COM303: Digital Signal Processing

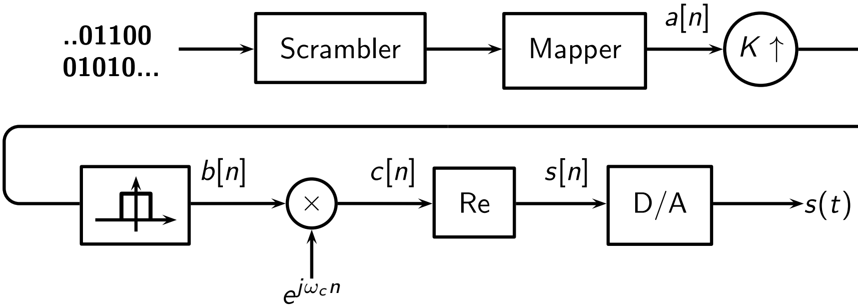
Lecture 23: Digital Communication Systems (II)

overview

- QAM receiver design
- ADSL

receiver design

QAM transmitter, final design



It's a dirty job...

but a receiver has to do it:

- propagation delay
- channel effects
- interference
- clock drift

It's a dirty job...

but a receiver has to do it:

- propagation delay
- channel effects
- interference
- clock drift

It's a dirty job...

but a receiver has to do it:

- propagation delay
- channel effects
- interference
- clock drift

It's a dirty job...

but a receiver has to do it:

- propagation delay
- channel effects
- interference
- clock drift

It's a dirty job...

but a receiver has to do it:

- propagation delay $\rightarrow$ handshake and delay estimation
- channel effects
- interference
- clock drift

It’s a dirty job...

but a receiver has to do it:

- propagation delay $\rightarrow$ handshake and delay estimation
- channel effects $\rightarrow$ adaptive equalization
- interference
- clock drift

It's a dirty job...

but a receiver has to do it:

- propagation delay $\rightarrow$ handshake and delay estimation
- channel effects $\rightarrow$ adaptive equalization
- interference $\rightarrow$ line probing
- clock drift

It's a dirty job...

but a receiver has to do it:

- propagation delay $\rightarrow$ handshake and delay estimation
- channel effects $\rightarrow$ adaptive equalization
- interference $\rightarrow$ line probing
- clock drift $\rightarrow$ timing recovery

A blast from the past

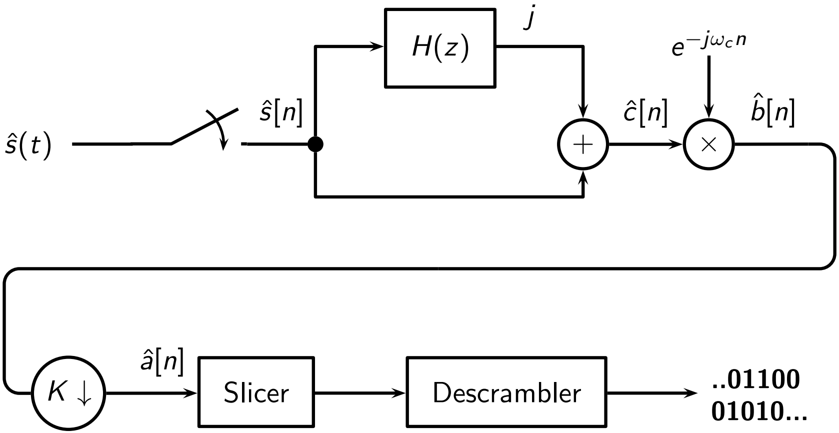
Play

A blast from the past

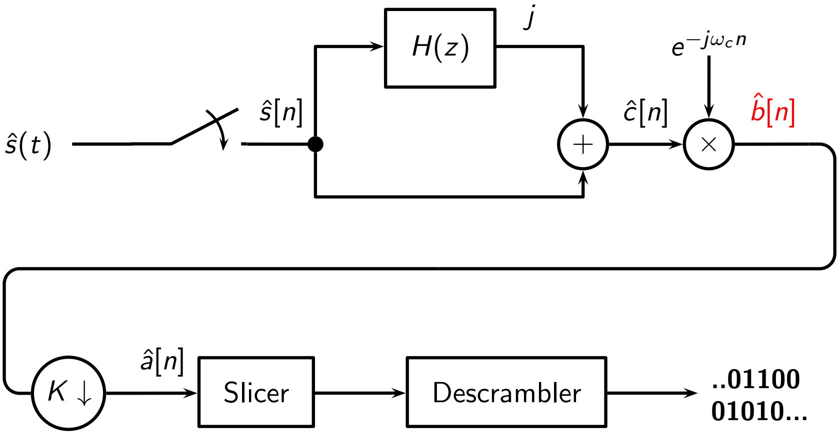
Play

- a sound familiar to anyone who's used a modem or a fax machine
- what's going on here?

Remember the (simplified) receiver



Remember the (simplified) receiver

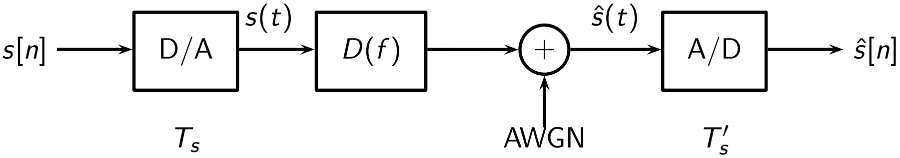


Pilot tones

$$\text{if } \hat{s}[n] = \cos((\omega_c + \omega_0)n) \text{ then } \hat{b}[n] = e^{j\omega_0 n}$$

Visually, in slow motion, plotting $b[n]$

The main problems



- noise
- propagation delay
- channel distortion $D(f)$
- different clocks $(T'_s \neq T_s)$

Delay compensation

Assume the channel is a simple delay: $\hat{s}(t) = s(t-d) \Rightarrow D(f) = e^{-j2\pi fd}$

- channel introduces a delay of d seconds
- we can write $d = (L+\tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- L is called the *bulk delay*
- τ is the fractional delay

Delay compensation

Assume the channel is a simple delay: $\hat{s}(t) = s(t-d) \Rightarrow D(f) = e^{-j2\pi fd}$

- channel introduces a delay of d seconds
- we can write $d = (L+\tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- L is called the *bulk delay*
- τ is the fractional delay

Delay compensation

Assume the channel is a simple delay: $\hat{s}(t) = s(t-d) \Rightarrow D(f) = e^{-j2\pi fd}$

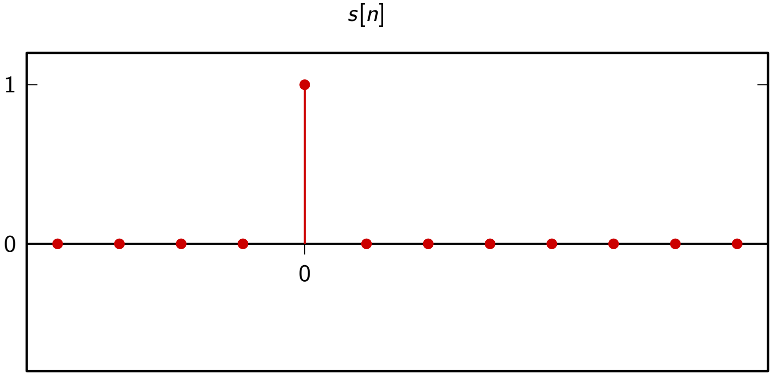
- channel introduces a delay of d seconds
- we can write $d = (L+\tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- L is called the *bulk delay*
- τ is the fractional delay

