Digital Signal Processing

Module 9: Digital Communication Systems

Module Overview:

- Module 9.1: The analog channel
- Module 9.2: Meeting the bandwidth constraint
- Module 9.3: Meeting the power constraint
- Module 9.4: Modulation and demodulation
- Module 9.5: Receiver design
- Module 9.6: ADSL

Digital Signal Processing

Module 9.1: Digital Communication Systems

Overview:

- The many incarnations of a signal
- Analog channel constraints
- Satisfying the constraints

Overview:

- The many incarnations of a signal
- Analog channel constraints
- Satisfying the constraints

Overview:

- The many incarnations of a signal
- Analog channel constraints
- Satisfying the constraints

Digital data throughputs

- Transatlantic cable:
  - 1866: 8 words per minute ($\approx$5 bps)
  - 1956: AT&T, coax, 48 voice channels ($\approx$3Mbps)
  - 2005: Alcatel Tera10, fiber, 8.4 Tbps (8.4×10^{12} bps)
  - 2012: fiber, 60 Tbps
- Voiceband modems
  - 1950s: Bell 202, 1200 bps
  - 1990s: V90, 56Kbps
  - 2008: ADSL2+, 24Mbps

Digital data throughputs

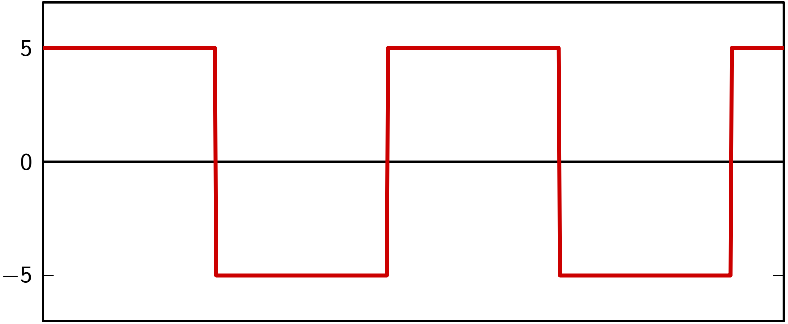
- Transatlantic cable:
  - 1866: 8 words per minute ($\approx$5 bps)
  - 1956: AT&T, coax, 48 voice channels ($\approx$3Mbps)
  - 2005: Alcatel Tera10, fiber, 8.4 Tbps (8.4×10^{12} bps)
  - 2012: fiber, 60 Tbps
- Voiceband modems
  - 1950s: Bell 202, 1200 bps
  - 1990s: V90, 56Kbps
  - 2008: ADSL2+, 24Mbps

Success factors for digital communications

1) power of the DSP paradigm:

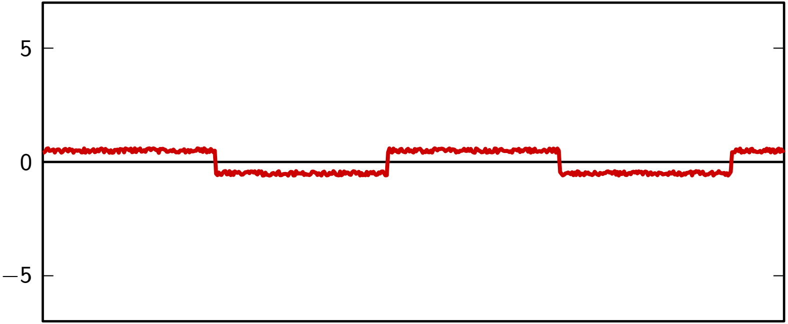
- integers are "easy" to regenerate
- good phase control
- adaptive algorithms

Regenerating signals



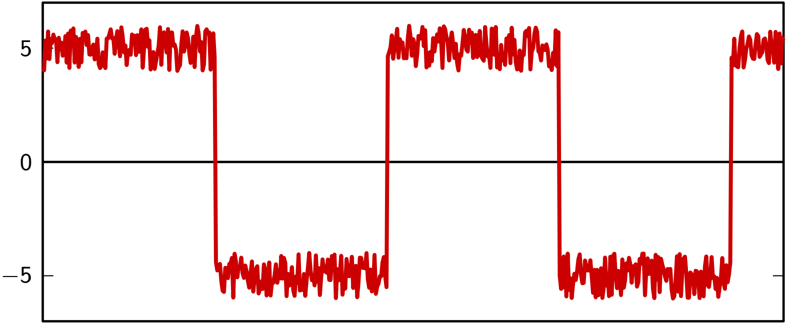
$x(t)$

Regenerating signals



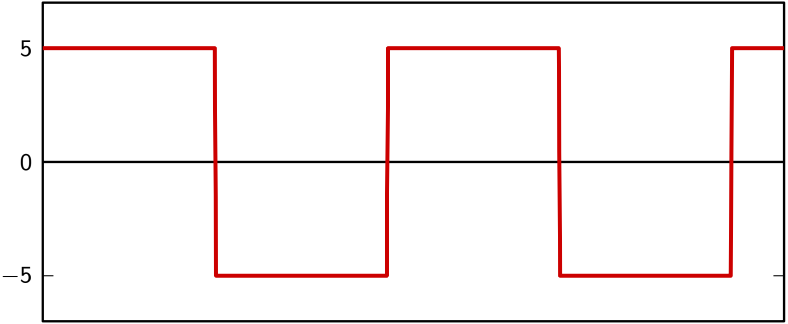
$$x(t)/G + \sigma(t)$$

Regenerating signals



$$G[x(t)/G + \sigma(t)] = x(t) + G\sigma(t)$$

Regenerating signals



$$\hat{x}_1(t) = G\,\mathrm{sgn}[x(t) + \sigma(t)]$$

Success factors for digital communications

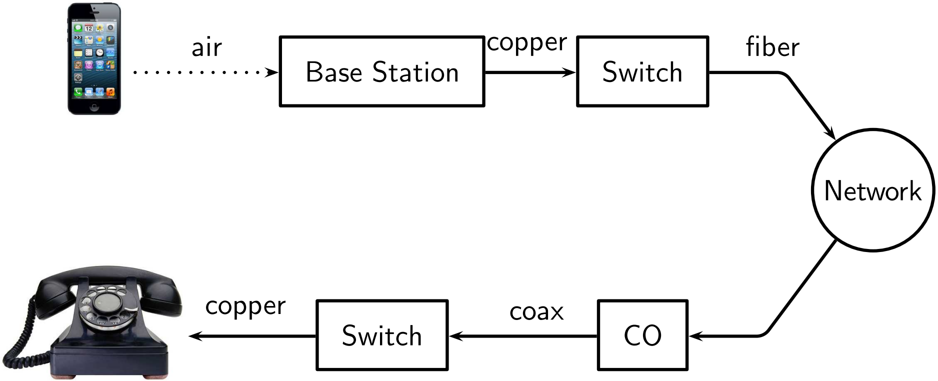
2) algorithmic nature of DSP is a perfect match with information theory:

- JPEG's entropy coding
- CD's and DVD's error correction
- trellis-coded modulation and Viterbi decoding

3) hardware advancement

- miniaturization
- general-purpose platforms
- power efficiency

The many incarnations of a conversation



The analog channel

unescapable "limits" of physical channels:

- bandwidth constraint
- power constraint

both constraints will affect the final *capacity* of the channel

The analog channel

unescapable "limits" of physical channels:

- bandwidth constraint
- power constraint

both constraints will affect the final *capacity* of the channel

The analog channel

unescapable “limits” of physical channels:

- bandwidth constraint
- power constraint

both constraints will affect the final *capacity* of the channel

The analog channel's capacity

maximum amount of information that can be reliably delivered over a channel
(bits per second)

Bandwidth vs capacity

simple thought experiment:

- we want to transmit information encoded as a sequence of digital samples over a continuous-time channel
- we interpolate the sequence of samples with a period T_s
- if we make T_s small we can send more info per unit of time...
- ... but the bandwidth of the signal will grow as $1/T_s$

Bandwidth vs capacity

simple thought experiment:

- we want to transmit information encoded as a sequence of digital samples over a continuous-time channel
- we interpolate the sequence of samples with a period T_s
- if we make T_s small we can send more info per unit of time...
- ... but the bandwidth of the signal will grow as $1/T_s$

Bandwidth vs capacity

simple thought experiment:

- we want to transmit information encoded as a sequence of digital samples over a continuous-time channel
- we interpolate the sequence of samples with a period T_s
- if we make T_s small we can send more info per unit of time...
- ... but the bandwidth of the signal will grow as $1/T_s$

Bandwidth vs capacity

simple thought experiment:

- we want to transmit information encoded as a sequence of digital samples over a continuous-time channel
- we interpolate the sequence of samples with a period T_s
- if we make T_s small we can send more info per unit of time...
- ... but the bandwidth of the signal will grow as $1/T_s$

Power and capacity

another thought experiment:

- all channels introduce noise; at the receiver we have to “guess” what was transmitted
- suppose noise variance is 1
- suppose we are transmitting integers between 1 and 10: lots of guessing errors
- transmit only odd numbers: fewer errors but less information

Power and capacity

another thought experiment:

- all channels introduce noise; at the receiver we have to "guess" what was transmitted
- suppose noise variance is 1
- suppose we are transmitting integers between 1 and 10: lots of guessing errors
- transmit only odd numbers: fewer errors but less information

Power and capacity

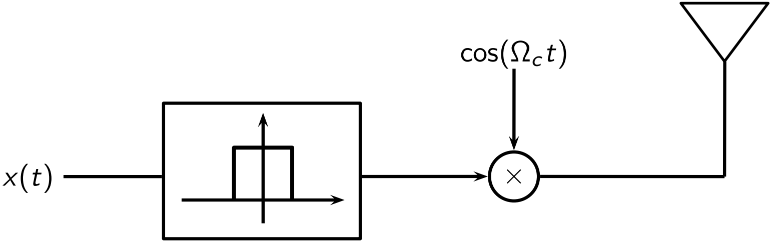
another thought experiment:

- all channels introduce noise; at the receiver we have to "guess" what was transmitted
- suppose noise variance is 1
- suppose we are transmitting integers between 1 and 10: lots of guessing errors
- transmit only odd numbers: fewer errors but less information

Power and capacity

another thought experiment:

- all channels introduce noise; at the receiver we have to "guess" what was transmitted
- suppose noise variance is 1
- suppose we are transmitting integers between 1 and 10: lots of guessing errors
- transmit only odd numbers: fewer errors but less information



$x(t)$
$\cos(\Omega_c t)$

Example: the AM radio channel

- from 530kHz to 1.7MHz
- each channel is 8KHz
- power limited by law:
  - daytime/nighttime
  - interference
  - health hazards

Example: the AM radio channel

- from 530kHz to 1.7MHz
- each channel is 8KHz
- power limited by law:
  - daytime/nighttime
  - interference
  - health hazards

Example: the AM radio channel

- from 530kHz to 1.7MHz
- each channel is 8KHz
- power limited by law:
  - daytime/nighttime
  - interference
  - health hazards

Example: the AM radio channel

- from 530kHz to 1.7MHz
- each channel is 8KHz
- power limited by law:
  - daytime/nighttime
  - interference
  - health hazards

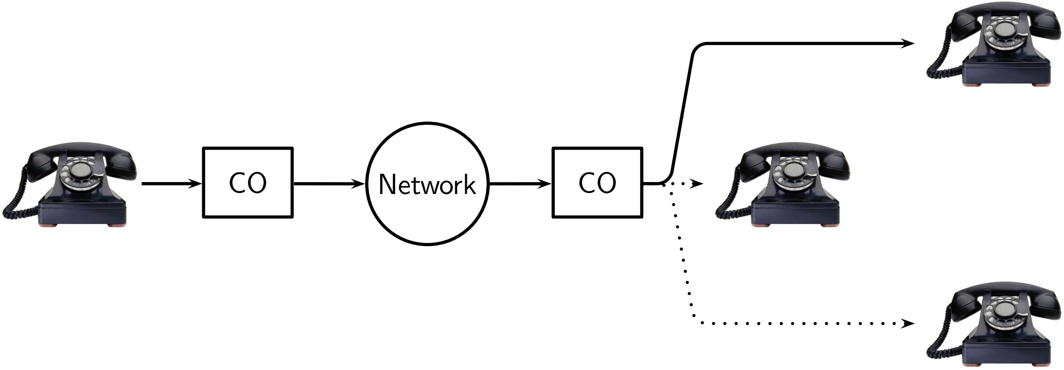
Example: the AM radio channel

- from 530kHz to 1.7MHz
- each channel is 8KHz
- power limited by law:
  - daytime/nighttime
  - interference
  - health hazards

Example: the AM radio channel

- from 530kHz to 1.7MHz
- each channel is 8KHz
- power limited by law:
  - daytime/nighttime
  - interference
  - health hazards

Example: the telephone channel



Example: the telephone channel

- one channel from around 300Hz to around 3000Hz
- power limited by law to 0.2-0.7V rms
- noise is rather low: SNR usually 30dB or more

Example: the telephone channel

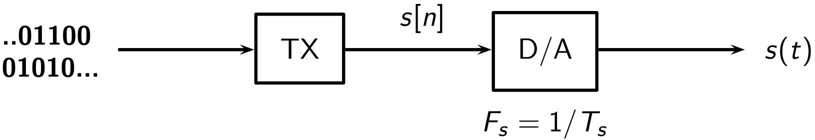
- one channel from around 300Hz to around 3000Hz
- power limited by law to 0.2-0.7V rms
- noise is rather low: SNR usually 30dB or more

Example: the telephone channel

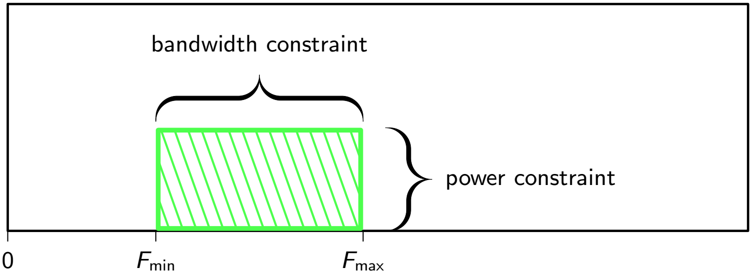
- one channel from around 300Hz to around 3000Hz
- power limited by law to 0.2-0.7V rms
- noise is rather low: SNR usually 30dB or more

The all-digital paradigm

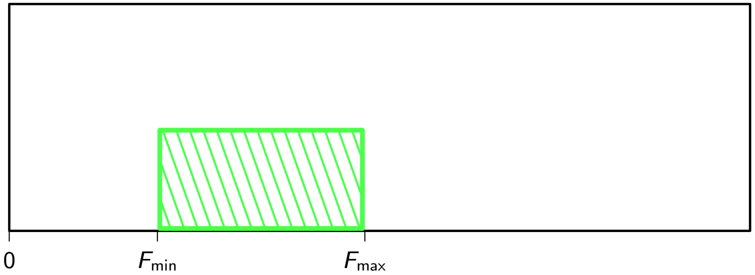
keep everything digital until we hit the physical channel



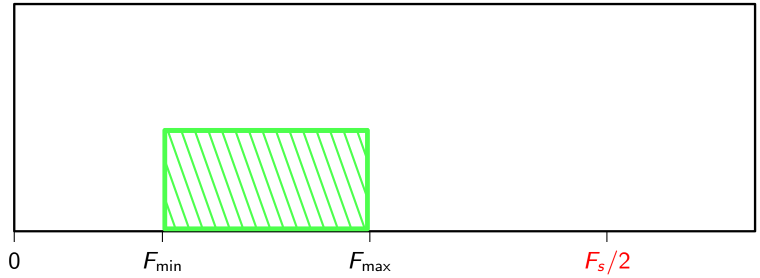
Let's look at the channel constraints



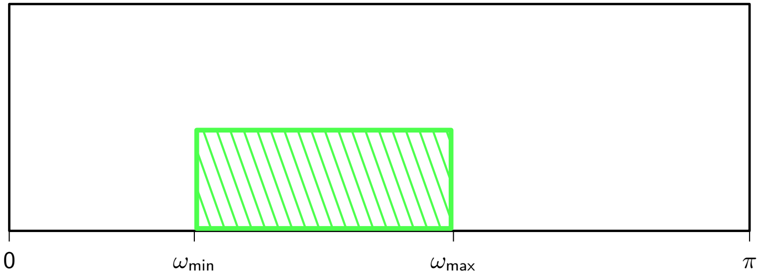
Converting the specs to a digital design



Converting the specs to a digital design



Converting the specs to a digital design



Transmitter design

some working hypotheses:

- convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- now we need to convert $a[n]$ into a continuous-time signal within the constraints

Transmitter design

some working hypotheses:

- convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- now we need to convert $a[n]$ into a continuous-time signal within the constraints

Transmitter design

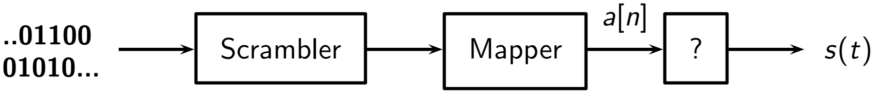
some working hypotheses:

- convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- now we need to convert $a[n]$ into a continuous-time signal within the constraints

Transmitter design

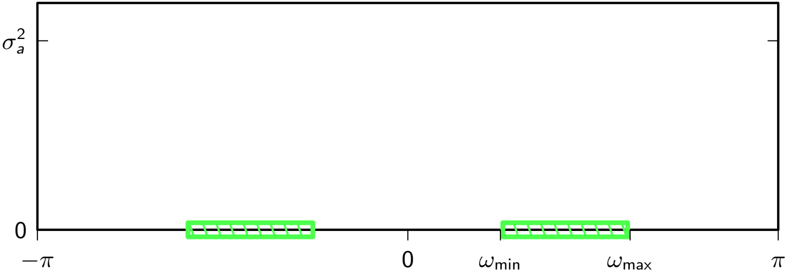
some working hypotheses:

- convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- now we need to convert $a[n]$ into a continuous-time signal within the constraints



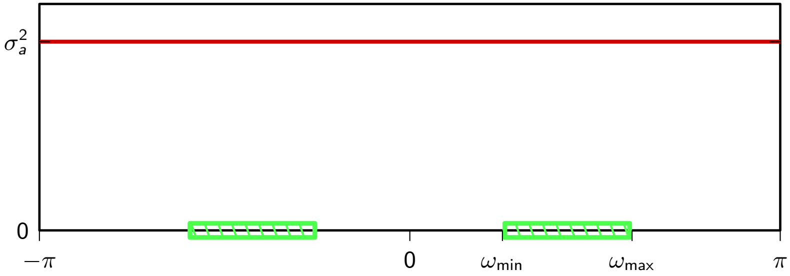
First problem: the bandwidth constraint

$$P_a(e^{j\omega}) = \sigma_a^2$$



First problem: the bandwidth constraint

$$P_a(e^{j\omega}) = \sigma_a^2$$



END OF MODULE 9.1

Digital Signal Processing

Module 9.2: Controlling the Bandwidth

Overview:

- Upsampling
- Fitting the transmitter's spectrum

Overview:

- Upsampling
- Fitting the transmitter's spectrum

Shaping the bandwidth

Our problem:

- bandwidth constraint requires us to control the spectral support of a signal
- we need to be able to "shrink" the support of a full-band signal
- the answer is *multirate* techniques

Shaping the bandwidth

Our problem:

- bandwidth constraint requires us to control the spectral support of a signal
- we need to be able to "shrink" the support of a full-band signal
- the answer is *multirate* techniques

Our problem:

- bandwidth constraint requires us to control the spectral support of a signal
- we need to be able to “shrink” the support of a full-band signal
- the answer is *multirate* techniques

Multirate signal processing

In a nutshell:

- increase or decrease the number of samples in a discrete-time signal
- equivalent to going to continuous time and resampling
- staying in the digital world is "cleaner"

Multirate signal processing

In a nutshell:

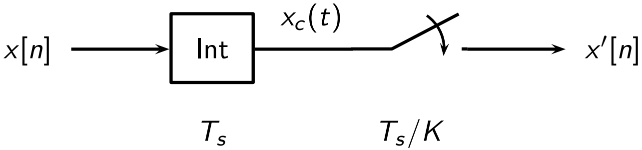
- increase or decrease the number of samples in a discrete-time signal
- equivalent to going to continuous time and resampling
- staying in the digital world is “cleaner”

Multirate signal processing

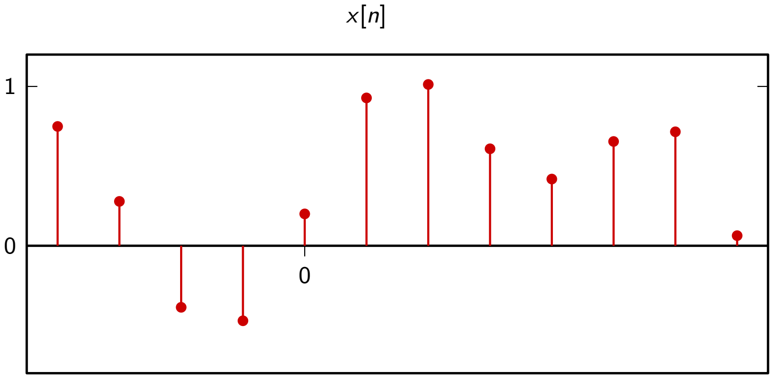
In a nutshell:

- increase or decrease the number of samples in a discrete-time signal
- equivalent to going to continuous time and resampling
- staying in the digital world is "cleaner"

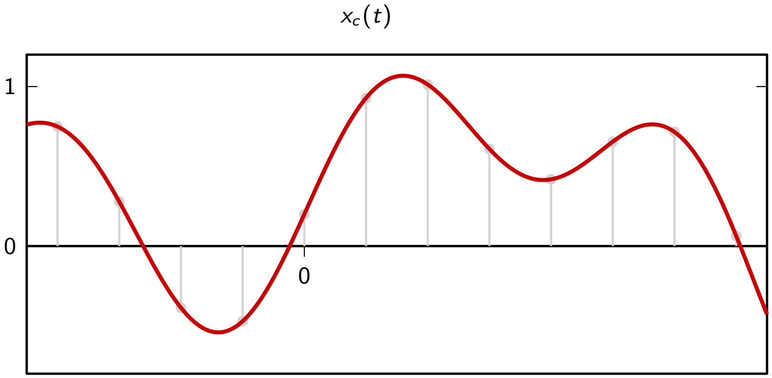
Upsampling via continuous time



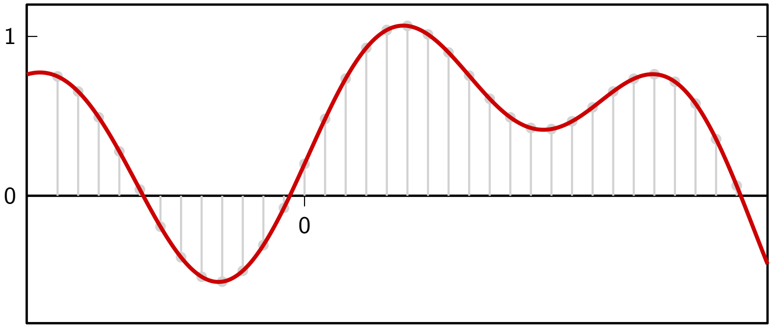
Upsampling ($K = 3$)



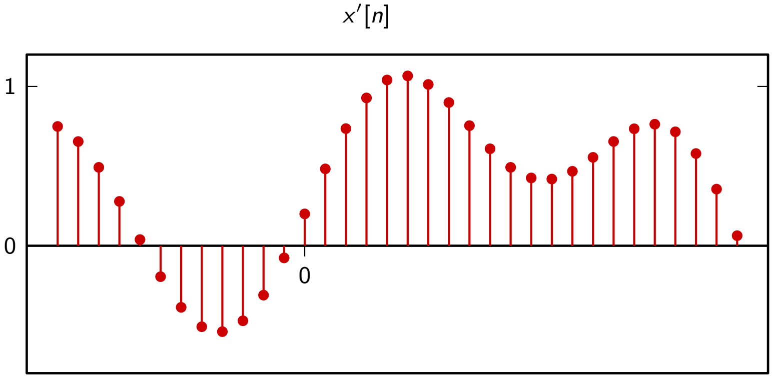
Upsampling ($K = 3$)



Upsampling ($K = 3$)



Upsampling ($K = 3$)



Upsampling

As per usual, we can choose $T_s = 1$...

$$x_c(t) = \sum_{m=-\infty}^{\infty} x[m]\,\mathrm{sinc}(t-m)$$

$$x'[n] = x_c(n/K)$$

$$= \sum_{m=-\infty}^{\infty} x[m]\,\mathrm{sinc}\left(\frac{n}{K} - m\right)$$

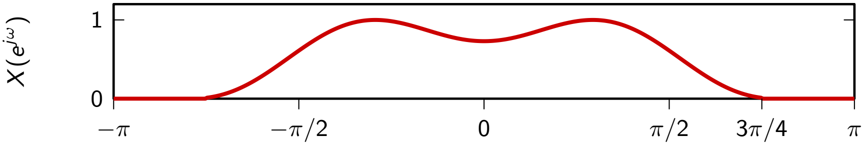
Upsampling

As per usual, we can choose $T_s = 1$...

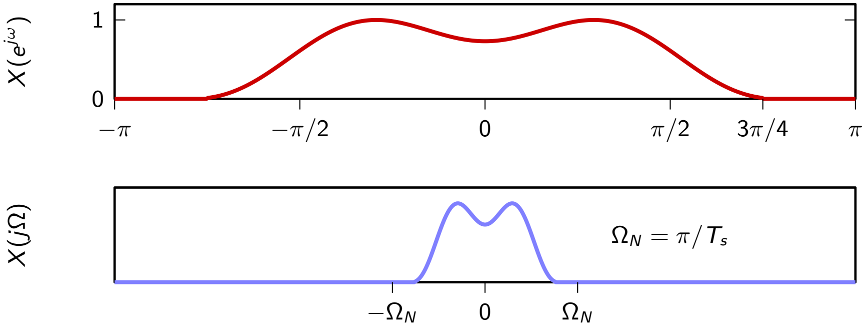
$$x_c(t) = \sum_{m=-\infty}^{\infty} x[m]\,\mathrm{sinc}(t-m)$$

$$\begin{aligned} x'[n] &= x_c(n/K) \\ &= \sum_{m=-\infty}^{\infty} x[m]\,\mathrm{sinc}\left(\frac{n}{K} - m\right) \end{aligned}$$

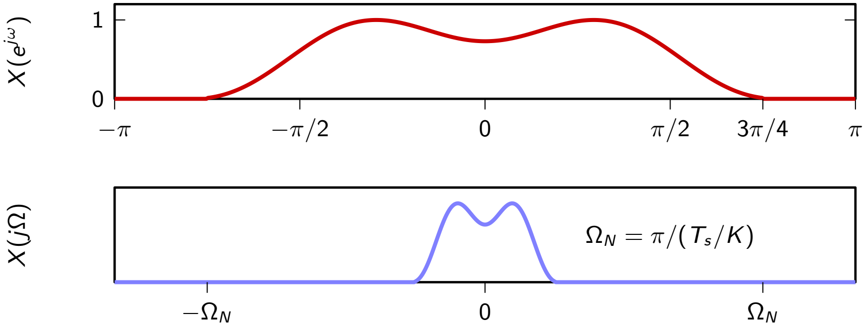
Upsampling (frequency domain)



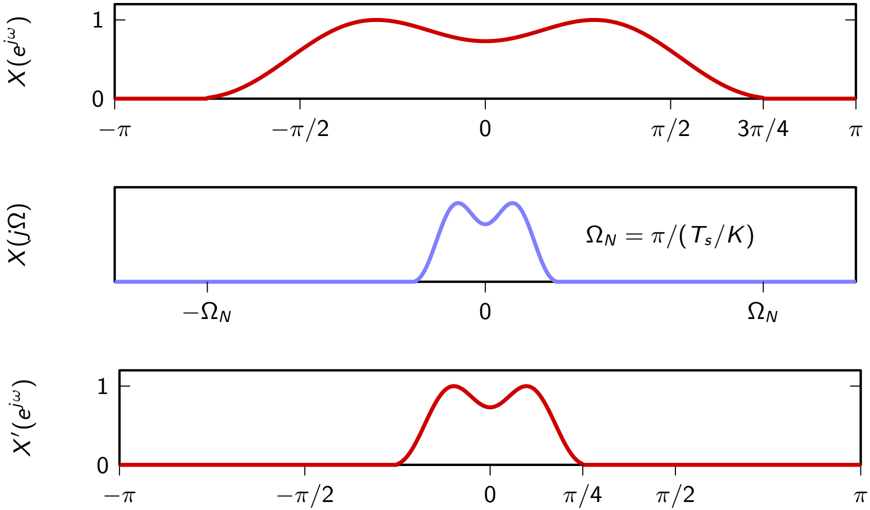
Upsampling (frequency domain)



Upsampling (frequency domain)



Upsampling (frequency domain)



Upsampling in the digital domain

what can we do purely digitally?

- we need to "increase" the number of samples by K
- obviously $x_U[m] = x[n]$ when m multiple of K
- for lack of a better strategy, put zeros elsewhere
- example for $K = 3$:

$$x_U[m] = \ldots x[0], 0, 0, x[1], 0, 0, x[2], 0, 0, \ldots$$

Upsampling in the digital domain

what can we do purely digitally?

- we need to "increase" the number of samples by K
- obviously $x_U[m] = x[n]$ when m multiple of K
- for lack of a better strategy, put zeros elsewhere
- example for $K = 3$:

$$x_U[m] = \ldots x[0], 0, 0, x[1], 0, 0, x[2], 0, 0, \ldots$$

Upsampling in the digital domain

what can we do purely digitally?

- we need to "increase" the number of samples by K
- obviously $x_U[m] = x[n]$ when m multiple of K
- for lack of a better strategy, put zeros elsewhere
- example for $K = 3$:

$$x_U[m] = \ldots x[0], 0, 0, x[1], 0, 0, x[2], 0, 0, \ldots$$

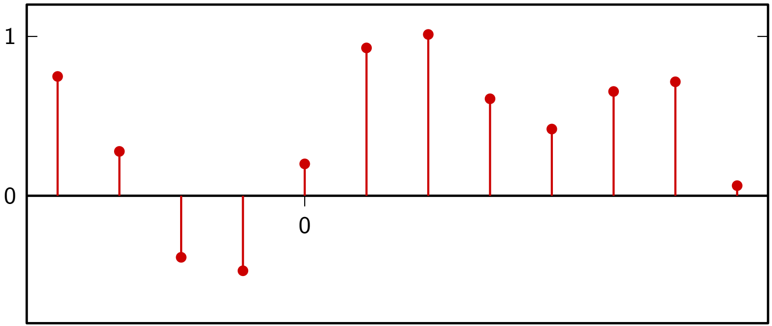
Upsampling in the digital domain

what can we do purely digitally?

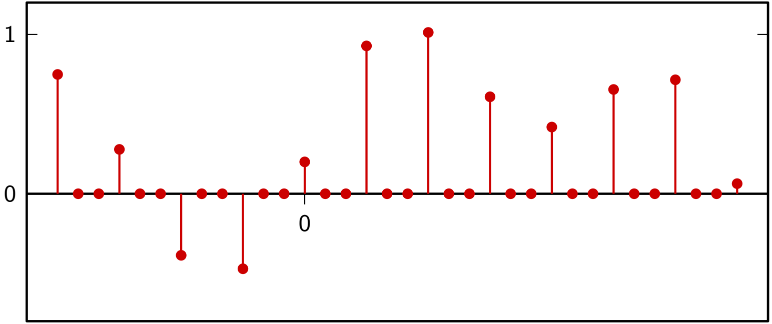
- we need to "increase" the number of samples by K
- obviously $x_U[m] = x[n]$ when m multiple of K
- for lack of a better strategy, put zeros elsewhere
- example for $K = 3$:

$$x_U[m] = \ldots x[0], 0, 0, x[1], 0, 0, x[2], 0, 0, \ldots$$

Upsampling in the Time Domain



Upsampling in the Time Domain



Upsampling in the digital domain

in the frequency domain

$$X_U(e^{j\omega}) = \sum_{m=-\infty}^{\infty} x_U[m]e^{-j\omega m}$$

$$= \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega nK}$$

$$= X(e^{j\omega K})$$

Upsampling in the digital domain

in the frequency domain

$$X_U(e^{j\omega}) = \sum_{m=-\infty}^{\infty} x_U[m]e^{-j\omega m}$$

$$= \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega nK}$$

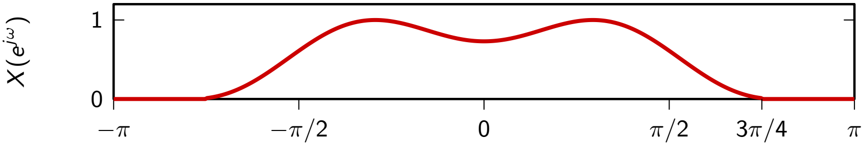
$$= X(e^{j\omega K})$$

Upsampling in the digital domain

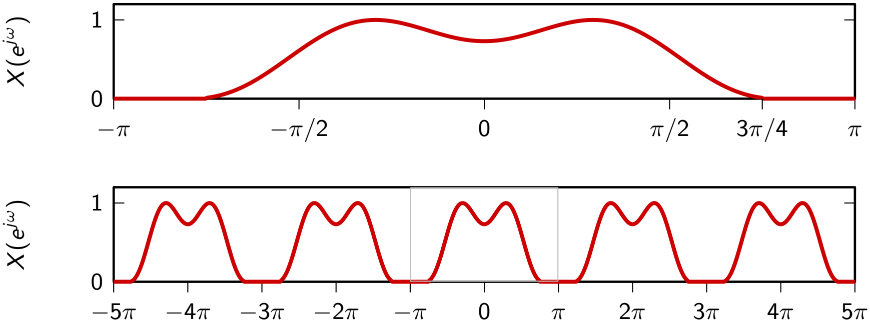
in the frequency domain

$$
\begin{aligned}
X_U(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} x_U[m]e^{-j\omega m} \\
&= \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega nK} \\
&= X(e^{j\omega K})
\end{aligned}
$$

