Multirate Systems

Ex: • YouTube fullscreen

increased the rate

• Watching videos at 2x speed.

Recall:

$x[n] = x_c(nT)$, $\frac{1}{T}$ sampling rate

Downsample by factor 2:

$(\downarrow 2)\, x[n] = x[2n] = x_c(2nT)$, $\frac{1}{2T}$ sampling rate — Slower

Intuition:

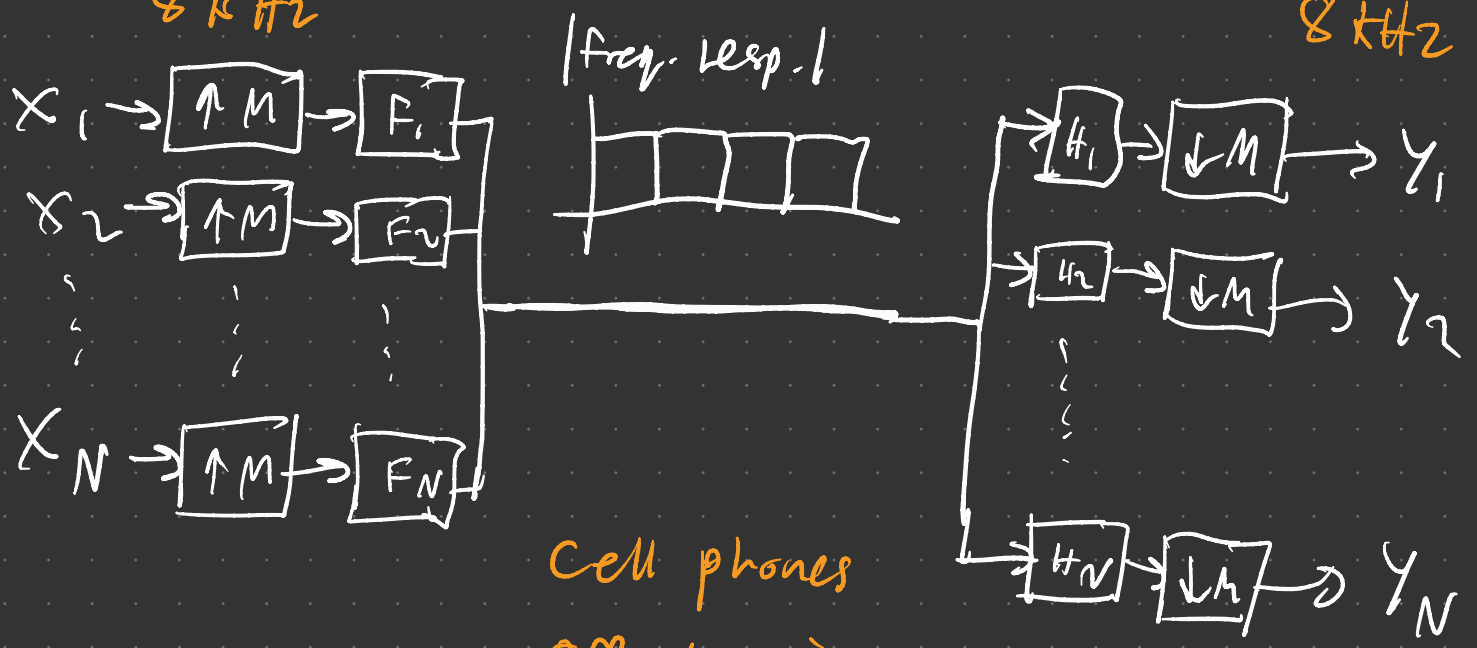
Downsampling reduces bandwidth

Upsampling increases bandwidth

Ex:

