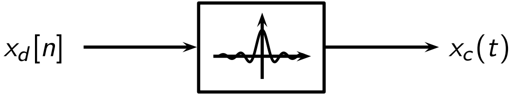
COM-202: Signal Processing

Chapter 7.c: multirate signal processing

sampling and interpolation

Sinc interpolation with timebase T

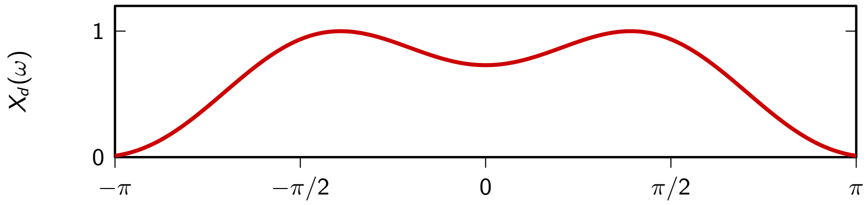


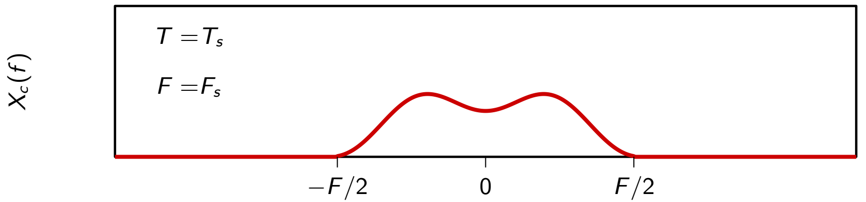
$$T = 1/F$$

$$x_c(t) = \sum_{n=-\infty}^{\infty} x_d[n] \operatorname{sinc}\left(\frac{t - nT}{T}\right)$$

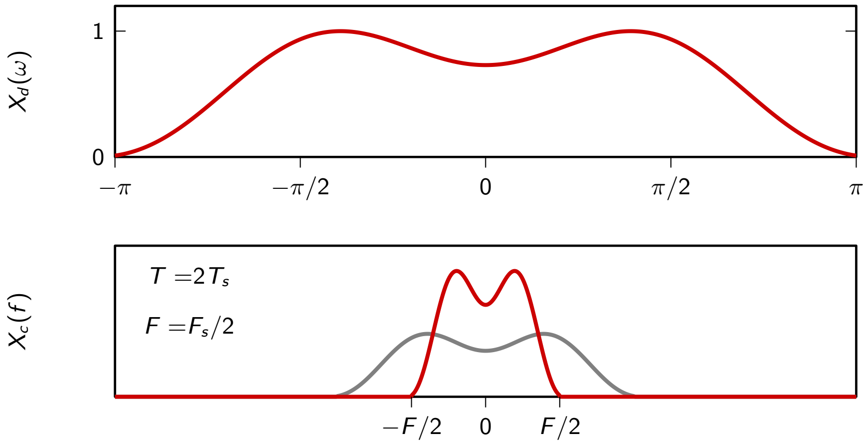
$$X_c(f) = \frac{1}{F} X_d\left(2\pi \frac{f}{F}\right) \operatorname{rect}\left(\frac{f}{F}\right) \in F\text{-BL}$$

Spectrum of interpolated signals

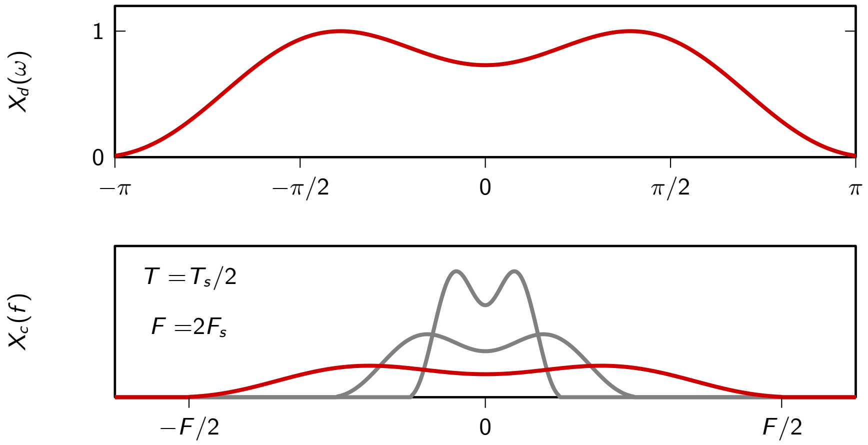




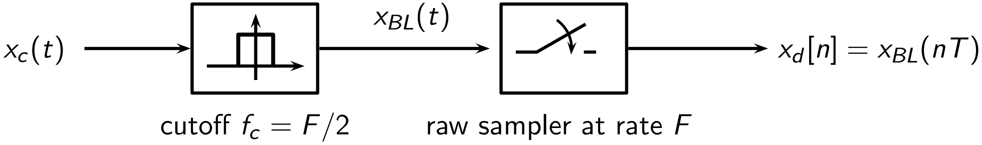
Spectrum of interpolated signals



Spectrum of interpolated signals



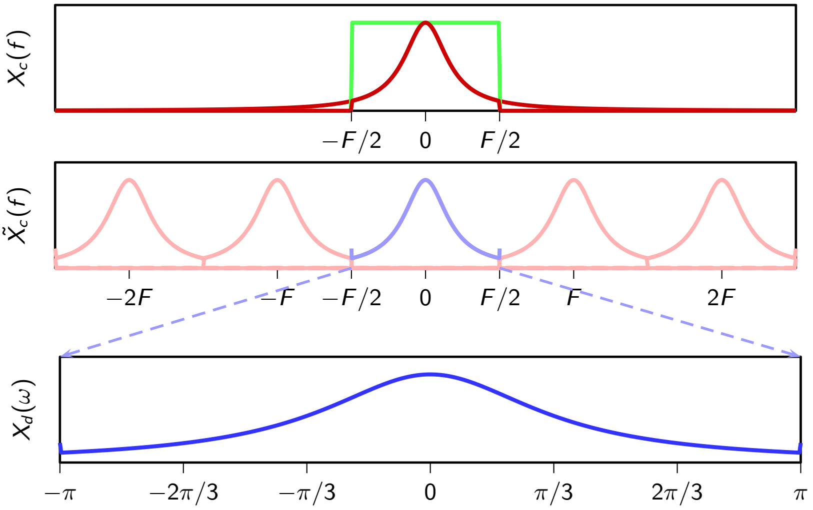
Sinc sampling with frequency F



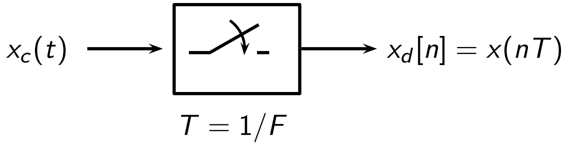
$$x_d[n] = \left\langle \operatorname{sinc}\left(\frac{t - nT}{T}\right), x_c(t) \right\rangle$$

$$X_d(\omega) = F\, X_c\left(F\left[\frac{\omega}{2\pi}\right]_{-1/2}^{+1/2}\right)$$

Sinc sampling includes an implicit antialiasing filter



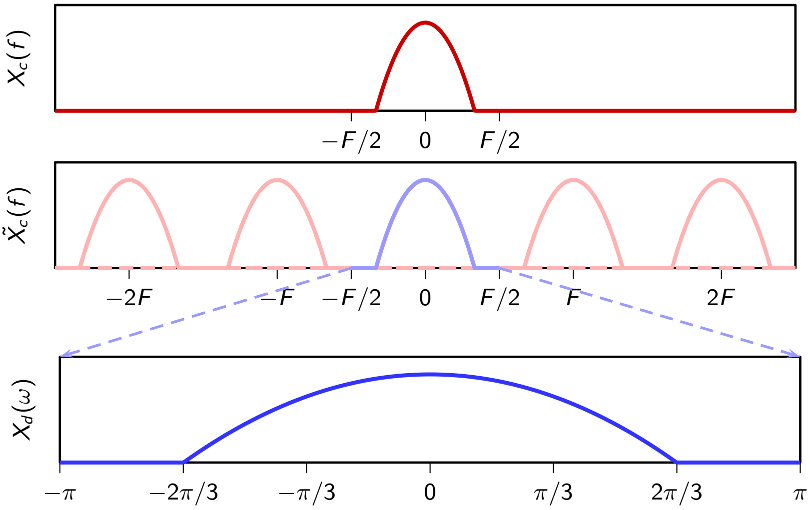
Raw sampling with frequency F



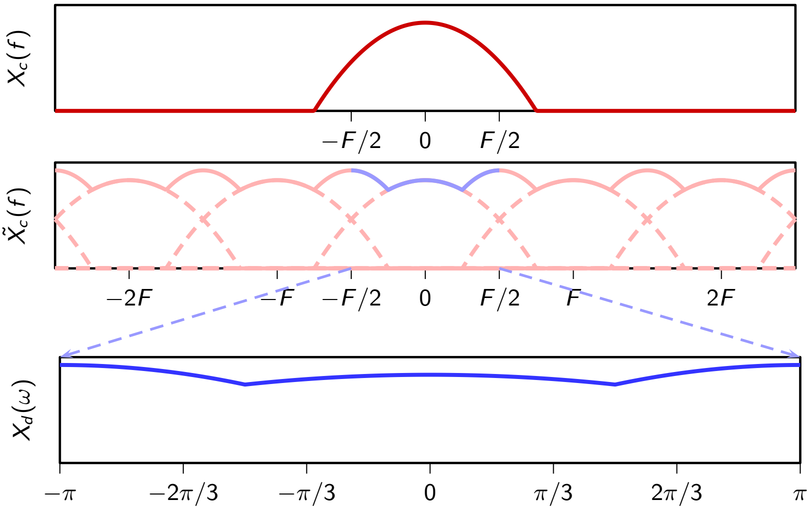
$$x_d[n] = x_c(nT)$$

$$X_d(\omega) = F \sum_{k=-\infty}^{\infty} X_c\left(\frac{\omega}{2\pi}F - kF\right)$$