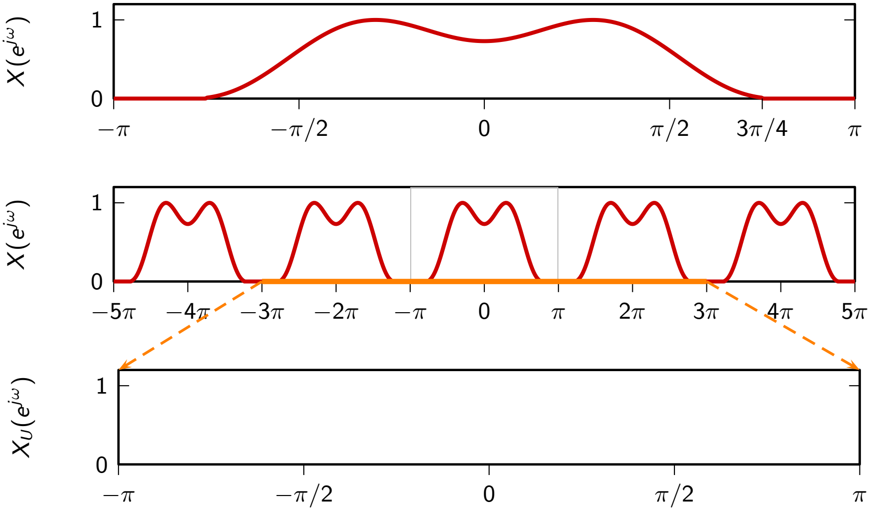
Upsampling in the digital domain



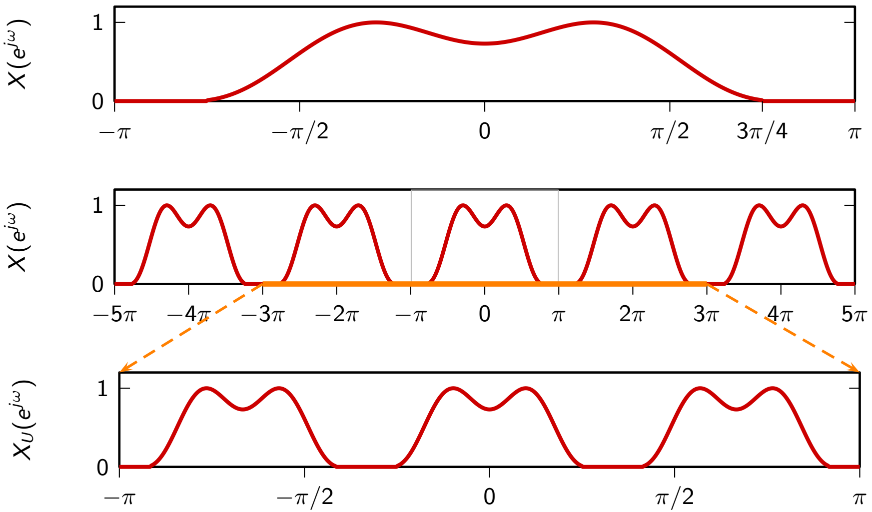
Upsampling in the digital domain



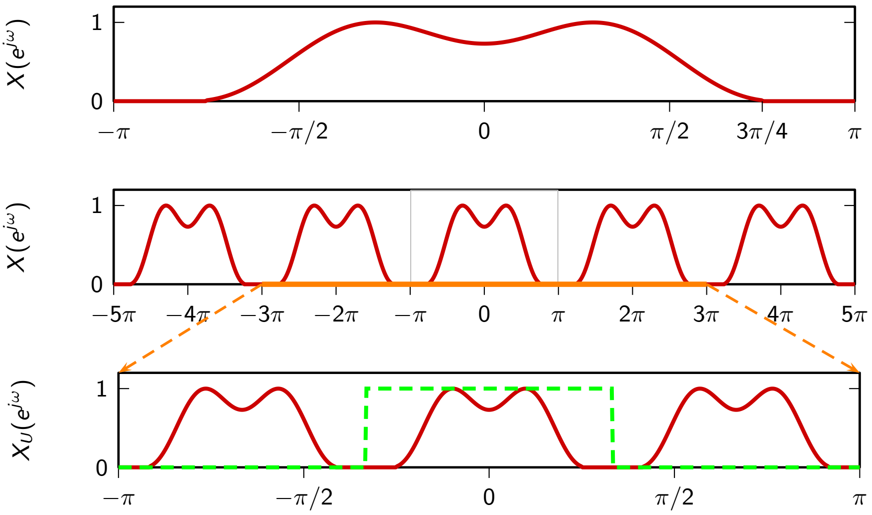
Upsampling in the digital domain



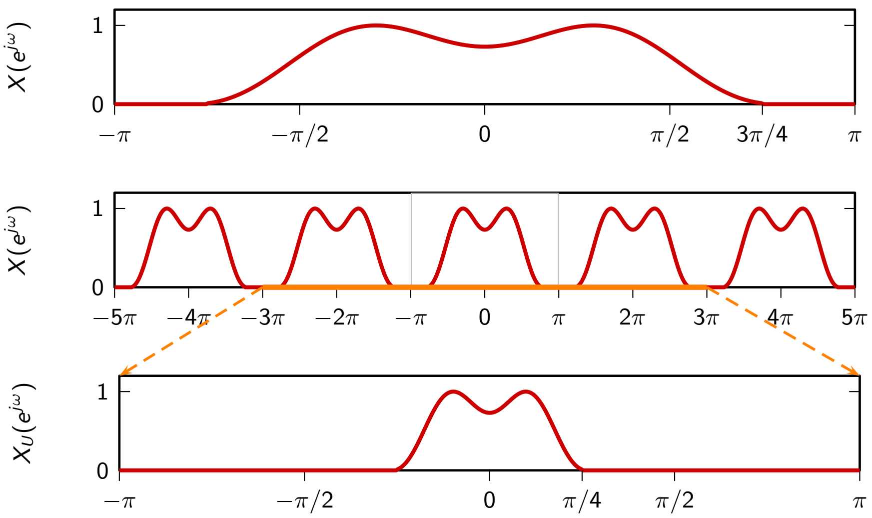
Upsampling in the digital domain



Upsampling in the digital domain



Upsampling in the digital domain



Upsampling in the digital domain

back in time domain...

- insert $K-1$ zeros after every sample
- ideal lowpass filtering with $\omega_c = \pi/K$

$$
\begin{aligned}
x'[n] &= x_U(n) * \operatorname{sinc}(n/K) \\
&= \sum_{i=-\infty}^{\infty} x_U[i] \operatorname{sinc}\left(\frac{n-i}{K}\right) \\
&= \sum_{m=-\infty}^{\infty} x[m] \operatorname{sinc}\left(\frac{n}{K} - m\right)
\end{aligned}
$$

Upsampling in the digital domain

back in time domain...

- insert $K-1$ zeros after every sample
- ideal lowpass filtering with $\omega_c = \pi/K$

$$
\begin{aligned}
x'[n] &= x_U(n) * \mathrm{sinc}(n/K) \\
&= \sum_{i=-\infty}^{\infty} x_U[i]\,\mathrm{sinc}\left(\frac{n-i}{K}\right) \\
&= \sum_{m=-\infty}^{\infty} x[m]\,\mathrm{sinc}\left(\frac{n}{K} - m\right)
\end{aligned}
$$

Downsampling

- given an upsampled signal we can always recover the original:

$$x[n] = x_U[nK]$$

- downsampling of generic signals more complicated (aliasing)

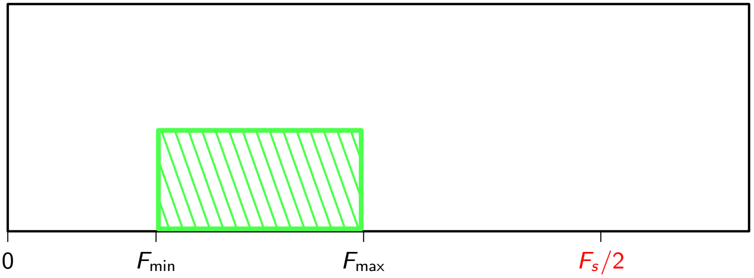
Downsampling

- given an upsampled signal we can always recover the original:

$$x[n] = x_U[nK]$$

- downsampling of generic signals more complicated (aliasing)

Remember the bandwidth constraint?



Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- $F_s > 2F_{\max}$ (obviously)
- $F_s = KW,\ K \in \mathbb{N}$
- $\omega_{\max} - \omega_{\min} = 2\pi \dfrac{W}{F_s} = \dfrac{2\pi}{K}$
- we can simply upsample by K

Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- $F_s > 2F_{\max}$ (obviously)
- $F_s = KW,\ K \in \mathbb{N}$
- $\omega_{\max} - \omega_{\min} = 2\pi \frac{W}{F_s} = \frac{2\pi}{K}$
- we can simply upsample by K

Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- $F_s > 2F_{\max}$ (obviously)
- $F_s = KW, \ K \in \mathbb{N}$

- $\omega_{\max} - \omega_{\min} = 2\pi \dfrac{W}{F_s} = \dfrac{2\pi}{K}$
- we can simply upsample by K

Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- $F_s > 2F_{\max}$ (obviously)
- $F_s = KW,\ K \in \mathbb{N}$

- $\omega_{\max} - \omega_{\min} = 2\pi \dfrac{W}{F_s} = \dfrac{2\pi}{K}$
- we can simply upsample by K

Data rates

- upsampling does not change the *data* rate
- we produce (and transmit) W symbols per second
- W is sometimes called the Baud rate of the system and is equal to the available bandwidth

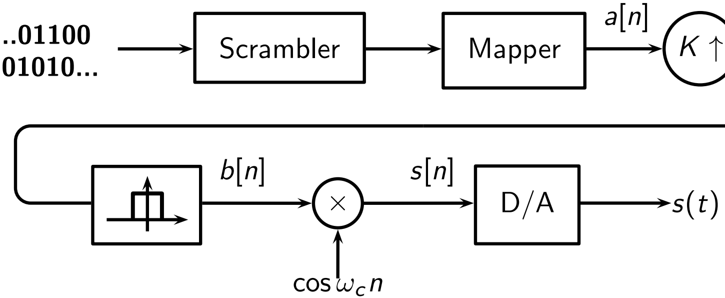
Data rates

- upsampling does not change the *data* rate
- we produce (and transmit) W symbols per second
- W is sometimes called the Baud rate of the system and is equal to the available bandwidth

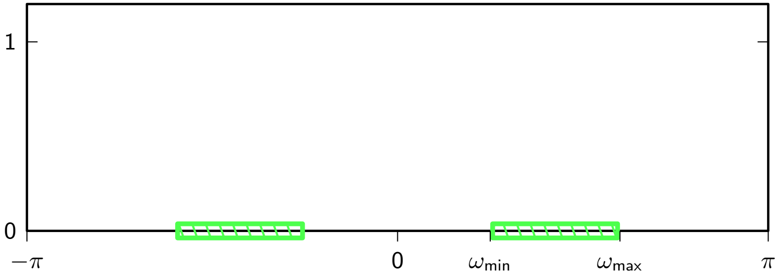
Data rates

- upsampling does not change the *data* rate
- we produce (and transmit) W symbols per second
- W is sometimes called the Baud rate of the system and is equal to the available bandwidth

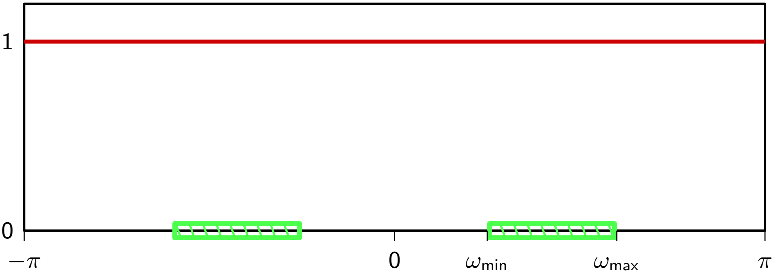
Transmitter design, continued



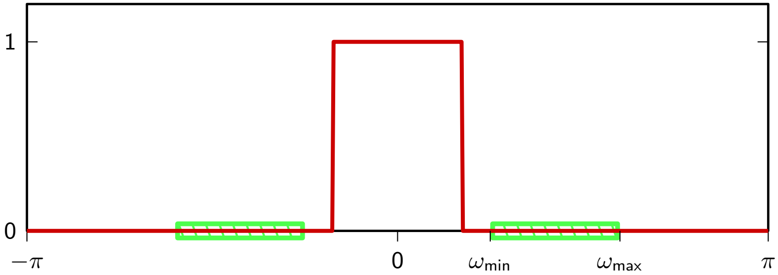
Meeting the bandwidth constraint



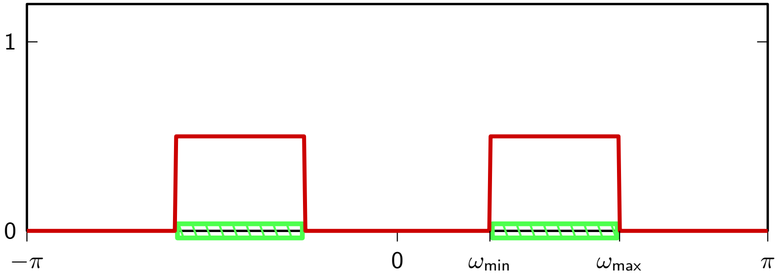
Meeting the bandwidth constraint



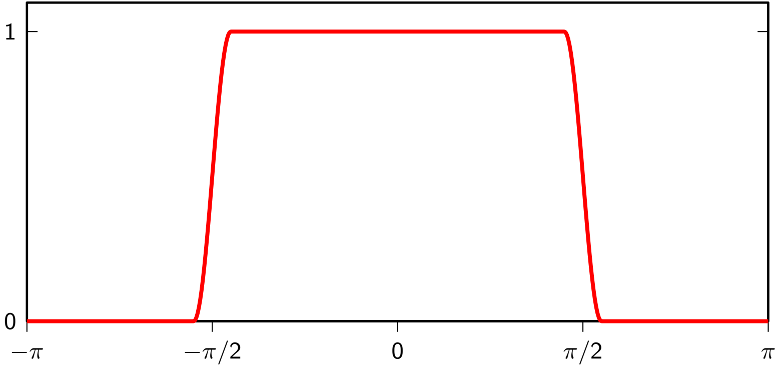
Meeting the bandwidth constraint



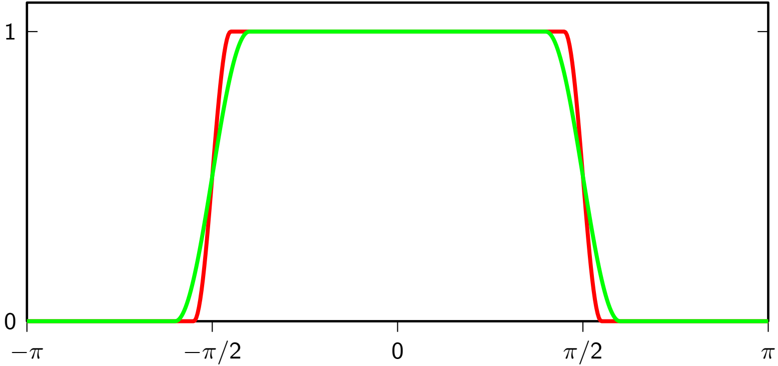
Meeting the bandwidth constraint



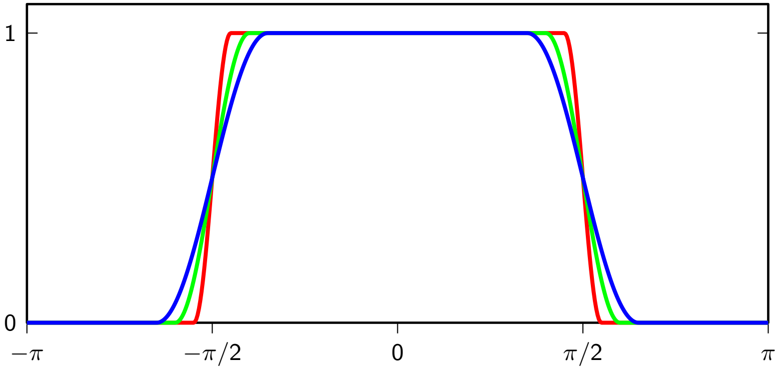
Raised Cosine



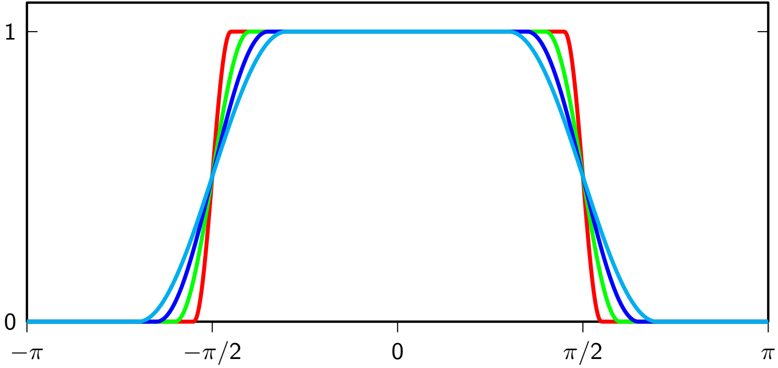
Raised Cosine



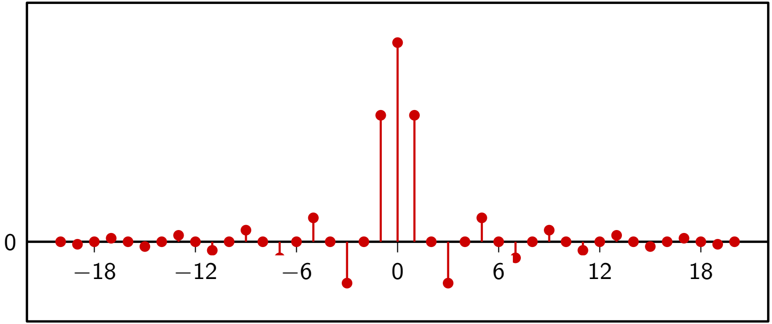
Raised Cosine



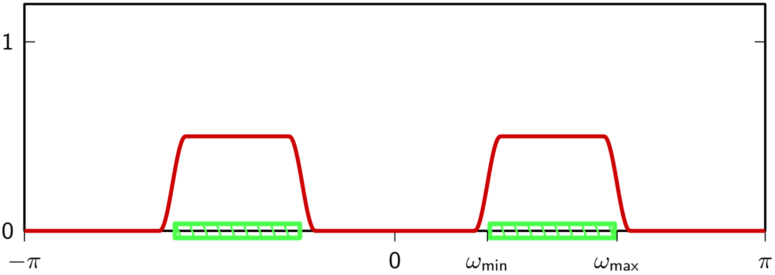
Raised Cosine



Raised Cosine



Spectral shaping with raised cosine



END OF MODULE 9.2

Digital Signal Processing

Module 9.3: Controlling the Power

Overview:

- Noise and probability of error
- Signaling alphabet and power
- QAM signaling

Overview:

- Noise and probability of error
- Signaling alphabet and power
- QAM signaling

Overview:

- Noise and probability of error
- Signaling alphabet and power
- QAM signaling

Transmission reliability

- transmitter sends a sequence of symbols $a[n]$
- receiver obtaines a sequence $\hat{a}[n]$
- even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- when noise is large, we make an error

Transmission reliability

- transmitter sends a sequence of symbols $a[n]$
- receiver obtaines a sequence $\hat{a}[n]$
- even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- when noise is large, we make an error

Transmission reliability

- transmitter sends a sequence of symbols $a[n]$
- receiver obtaines a sequence $\hat{a}[n]$
- even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- when noise is large, we make an error

Transmission reliability

- transmitter sends a sequence of symbols $a[n]$
- receiver obtaines a sequence $\hat{a}[n]$
- even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- when noise is large, we make an error

Probability of error

depends on:

- power of the noise wrt power of the signal
- decoding strategy
- *alphabet* of transmission symbols

Probability of error

depends on:

- power of the noise wrt power of the signal
- decoding strategy
- *alphabet* of transmission symbols

Probability of error

depends on:

- power of the noise wrt power of the signal
- decoding strategy
- *alphabet* of transmission symbols

Signaling alphabets

- we have a (randomized) bitstream coming in
- we want to send some upsampled and interpolated samples over the channel
- how do we go from bitstream to samples?

Signaling alphabets

- we have a (randomized) bitstream coming in
- we want to send some upsampled and interpolated samples over the channel
- how do we go from bitstream to samples?

Signaling alphabets

- we have a (randomized) bitstream coming in
- we want to send some upsampled and interpolated samples over the channel
- how do we go from bitstream to samples?