Delay compensation

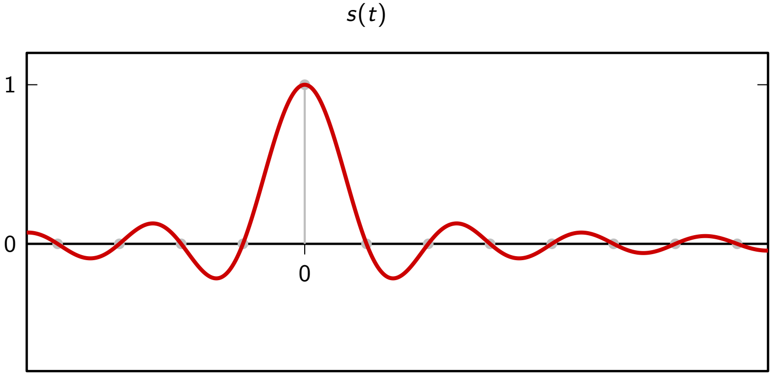
Assume the channel is a simple delay: $\hat{s}(t) = s(t - d) \Rightarrow D(f) = e^{-j2\pi fd}$

- channel introduces a delay of d seconds
- we can write $d = (L + \tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- L is called the *bulk delay*
- τ is the fractional delay

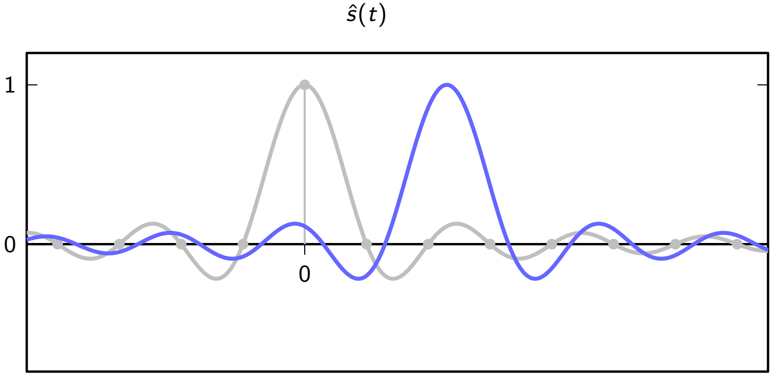
Estimating the bulk delay ($T_s = 1$)



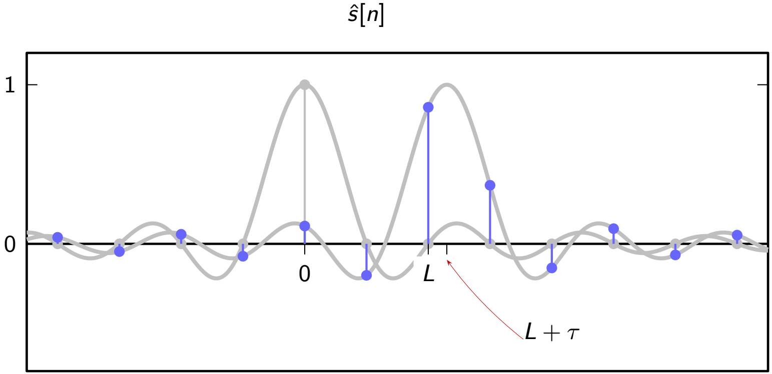
Estimating the bulk delay ($T_s = 1$)



Estimating the bulk delay ($T_s = 1$)



Estimating the bulk delay ($T_s = 1$)



Estimating the fractional delay

- transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n)$)
- receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$
- after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

- multiply by known frequency

$$\hat{b}[n]\, e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Estimating the fractional delay

- transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n))$
- receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$
- after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

- multiply by known frequency

$$\hat{b}[n]\, e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Estimating the fractional delay

- transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n))$
- receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$
- after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

- multiply by known frequency

$$\hat{b}[n]\, e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Estimating the fractional delay

- transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n)$)
- receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$
- after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

- multiply by known frequency

$$\hat{b}[n]\, e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Compensating for the fractional delay

- $\hat{s}[n] = s((n - \tau)T_s)$ (after offsetting bulk delay)
- we need to compute subsample values
- in theory, compensate with a sinc fractional delay $h[n] = \text{sinc}(n + \tau)$
- in practice, use local Lagrange approximation

$$x_L(n; t) = \sum_{k=-N}^{N} x[n+k] L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^{N} \frac{t - i}{k - i} \qquad\qquad k = -N, \ldots, N$$

Compensating for the fractional delay

- $\hat{s}[n] = s((n - \tau)T_s)$ (after offsetting bulk delay)
- we need to compute subsample values
- in theory, compensate with a sinc fractional delay $h[n] = \mathrm{sinc}(n + \tau)$
- in practice, use local Lagrange approximation

$$x_L(n; t) = \sum_{k=-N}^{N} x[n+k] L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^{N} \frac{t-i}{k-i} \qquad\qquad k = -N, \ldots, N$$

Compensating for the fractional delay

- $\hat{s}[n] = s((n-\tau)T_s)$ (after offsetting bulk delay)
- we need to compute subsample values
- in theory, compensate with a sinc fractional delay $h[n] = \mathrm{sinc}(n+\tau)$
- in practice, use local Lagrange approximation

$$x_L(n;t) = \sum_{k=-N}^{N} x[n+k]L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i\neq n}}^{N} \frac{t-i}{k-i} \qquad\qquad k = -N, \ldots, N$$

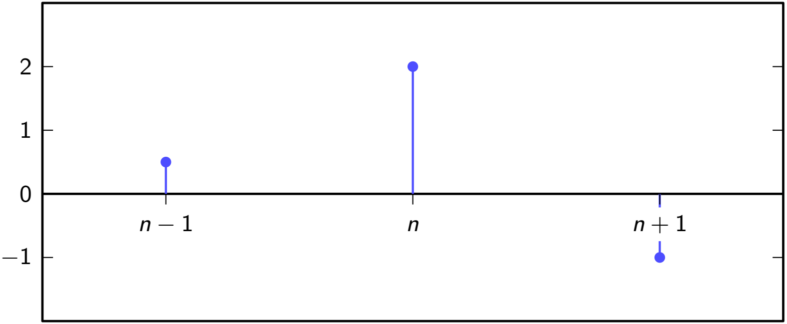
Compensating for the fractional delay

- $\hat{s}[n] = s((n - \tau)T_s)$ (after offsetting bulk delay)
- we need to compute subsample values
- in theory, compensate with a sinc fractional delay $h[n] = \text{sinc}(n + \tau)$
- in practice, use local Lagrange approximation

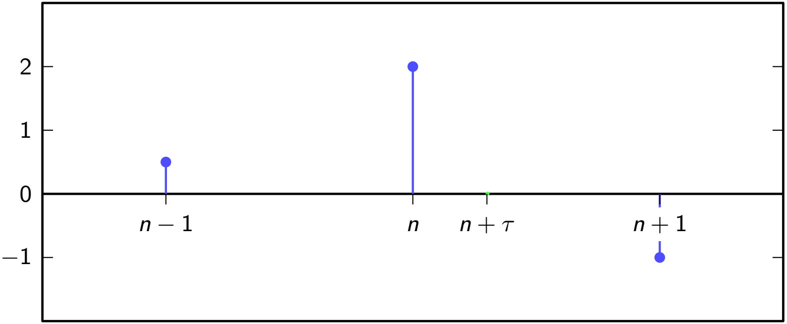
$$x_L(n; t) = \sum_{k=-N}^{N} x[n+k] L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^{N} \frac{t - i}{k - i} \qquad\qquad k = -N, \ldots, N$$