Example: signal F_s-bandlimited and rate $F > F_s$

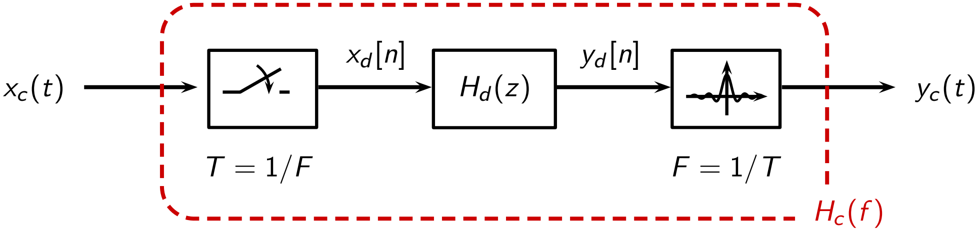


Example: signal F_s-bandlimited and rate $F < F_s$



discrete-time processing of analog signals

Equivalent analog response: basic setup

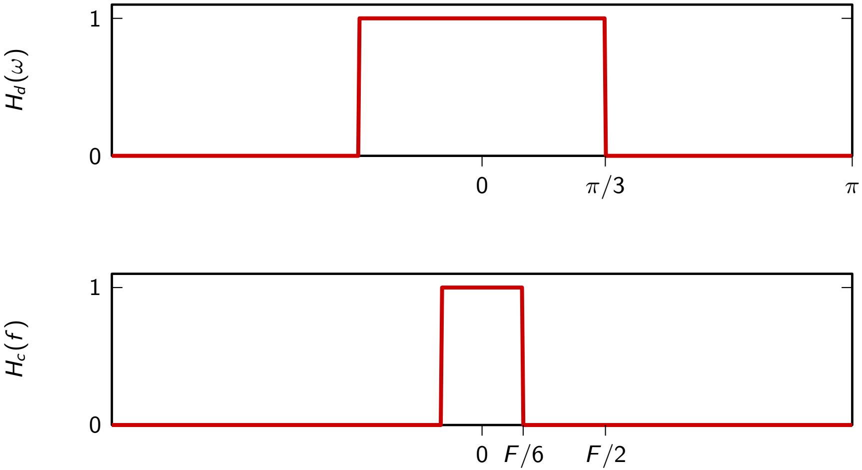


assume $x_c(t)$ is F_S-BL and $F > F_s$

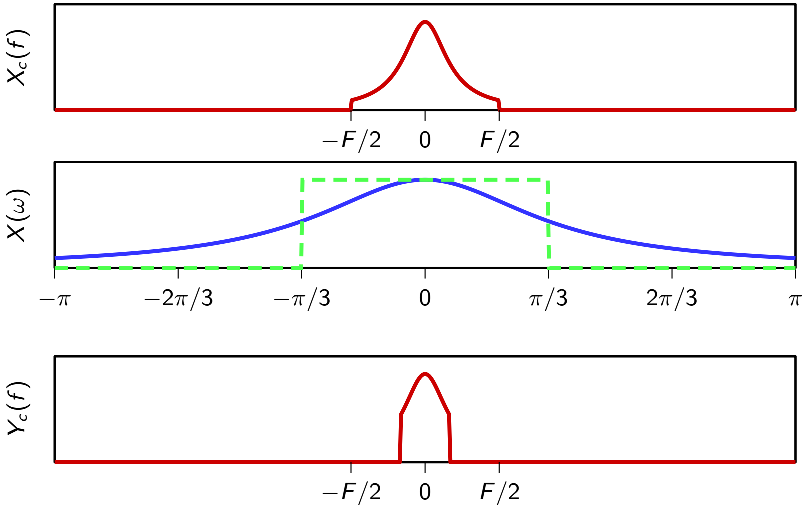
- $X_d(\omega) = F\,X_c\left(F\left[\frac{\omega}{2\pi}\right]_{-1/2}^{+1/2}\right), \qquad Y_d(\omega) = H_d(\omega)\,X_d(\omega)$
- $Y_c(f) = (1/F)Y_d(2\pi f/F)\,\mathrm{rect}(f/F) = H_d\,(2\pi f/F)\,X_c(f)$

$$H_c(f) = H_d\left(2\pi\frac{f}{F}\right)$$

Equivalent analog response



DT processing of CT signals



Example: analog bandpass with digital processing

- we want to implement a bandpass filter to select frequencies from 1 kHz to 2 kHz
- input signals are bandlimited with max positive frequency $F_N = 4\ kHz$
- we want to use digital processing

Example: analog bandpass with digital processing

analog bandpass filter:

- filter passband is $2f_c = 1$ kHz ($f_c = 500$ Hz)
- filter center frequency is $f_0 = 1500$ Hz

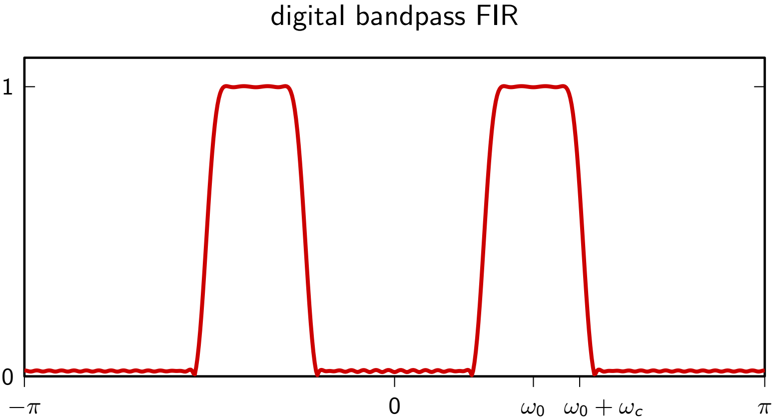
discrete-time processing chain

- input is 8 kHz-BL so we can use a sampling frequency $F_s = 8$ kHz
- design a FIR lowpass with cutoff $\omega_c = 2\pi(f_c/F_s)$
- modulate the impulse respose with $\omega_0 = 2\pi(f_0/F_s)$

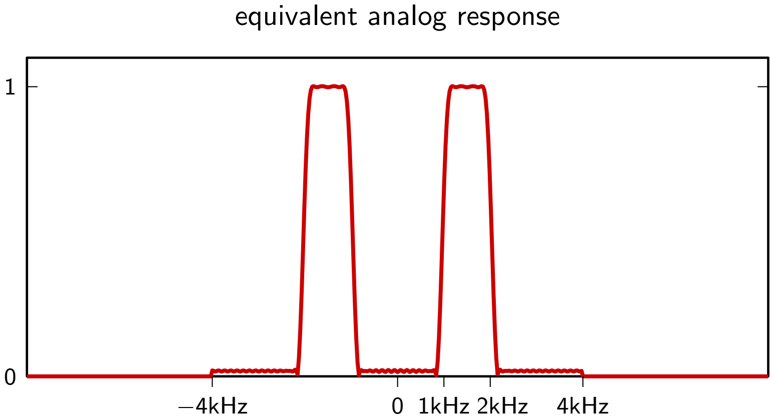
Example: analog bandpass with digital processing

```
import scipy.signal as sp

fc, f0, Fs = 500, 1500, 8000
wc, w0 = fc / Fs, f0 / Fs

N = 61
tbp = 0.2 # 20% transition band

h = sp.signal.remez(N, [0, wc*(1-tbp), wc*(1+tbp), 0.5], [1, 0], weight=[10, 1])
h *= 2 * np.cos(2 * np.pi * w0 * np.arange(len(h)))
```

Example: analog bandpass with digital processing

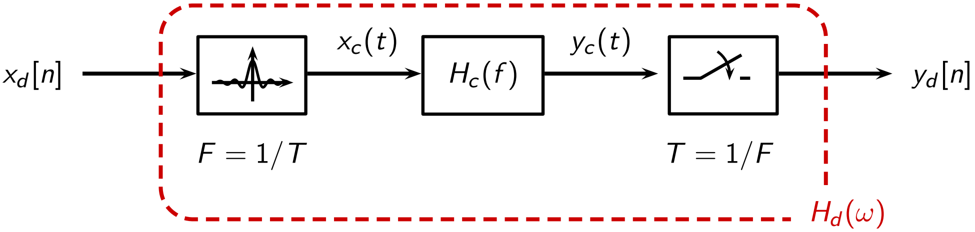


Example: analog bandpass with digital processing



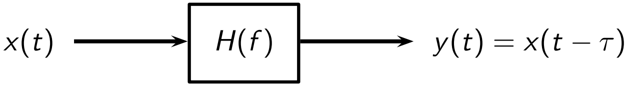
two more ideal filters

Dual setup



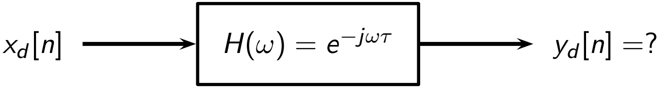
- $X_c(f) = (1/F)X_d(2\pi f/F)\,\mathrm{rect}(f/F)$
- $Y_c(f) = H_c(f)X_c(f)$
- $Y_d(\omega) = FY_c(\frac{\omega}{2\pi}F) = H_c(\frac{\omega}{2\pi}F)X_d(\omega)$
- $H_d(\omega) = H_c(\frac{\omega}{2\pi}F)$

Delays in continuous time



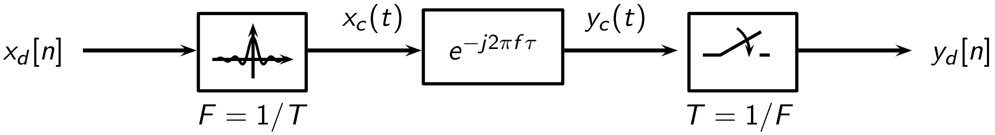
- in continuous time, delays are well defined for all $\tau \in \mathbb{R}$
- $H(f) = e^{-j2\pi f \tau}$

Delays in discrete time



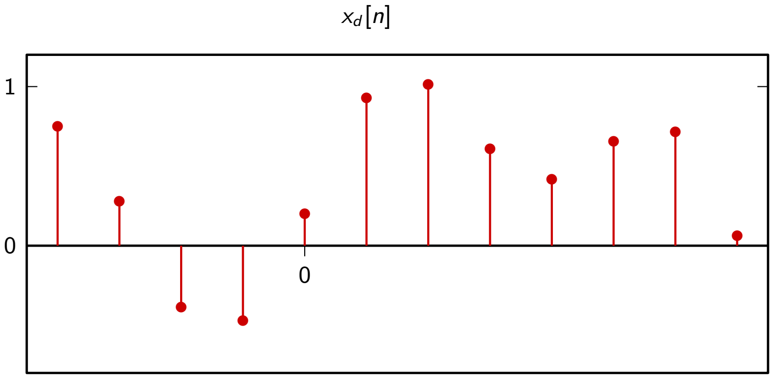
- when $\tau \in \mathbb{Z}$, then $y[n] = x[n - \tau]$
- what happens when τ is not an integer?

