Sampling:
DAC and ADC conversion

Course Objectives

Specific Course Topics:

-Basic test signals and their properties

-Basic system examples and their properties

-Signals and systems interaction (Time Domain: Impulse Response and convolution, Frequency Domain: Frequency Response)

-Applications that exploit signal & systems interaction: system id, audio effects, noise filtering, AM / FM radio

-Signal sampling and reconstruction

ADC and DAC

Goals

I. From analog signals to digital signals (ADC)

- The sampling theorem
- Signal quantization and dithering

II. From digital signals to analog signals (DAC)

- Signal reconstruction from sampled data

III. DFT and FFT

ADC and DAC

Analog-to-digital conversion (ADC) and **digital-to-analog conversion** (DAC) are processes that allow computers to interact with analog signals. These conversions take place in e.g., CD/DVD players.

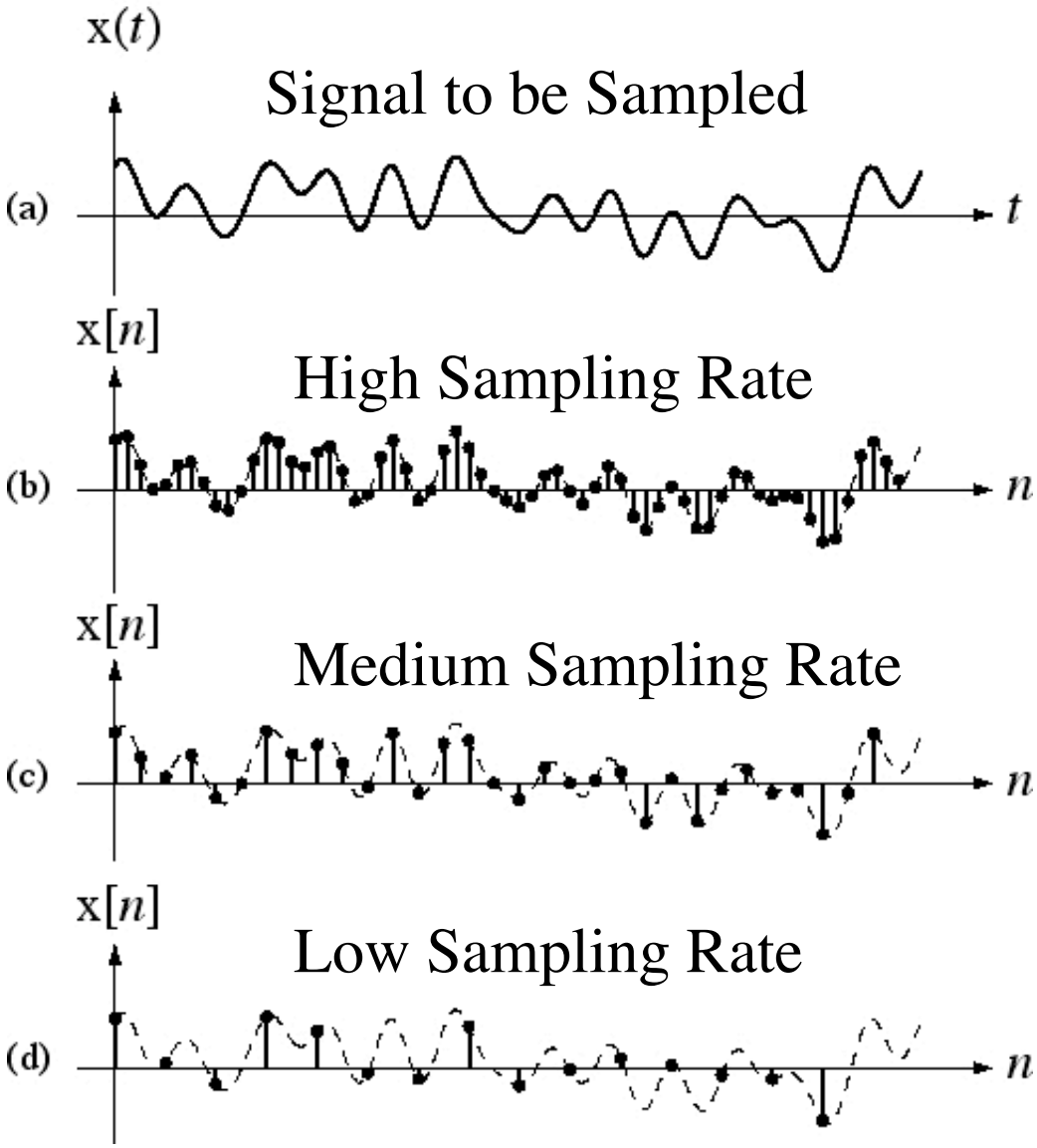
Digital information is different from the analog counterpart in two important respects: it is *sampled* and it is *quantized.*

These operations restrict the information a digital signal can contain. Thus, it is important to understand what information you need to retain and what information you can afford to lose. These are *information management* questions.

The understanding of these restrictions will allow us decide how to select the *sampling frequency, number of bits, and the type of filter* for converting between the analog and digital realms.

Sampling

The fundamental consideration in sampling is how fast to sample a signal to be able to reconstruct it.



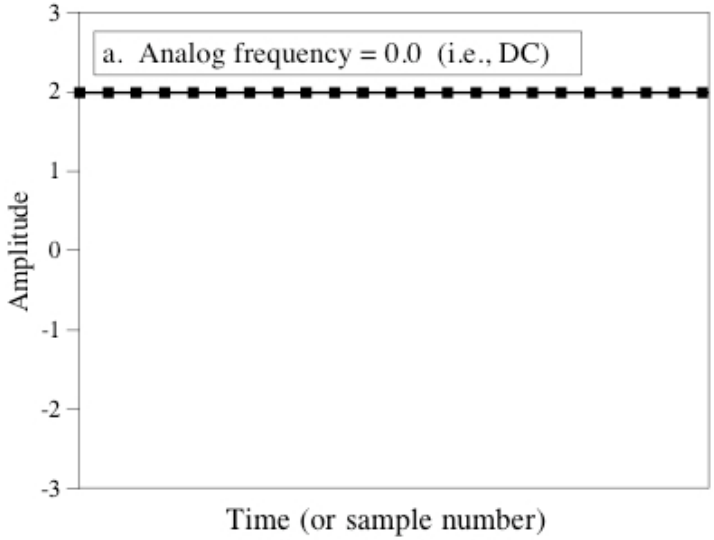
The sampling theorem

Suppose you sample a signal in some way. If you can **exactly reconstruct** the signal from the samples, then you have done a **proper sampling** and **captured the key** signal **information**

Definition: The **sampling frequency** f_s , is the number of samples per second. This is to be compared with the signal cyclic frequencies

Examples of proper sampling:

The continuous analog signal (constant DC value or cosine with zero frequency) can be reconstructed from the samples

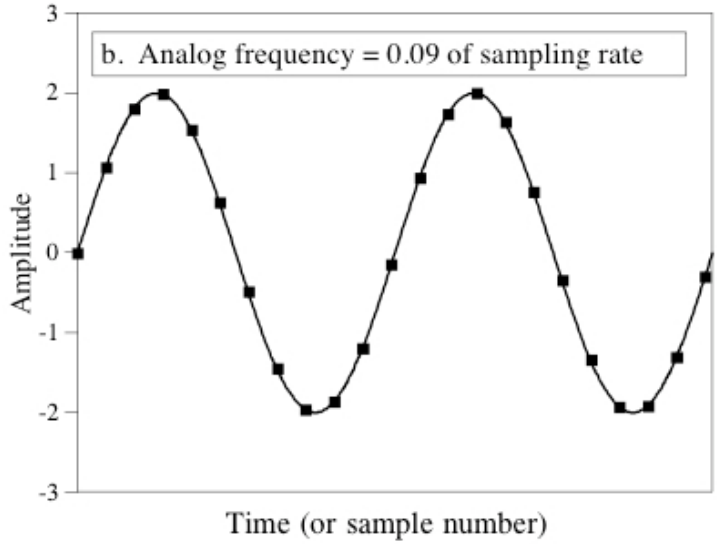


The sampling theorem

Example of proper sampling (II):

This is a $f = 90$ cycle/second sinusoid sampled at $f_s = 1000$ samples/second.
In other words, the wave has a frequency of 0.09 of the sampling rate:

$$f = 0.09 \times f_s = 0.09 \times 1000$$



Equivalently, there are $1000/90 = 11.1$ samples taken over a complete cycle of the sinusoid

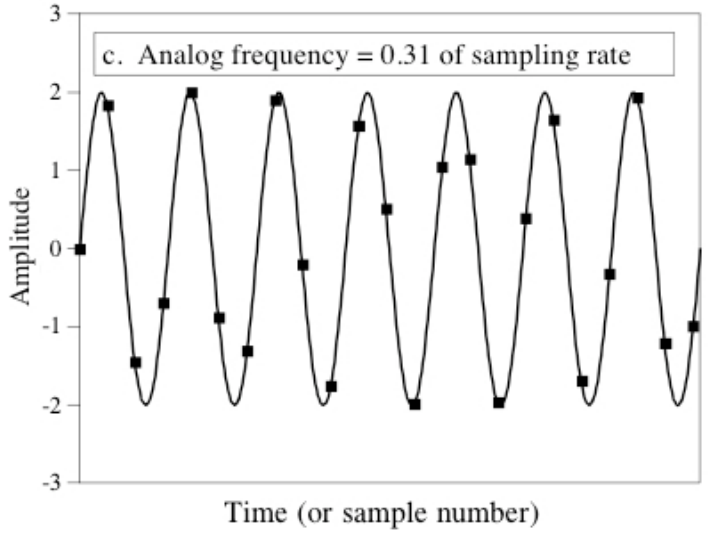
These samples represent accurately the sinusoid because there is no other sinusoid that can produce the same samples

The sampling theorem

Example of proper sampling (III):

Here, $f = 0.31 \times f_s$

This results into 3.2 samples per sine wave cycle. The samples are so sparse they don't appear to follow the analog wave



Strange as it seems**, it can be proven that no other sine wave can produce the same type of samples**

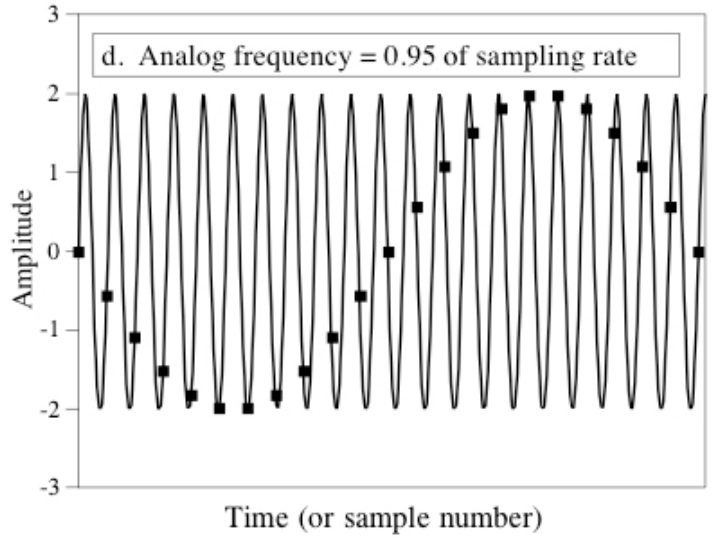
Since the samples represent accurately the sinusoid they constitute a proper sampling

The sampling theorem

Example of improper sampling:

Here, $f = 0.95 \times f_s$

This results into 1.05 samples per sine wave cycle



Clearly, this is an **improper sampling** of the signal because **another sine wave can produce the same samples**

The original sine misrepresents itself as another sine. This phenomenon is called **aliasing**. (The original sine has hidden its true identity!)