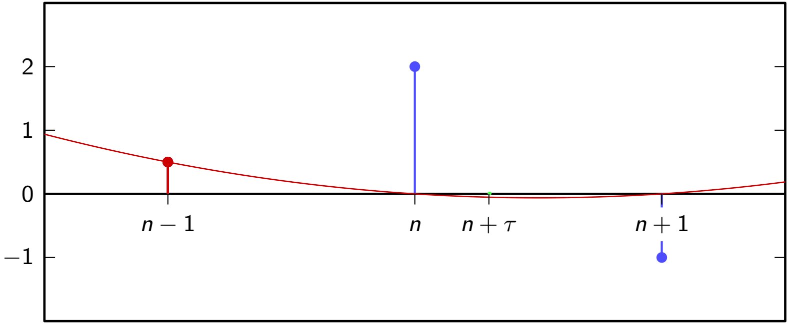
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



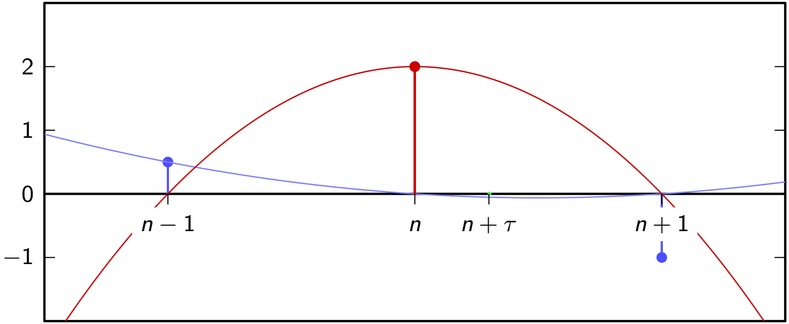
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



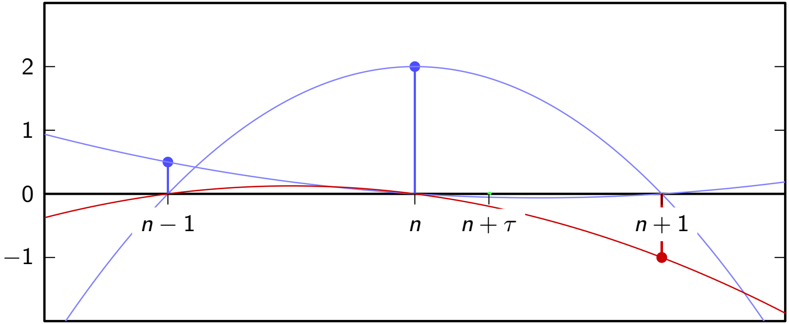
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



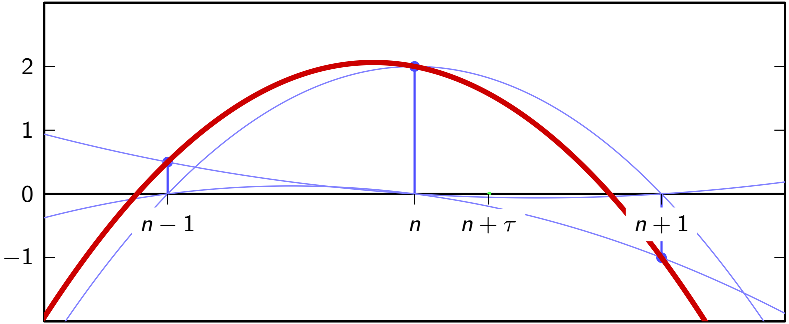
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



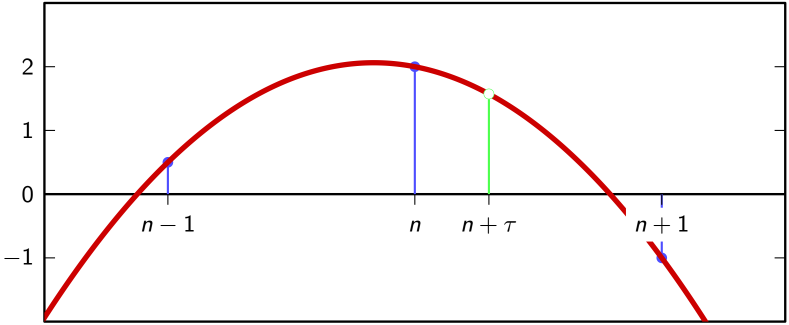
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



Lagrange interpolation as an FIR

- $x(n+\tau) \approx x_L(n;\tau)$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^{N} x[n-k] L_{-k}^{(N)}(\tau)$
- define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \ldots, N$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n-k] d_\tau[k]$
- $x_L(n;\tau) = (x * d_\tau)[n]$
- $d_\tau[k]$ is a $(2N+1)$-tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- $x(n+\tau) \approx x_L(n;\tau)$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^{N} x[n-k] L_{-k}^{(N)}(\tau)$
- define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \ldots, N$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n-k] d_\tau[k]$
- $x_L(n;\tau) = (x * d_\tau)[n]$
- $d_\tau[k]$ is a $(2N+1)$-tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- $x(n+\tau) \approx x_L(n;\tau)$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n+k]L_k^{(N)}(\tau) = \sum_{k=-N}^{N} x[n-k]L_{-k}^{(N)}(\tau)$
- define $d_\tau[k] = L_{-k}^{(N)}(\tau),\ k = -N, \ldots, N$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n-k]d_\tau[k]$
- $x_L(n;\tau) = (x * d_\tau)[n]$
- $d_\tau[k]$ is a $(2N+1)$-tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- $x(n+\tau) \approx x_L(n;\tau)$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n+k]L_k^{(N)}(\tau) = \sum_{k=-N}^{N} x[n-k]L_{-k}^{(N)}(\tau)$
- define $d_\tau[k] = L_{-k}^{(N)}(\tau),\ k = -N, \ldots, N$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n-k]d_\tau[k]$
- $x_L(n;\tau) = (x * d_\tau)[n]$
- $d_\tau[k]$ is a $(2N+1)$-tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- $x(n+\tau) \approx x_L(n;\tau)$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^{N} x[n-k] L_{-k}^{(N)}(\tau)$
- define $d_\tau[k] = L_{-k}^{(N)}(\tau),\ k = -N, \ldots, N$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n-k] d_\tau[k]$
- $x_L(n;\tau) = (x * d_\tau)[n]$
- $d_\tau[k]$ is a $(2N+1)$-tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- $x(n+\tau) \approx x_L(n;\tau)$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n+k]L_k^{(N)}(\tau) = \sum_{k=-N}^{N} x[n-k]L_{-k}^{(N)}(\tau)$
- define $d_\tau[k] = L_{-k}^{(N)}(\tau),\ k = -N, \ldots, N$
- $x_L(n;\tau) = \sum_{k=-N}^{N} x[n-k]d_\tau[k]$
- $x_L(n;\tau) = (x * d_\tau)[n]$
- $d_\tau[k]$ is a $(2N+1)$-tap FIR (dependent on τ)