Interpretation by duality

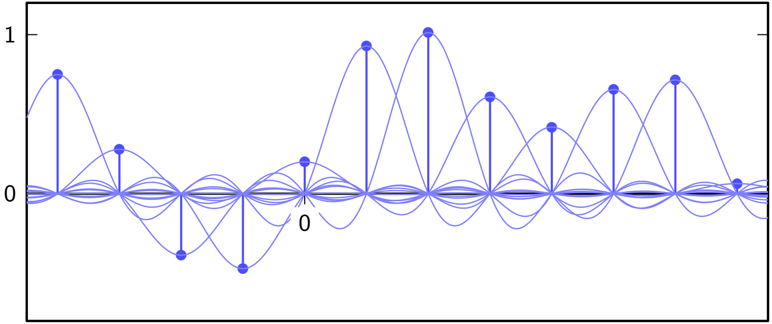


- a discrete-time delay could be implemented via interpolation, delay, and resampling
- equivalent filter: $H_d(\omega) = H_c(\omega/(2\pi)F) = e^{-j\omega\sigma}$ with $\sigma = \tau/T \in \mathbb{R}$
- impulse response: $h[n] = \text{sinc}(n - \sigma)$
- if $\sigma \in \mathbb{Z}$ then $h[n] = \delta[n - \sigma]$ (normal delay) otherwise we have an ideal filter!

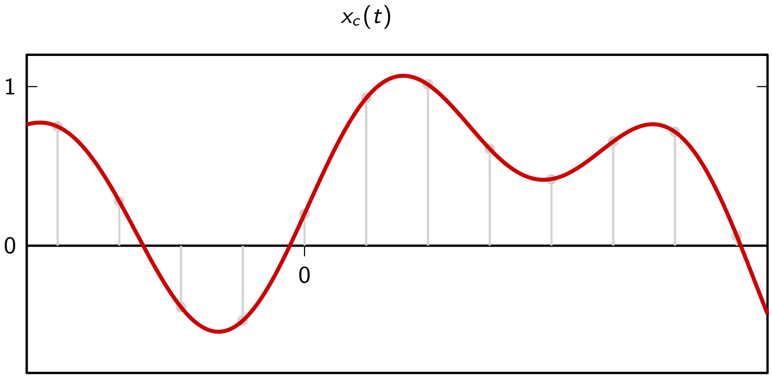
Fractional delay



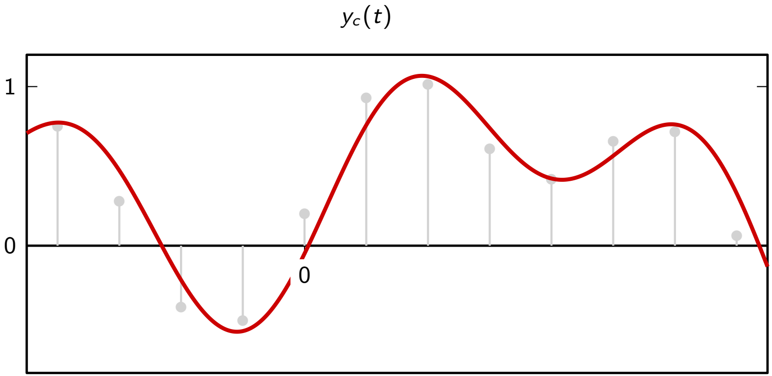
Fractional delay



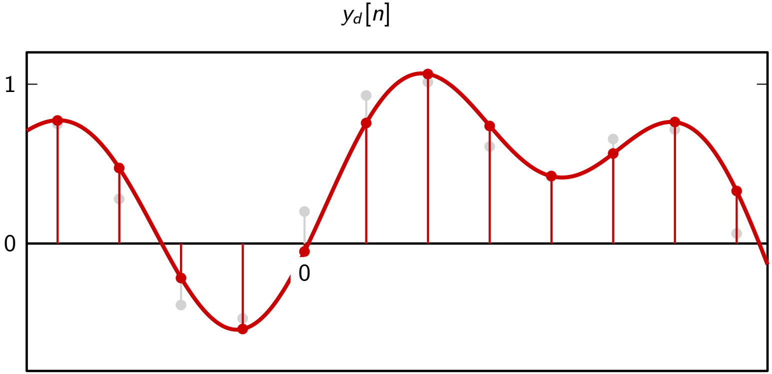
Fractional delay



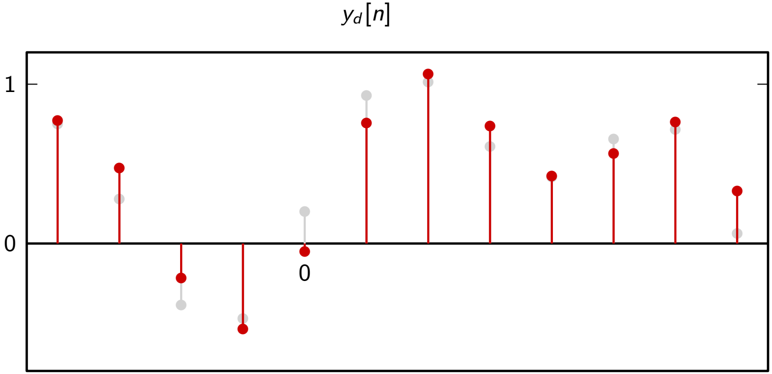
Fractional delay



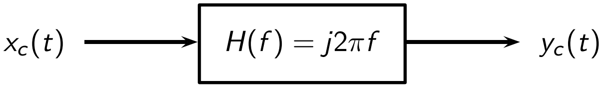
Fractional delay



Fractional delay

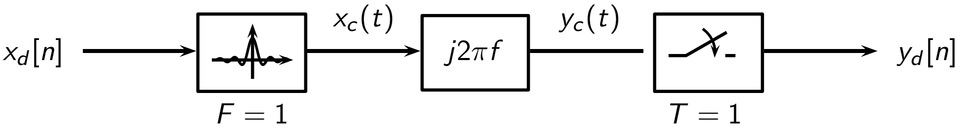


Differentiation in continuous time



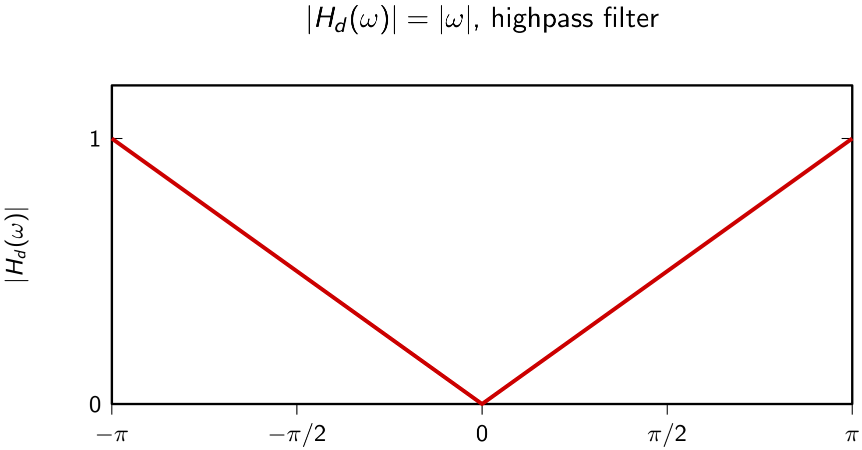
- easy to show that $y_c(t) = x_c'(t) = \frac{\partial}{\partial t} x_c(t)$
- first derivative can be computed exactly via filtering

By duality



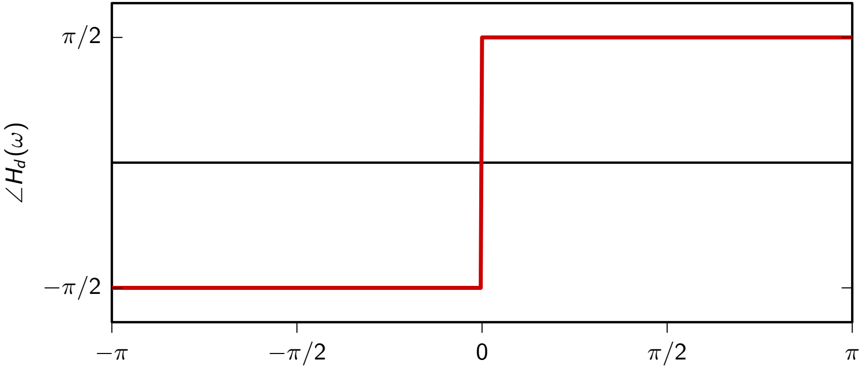
- chain interpolates the discrete-time input, differentiates the interpolation and resamples it
- equivalent filter $H_d(\omega) = H_c(\omega/(2\pi)) = j\omega$
- $H_d(\omega)$ is a “digital differentiator”

