MIT OpenCourseWare
http://ocw.mit.edu

2.161 Signal Processing: Continuous and Discrete
Fall 2008

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING
2.161 Signal Processing – Continuous and Discrete

Data Resampling: Interpolation (Up-sampling) and Decimation (Down-sampling) [1]

1 Up-Sampling (Interpolation) by an Integer Factor

Consider a data set $\{f_n\}$ of length N, where $f_n = f(n\Delta T)$, $n = 0 \dots N-1$ and ΔT is the sampling interval. The task is to resample the data at a higher rate so as to create a new data set $\{\hat{f}_n\}$. of length KN, representing samples of the same continuous waveform $f(t)$, sampled at intervals $\Delta T/K$. Figure 1(a) shows a cosinusoidal data set with $N = 8$ samples, and Fig. 1(b) shows the

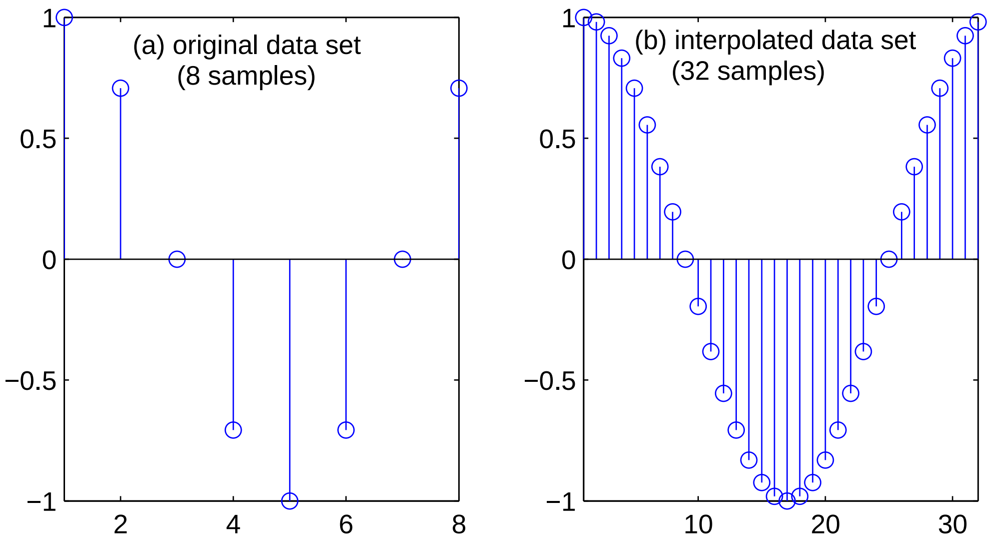


Figure 1: (a) a data set with $N = 8$ samples, and (b) an interpolated data set ($K = 4$) derived from (a).

same data set interpolated by a factor $K = 4$.

1.1 Frequency Domain Method

This method is useful for a finite-sized data record. Consider the DFTs $\{F_m\}$ and $\{\hat{F}_m\}$ of a pair of sample sets $\{f_n\}$ and $\{\hat{f}_n\}$, both recorded from $f(t)$ from $0 \le t < T$, but with sampling intervals ΔT and $\Delta T/K$ respectively. Let N and KN be the corresponding sample sizes. It is assumed that ΔT has been chosen to satisfy the Nyquist criterion:

[1] D. Rowell August 18, 2008

- Let $F(j\Omega) = \mathcal{F}\{f(t)\}$ be the Fourier transform of $f(t)$, and let $f(t)$ be sampled at intervals ΔT to produce $f^*(t)$. Then

$$F^*(j\Omega) = \frac{1}{\Delta T} \sum_{n=0}^{\infty} F\left(j\left(\Omega - \frac{2\pi n}{\Delta T}\right)\right) \tag{1}$$

is periodic with period $2\pi/\Delta T$, and consists of scaled and shifted replicas of $F(j\Omega)$. Let the total sampling interval be T to produce $N = T/\Delta T$ samples.

- If the same waveform $f(t)$ is sampled at intervals $\Delta T/K$ to produce $\hat{f}^*(t)$ the period of its Fourier transform $\hat{F}^*(j\Omega)$ is $2\pi K/\Delta T$ and

$$\hat{F}^*(j\Omega) = \frac{K}{\Delta T} \sum_{n=0}^{\infty} F\left(j\left(\Omega - \frac{2\pi Kn}{\Delta T}\right)\right) \tag{2}$$

which differs only by a scale factor, and an increase in the period. Let the total sampling period be T as above, to generate KN samples.