Example ($N = 1$, second order approximation)

$$L_{-1}^{(1)}(t) = t\frac{t-1}{2}$$

$$L_{0}^{(1)}(t) = (1-t)(1+t)$$

$$L_{1}^{(1)}(t) = t\frac{t+1}{2}$$

Example ($N = 1$, second order approximation)

$$d_{0.2}[n] = \begin{cases} 0.12 & n = -1 \\ 0.96 & n = 0 \\ -0.08 & n = 1 \\ 0 & \text{otherwise} \end{cases}$$

Delay compensation algorithm

- estimate the delay τ
- compute the $2N + 1$ Lagrangian coefficients
- filter with the resulting FIR

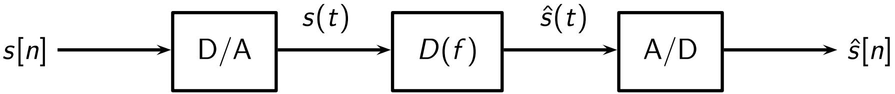
Delay compensation algorithm

- estimate the delay τ
- compute the $2N + 1$ Lagrangian coefficients
- filter with the resulting FIR

Delay compensation algorithm

- estimate the delay τ
- compute the $2N + 1$ Lagrangian coefficients
- filter with the resulting FIR

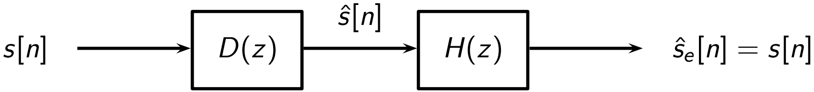
Compensating for the distortion



Compensating for the distortion

$$s[n] \longrightarrow \boxed{D(z)} \longrightarrow \hat{s}[n]$$

Example: adaptive equalization



Example: adaptive equalization

- in theory, $H(z) = 1/D(z)$
- but we don't know $D(z)$ in advance
- $D(z)$ may change over time

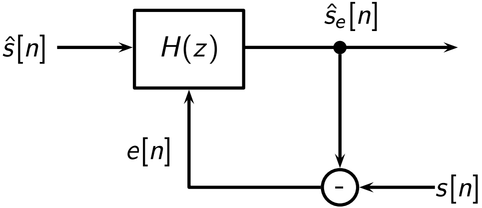
Example: adaptive equalization

- in theory, $H(z) = 1/D(z)$
- but we don't know $D(z)$ in advance
- $D(z)$ may change over time

Example: adaptive equalization

- in theory, $H(z) = 1/D(z)$
- but we don't know $D(z)$ in advance
- $D(z)$ may change over time

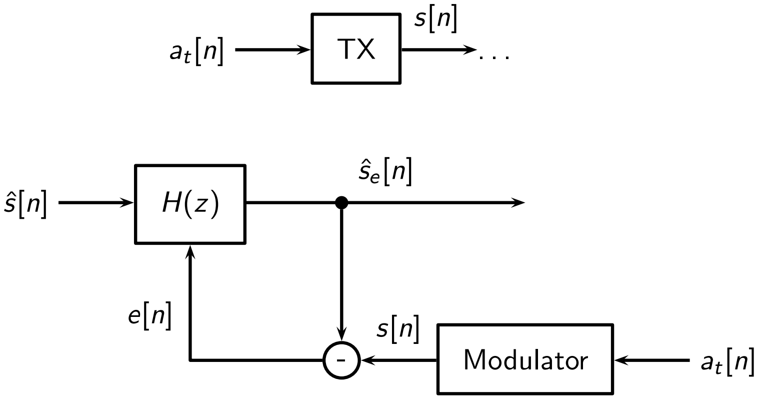
Adaptive equalization



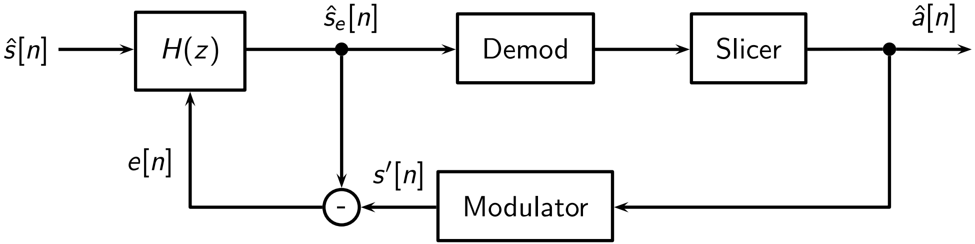
Adaptive equalization: bootstrapping via a training sequence

$$a_t[n] \longrightarrow \boxed{\text{TX}} \xrightarrow{s[n]} \ldots$$

Adaptive equalization: bootstrapping via a training sequence



Adaptive equalization: online mode



Adaptive equalization: the LMS algorithm

FIR equalization:

$$\hat{s}_e[n] = \sum_{k=0}^{N-1} h_n[k]\hat{s}[n-k]$$

$$e[n] = \hat{s}_e[n] - s[n]$$

adapting the coefficients:

$$h_{n+1}[k] = h_n[k] + \alpha e[n]x[n-k], \qquad k = 0, 1, \ldots, N-1$$

So much more to do...

- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

advanced topics in communication system design

ADSL

Overview:

- Channel
- Signaling strategy
- Discrete Multitone Modulation (DMT)

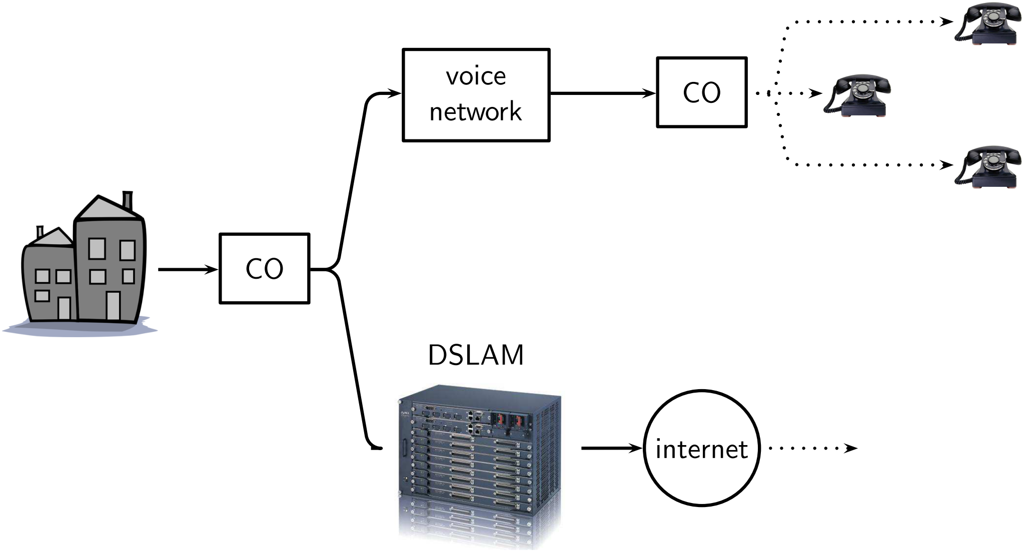
Overview:

- Channel
- Signaling strategy
- Discrete Multitone Modulation (DMT)