Digital differentiator, magnitude response



Digital differentiator, phase response

$$\angle H_d(\omega) = (\pi/2)\,\mathrm{sign}(\omega)$$

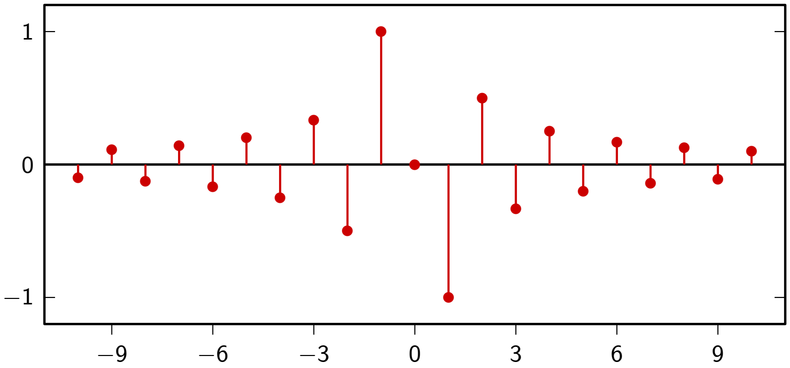


Digital differentiator, impulse response

$$
\begin{aligned}
h_d[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} j\omega e^{j\omega n} d\omega \\
&= \ldots \textit{(integration by parts)} \ldots \\
&= \begin{cases} 0 & n = 0 \\ \dfrac{(-1)^n}{n} & n \neq 0 \end{cases}
\end{aligned}
$$

the differentiator is an ideal filter

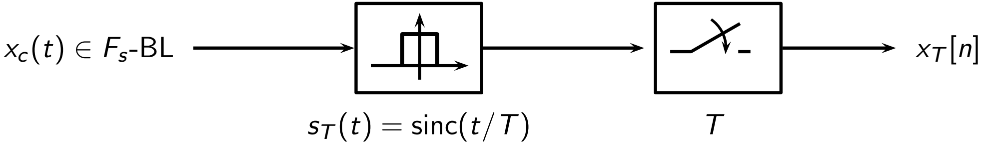
Digital differentiator, impulse response



multirate signal processing

Changing the sampling rate of a discrete-time signal

sinc sampling:

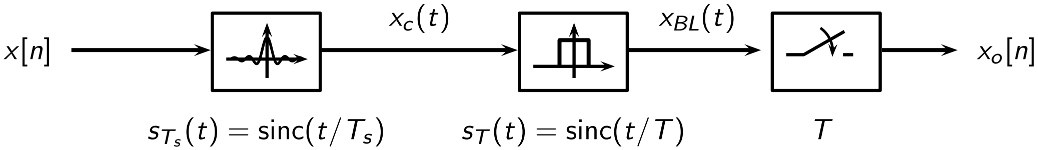


- for $T = T_s = 1/F_s$, $x_{T_s}[n] = x[n] = T_s x(nT_s)$
- given an arbitrary value of T can we convert $x[n]$ into $x_T[n]$?
- can we do this entirely in discrete time?

Decimation and interpolation

- sampling rate change factor: $\alpha = T/T_s$
- *decimation*: when $T > T_s$ there will be *fewer* output samples than input samples ($\alpha > 1$)
- *interpolation*: when $T < T_s$ there will be *more* output ($\alpha < 1$)
- we can always interpolate safely but decimation may cause loss of information

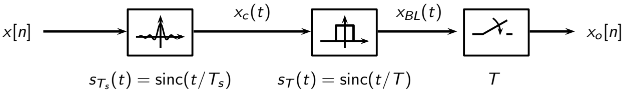
If we went back to continuous time...



$$x_o[n] = x_{BL}(nT) = (x * s_T)(nT)$$

$$x_c(t) = \sum_{k=-\infty}^{\infty} x[k] s_{T_s}(t - nT_s)$$

If we went back to continuous time...



$$
\begin{aligned}
x_o[n] = (x_c * s_T)(nT) &= \int_{-\infty}^{\infty} x_c(\tau) s_T(nT - \tau) d\tau \\
&= \int_{-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x[k] s_{T_s}(\tau - kT_s) \right) s_T(nT - \tau) d\tau \\
&= \sum_{k=-\infty}^{\infty} x[k] \int_{-\infty}^{\infty} s_{T_s}(\tau - kT_s) s_T(nT - \tau) d\tau \\
&= \sum_{k=-\infty}^{\infty} x[k] \int_{-\infty}^{\infty} s_{T_s}(t) s_T(nT - kT_s - t) dt
\end{aligned}
$$

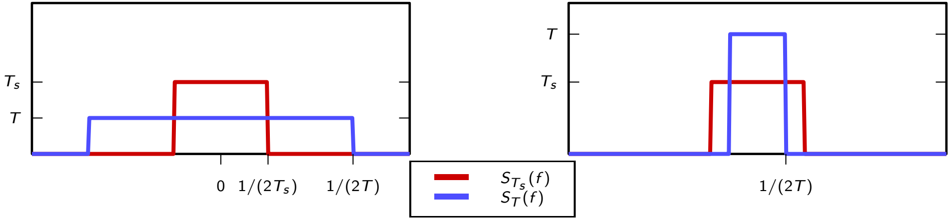
Two competing lowpass filters

$$\int_{-\infty}^{\infty} s_{T_s}(\tau) s_T(t-\tau) d\tau = (s_{T_s} * s_T)(t)$$

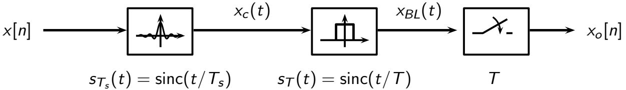
$$s_T(t) \xleftrightarrow{\text{CTFT}} T\,\text{rect}(Tf) = S_T(f)$$

interpolation $(T < T_s)$

decimation $(T > T_s)$



If we went back to continuous time...



$$x_o[n] = \sum_{k=-\infty}^{\infty} x[k] \operatorname{sinc}\left(\frac{nT - kT_s}{\max\{T_s, T\}}\right)$$

$$= \begin{cases} \sum_{k=-\infty}^{\infty} x[k] \operatorname{sinc}(\alpha n - k) & \alpha = T/T_s < 1 \quad \textit{(interpolation)} \\ \sum_{k=-\infty}^{\infty} x[k] \operatorname{sinc}(n - k/\alpha) & \alpha = T/T_s > 1 \quad \textit{(decimation)} \end{cases}$$

good news: we can do this entirely in discrete time!

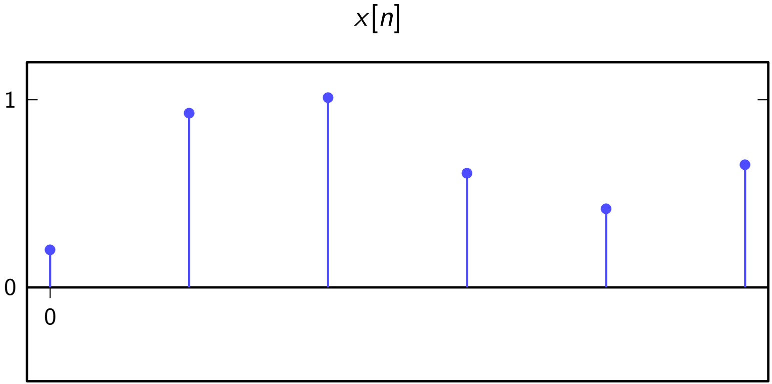
Interpolation by an integer factor

$$T = T_s/N$$

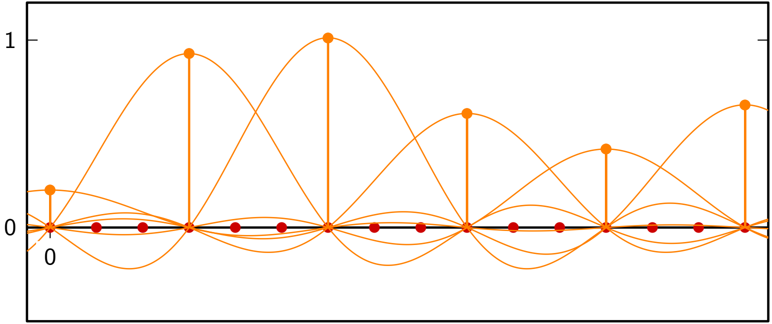
$$\alpha = 1/N$$

$$x_o[n] = \sum_{k=-\infty}^{\infty} x[k] \operatorname{sinc}\left(\frac{n - kN}{N}\right)$$

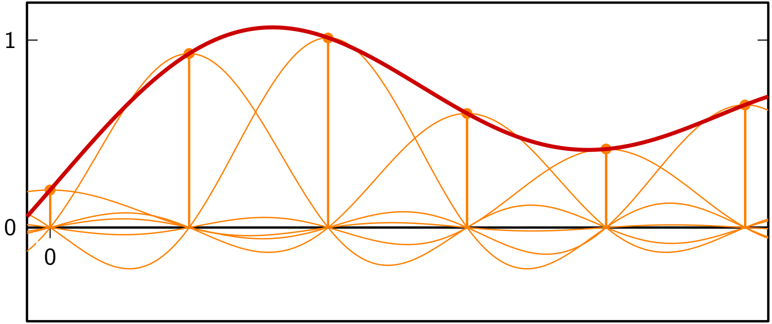
Example: increasing the sampling rate by a factor of 3



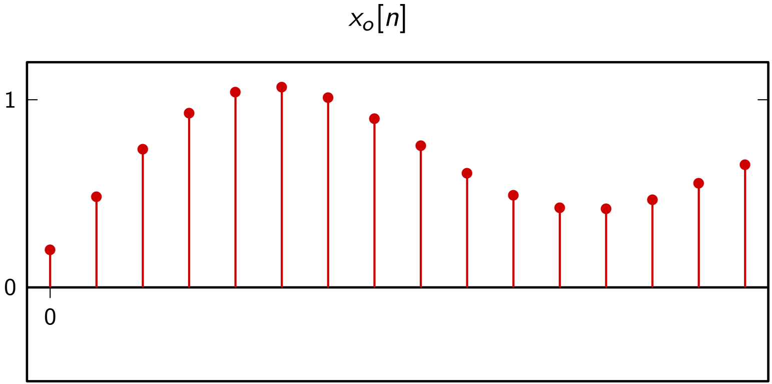
Example: increasing the sampling rate by a factor of 3



Example: increasing the sampling rate by a factor of 3

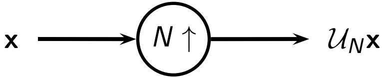


Example: increasing the sampling rate by a factor of 3



The upsampling operator

$$(\mathcal{U}_N \mathbf{x})[n] = \begin{cases} x[n/N] & \text{if } n \text{ is a multiple of } N \\ 0 & \text{otherwise.} \end{cases}$$

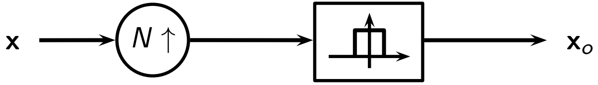


Interpolation by an integer factor

$$
\begin{aligned}
x_o[n] &= \sum_{k=-\infty}^{\infty} x[k] \operatorname{sinc}\left(\frac{n-kN}{N}\right) \\
&= \sum_{k=-\infty}^{\infty} (\mathcal{U}_N \mathbf{x})[kN] \operatorname{sinc}\left(\frac{n-kN}{N}\right) \\
&= \sum_{m=-\infty}^{\infty} (\mathcal{U}_N \mathbf{x})[m] \operatorname{sinc}\left(\frac{n-m}{N}\right) \\
&= (\mathcal{U}_N \mathbf{x} * \mathbf{s}_N)[n]
\end{aligned}
$$

- $\mathbf{s}_N$ is the impulse response of an ideal discrete-time lowpass with cutoff $\omega_c = \pi/N$
- interpolation can be performed entirely in discrete time!