- We consider the DFTs to be sampled representations of a single period of $F^*(j\Omega)$ and $\hat{F}^*(j\Omega)$. The equivalent line spacing in the DFT depents only on the total duration of the sample set T, and is $\Delta\Omega = 2\pi/T$ in each case:

$$\begin{aligned} F_m &= F^*\left(j\left(\frac{2\pi m}{T}\right)\right), \quad m = 0, 1, \ldots N-1 \\ \hat{F}_m &= \hat{F}^*\left(j\left(\frac{2\pi m}{T}\right)\right), \quad m = 0, 1, \ldots KN-1. \end{aligned}$$

From Eqs. (1) and (2) the two DFTs $\{F_m\}$ and $\left\{\hat{F}_m\right\}$ are related:

$$\hat{F}_m = \begin{cases} KF_m & m = 0, 1, \ldots, N/2-1 \\ 0 & m = N/2, \ldots, NK - N/2 - 1 \\ KF_{m-(K-1)N} & m = NK - N/2, \ldots, KN-1 \end{cases}$$

- The effect of increasing N (or decreasing ΔT) in the sample set, while maintaining $T = N\Delta T$ constant, is to increase the length of the DFT by raising the effective Nyquist frequency Ω_N.

$$\Omega_N = \frac{\pi}{\Delta T} = \frac{N\pi}{T}$$

Figure 2 demonstrates these effects by schematically, by comparing the DFT of (a) a data set of length N derived by sampling at intervals ΔT, and (b) a data set of length $2N$ resulting from sampling at intervals $\Delta T/2$. The low frequency region of both spectra are similar, except for a scale factor, and the primary difference lies in the "high frequency" region, centered around the Nyquist frequency, in which all data points are zero.

The above leads to an algorithm for the interpolation of additional points into a data set, by a constant factor K:

1. Take the DFT of the original data set to create $\{F_m\}$ of length N.

2. Insert $(K-1)N$ zeros into the *center* of the DFT to create a length KN array.

3. Take the IDFT of the expanded array, and scale the sequence by a factor K.

Appendix A describes a simple MATLAB function to interpolate a real data set using this method.