The sampling theorem

Suppose a signal's highest frequency is f_m (a low-pass or a band-pass signal). Then a proper sampling requires a sampling frequency f_s at least satisfying

$$f_m < \frac{f_s}{2} = 0.5 \times f_s$$

The number $\frac{f_s}{2}$ is called the **Nyquist frequency**

The number $2f_m$ is called the **Nyquist rate**

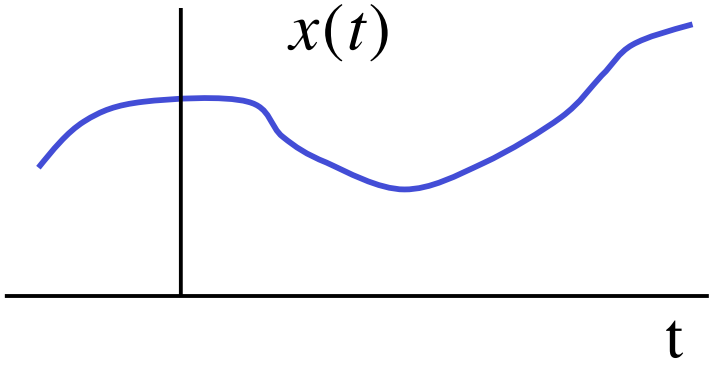
Example: Consider an analog signal with frequencies between 0 and 3kHz. A proper sampling requires a 6kHZ sampling frequency or higher

Effects of aliasing: It can change the signal real frequency and the signal real phase (as we saw in previous slide)

Discrete-time Signals

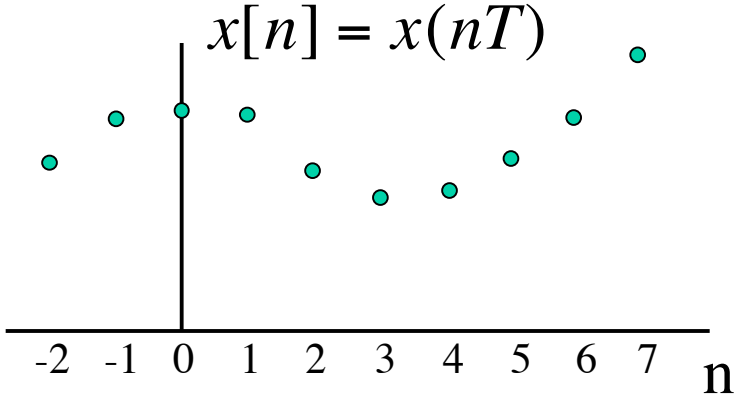
Continuous-time signal $x(t)$

math: real function of t



Discrete-time signal x_n

math: sequence

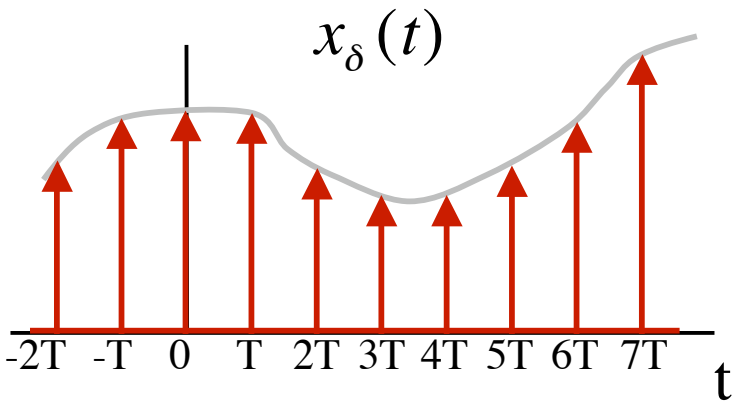


Impulse-sampled signal $x_\delta(t)$

math: train of impulses

$$x_\delta(t) = x(t)\sum_{n=-\infty}^{\infty}\delta(t-nT)$$

$$= \sum_{n=-\infty}^{\infty} x[n]\delta(t-nT)$$



Sampling in the Frequency Domain

A **train of impulses** has Fourier Transform

$$\delta_{T_s}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \quad \longleftrightarrow \quad \delta_{\omega_s}(j\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta\Big(j\big(\omega - k\omega_s\big)\Big), \quad \omega_s = \frac{2\pi}{T_s}$$

So **sampling** a continuous-time signal **in the time domain**

$$x_\delta(t) = x(t)\delta_{T_s}(t) = \sum_{n=-\infty}^{n=\infty} x(nT_s)\delta(t - nT_s)$$

is a **convolution in the frequency domain**

$$X_\delta(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X\Big(j\big(\omega - \sigma\big)\Big)\, \delta_{\omega_s}\big(j\sigma\big)\, d\sigma$$

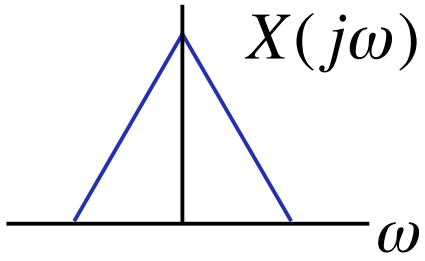
$$= \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X\Big(j\big(\omega - k\omega_s\big)\Big)$$

Sampling in the Frequency Domain

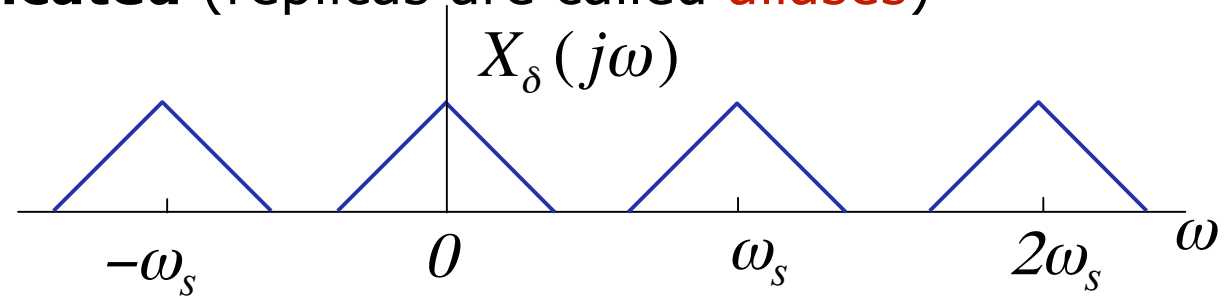
Graphical interpretation of the formula

$$X_\delta(j\omega) = \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X\Big(j\big(\omega - k\omega_s\big)\Big), \quad \omega_s = 2\pi f_s, \quad f_s = \frac{1}{T_s}$$

Continuous-time spectrum



is **scaled** and **replicated** (replicas are called aliases)

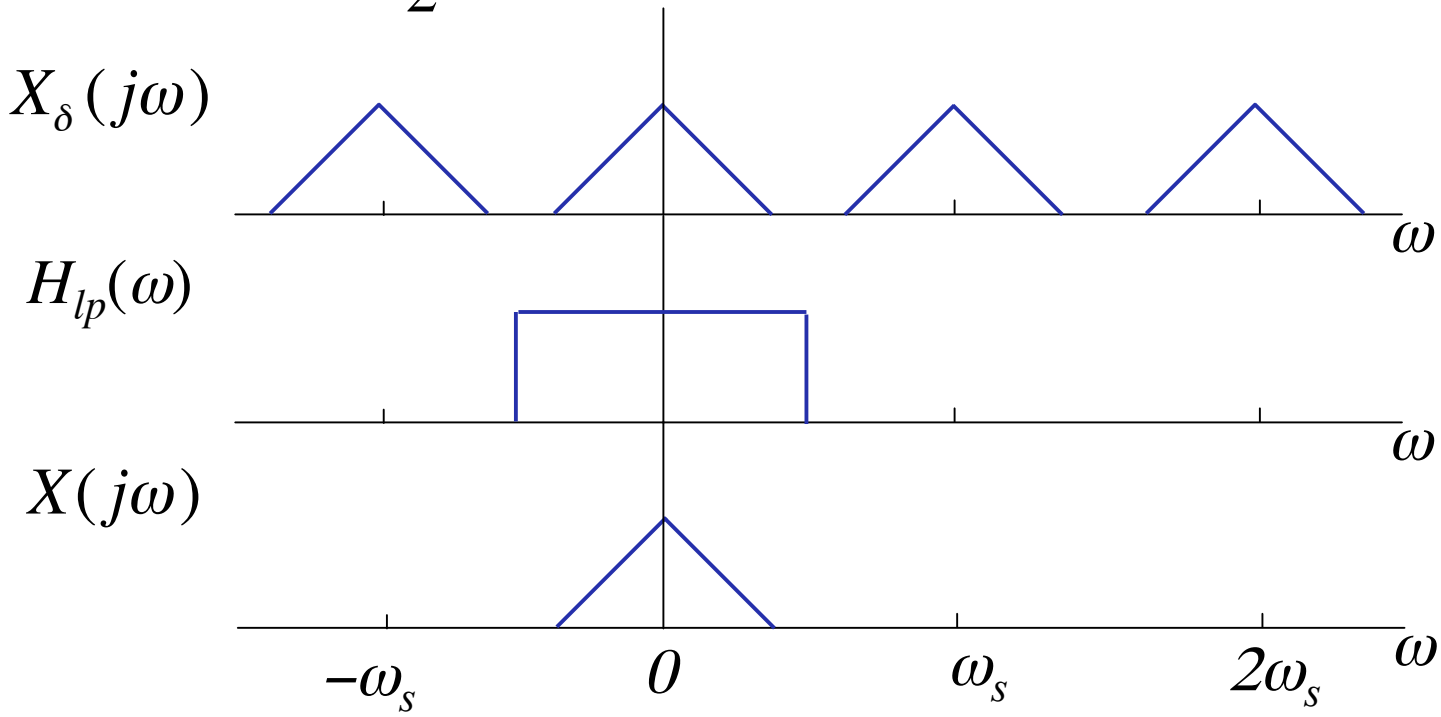


Aliasing: If signal bandwith $f_B \geq \frac{f_s}{2}$ then spectrum overlaps!

Reconstruction from sampled signal

Question: When can we reconstruct $x(t)$ from $x_\delta(t)$?

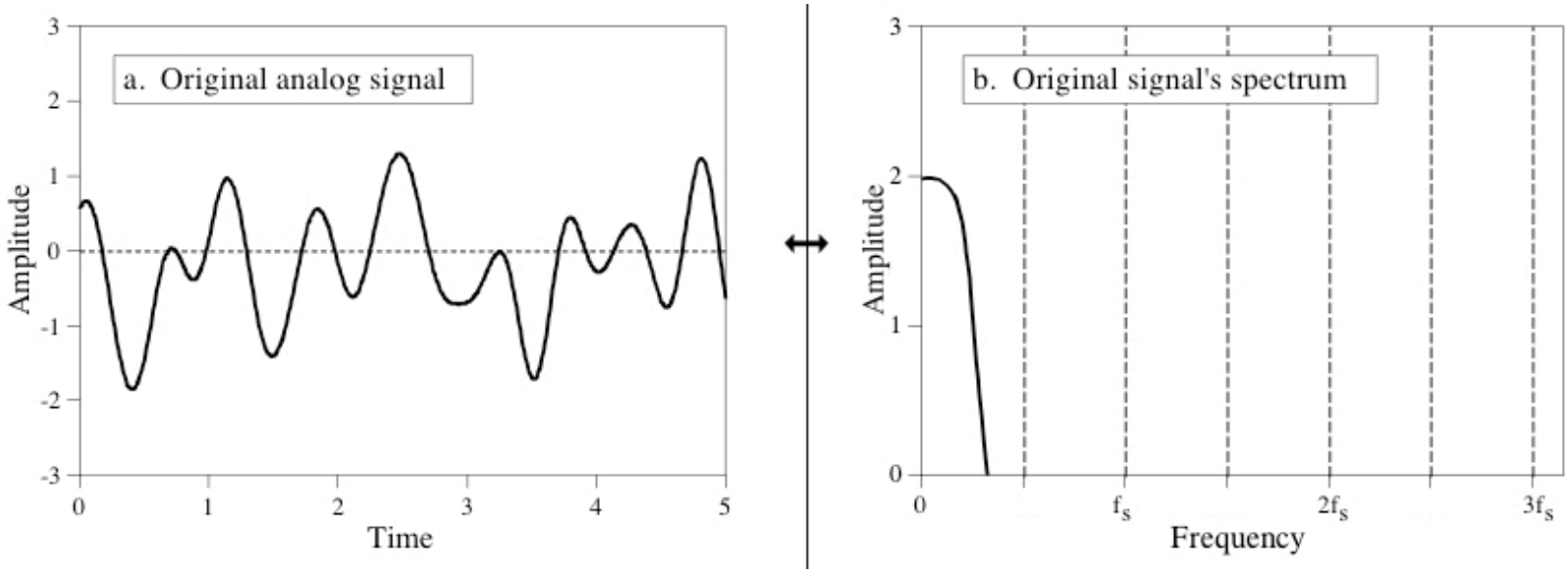
Answer: If $f_B < \frac{f_s}{2}$ (no aliasing) use a **low-pass filter**!



