Multirate Structures for Sampling Rate Conversion

- From the sampling theorem it is known that the sampling rate of a critically sampled discrete-time signal with a spectrum occupying the full Nyquist range cannot be reduced any further since such a reduction will introduce aliasing
- Hence, the bandwidth of a critically sampled signal must be reduced by lowpass filtering before its sampling rate is reduced by a down-sampler

1

Multirate Structures for Sampling Rate Conversion

- Likewise, the zero-valued samples introduced by an up-sampler must be interpolated to more appropriate values for an effective sampling rate increase
- We shall show shortly that this interpolation can be achieved simply by digital lowpass filtering

2

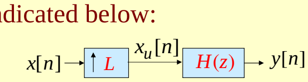
Multirate Structures for Sampling Rate Conversion

- Since a fractional-rate sampling rate converter with a rational conversion factor can be realized by cascading an interpolator with a decimator, filters are also needed in the design of such multirate systems

3

Basic Structures

- Since up-sampling by an integer factor L causes periodic repetition of the basic spectrum, the basic interpolator structure for integer-valued sampling rate increase consists of an up-sampler followed by a low-pass filter $H(z)$ with a cutoff at π/L as indicated below:



4

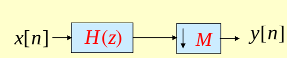
Basic Structures

- The lowpass filter $H(z)$, called the interpolation filter, removes the unwanted images in the spectra of the up-sampled signal $x_u[n]$
- On the other hand, down-sampling by an integer factor M may result in aliasing

5

Basic Structures

- Hence, the basic decimator structure for integer-valued sampling rate decrease consists of a lowpss filter $H(z)$ with a cutoff at π/M, followed by the down-sampler as shown below



6

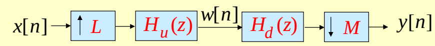
Basic Structures

- Here, the lowpass filter $H(z)$, called the decimation filter, bandlimits the input signal $x[n]$ to $|\omega| < \pi/M$ prior to down-sampling, to ensure no aliasing
- It can be shown that the transpose of a factor-of-M decimator is a factor-of-M interpolator

7

Basic Structures

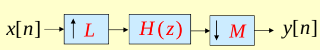
- A fractional change in the sampling rate by a rational factor L/M can be achieved by cascading a factor-of-L interpolator with a factor-of-M decimator
- The interpolator must precede the decimator as shown below to ensure that the baseband of $w[n]$ is greater than or equal to that of $x[n]$ or $y[n]$



8

Basic Structures

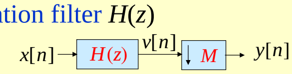
- As both the interpolation filter $H_u(z)$ and the decimation filter $H_d(z)$ operate at the same sampling rate, they can be replaced with a single filter designed to avoid aliasing that may be caused by down-sampling and eliminate images resulting from up-sampling



9

Input-Output Relation of the Decimator

- For the decimator structure shown below, let $h[n]$ denote the impulse response of the decimation filter $H(z)$



- Then

$$v[n] = \sum_{\ell=-\infty}^{\infty} h[n-\ell]x[\ell]$$

and

$$y[n] = v[Mn]$$

10

Input-Output Relation of the Decimator

- Combining the last two equations we arrive at the desired input-output relation of the decimator given by

$$y[n] = \sum_{\ell=-\infty}^{\infty} h[Mn-\ell]x[\ell]$$

- In the z-domain, the input-output relation of the decimation filter is given by

$$V(z) = H(z)X(z)$$

11

Input-Output Relation of the Decimator

- Now the input-output relation of the dowm-sampler is given by

$$Y(z) = \frac{1}{M}\sum_{k=0}^{M-1} V(z^{1/M}W_M^{-k})$$

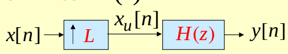
- Combining the last two equations we arrive at the input-output relation of the decimator as

$$Y(z) = \frac{1}{M}\sum_{k=0}^{M-1} H(z^{1/M}W_M^{-k})X(z^{1/M}W_M^{-k})$$

12

Input-Output Relation of the Interpolator

- For the interpolator structure shown below, let $h[n]$ denote the impulse response of the decimation filter $H(z)$



