Chapter 2

INTERPOLATION
FILTERS
2. INTERPOLATION FILTERS

The basic problem caused by the nature of the discrete-time signals is that the value of the signal is not known between the consecutive samples. This problem can be solved by using an interpolation filter. This discrete-time filter structure is used in numerous applications [7] to interpolate new sample values at arbitrary points between the discrete-time input samples.

In this chapter, the function and the input parameters for the interpolation filter are defined. A general hybrid analog/digital model is presented which is then used for analyzing and synthesizing interpolation filters. After that, two realization structures for interpolation filters are discussed with the main emphasis on the Farrow structure. This special filter structure can be used only for polynomial-based interpolation filters.

2.1 Hybrid Analog/Digital Model for Interpolation Filter:

Figure (2.1) shows a simplified block diagram for the interpolation filter, whereas the interpolation process in the time domain is illustrated in Fig.(2.2). The purpose of the overall process is to generate the output samples \( y(l) \) at the given time instants \( t = T(l) \) for \( l = 0,1,2,\ldots \) based on the existing discrete-time input samples. These samples are denoted by \( x(n(l)) \), instead of \( x(n) \), to emphasize their use for producing the output samples \( y(l) \). There are two control parameters for the interpolation filter, namely \( n(l) \) and \( \mu(l) \). The parameter \( n(l) \) determines, in a manner to be described
later, what input samples are taken into consideration when producing the output sample $y(l)$, whereas the parameter $\mu(l)$ determines how to process these samples. Given the time instant $t = T(l)$, these parameters are determined as follows [8]:

$$T(l) = (n(l) + \mu(l)) T_s$$                  \hspace{1cm} (2.1a)

where

$$n(l) = \lfloor T(l)/T_s \rfloor$$                  \hspace{1cm} (2.1b)

and

$$\mu(l) = T(l)/T_s - \lfloor T(l)/T_s \rfloor$$                    \hspace{1cm} (2.1c)

Here $\lfloor T(l)/T_s \rfloor$ stands for the integer part of $T(l)/T_s$, and $T_s = 1/F_s$ is the input sampling interval. In the above, the role of the integer $n(l)$ is to determine the time instant $t = n(l) T_s$ for the discrete-time input sample occurring at or before $t - T(l)$. Simultaneously, it specifies the sampling interval $n(l) T_s \leq t < (n(l) + 1) T_s$, where the $t$th output sample $y(l)$ is located. The parameter $\mu(l) \in [0,1)$, called the fractional interval, is used to exactly determine the time instant of $y(l)$ inside this interval by giving the distance between $t = T(l)$ and $t = n(l) T_s$ as a fraction of $T_s$. It is obviously required that $n(l) \leq n(l+1)$ for $l = 0,1,2,\cdots$. If $n(l) = n(l+1) = \cdots = n(l+r)$, then there are $r+1$ output samples $y(l), y(l+1), \cdots, y(l+r)$ inside the interval $n(l) T_s \leq t < (n(l) + 1) T_s$.

\[1\] A different direction for $\mu(l)$ is used in the case of fractional delay filters in order to have a positive delay [7], [6]. In this case, the time instant for the output sample is given by $T(l) = (n(l) - \mu(l)) T_s$. 

12
After knowing the two control parameters \( n(l) \) and \( \mu(l) \), the role of the interpolation filter is to use the existing discrete-time samples before and after \( x(n(l)) \), and to form a convolution sum with the impulse-response coefficients \( h(k, \mu(l)) \) in such a way that the output sample satisfies \( y(l) = y_o(T(l)) = y_o((n(l) + \mu(l))T_s) \). In the ideal case, the underlying continuous-time signal \( y_o(t) \) shown in Fig. (2.2) is generated, according to the sampling theorem, in the interval \( n(l)T_s \leq t < (n(l) + 1)T_s \) using the sinc interpolation and an infinite number of existing input samples. In practice, a finite number of samples are used and \( h(k, \mu(l)) \) provides an approximation to the ideal response according to some time-domain or frequency-domain criterion.