Filter Bank

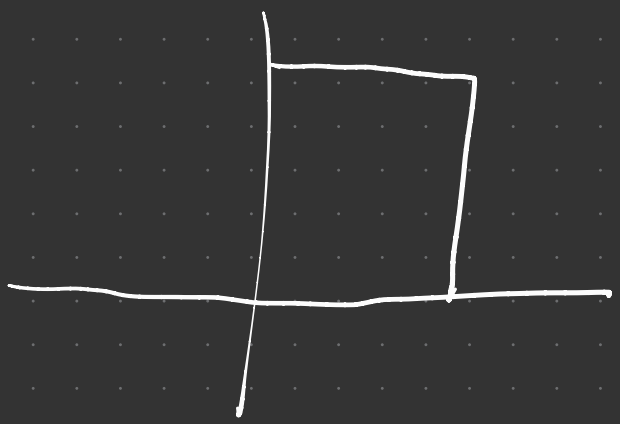
Voice is sampled at 8 kHz



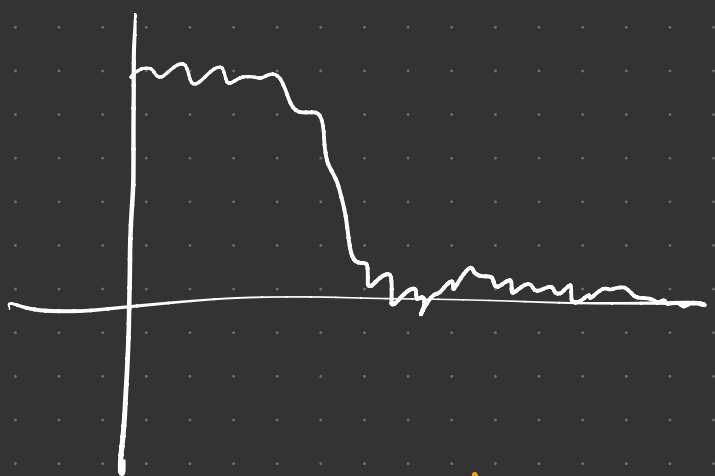
Cell phones operate in MHz of GHz CRF spectrum)

Also how cable TV works

$|H_i(e^{j\omega})|$



ideal filter



actual filter

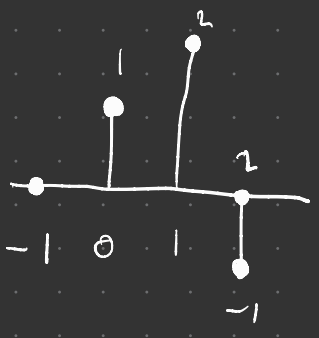
Upsampling & Interp by factor 2

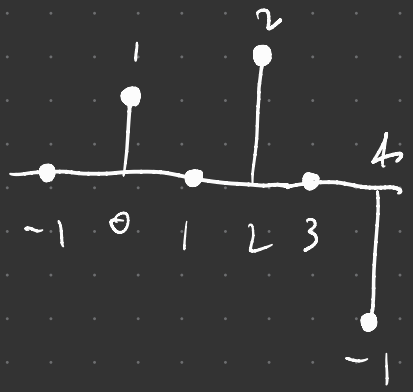
Q: What is upsampling?

A: Putting zeros between samples

upsampling is always followed by interpolation

Ex:





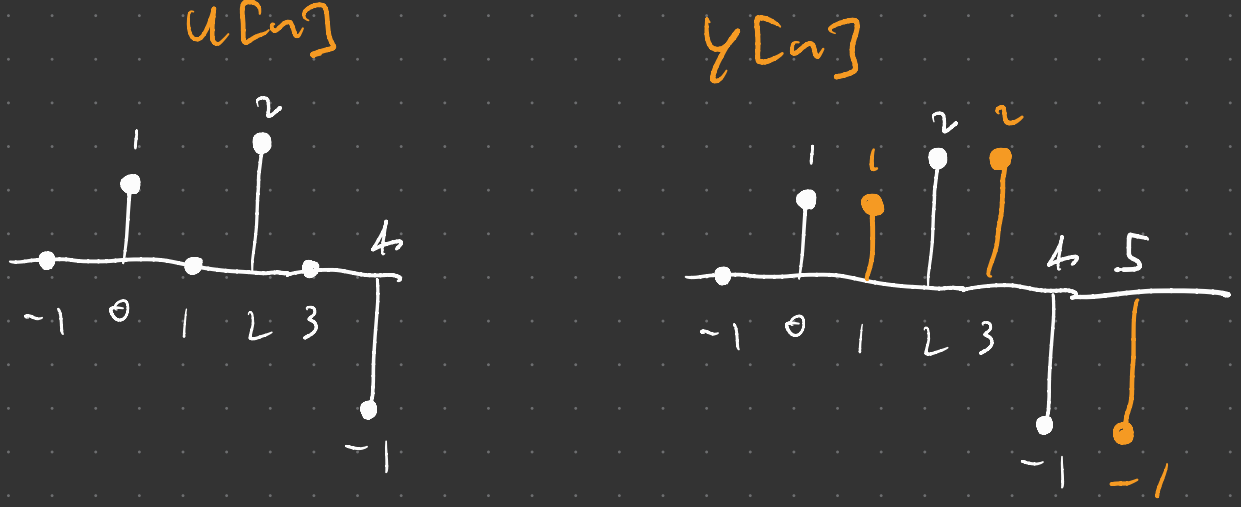
not useful e.g. in an image this would be dark lines