- Then

$$y[n] = \sum_{\ell=-\infty}^{\infty} h[n-\ell] x_u[\ell]$$

and

$$x_u[Lm] = x[m], \quad m = 0, \pm 1, \pm 2, \ldots$$

13

Input-Output Relation of the Interpolator

- Combining the last two equations and making a change of a variable, we arrive at the desired time-domain input-output relation of the interpolator as

$$y[n] = \sum_{m=-\infty}^{\infty} h[n-Lm] x[m]$$

- In the z-domain, the input-output relation of the interpolator is thus given by

$$Y(z) = H(z) X(z^L)$$

14

Input-Output Relation of the Fractional-Rate Converter

- Here, in the time-domain the input-output relation is given by

$$y[n] = \sum_{m=-\infty}^{\infty} h[Mn-Lm] x[m]$$

- In the z-domain it is given by

$$Y(z) = \frac{1}{M} \sum_{k=0}^{M-1} H(z^{1/M} W_M^{-k}) X(z^{L/M} W_M^{-kL})$$

15

Interpolation Filter Specifications

- Assume $x[n]$ has been obtained by sampling a continuous-time signal $x_a(t)$ at the Nyquist rate
- If $X_a(j\Omega)$ and $X(e^{j\omega})$ denote the Fourier transforms of $x_a(t)$ and $x[n]$, respectively, then it can be shown

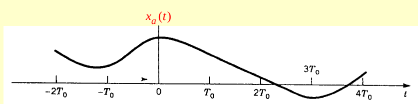
$$X(e^{j\omega}) = \frac{1}{T_o} \sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T_o}\right)$$

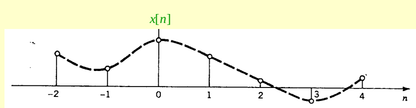
16

- where T_o is the sampling period

Interpolation Filter Specifications

- Figures below show $x_a(t)$ and $x[n]$ obtained by sampling $x_a(t)$ at the Nyquist rate

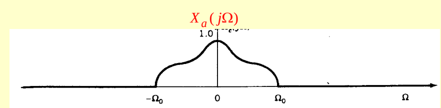


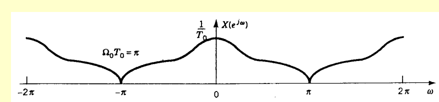


17

Interpolation Filter Specifications

- Figures below show the Fourier transforms of $x_a(t)$ and $x[n]$





18

Interpolation Filter Specifications

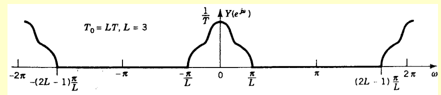
- Since the sampling is being performed at the Nyquist rate, there is no overlap between the shifted spectras of $X(j\omega/T_o)$
- If we instead sample $x_a(t)$ at a much higher rate $T = T_o/L$ yielding $y[n]$, its Fourier transform $Y(e^{j\omega})$ is related to $X_a(j\Omega)$ through

$$Y(e^{j\omega}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T}\right) = \frac{L}{T_o}\sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T_o/L}\right)$$

19

Interpolation Filter Specifications

- Figure below show the Fourier transform of $y[n]$

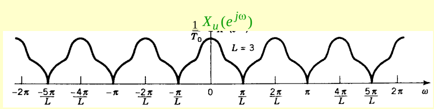


20

Interpolation Filter Specifications

- On the other hand, if we pass $x[n]$ through a factor-of-L up-sampler generating $x_u[n]$, the relation between the Fourier transforms of $x[n]$ and $x_u[n]$ are given by

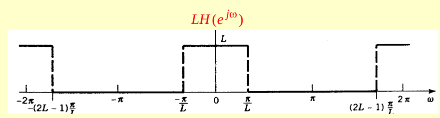
$$X_u(e^{j\omega}) = X(e^{j\omega L})$$



21

Interpolation Filter Specifications

- It therefore follows that if $x_u[n]$ is passed through an ideal lowpass filter $H(z)$ with a cutoff at π/L and a gain of L, the output of the filter will be precisely $y[n]$



22

Interpolation Filter Specifications

- In practice, a transition band is provided to ensure the realizability and stability of the lowpass interpolation filter $H(z)$
- Hence, the desired lowpass filter should have a stopband edge at $\omega_s = \pi/L$ and a passband edge ω_p close to ω_s to reduce the distortion of the spectrum of $x[n]$

23

Interpolation Filter Specifications

- If ω_c is the highest frequency that needs to be preserved in $x[n]$, then

$$\omega_p = \omega_c/L$$

- Summarizing the specifications of the lowpass interpolation filter are thus given by

$$\left|H(e^{j\omega})\right| = \begin{cases} L, & |\omega| \le \omega_c/L \\ 0, & \pi/L \le |\omega| \le \pi \end{cases}$$