Mappers and slicers

mapper:

- split incoming bitstream into chunks
- assign a symbol $a[n]$ from a finite alphabet $\mathcal{A}$ to each chunk

slicer:

- receive a value $\hat{a}[n]$
- decide which symbol from $\mathcal{A}$ is "closest" to $\hat{a}[n]$
- piece back together the corresponding bitstream

Example: two-level signaling

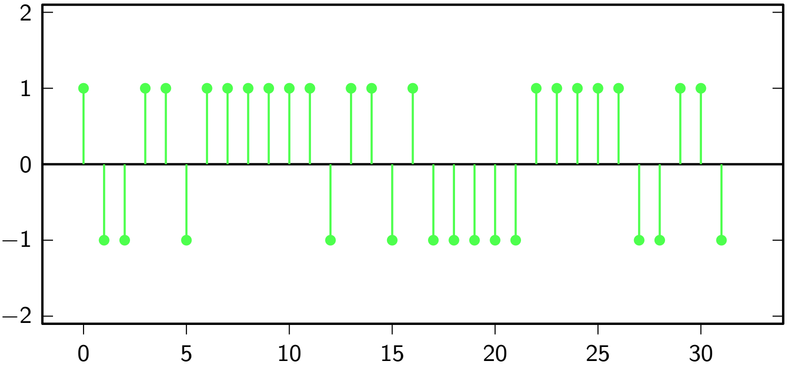
mapper:

- split incoming bitstream into single bits
- $a[n] = G$ if the bit is 1, $a[n] = -G$ if the bit is 0

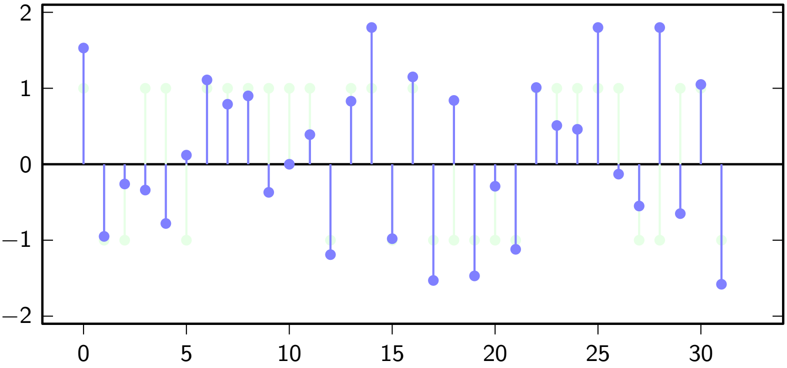
slicer:

- n-th bit $= \begin{cases} 1 & \text{if } \hat{a}[n] > 0 \\ 0 & \text{otherwise} \end{cases}$

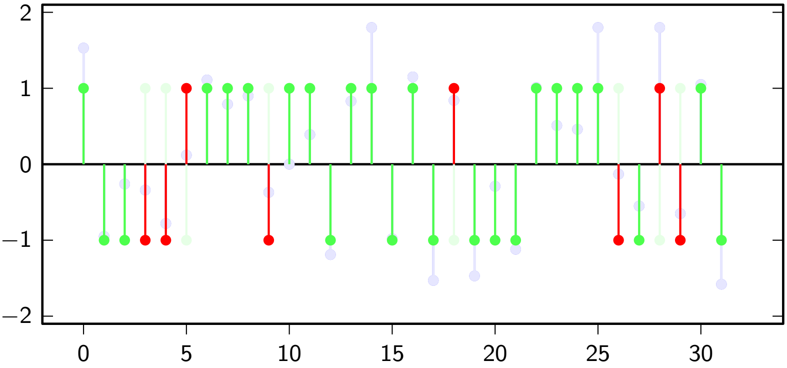
Example: two-level signaling



Example: two-level signaling



Example: two-level signaling



Example: two-level signaling

let's look at the probability of error after making some hypotheses:

- $\hat{a}[n] = a[n] + \eta[n]$
- bits in bitstream are equiprobable
- noise and signal are independent
- noise is additive white Gaussian noise with zero mean and variance σ_0

Example: two-level signaling

$$
\begin{aligned}
P_{\text{err}} = P[\ \eta[n] < -G \ \mid \ n\text{-th bit is } 1\ ]\, P[\ n\text{-th bit is } 1\ ] + \\
P[\ \eta[n] > G \ \mid \ n\text{-th bit is } 0\ ]\, P[\ n\text{-th bit is } 0\ ] \\
= (P[\ \eta[n] < -G\ ] + P[\ \eta[n] > G\ ])/2 \\
= P[\ \eta[n] > G\ ] \\
= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}}\, d\tau \\
= Q(G/\sigma_0) = \frac{1}{2}\operatorname{erfc}((G/\sigma_0)/\sqrt{2})
\end{aligned}
$$

Example: two-level signaling

$$
\begin{aligned}
P_{\text{err}} &= P[\ \eta[n] < -G \ \mid\ n\text{-th bit is } 1\ ]\, P[\ n\text{-th bit is } 1\ ] + \\
&\qquad P[\ \eta[n] > G \ \mid\ n\text{-th bit is } 0\ ]\, P[\ n\text{-th bit is } 0\ ] \\
&= (P[\ \eta[n] < -G\ ] + P[\ \eta[n] > G\ ])/2 \\
&= P[\ \eta[n] > G\ ] \\
&= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}}\, d\tau \\
&= Q(G/\sigma_0) = \frac{1}{2}\text{erfc}((G/\sigma_0)/\sqrt{2})
\end{aligned}
$$

Example: two-level signaling

$$
\begin{aligned}
P_{\text{err}} &= P[\ \eta[n] < -G \ \mid\ n\text{-th bit is } 1\ ]\,P[\ n\text{-th bit is } 1\ ] + \\
&\qquad P[\ \eta[n] > G \ \mid\ n\text{-th bit is } 0\ ]\,P[\ n\text{-th bit is } 0\ ] \\
&= (P[\ \eta[n] < -G\ ] + P[\ \eta[n] > G\ ])/2 \\
&= P[\ \eta[n] > G\ ] \\
&= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}}\, d\tau \\
&= Q(G/\sigma_0) = \frac{1}{2}\operatorname{erfc}((G/\sigma_0)/\sqrt{2})
\end{aligned}
$$

$$
\begin{aligned}
P_{\text{err}} &= P[\ \eta[n] < -G \ \mid\ n\text{-th bit is } 1\ ]\, P[\ n\text{-th bit is } 1\ ] + \\
&\quad P[\ \eta[n] > G \ \mid\ n\text{-th bit is } 0\ ]\, P[\ n\text{-th bit is } 0\ ] \\
&= (P[\ \eta[n] < -G\ ] + P[\ \eta[n] > G\ ])/2 \\
&= P[\ \eta[n] > G\ ] \\
&= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau \\
&= Q(G/\sigma_0) = \frac{1}{2}\operatorname{erfc}((G/\sigma_0)/\sqrt{2})
\end{aligned}
$$

Example: two-level signaling

$$
\begin{aligned}
P_{\text{err}} = {} & P[\ \eta[n] < -G \ \mid\ n\text{-th bit is } 1\ ]\, P[\ n\text{-th bit is } 1\ ] + \\
& P[\ \eta[n] > G \ \mid\ n\text{-th bit is } 0\ ]\, P[\ n\text{-th bit is } 0\ ] \\
= {} & (P[\ \eta[n] < -G\ ] + P[\ \eta[n] > G\ ])/2 \\
= {} & P[\ \eta[n] > G\ ] \\
= {} & \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau \\
= {} & Q(G/\sigma_0) = \frac{1}{2}\operatorname{erfc}((G/\sigma_0)/\sqrt{2})
\end{aligned}
$$

Example: two-level signaling

transmitted power

$$\sigma_s^2 = G^2\, P[n\text{-th bit is 1}] + G^2\, P[n\text{-th bit is 0}]$$

$$= G^2$$

$$P_{\text{err}} = Q(\sigma_s/\sigma_0) = Q(\sqrt{\text{SNR}})$$

Example: two-level signaling

transmitted power

$$\sigma_s^2 = G^2\, P[n\text{-th bit is 1}] + G^2\, P[n\text{-th bit is 0}]$$
$$= G^2$$

$$P_{\text{err}} = Q(\sigma_s/\sigma_0) = Q(\sqrt{\text{SNR}})$$

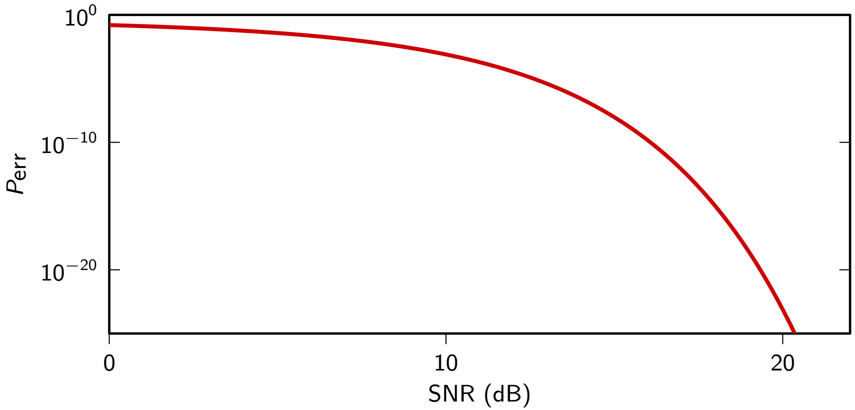
Example: two-level signaling

transmitted power

$$\sigma_s^2 = G^2\, P[n\text{-th bit is 1}] + G^2\, P[n\text{-th bit is 0}]$$
$$= G^2$$

$$P_{\text{err}} = Q(\sigma_s/\sigma_0) = Q(\sqrt{\text{SNR}})$$

Probability of error



Lesson learned:

- to reduce the probability of error increase G
- increasing G increases the power
- we can't go above the channel's power constraint!

Lesson learned:

- to reduce the probability of error increase G
- increasing G increases the power
- we can't go above the channel's power constraint!

Lesson learned:

- to reduce the probability of error increase G
- increasing G increases the power
- we can't go above the channel's power constraint!

Multilevel signaling

- binary signaling is not very efficient (one bit at a time)
- to increase the throughput we can use multilevel signaling
- many ways to do so, we will just scratch the surface

Multilevel signaling

- binary signaling is not very efficient (one bit at a time)
- to increase the throughput we can use multilevel signaling
- many ways to do so, we will just scratch the surface

Multilevel signaling

- binary signaling is not very efficient (one bit at a time)
- to increase the throughput we can use multilevel signaling
- many ways to do so, we will just scratch the surface

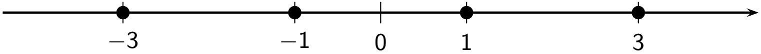
mapper:

- split incoming bitstream into chunks of M bits
- chunks define a sequence of integers $k[n] \in \{0, 1, \ldots, 2^M - 1\}$
- $a[n] = G((-2^M + 1) + 2k[n])$ (odd integers around zero)

slicer:

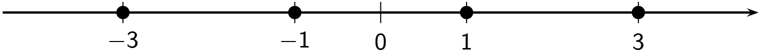
- $a'[n] = \arg\min_{a \in \mathcal{A}} [|\hat{a}[n] - a|]$

PAM, $M = 2, G = 1$



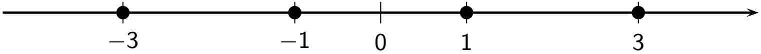
- distance between points is $2G$
- using odd integers creates a zero-mean sequence

PAM, $M = 2$, $G = 1$



- distance between points is $2G$
- using odd integers creates a zero-mean sequence

PAM, $M = 2$, $G = 1$



- distance between points is $2G$
- using odd integers creates a zero-mean sequence

From PAM to QAM

- error analysis for PAM along the lines of binary signaling
- can we increase the throughput even further?
- here's a wild idea, let's use complex numbers

From PAM to QAM

- error analysis for PAM along the lines of binary signaling
- can we increase the throughput even further?
- here's a wild idea, let's use complex numbers

From PAM to QAM

- error analysis for PAM along the lines of binary signaling
- can we increase the throughput even further?
- here's a wild idea, let's use complex numbers

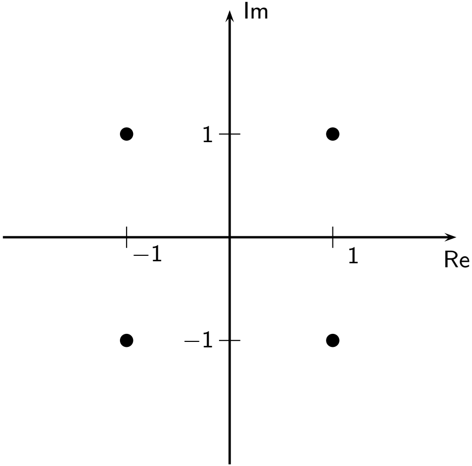
mapper:

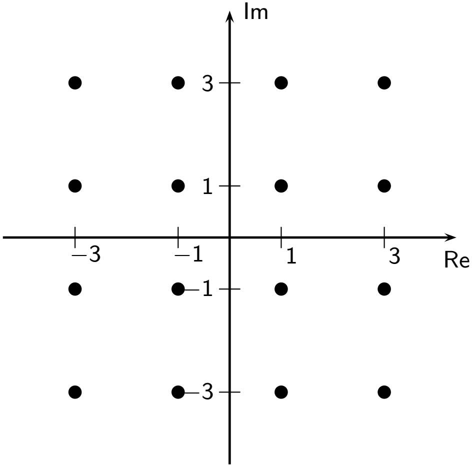
- split incoming bitstream into chunks of M bits, M even
- use $M/2$ bits to define a PAM sequence $a_r[n]$
- use the remaining $M/2$ bits to define an independent PAM sequence $a_i[n]$
- $a[n] = G(a_r[n] + ja_i[n])$

slicer:

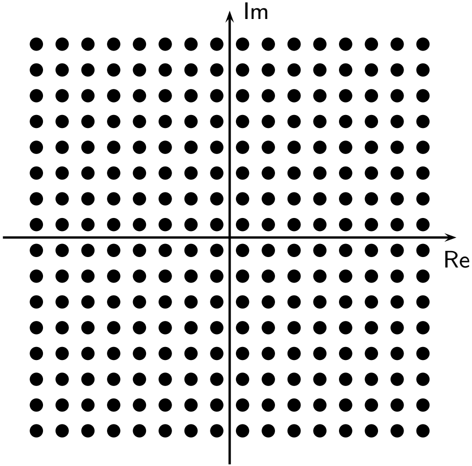
- $a'[n] = \arg\min_{a\in\mathcal{A}}[|\hat{a}[n] - a|]$

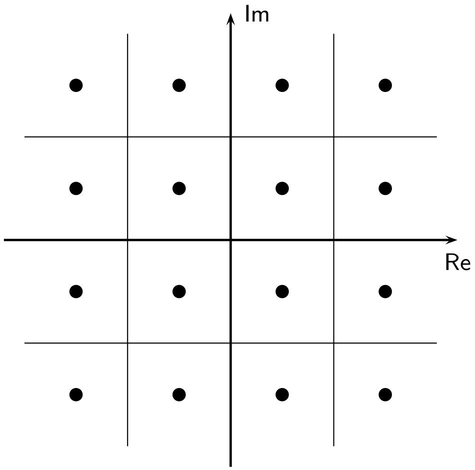
QAM, $M = 2$, $G = 1$



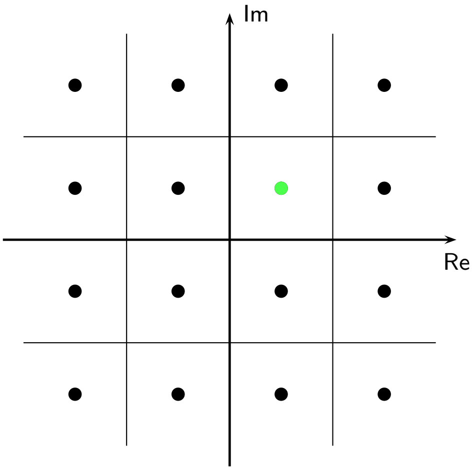

Im
3
1
−3
−1
1
3
Re
−1
−3

QAM, $M = 8$, $G = 1$

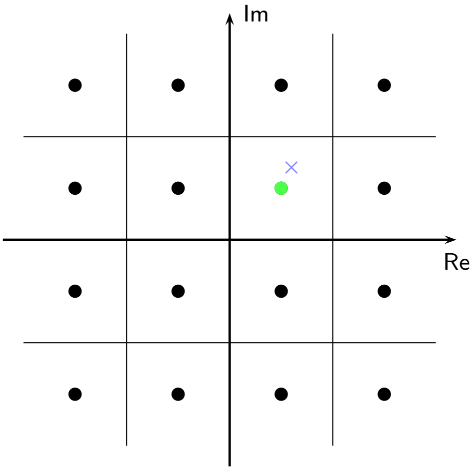



Im
Re

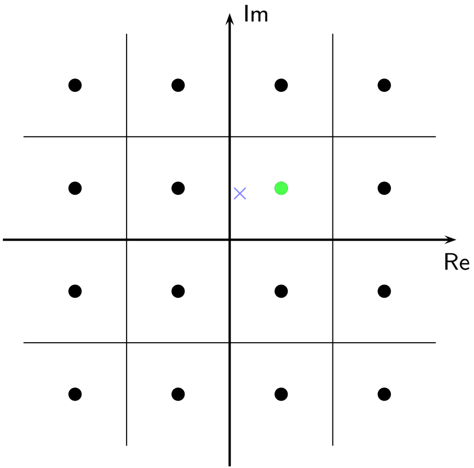
QAM



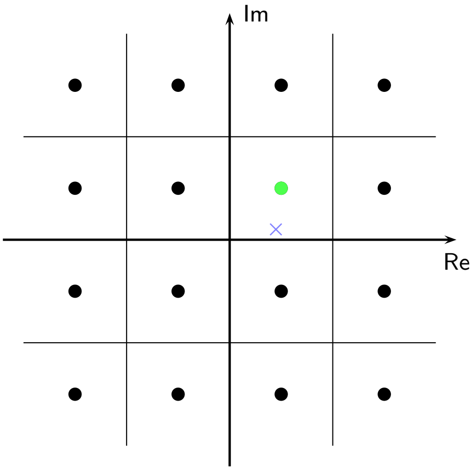
QAM

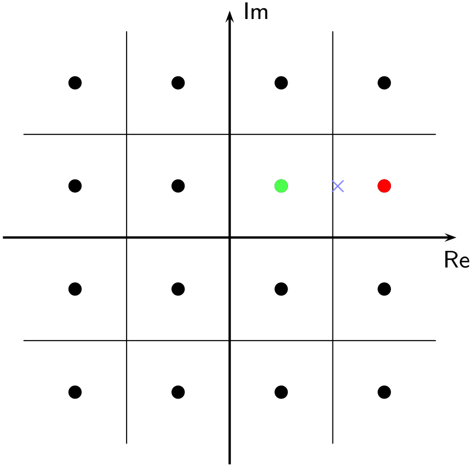


QAM



QAM




Im
Re

QAM, probability of error

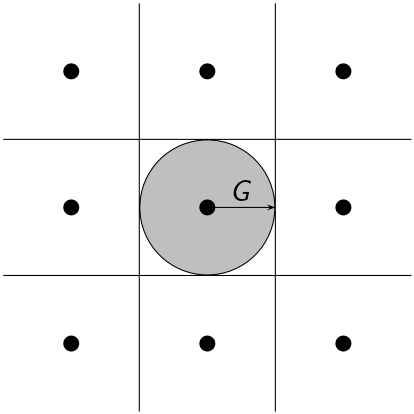
$$P_{\text{err}} = 1 - P[|\operatorname{Re}(\eta[n])| < G \wedge |\operatorname{Im}(\eta[n])| < G]$$

$$= 1 - \int_D f_\eta(z)\, dz$$

QAM, probability of error

$$
\begin{aligned}
P_{\text{err}} &= 1 - P[|\operatorname{Re}(\eta[n])| < G \wedge |\operatorname{Im}(\eta[n])| < G] \\
&= 1 - \int_D f_\eta(z)\, dz
\end{aligned}
$$