If $f_B \geq \frac{f_s}{2}$ (with aliasing) original signal has been lost!

Aliasing in the frequency domain

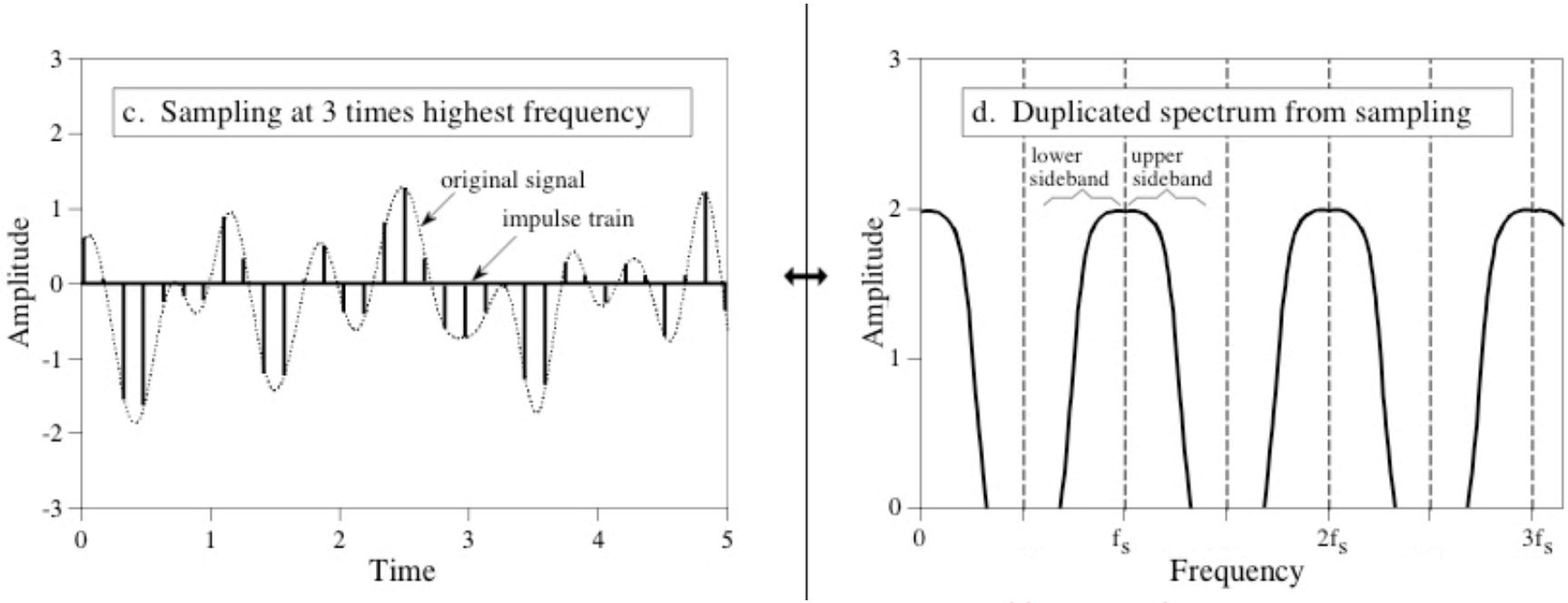
In order to understand better why aliasing is produced, and to demonstrate the sampling theorem, let us look at the signals in the frequency domain



Here the analog signal has a frequency of $0.33 \times f_s < 0.5 \times f_s$

Aliasing in the frequency domain

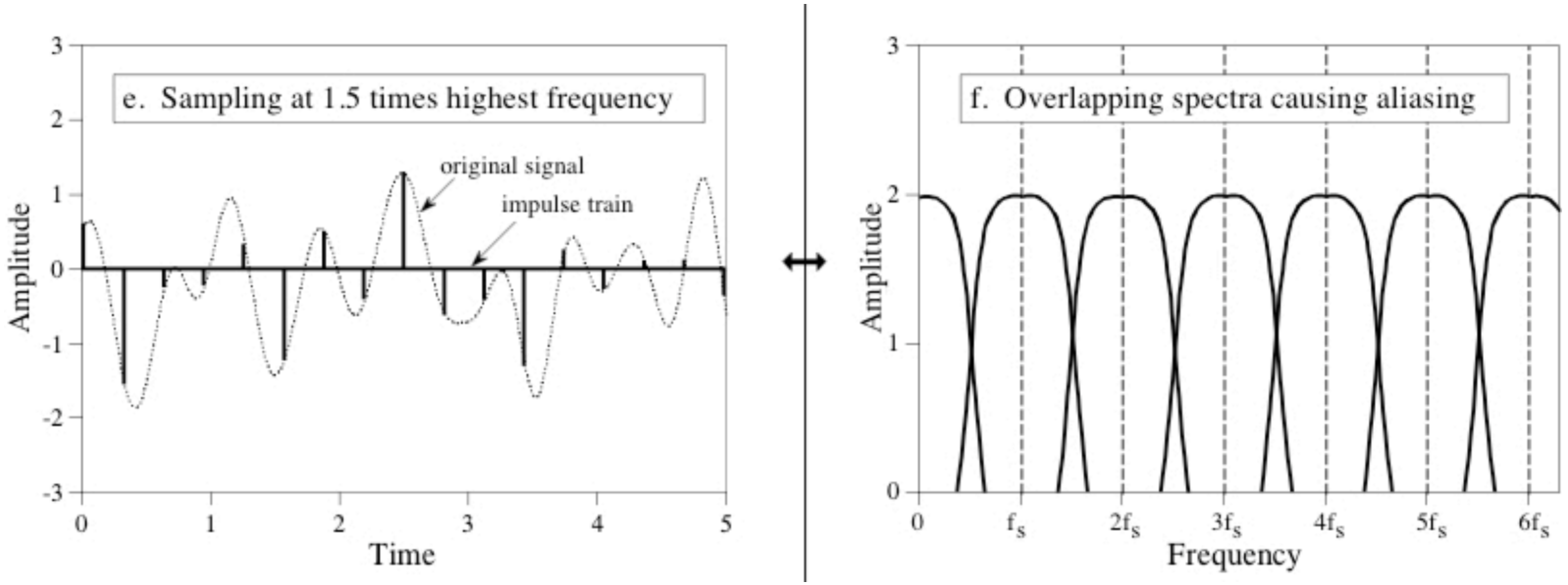
The sampling of the signal corresponds to doing a **convolution of the signal with a delta impulse train**. We call the resulting signal **impulse train**. The magnitude of this impulse train is shown on the right



As you can see, the sampling has the effect of **duplicating the spectrum** of the original signal an infinite number of times. In other words, **sampling introduces new frequencies**

Aliasing in the frequency domain

Compare the previous plot with the following one for a different sampling:



Here, $f = 0.66 \times f_s > 0.5 \times f_s$, so this is **an improper sampling**. In the FD this means that **the repeated spectra overlap**. Since there is no way to separate the overlap, the **signal information is lost**

Example

At what rate should we sample the following signal to be able to perfectly reconstruct it?

$$x(t) = 32\text{sinc}(101t)\cos(200\pi t)$$

Go to the frequency domain

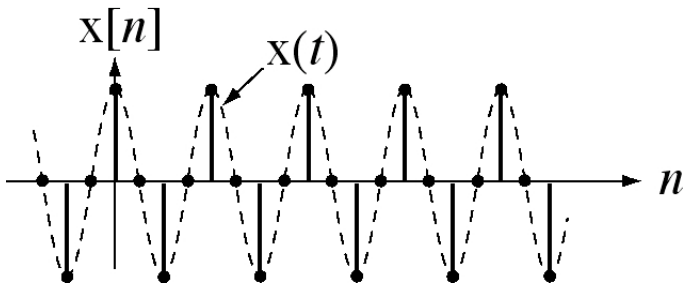
$$X(f) = 32 \times 101 \times \text{rect}(\frac{f}{101}) * \frac{1}{2}\left(\delta(f-100) + \delta(f+100)\right)$$

$$= 16 \times 101 \times \left(\text{rect}(\frac{f-100}{101}) + \text{rect}(\frac{f+100}{101})\right)$$

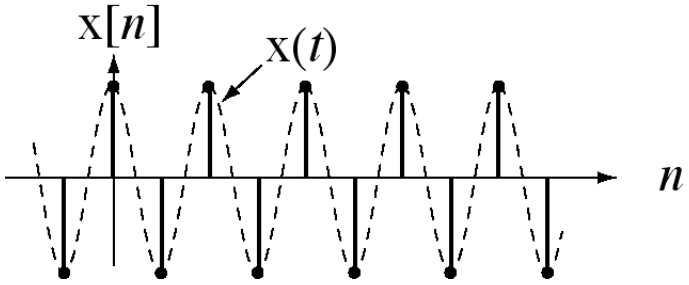
Hence $f_m = 301/2 Hz$

Nyquist rate is $2f_m = 301 Hz$

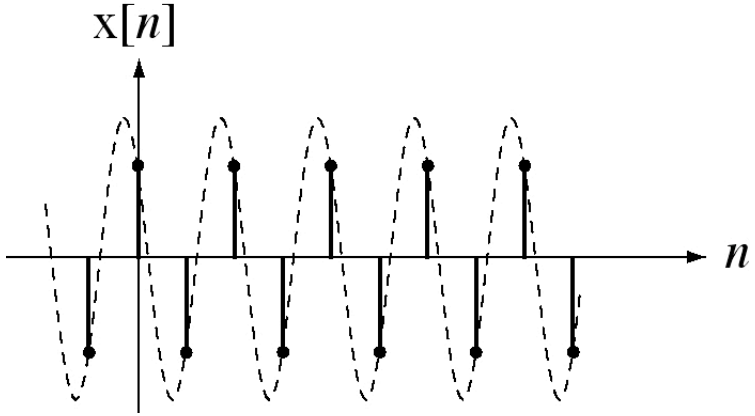
Sampling must be done above the Nyquist rate



Cosine sampled at twice its Nyquist rate. Samples uniquely determine the signal.

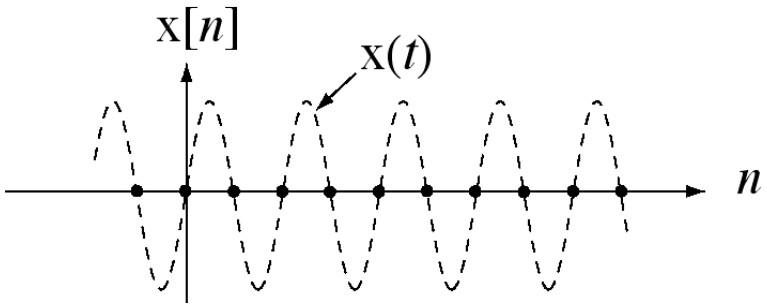


Cosine sampled at exactly its Nyquist rate. Samples *do not* uniquely determine the signal.