24

Decimation Filter Specifications

- In a similar manner, we can develop the specifications for the lowpass decimation filter that are given by

$$|H(e^{j\omega})| = \begin{cases} 1, & |\omega| \le \omega_c / M \\ 0, & \pi / M \le |\omega| \le \pi \end{cases}$$

25

Filter Design Methods

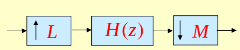
- The design of the filter $H(z)$ is a standard IIR or FIR lowpass filter design problem
- Any one of the techniques outlined in Chapter 7 can be applied for the design of these lowpass filters

26

Filters for Fractional Sampling Rate Alteration

- For the fractional sampling rate structure shown below, the lowpass filter $H(z)$ has a stopband edge frequency given by

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



27

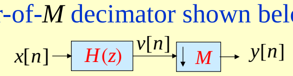
Computational Requirements

- The lowpass decimation or interpolation filter can be designed either as an FIR or an IIR digital filter
- In the case of single-rate digital signal processing, IIR digital filters are, in general, computationally more efficient than equivalent FIR digital filters, and are therefore preferred where computational cost needs to be minimized

28

Computational Requirements

- This issue is not quite the same in the case of multirate digital signal processing
- To illustrate this point further, consider the factor-of-M decimator shown below



- If the decimation filter $H(z)$ is an FIR filter of length N implemented in a direct form, then

$$v[n] = \sum_{m=0}^{N-1} h[m]x[n-m]$$

29

Computational Requirements

- Now, the down-sampler keeps only every M-th sample of $v[n]$ at its output
- Hence, it is sufficient to compute $v[n]$ only for values of n that are multiples of M and skip the computations of in-between samples
- This leads to a factor of M savings in the computational complexity

30

Computational Requirements

- Now assume $H(z)$ to be an IIR filter of order K with a transfer function

$$\frac{V(z)}{X(z)} = H(z) = \frac{P(z)}{D(z)}$$

where

$$P(z) = \sum_{n=0}^{K} p_n z^{-n}$$

$$D(z) = 1 + \sum_{n=1}^{K} d_n z^{-n}$$

31

Computational Requirements

- Its direct form implementation is given by

$$w[n] = -d_1 w[n-1] - d_2 w[n-2] - \cdots - d_K w[n-K] + x[n]$$

$$v[n] = p_0 w[n] + p_1 w[n-1] + \cdots + p_K w[n-K]$$

- Since $v[n]$ is being down-sampled, it is sufficient to compute $v[n]$ only for values of n that are integer multiples of M

32

Computational Requirements

- However, the intermediate signal $w[n]$ must be computed for all values of n
- For example, in the computation of

$$v[M] = p_0 w[M] + p_1 w[M-1] + \cdots + p_K w[M-K]$$

K+1 successive values of $w[n]$ are still required

- As a result, the savings in the computation in this case is going to be less than a factor of M

33

Computational Requirements

- Example - We compare the computational complexity of various implementations of a factor-of-M decimator
- Let the sampling frequency be F_T
- Then the number of multiplications per second, to be denoted as $\mathcal{R}_M$, are as follows for various computational schemes

34

Computational Requirements

- FIR $H(z)$ of length N :

$$\mathcal{R}_{M,FIR} = N \times F_T$$

- FIR $H(z)$ of length N followed by a down-sampler:

$$\mathcal{R}_{M,FIR-DEC} = N \times F_T / M$$

- IIR $H(z)$ of order K:

$$\mathcal{R}_{M,IIR} = (2K+1) \times F_T$$

- IIR $H(z)$ of order K followed by a down-sampler :

$$\mathcal{R}_{M,IIR-DEC} = K \times F_T + (K+1) \times F_T / M$$

35

Computational Requirements

- In the FIR case, savings in computations is by a factor of M
- In the IIR case, savings in computations is by a factor of $M(2K+1)/[(M+1)K+1]$, which is not significant for large K
- For $M = 10$ and $K = 9$, the savings is only by a factor of 1.9
- There are certain cases where the IIR filter can be computationally more efficient

36

Computational Requirements

- For the case of interpolator design, very similar arguments hold
- If $H(z)$ is an FIR interpolation filter, then the computational savings is by a factor of L (since $v[n]$ has $L-1$ zeros between its consecutive nonzero samples)
- On the other hand, computational savings is significantly less with IIR filters

