequency bands, typically by using a fast Fourier transform, and encodes each one independently. This decomposition is often the first step in data compression for audio and video signals. The utility of SBC is perhaps best illustrated with a specific example. When used for audio compression, SBC exploits auditory masking in the auditory system. Human ears are normally sensitive to a wide range of frequencies, but when a sufficiently loud signal is present at one frequency, the ear will not hear weaker signals at nearby frequencies. We say that the louder signal masks the softer ones.

The basic idea of SBC is to enable a data reduction by discarding information about frequencies which are masked. The result differs from the original signal, but if the discarded information is chosen carefully, the difference will not be noticeable, or more importantly, objectionable.

First, a digital filter bank divides the input signal spectrum into some number (e.g., 32) of subbands. The psychoacoustic model looks at the energy in each of these subbands, as well as in the original signal, and computes masking thresholds using psychoacoustic information. Each of the subband samples is quantized and encoded so as to keep the quantization noise below the

dynamically computed masking threshold. The final step is to format all these quantized samples into groups of data called frames, to facilitate eventual playback by a decoder. Decoding is much easier than encoding, since no psychoacoustic model is involved. The frames are unpacked, subband samples are decoded, and a frequency-time mapping reconstructs an output audio signal.

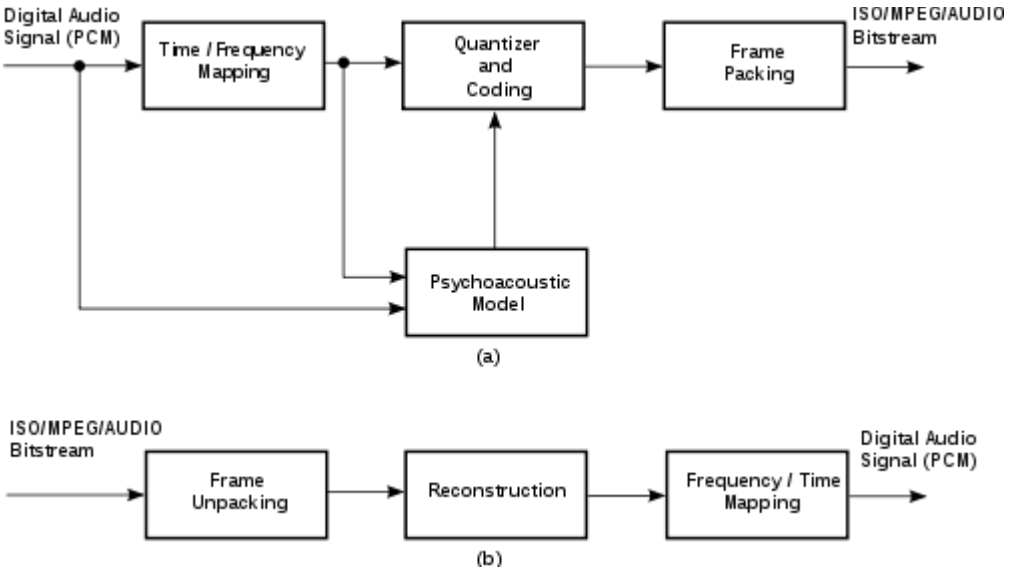


Fig. 11. Sub band coding of audio signal