DSP Lecture (03/01/2023. Wednesday.)

Review: sampling theorem = $f_s > 2f_{max}$

f_{max} = highest frequency of interest

↳ tells us how to determine f_s

but we have lots of options for f_s!

what's the trade-off of higher/lower f_s?

⇒ runtime implementation complexity of ...

$f_s = 4f_{max}$ vs. $f_s = 8f_{max}$

⇒ storage of samples/second is doubled

★ f_s = in units of "Hz" or "samples/second"

⇓

you need f_s samples in 1 second

for FIR filter with N coefficients, ⇒ means higher implementation complexity

needs N multiplications/sample

what's the rate the FIR filter operates at?

→ at a rate f_s samples/second

→ filter operates at $N \cdot f_s$ multiplications/second

⇒ generally, as f_s increases, filter order N of FIR filter or IIR filter also increases in order to meet the more stringent transition region bandwidth requirement

★ note: in MATLAB demonstration, $f_s = 24 f_{max}$ to plot filter responses (to get smooth plots)

★ ⇒ at some point, you cannot hear/see an improvement in signal quality as f_s increases. (diminishing returns as f_s increases)

→ @ 10:44AM, moving to lecture slides, # 4, Sampling Theorem

Nyquist Rate : $2f_{max}$

Nyquist Frequency : $\frac{1}{2}f_s$ ⇒ max frequency you can capture

3 idealizations for Sampling Thm. : (unrealistic assumptions)

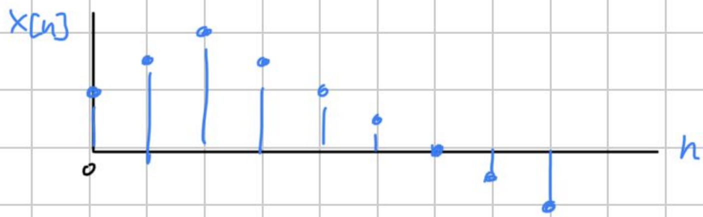
1. 2 sided observation is time
2. no thermal noise
3. perfect rectangular pulse filter

how to actually "recover your C.T. signal from D.T. sampled signal"?

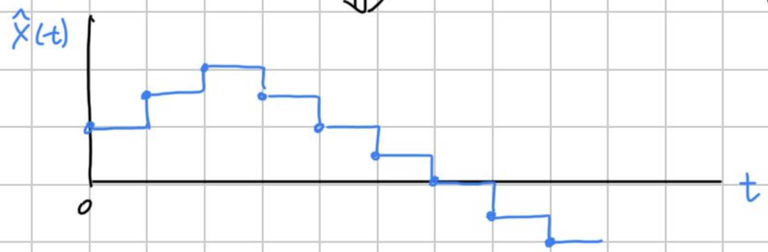
→ "reconstruction of your original signal"

Reconstruction Options

1. sample & hold (slide 4-6)



⇓



"stair case approximation"

→ very blocky

↳ can be smoothed with a Lowpass Filter

☆ a quick aside : using zero-crossings to estimate the frequency of a signal.

→ count # zero-crossings

divide by 2

→ that is an estimate of your signal frequency.

(demos @ 11:00 AM)

11:00 ① zero-order hold / sample & hold / square pulse shaping

⇒ in time domain, it is sinc(·) fxn

11:05 ② linear interpolation / triangular pulse shaping

⇒ falls off faster in time domain

(in time domain, triangular pulse is $\mathrm{sinc}^2(\cdot)$ fxn)

11:08 ③ truncated sinc pulse shaping

⇒ you have more flexibility w/ this pulse shape

⇒ you can select the width of sinc pulse

⇒ width affects implementation complexity ☆

(break @ 11:11 ish?)

(return @ 11:17 AM)

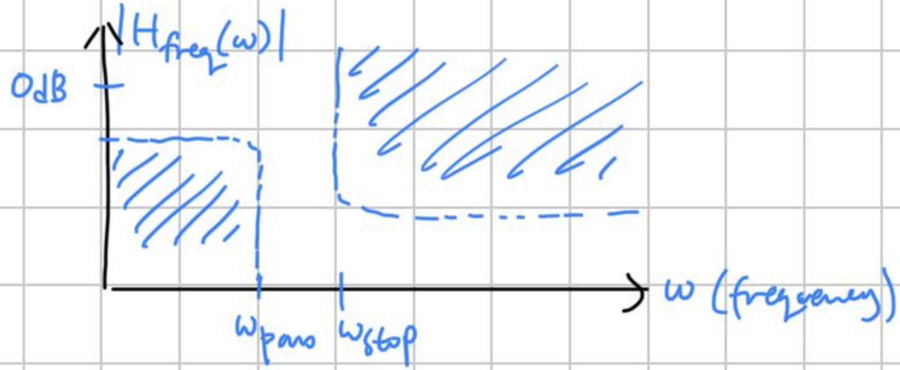
Why does FIR filter order increase with f_s?

⇒ magnitude specifications is in Hz: f_{pass}, f_{stop}

$$x(t) = \cos(2\pi f_0 t)$$
$$x[n] = x(nT_s) = x\left(\frac{n}{f_s}\right) = \cos\left(2\pi f_0 \left(n/f_s\right)\right) = \cos\left(2\pi \frac{f_0}{f_s} n\right)$$

$$\omega_{pass} = 2\pi f_{pass}/f_s$$
$$\omega_{stop} = 2\pi f_{stop}/f_s$$

Lowpass Filter Design Example



⇒ so as f_s increases, filter order increases because...

1. passband width decreases
2. transition band decreases
   ↳ $\Delta\omega = \omega_{stop} - \omega_{pass} = \frac{2\pi}{f_s}(f_{stop} - f_{pass})$

⇒ to meet these specifications, N (filter order) must increase

(@ 11:26 AM return to lecture slides) slides 4-3

Sampling (Analog to Digital Conversion)

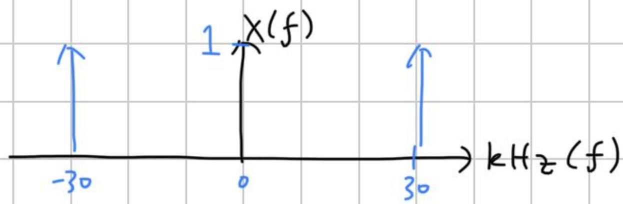
⇒ to remove effects of aliasing ⇒ apply Lowpass Filter First! (before sampling & quantization)

(this removes frequencies that would alias)

Aliasing : Sinusoidal Example (with only sampling)

⇒ notation for sampling
we sample incoming signal x(t) at sampling rate fs to get the sampled signal y(t)

$f_0 = 30\text{kHz}$
$f_s = 24\text{kHz}$



⇓ sampling @ 24kHz

aliasing, NOT good!

(demo at 11:39 AM) ⇒ visual demonstration of Aliasing

return to slides @ 11:45 AM

⇒ slides 4-9, Increasing Sampling Rates

⇒ (interpolation demo @ 11:48 AM)