Q: what is the job of the filter?

A: To interpolate the zeros.

Q: what's the easiest way to interpolate?

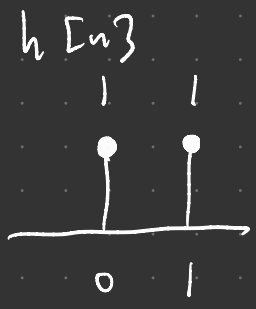
A: Repeat value to the left.



Exercise: What is $h[n]$ such that

$$(h * u)[n] = y[n] \; ?$$

Soln:



Why?:

$$y[n] = u[n] + u[n-1]$$

$$h[n] = \delta[n] + \delta[n-1]$$

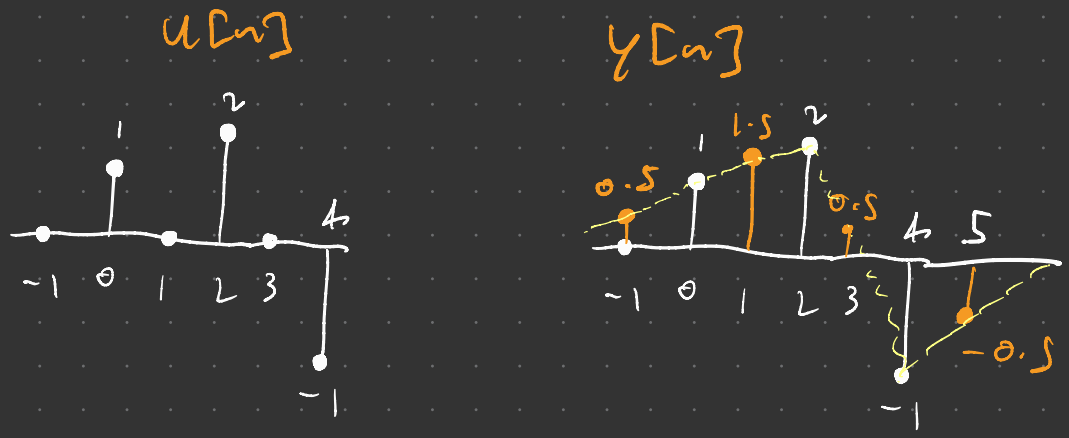
Q: Why is this bad?

A: Blocking artifacts.

(Imagine upsampling by factor 8)

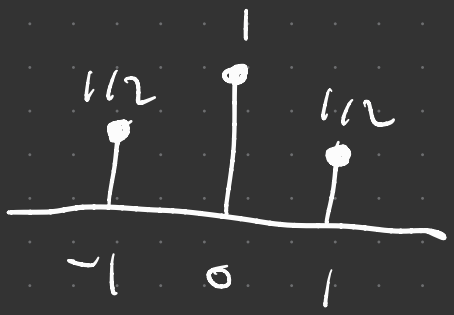
Remark: Interpolation filter design is a kind of art.

Q: What about linear interpolation?



Exercise: what is $h[n]$?

Soln:



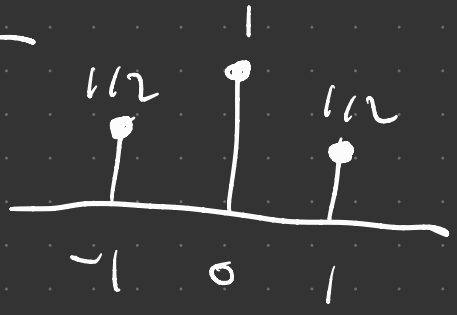
$$y[n] = \frac{u[n+1] + u[n-1]}{2} + u[n]$$

$$h[n] = \frac{1}{2}\delta[n+1] + \delta[n] + \frac{1}{2}\delta[n-1]$$

Q: Why is this bad?