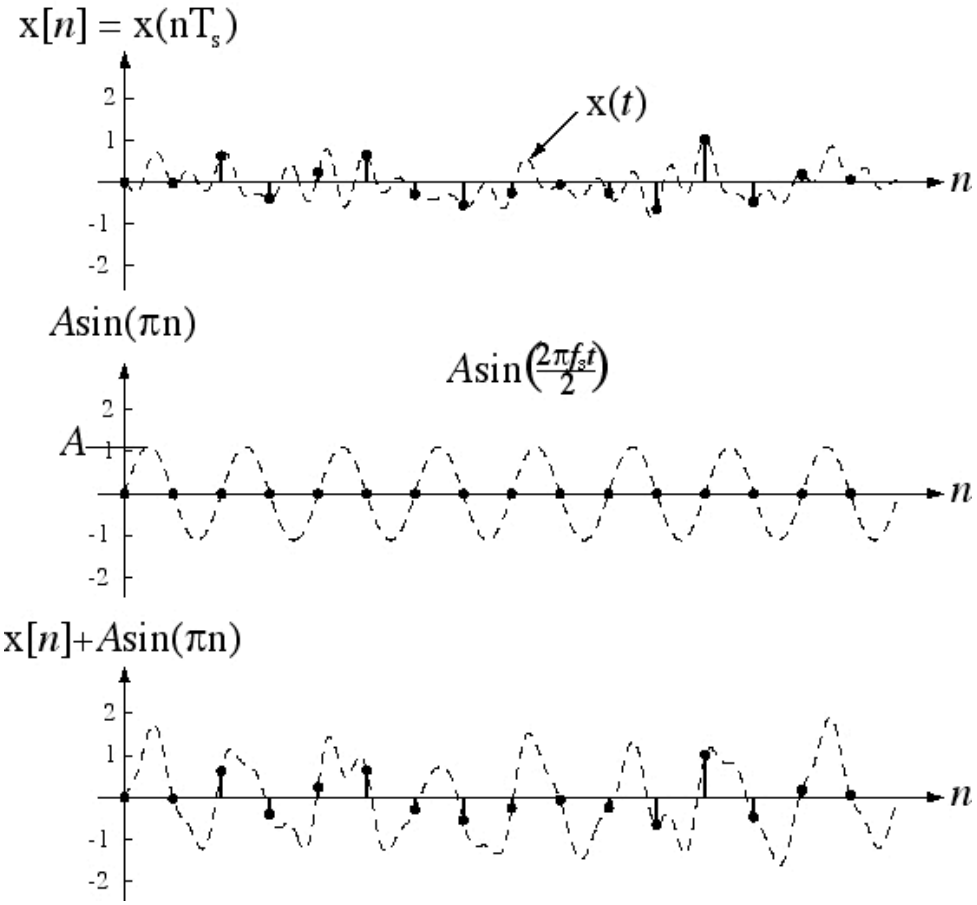
A different sinusoid of the same frequency with exactly the same samples as above.

Sampling a Sinusoid



Sine sampled at its Nyquist rate. All the samples are zero.

Adding a sine at the Nyquist frequency (half the Nyquist rate) to any signal does not change the samples.



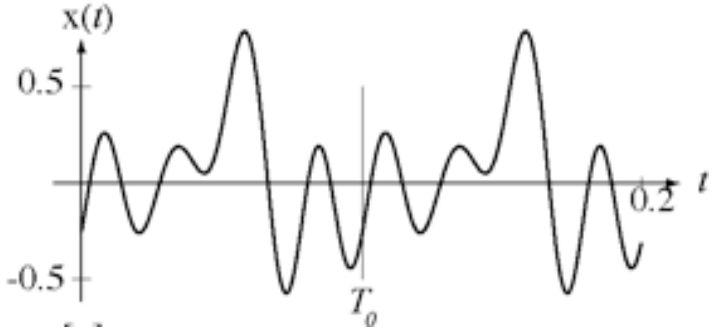
Bandlimited Periodic Signals

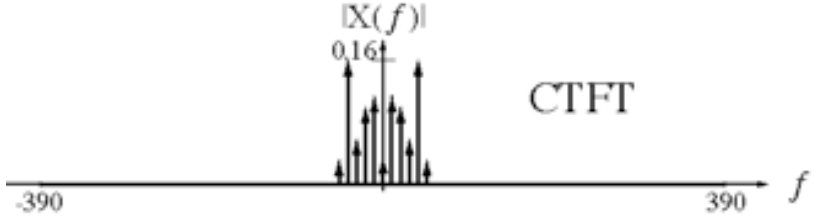
If a signal is bandlimited it can be properly sampled according to the sampling theorem.

If that signal is also periodic its CTFT consists only of impulses.

Since it is bandlimited, there is a finite number of (non-zero) impulses.

Therefore the signal can be exactly represented by a **finite set of numbers**, the impulse strengths.



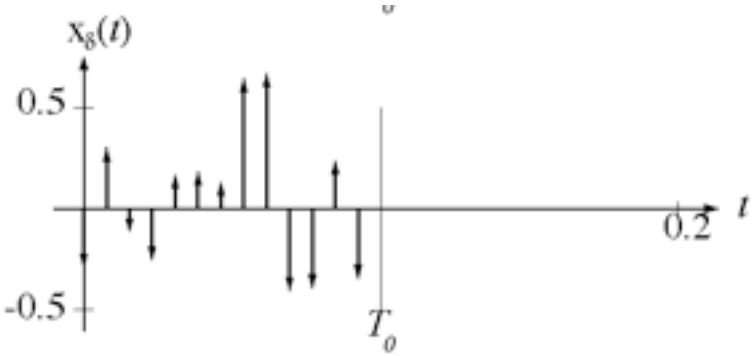


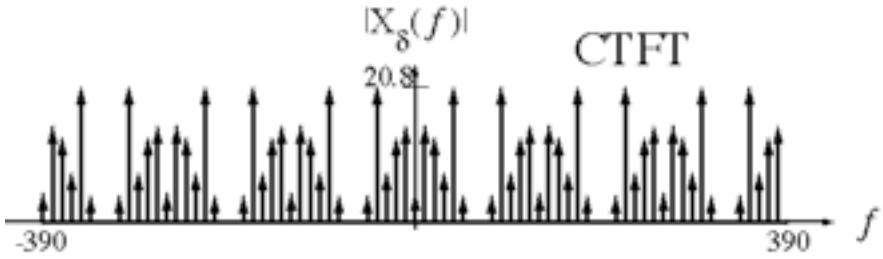
Bandlimited Periodic Signals

If a bandlimited periodic signal is sampled above the Nyquist rate over exactly one fundamental period, that set of numbers is sufficient to completely describe it

If the sampling continued, these same samples would be repeated in every fundamental period

So a finite sequence of numbers is needed to completely describe the signal in both time and frequency domains





ADC and DAC

Goals

I. From analog signals to digital signals (ADC)

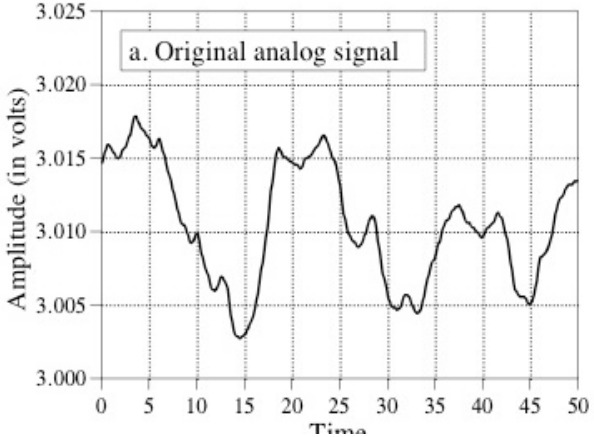
- The sampling theorem
- Signal quantization and dithering

II. From digital signals to analog signals (DAC)

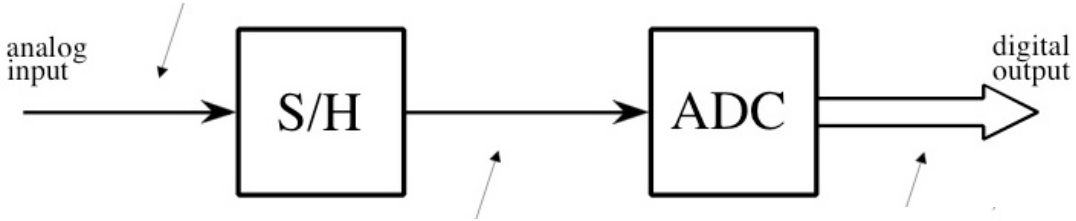
- Signal reconstruction from sampled data

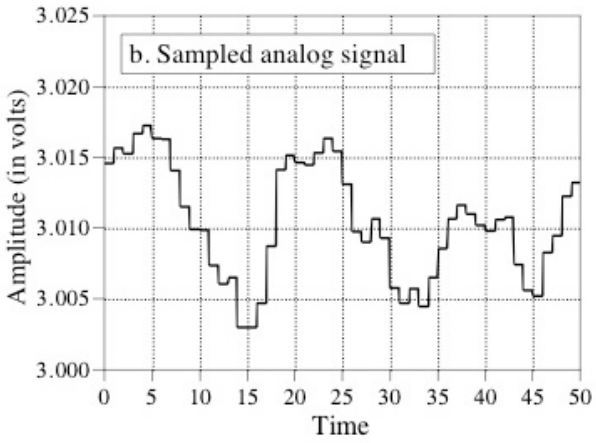
III. DFT and FFT

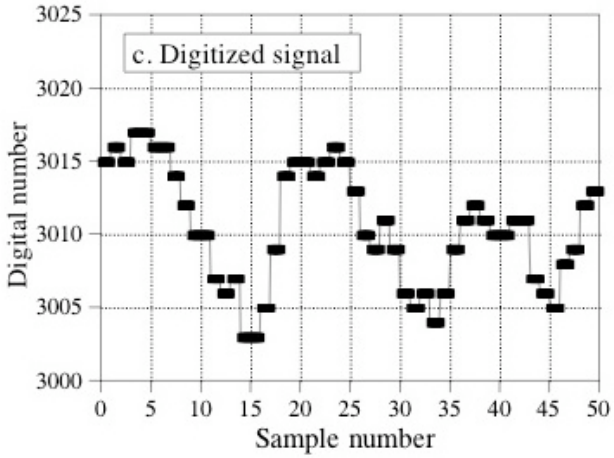
Illustration of digitization process



We will look into two stages of the ADC process: *sample and hold* and *quantization. After* that, the signal is *encoded* into bits.