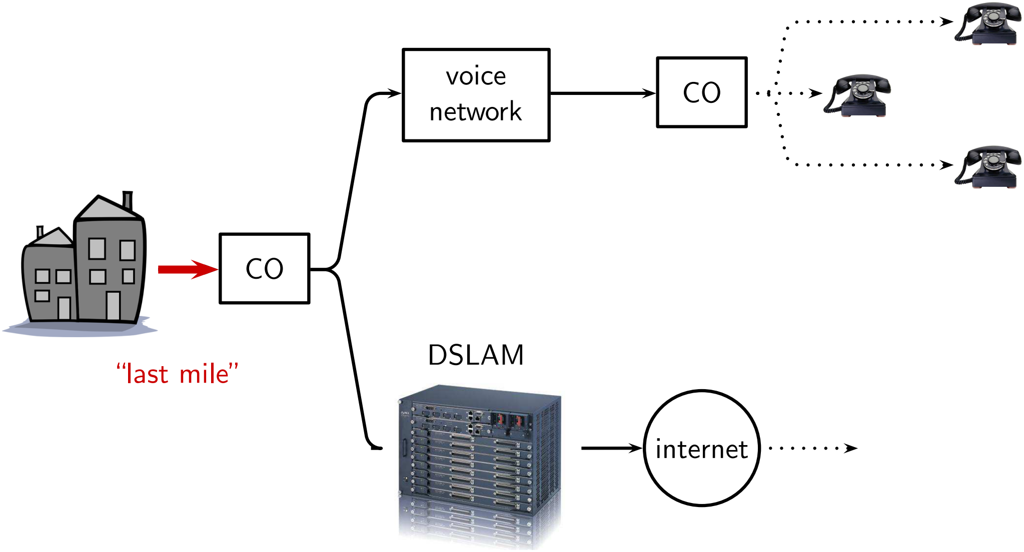
Overview:

- Channel
- Signaling strategy
- Discrete Multitone Modulation (DMT)

The telephone network today



The telephone network today



The last mile

- copper wire (twisted pair) between home and nearest CO
- very large bandwidth (well over 1MHz)
- very uneven spectrum: noise, attenuation, interference, etc.

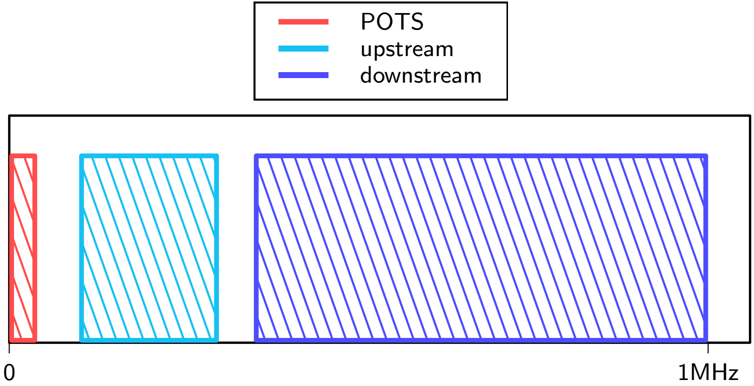
The last mile

- copper wire (twisted pair) between home and nearest CO
- very large bandwidth (well over 1MHz)
- very uneven spectrum: noise, attenuation, interference, etc.

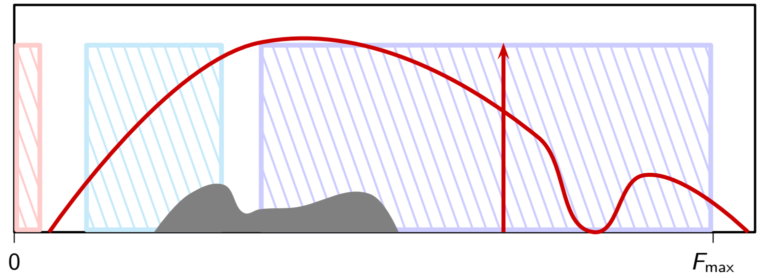
The last mile

- copper wire (twisted pair) between home and nearest CO
- very large bandwidth (well over 1MHz)
- very uneven spectrum: noise, attenuation, interference, etc.

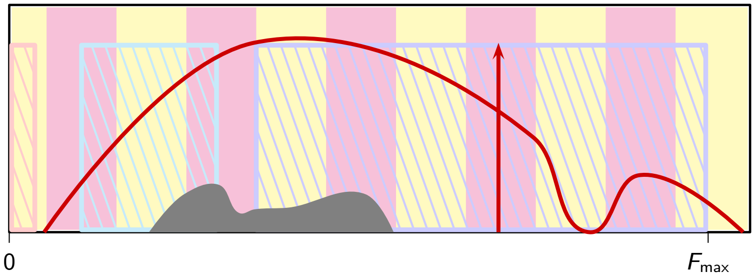
The ADSL channel



The ADSL channel



Idea: split the band into independent subchannels



Subchannel structure

- allocate N subchannels over the total positive bandwidth
- equal subchannel bandwidth $W = F_{\text{max}}/N$
- equally spaced subchannels with center frequency kF_{max}/N, $k = 0, \ldots, N-1$

Subchannel structure

- allocate N subchannels over the total positive bandwidth
- equal subchannel bandwidth $W = F_{\max}/N$
- equally spaced subchannels with center frequency $kF_{\max}/N$, $k = 0, \ldots, N-1$

Subchannel structure

- allocate N subchannels over the total positive bandwidth
- equal subchannel bandwidth $W = F_{\max}/N$
- equally spaced subchannels with center frequency $kF_{\max}/N$, $k = 0, \ldots, N-1$

The digital design

- pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- center frequency for each subchannel $\omega_k = 2\pi \dfrac{kW}{F_s} = \dfrac{2\pi}{2N} k$
- bandwidth of each subchannel $\dfrac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K = 2N$

The digital design

- pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- center frequency for each subchannel $\omega_k = 2\pi \dfrac{kW}{F_s} = \dfrac{2\pi}{2N}k$
- bandwidth of each subchannel $\dfrac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K = 2N$

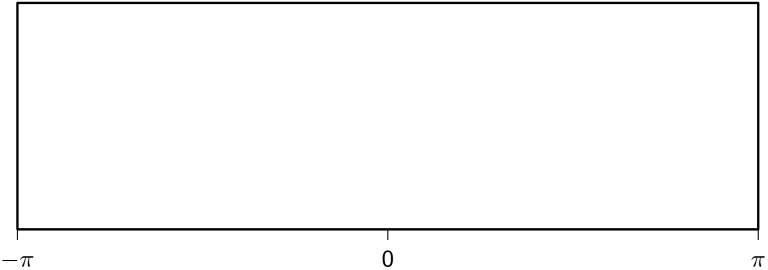
The digital design

- pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- center frequency for each subchannel $\omega_k = 2\pi \frac{kW}{F_s} = \frac{2\pi}{2N}k$
- bandwidth of each subchannel $\frac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K = 2N$