A: Looks too sharp. We want something that looks smooth.

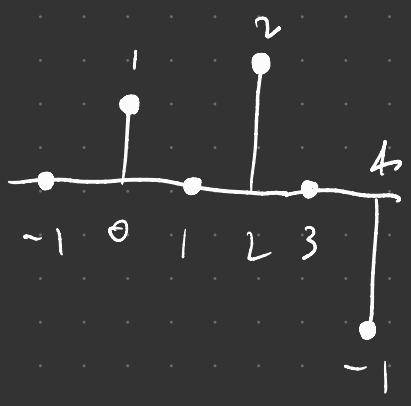
Obs:



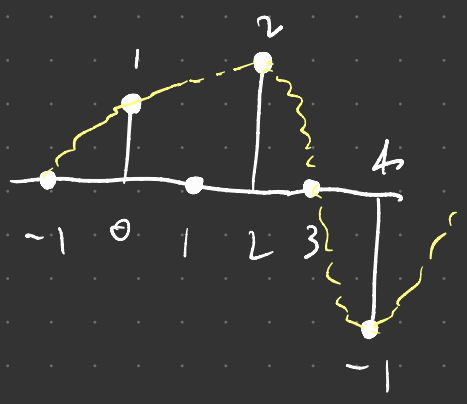
looks kinda like a sinc ⇒ low-pass

Goal:

u[n]

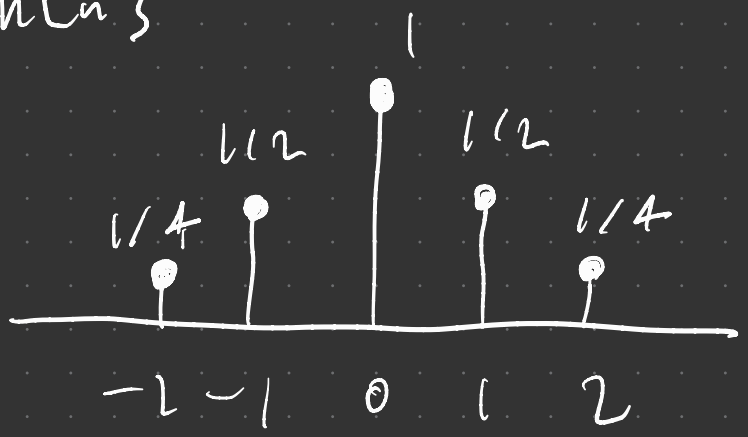


y[n]



Exercise: What about the filter

h[n]



?

Obs: It's bad because it changes the original samples!

Check what happens @ $y[0]$

$$y[0] = 1 + 0 + \frac{1}{4}(2) = \frac{3}{2} \neq u[0]$$

we destroyed the original samples!

Goal: Interpolate and preserve samples.

Obs: The issue is at even coeff. of $h[n]$.

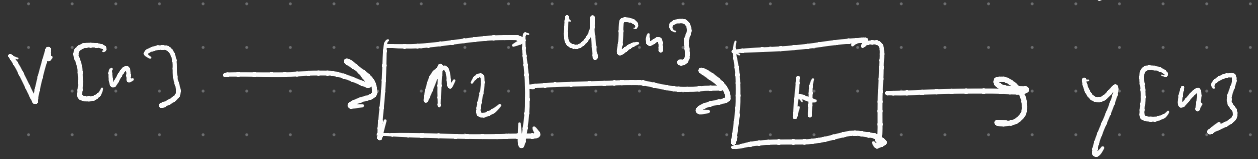
Property: An interpolation filter must satisfy