![Interpolation filter diagram](image)

Fig.(2.1). Interpolation filter with the input signal \( x(n(l)) \) and the interpolated output samples \( y(l) \) for \( l = 0, 1, 2, \ldots \). The input parameters \( n(l) \) and \( \mu(l) \) are used to determine the time instant for the output samples \( y(l) \). Here \( n(l) \) is an integer, and \( \mu(l) \in [0,1) \).
It is obvious that the interpolation filter depicted in Fig.(2.1) corresponds to a linear system having the impulse response denoted by \( h(k, \mu(l)) \). However, the interpolation filter is not a time-invariant system because the impulse response \( h(k, \mu(l)) \) is a function of the fractional interval \( \mu(l) \) which can be changed for each output sample \( y(l) \). This time-varying nature of the interpolation filter makes its analysis and synthesis difficult. One alternative to analyze the frequency-domain behavior of the interpolation filter is to plot the amplitude response as a function of the frequency and fractional interval \( \mu \) [21].

![Diagram of interpolation in the time domain.](image)

Fig.(2.2). Interpolation in the time domain. The input samples \( x(n(l)) \) are denoted by black squares and the output samples \( y(l) \) of the interpolation filter are denoted by circles. The approximating continuous-time signal \( y_a(t) \) (dashed line) is sampled at \( T(l) = (n(l) + \mu(l))T \) to obtain \( y(l) \). Note that \( n(l) \) is the index used only for those input samples immediately preceding the output samples.
A convenient solution to avoid the use of the time-varying impulse response is to utilize a hybrid analog/digital model for the interpolation filter as shown in Fig. (2.3) [5]. In this model, the digital input signal $x(n)$ is first converted using a DAC to the following sequence of weighted and shifted impulses:

$$x_s(t) = \sum_{n} x(n)\delta(t-nT_s) \quad \text{..... (2.2)}$$

where $\delta(t)$ is the analog Dirac delta-function [45]. This sequence is then filtered using an analog anti-imaging filter with the impulse response $h_a(t)$ to obtain the reconstructed signal $y_a(t)$ that can be given by

$$y_a(t) = \int x_s(t)h_a(t)dt = \sum_k x(k) \int \delta(t-kT_s)h_a(t)dt$$

$$= \sum_k x(k)h_a(t-kT_s) \quad \text{..... (2.3)}$$

Fig. (2.3). The hybrid analog/digital model for the interpolation filter.
The interpolated output sample $y(l)$ is obtained by sampling $y_a(t)$ at $t = T(l) = (n(l) + \mu(l))T_s$. If it is assumed that the non-causal anti-imaging filter $h_a(t)$ is zero outside the interval $-NT_s/2 \leq t \leq NT_s/2$, then the interpolated output sample can be expressed as

$$y(l) = y_a(T(l)) = \sum_{k = n(l)/N + 1}^{n(l)/N + N/2} x(k)h_a\left((n(l) + \mu(l) - k)T_s\right)$$

$$= \sum_{k = N/2 + 1}^{N/2} x(k + n(l))h_a\left((\mu(l) - k)T_s\right)$$

(2.4)

The summation in Eq.(2.4) describes the discrete-time convolution, and it gives the relationship between the hybrid analog/digital model and the interpolation filter in Fig.(2.1), that is, the continuous-time impulse response of the anti-imaging filter $h_a(t)$ and the discrete-time impulse response of the interpolation filter $h(k, \mu(l))$ are related to each other as follows:

$$h(k, \mu(l)) = h_a((\mu(l) + k)T_s)$$

(2.5)

for $k = -N/2, -N/2 + 1, \cdots, N/2 - 1$. The result of Eq.(2.5) suggests that discrete-time interpolation filters have an underlying continuous-time impulse response $h_a(t)$ which can be used to analyze and synthesize these filters.

Note that the first part of the hybrid analog/digital model up to the resampling of $y_a(t)$ is a linear time-invariant system, which means that the
reconstruction of $y_n(l)$ is independent of the fractional interval $\mu(l)$. This makes the analysis of the interpolation filters easier. Another advantage of the model is that it turns the interpolation from the time-domain problem to the frequency-domain filtering problem. This allows us to utilize the frequency-domain information of the signal when analyzing and synthesizing interpolation filters. This is usually a better approach in the DSP applications than the conventional time-domain error analysis which is used in mathematics [3].