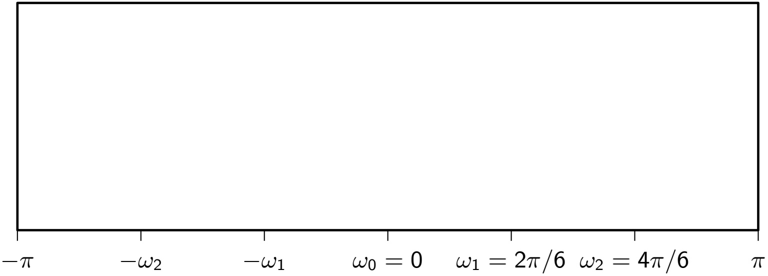
The digital design

- pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- center frequency for each subchannel $\omega_k = 2\pi\dfrac{kW}{F_s} = \dfrac{2\pi}{2N}k$
- bandwidth of each subchannel $\dfrac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K = 2N$

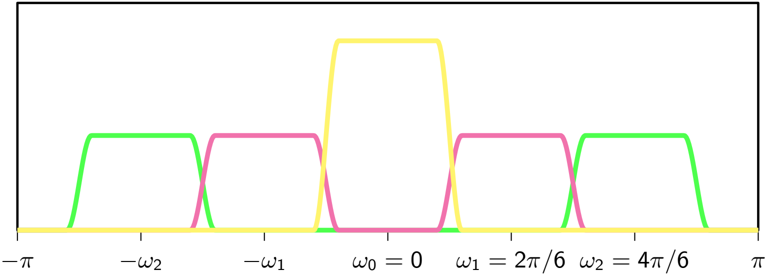
The digital design ($N = 3$)



The digital design ($N = 3$)



The digital design ($N = 3$)



The digital design

- put a QAM modem on each channel
- decide on constellation size independently
- noisy or forbidden subchannels send zeros

The digital design

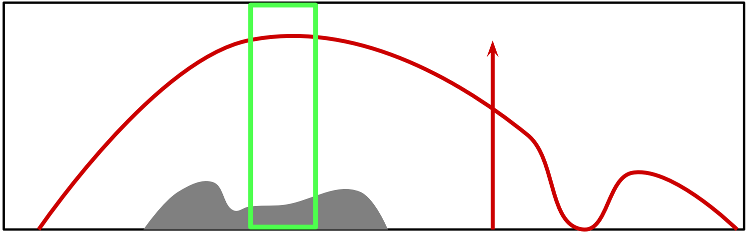
- put a QAM modem on each channel
- decide on constellation size independently
- noisy or forbidden subchannels send zeros

The digital design

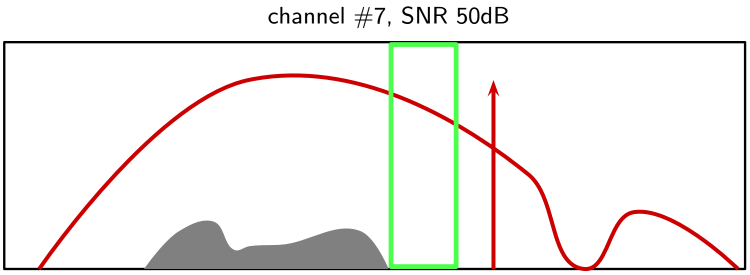
- put a QAM modem on each channel
- decide on constellation size independently
- noisy or forbidden subchannels send zeros

Idea: split the band into independent subchannels

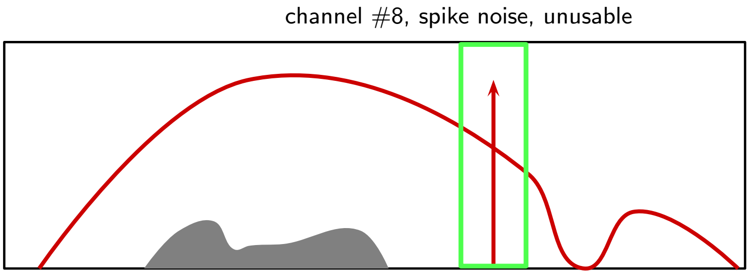
channel #5, SNR 20dB



Idea: split the band into independent subchannels

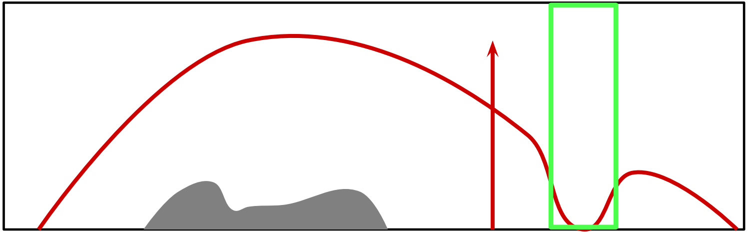


Idea: split the band into independent subchannels

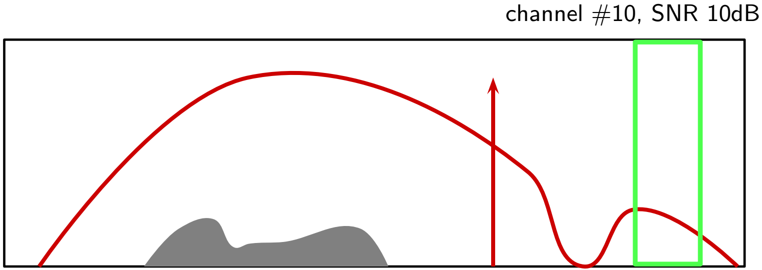


Idea: split the band into independent subchannels

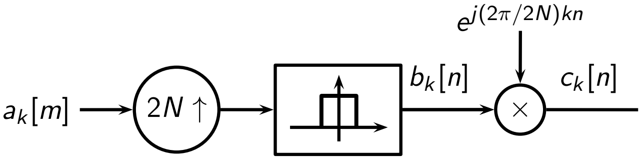
channel #9, too much attenuation, unusable



Idea: split the band into independent subchannels



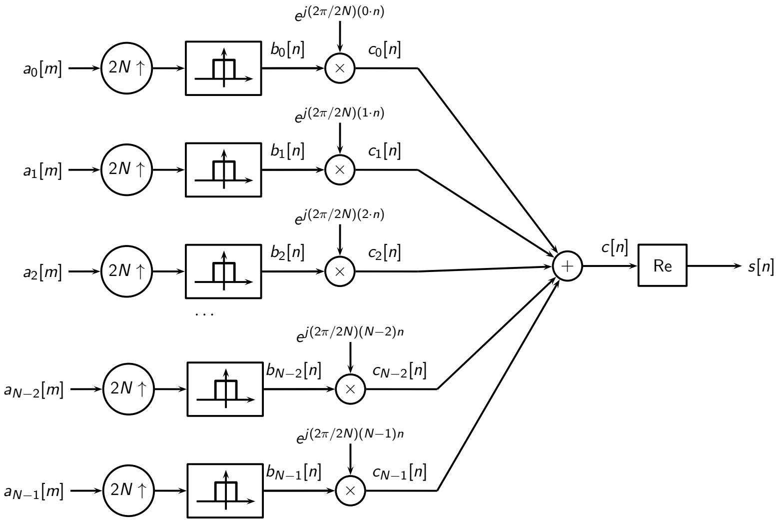
The subchannel modem



rate: W symbols/sec

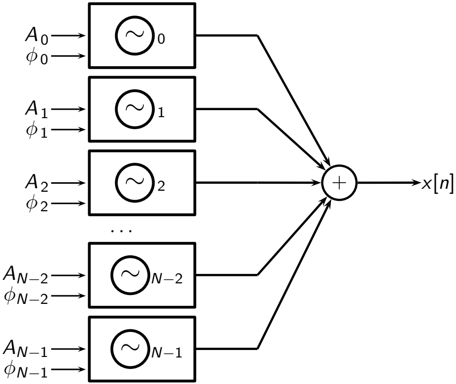
$2NW = F_s$ samples/sec

The bank of modems



If it looks familiar...

remember the DFT reconstruction formula?