$$h[2n] = \delta[n] \quad \rightarrow \quad (\downarrow 2)\, h[n]$$

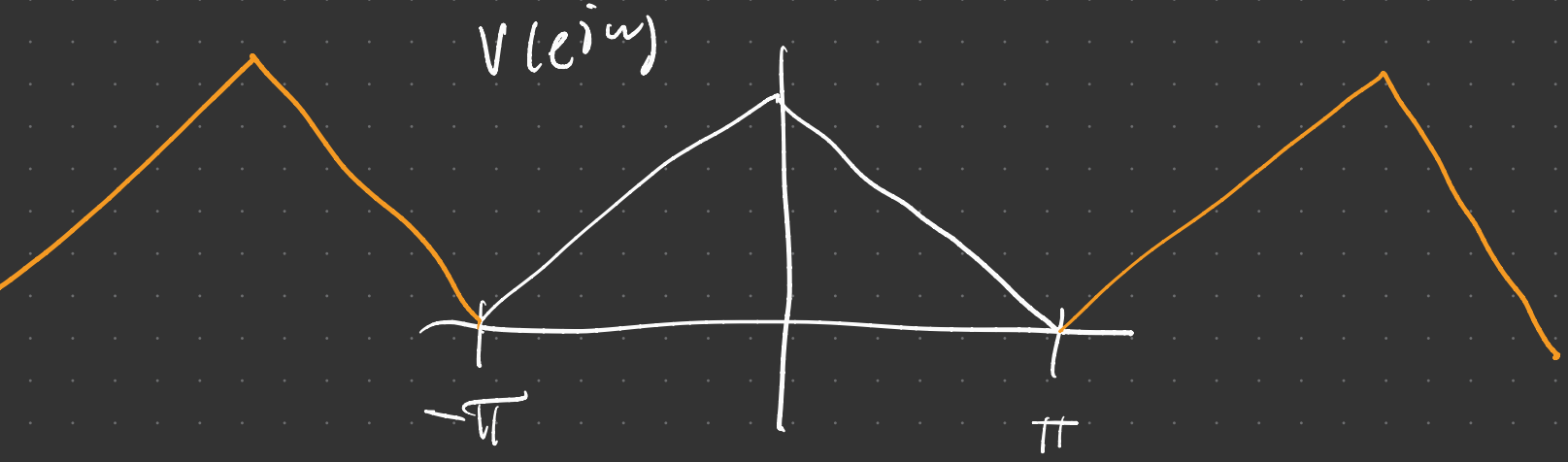
half-band condition

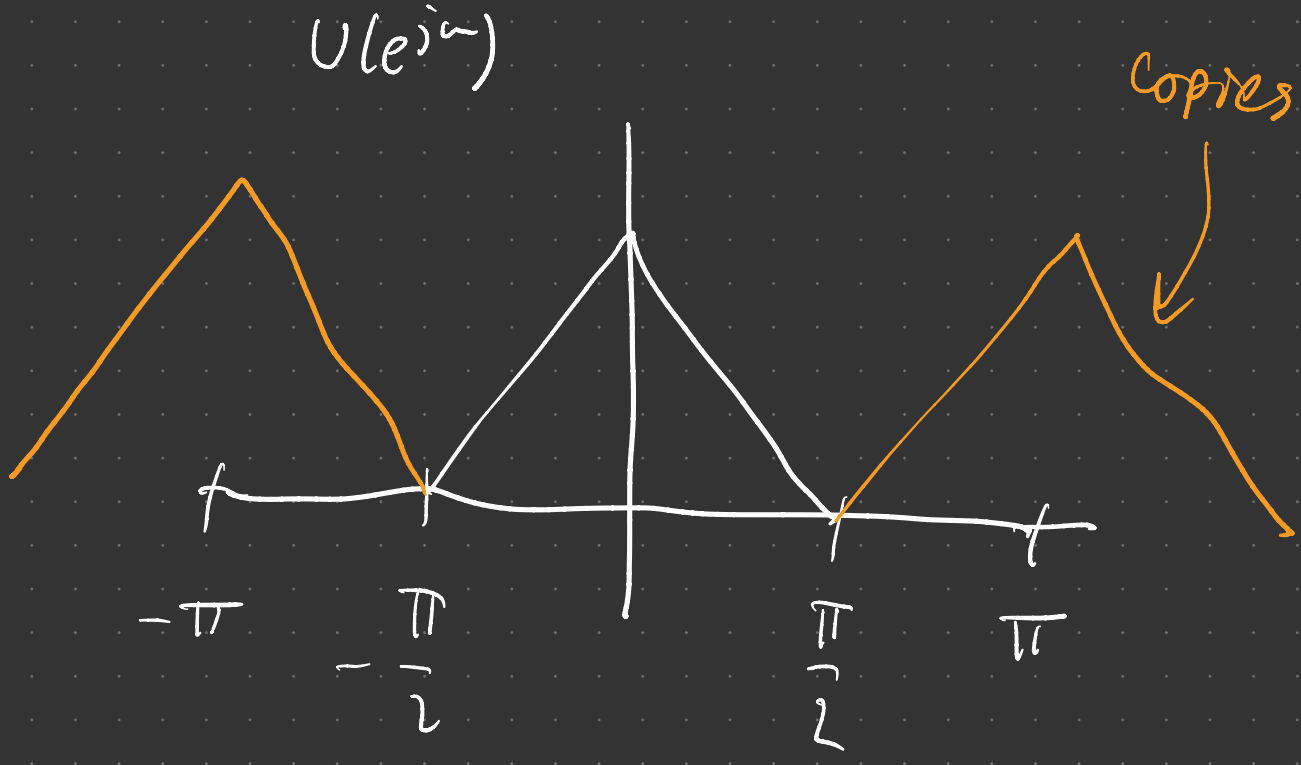
Obs: Interpolation filter design is only about the odd coeff.