2.2 FIR Filters Having a Fractional Delay:

In some applications the interpolation filter is used to generate a constant fractional delay of $\mu$. In these cases the output signal $y(l)$ is just a delayed version of the input signal $x(n)$, and thus $n(l) = 0, 1, 2, \ldots$ and $\mu(l) = \mu$. Therefore, in these applications, the interpolation filter of Fig (2.1) can be implemented using an FIR filter having the desired fractional delay $\mu$. The impulse response of this filter is denoted by $h(k, \mu)$, and its frequency response in the non-causal case is given by

$$H(e^{j\omega}, \mu) = \sum_{k=\lfloor N/2 \rfloor}^{N/2 + 1} h(k, \mu)e^{-j\omega k} \quad \ldots \quad (2.6)$$

where $N$ is the length of the filter. In the ideal case, the FD FIR filter is an all-pass filter and its frequency response is given by

$$H_{id}(e^{j\omega}, \mu) = e^{-j\omega \mu} \quad \ldots \quad (2.7)$$
The optimization of the FD FIR filters can be done by using, for example, the minimax or the least-mean-square (LMS) criteria. The optimization problems in these cases can be stated as follows [7]:

**Minimax optimization problem**: Given the length of the filter \( N \), the fractional delay \( \mu \), and the passband edge \( \zeta_p = 2\pi f_p / f_s < \pi \), find the filter coefficients \( h(k, \mu) \) to minimize

\[
\delta_m = \max_{\omega_{<p}} | H(e^{j\omega}, \mu) - H_m(e^{j\omega}, \mu) | \quad \ldots (2.8)
\]

**Least-mean-square optimization problem**: Given the length of the filter \( N \), the fractional delay \( \mu \), and the passband edge \( \zeta_p \), find the filter coefficients \( h(k, \mu) \) to minimize

\[
\delta_L = \int_0^{\frac{\pi}{f_s}} | H(e^{j\omega}, \mu) - H_m(e^{j\omega}, \mu) |^2 d\omega \quad \ldots (2.9)
\]

In the practical implementation, the FD FIR filters are causal, and therefore, the total delay is \( D = D_m + \mu \), where \( D_m = N/2 - 1 \) is the integer delay and \( N \) is the length of the filter.
As an example, Figs.(2.4) and (2.5) illustrate the amplitude and phase delay responses for five different FD FIR filters. The length of these filters is $N = 10$, the passband edge is $f_p = 0.375F_s$, and the values of the fractional delay are $\mu = 0.1, 0.2, 0.3, 0.4, \text{ and } 0.5$. The integer delay is $D_m = 4$. These filters have been designed using the minimax method introduced by Oetken [38].

Fig.(2.4). The amplitude responses for the Example minimax FD FIR Filters of length 10.
2.3 Coefficient Memory Implementation for Interpolation Filters:

FD FIR filters can also be used to implement the interpolation filter of Fig. (2.1) in the general case where $\mu(I)$ is adjustable. This can be done by using the so-called coefficient memory implementation, where $K$ different FD FIR filters are designed having the fractional delays of $\mu_k = k / K$ for $k = 0, 1, \ldots, K - 1$. Here $K$ is a large integer which determines the resolution of $\mu(I)$. The coefficients of these filters are then stored into a memory, and, during the computation, the desired delay can be
obtained by choosing the corresponding filter coefficients from the memory [18]. Since the impulse response of the FD FIR filter having the delay of $\mu$ is the time reversed version of the filter having the delay of $1 - \mu$, the total number of filter coefficients in the memory is $NK/2$, where $N$ is the length of the filters [7].

One drawback of this implementation is the need for a large memory. Since the size of this memory is directly proportional to $K$, one critical question in the design process is: What is the smallest increment $1/K$ in the fractional interval needed in the application?

The theoretical performance of the interpolation filter based on the coefficient memory implementation can be analyzed when $K \to \infty$ by using the hybrid analog/digital model of Fig.(2.3). The underlying continuous-time impulse response $h_c(t)$ cannot be expressed in a closed form, but it can be approximated by using the impulse responses of the FD FIR filters having the delay values of $\mu_k = k/K$ for $k = 0, 1, \ldots, K - 1$. The impulse responses of these FD filters are arranged as overlapping polyphase components. The resulting polyphase-type impulse response then approaches the continuous-time response $h_c(t)$ when $K \to \infty$.