QAM, probability of error



QAM, probability of error

$$P_{\text{err}} \approx e^{-\frac{G^2}{\sigma_0^2}}$$

QAM, probability of error

transmitted power (all symbols equiprobable and independent):

$$\sigma_s^2 = G^2 \frac{1}{2^M} \sum_{a \in \mathcal{A}} |a|^2$$

$$= G^2 \frac{2}{3}(2^M - 1)$$

$$P_{err} \approx e^{-\frac{G^2}{\sigma_0^2}} \approx e^{-3 \cdot 2^{-(M+1)}\,\mathrm{SNR}}$$

QAM, probability of error

transmitted power (all symbols equiprobable and independent):

$$\sigma_s^2 = G^2 \frac{1}{2^M} \sum_{a \in \mathcal{A}} |a|^2$$

$$= G^2 \frac{2}{3}(2^M - 1)$$

$$P_{\text{err}} \approx e^{-\frac{G^2}{\sigma_0^2}} \approx e^{-3 \cdot 2^{-(M+1)}\,\text{SNR}}$$

QAM, probability of error

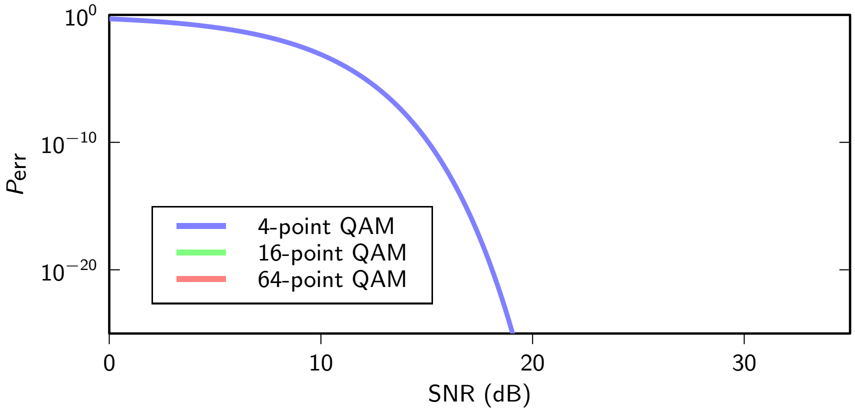
transmitted power (all symbols equiprobable and independent):

$$\sigma_s^2 = G^2 \frac{1}{2^M} \sum_{a \in \mathcal{A}} |a|^2$$

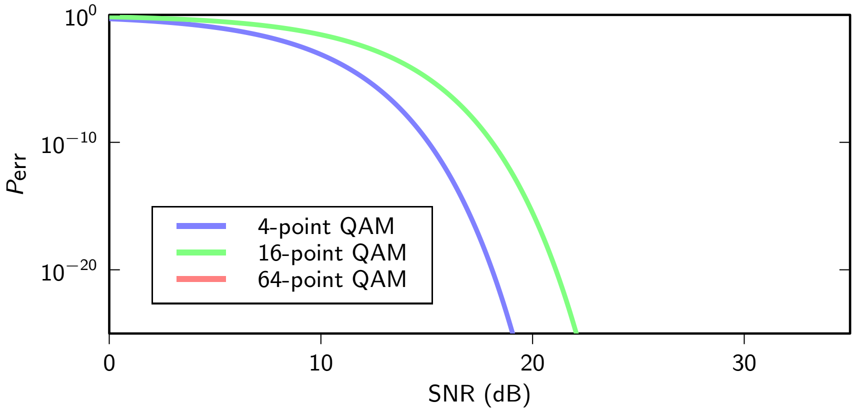
$$= G^2 \frac{2}{3}(2^M - 1)$$

$$P_{\text{err}} \approx e^{-\frac{G^2}{\sigma_0^2}} \approx e^{-3 \cdot 2^{-(M+1)}\,\text{SNR}}$$

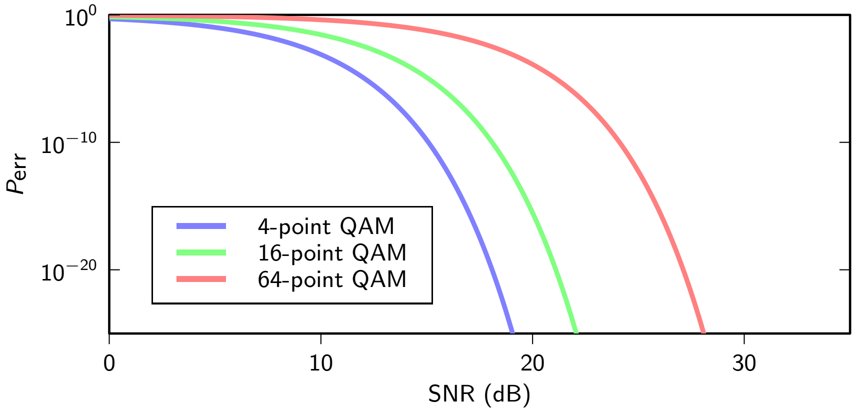
Probability of error



Probability of error



Probability of error



QAM, the recipe

- pick a probability of error you can live with (e.g. 10^{-6})
- find out the SNR imposed by the channel's power constraint
- $M = \log_2\left(1 - \frac{3}{2}\frac{\text{SNR}}{\ln(p_e)}\right)$
- final throughput will be MW

QAM, the recipe

- pick a probability of error you can live with (e.g. 10^{-6})
- find out the SNR imposed by the channel's power constraint
- $M = \log_2\left(1 - \frac{3}{2}\frac{\mathrm{SNR}}{\ln(p_e)}\right)$
- final throughput will be MW

QAM, the recipe

- pick a probability of error you can live with (e.g. 10^{-6})
- find out the SNR imposed by the channel's power constraint
- $M = \log_2\left(1 - \frac{3}{2}\frac{\mathrm{SNR}}{\ln(p_e)}\right)$
- final throughput will be MW

QAM, the recipe

- pick a probability of error you can live with (e.g. 10^{-6})
- find out the SNR imposed by the channel's power constraint
- $M = \log_2\left(1 - \frac{3}{2}\frac{\text{SNR}}{\ln(p_e)}\right)$
- final throughput will be MW

where we stand:

- we know how to fit the bandwidth constraint
- with QAM, we know how many bits per symbol we can use given the power constraint
- we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

where we stand:

- we know how to fit the bandwidth constraint
- with QAM, we know how many bits per symbol we can use given the power constraint
- we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

where we stand:

- we know how to fit the bandwidth constraint
- with QAM, we know how many bits per symbol we can use given the power constraint
- we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

where we stand:

- we know how to fit the bandwidth constraint
- with QAM, we know how many bits per symbol we can use given the power constraint
- we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

END OF MODULE 9.3

Digital Signal Processing

Module 9.4: Modulation and Demodulation

Overview:

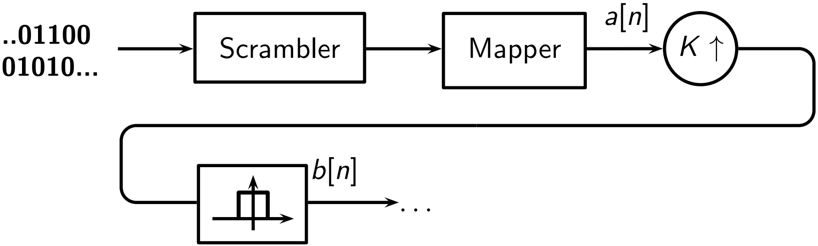
- Trasmitting and recovering the complex passband signal
- Design example
- Channel capacity

Overview:

- Trasmitting and recovering the complex passband signal
- Design example
- Channel capacity

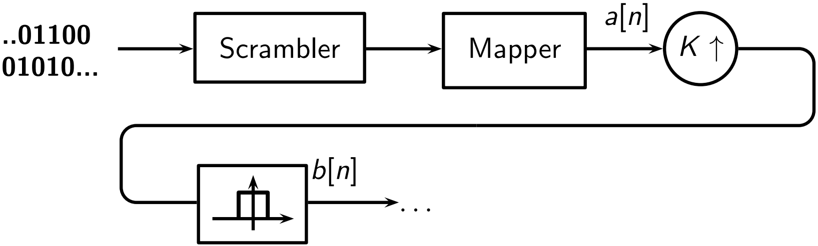
Overview:

- Trasmitting and recovering the complex passband signal
- Design example
- Channel capacity



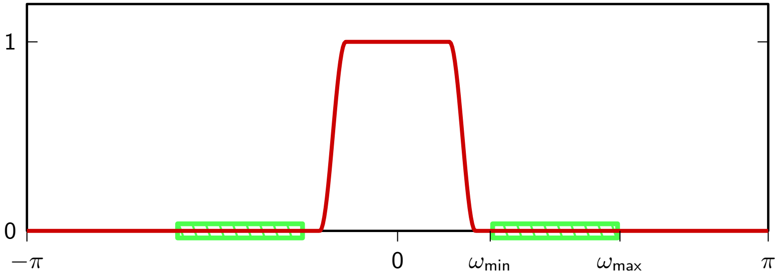
$b[n] = b_r[n] + jb_i[n]$ is a complex-valued baseband signal

QAM transmitter design



$b[n] = b_r[n] + jb_i[n]$ is a complex-valued baseband signal

Complex baseband signal



The passband signal

$$
\begin{aligned}
s[n] &= \mathrm{Re}\{b[n]\, e^{j\omega_c n}\} \\
&= \mathrm{Re}\{(b_r[n] + jb_i[n])(\cos\omega_c n + j\sin\omega_c n)\} \\
&= b_r[n]\cos\omega_c n - b_i[n]\sin\omega_c n
\end{aligned}
$$

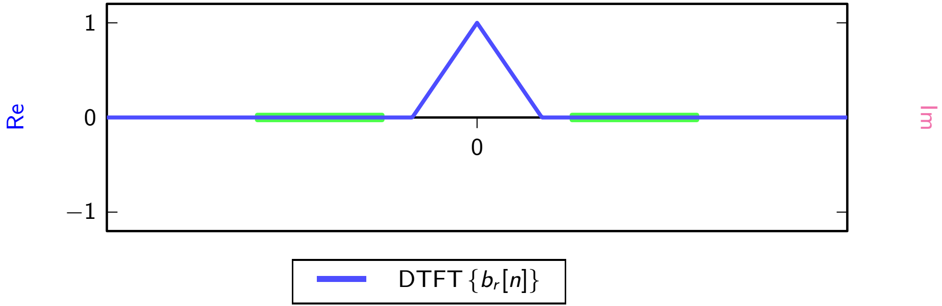
The passband signal

$$
\begin{aligned}
s[n] &= \mathrm{Re}\{b[n]\, e^{j\omega_c n}\} \\
&= \mathrm{Re}\{(b_r[n] + jb_i[n])(\cos\omega_c n + j\sin\omega_c n)\} \\
&= b_r[n]\cos\omega_c n - b_i[n]\sin\omega_c n
\end{aligned}
$$

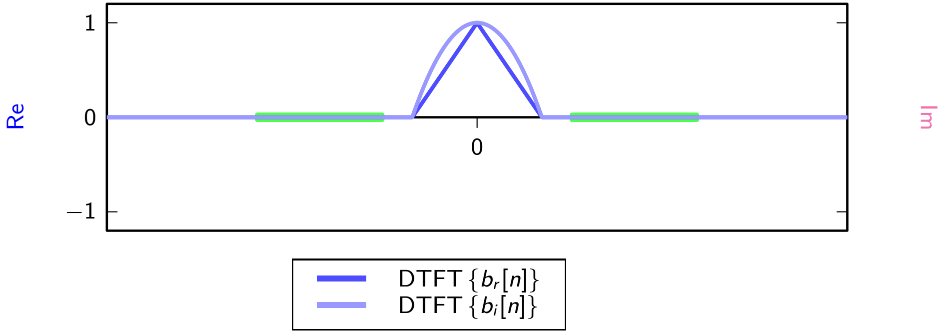
The passband signal

$$
\begin{aligned}
s[n] &= \mathrm{Re}\{b[n]\, e^{j\omega_c n}\} \\
&= \mathrm{Re}\{(b_r[n] + jb_i[n])(\cos\omega_c n + j\sin\omega_c n)\} \\
&= b_r[n]\cos\omega_c n - b_i[n]\sin\omega_c n
\end{aligned}
$$

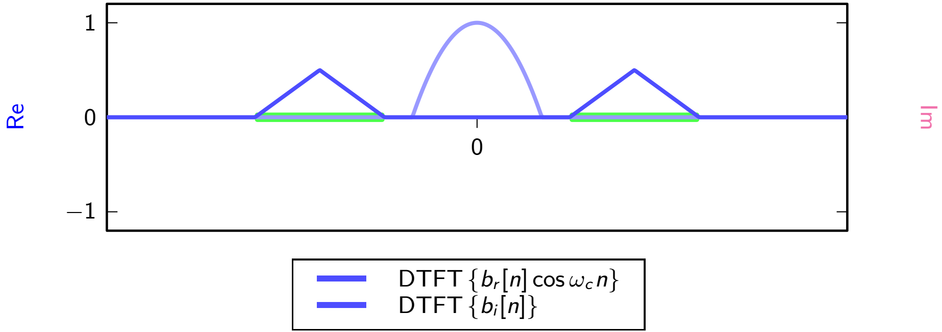
Complex baseband signal

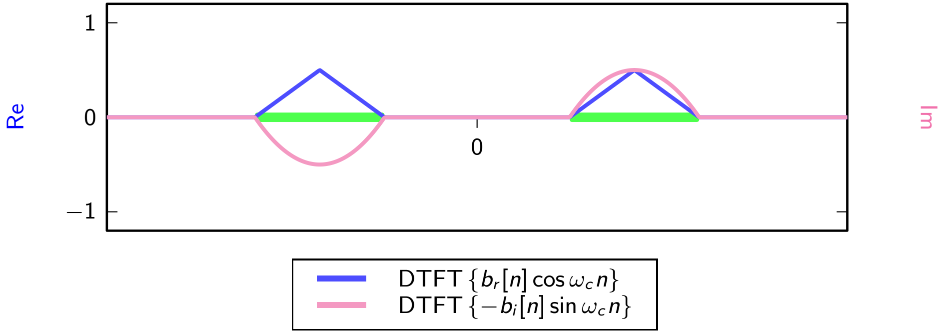


Complex baseband signal



Complex baseband signal





1
0
−1
0
Re
Im
DTFT $\{b_r[n]\cos\omega_c n\}$
DTFT $\{-b_i[n]\sin\omega_c n\}$

let's try the usual method (multiplying by the carrier, see Module 5.5):

$$
\begin{aligned}
s[n]\cos\omega_c n &= b_r[n]\cos^2\omega_c n - b_i[n]\sin\omega_c n\cos\omega_c n \\
&= b_r[n]\frac{1+\cos 2\omega_c n}{2} - b_i[n]\frac{\sin 2\omega_c n}{2} \\
&= \frac{1}{2}b_r[n] + \frac{1}{2}(b_r[n]\cos 2\omega_c n - b_i[n]\sin 2\omega_c n)
\end{aligned}
$$

Recovering the baseband signal

let's try the usual method (multiplying by the carrier, see Module 5.5):

$$s[n]\cos\omega_c n = b_r[n]\cos^2\omega_c n - b_i[n]\sin\omega_c n\cos\omega_c n$$

$$= b_r[n]\frac{1+\cos 2\omega_c n}{2} - b_i[n]\frac{\sin 2\omega_c n}{2}$$

$$= \frac{1}{2}b_r[n] + \frac{1}{2}(b_r[n]\cos 2\omega_c n - b_i[n]\sin 2\omega_c n)$$

Recovering the baseband signal

let's try the usual method (multiplying by the carrier, see Module 5.5):

$$
\begin{aligned}
s[n]\cos\omega_c n &= b_r[n]\cos^2\omega_c n - b_i[n]\sin\omega_c n\cos\omega_c n \\
&= b_r[n]\frac{1+\cos 2\omega_c n}{2} - b_i[n]\frac{\sin 2\omega_c n}{2} \\
&= \frac{1}{2}b_r[n] + \frac{1}{2}(b_r[n]\cos 2\omega_c n - b_i[n]\sin 2\omega_c n)
\end{aligned}
$$

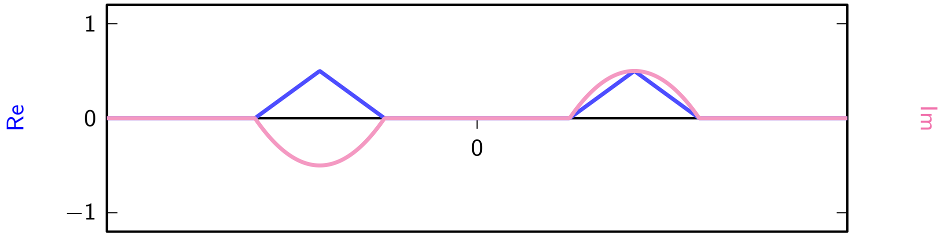
Recovering the baseband signal

let's try the usual method (multiplying by the carrier, see Module 5.5):

$$
\begin{aligned}
s[n]\cos\omega_c n &= b_r[n]\cos^2\omega_c n - b_i[n]\sin\omega_c n\cos\omega_c n \\
&= b_r[n]\frac{1+\cos 2\omega_c n}{2} - b_i[n]\frac{\sin 2\omega_c n}{2} \\
&= \frac{1}{2}b_r[n] + \frac{1}{2}(b_r[n]\cos 2\omega_c n - b_i[n]\sin 2\omega_c n)
\end{aligned}
$$