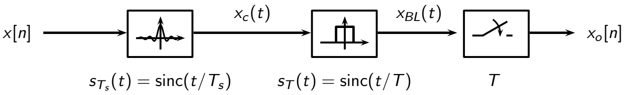
Interpolation by an integer factor in discrete time



cutoff $\omega_c = \pi / N$

- $\mathbf{x}_o = \mathbf{s}_N * \mathcal{U}_N \mathbf{x}$
- interpolation first introduces $N - 1$ zeros for every input sample and then fills the gaps via a lowpass filter
- in practice we use a realizable lowpass with cutoff $\omega_c = \pi / N$

If we went back to continuous time...



$$x_o[n] = \begin{cases} \sum_{k=-\infty}^{\infty} x[k]\,\mathrm{sinc}(\alpha n - k) & \alpha = T/T_s < 1 \quad \textit{(interpolation)} \\ \sum_{k=-\infty}^{\infty} x[k]\,\mathrm{sinc}(n - k/\alpha) & \alpha = T/T_s > 1 \quad \textit{(decimation)} \end{cases}$$

Decimation by an integer factor

$$T = NT_s$$

$$\alpha = N$$

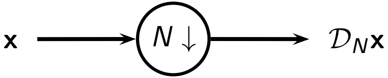
$$x_o[n] = \sum_{k=-\infty}^{\infty} x[k] \,\mathrm{sinc}\left(\frac{Nn - k}{N}\right)$$

$$= (\mathbf{x} * \mathbf{s}_N)[Nn]$$

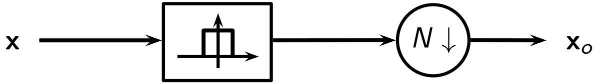
- $\mathbf{s}_N$ is the impulse response of an ideal discrete-time lowpass with cutoff $\omega_c = \pi/N$
- we discard $N - 1$ out of N filter output samples
- decimation can be performed entirely in discrete time!

The downsampling operator

$$(\mathcal{D}_N \mathbf{x})[n] = x[nN]$$



Decimation by an integer factor in discrete time

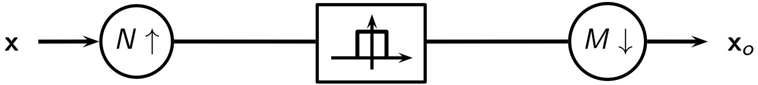


cutoff $\omega_c = \pi/N$

- $\mathbf{x}_o = \mathcal{D}_N(\mathbf{s}_N * \mathbf{x})$
- decimation first "bandlimits" the input and then discards $N-1$ samples out of N
- in practice we use a realizable lowpass with cutoff $\omega_c = \pi/N$
- if $x_c(t) \in (F_s/N)$-BL, then $\mathbf{s}_N * \mathbf{x} = \mathbf{x}$ and so $\mathbf{x}_o = \mathcal{D}_N \mathbf{x}$

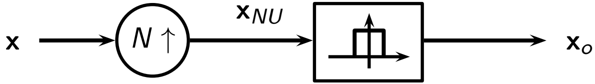
Rational Sampling Rate Change

$$\alpha = T/T_s = M/N$$



$$\omega_c = \min\{\pi/N, \pi/M\}$$

Interpolation by an integer factor in the frequency domain



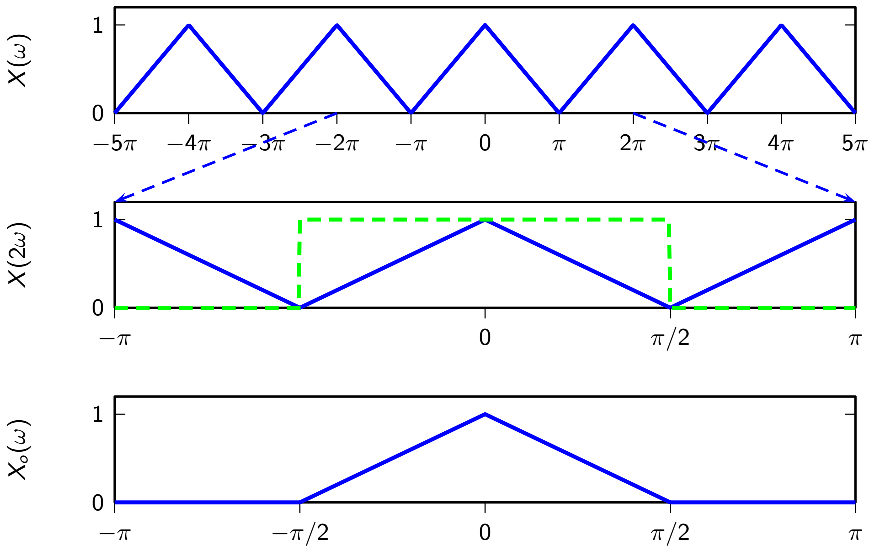
cutoff $\omega_c = \pi/N$

$$X_{NU}(\omega) = \sum_{n=-\infty}^{\infty} x_{NU}[n]e^{-j\omega n}$$

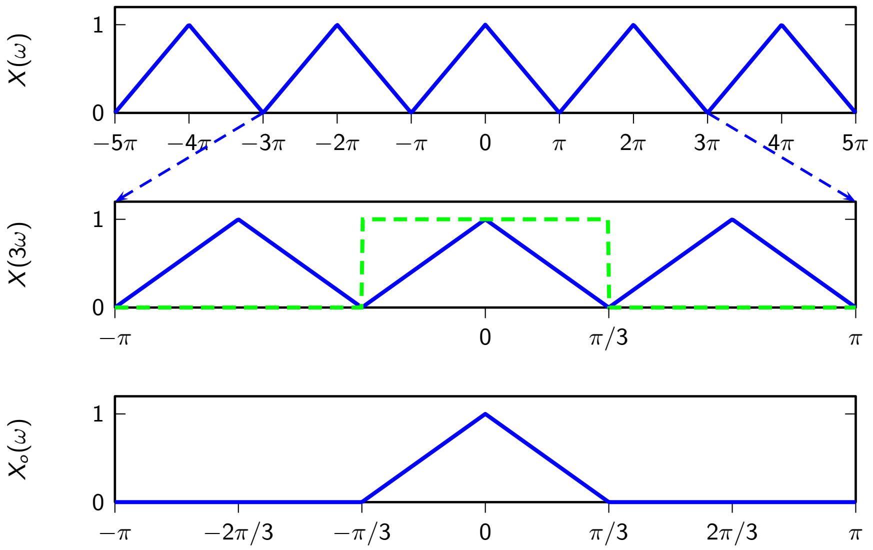
$$= \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega Nn} = X(N\omega)$$

$$X_o(\omega) = X(N\omega)\,\mathrm{rect}\left(\frac{N\omega}{2\pi}\right)$$

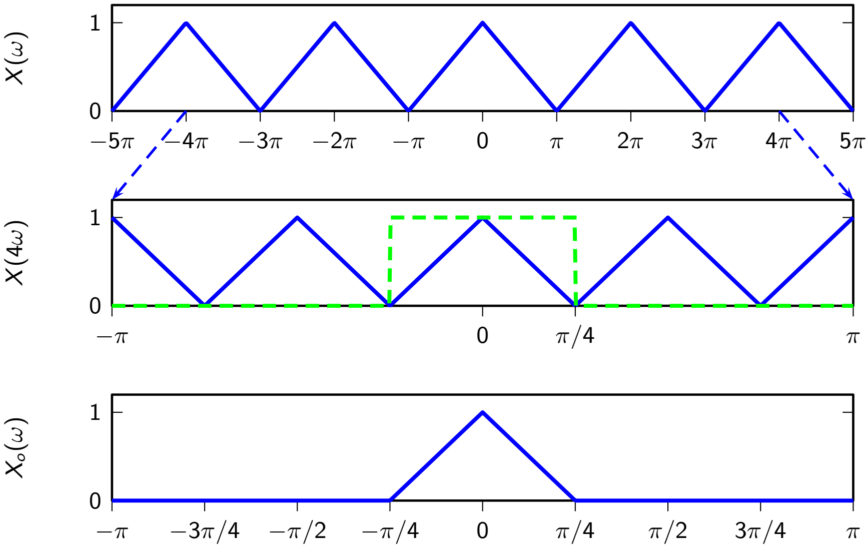
Interpolation by 2



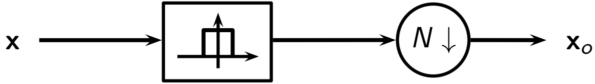
Interpolation by 3



Interpolation by 4



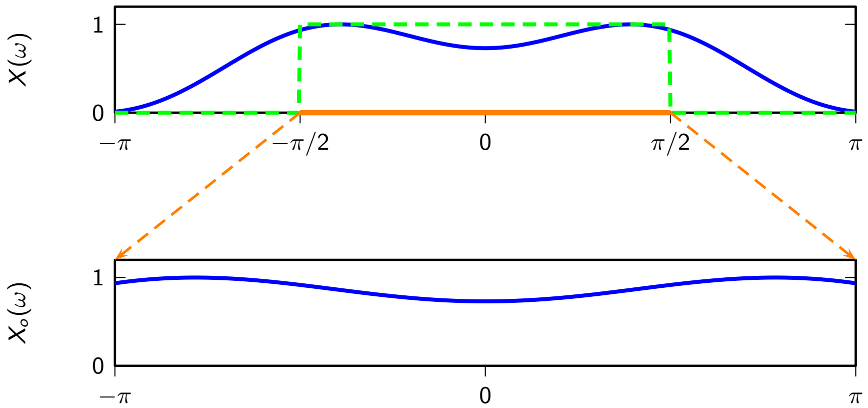
Decimation by an integer factor in the frequency domain