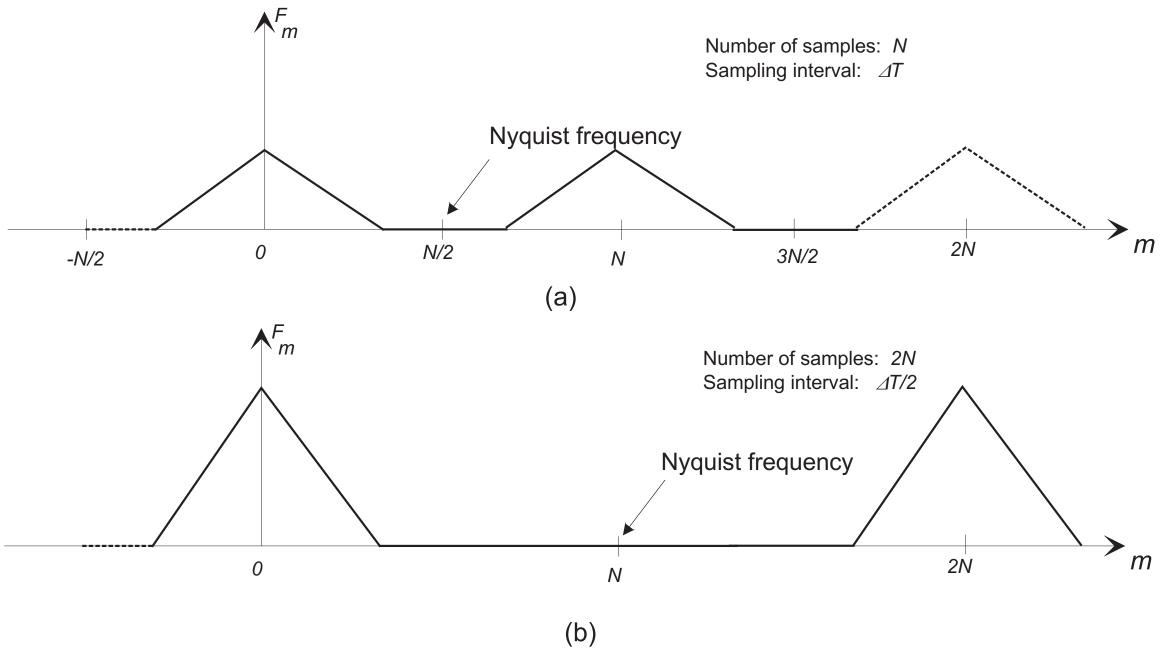


Figure 2: Envelopes of (a) the DFT of a sample set with N samples, and (b) the DFT of the same waveform with $2N$ samples.

1.2 A Time-Domain Method

We now examine an interpolation scheme that may implemented in real-time using time domain processing alone. As before, assume that the process $f(t)$ is sampled at intervals ΔT, generating a sequence $\{f_n\} = \{f(n\Delta T)\}$. Now assume that $K-1$ zeros are inserted between the samples to form a sequence $\{\hat{f}_k\}$ at intervals $\Delta T/K$. This is illustrated in Fig. 3, where a data record with $N = 8$ samples has been expanded by a factor $K = 3$ to form a new data record of length $N = 24$ formed by inserting two zero samples between each of the original data points.

We now examine the effect of inserting $K-1$ samples with amplitude 0 after each sample. The DFT of the original data set is

$$F_m = \sum_{n=0}^{N-1} f_n e^{-j\frac{2\pi mn}{N}}, \qquad m = 0 \dots N-1$$

and for the extended data set $\{\hat{f}_n\}$, $n = 0 \dots KN-1$

$$\hat{F}_m = \sum_{n=0}^{KN-1} \hat{f}_n e^{-j\frac{2\pi mn}{KN}}, \qquad m = 0 \dots KN-1$$

However, only the original N samples contribute to the sum, so that we can write

$$\begin{aligned} \hat{F}_m &= \sum_{k=0}^{N-1} \hat{f}_{Kk} e^{-j\frac{2\pi mk}{N}} \\ &= F_m, \qquad m = 0 \dots KN-1 \end{aligned}$$

since $\hat{f}_{Kk} = f_k$. We note that $\{F_m\}$ is periodic with period N, and $\{\hat{F}_m\}$ is periodic with period KN, so that $\{\hat{F}_m\}$ will contain K repetitions of $\{F_m\}$. This is illustrated in Fig. 4, which shows

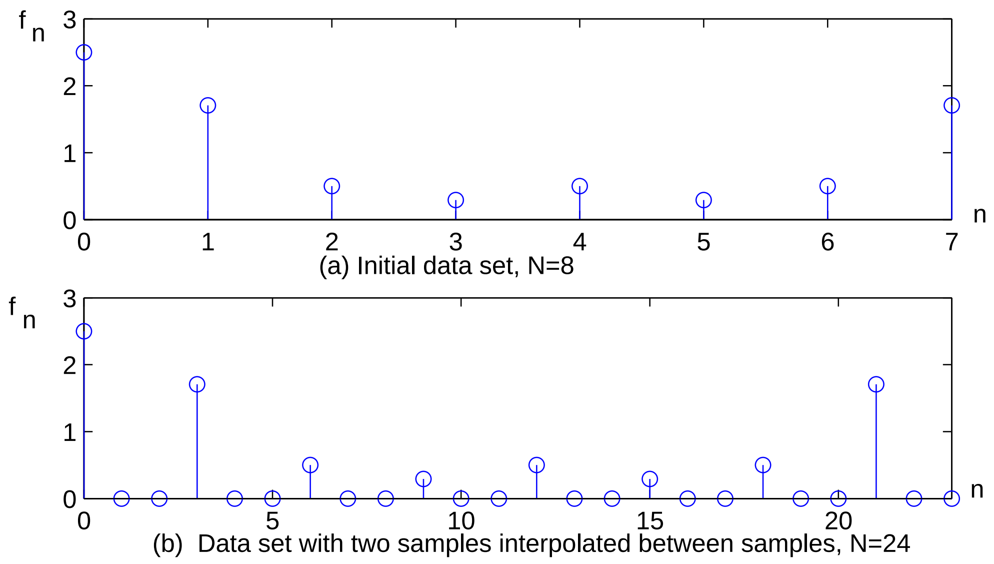


Figure 3: (a) A sample set with $N = 8$ samples, and (b) the same waveform with 2 zero amplitude samples interpolated between each of the original samples.

