Polyphase Decomposition

The Decomposition

- Consider an arbitrary sequence $\{x[n]\}$ with a z-transform $X(z)$ given by

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n}$$

- We can rewrite $X(z)$ as

$$X(z) = \sum_{k=0}^{M-1} z^{-k} X_k(z^M)$$

where

$$X_k(z) = \sum_{n=-\infty}^{\infty} x_k[n] z^{-n} = \sum_{n=-\infty}^{\infty} x[Mn+k] z^{-n}$$

$$0 \le k \le M-1$$

Polyphase Decomposition

- The subsequences $\{x_k[n]\}$ are called the **polyphase components** of the parent sequence $\{x[n]\}$
- The functions $X_k(z)$, given by the z-transforms of $\{x_k[n]\}$, are called the **polyphase components** of $X(z)$

Polyphase Decomposition

- The relation between the subsequences $\{x_k[n]\}$ and the original sequence $\{x[n]\}$ are given by

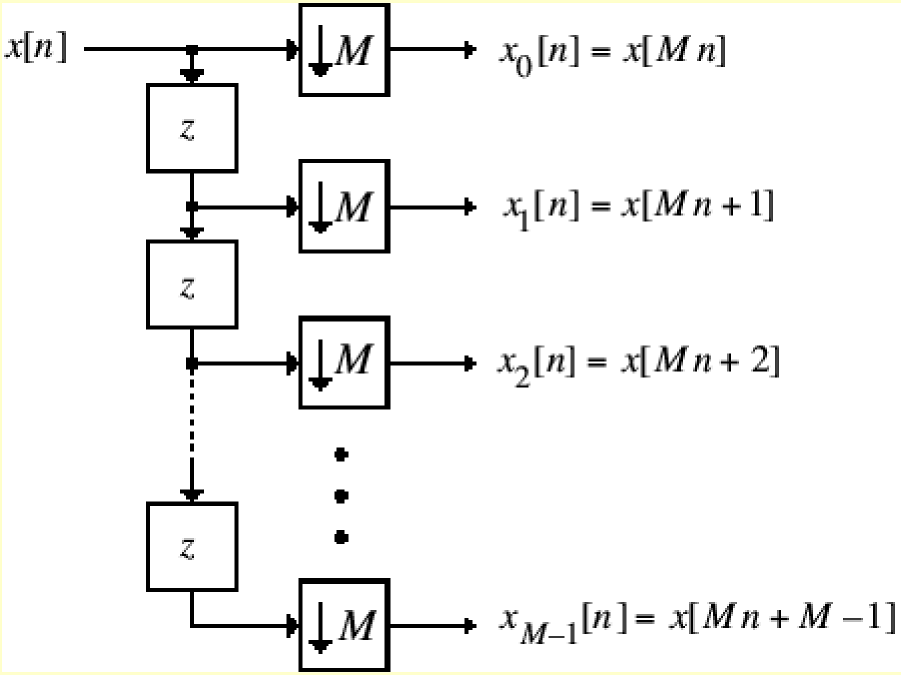
$$x_k[n] = x[Mn+k], \quad 0 \le k \le M-1$$

- In matrix form we can write

$$X(z) = \begin{bmatrix} 1 & z^{-1} & \cdots & z^{-(M-1)} \end{bmatrix} \begin{bmatrix} X_0(z^M) \\ X_1(z^M) \\ \vdots \\ X_{M-1}(z^M) \end{bmatrix}$$

Polyphase Decomposition

- A multirate structural interpretation of the polyphase decomposition is given below



Polyphase Decomposition

- The polyphase decomposition of an FIR transfer function can be carried out by inspection
- For example, consider a length-9 FIR transfer function:

$$H(z) = \sum_{n=0}^{8} h[n] z^{-n}$$

Polyphase Decomposition

- Its 4-branch polyphase decomposition is given by

$$H(z) = E_0(z^4) + z^{-1}E_1(z^4) + z^{-2}E_2(z^4) + z^{-3}E_3(z^4)$$

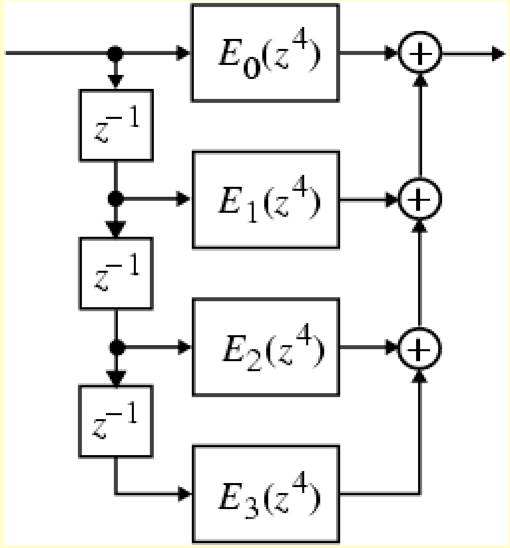
where

$$E_0(z) = h[0] + h[4]z^{-1} + h[8]z^{-2}$$

$$E_1(z) = h[1] + h[5]z^{-1}$$

$$E_2(z) = h[2] + h[6]z^{-1}$$

$$E_3(z) = h[3] + h[7]z^{-1}$$



Polyphase Decomposition

- The polyphase decomposition of an IIR transfer function $H(z) = P(z)/D(z)$ is not that straight forward
- One way to arrive at an M-branch polyphase decomposition of $H(z)$ is to express it in the form $P'(z)/D'(z^M)$ by multiplying $P(z)$ and $D(z)$ with an appropriately chosen polynomial and then apply an M-branch polyphase decomposition to $P'(z)$

Polyphase Decomposition

- Example - Consider

$$H(z) = \frac{1-2z^{-1}}{1+3z^{-1}}$$

- To obtain a 2-band polyphase decomposition we rewrite $H(z)$ as

$$H(z) = \frac{(1-2z^{-1})(1-3z^{-1})}{(1+3z^{-1})(1-3z^{-1})} = \frac{1-5z^{-1}+6z^{-2}}{1-9z^{-2}} = \frac{1+6z^{-2}}{1-9z^{-2}} + \frac{-5z^{-1}}{1-9z^{-2}}$$

- Therefore,

$$H(z) = E_0(z^2) + z^{-1}E_1(z^2)$$

where

$$E_0(z) = \frac{1+6z^{-1}}{1-9z^{-1}}, \quad E_1(z) = \frac{-5}{1-9z^{-1}}$$

Polyphase Decomposition

- Note: The above approach increases the overall order and complexity of $H(z)$
- However, when used in certain multirate structures, the approach may result in a more computationally efficient structure
- An alternative more attractive approach is discussed in the following example