This is illustrated in Fig.(2.6) for $K = 4$. In this figure, the discrete-time impulse response consists of the overlapping impulse responses of the FD FIR filters having the delay values of $\mu = 0, 0.25, 0.5$, and $0.75$. The solid line illustrates the
Fig.(2.6). The impulse responses $h(k, \mu)$ for the FD filters optimized in $L_2$ sense [7] and the underlying continuous-time impulse response $h_n(t)$. In order to avoid a special treatment of FD filters, the $h(k, \mu)$'s have been reversed.

continuous-time response $h_n(t)$ that is obtained when $K \to \infty$. Figure (2.7) shows the corresponding magnitude response $|H_n(j\Omega)| = |H_n(j2\pi f)|$. In this example, the coefficients for the FD FIR filters are optimized in the least-mean-square sense [7]. The length of these filters is $N = 10$ and the passband edge is $f_p = 0.375f_c$. Note that the
The magnitude response of the continuous-time system having the impulse response $h_n(t)$ shown in Fig. (2.6) don’t care bands $^2$, which are centered at $(k + 0.5)F_s$ for $k = 1, 2, \cdots$ in Fig.(2.7) do not have an effect on the performance of the interpolation filter provided that the highest baseband frequency component of the input signal $x(n)$ is at most $0.375F_s$.

$^2$ The don’t care bands consist of the frequency bands that are not included in the optimization.
2.4 Interpolation Filters with a Continuous-Time Impulse Response:

The use of a large coefficient memory in the implementation of the interpolation filters can be avoided if the continuous-time impulse response $h_a(t)$ can be expressed in a closed form. In this case there is no need to store the filter coefficients $h(k, \mu(t))$ in the memory because they can be obtained from $h_a(t)$ according to Eq.(2.5). This idea is utilized in the polynomial-based interpolation filters where $h_a(t)$ is expressed by means of polynomials (to be discussed later). Other methods include the sinc and the trigonometric interpolation. In the former method, the impulse response $h_a(t)$ is given by

$$h_a(t) = \sin c(t/T_a) \quad \text{for} \quad -NT_a/2 \leq t \leq NT_a/2 \quad \ldots \quad (2.10)$$

where $N$ is the length of the filter [7],[43]. The direct truncation is not usually used because of the Gibbs phenomenon which can be avoided by weighting the truncated sinc with some window function [46].

Figure (2.8) shows the continuous-time impulse response $h_a(t)$ for the windowed sinc interpolation filter along with the discrete-time impulse response $h(k, \mu)$ for $\mu = 0.25$ and 0.5. The magnitude response $|H_a(j2\pi f)|$ is shown in Fig.(2.9). The length of the filter is 10 (i.e., $10T_a$) and the Hamming window is used in weighting.
In the trigonometric interpolation, a trigonometric polynomial of degree $M = N - 1$ is fitted to $N$ consecutive sample values [3]. This polynomial is then used to calculate the value of the output sample $y(t)$. In [42], Fu and Willson presented an efficient realization structure for the trigonometric interpolation. In the same paper, it was shown that the impulse response of the interpolation filter based on the trigonometric interpolation can be given by

$$h_s(t) = \frac{1}{N} \sin(\pi t/T_s) \cos(\pi t/(N T_s)) \quad \text{for } -NT_s/2 \leq t \leq NT_s/2 \quad \ldots \quad (2.11)$$
The drawback of the trigonometric interpolation is low attenuation level near the transition band, which can be seen from the example depicted in Fig.(2.11). The corresponding impulse response is shown in Fig.(2.10). In these figures, the degree of interpolation is $M = 9$. For the trigonometric interpolation, the length of the filter is always $N = M + 1$, and therefore the filter length in this example is $N = 10$.

**Fig.(2.9).** The magnitude response $|H_x(j2\pi f)|$ for the windowed sinc interpolation filter
2.5 Interpolation Filters with a Polynomial-Based Impulse Response:

It was mentioned in the previous section that the interpolation filters can be implemented without a large coefficient memory if \( h_a(t) \) can be expressed in a closed form by means of some mathematical function like the sinc function. However, from the implementation point of view, the evaluation of the value of the sinc function is not an easy problem in the applications implemented using a signal processor or a VLSI circuit. The sinc function can be approximated by using a power-series expansion. Another solution is to store the values of the sinc function in to a lookup table but this leads back to the coefficient memory implementation.