Frequency-Domain Analysis



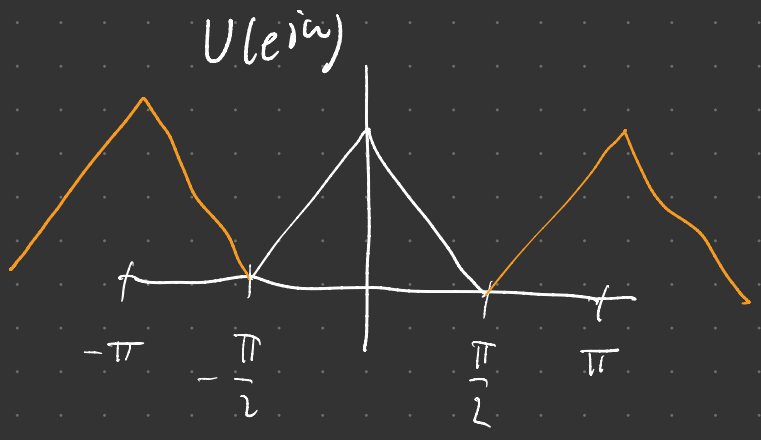
$$U(z) = V(z^2) \Rightarrow U(e^{j\omega}) = V(e^{j2\omega})$$

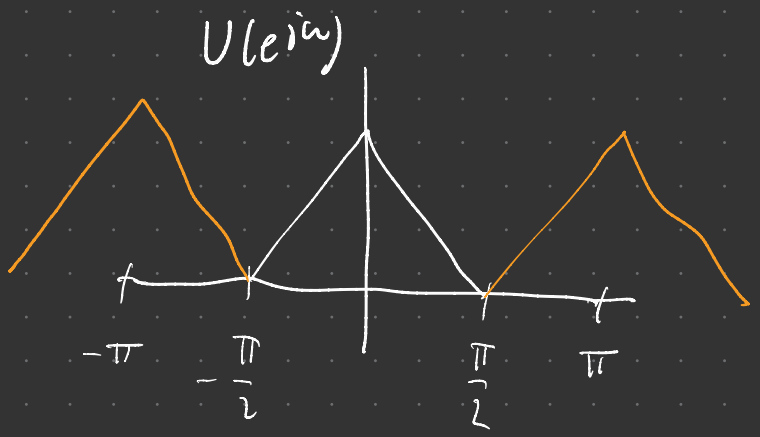


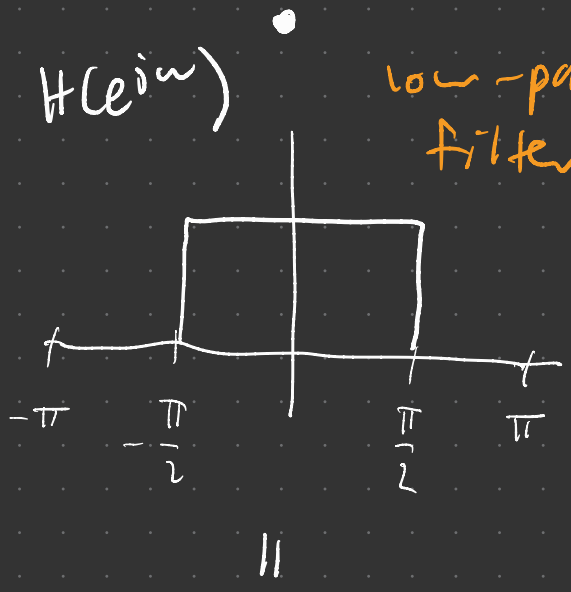


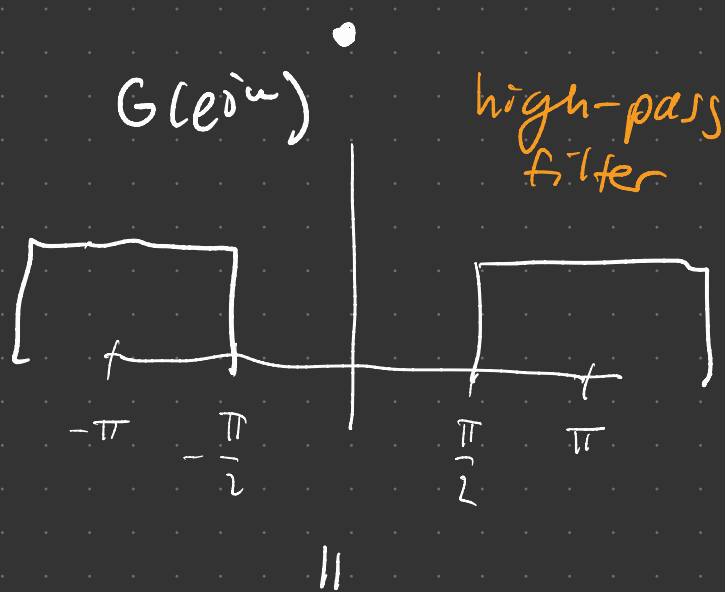
Remark: We don't want the spectrum to look like this!