Polyphase Decomposition

- Example - Consider the transfer function of a 5-th order Butterworth lowpass filter with a 3-dB cutoff frequency at 0.5π:

$$H(z) = \frac{0.0527864(1+z^{-1})^5}{1+0.633436854z^{-2}+0.0557281z^{-2}}$$

- It is easy to show that $H(z)$ can be expressed as

$$H(z) = \frac{1}{2}\left[\left(\frac{0.105573+z^{-2}}{1+0.105573z^{-2}}\right)+z^{-1}\left(\frac{0.52786+z^{-2}}{1+0.52786z^{-2}}\right)\right]$$

Polyphase Decomposition

- Therefore $H(z)$ can be expressed as

$$H(z) = E_0(z^2) + z^{-1} E_1(z^2)$$

where

$$E_0(z) = \frac{1}{2}\left(\frac{0.105573 + z^{-1}}{1 + 0.105573 z^{-1}}\right)$$

$$E_1(z) = \frac{1}{2}\left(\frac{0.52786 + z^{-1}}{1 + 0.52786 z^{-1}}\right)$$

Polyphase Decomposition

- Note: In the above polyphase decomposition, branch transfer functions $E_i(z)$ are stable allpass functions
- Moreover, the decomposition has not increased the order of the overall transfer function $H(z)$

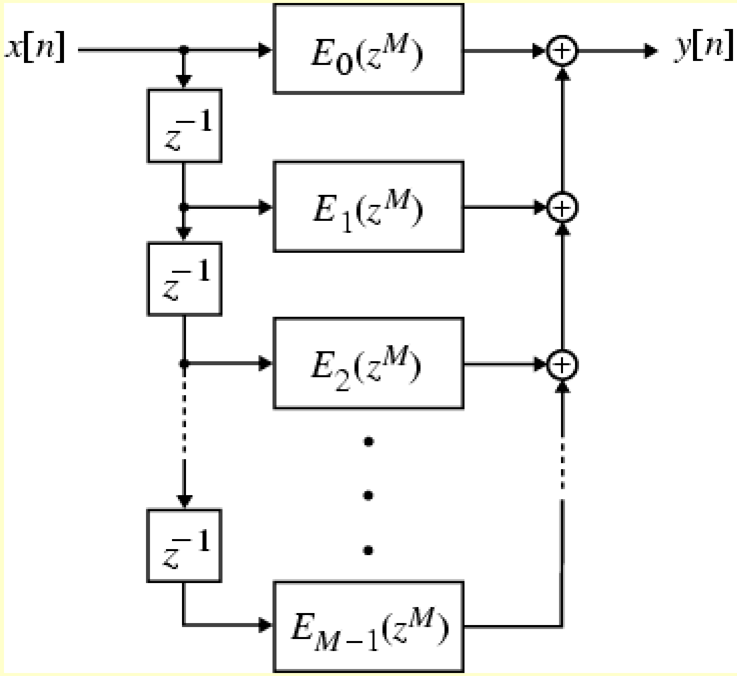
FIR Filter Structures Based on Polyphase Decomposition

- We shall demonstrate later that a parallel realization of an FIR transfer function $H(z)$ based on the polyphase decomposition can often result in computationally efficient multirate structures
- Consider the M-branch Type I polyphase decomposition of $H(z)$:

$$H(z) = \sum_{k=0}^{M-1} z^{-k} E_k(z^M)$$

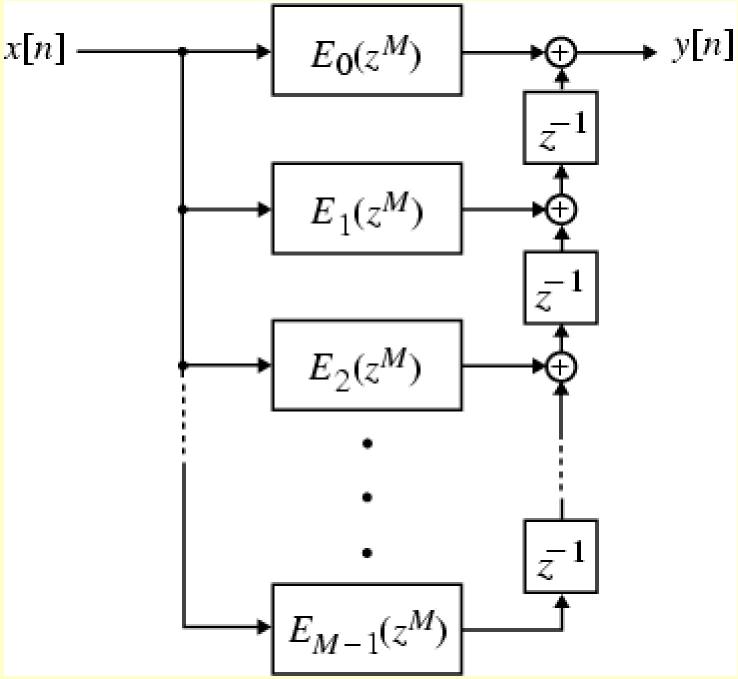
FIR Filter Structures Based on Polyphase Decomposition

- A direct realization of $H(z)$ based on the Type I polyphase decomposition is shown below



FIR Filter Structures Based on Polyphase Decomposition

- The transpose of the Type I polyphase FIR filter structure is indicated below



FIR Filter Structures Based on Polyphase Decomposition

- An alternative representation of the transpose structure shown on the previous slide is obtained using the notation