![Impulse response](image)

Fig.(2.10). The impulse response \( h_a(t) \) for the trigonometric interpolation filter of degree 9.
A more straightforward way to avoid the coefficient memory implementation is to use polynomial-based interpolation filters [21]. For these filters, the underlying continuous-time impulse response $h_c(t)$ is expressed in each interval of length $T_s$ by means of a polynomial as follows:

$$h_c((\mu(l) - k)T_s) = \sum_{m=0}^{M} c_m(k)(\mu(l))^m$$  \hspace{1cm} (2.12)

for $k = -N/2 + 1, -N/2 + 2, \cdots, N/2$, and for $\mu(l) \in [0,1)$. Here the $c_m(k)$'s denote the polynomial coefficients for $h_c((\mu(l) - k)T_s)$ and $M$ is the degree of polynomials.

Fig.(2.11). The magnitude response $|H_c(j2\pi f)|$ for the trigonometric interpolation filter of degree 9.
In order to derive an efficient realization structure for polynomial-based interpolation filters, Eq.(2.12) is substituted into Eq.(2.4) giving the following expression for the \( l \)th interpolated output sample:

\[
y(l) = \sum_{k=-N/2}^{N/2} x(k+n(l)) \sum_{m=0}^{M} c_m(k) (\mu(l))^m = \sum_{m=0}^{M} v_m(n(l)) (\mu(l))^m \quad \ldots \quad (2.13)
\]

where

\[
v_m(n(l)) = \sum_{k=-N/2}^{N/2} x(k+n(l)) c_m(k) \quad \ldots \quad (2.14)
\]

As Eqs.(2.13) and (2.14) suggest, the polynomial-based interpolation filter can be implemented by filtering the input signal \( x(n(l)) \) with \( M+1 \) parallel FIR filters of length \( N \) having the coefficient values of \( c_m(k) \) to obtain the output samples \( v_m(n(l)) \). In the causal case, the transfer functions of these FIR filters are given by

\[
C_m(z) = \sum_{k=0}^{N-1} c_m(-k+N/2) z^{-k} \quad \ldots \quad (2.15)
\]

for \( m = 0,1,\ldots,M \). The interpolated sample value \( y(l) \) is then obtained by multiplying the output samples \( v_m(n(l)) \) by \( (\mu(l))^m \) for \( m = 0,1,\ldots,M \). This realization structure for the polynomial-based interpolation filters, the 
Farrow structure [21], is shown in Fig.(2.12). In this structure, the FIR filters are usually non-linear-phase filters provided that \( h_m(t) \) is symmetrical and \( \mu(t) \in [0,1) \).
It is also possible to express $h_n(t)$ in Eq (2.12) as a function of $2\mu(l) - 1$ for $\mu(l) \in [0,1)$. This results in a modified Farrow structure [P4]. If $h_n(t)$ is symmetrical, then the FIR filters in the modified Farrow structure have either a symmetric or anti-symmetric impulse responses of even length, that is, they are linear-phase Type II and Type IV filters [P4]. This property of the FIR filters in the modified Farrow structure is utilized in [P8] to express the frequency response of the corresponding anti-imaging filter $H_n(j2\pi f)$ by means of the zero-phase frequency responses of these linear-phase FIR filters and certain weighting functions. This arrangement makes the filter optimization very straightforward. The coefficient symmetry can also be exploited in reducing the number of multipliers required in the overall implementation.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig.png}
\caption{The Farrow structure.}
\end{figure}
The main advantage of the Farrow structure is that the filter coefficients are fixed, and the only changeable parameter is the fractional interval $\mu(l)$. Besides this, the control of $\mu(l)$ is easier during the computation than in the coefficient memory implementation and the resolution of $\mu(l)$ is limited only by the precision of the arithmetic and not by the size of the memory. These characteristics of the Farrow structure make it a very attractive structure to be implemented using a VLSI circuit or a signal processor.