37

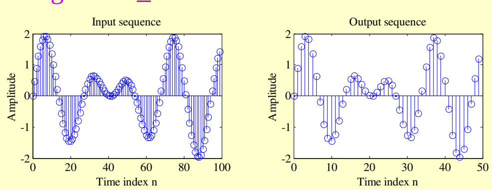
Sampling Rate Alteration Using MATLAB

- The function `decimate` can be employed to reduce the sampling rate of an input signal vector `x` by an integer factor `M` to generate the output signal vector `y`
- The decimation of a sequence by a factor of M can be obtained using Program 10_5 which employs the function `decimate`

38

Sampling Rate Alteration Using MATLAB

- Example - The input and output plots of a factor-of-2 decimator designed using the Program 13_5 are shown below



39

Sampling Rate Alteration Using MATLAB

- The function `interp` can be employed to increase the sampling rate of an input signal `x` by an integer factor `L` generating the output vector `y`
- The lowpass filter designed by the M-file is a symmetric FIR filter

40

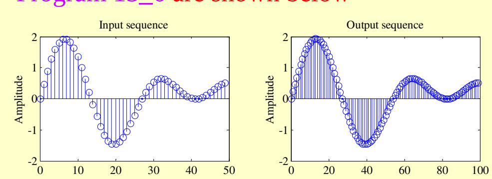
Sampling Rate Alteration Using MATLAB

- The filter allows the original input samples to appear as is in the output and finds the missing samples by minimizing the mean-square errors between these samples and their ideal values
- The interpolation of a sequence `x` by a factor of `L` can be obtained using the Program 13_6 which employs the function `interp`

41

Sampling Rate Alteration Using MATLAB

- Example - The input and output plots of a factor-of-2 interpolator designed using Program 13_6 are shown below



42

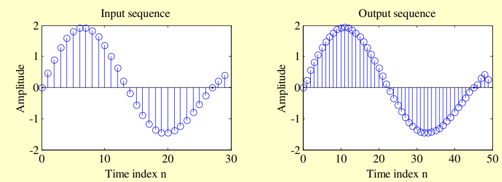
Sampling Rate Alteration Using MATLAB

- The function `resample` can be employed to increase the sampling rate of an input vector `x` by a ratio of two positive integers, `L/M`, generating an output vector `y`
- The M-file employs a lowpass FIR filter designed using `fir1` with a Kaiser window
- The fractional interpolation of a sequence can be obtained using Program 13_7 which employs the function `resample`

43

Sampling Rate Alteration Using MATLAB

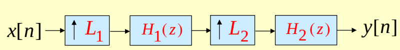
- Example - The input and output plots of a factor-of-5/3 interpolator designed using Program 13_7 are given below



44

Multistage Design of Decimator and Interpolator

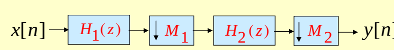
- The interpolator and the decimator can also be designed in more than one stages
- For example if the interpolation factor L can be expressed as a product of two integers, L_1 and L_2, then the factor-of-L interpolator can be realized in two stages as shown below



45

Multistage Design of Decimator and Interpolator

- Likewise if the decimator factor M can be expressed as a product of two integers, M_1 and M_2, then the factor-of-M interpolator can be realized in two stages as shown below



46

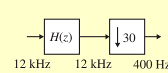
Multistage Design of Decimator and Interpolator

- Of course, the design can involve more than two stages, depending on the number of factors used to express L and M, respectively
- In general, the computational efficiency is improved significantly by designing the sampling rate alteration system as a cascade of several stages
- We consider the use of FIR filters here

47

Multistage Design of Decimator and Interpolator

- Example - Consider the design of a decimator for reducing the sampling rate of a signal from 12 kHz to 400 Hz
- The desired down-sampling factor is therefore M = 30 as shown below

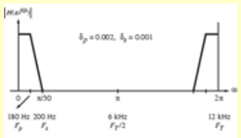


48

Multistage Design of Decimator and Interpolator

- Specifications for the decimation filter $H(z)$ are assumed to be as follows:

$$F_p = 180\,\text{Hz},\ F_s = 200\,\text{Hz},$$
$$\delta_p = 0.002,\ \delta_s = 0.001$$