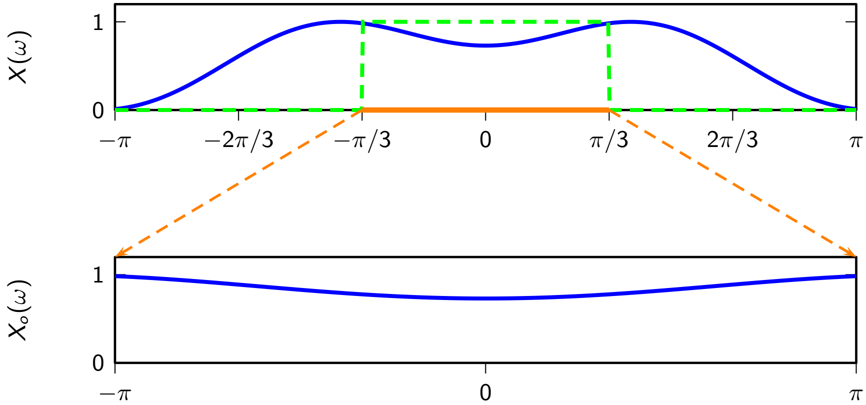
cutoff $\omega_c = \pi/N$

$$X_o(\omega) = X\left(\left[\frac{\omega}{N}\right]_{-\pi/N}^{+\pi/N}\right)$$

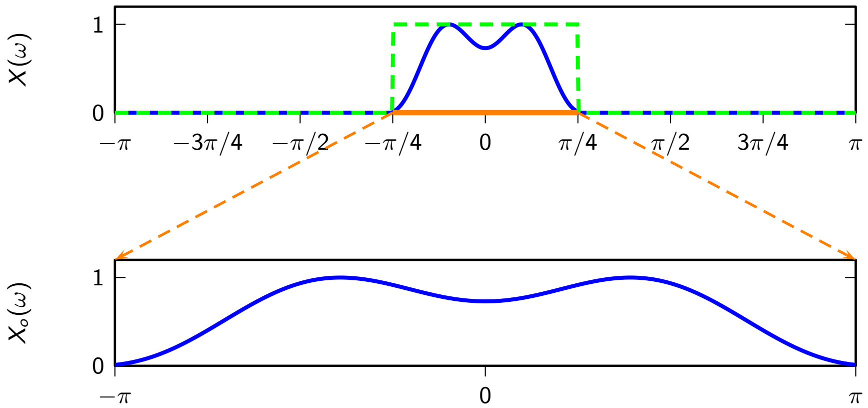
Decimation by 2



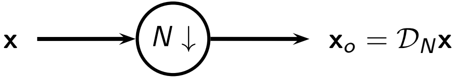
Decimation by 3



Decimation by 4



What happens if we just downsample (no lowpass)?

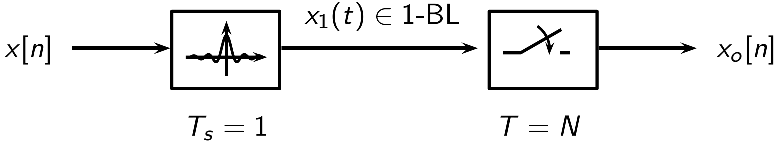


$$x_o[n] = x[nN] = x_c(nNT_s) = x_1(nN)$$

$$x_1(t) = x_c(t/T_s) \in \text{1-BL}$$

we're sampling a 1-BL signal with a sampling frequency $F = 1/N < 1$: aliasing

Downsampling and aliasing



periodization (with possible overlap) due to raw sampling at $F = 1/N$

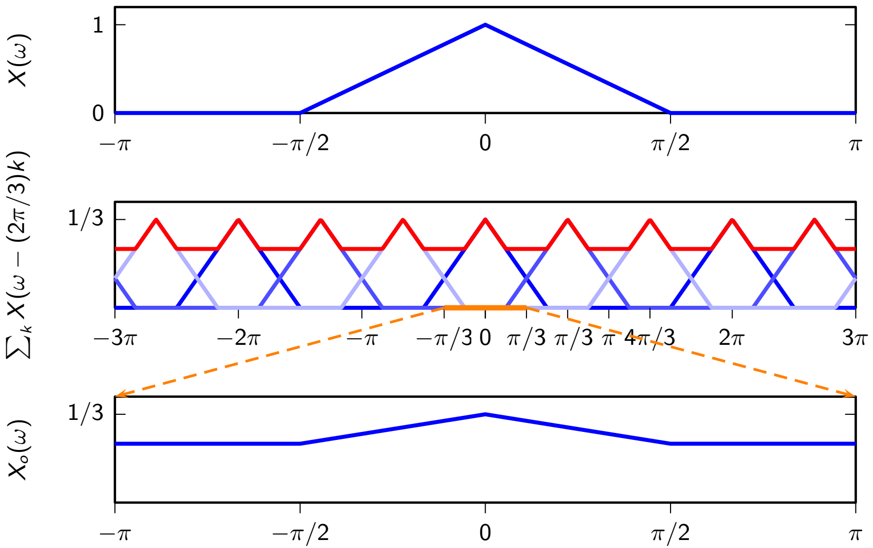
$$X_o(\omega) = \frac{1}{N} \sum_k X_1\left(\frac{\omega - 2\pi k}{2\pi N}\right)$$

spectrum of reconstructed 1-BL input: $X_1(f) = X(2\pi f)$

$$= \frac{1}{N} \sum_k X(2\pi f)|_{f=(\omega-2\pi k)/(2\pi N)}$$

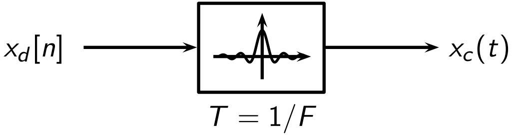
$$= \frac{1}{N} \sum_k X\left(\frac{\omega - 2\pi k}{N}\right)$$

Downsampling by 3, with aliasing



practical digital to analog interpolation methods

Sinc interpolation

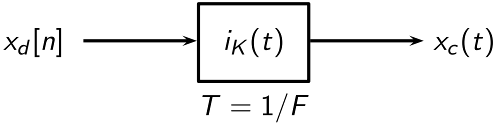


$$x_c(t) = \sum_{n=-\infty}^{\infty} x_d[n] \operatorname{sinc}\left(\frac{t - nT}{T}\right)$$

$$X_c(f) = \frac{1}{F} X_d\left(2\pi \frac{f}{F}\right) \operatorname{rect}\left(\frac{f}{F}\right) \in F\text{-BL}$$

- sinc interpolation cannot be implemented in practice (non-causal, ideal response)
- interpolation kernels are *analog* filters, i.e. expensive to build
- for cost reasons we can only use a low-order interpolator like the zero-order hold

Realistic continuous-time interpolation

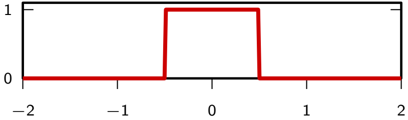
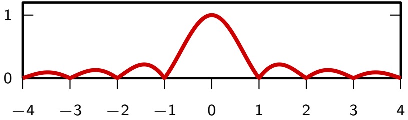
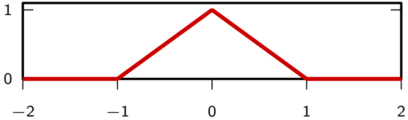
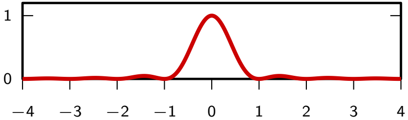
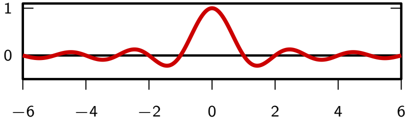
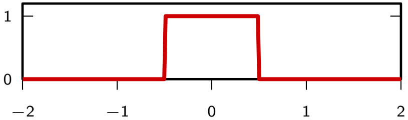


$$x_c(t) = \sum_{n=-\infty}^{\infty} x_d[n]\, i_K\left(\frac{t - nT}{T}\right)$$

$$i_K(t) \xleftrightarrow{\text{CTFT}} I_K(f)$$

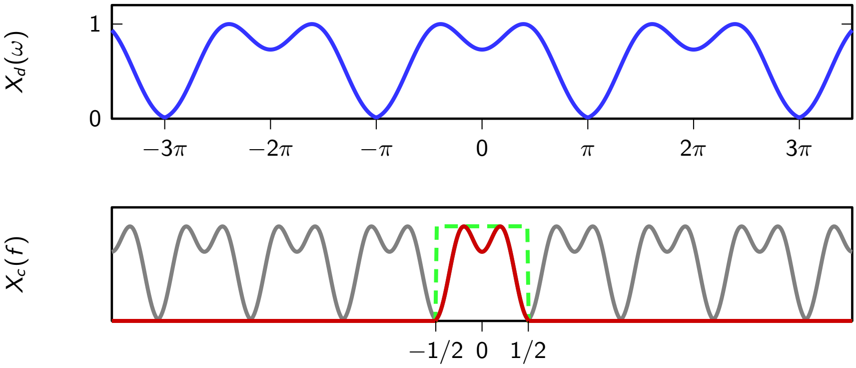
$$X_c(f) = \frac{1}{F}\, X_d\left(2\pi\frac{f}{F}\right)\, I_K\left(\frac{f}{F}\right)$$