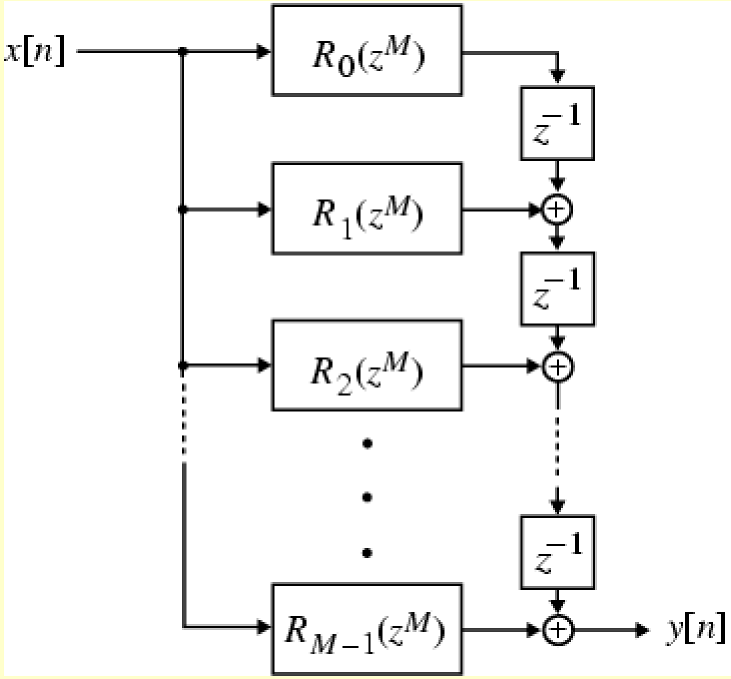
$$R_\ell(z^M) = E_{M-1-\ell}(z^M), \qquad 0 \le \ell \le M-1$$

- Substituting the above notation in the Type I polyphase decomposition we arrive at the Type II polyphase decomposition:

$$H(z) = \sum_{\ell=0}^{M-1} z^{-(M-1-\ell)} R_\ell(z^M)$$

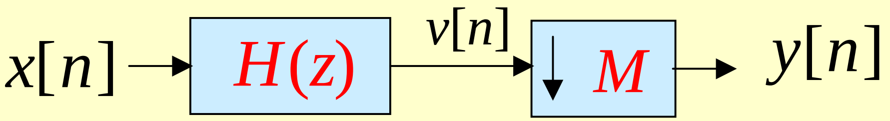
FIR Filter Structures Based on Polyphase Decomposition

- A direct realization of $H(z)$ based on the Type II polyphase decomposition is shown below



Computationally Efficient Decimators

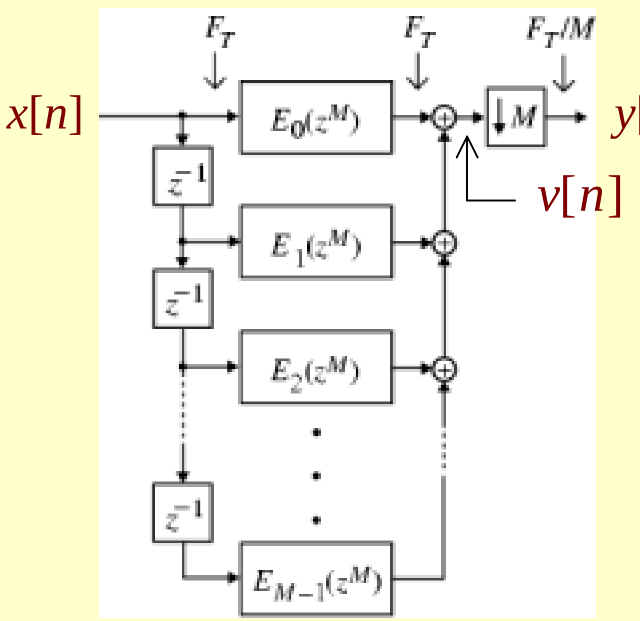
- Consider first the single-stage factor-of-M decimator structure shown below

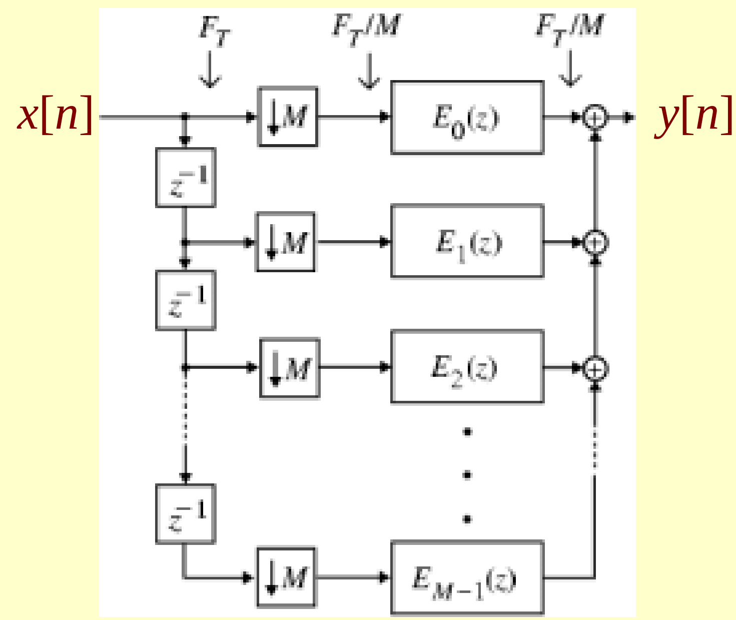


- We realize the lowpass filter $H(z)$ using the Type I polyphase structure as shown on the next slide

Computationally Efficient Decimators

- Using the cascade equivalence #1 we arrive at the computationally efficient decimator structure shown below on the right





Decimator structure based on Type I polyphase decomposition

Computationally Efficient Decimators

- To illustrate the computational efficiency of the modified decimator structure, assume $H(z)$ to be a length-N structure and the input sampling period to be $T = 1$
- Now the decimator output $y[n]$ in the original structure is obtained by down-sampling the filter output $v[n]$ by a factor of M

Computationally Efficient Decimators

- It is thus necessary to compute $v[n]$ at

$$n = \ldots, -2M, -M, 0, M, 2M, \ldots$$

- Computational requirements are therefore N multiplications and $(N-1)$ additions per output sample being computed
- However, as n increases, stored signals in the delay registers change

Computationally Efficient Decimators

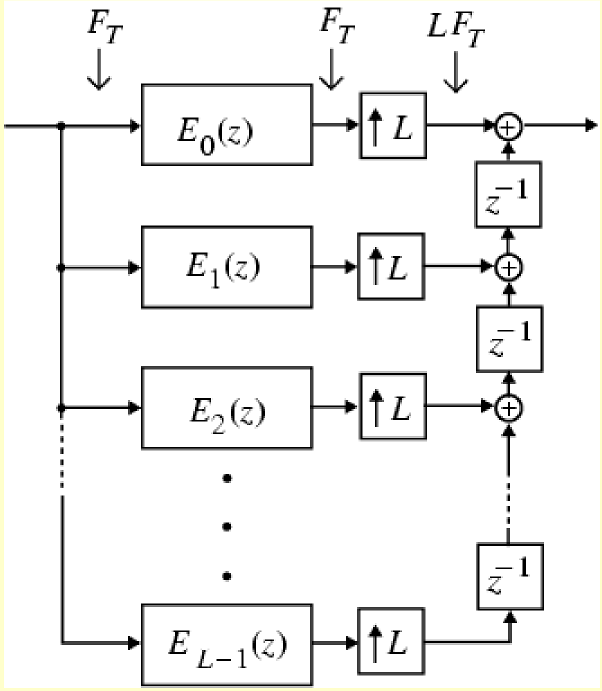
- Hence, all computations need to be completed in one sampling period, and for the following $(M-1)$ sampling periods the arithmetic units remain idle
- The modified decimator structure also requires N multiplications and $(N-1)$ additions per output sample being computed