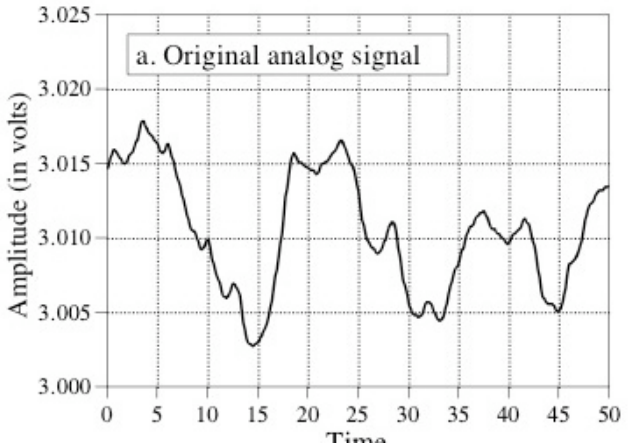


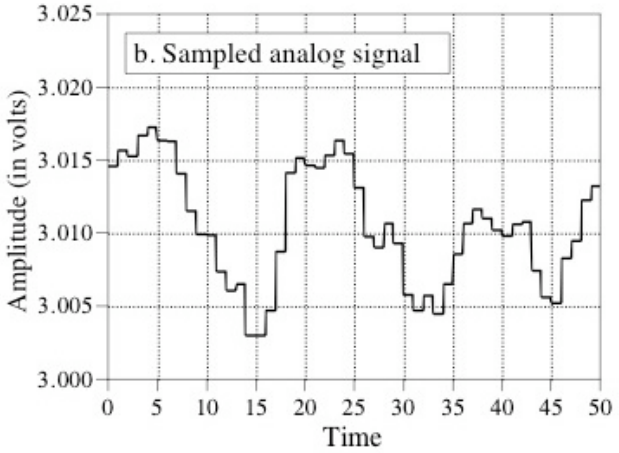


Digitization process

Sample and hold:

The output only changes at periodic instants of time. The independent variable now takes values in a discrete set





Before:

$$t \in (0,50),$$

$$y_A(t) \in (3000,3025),$$

After sampling:

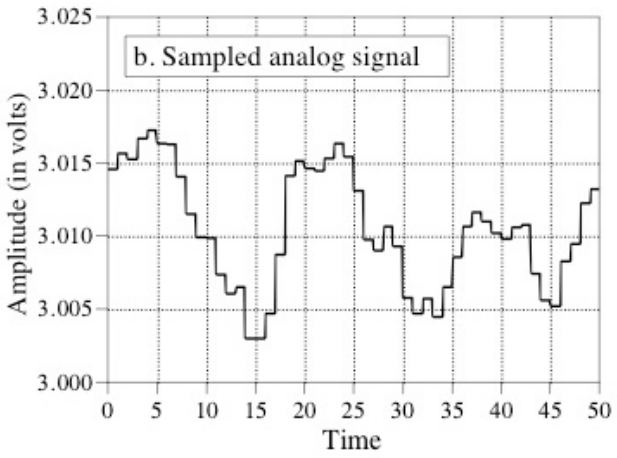
$$t \in \{0,0.1,...50\},$$

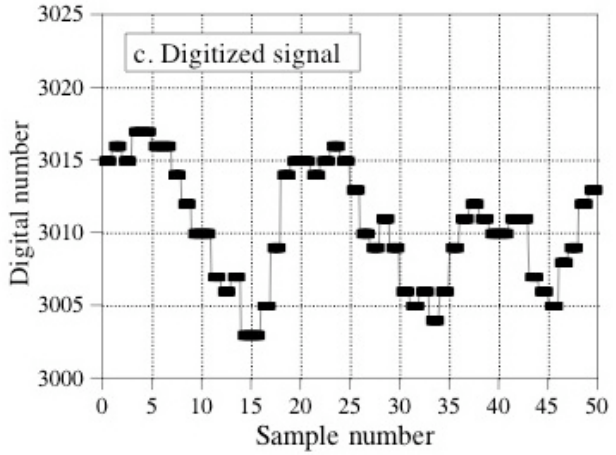
$$y_S(t) \in (3000,3025),$$

Digitization process

Quantization:

Each flat region in the sampled signal is "rounded-off" to the nearest member of a set of discrete values (e.g., nearest integer)





Before: $t \in \{0, 0.1, \ldots 50\}$,

$y_A(t) \in (3000, 3025)$,

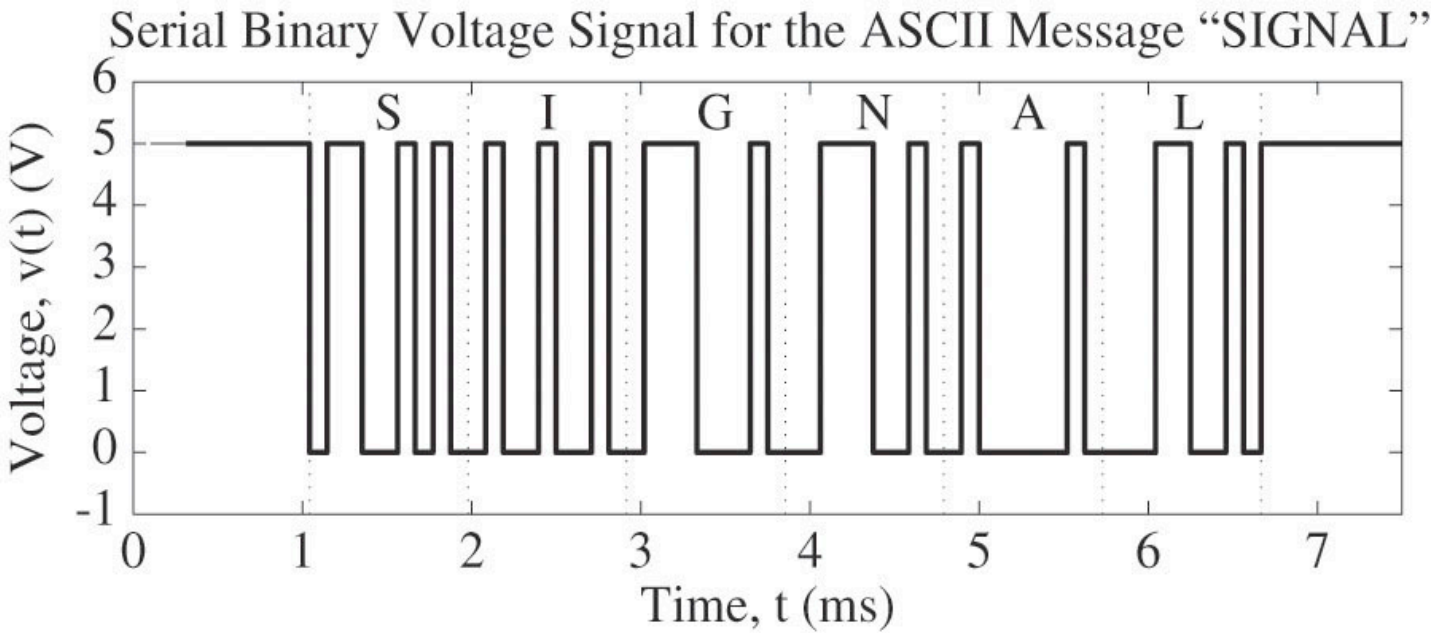
After quantization: $t \in \{0, 0.1, \ldots 50\}$,

$y_Q(t) \in \{3000, 3001, \ldots, 3025\}$

The quantized signal can be then encoded into bits. A bit is a binary digit (0 or 1). A total of $2^7 = 128$ characters (letters of the alphabet, numbers 0 to 9, and some punctuation characters) can be encoded into a sequence of 7 binary bits

Digitization process

Illustration of the code for the word “signal”:



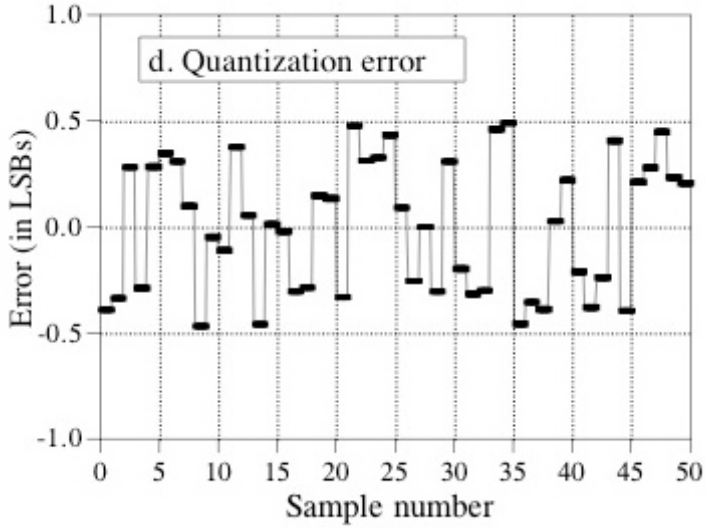
Bits are sent sequentially, they are preceded by a *start* bit followed by one or two *stop* bits

In direct-wired connections between digital equipment, bit can be represented by a higher voltage (2 to 5V) for a 1 and a lower voltage (around 0V) for a 0

Signal quantization

Main effect of quantization: introduces error in the signal

This graph measures the difference between the sampled and quantized signals. The error is measured in LSBs (least significant bit, a DSP jargon) and is between the - ½ and ½ values.



The quantization error looks very much like random noise. Because of this, it is usually modeled as a random number between – ½ and ½, with zero mean and standard deviation of 1/sqrt(12) LSB (uniform probability distribution)

Signal quantization

Facts:

(1) The random noise or *quantization error will add to whatever noise is present* in the analog signal

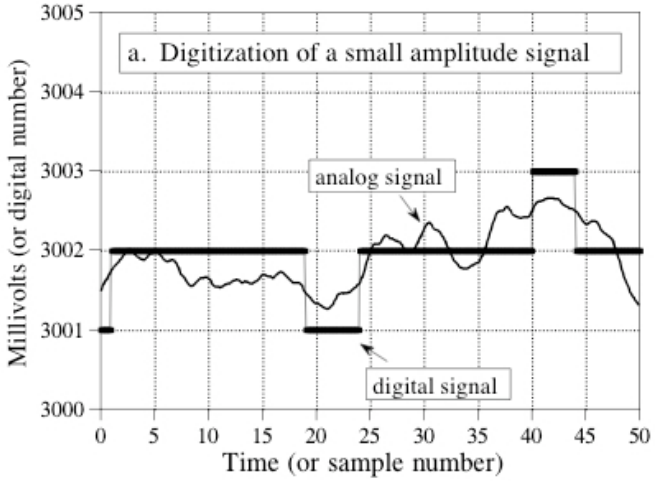
(2) The quantization error is determined by the number of bits that are later used to encode the signal. *If we increase the number of bits, the error will decrease*

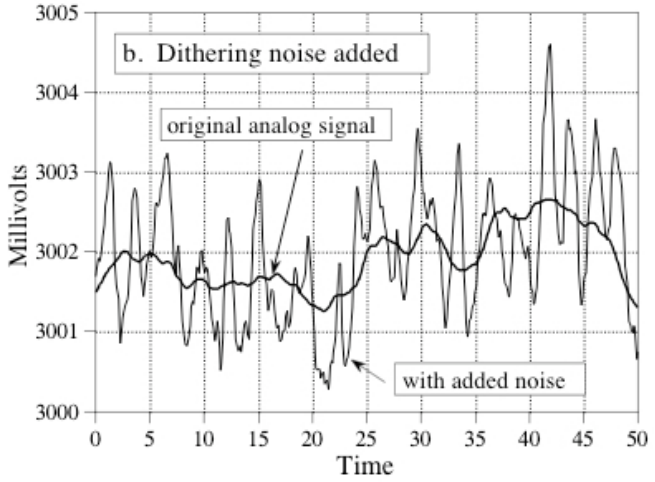
Question: How many bits do we need in the ADC? How fine should the quantization be? (equivalent questions)

Answer: the number of bits chosen should be enough so that the quantization error is small in comparison with the noise present in the signal

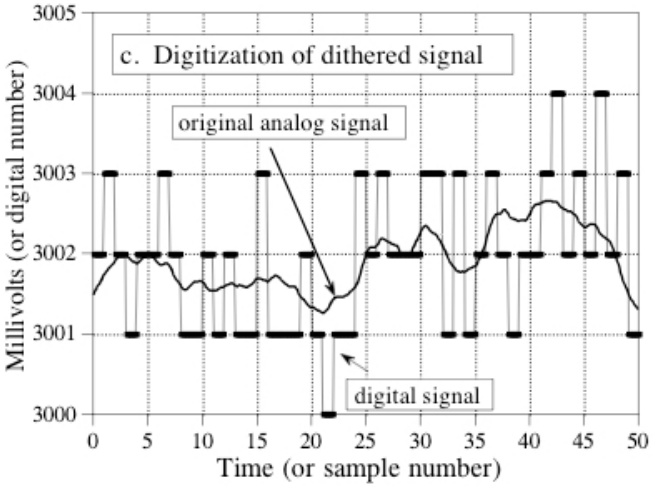
Dithering

When isn't this model of quantization as noise valid? When the analog signal remains about the same value for many consecutive samples





In the situation above, **dithering**; i.e., adding small noise to the signal before quantizing improves things!