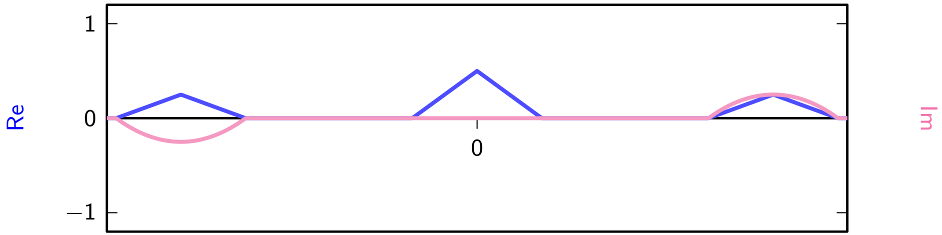
Complex baseband signal

$$\text{DTFT}\left\{b_r[n]\cos\omega_c n - b_i[n]\sin\omega_c n\right\}$$



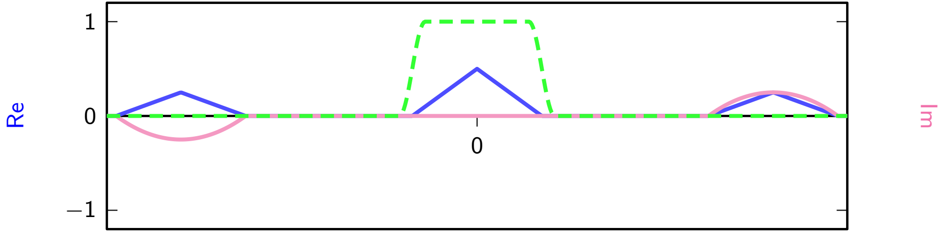
Complex baseband signal

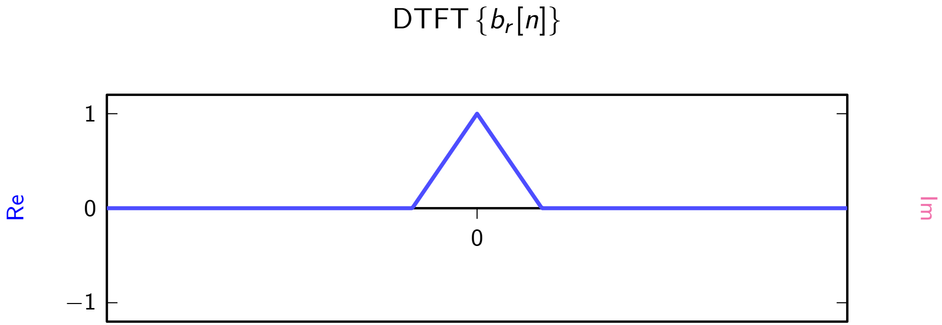
$$\mathrm{DTFT}\left\{\left(b_r[n]\cos\omega_c n - b_i[n]\sin\omega_c n\right)\cos\omega_c n\right\}$$



Complex baseband signal

$$\text{DTFT}\left\{(b_r[n]\cos\omega_c n - b_i[n]\sin\omega_c n)\cos\omega_c n\right\}$$




DTFT $\{b_r[n]\}$
1
0
−1
0
Re
Im

Recovering the baseband signal

- as a lowpass filter, you can use the same filter used in upsampling
- matched filter technique

Recovering the baseband signal

- as a lowpass filter, you can use the same filter used in upsampling
- matched filter technique

similarly:

$$
\begin{aligned}
s[n]\sin\omega_c n &= b_r[n]\cos\omega_c n \sin\omega_c n - b_i[n]\sin^2\omega_c n \\
&= -\frac{1}{2}b_i[n] + \frac{1}{2}(b_r[n]\sin 2\omega_c n - b_i[n]\cos 2\omega_c n)
\end{aligned}
$$

similarly:

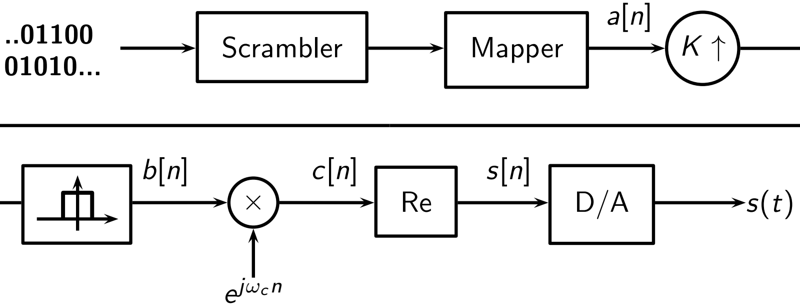
$$s[n]\sin\omega_c n = b_r[n]\cos\omega_c n \sin\omega_c n - b_i[n]\sin^2\omega_c n$$

$$= -\frac{1}{2}b_i[n] + \frac{1}{2}(b_r[n]\sin 2\omega_c n - b_i[n]\cos 2\omega_c n)$$

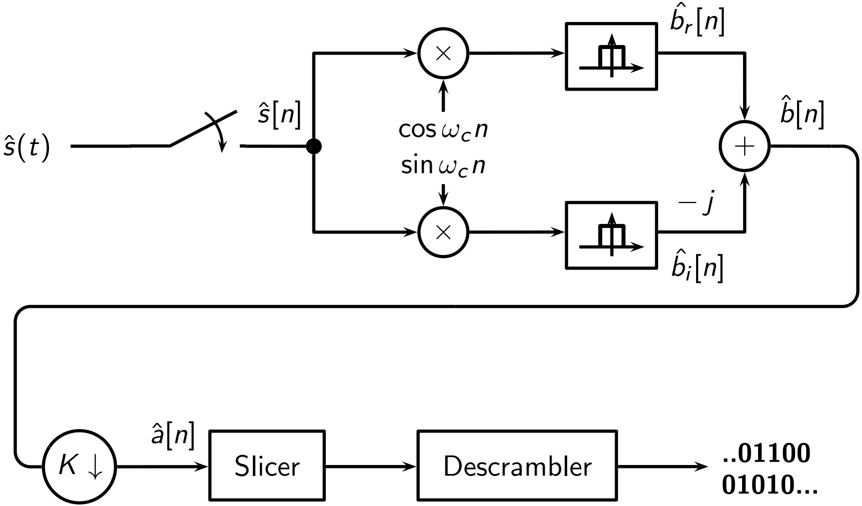
similarly:

$$
\begin{aligned}
s[n]\sin\omega_c n &= b_r[n]\cos\omega_c n \sin\omega_c n - b_i[n]\sin^2\omega_c n \\
&= -\frac{1}{2}b_i[n] + \frac{1}{2}\left(b_r[n]\sin 2\omega_c n - b_i[n]\cos 2\omega_c n\right)
\end{aligned}
$$

QAM transmitter, final design



QAM receiver, idealized design



Example: the V.32 voiceband modem

- analog telephone channel: $F_{\text{min}} = 450\text{Hz}$, $F_{\text{max}} = 2850\text{Hz}$
- usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- analog telephone channel: $F_{\text{min}} = 450\text{Hz}$, $F_{\text{max}} = 2850\text{Hz}$
- usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- analog telephone channel: $F_{\text{min}} = 450\text{Hz}$, $F_{\text{max}} = 2850\text{Hz}$
- usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- analog telephone channel: $F_{\text{min}} = 450\text{Hz}$, $F_{\text{max}} = 2850\text{Hz}$
- usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- maximum SNR: 22dB
- pick $P_{\text{err}} = 10^{-6}$
- using QAM, we find

$$M = \log_2\left(1 - \frac{3}{2}\frac{10^{22/10}}{\ln(10^{-6})}\right) \approx 4.1865$$

  so we pick $M = 4$ and use a 16-point constellation
- final data rate is $WM = 9600$ bits per second

Example: the V.32 voiceband modem

- maximum SNR: 22dB
- pick $P_{\text{err}} = 10^{-6}$
- using QAM, we find

$$M = \log_2 \left(1 - \frac{3}{2} \frac{10^{22/10}}{\ln(10^{-6})} \right) \approx 4.1865$$

  so we pick $M = 4$ and use a 16-point constellation
- final data rate is $WM = 9600$ bits per second

Example: the V.32 voiceband modem

- maximum SNR: 22dB
- pick $P_{\text{err}} = 10^{-6}$
- using QAM, we find

$$M = \log_2\left(1 - \frac{3}{2}\frac{10^{22/10}}{\ln(10^{-6})}\right) \approx 4.1865$$

so we pick $M = 4$ and use a 16-point constellation

- final data rate is $WM = 9600$ bits per second

Example: the V.32 voiceband modem

- maximum SNR: 22dB
- pick $P_{\text{err}} = 10^{-6}$
- using QAM, we find

$$M = \log_2\left(1 - \frac{3}{2}\frac{10^{22/10}}{\ln(10^{-6})}\right) \approx 4.1865$$

so we pick $M = 4$ and use a 16-point constellation

- final data rate is $WM = 9600$ bits per second

Theoretical channel capacity

- we used very specific design choices to derive the throughput
- what is the best one can do?
- Shannon's capacity formula is the upper bound

$$C = W \log_2(1 + SNR)$$

- for instance, for the previous example $C \approx 17500$ bps
- the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- we used very specific design choices to derive the throughput
- what is the best one can do?
- Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- for instance, for the previous example $C \approx 17500$ bps
- the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- we used very specific design choices to derive the throughput
- what is the best one can do?
- Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- for instance, for the previous example $C \approx 17500$ bps
- the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- we used very specific design choices to derive the throughput
- what is the best one can do?
- Shannon's capacity formula is the upper bound

$$C = W \log_2(1 + SNR)$$

- for instance, for the previous example $C \approx 17500$ bps
- the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- we used very specific design choices to derive the throughput
- what is the best one can do?
- Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- for instance, for the previous example $C \approx 17500$ bps
- the gap can be narrowed by more advanced coding techniques

END OF MODULE 9.4

Digital Signal Processing

Module 9.5: Receiver Design

Overview:

- Adaptive equalization
- Timing recovery

Overview:

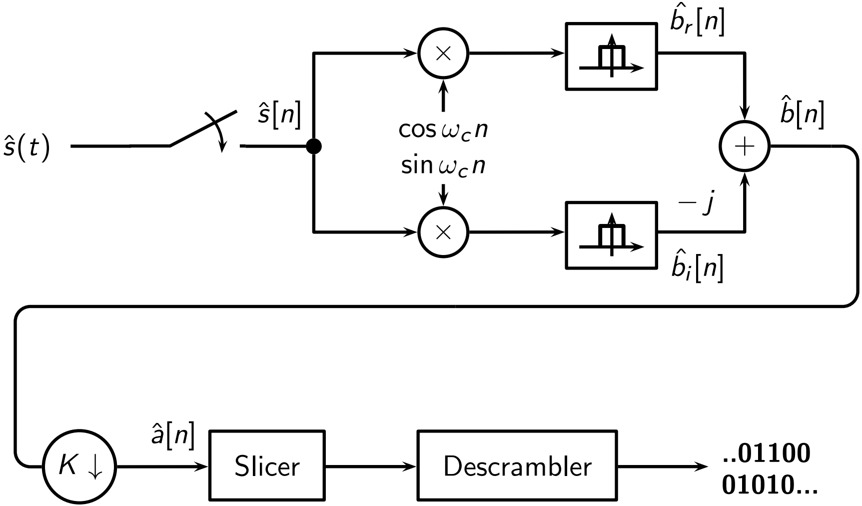
- Adaptive equalization
- Timing recovery

A blast from the past

A blast from the past

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

Graphically



if $\hat{s}[n] = \cos((\omega_c + \omega_0)n)$:

$$\begin{aligned}
\hat{b}[n] &= \mathcal{H}\{\cos((\omega_c + \omega_0)n)\cos(\omega_c n) - j\cos((\omega_c + \omega_0)n)\sin(\omega_c n)\} \\
&= \mathcal{H}\{\cos(\omega_0 n) + \cos((2\omega_c + \omega_0)n) - j\sin((2\omega_c + \omega_0)n) + j\sin(\omega_0 n)\} \\
&= \cos(\omega_0 n) + j\sin(\omega_0 n) \\
&= e^{j\omega_0 n}
\end{aligned}$$

Pilot tones

if $\hat{s}[n] = \cos((\omega_c + \omega_0)n)$:

$$\hat{b}[n] = \mathcal{H}\{\cos((\omega_c + \omega_0)n)\cos(\omega_c n) - j\cos((\omega_c + \omega_0)n)\sin(\omega_c n)\}$$

$$= \mathcal{H}\{\cos(\omega_0 n) + \cos((2\omega_c + \omega_0)n) - j\sin((2\omega_c + \omega_0)n) + j\sin(\omega_0 n)\}$$

$$= \cos(\omega_0 n) + j\sin(\omega_0 n)$$

$$= e^{j\omega_0 n}$$

Pilot tones

$$\text{if } \hat{s}[n] = \cos((\omega_c + \omega_0)n):$$

$$\begin{aligned}
\hat{b}[n] &= \mathcal{H}\{\cos((\omega_c + \omega_0)n)\cos(\omega_c n) - j\cos((\omega_c + \omega_0)n)\sin(\omega_c n)\} \\
&= \mathcal{H}\{\cos(\omega_0 n) + \cos((2\omega_c + \omega_0)n) - j\sin((2\omega_c + \omega_0)n) + j\sin(\omega_0 n)\} \\
&= \cos(\omega_0 n) + j\sin(\omega_0 n) \\
&= e^{j\omega_0 n}
\end{aligned}$$

Pilot tones

$$\text{if } \hat{s}[n] = \cos((\omega_c + \omega_0)n):$$

$$\begin{aligned}
\hat{b}[n] &= \mathcal{H}\{\cos((\omega_c + \omega_0)n)\cos(\omega_c n) - j\cos((\omega_c + \omega_0)n)\sin(\omega_c n)\} \\
&= \mathcal{H}\{\cos(\omega_0 n) + \cos((2\omega_c + \omega_0)n) - j\sin((2\omega_c + \omega_0)n) + j\sin(\omega_0 n)\} \\
&= \cos(\omega_0 n) + j\sin(\omega_0 n) \\
&= e^{j\omega_0 n}
\end{aligned}$$

In slow motion

It's a dirty job...

but a receiver has to do it:

- interference
- propagation delay
- linear distortion
- clock drifts

It's a dirty job...

but a receiver has to do it:

- interference
- propagation delay
- linear distortion
- clock drifts

but a receiver has to do it:

- interference
- propagation delay
- linear distortion
- clock drifts

It's a dirty job...

but a receiver has to do it:

- interference
- propagation delay
- linear distortion
- clock drifts

It's a dirty job...

but a receiver has to do it:

- interference $\rightarrow$ handshake and line probing
- propagation delay
- linear distortion
- clock drifts

It's a dirty job...

but a receiver has to do it:

- interference $\rightarrow$ handshake and line probing
- propagation delay $\rightarrow$ delay estimation
- linear distortion
- clock drifts

It's a dirty job...

but a receiver has to do it:

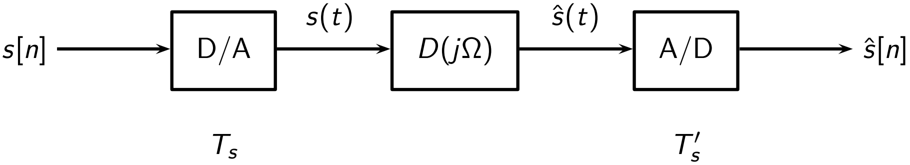
- interference $\rightarrow$ handshake and line probing
- propagation delay $\rightarrow$ delay estimation
- linear distortion $\rightarrow$ adaptive equalization
- clock drifts

It's a dirty job...

but a receiver has to do it:

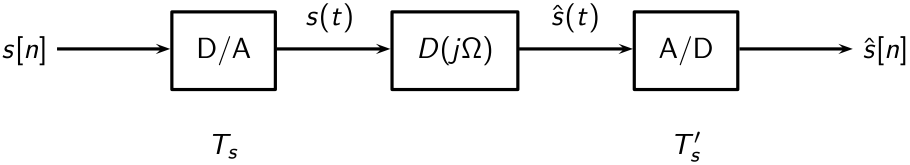
- interference → handshake and line probing
- propagation delay → delay estimation
- linear distortion → adaptive equalization
- clock drifts → timing recovery

The two main problems



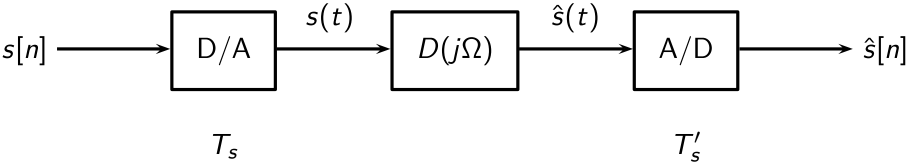
- channel distortion $D(j\Omega)$
- (time-varying) discrepancies in clocks $T_s' = T_s$

The two main problems



- channel distortion $D(j\Omega)$
- (time-varying) discrepancies in clocks $T'_s = T_s$

The two main problems



- channel distortion $D(j\Omega)$
- (time-varying) discrepancies in clocks $T'_s = T_s$

Delay compensation

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

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

Delay compensation

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

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

Delay compensation

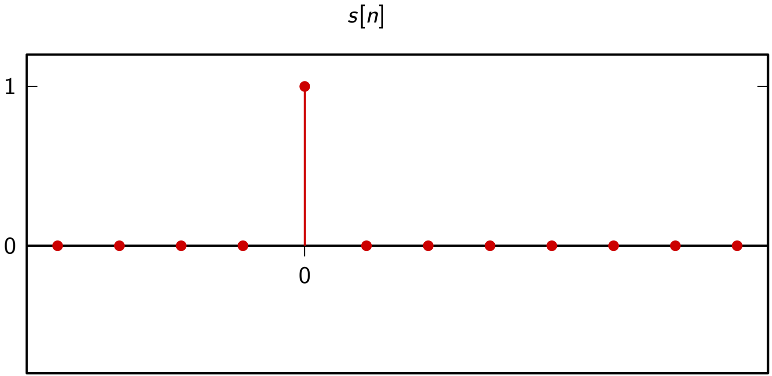
Assume the channel is a simple delay: $\hat{s}(t) = s(t-d) \Rightarrow D(j\Omega) = e^{-j\Omega d}$