Computationally Efficient Decimators and Interpolators

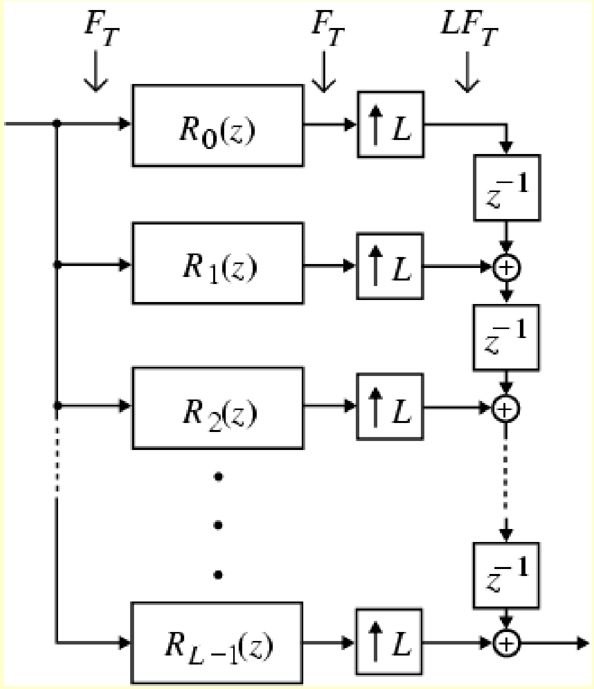
- However, here the arithmetic units are operative at all instants of the output sampling period which is M times that of the input sampling period
- Similar savings are also obtained in the case of the interpolator structure developed using the polyphase decomposition

Computationally Efficient Interpolators

- Figures below show the computationally efficient interpolator structures



Interpolator based on
Type I polyphase decomposition



Interpolator based on
Type II polyphase decomposition

Computationally Efficient Decimators and Interpolators

- More efficient interpolator and decimator structures can be realized by exploiting the symmetry of filter coefficients in the case of linear-phase filters $H(z)$
- Consider for example the realization of a factor-of-3 ($M = 3$) decimator using a length-12 Type 1 linear-phase FIR lowpass filter

Computationally Efficient Decimators and Interpolators

- The corresponding transfer function is

$$H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[4]z^{-4} + h[5]z^{-5}$$
$$+ h[5]z^{-6} + h[4]z^{-7} + h[3]z^{-8} + h[2]z^{-9} + h[1]z^{-10} + h[0]z^{-11}$$

- A conventional polyphase decomposition of $H(z)$ yields the following subfilters:

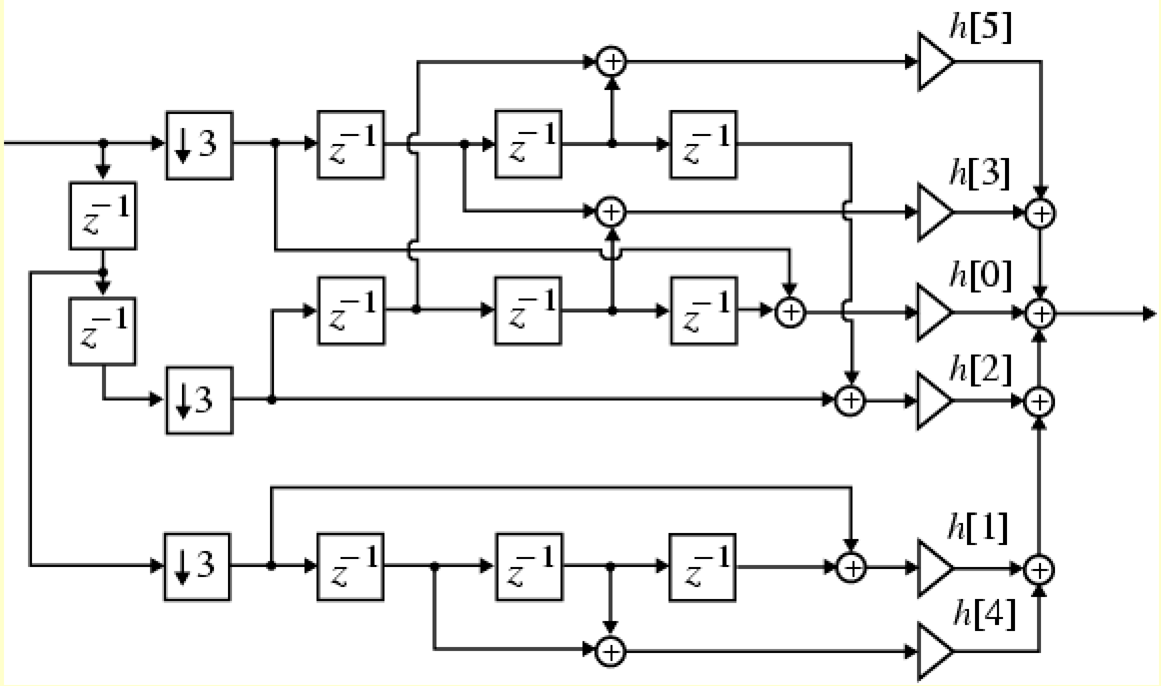
$$E_0(z) = h[0] + h[3]z^{-1} + h[5]z^{-2} + h[2]z^{-3}$$
$$E_1(z) = h[1] + h[4]z^{-1} + h[4]z^{-2} + h[1]z^{-3}$$
$$E_2(z) = h[2] + h[5]z^{-1} + h[3]z^{-2} + h[0]z^{-3}$$

Computationally Efficient Decimators and Interpolators

- Note that $E_1(z)$ still has a symmetric impulse response, whereas $E_0(z)$ is the mirror image of $E_2(z)$
- These relations can be made use of in developing a computationally efficient realization using only 6 multipliers and 11 two-input adders as shown on the next slide