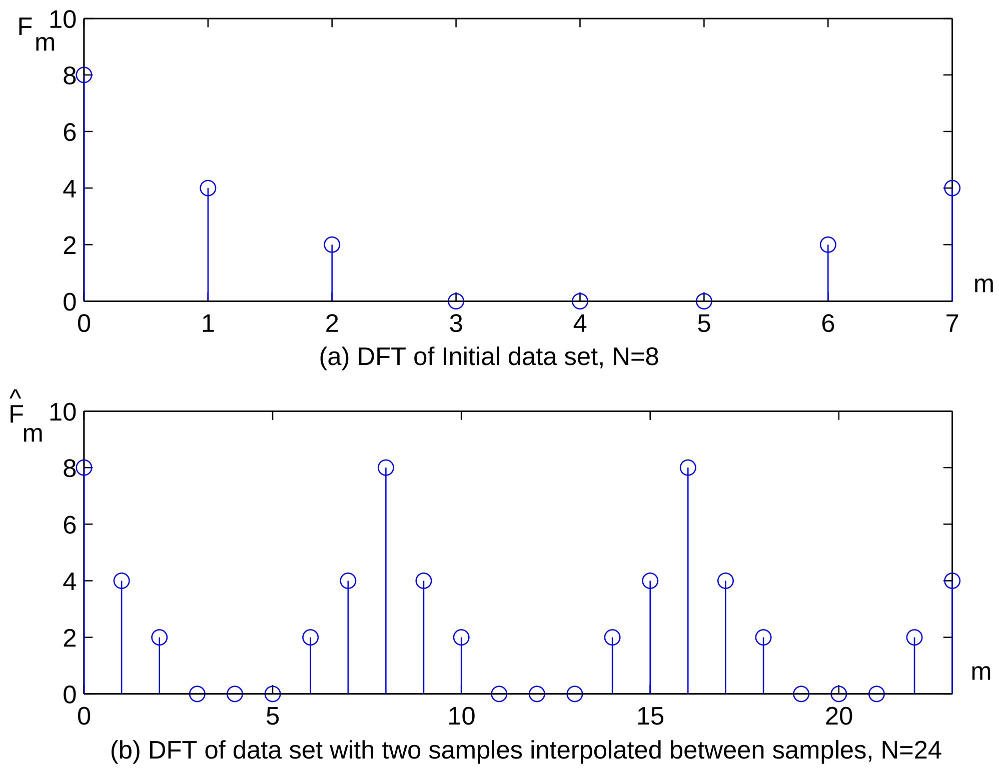


Figure 4: (a) DFTs of the sample sets shown in Fig. 3. In (a) the DFT $\{F_m\}$ is periodic with period 8, and in (b) the DFT $\{\hat{F}_m\}$ is periodic with period 24.

the magnitude of the DFTs of the two waveforms in Fig. 3. The effect of inserting the $K-1$ zeros between the original samples has been to generate a waveform with an equivalent sampling interval of $\Delta T/K$ s, and a Nyquist frequency of $K\pi/\Delta T$ rad/s. The line resolution is unchanged, and the original DFT $\{F_m\}$ is replicated K times within the frequency span of $2K\pi/\Delta T$ rad/s.

The fully interpolated waveform may be reconstructed by elimination of the replications of the original spectral components. While this might be done in the frequency domain, the most common method is to low-pass filter the padded data sequence to retain only the base-band portion of the spectrum as shown in Fig. 5.