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

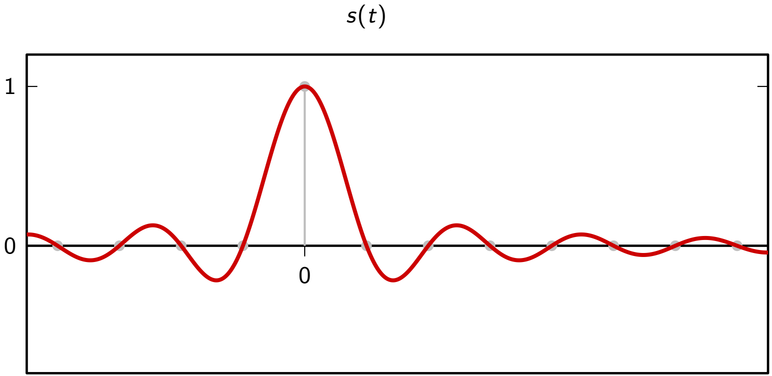
Assume the channel is a simple delay: $\hat{s}(t) = s(t - d) \Rightarrow D(j\Omega) = e^{-j\Omega d}$

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

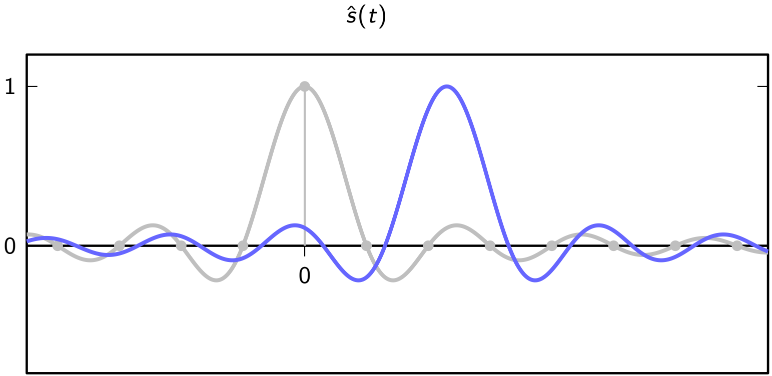
Offsetting the bulk delay ($T_s = 1$)



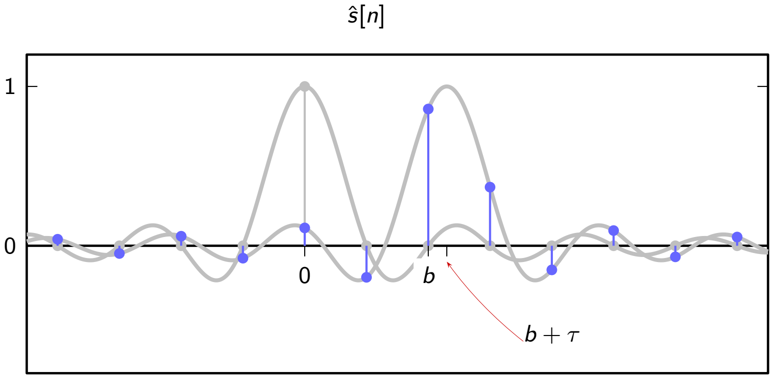
Offsetting the bulk delay ($T_s = 1$)



Offsetting the bulk delay ($T_s = 1$)



Offsetting 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 $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - b - \tau))$
- 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 $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - b - \tau))$
- 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 $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - b - \tau))$
- 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 $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - b - \tau))$
- 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

Compensating for the fractional delay

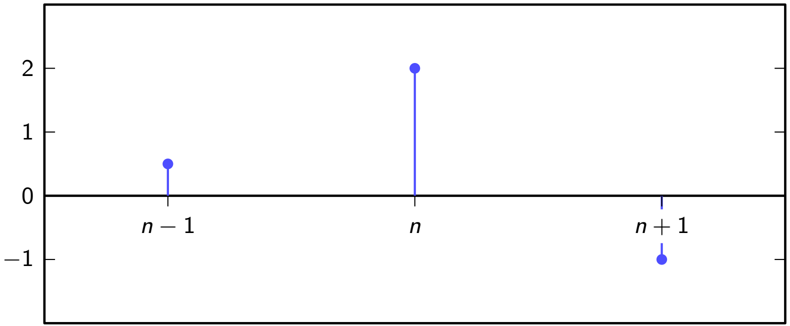
- $\hat{s}[n] = s(n - \tau)T_s$ (after offsetting bulk delay)
- we need to compute subsample values
- in theory, compensate with a sinc fractional delay $h[n] = \text{sinc}(n + \tau)$
- in practice, use local Lagrange approximation

Compensating for the fractional delay

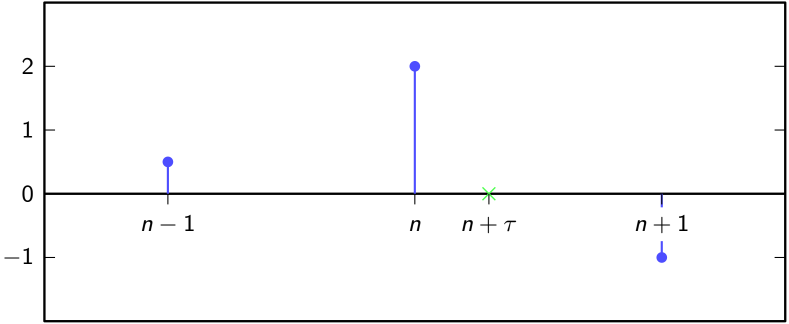
- $\hat{s}[n] = s(n - \tau)T_s$ (after offsetting bulk delay)
- we need to compute subsample values
- in theory, compensate with a sinc fractional delay $h[n] = \mathrm{sinc}(n + \tau)$
- in practice, use local Lagrange approximation

- $\hat{s}[n] = s(n - \tau)T_s$ (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

Compensating for the fractional delay



Compensating for the fractional delay



Lagrange approximation (see Module 6.2)

as per usual, choose $T_s = 1$

- we want to compute $x(n+\tau)$, with $|\tau| < 1/2$
- local Lagrange approximation around n

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

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

- $x(n+\tau) \approx x_L(n;\tau)$

Lagrange approximation (see Module 6.2)

as per usual, choose $T_s = 1$

- we want to compute $x(n+\tau)$, with $|\tau| < 1/2$
- local Lagrange approximation around n

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

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

- $x(n+\tau) \approx x_L(n;\tau)$

Lagrange approximation (see Module 6.2)

as per usual, choose $T_s = 1$

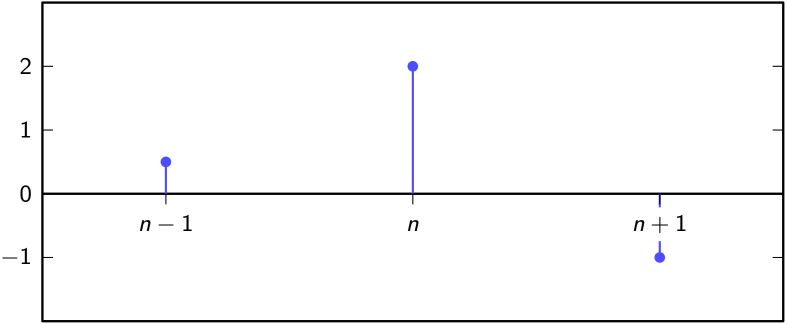
- we want to compute $x(n+\tau)$, with $|\tau| < 1/2$
- local Lagrange approximation around n

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

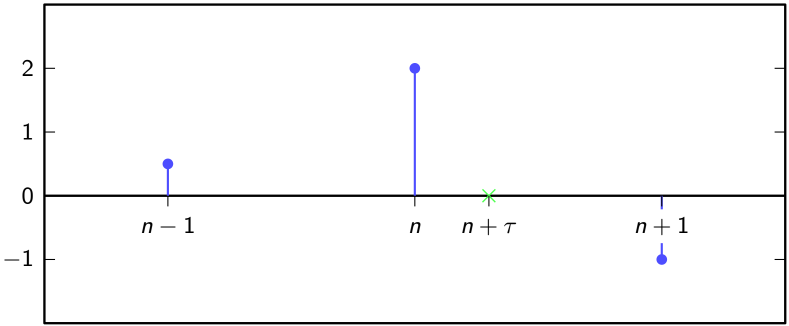
$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq k}}^{N} \frac{t-i}{k-i} \qquad\qquad k = -N, \ldots, N$$

- $x(n+\tau) \approx x_L(n;\tau)$

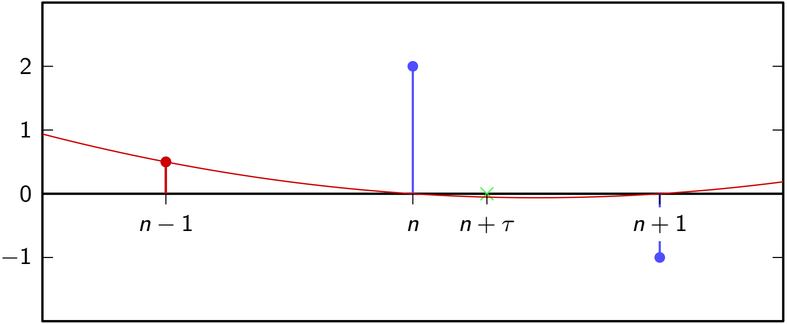
Lagrange interpolation ($N = 1$)



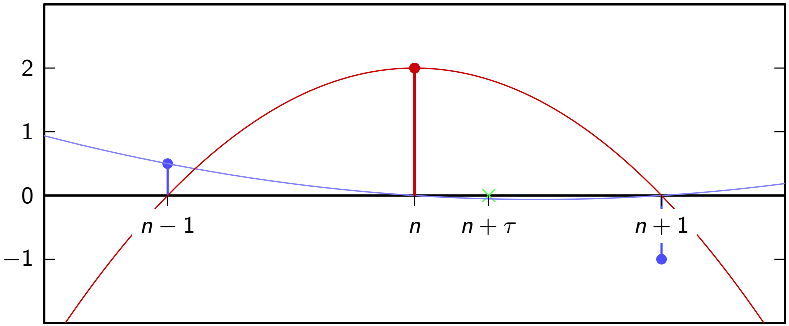
Lagrange interpolation ($N = 1$)



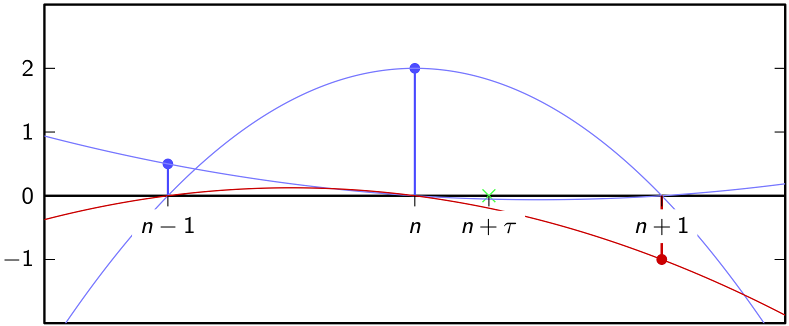
Lagrange interpolation ($N = 1$)



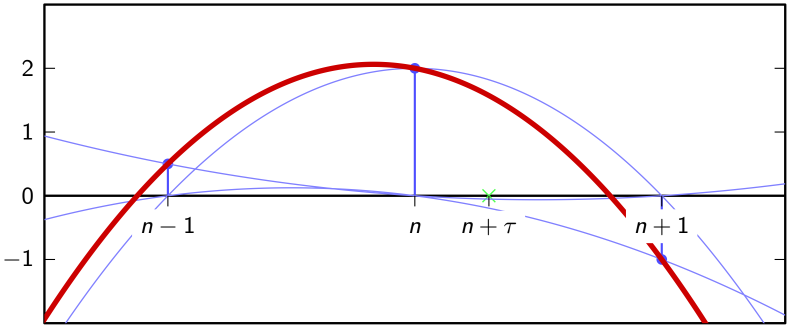
Lagrange interpolation ($N = 1$)



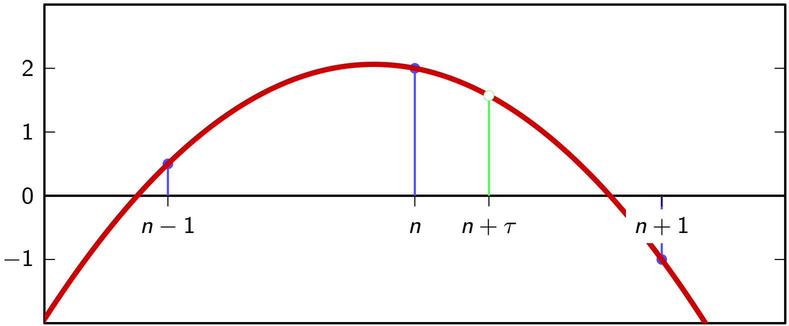
Lagrange interpolation ($N = 1$)



Lagrange interpolation ($N = 1$)



Lagrange interpolation ($N = 1$)



Lagrange interpolation as an FIR

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

Lagrange interpolation as an FIR

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

Lagrange interpolation as an FIR

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

Lagrange interpolation as an FIR

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

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

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

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

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

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

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

Delay compensation algorithm

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

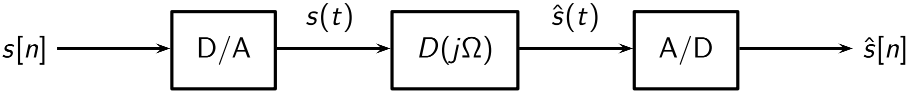
Delay compensation algorithm

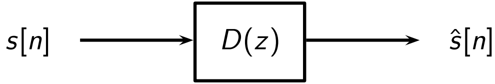
- estimate the delay τ
- compute the $2N+1$ Lagrangian coefficients
- filter with the resulting FIR

Delay compensation algorithm

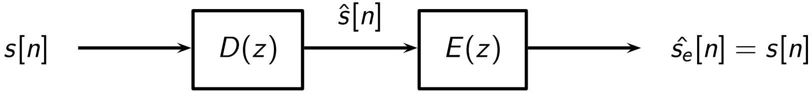
- estimate the delay τ
- compute the $2N + 1$ Lagrangian coefficients
- filter with the resulting FIR

Compensating for the distortion




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

Example: adaptive equalization



Example: adaptive equalization

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

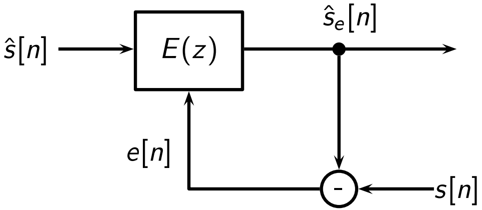
Example: adaptive equalization

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

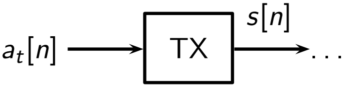
Example: adaptive equalization

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

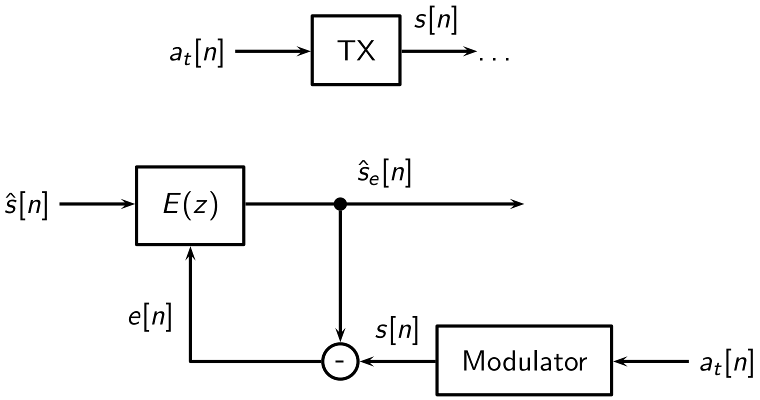
Adaptive equalization



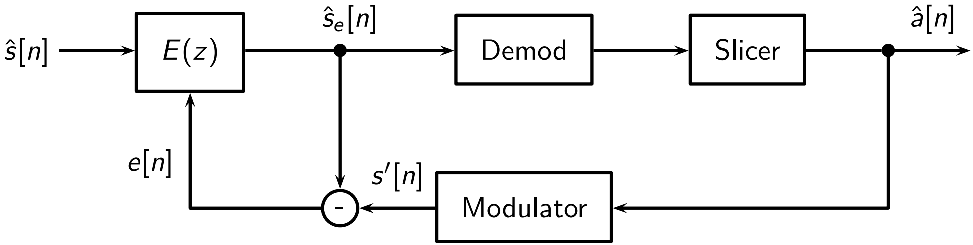
Adaptive equalization: bootstrapping via a training sequence



Adaptive equalization: bootstrapping via a training sequence



Adaptive equalization: online mode



So much more to do...

- how do we perform the adaptation of the coefficients?
- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

adaptive signal processing

So much more to do...

- how do we perform the adaptation of the coefficients?
- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

adaptive signal processing

So much more to do...

- how do we perform the adaptation of the coefficients?
- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

adaptive signal processing

So much more to do...

- how do we perform the adaptation of the coefficients?
- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

adaptive signal processing

So much more to do...

- how do we perform the adaptation of the coefficients?
- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

adaptive signal processing

So much more to do...