ADC and DAC

Goals

I. From analog signals to digital signals (ADC)

- The sampling theorem
- Signal quantization and dithering

II. From digital signals to analog signals (DAC)

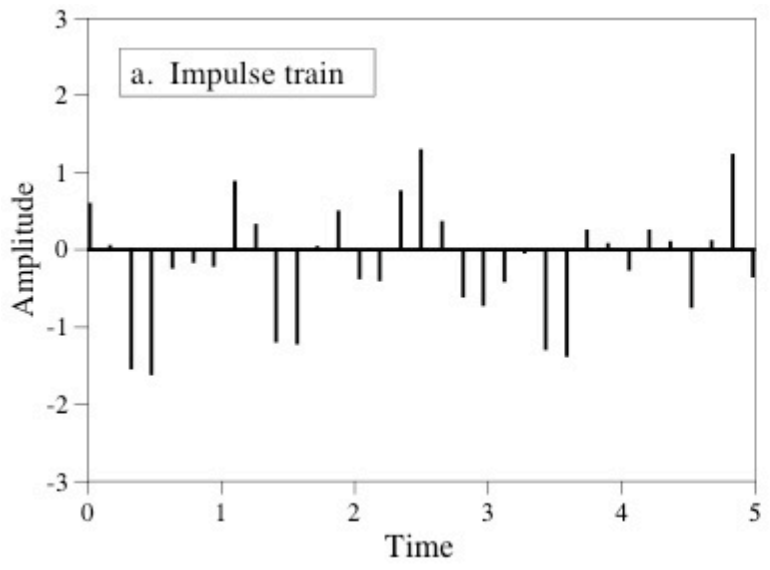
- Signal reconstruction from sampled data

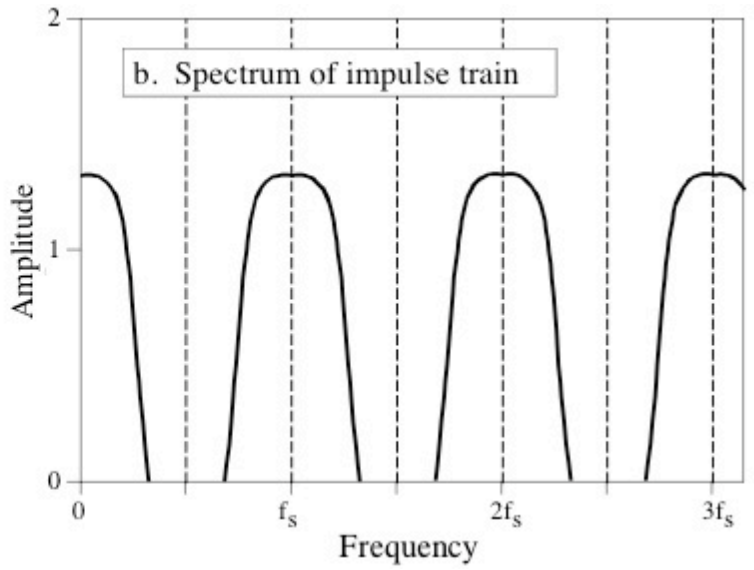
III. DFT and FFT

Digital-to-analog conversion

The DAC reverses the ADC process:

(1) It decodes the signal making a conversion from a bit sequence to an impulse train:

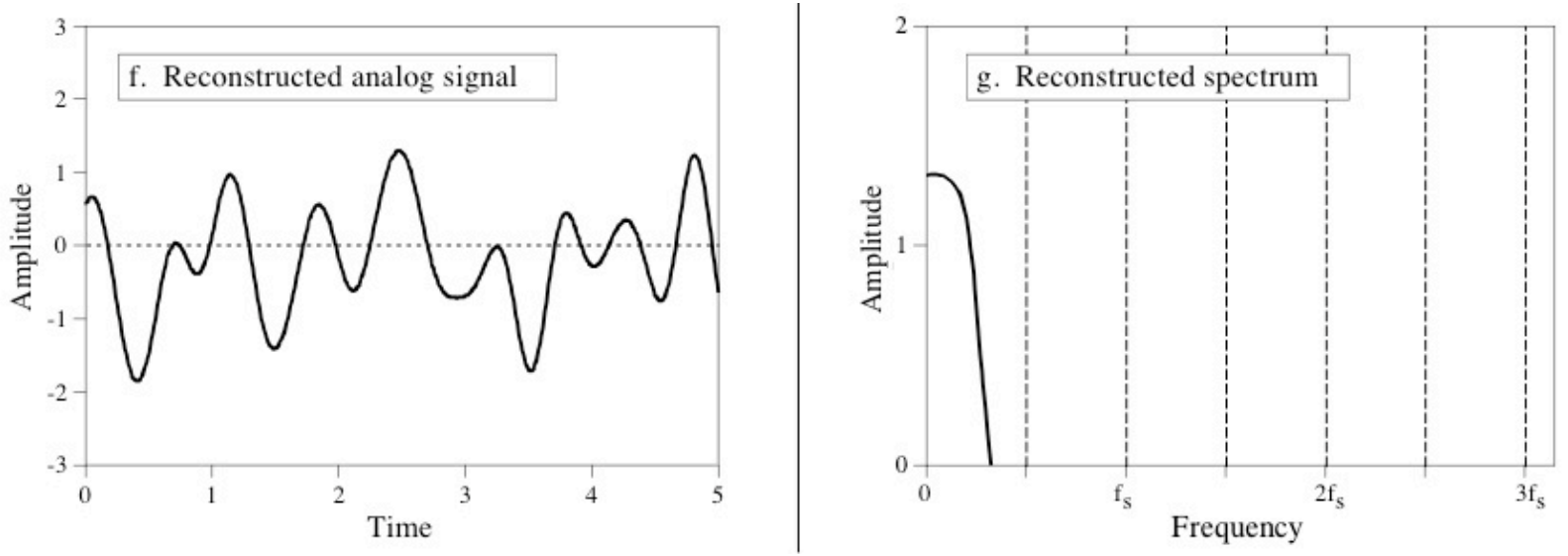




Digital-to-analog conversion

(2) The signal is reconstructed with an electronic low-pass filter to remove the frequencies about ½ the sampling rate

After filtering the impulse train with a such a low pass filter, we would obtain:



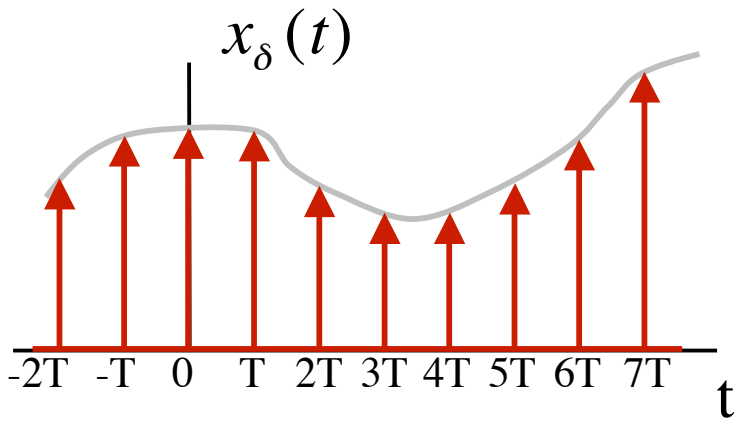
Digital-to-analog conversion

However, the ideal operation just described assumes availability of infinitely many samples - not realistic!

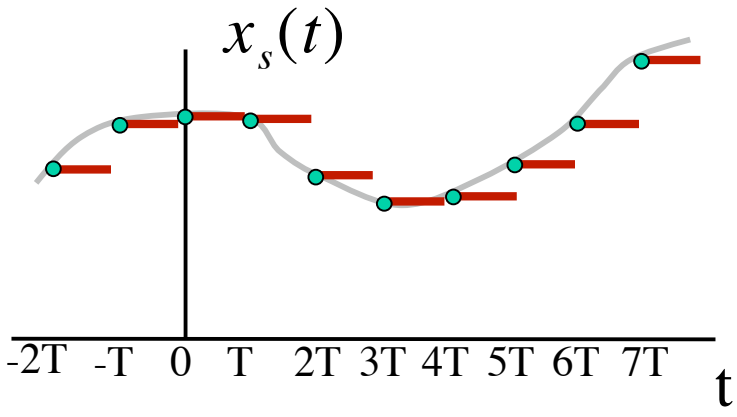
Practical operation uses only a finite number of samples. Many techniques can be used to approximately reconstruct the signal.

One such technique is a **zero-order holder**.

Modulated impulses

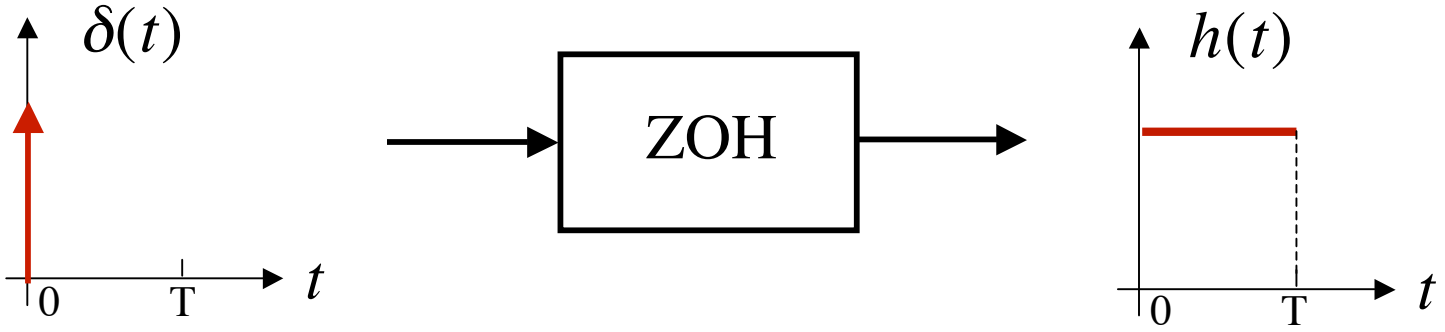


After Zero-Order Holder



Sampling with Zero-order Holder

A zero-order holder has impulse response

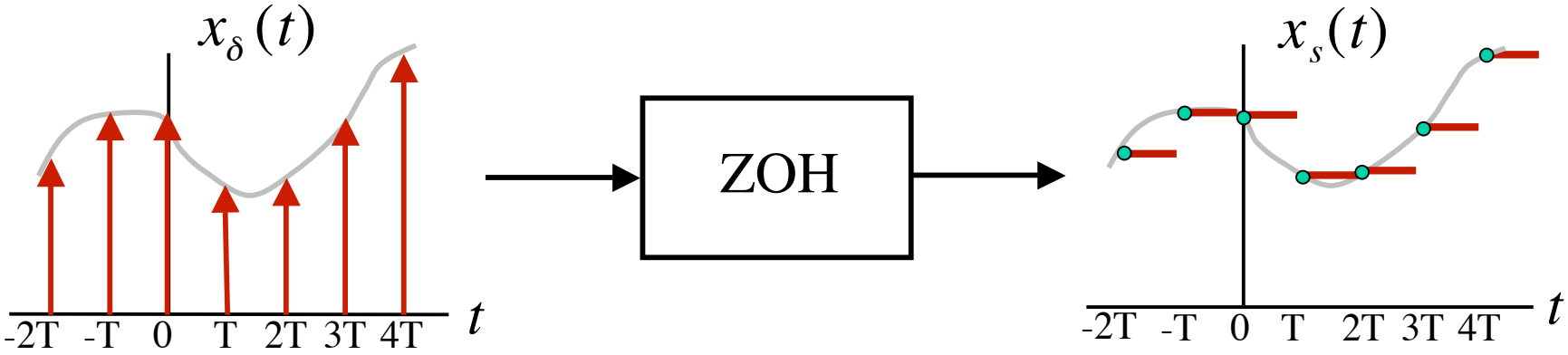


In the frequency domain

$$H(f) = \mathcal{F}\left\{ rect\left(\frac{t - T_s/2}{T_s} \right) \right\} = e^{-j\pi f T_s} \operatorname{sinc}\left(f T_s \right)$$