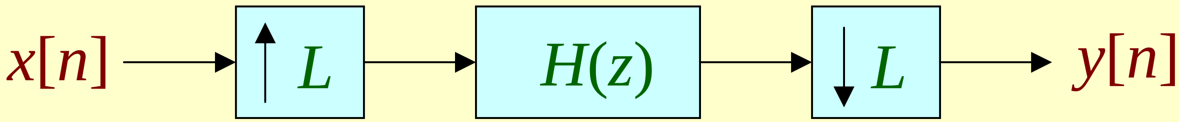
Computationally Efficient Decimators and Interpolators

- Factor-of-3 decimator with a linear-phase decimation filter

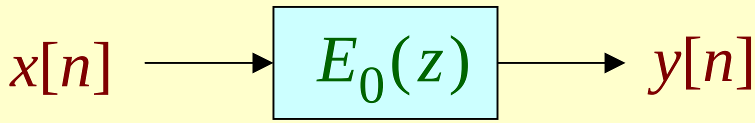


A Useful Identity

- The cascade multirate structure shown below appears in a number of applications



- Equivalent time-invariant digital filter obtained by expressing $H(z)$ in its L-term Type I polyphase form $\sum_{k=0}^{L-1} z^{-k} E_k(z^L)$ is shown below

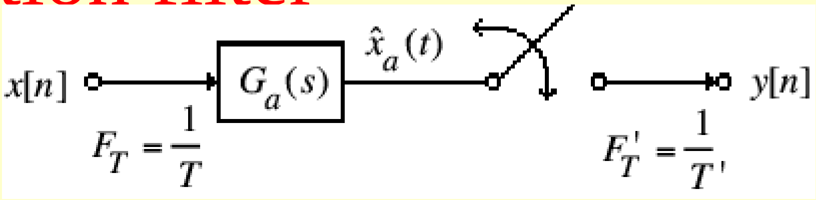


Arbitrary-Rate Sampling Rate Converter

- The estimation of a discrete-time signal value at an arbitrary time instant between a consecutive pair of known samples can be solved by using some type of interpolation
- In this approach an approximating continuous-time signal is formed from a set of known consecutive samples of the given discrete-time signal

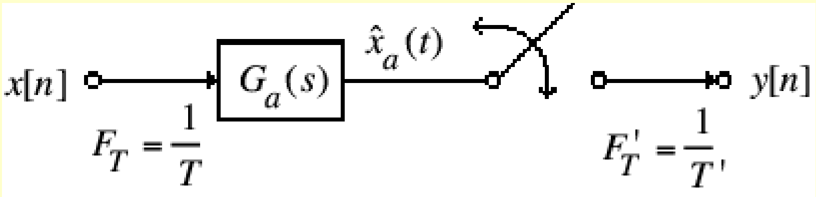
Arbitrary-Rate Sampling Rate Converter

- The value of the approximating continuous-time signal is then evaluated at the desired time instant
- This interpolation process can be directly implemented by designing a digital interpolation filter



Ideal Sampling Rate Converter

- In principle, a sampling rate conversion by an arbitrary conversion factor can be implemented as follows
- The input digital signal is passed through an ideal analog reconstruction lowpass filter whose output is resampled at the desired output rate as indicated below



Ideal Sampling Rate Converter

- Let the impulse response of the analog lowpass filter is denoted by $g_a(t)$
- Then the output of the filter is given by

$$\hat{x}_a(t) = \sum_{\ell=-\infty}^{\infty} x[\ell] g_a(t - \ell T)$$

- If the analog filter is chosen to bandlimit its output to the frequency range $F_g < F_T^{'}/2$, its output $\hat{x}_a(t)$ can then be resampled at the rate $F_T^{'}$

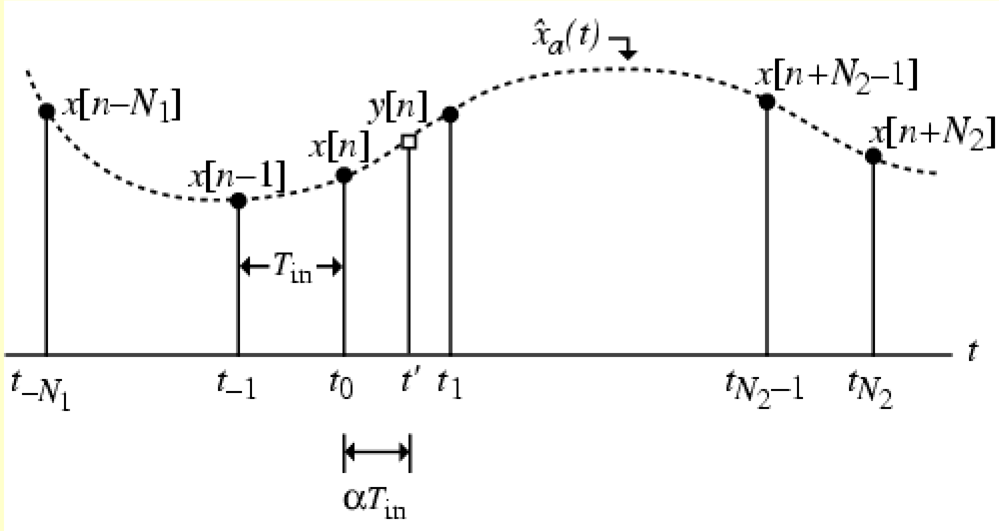
Ideal Sampling Rate Converter

- Since the impulse response $g_a(t)$ of an ideal lowpass analog filter is of infinite duration and the samples $g_a(nT' - \ell T)$ have to be computed at each sampling instant, implementation of the ideal bandlimited interpolation algorithm in exact form is not practical
- Thus, an approximation is employed in practice

Ideal Sampling Rate Converter

- Problem statement: Given $N_2 + N_1 + 1$ input signal samples, $x[k]$, $k = -N_1, \ldots, N_2$, obtained by sampling an analog signal $x_a(t)$ at $t = t_k = t_0 + kT_{in}$, determine the sample value $x_a(t_0 + kT_{in}) = y[\alpha]$ at time instant $t' = t_0 + kT_{in}$ where $-N_1 \le \alpha \le N_2$
- Figure on the next slide illustrates the interpolation process by an arbitrary factor

Ideal Sampling Rate Converter



- We describe next a commonly employed interpolation algorithm based on a finite weighted sum of input samples