Interpolation kernels

K	$i_k(t)$	$I_K(f)$
0	$i_0(t) = \text{rect}(t)$	$I_0(f) = \text{sinc}(f)$
		
1	$i_1(t) = 1 - \lvert t \rvert$	$I_0(f) = \text{sinc}^2(f)$
		
. . .		
∞	$i_\infty(t) = \text{sinc}(t)$	$I_\infty(f) = \text{rect}(f)$
		

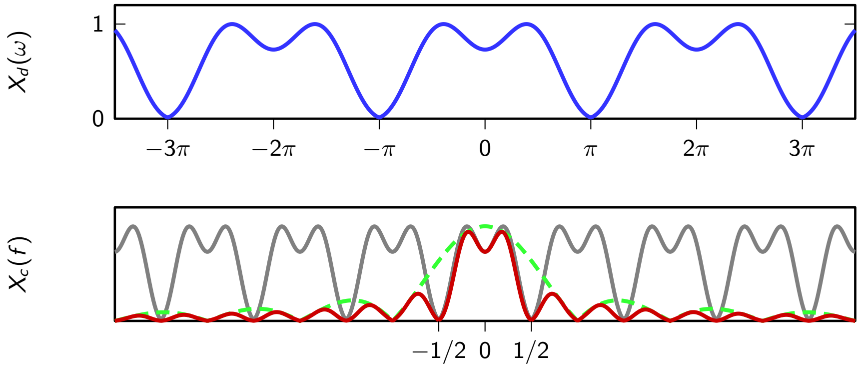
Sinc interpolation, $K = \infty$

$$X_c(f) = X_d(2\pi f)\, I_\infty(f) \qquad (F_s = 1)$$



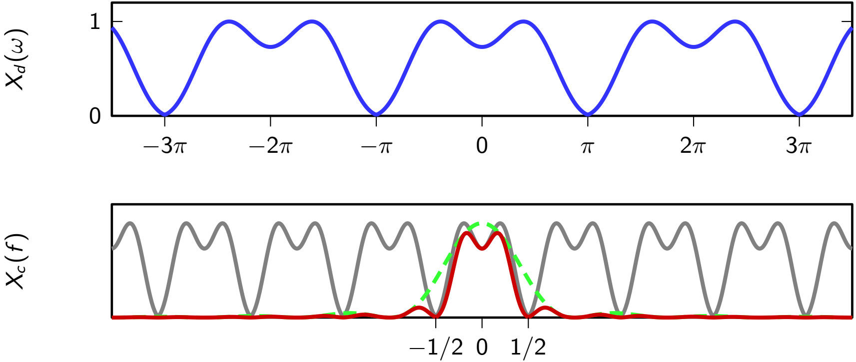
Zero-order hold interpolation, $K = 0$

$$X_c(f) = X_d(2\pi f)\, I_0(f) \qquad (F_s = 1)$$



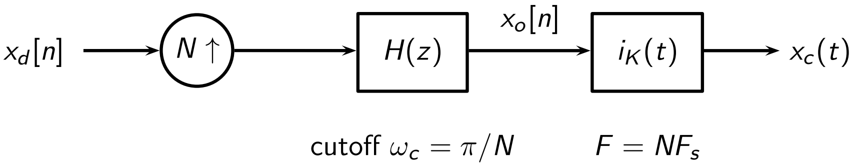
First-order interpolation, $K = 1$

$$X_c(f) = X_d(2\pi f)\, I_1(f) \qquad (F_s = 1)$$

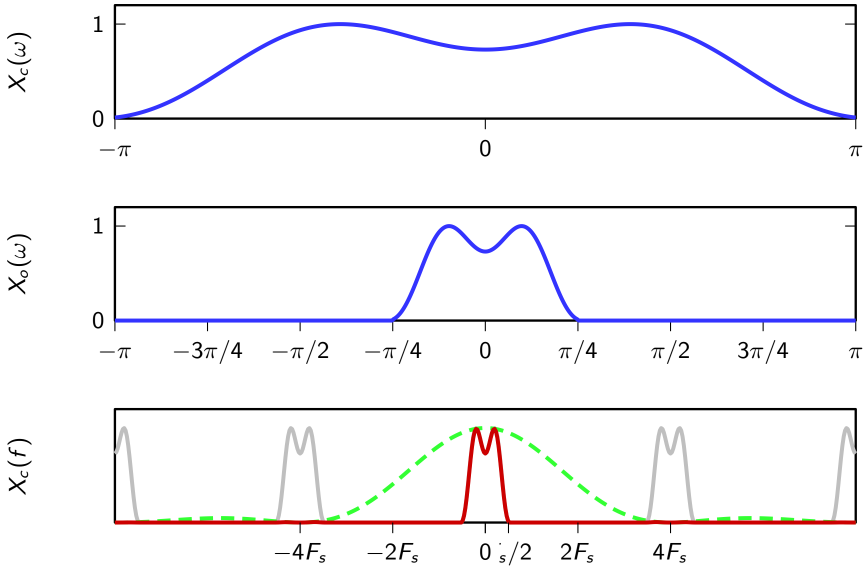


Problems with low-order kernels

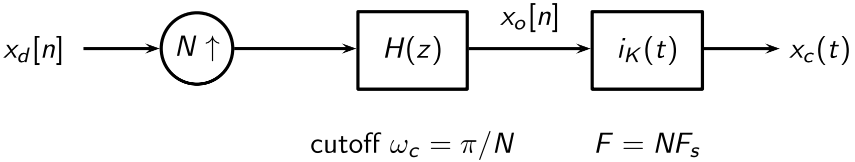
- low-order kernels decay slowly in frequency and cannot filter out the replicas
- idea: space out the replicas in the digital spectrum to make room
- with digital multirate techniques we can do this very well
- we use lots of cheap digital processing instead of expensive analog filters



Oversampled continuous-time interpolation ($N = 4$)

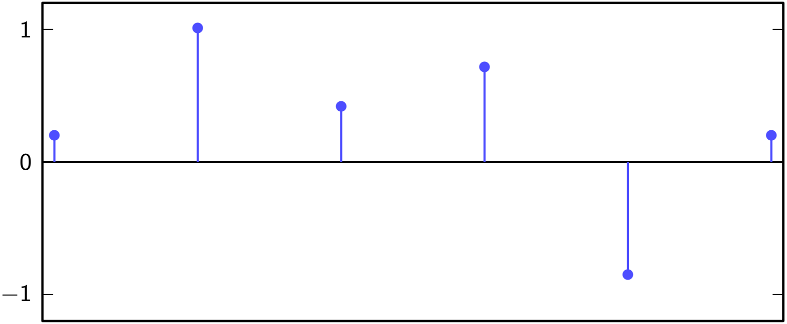


Oversampled interpolation: time-domain analysis

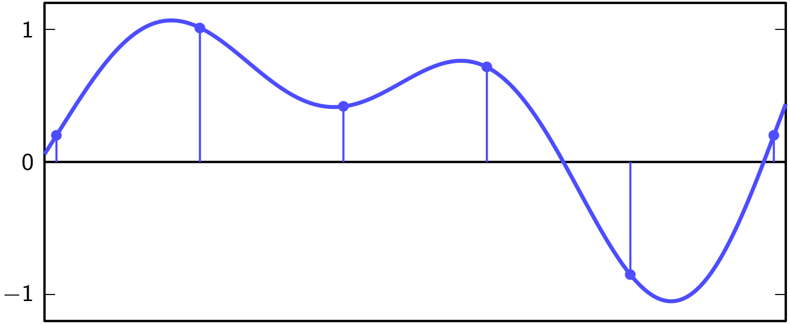


- digitally oversampling $\mathbf{x}_d$ is equivalent to densely sampling a continuous-time interpolation of $\mathbf{x}_d$
- by using a very good digital lowpass after the upsampler, we can approximate a sinc interpolation in discrete time
- as we increase the oversampling factor, the samples become sufficiently closer that a ZOH interpolator is enough

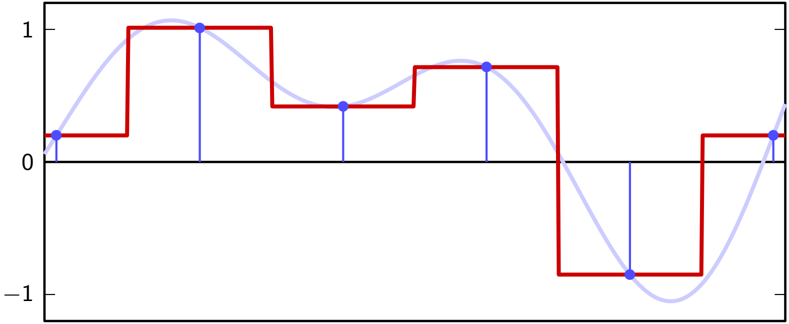
Increasing the oversampling factor before ZOH



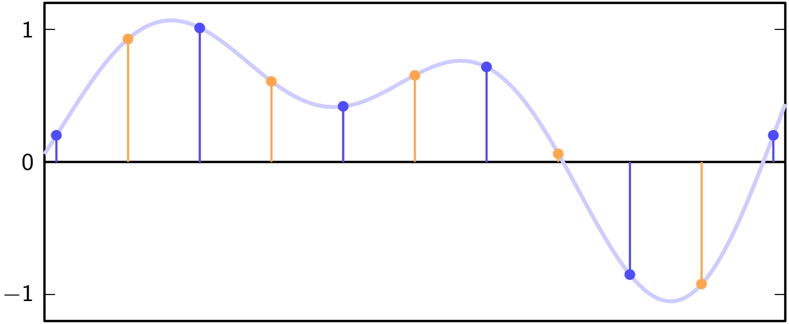
Increasing the oversampling factor before ZOH



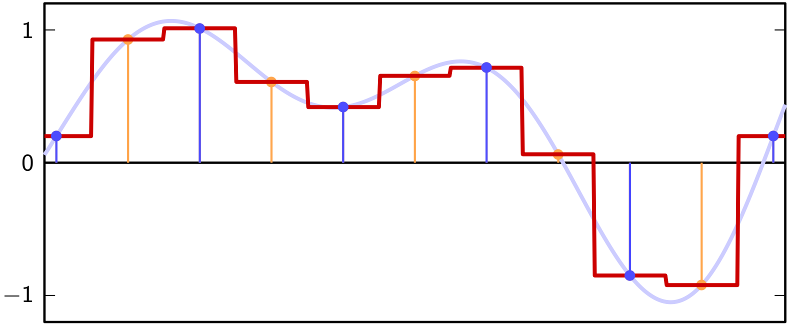
Increasing the oversampling factor before ZOH