Sampling with Zero-order Holder

A sampled signal through a zero-order holder



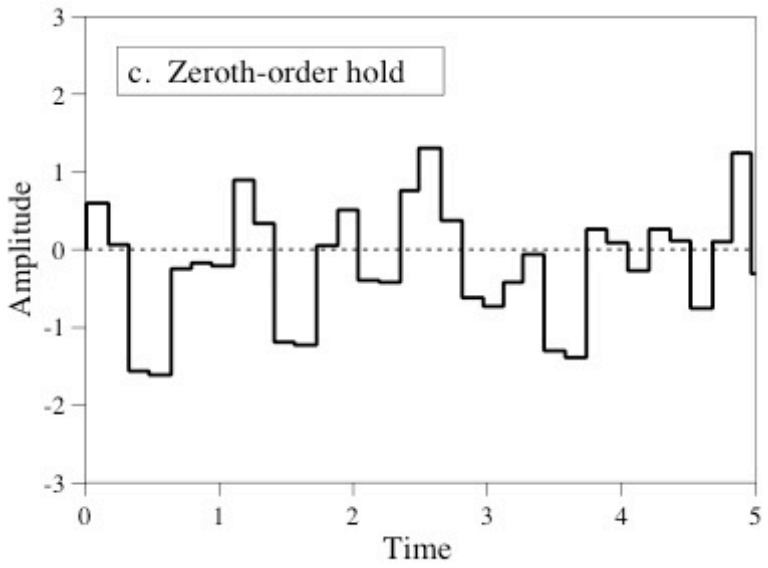
In the frequency domain

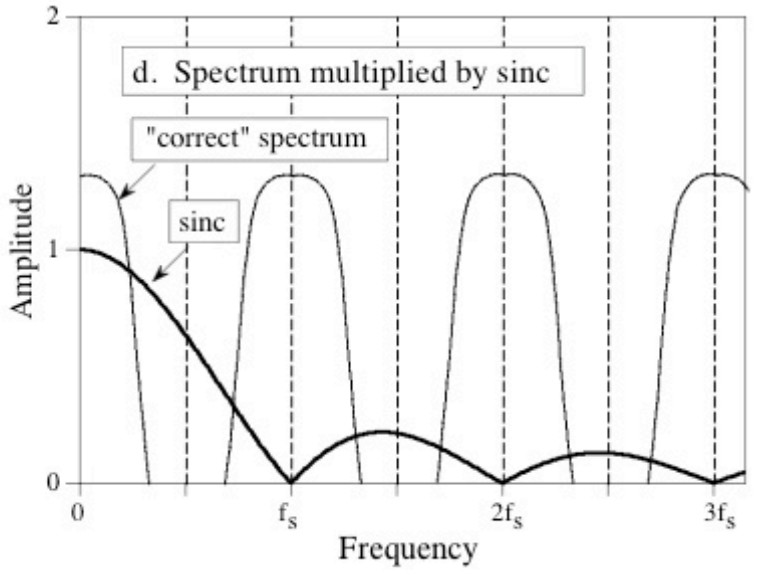
$$X_s(f) = \mathcal{F}\left\{ x_\delta(t) * rect\left(\frac{t - T_s/2}{T_s} \right) \right\}$$

$$= \frac{e^{-j\pi f T_s} \operatorname{sinc}(f T_s)}{T_s} \sum_{k=-\infty}^{\infty} X(f - n f_s)$$

Sampling with Zero-order Holder

In the frequency domain, the zero-order holder translates into a multiplication of the real signal spectrum by a sinc function!





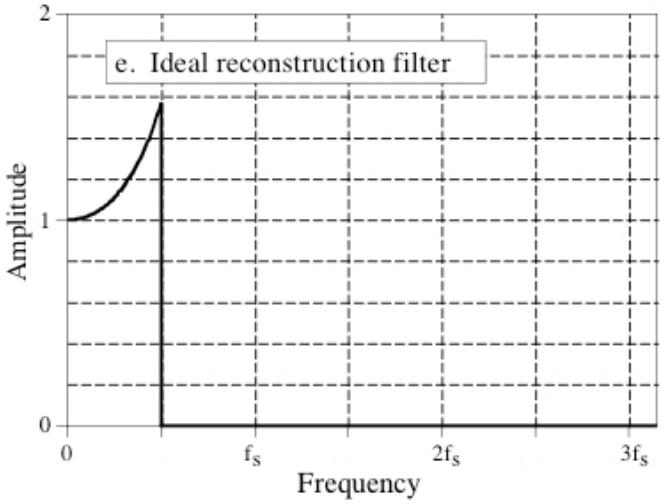
Digital-to-analog conversion

In this case you can do several things:

(1) Ignore the effect of the zeroth-order hold and accept the consequences

(2) Design an analog filter to remove the sinc effect. For example an ideal filter that works has this shape:

This will boost the frequencies that have been affected by the sinc function



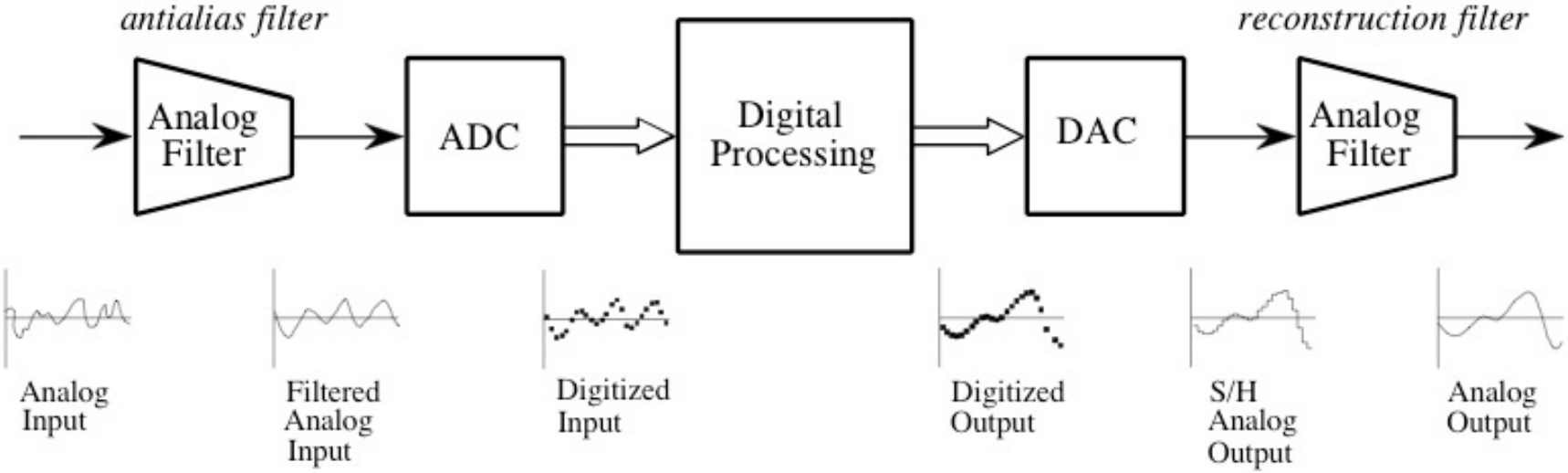
Other more sophisticated possibilities are:

(3) Use a fancy "multirate filter" (downsampling, upsampling)

(4) Correct for the zeroth order hold before the DAC

Connection to electronic filters

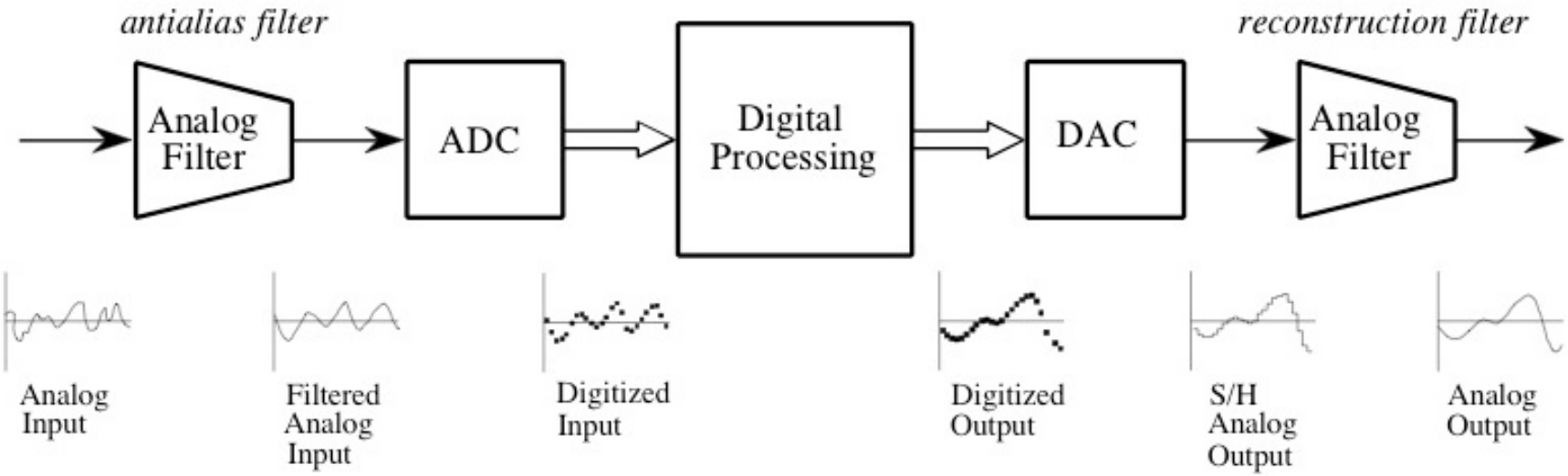
The full block diagram of a DSP system is the following:



Analog electronic filters are used to comply with the sampling theorem. The filter placed before ADC is an antialias filter. It removes frequencies higher than half the sampling rate. The filter placed after the DAC is a reconstruction filter. It may include a correction for the zeroth-order hold.

Filters for data conversion

Problem with this design:



Analog filters are not ideal! Thus, knowledge of lowpass filters is important to be able to deal with different types of signals (analog filters are being substituted by digital ones)

The filters you choose depend on the application!

ADC and DAC

Goals

I. From analog signals to digital signals (ADC)

- The sampling theorem
- Signal quantization and dithering

II. From digital signals to analog signals (DAC)

- Signal reconstruction from sampled data

III. DFT and FFT

What is the CTFT of an impulse-sampled signal?