Lagrange Interpolation Algorithm

- Here, a polynomial approximation $\hat{x}_a(t)$ to $x_a(t)$ is defined as

$$\hat{x}_a(t) = \sum_{k=-N_1}^{N_2} P_k(t)\,x[n+k]$$

where $P_k(t)$ are the Lagrange polynomials given by

$$P_k(t) = \prod_{\substack{\ell=-N_1 \\ \ell \neq k}}^{N_2} \left(\frac{t - t_k}{t_k - t_\ell} \right), \quad -N_1 \le k \le N_2$$

Lagrange Interpolation Algorithm

- Example - Design a fractional-rate interpolator with an interpolation factor of 3/2 using a 3rd-order polynomial approximation with $N_1 = 2$ and $N_2 = 1$
- The output $y[n]$ of the interpolator is thus computed using

$$y[n] = P_{-2}(\alpha)x[n-2] + P_{-1}(\alpha)x[n-1] + P_0(\alpha)x[n] + P_1(\alpha)x[n+1]$$

Lagrange Interpolation Algorithm

- Here, the Lagrange polynomials are given by

$$P_{-2}(\alpha) = \frac{(\alpha+1)\alpha(\alpha-1)}{-6} = \frac{1}{6}(-\alpha^3 + \alpha)$$

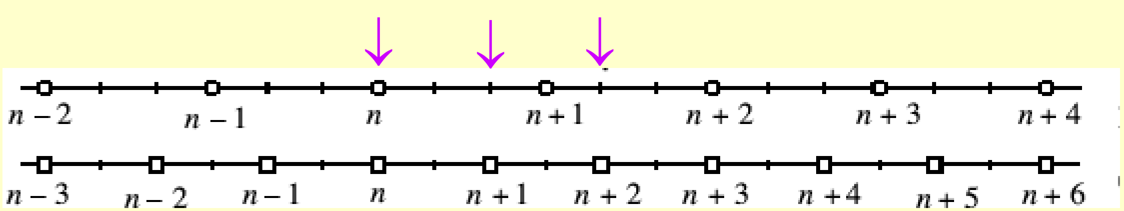
$$P_{-1}(\alpha) = \frac{(\alpha+2)\alpha(\alpha-1)}{2} = \frac{1}{2}(\alpha^3 + \alpha^2 - 2\alpha)$$

$$P_{0}(\alpha) = \frac{(\alpha+2)(\alpha+1)(\alpha-1)}{-2} = -\frac{1}{2}(\alpha^3 + 2\alpha^2 - \alpha - 2)$$

$$P_{1}(\alpha) = \frac{(\alpha+2)(\alpha+1)\alpha}{-6} = \frac{1}{6}(\alpha^3 + 3\alpha^2 + \alpha)$$

Lagrange Interpolation Algorithm

- Figure below shows the locations of the samples of the input and the output for an interpolator with a conversion factor of 3/2
- Locations of the output samples $y[0]$, $y[1]$, and $y[2]$ in the input sample domain are marked with an arrow



Input sample index

Output sample index

Lagrange Interpolation Algorithm

- From the figure on the previous slide it can be seen that the value of α for computation of $y[n]$, to be labeled α_0 , is 0
- Substituting this value of α in the expressions for the Lagrange polynomial coefficients derived earlier we get

$$P_{-2}(\alpha_0) = 0, \quad P_{-1}(\alpha_0) = 0$$
$$P_0(\alpha_0) = 1, \quad P_1(\alpha_0) = 0$$

Lagrange Interpolation Algorithm

- The value of α for computation of $y[n+1]$, to be labeled α_1 , is 2/3
- Substituting this value of α in the expressions for the Lagrange polynomial coefficients we get

$$P_{-2}(\alpha_1) = 0.0617 \, , \; P_{-1}(\alpha_1) = -0.2963$$
$$P_0(\alpha_1) = 0.7407 \, , \; P_1(\alpha_1) = 0.4938$$

Lagrange Interpolation Algorithm

- The value of α for computation of $y[n+2]$, to be labeled α_2, is 4/3
- Substituting this value of α in the expressions for the Lagrange polynomial coefficients we get

$$P_{-2}(\alpha_2) = -0.1728\,, \quad P_{-1}(\alpha_2) = 0.7407$$
$$P_0(\alpha_2) = -1.2963\,, \quad P_1(\alpha_2) = 1.7284$$

Lagrange Interpolation Algorithm

- Substituting the values of the Lagrange polynomial coefficients in the interpolator output equation for n, $n+1$, and $n+2$, and combining the three equations into a matrix form we arrive at

$$\begin{bmatrix} y[n] \\ y[n+1] \\ y[n+2] \end{bmatrix} = \begin{bmatrix} P_{-2}(\alpha_0) & P_{-1}(\alpha_0) & P_0(\alpha_0) & P_1(\alpha_0) \\ P_{-2}(\alpha_1) & P_{-1}(\alpha_1) & P_0(\alpha_1) & P_1(\alpha_1) \\ P_{-2}(\alpha_2) & P_{-1}(\alpha_2) & P_0(\alpha_2) & P_1(\alpha_2) \end{bmatrix} \begin{bmatrix} x[n-2] \\ x[n-1] \\ x[n] \\ x[n+1] \end{bmatrix}$$

Lagrange Interpolation Algorithm

- The input-output relation of the interpolation filter can be compactly written as

$$\begin{bmatrix} y[n] \\ y[n+1] \\ y[n+2] \end{bmatrix} = \mathbf{H} \begin{bmatrix} x[n-2] \\ x[n-1] \\ x[n] \\ x[n+1] \end{bmatrix}$$

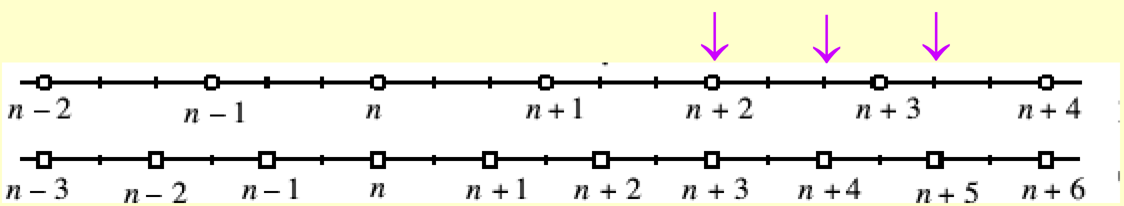
where **H** is the block coefficient matrix

Lagrange Interpolation Algorithm

- For the factor-of-3/2 interpolator, we have

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0.0617 & -0.2963 & 0.7407 & 0.4938 \\ -0.1728 & 0.7407 & -1.2963 & 1.7284 \end{bmatrix}$$