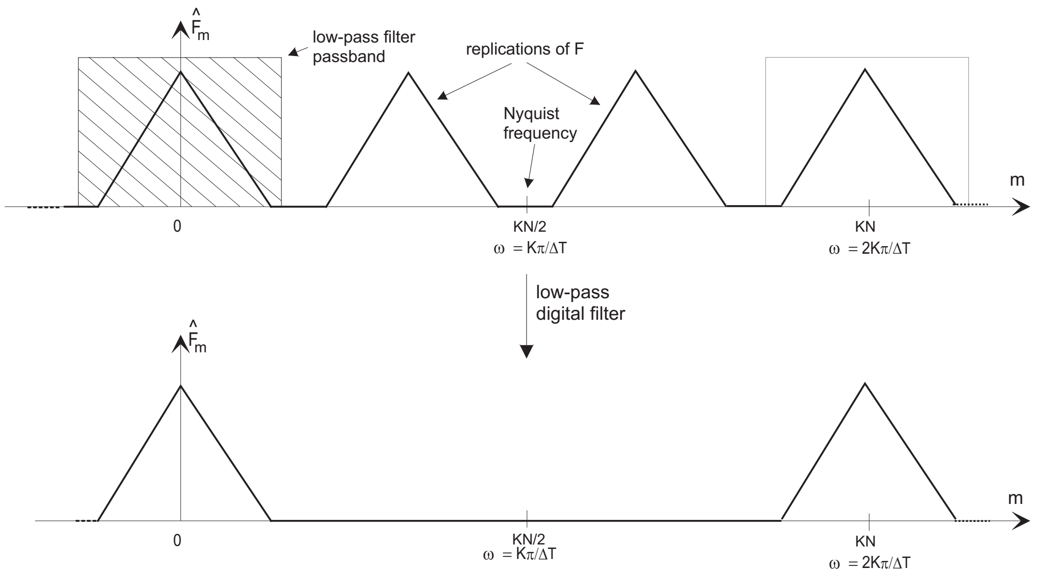


Figure 5: Digital low-pass filtering used to eliminate spectral replications introduced by extra samples inserted into the waveform.

Figure 6 shows the complete time-domain interpolation scheme.

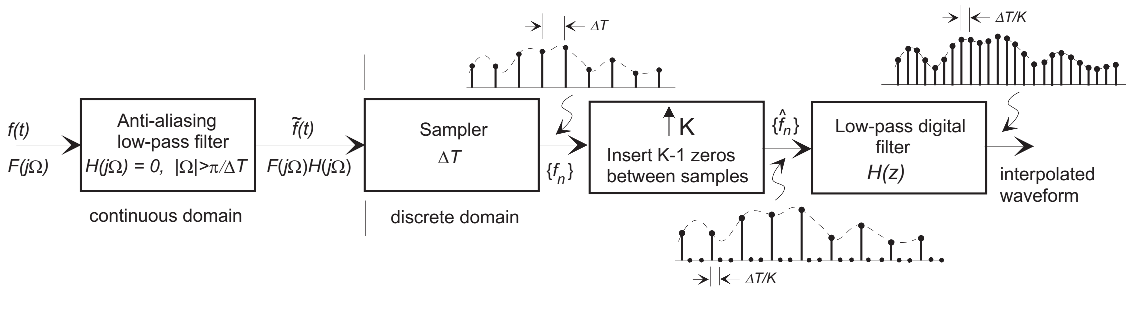


Figure 6: Real-time processing to interpolate a data stream by a factor K.

1.3 The Interpolation Filter

The ideal interpolation filter is a low-pass filter, operating at the high frequency rate, with a cut-off frequency at the Nyquist frequency of the sampler $\Omega_N = \pi/\Delta T$. The impulse response is therefore

$$h_n = \frac{\sin(\pi n/K)}{\pi n/K}$$

which is clearly acausal. An FIR approximation to the ideal interpolator is obtained by truncating h_n to a finite length, say $N = 2KM + 1$, and delaying it by KM samples so that

$$h_n = \frac{\sin\left(\pi\left(n - KM\right)/K\right)}{\pi\left(n - KM\right)/K} \qquad n = 0, 1, 2 \ldots N - 1$$

and windowing with an appropriate window, for example a Kaiser or Hamming window. The transfer function may be written

$$H(z) = \sum_{n=0}^{M-1} b_n z^{-n}.$$

Figure 7 shows the complete interpolation scheme for an up-sampling factor $K = 3$.