DMT via IFFT

- we will show that transmission can be implemented efficiently via an IFFT
- Discrete Multitone Modulation

DMT via IFFT

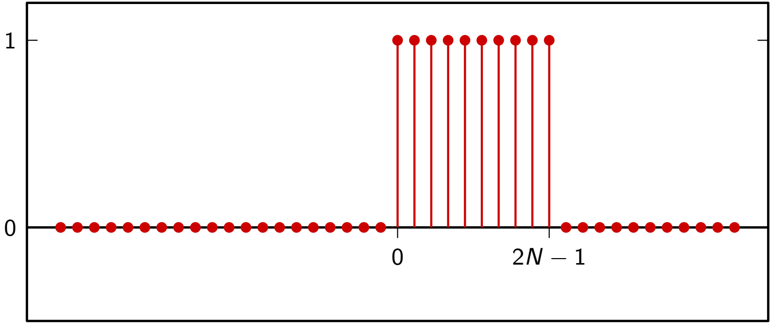
- we will show that transmission can be implemented efficiently via an IFFT
- Discrete Multitone Modulation

The great ADSL trick

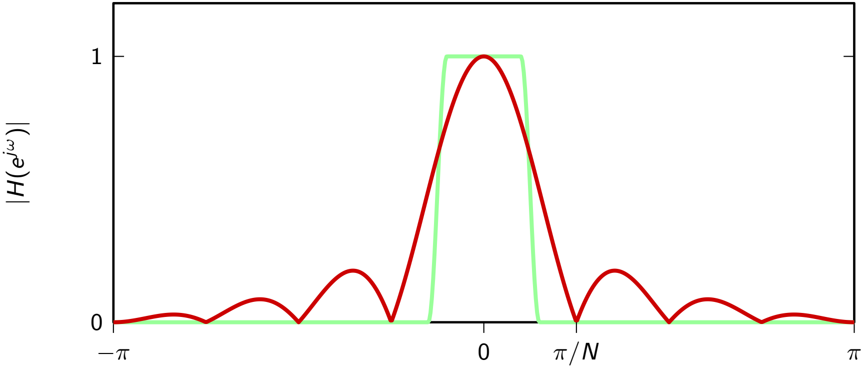
instead of using a "semi-ideal" digital interpolator (e.g. raised cosine), use a zero-order hold:

$$h[n] = \begin{cases} 1 & \text{for } 0 \leq n < 2N \\ 0 & \text{otherwise} \end{cases}$$

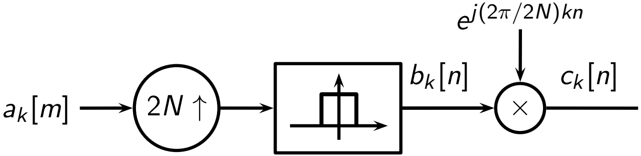
Digital ZOH: interval indicator signal



DTFT of interval signal



Back to the subchannel modem



rate: W symbols/sec

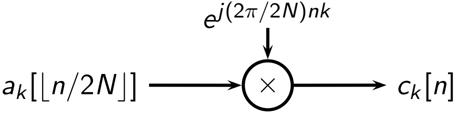
m changes every $1/W$ sec

$2NW = F_s$ samples/sec

n changes every $1/F_s = (1/W)/(2N)$

Back to the subchannel modem

by using the ZOH interpolator:

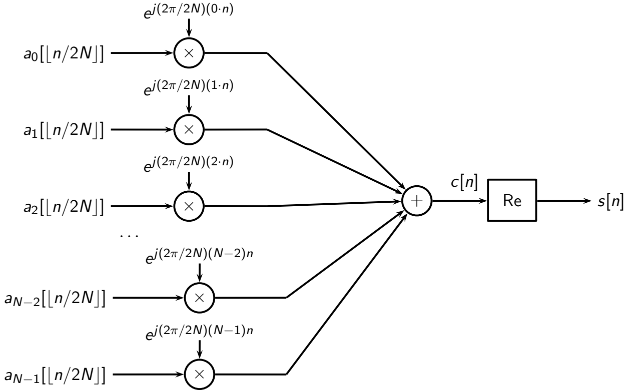


rate: W symbols/sec

$2NW = F_s$ samples/sec

n changes every $1/F_s$

The bank of modems, revisited



rate: $NW = F_{\max}$ symbols/sec

$2NW = F_s$ samples/sec

The complex output signal

$$c[n] = \sum_{k=0}^{N-1} a_k[\lfloor n/2N \rfloor] e^{j\frac{2\pi}{2N}nk}$$

for every successive block of $2N$ output samples:

- $a_k[\lfloor n/2N \rfloor] = a_k[p]$ doesn't change
- $c[n] = 2N \cdot \mathrm{IDFT}_{2N}\left\{\left[a_0[p] \quad a_1[p] \quad \ldots \quad a_{N-1}[p] \quad 0 \quad 0 \quad \ldots \quad 0\right]\right\}[n]$

The complex output signal

$$c[n] = \sum_{k=0}^{N-1} a_k[\lfloor n/2N \rfloor] e^{j\frac{2\pi}{2N}nk}$$

for every successive block of $2N$ output samples:

- $a_k[\lfloor n/2N \rfloor] = a_k[p]$ doesn't change
- $c[n] = 2N \cdot \mathrm{IDFT}_{2N}\left\{\begin{bmatrix} a_0[p] & a_1[p] & \ldots & a_{N-1}[p] & 0 & 0 & \ldots & 0 \end{bmatrix}\right\}[n]$

The complex output signal: vector space approach

for every successive block of $2N$ output samples:

$$\mathbf{c} = \sum_{k=0}^{N-1} a_k \, \mathbf{w}^{(k)}$$

- $\mathbf{c}, \mathbf{w}^{(k)} \in \mathbb{C}^{2N}$
- $\mathbf{w}^{(k)}$ Fourier basis vector for $\mathbb{C}^{2N}$
- orthogonality of basis vector makes detection easy: $a_k = \langle \mathbf{w}^{(k)}, \mathbf{c} \rangle / (2N)$
- can obtain all a_k's with a single DFT

Neat implementation detail

- we are interested in $s[n] = \mathrm{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- it is easy to prove (exercise) that:

$$\mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_1 & x_2 & \ldots & x_{N-2} & x_{N-1} \end{bmatrix}\right\}^* = \mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_{N-1} & x_{N-2} & \ldots & x_2 & x_1 \end{bmatrix}^*\right\}$$

- $c[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & a_1[p] & \ldots & a_{N-1}[p] & 0 & 0 & \ldots & 0 \end{bmatrix}\right\}[n]$
- $c^*[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & 0 & 0 & \ldots & 0 & a^*_{N-1}[p] & \ldots & a^*_1[p] \end{bmatrix}\right\}[n]$
- therefore

$$s[n] = N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} 2a_0[p] & a_1[p] & \ldots & a_{N-1}[p] & a^*_{N-1}[p] & a^*_{N-2}[p] & \ldots & a^*_1[p] \end{bmatrix}\right\}[n]$$