Two Options:

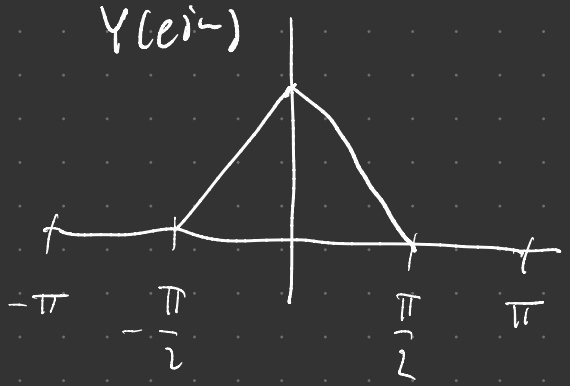


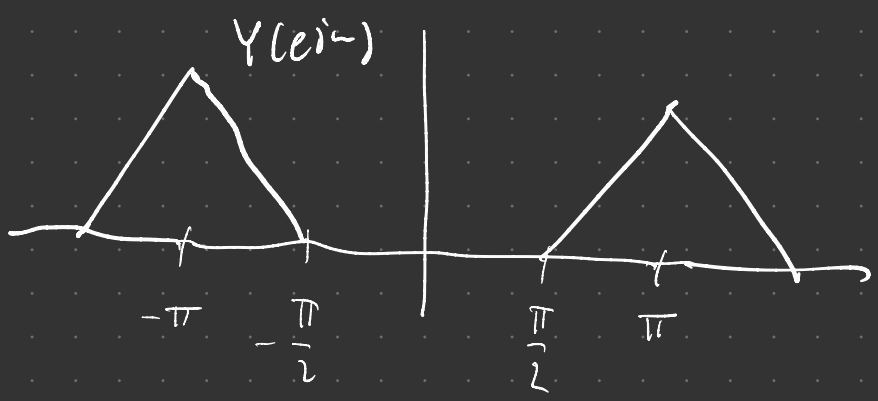






$\parallel$ $\parallel$





Low-pass Interp.

High-pass Interp.

We have already seen low-pass interpolation.

Q: What is high-pass interpolation?

Q: Given a low-pass filter, is there a corresponding high-pass filter?