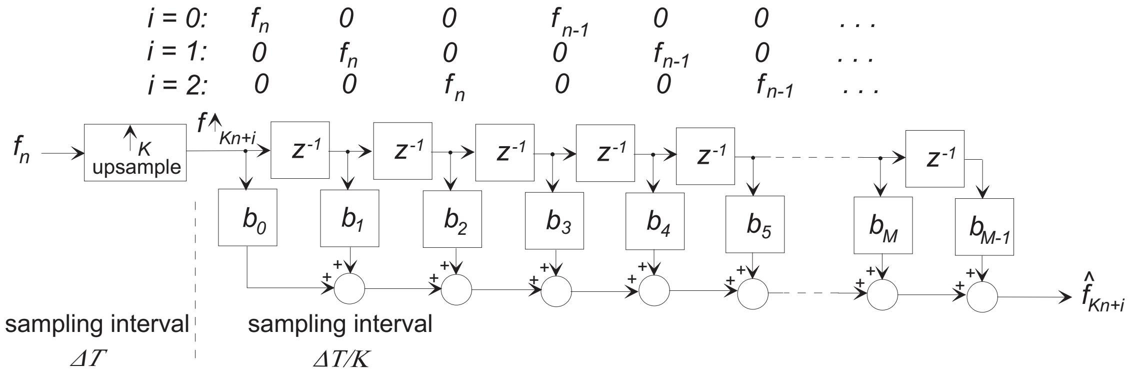


Figure 7: Complete interpolation scheme (K=3), with an FIR interpolation filter.

1.4 Polyphase Interpolation Filtering

A substantial economy of computation may be achieved by the use of a polyphase filter structure. As can be seen from Fig. 7, at each filter step many of the up-sampled data values are zero, and do not contribute to the filter output. Therefore at each of the computation steps only a subset of the filter coefficients are required. This leads to a structure involving a set of shorter filters applied the original data sequence.

(1) Design an FIR interpolation filter of length M, based upon the up-sampling factor K.

(2) Form K "sub-filters" from the filter coefficients b_m, $m = 0 \ldots M - 1$, by grouping them as follows:

Sub-filter 0:	b_0	b_K	b_{2K}	b_{3K}	$\cdots$
Sub-filter 1:	b_1	b_{K+1}	b_{2K+1}	b_{3K+1}	$\cdots$
Sub-filter 2:	b_2	b_{K+2}	b_{2K+2}	b_{3K+2}	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Sub-Filter K-1:	b_{K-1}	b_{2K-1}	b_{3K-1}	b_{4K-1}	$\cdots$

(3) At each major time step the interpolated data stream may be created by passing the data sequence $\{f_n\}$ through each of the sub-filters, so that the interpolated value $\hat{f}_{Kn+i}$ is the output of sub-filter i. The polyphase structure is shown in Fig. 8

Note that it is not necessary to perform the up-sampling step of inserting zero data points into the data steam with this method.

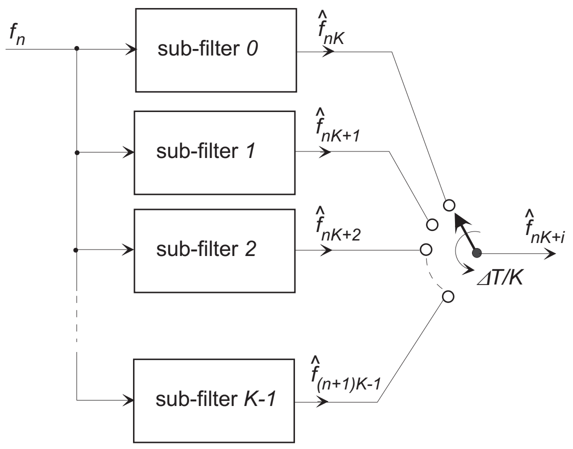


Figure 8: Polyphase filtering interpolation scheme.

2 Down-sampling (Decimation) by an Integer Factor

Decimation by an integer factor K is the reverse of interpolation, that is increasing the sampling interval T by an an integer factor (or decreasing the sampling frequency). At first glance the process seems to be simple: simply retain every Kth sample from the data sequence so that $\tilde{f}_n = f_{nK}$ where $\{\tilde{f}_n\}$ is the down-sampled sequence, as shown in Fig. 9. Caution must be taken however to prevent aliasing in the decimated sequence. It is not valid to directly down-sample a sequence directly unless it is known a-priori that the spectrum of the data set is identically zero at frequencies at and above the Nyquist frequency defined by the lower sampling frequency.

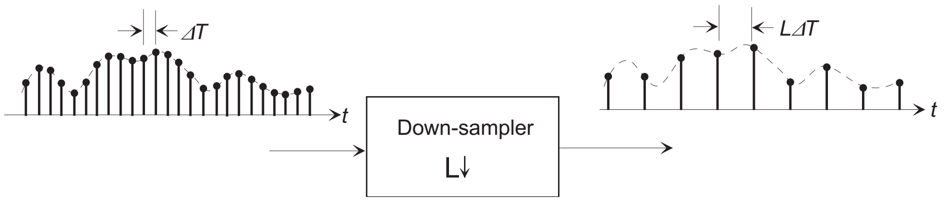


Figure 9: Basic temporal down-sampling by an integer factor L. Every Lth sample of the input sequence is retained in the output; all other data points are "thrown away".