- It should be evident from an examination of



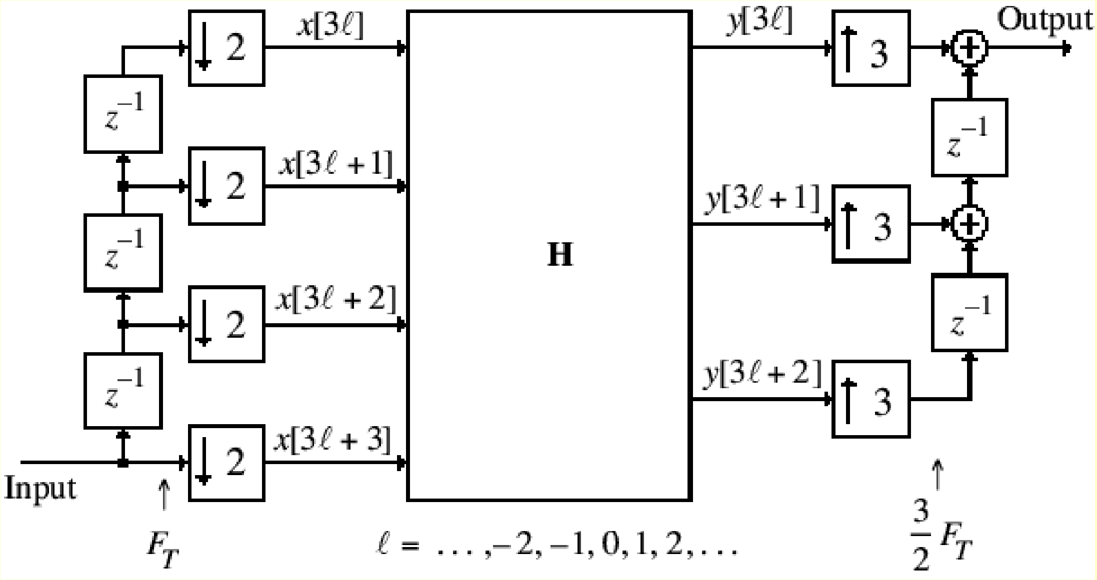
Input sample index

Output sample index

that the filter coefficients to compute $y[n+3]$, $y[n+4]$, and $y[n+5]$ are again given by the same block matrix $\mathbf{H}$

Lagrange Interpolation Algorithm

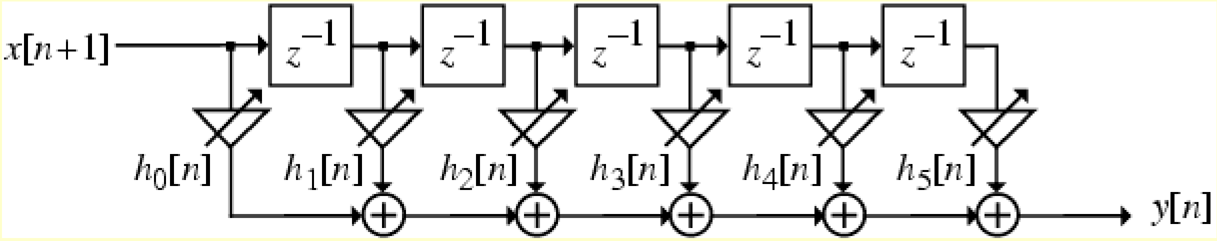
- The desired interpolation filter is a time-varying filter
- A realization of the interpolator is given below



Lagrange Interpolation Algorithm

- Note: In practice, the overall system delay will be 3 sample periods
- Hence, the output sample $y[n]$ actually will appear at the time index $n+3$
- A realization of the factor-of-3 interpolator in the form of a time-varying filter is shown on the next slide

Lagrange Interpolation Algorithm



- The coefficients of the 5-th order time-varying FIR filter have a period of 3 and are assigned the values indicated below

Time	$h_0[n]$	$h_1[n]$	$h_2[n]$	$h_3[n]$	$h_4[n]$	$h_5[n]$
3ℓ	$P_1(\alpha_0)$	$P_0(\alpha_0)$	$P_{-1}(\alpha_0)$	$P_{-2}(\alpha_0)$	0	0
$3\ell+1$	0	$P_1(\alpha_1)$	$P_0(\alpha_1)$	$P_{-1}(\alpha_1)$	$P_{-2}(\alpha_1)$	0
$3\ell+2$	0	0	$P_1(\alpha_2)$	$P_0(\alpha_2)$	$P_{-1}(\alpha_2)$	$P_{-2}(\alpha_2)$

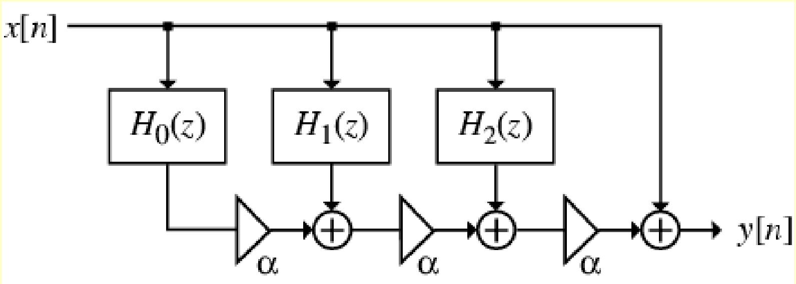
Lagrange Interpolation Algorithm

- Substituting the expressions for the Lagrange polynomials in the output equation we arrive at

$$
\begin{aligned}
y[n] = {} & \alpha^3\left(-\tfrac{1}{6}x[n-2] + \tfrac{1}{2}x[n-1] - \tfrac{1}{2}x[n] + \tfrac{1}{6}x[n+1]\right) \\
& + \alpha^2\left(\tfrac{1}{2}x[n-1] - x[n] + \tfrac{1}{2}x[n+1]\right) \\
& + \alpha\left(\tfrac{1}{6}x[n-2] - x[n-1] + \tfrac{1}{2}x[n] + \tfrac{1}{3}x[n+1]\right) \\
& + x[n]
\end{aligned}
$$

Lagrange Interpolation Algorithm

- A digital filter realization of the equation on the previous slide leads to the Farrow structure shown below