A: Yes.

$$G(e^{j\omega}) = H(e^{j(\omega-\pi)})$$ shift in freq.

$$g[n] = e^{j\pi n} h[n]$$ modulation in time

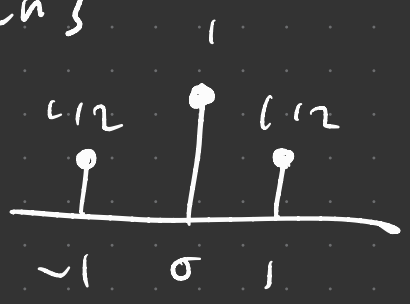
$$= (-1)^n h[n]$$

obs: If $h[n]$ satisfies half band conditions, then so does $g[n]$. (even index stays the same).

Trick: Flip every other coeff of $h[n]$ to get $g[n]$.

Exercise: Given the linear interp-filter

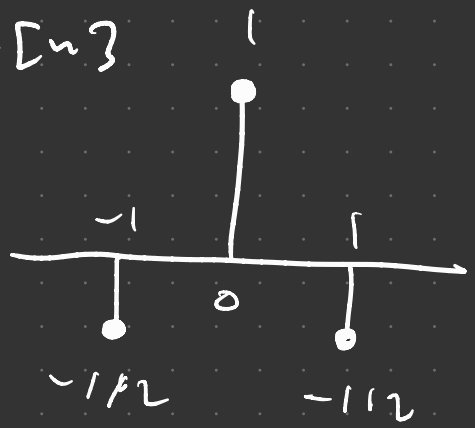
$h[n]$



what is the corresponing high-pass filter $g[n]$?

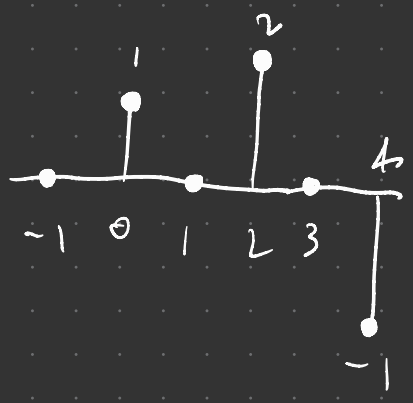
Soln:

$g[n]$

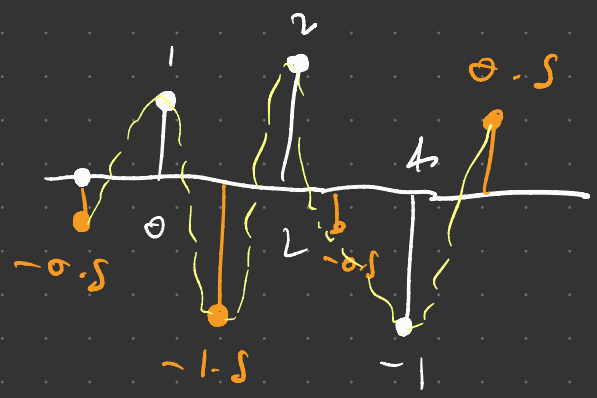


really wiggly
⇒ high-pass

$u[n]$



$y[n]$



Obs: Original samples are preserved because $g[n]$ satisfies the half-band condition.

Q: Is upsampling linear?

A: Yes $\quad \alpha, \beta \in \mathbb{R}$

Proof:

$$(\uparrow 2)\left(\alpha V_1[n] + \beta V_2[n]\right)$$

$$= \begin{cases} \alpha V_1\left[\frac{n}{2}\right] + \beta V_2\left[\frac{n}{2}\right], & n \text{ even} \\ 0, & n \text{ odd} \end{cases}$$

$$= \alpha \begin{cases} V_1\left[\frac{n}{2}\right], & n \text{ even} \\ 0, & n \text{ odd} \end{cases} + \beta \begin{cases} V_2\left[\frac{n}{2}\right], & n \text{ even} \\ 0, & n \text{ odd} \end{cases}$$

$$= \alpha (\uparrow 2) V_1[n] + \beta (\uparrow 2) V_2[n]$$

□

Q: Is upsampling time-invariant?

A: No.

Proof: After upsampling, the signal always has zero for odd coeff.

$V[n] \to [z^{-1}] \to [\uparrow 2] \to Y_1[n]$ odd coeff zero

$V[n] \to [\uparrow 2] \to [z^{-1}] \to Y_2[n]$ even coeff zero

$Y_1[n] \neq Y_2[n]$ ⇒ not time-invariant

□

Remark: This is why multirate DSP is hard: Operations don't commute.