In discussing sampling of continuous waveforms, we described the use of a pre-aliasing filter to eliminate (or at least significantly reduce) spectral components that would introduce aliasing into the sampled data set. When down-sampling, the digital equivalent is required: a digital low-pass filter is used to eliminate all spectral components that would cause aliasing in the resampled data set. The complete down-sampling scheme is shown in Fig. 10.

3 Up and Down-sampling with a non-integer factor

Assume that the goal is to re-sample a sequence by a non-integer factor p that can be expressed as a rational fraction, that is

$$P = \frac{N}{M}$$

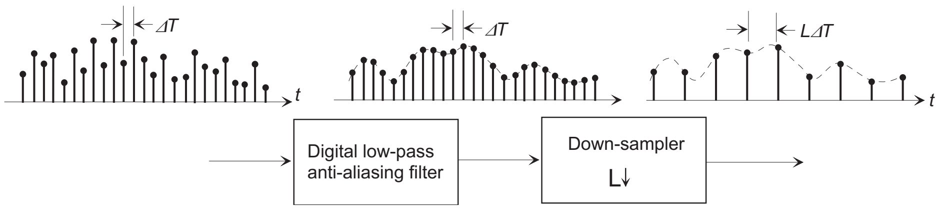
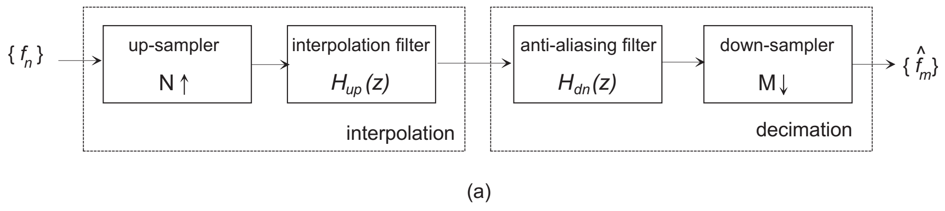


Figure 10: Temporal down-sampling by an integer factor L with a digital low-pass filter to eliminate components in the input sequence that would cause aliasing through the down-sampling operation.

where N and M are positive integers. This can be achieved by (1) interpolation by a factor N, followed by (2) decimation by a factor M, as shown in Fig. 11(a). However, since the two low-pass filters are cascaded, they may be replaced with a single filter with a cut-off frequency that is the lower of the two filters as is shown in Fig. 11(b).



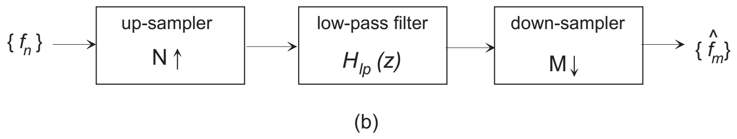


Figure 11: Combined interpolation/decimation to achieve re-sampling by a non-integer factor. In (a) an interpolator and decimator have been connected in series. In (b) the two low-pass filters have been replaced by a single filter with a cut-off trequency that is the lower of the two filters in (a).

4 Matlab Functions for Resampling

`Y = RESAMPLE(X,P,Q)` resamples the sequence in vector X at P/Q times the original sample rate using a polyphase implementation. Y is P/Q times the length of X (or the ceiling of this if P/Q is not an integer). P and Q must be positive integers.

`Y = UPFIRDN(X,H,P,Q)` is a cascade of three systems applied to input signal X:

(1) Upsampling by P (zero insertion). P defaults to 1 if not specified.

(2) FIR filtering with the filter specified by the impulse response given in H.

(3) Downsampling by Q (throwing away samples). Q defaults to 1 if not specified.

`UPFIRDN` uses an efficient polyphase implementation.

`Y = INTERP(X,R)` resamples the sequence in vector X at R times the original sample rate. The resulting resampled vector Y is R times longer, LENGTH(Y) = R*LENGTH(X).