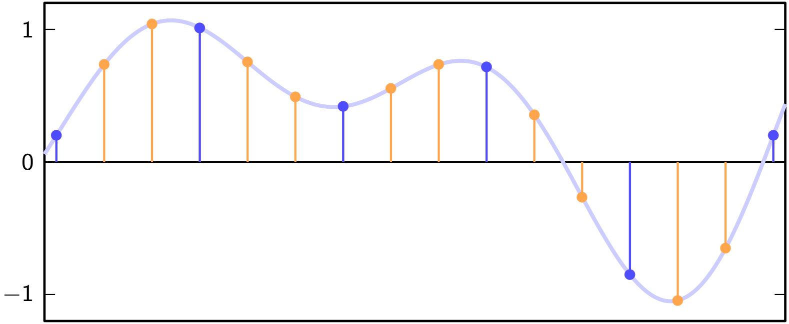
Increasing the oversampling factor before ZOH



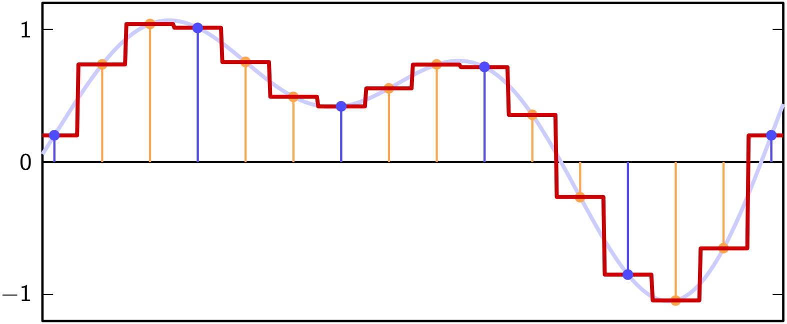
Increasing the oversampling factor before ZOH



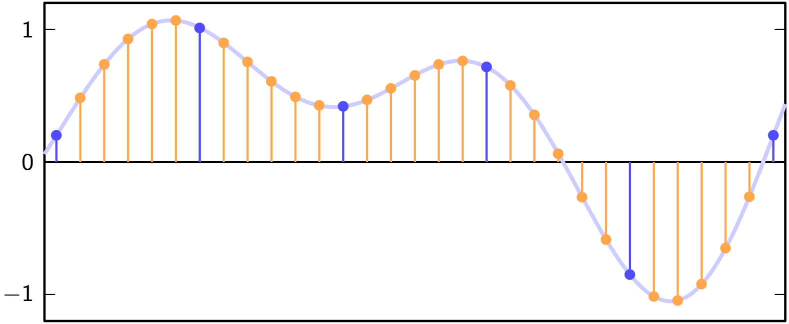
Increasing the oversampling factor before ZOH



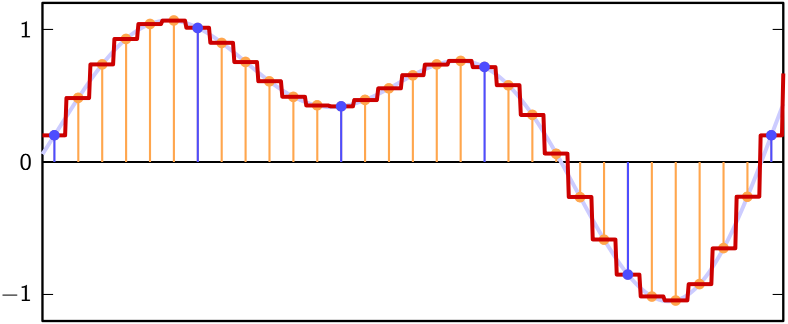
Increasing the oversampling factor before ZOH



Increasing the oversampling factor before ZOH



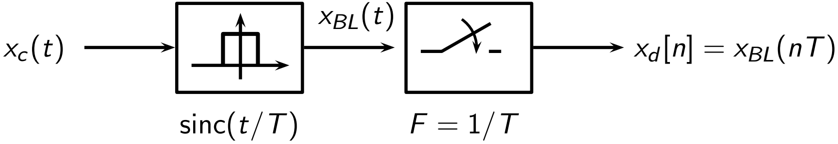
Increasing the oversampling factor before ZOH



sampling in practice

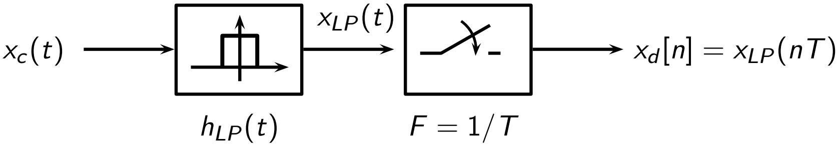
Sinc sampling

$$x_d[n] = \left\langle \text{sinc}\left(\frac{t - nT}{T}\right), x_c(t) \right\rangle = (x_c * \text{sinc}_T)(nT)$$



- sinc sampling is equivalent to an analog anti-alias filter followed by a raw sampler
- in theory the filter should be an ideal lowpass with cutoff $F/2$
- but we know we can't implement ideal filters

Realisting sampling

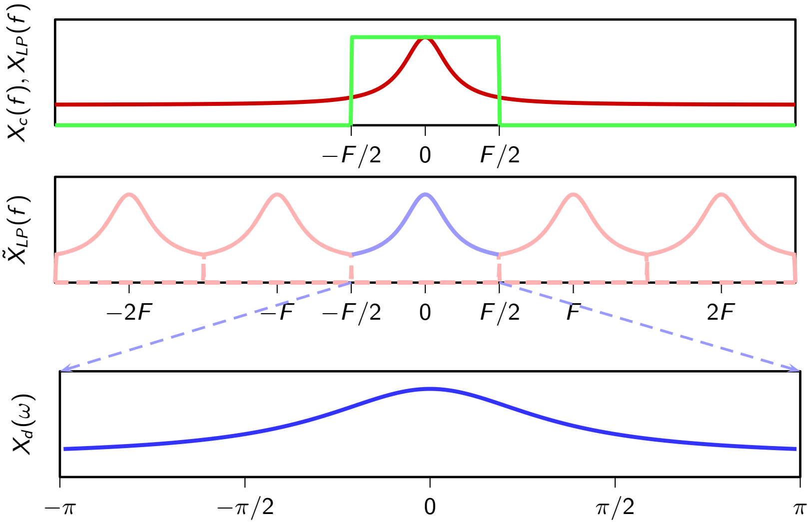


- in a practical sampler we use a realizable analog lowpass filter
- the filter won't be perfect so some aliasing will occur

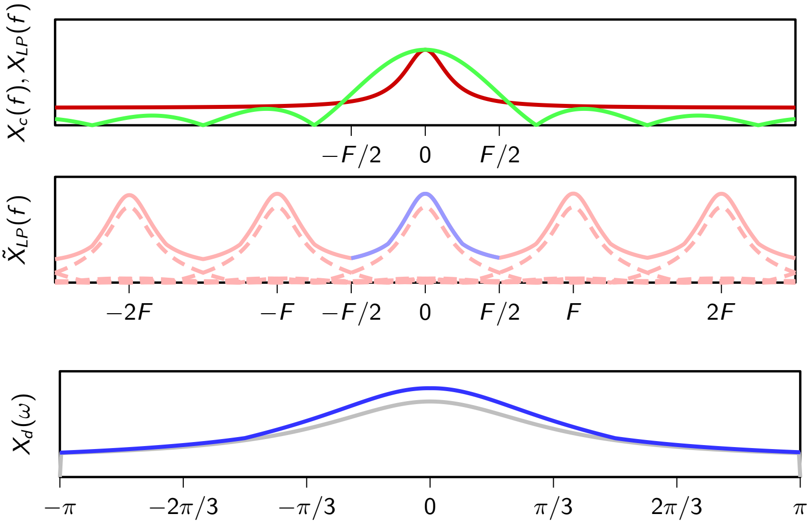
$$X_{LP}(f) = H_{LP}(f)X_c(f)$$

$$X_d(\omega) = F \sum_{k=-\infty}^{\infty} X_{LP}\left(\frac{\omega}{2\pi}F - kF\right)$$

Ideal case, $h_{LP}(t) = \mathrm{sinc}(t/T)$

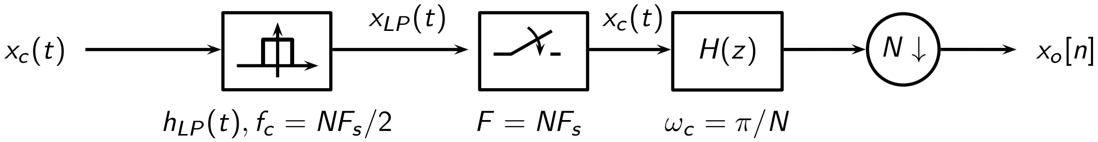


Realistic case, using a “cheap” lowpass

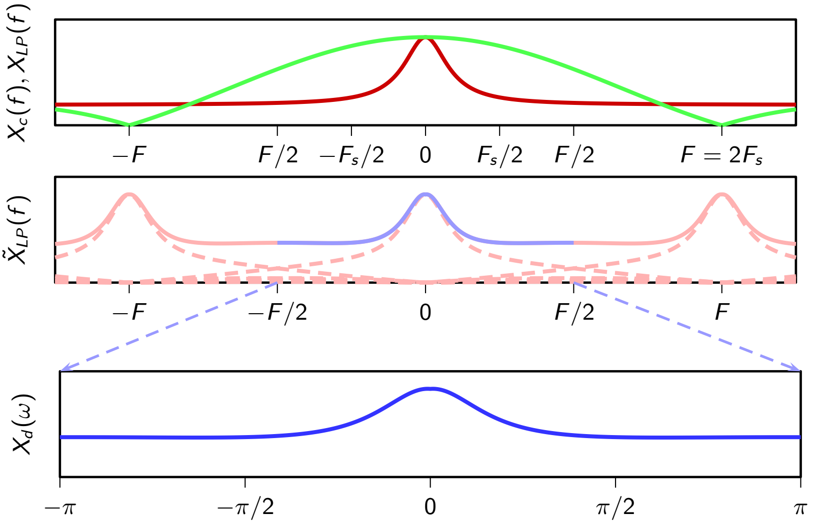


Problems with antialiasing filters

- sharp analog lowpass filters are complicated and expensive
- we would like to do the difficult processing in discrete time
- idea: sample at N times the nominal rate using a cheap antialias
- the wider spacing will reduce overlap since most signals are lowpass
- use decimation by N with a good digital lopass to go back to the intended sampling rate



Example: two-times oversampling $(F = 2F_s)$



Example: decimation of two-times oversampled signal