49

Multistage Design of Decimator and Interpolator

- Assume $H(z)$ to be designed as an equiripple linear-phase FIR filter
- Now Kaiser's formula for estimating the order of $H(z)$ to meet the specifications is given by

$$N = \frac{-20\log_{10}\sqrt{\delta_p\delta_s} - 13}{14.6\Delta f}$$

where $\Delta f = (F_s - F_p)/F_T$ is the normalized transition bandwidth

50

Multistage Design of Decimator and Interpolator

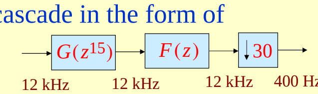
- The M-file kaiord determines the filter order using Kaiser's formula
- Using kaiord we obtain $N = 1808$
- Therefore, the number of multiplications per second in the single-stage implementation of the factor-of-30 decimator is

$$\mathcal{R}_{M,H} = 1809 \times \frac{12{,}000}{30} = 723{,}600$$

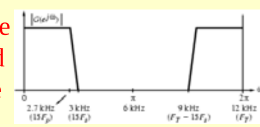
51

Multistage Design of Decimator and Interpolator

- We next implement $H(z)$ using the IFIR approach as a cascade in the form of $G(z^{15})F(z)$



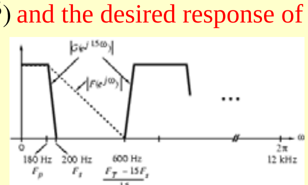
- The specifications of the parent filter $G(z)$ should thus be as shown on the right



52

Multistage Design of Decimator and Interpolator

- This corresponds to stretching the specifications of $H(z)$ by 15
- Figure below shows the magnitude response of $G(z^{15})$ and the desired response of $F(z)$



53

Multistage Design of Decimator and Interpolator

- Note: The desired response of $F(z)$ has a wider transition band as it takes into account the spectral gaps between the passbands of $G(z^{15})$
- Because of the cascade connection, the overall ripple of the cascade in dB is given by the sum of the passband ripples of $F(z)$ and $G(z^{15})$ in dB

54

Multistage Design of Decimator and Interpolator

- This can be compensated for by designing $F(z)$ and $G(z)$ to have a passband ripple of $\delta_p = 0.001$ each
- On the other hand, the cascade of $F(z)$ and $G(z^{15})$ has a stopband at least as good as $F(z)$ or $G(z^{15})$, individually
- So we can choose $\delta_s = 0.001$ for both filters

Multistage Design of Decimator and Interpolator

- Thus, specifications for the two filters $G(z)$ and $F(z)$ are as follows:

$$G(z): \delta_p = 0.001, \ \delta_s = 0.001, \ \Delta f = \frac{300}{12{,}000}$$

$$F(z): \delta_p = 0.001, \ \delta_s = 0.001, \ \Delta f = \frac{420}{12{,}000}$$

- The filter orders obtained using the M-file kaiord are: Order of $G(z)$ =129

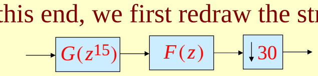
  Order of $F(z)$ = 92

Multistage Design of Decimator and Interpolator

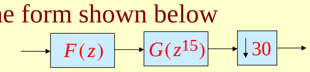
- The length of $H(z)$ for a direct implementation is 1809
- The length of cascade implementation $G(z^{15})F(z)$ is $92 + 15 \times 129 + 1 = 2028$
- ⇨ The length of the cascade structure is higher

Multistage Design of Decimator and Interpolator

- The computational complexity of the decimator implemented using the cascade structure can be dramatically reduced by making use of the cascade equivalence #1
- To this end, we first redraw the structure

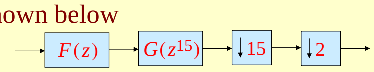


in the form shown below

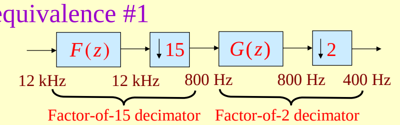


Multistage Design of Decimator and Interpolator

- The last structure is equivalent to the one shown below



- The above can be redrawn as indicated below by making use of the cascade equivalence #1



Multistage Design of Decimator and Interpolator

- From the last realization we observe that the implementation of $G(z)$ followed by a factor-of-2 down-sampler requires

$$\mathcal{R}_{M,G} = 130 \times \frac{800}{2} = 52{,}000 \quad \text{mult/sec}$$

- Likewise, the implementation of $F(z)$ followed by a factor-of-15 down-sampler requires

$$\mathcal{R}_{M,F} = 93 \times \frac{12{,}000}{15} = 74{,}400 \quad \text{mult/sec}$$

Multistage Design of Decimator and Interpolator

- The total complexity of the IFIR-based implementation of the factor-of-30 decimator is therefore

  52,000 + 74,400 = 126,400 mult/sec

  which is about 5.72 times smaller than that of a direct implementation of the decimation filter $H(z)$