Neat implementation detail

- we are interested in $s[n] = \mathrm{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- it is easy to prove (exercise) that:

$$\mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_1 & x_2 & \dots & x_{N-2} & x_{N-1} \end{bmatrix}\right\}^* = \mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_{N-1} & x_{N-2} & \dots & x_2 & x_1 \end{bmatrix}^*\right\}$$

- $c[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & a_1[p] & \dots & a_{N-1}[p] & 0 & 0 & \dots & 0 \end{bmatrix}\right\}[n]$
- $c^*[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & 0 & 0 & \dots & 0 & a^*_{N-1}[p] & \dots & a^*_1[p] \end{bmatrix}\right\}[n]$
- therefore

$$s[n] = N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} 2a_0[p] & a_1[p] & \dots & a_{N-1}[p] & a^*_{N-1}[p] & a^*_{N-2}[p] & \dots & a^*_1[p] \end{bmatrix}\right\}[n]$$

Neat implementation detail

- we are interested in $s[n] = \text{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- it is easy to prove (exercise) that:

$$\text{IDFT}\left\{\begin{bmatrix} x_0 & x_1 & x_2 & \ldots & x_{N-2} & x_{N-1} \end{bmatrix}\right\}^* = \text{IDFT}\left\{\begin{bmatrix} x_0 & x_{N-1} & x_{N-2} & \ldots & x_2 & x_1 \end{bmatrix}^*\right\}$$

- $c[n] = 2N \cdot \text{IDFT}\left\{\begin{bmatrix} a_0[p] & a_1[p] & \ldots & a_{N-1}[p] & 0 & 0 & \ldots & 0 \end{bmatrix}\right\}[n]$
- $c^*[n] = 2N \cdot \text{IDFT}\left\{\begin{bmatrix} a_0[p] & 0 & 0 & \ldots & 0 & a^*_{N-1}[p] & \ldots & a^*_1[p] \end{bmatrix}\right\}[n]$
- therefore

$$s[n] = N \cdot \text{IDFT}\left\{\begin{bmatrix} 2a_0[p] & a_1[p] & \ldots & a_{N-1}[p] & a^*_{N-1}[p] & a^*_{N-2}[p] & \ldots & a^*_1[p] \end{bmatrix}\right\}[n]$$

Neat implementation detail

- we are interested in $s[n] = \mathrm{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- it is easy to prove (exercise) that:

$$\mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_1 & x_2 & \ldots & x_{N-2} & x_{N-1}\end{bmatrix}\right\}^* = \mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_{N-1} & x_{N-2} & \ldots & x_2 & x_1\end{bmatrix}^*\right\}$$

- $c[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & a_1[p] & \ldots & a_{N-1}[p] & 0 & 0 & \ldots & 0\end{bmatrix}\right\}[n]$
- $c^*[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & 0 & 0 & \ldots & 0 & a^*_{N-1}[p] & \ldots & a^*_1[p]\end{bmatrix}\right\}[n]$
- therefore

$$s[n] = N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} 2a_0[p] & a_1[p] & \ldots & a_{N-1}[p] & a^*_{N-1}[p] & a^*_{N-2}[p] & \ldots & a^*_1[p]\end{bmatrix}\right\}[n]$$

Neat implementation detail

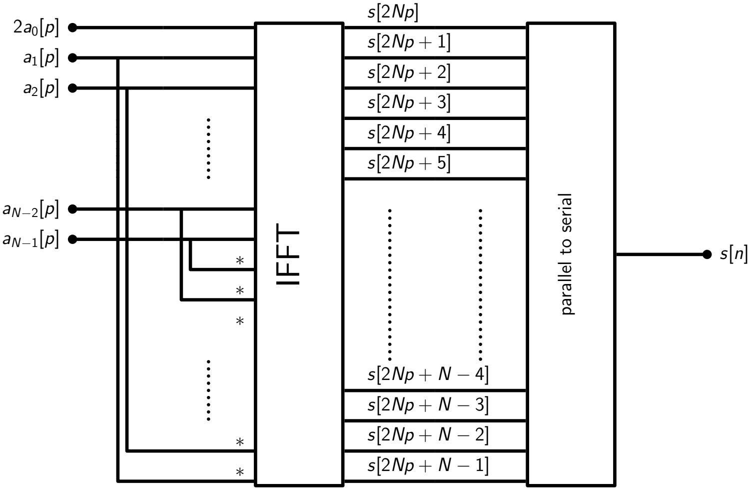
- we are interested in $s[n] = \mathrm{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- it is easy to prove (exercise) that:

$$\mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_1 & x_2 & \dots & x_{N-2} & x_{N-1} \end{bmatrix}\right\}^* = \mathrm{IDFT}\left\{\begin{bmatrix} x_0 & x_{N-1} & x_{N-2} & \dots & x_2 & x_1 \end{bmatrix}^*\right\}$$

- $c[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & a_1[p] & \dots & a_{N-1}[p] & 0 & 0 & \dots & 0 \end{bmatrix}\right\}[n]$
- $c^*[n] = 2N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} a_0[p] & 0 & 0 & \dots & 0 & a^*_{N-1}[p] & \dots & a^*_1[p] \end{bmatrix}\right\}[n]$
- therefore

$$s[n] = N \cdot \mathrm{IDFT}\left\{\begin{bmatrix} 2a_0[p] & a_1[p] & \dots & a_{N-1}[p] & a^*_{N-1}[p] & a^*_{N-2}[p] & \dots & a^*_1[p] \end{bmatrix}\right\}[n]$$

ADSL transmitter



rate: NW symbols/sec

$2NW = F_s$ samples/sec

ADSL specs

- $F_{\max} = 1104$ KHz
- $N = 256$
- subchannel width $W = 4312.5$ Hz
- each QAM can send from 0 to 15 bits per symbol
- forbidden channels: 0 to 7 (voice)
- channels 7 to 31: upstream data
- max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- $F_{\text{max}} = 1104$ KHz
- $N = 256$
- subchannel width $W = 4312.5$ Hz
- each QAM can send from 0 to 15 bits per symbol
- forbidden channels: 0 to 7 (voice)
- channels 7 to 31: upstream data
- max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- $F_{\max} = 1104$ KHz
- $N = 256$
- subchannel width $W = 4312.5$ Hz
- each QAM can send from 0 to 15 bits per symbol
- forbidden channels: 0 to 7 (voice)
- channels 7 to 31: upstream data
- max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- $F_{\max} = 1104$ KHz
- $N = 256$
- subchannel width $W = 4312.5$ Hz
- each QAM can send from 0 to 15 bits per symbol
- forbidden channels: 0 to 7 (voice)
- channels 7 to 31: upstream data
- max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- $F_{\max} = 1104$ KHz
- $N = 256$
- subchannel width $W = 4312.5$ Hz
- each QAM can send from 0 to 15 bits per symbol
- forbidden channels: 0 to 7 (voice)
- channels 7 to 31: upstream data
- max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- $F_{\max} = 1104$ KHz
- $N = 256$
- subchannel width $W = 4312.5$ Hz
- each QAM can send from 0 to 15 bits per symbol
- forbidden channels: 0 to 7 (voice)
- channels 7 to 31: upstream data
- max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- $F_{\text{max}} = 1104$ KHz
- $N = 256$
- subchannel width $W = 4312.5$ Hz
- each QAM can send from 0 to 15 bits per symbol
- forbidden channels: 0 to 7 (voice)
- channels 7 to 31: upstream data
- max theoretical throughput: 14.9 Mbps (downstream)