`Y = DECIMATE(X,R)` resamples the sequence in vector X at 1/R times the original sample rate. The resulting resampled vector Y is R times shorter, i.e., LENGTH(Y) = CEIL(LENGTH(X)/R). By default, DECIMATE filters the data with an 8th order Chebyshev Type I low-pass filter with cutoff frequency .8*(Fs/2)/R, before resampling.

`Y = DOWNSAMPLE(X,N)` downsamples input signal X by keeping every N^{th} sample starting with the first. If X is a matrix, the down-sampling is done along the columns of X.

`Y = UPSAMPLE(X,N)` upsamples input signal X by inserting N-1 zeros between input samples. X may be a vector or a signal matrix (one signal per column).

Appendix A: A Matlab Function for DFT Based Interpolation.

```
%------------------------------------------------------------------------------
% 2.161 MATLAB Example: Data interpolation (up-sampling) using the DFT
%
%  Author: D. Rowell   4/14/07
%
%  Usage:  fout = dftupsample(f, K)
%          where f is an input data record (real) of length N,
%          fout is the output record of length KN, and K is the integer
%          interpolation factor, that is (K-1) interpolated data
%          points will be inserted between each of the original data points.
%
%  Note:   MATLAB does not require its FFT routine to be restricted to radix-2
%          lengths.   Therefore this function may be used for arbitrary length
%          records f and interpolation factors K.
%
%-----------------------------------------------------------------------------
%
function fout = dftupsample(fin,K)
%
N = length(fin);
% Take the DFT
Fin = fft(fin);
% Create an extended empty array, and fill the ends with the upper and lower
% halves of the DFT. (Note that MATLAB arrays run from 1 to N)
Fout = zeros(length(Fin)*K,1);
Fout(1:N/2) = Fin(1:N/2);
Fout(N*K - N/2+1: N*K) = Fin(N/2+1:N);
% Take the inverse DFT, scale by K, and preserve the real part
fout = real(ifft(Fout))*K;
%-----------------------------------------------------------------------------
```

Example: The following MATLAB fragment creates a data record of length 8, then uses `dftupsample()` to up-sample by a factor of $K = 11$:

```
>> i=[0:7];
>> f8=cos(2*pi*(i-1)/8) + 3*cos(4*pi*(i-1)/8)+ sin(6*pi*(i-1)/8) ;
>> f88 = dftupsample(f8,11);
>> subplot(2,1,1), stem([0:1:7],f8)
>> subplot(2,1,2), stem([0:1/11:87/11],f88)
>>
```

Plots of the input and output data sets are shown in Fig. 12.

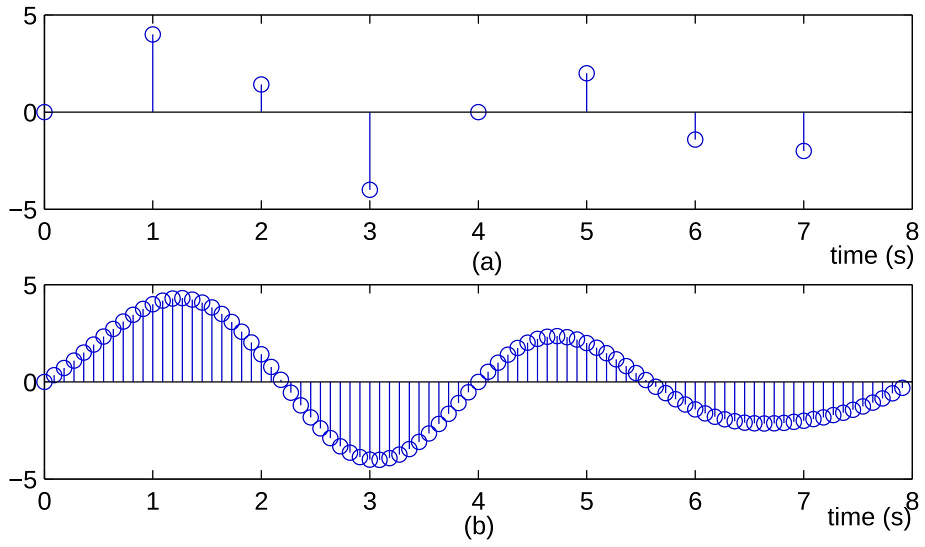


Figure 12: Matlab example: (a) input data sequence of length 8, and (b) output data sequence of length 88.