INTRODUCTION

We have discussed so far the decimation and interpolation where the sampling rate conversion factor is an integer. However, the need for a non-integer sampling rate conversion appears when the two systems operating at different sampling rates have to be connected, or when there is a need to convert the sampling rate of the recorded data into another sampling rate for further processing or reproduction. Such applications are very common in telecommunications, digital audio, multimedia and others.

In this chapter, we consider the sampling rate conversion by a rational factor, called sometimes a fractional sampling rate conversion. We use MATLAB functions from the Signal Processing and Filter Design Toolbox to demonstrate the fractional sampling rate conversion. We present the technique for constructing efficient fractional sampling rate converters based on FIR filters and the polyphase decomposition. In the sequel, we consider the sampling rate alteration with an arbitrary conversion factor. We present the polynomial-based approximation of the impulse response of a hybrid analog/digital model, and the implementation based on the Farrow structure. We also consider the fractional-delay filter problem. This chapter concludes with MATLAB exercises for individual study.

SAMPLING RATE CONVERSION BY A RATIONAL FACTOR

The change of the sampling frequency by a rational factor \( L/M \), sometimes called the fractional sampling rate alteration or resampling, can be achieved by increasing the sampling frequency by \( L \) first, and then decreasing by \( M \). Hence, the sampling rate conversion by \( L/M \) is achieved by a cascading factor-of-\( L \) interpolator and a factor-of-\( M \) decimator as indicated in Figure 6.1(a). Here, factors \( L \) and \( M \) are positive relatively prime integers, i.e. there is no common integer between \( L \) and \( M \). In the implementation
Sampling Rate Conversion by a Fractional Factor

Figure 6.1. Sampling rate conversion by $L/M$. (a) General implementation scheme. (b) Efficient implementation

scheme of Figure 6.1(a), the original signal $\{x[n]\}$ is up-sampled-by-$L$ and then filtered by the lowpass interpolation filter $H_I(z)$. The interpolated signal $\{w[r]\}$ is filtered with the lowpass antialiasing filter $H_D(z)$, and then down-sampled-by-$M$. The sampling rate of the output signal $\{y[m]\}$ is $L/M$ times the sampling rate of the original signal $\{x[n]\}$. Since the interpolation filter $H_I(z)$ and the decimation filter $H_D(z)$ operate at the same sampling rate, they can be replaced by the single lowpass filter $H(z)$ as indicated in Figure 6.1 (b). The lowpass filter $H(z)$ should be designed to eliminate imaging caused by the up-sampling, and to avoid aliasing produced in down-sampling. With the properly designed filter $H(z)$, the fractional sampling rate conversion can be implemented by using the computationally efficient structure of Figure 6.1 (b).

The role of filter $H(z)$ in the efficient fractional sampling-rate converter of Figure 6.1(b) is twofold: it acts as the antiimaging filter $H_I(z)$, and also as the antialiasing filter $H_D(z)$. For the adequate removal of images, the stopband edge frequency of the low-pass filter $H(z)$ must be below $\pi/L$, and avoiding of aliasing requires the stopband edge below $\pi/M$. Therefore, the low-pass filter $H(z)$ in the implementation scheme of Figure 6.1 (b) has the stopband edge frequency at $\omega_s$, which is given by

$$\omega_s = \min \left( \frac{\pi}{L}, \frac{\pi}{M} \right). \quad (6.1)$$

Choosing $\omega_s$ according to (6.1) ensures the elimination of imaging which appears in interpolation, and at the same time ensures the suppression of aliasing that may be caused by decimation. Hence, the ideal specifications for the magnitude response of $H(z)$ are given by

$$|H(e^{j\omega})| = \begin{cases} 
L, & |\omega| \leq \min \left( \frac{\pi}{L}, \frac{\pi}{M} \right) \\
0, & \text{otherwise}
\end{cases} \quad (6.2)$$

It is important to observe that for a large $L$ or $M$, filters with very narrow passbands are requested.

In MATLAB, there are several functions that perform the fractional sampling rate alteration according to the implementation scheme of Figure 6.1(b). The simplest for use is the function resample from Signal Processing Toolbox. For the given original signal stored in vector $x$, and the sampling rate conversion factor $L/M$, the function resample returns the resampled signal $y$,

$$y = \text{resample}(x,L,M); \quad \% \text{Fractional sampling rate alteration by } L/M$$

Here, the up-sampling factor $L$, and the down-sampling factor $M$ are integers. For the description of function resample including the related options, see the Signal Processing Toolbox User’s Guide. In the following, we illustrate the fractional sampling rate alteration on the example sinusoidal sequence.
Related Content

Goal Modelling for Security Problem Matching and Pattern Enforcement
[www.irma-international.org/article/goal-modelling-for-security-problem-matching-and-pattern-enforcement/201215/](www.irma-international.org/article/goal-modelling-for-security-problem-matching-and-pattern-enforcement/201215/)

Fault-Prone Module Prediction Approaches Using Identifiers in Source Code

A Survey to Nature Inspired Soft Computing
[www.irma-international.org/article/a-survey-to-nature-inspired-soft-computing/199006/](www.irma-international.org/article/a-survey-to-nature-inspired-soft-computing/199006/)

Toward an Interdisciplinary Engineering and Management of Complex IT-Intensive Organizational Systems: A Systems View
[www.irma-international.org/chapter/toward-interdisciplinary-engineering-management-complex/38170/](www.irma-international.org/chapter/toward-interdisciplinary-engineering-management-complex/38170/)

RT-Llama: Providing Middleware Support for Real-Time SOA
[www.irma-international.org/article/llama-providing-middleware-support-real/39099/](www.irma-international.org/article/llama-providing-middleware-support-real/39099/)