- In the above structure

$$H_0(z) = -\frac{1}{6}z^{-2} + \frac{1}{2}z^{-1} - \frac{1}{2} + \frac{1}{6}z$$

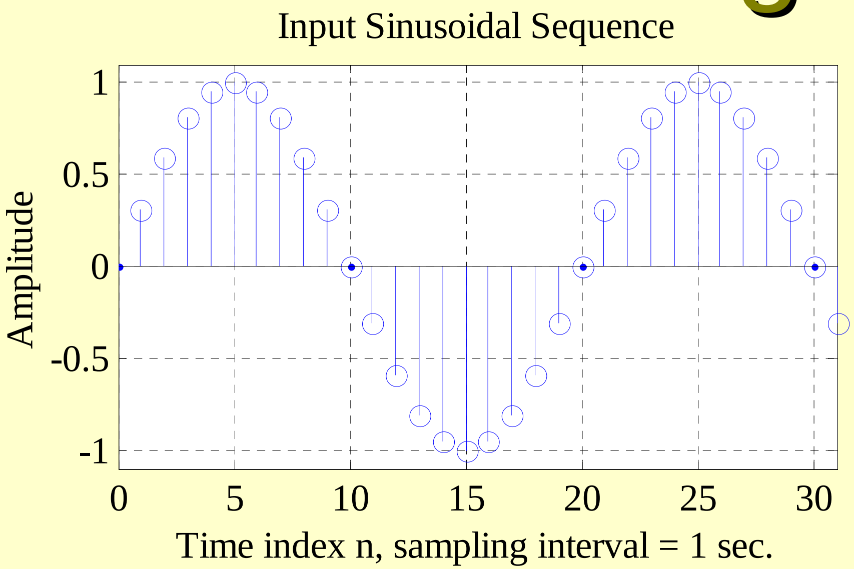
$$H_1(z) = \frac{1}{2}z^{-1} - 1 + \frac{1}{2}z$$

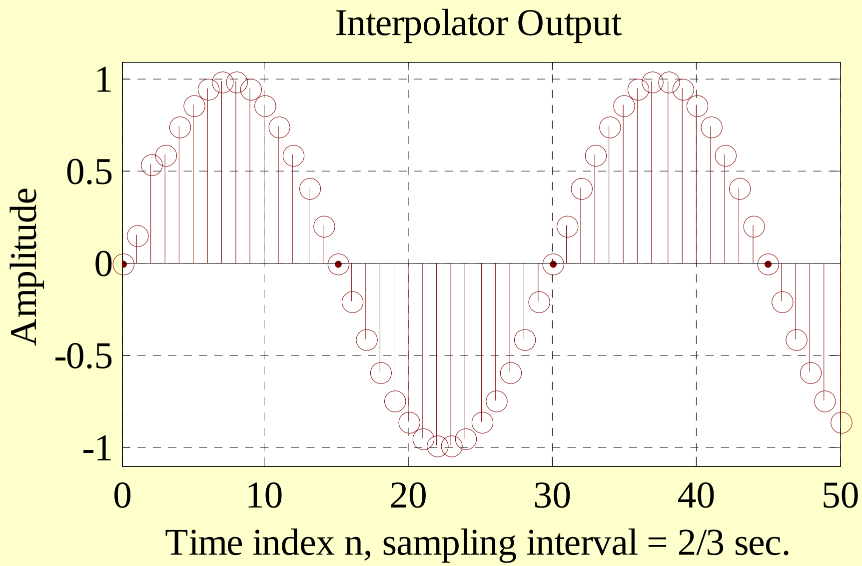
$$H_2(z) = \frac{1}{6}z^{-2} - z^{-1} + \frac{1}{2} + \frac{1}{3}z$$

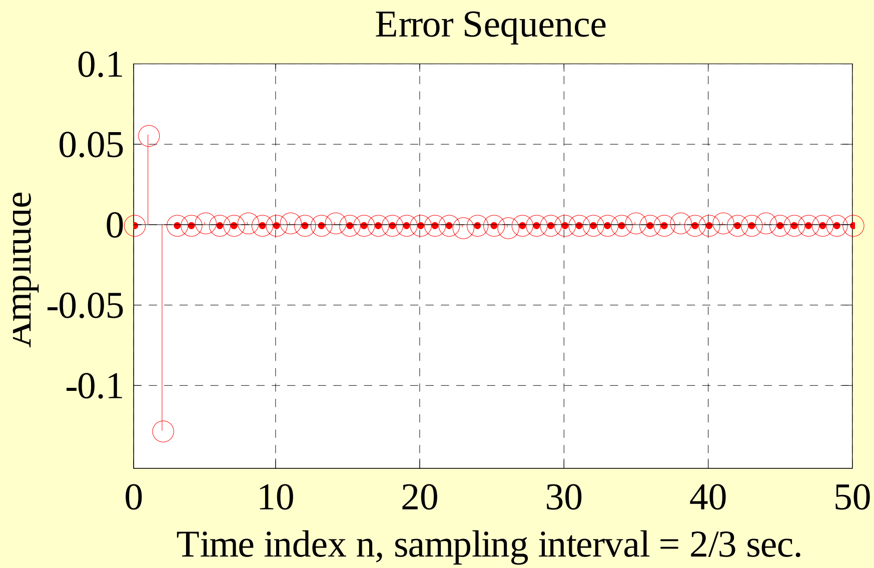
Lagrange Interpolation Algorithm

- In the Farrow structure only the value of a is changed periodically with the remaining digital filter structure kept unchanged
- Figures on the next slide show the input and the output of the above interpolator for a sinusoidal input of frequency of 0.05 Hz sampled at a 1-Hz rate

Lagrange Interpolation Algorithm







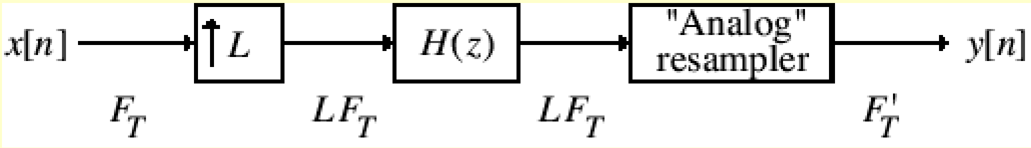
Arbitrary-Rate Sampling Rate Converter

Practical Considerations

- A direct design of a fractional-rate sampling rate converter in most applications is not practical
- This is due to two main reasons:
  - length of the time-varying filter needed is usually very large
  - real-time computation of the corresponding filter coefficients is nearly impossible

Arbitrary-Rate Sampling Rate Converter

- As a result, the fractional-rate sampling rate converter is almost realized in a hybrid form as indicated below for the case of an interpolator



- The digital sampling rate converter can be implemented in a multistage form to reduce the computational complexity