We have already seen

$$x_\delta(t) = x(t)\delta_{T_s}(t) \longleftrightarrow X_\delta(f) = X(f) * f_s\delta_{f_s}(f) = f_s \sum_{n=-\infty}^{n=\infty} X(f - nf_s)$$

Alternatively,

$$x_\delta(t) = \sum_{n=-\infty}^{n=\infty} x(nT_s)\delta(t - nT_s) \longleftrightarrow X_\delta(f) = \sum_{n=-\infty}^{n=\infty} x(nT_s)e^{-j2\pi fnT_s}$$

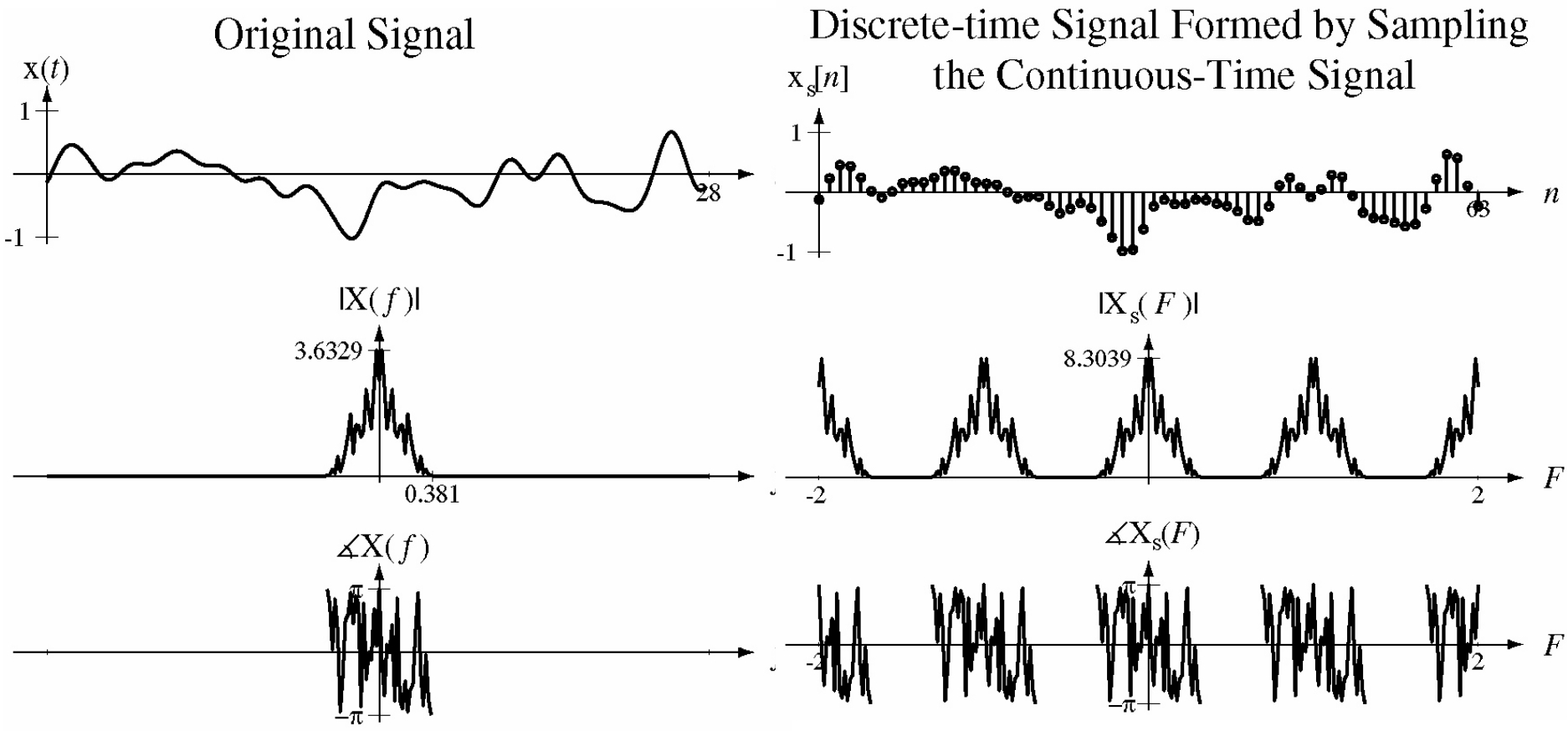
Define the Discrete-Time Fourier Transform of $x[n]$

$$X_F(F) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j2\pi Fn}$$

Then

$$X_\delta(f) = X_F(fT_s)$$

The Discrete-Time Fourier Transform



The Discrete-Time Fourier Series

The Discrete-Time Fourier Transform of discrete-time signal $x[n]$ is

$$X(F) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j2\pi Fn} \longleftrightarrow x[n] = \int_{0}^{1} X(F) e^{j2\pi Fn}\, dF$$

$X(F)$ is periodic in $F \in (0,1]$

The Discrete-Time Fourier Series of periodic discrete-time signal $x[n]$ of period N_0 is ($N_F = mN_0$)

$$X[k] = \frac{1}{N_F} \sum_{k=<N_F>} x[n]\, e^{-j2\pi \frac{nk}{N_F}} \longleftrightarrow x[n] = \sum_{k=<N_F>} X[k] e^{j2\pi \frac{nk}{N_F}}$$

The DTFT of a periodic signal is the DTFS:

$$X(F) = \sum_{k=-\infty}^{k=\infty} X[k]\, \delta(F - k/N_0)$$

The Discrete Fourier Transform (DFT)

Instead of the DTFS, Matlab implements the Discrete Fourier Transform,

$$x[n] = \frac{1}{N_F} \sum_{k=0}^{N_F-1} X[k] e^{j2\pi \frac{nk}{N_F}} \quad \overset{DFT}{\longleftrightarrow} \quad X[k] = \sum_{n=0}^{N_F-1} x[n] e^{-j2\pi \frac{nk}{N_F}}$$

which is almost identical to the DTFS

$$x[n] = \sum_{k=0}^{N_F-1} X[k] e^{j2\pi \frac{nk}{N_F}} \quad \overset{FS}{\longleftrightarrow} \quad X[k] = \frac{1}{N_F} \sum_{n=0}^{N_F-1} x[n] e^{-j2\pi \frac{nk}{N_F}}$$

The difference is only a scaling factor

Two different names for essentially the same object because of historical reasons.

Computing the DFT

One could write a Matlab program to compute the DFT like this

```
.
.
%         (Acquire the input data in an array x with NF elements.)
.
.
%
%         Initialize the DFT array to a column vector of zeros.
%
X = zeros(NF,1) ;
%
%         Compute the Xn's in a nested, double for loop.
%
for k = 0:NF-1
          for n = 0:NF-1
                    X(k+1) = X(k+1)+x(n+1)*exp(-j*2*pi*n*k/NF) ;
          end
end
.
.
```

Quadratic in N_F. Instead, the fast Fourier transform (matlab command **fft**) is an efficient algorithm for computing the DFT.

Fast Fourier Transform (FFT)

The algorithm FFT can compute the DFT in

$O(N_F \log_2 N_F)$ operations

The FFT takes an *N*-sample time-sequence $\{x_n\}$

And gives us an *N*-sample frequency-sequence $\{X_k\}$

This is reversible via the IFFT operation

No data is lost in either direction

Recall periodicity of DFT $X[k + N_F] = X[k]$

The FFT describes the signal in terms of its frequency content

If we sampled fast enough, this is the same as the Fourier Transform of the original continuous-time signal

The FFT is a primary tool for data analysis

Speed comparison DFT/FFT

Below is a speed comparison for various numbers of samples (N). (A means # additions and M means # multiplications.)

γ	$N = 2^{\gamma}$	A_{DFT}	M_{DFT}	A_{FFT}	M_{FFT}	A_{DFT} / A_{FFT}	M_{DFT} / M_{FFT}
1	2	2	4	2	1	1	4
2	4	12	16	8	4	1.5	4
3	8	56	64	24	12	2.33	5.33
4	16	240	256	64	32	3.75	8
5	32	992	1024	160	80	6.2	12.8
6	64	4032	4096	384	192	10.5	21.3
7	128	16256	16384	896	448	18.1	36.6
8	256	65280	65536	2048	1024	31.9	64
9	512	261632	262144	4608	2304	56.8	113.8
10	1024	1047552	1048576	10240	5120	102.3	204.8

The z-Transform

What is the Laplace transform of a sampled signal?

$$X_s(s) = \int_0^\infty x(t)\,\delta_T(t)\,e^{-st}\,dt$$

$$= \int_0^\infty x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)\,e^{-st}\,dt = \sum_{n=0}^{\infty} x(nT) e^{-snT}$$

$$= X(e^{sT})$$

where

$$X(z) = \sum_{n=0}^{\infty} x_n\ z^{-n}$$

is the **z-Transform**. The inverse formula is a bit more involved but often not needed!

Generalizes the DTFT. Surprised?

Discrete-time Systems with the z-Transform

Some properties of the z-Transform

$$Z\{x_{n-1}\} = z^{-1} Z\{x_n\}$$ (integration in Laplace)

$$Z\{x_{n+1}\} = z\left(Z\{x_n\} - x_0\right)$$ (differentiation in Laplace)

$$Z\{x_n * y_n\} = Z\{x_n\}Z\{y_n\}$$ (convolution)

Proofs are easier, e.g.

$$Z\{x_{n-1}\} = z^{-1}\left(\sum_{k=1}^{\infty} x_{k-1} z^{-k}\right) z = z^{-1}\sum_{k=1}^{\infty} x_{k-1} z^{-(k-1)} = z^{-1}\sum_{n=0}^{\infty} x_n z^{-n} = z^{-1} Z\{x_n\}$$

Discrete-time Systems with the z-Transform

We use the basic property

$$Z\{x_{n-1}\} = z^{-1}\, Z\{x_n\}$$

(integration in Laplace)

to study the system described by the linear difference equation

$$y_n - \alpha\, y_{n-1} = x_n$$

as with Laplace trasforms we have

$$Y(z) - \alpha\, z^{-1} Y(z) = X(z)$$

from where we obtain the transfer function

$$Y(z) = H(z)X(z) \qquad H(z) = \frac{z}{z-\alpha} = \frac{1}{1-\alpha z^{-1}}$$

Discrete-time Systems with the z-Transform

When $x_n = \delta_n$ is a pulse ($\delta_0 = 1, \qquad \delta_k = 0, k \neq 0$)

$$X(z) = Z\{x_n\} = Z\{\delta_n\} = 1$$

and

$$Y(z) = H(z) = \frac{1}{1 - \alpha z^{-1}} = 1 + \alpha z^{-1} + \alpha^2 z^{-2} + \alpha^3 z^{-3} + \cdots = \sum_{n=0}^{\infty} \alpha^n z^{-n}$$

from where we compute the **impulse response**

$$h_n = Z^{-1}\{H(z)\} = \alpha^n$$

Discrete-time Systems with the z-Transform

For **stability** we need

$$\sum_{n=-\infty}^{\infty}|h_n| = \sum_{n=0}^{\infty}|\alpha^n| < \infty$$

which converges if $|\alpha| < 1$ (pole inside the unit circle)

A discrete-time **convolution** formula holds

$$y_n = \sum_{k=0}^{n} h_{k-n} x_k$$

and $X(e^{j\Omega})$ is the DTFT

We can do frequency domain analysis directly in the z-domain!

Discretized Continuous-Time Systems

Response of a continuous-time LTI system to a sampled input

$$x_\delta(t) = T\, x(t)\, \delta_T(t)$$

is given by the convolution

$$y_\delta(t) = \int_{-\infty}^{\infty} x_\delta(\tau)\, h(t-\tau)\, d\tau$$

$$= T \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} x(\tau)\, \delta(\tau - nT)\, h(t-\tau)\, d\tau$$

$$= \sum_{n=0}^{\infty} x(nT)\, T\, h(t-nT)$$

Discretized Continuous-Time Systems

If we sample the output at multiples of T

$$y[n] = y_\delta(nT) = \sum_{k=0}^{\infty} x[k]\, h[n-k] = x[n] * h[n]$$

which is a **discrete-time convolution** if we define

$$h[n] = T\, h_\delta(nT) = T\, h(nT)$$

From there we can obtain the z-transform of the discretized system

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

Discretized Continuous-Time Systems

For example

$$h(t) = be^{-at}u(t) \qquad H(s) = \frac{b}{s+a}$$

we have

$$H(z) = \sum_{n=0}^{\infty} Tbe^{-anT} z^{-n} = Tb\sum_{n=0}^{\infty}\left(e^{-aT}\right)^n z^{-n} = \frac{Tb}{1-e^{-aT}z^{-1}}$$

$$= \frac{Tbz}{z-e^{-aT}}$$

Continuous-time pole $s=-a$ becomes $z=e^{-aT}$!