- how do we perform the adaptation of the coefficients?
- how do we compensate for differences in clocks?
- how do we recover from interference?
- how do we improve resilience to noise?

adaptive signal processing

END OF MODULE 9.5

Digital Signal Processing

Module 9.6: ADSL

Overview:

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

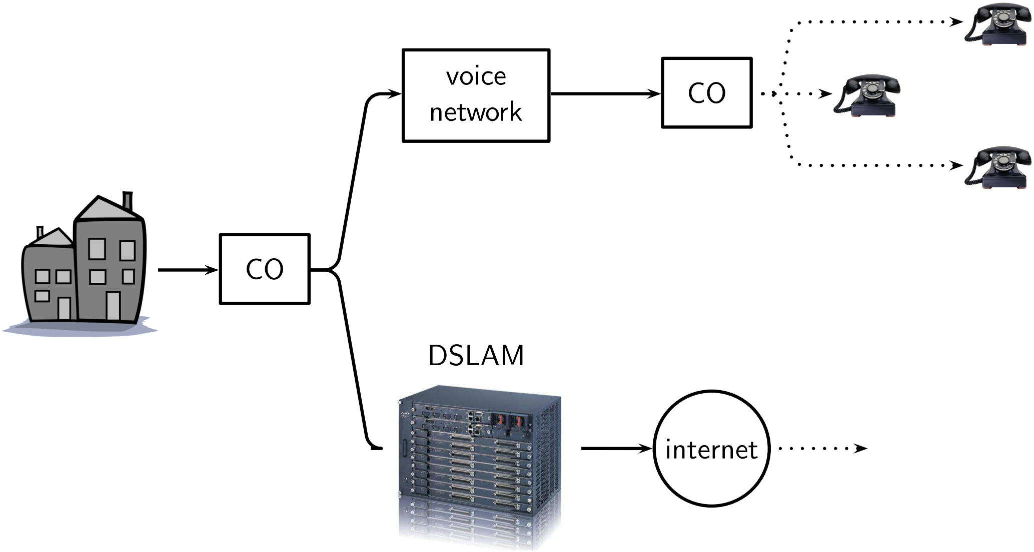
Overview:

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

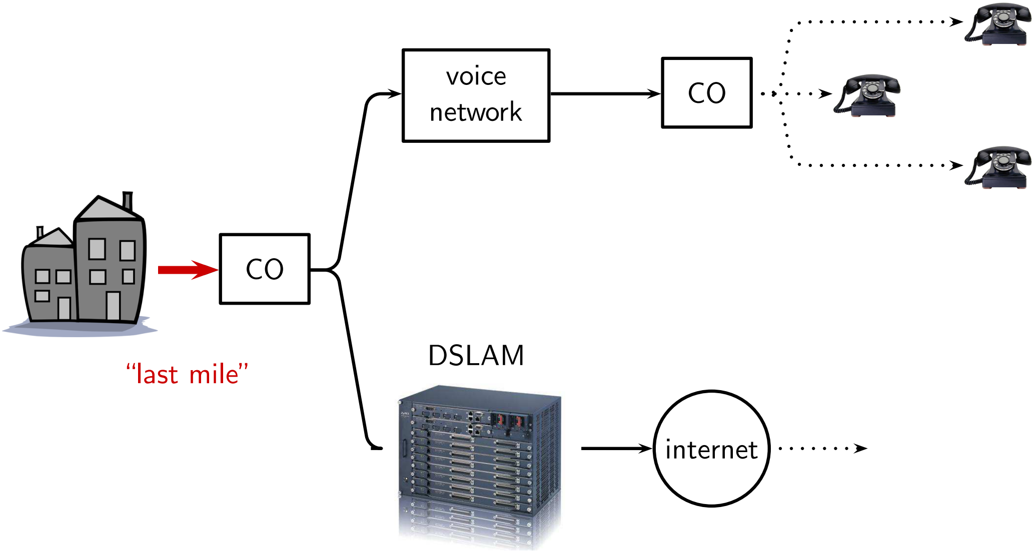
Overview:

- 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.

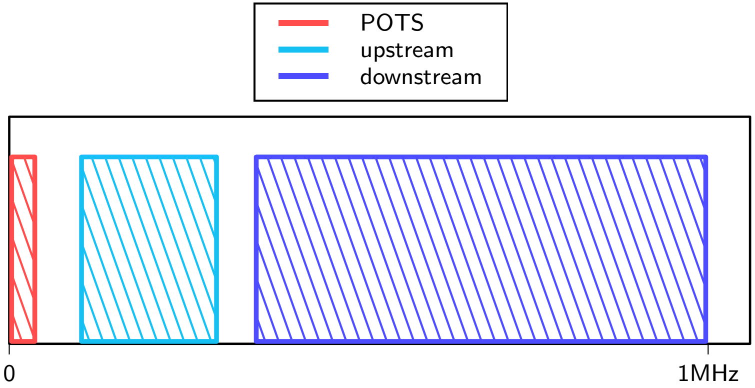
The last mile

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

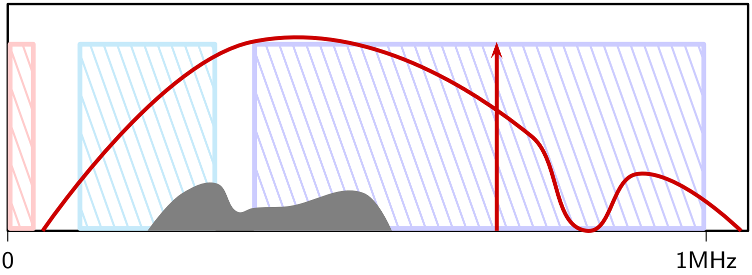
The last mile

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

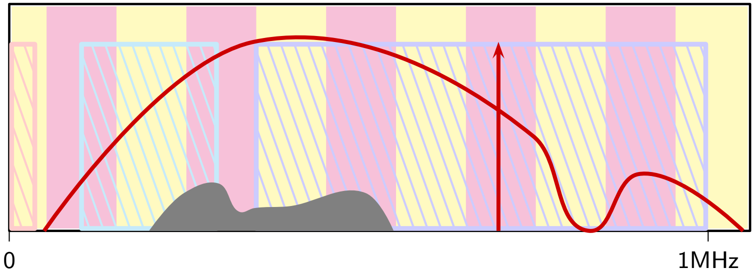
The ADSL channel



The ADSL channel



Idea: split the band into independent subchannels



Subchannel structure

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

The digital design

- pick $F_s = 2F_{\max}$ ($F_{\max}$ is high now!)
- center frequency for each subchannel $\omega_k = 2\pi \frac{kF_{\max}/N}{F_s} = \frac{2\pi}{2N}k$
- bandwidth of each subchannel $\frac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K \geq 2N$

The digital design

- pick $F_s = 2F_{\text{max}}$ (F_{max} is high now!)
- center frequency for each subchannel $\omega_k = 2\pi \frac{kF_{\text{max}}/N}{F_s} = \frac{2\pi}{2N}k$
- bandwidth of each subchannel $\frac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K \geq 2N$

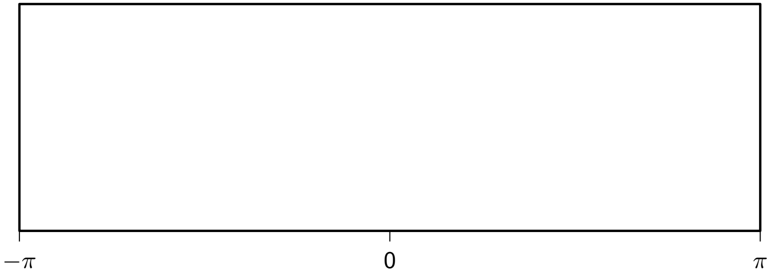
The digital design

- pick $F_s = 2F_{\max}$ ($F_{\max}$ is high now!)
- center frequency for each subchannel $\omega_k = 2\pi \frac{kF_{\max}/N}{F_s} = \frac{2\pi}{2N}k$
- bandwidth of each subchannel $\frac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K \geq 2N$

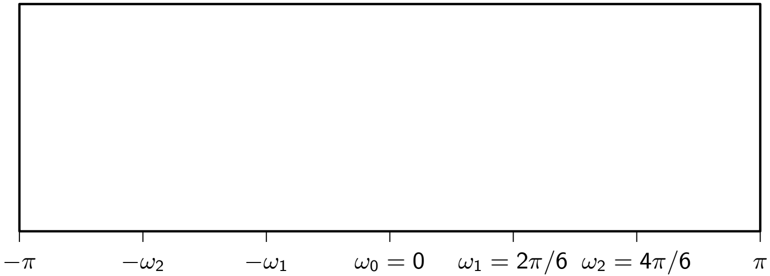
The digital design

- pick $F_s = 2F_{\text{max}}$ (F_{max} is high now!)
- center frequency for each subchannel $\omega_k = 2\pi \frac{kF_{\text{max}}/N}{F_s} = \frac{2\pi}{2N}k$
- bandwidth of each subchannel $\frac{2\pi}{2N}$
- to send symbols over a subchannel: upsampling factor $K \geq 2N$

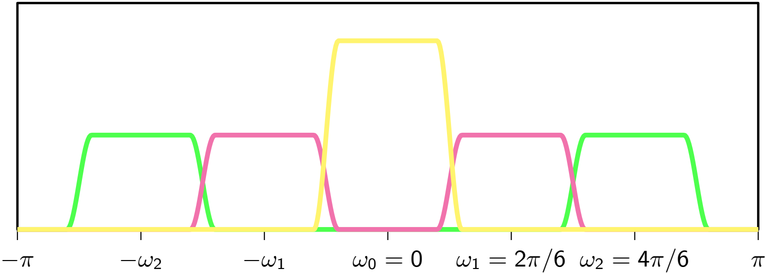
The digital design ($N = 3$)



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

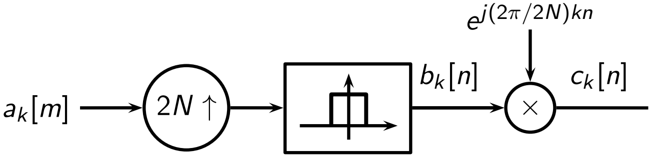
The digital design

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

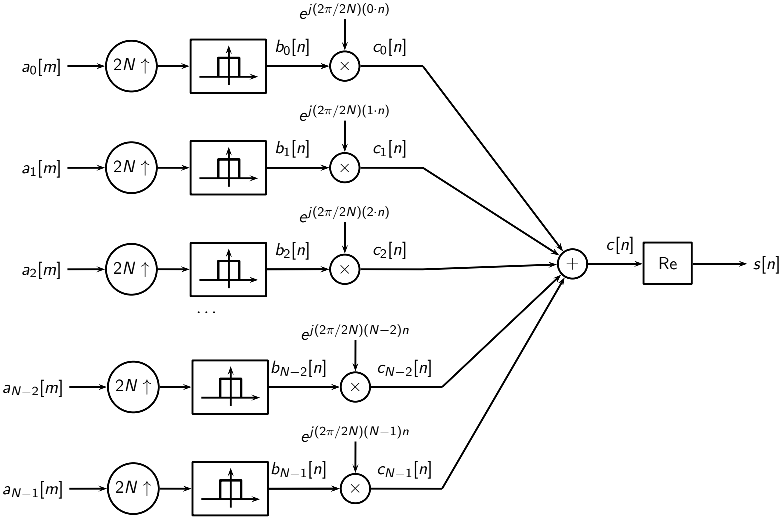
The digital design

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

The subchannel modem

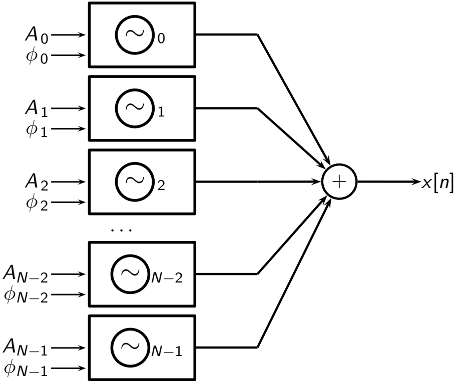


The bank of modems



If it looks familiar...

check back Module 4.3, 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

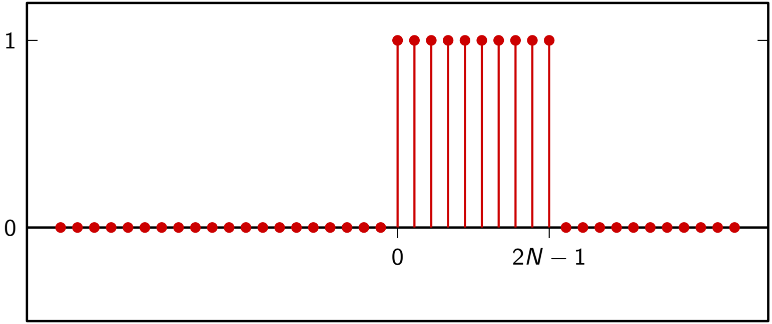
- we will show that transmission can be implemented efficiently via an IFFT
- Discrete Multitone Modulation

The great ADSL trick

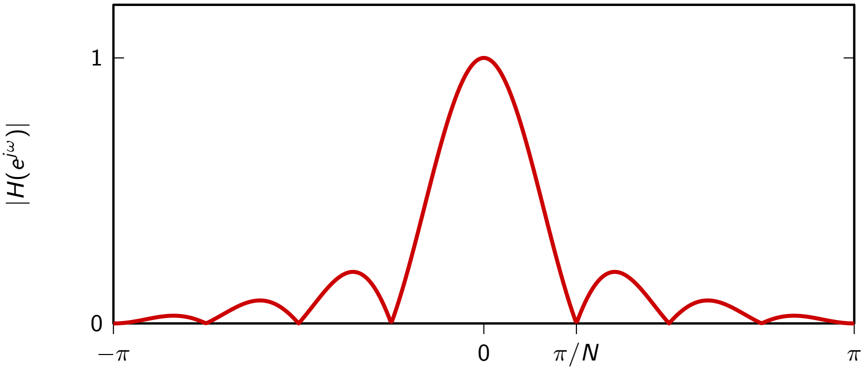
instead of using a good lowpass filter, use the $2N$-tap interval indicator:

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

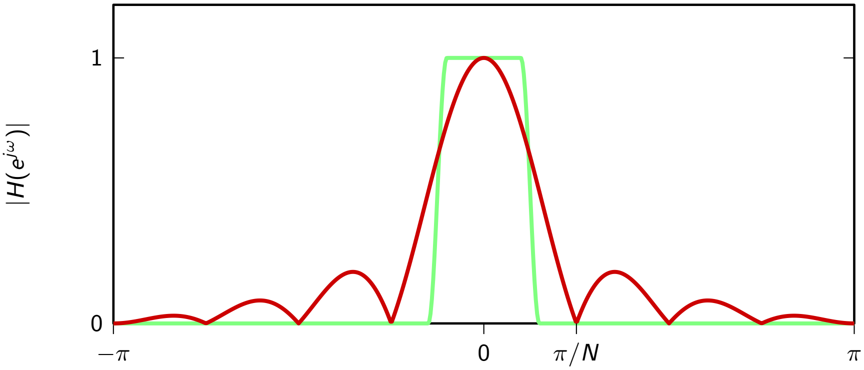
Interval indicator signal (Module 4.7)



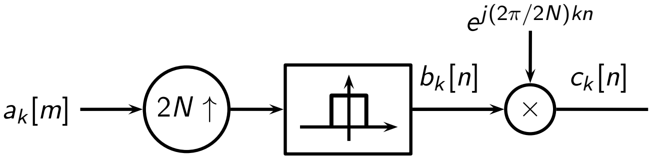
DTFT of interval signal (Module 4.7)



DTFT of interval signal (Module 4.7)



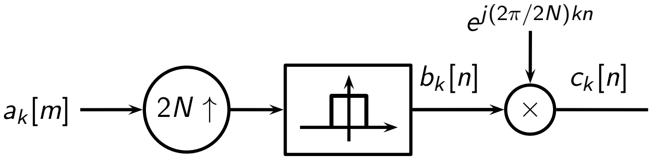
Back to the subchannel modem



rate: B symbols/sec

$2NB$ samples/sec

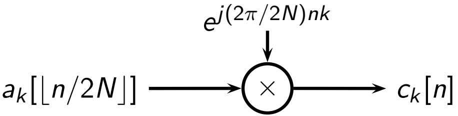
Back to the subchannel modem



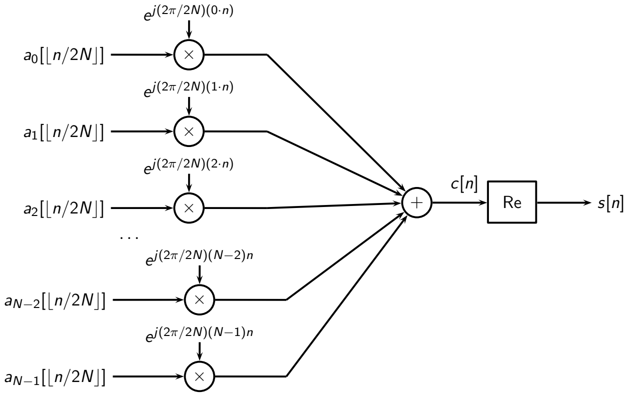
rate: B symbols/sec

$2NB$ samples/sec

by using the indicator function as a lowpass:



The bank of modems, revisited



The complex output signal

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

$$= 2N \cdot \mathrm{IDFT}_{2N}\left\{\begin{bmatrix} a_0[m] & a_1[m] & \ldots & a_{N-1}[m] & 0 & 0 & \ldots & 0 \end{bmatrix}\right\}[n]$$

$$(m = \lfloor n/2N \rfloor)$$

The complex output signal

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

$$= 2N \cdot \mathrm{IDFT}_{2N}\left\{\begin{bmatrix} a_0[m] & a_1[m] & \ldots & a_{N-1}[m] & 0 & 0 & \ldots & 0 \end{bmatrix}\right\}[n]$$

$$(m = \lfloor n/2N \rfloor)$$

We can do even better!

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

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

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

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

We can do even better!

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

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

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

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

We can do even better!

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

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

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

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

We can do even better!

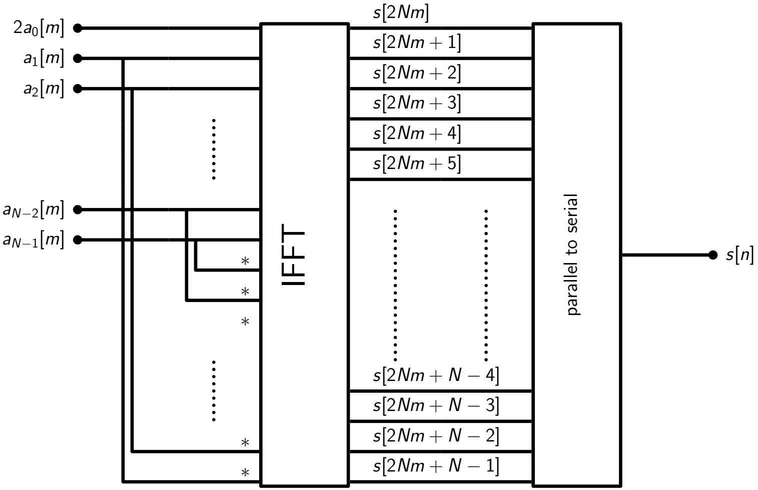
- we are interested in $s[n] = \mathrm{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- it is easy to prove (exercise) that:

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

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

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

ADSL transmitter



ADSL specs

- $F_{\text{max}} = 1104\text{KHz}$
- $N = 256$
- 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: 14.9Mbps (downstream)

ADSL specs

- $F_{\text{max}} = 1104\text{KHz}$
- $N = 256$
- 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: 14.9Mbps (downstream)

ADSL specs

- $F_{\max} = 1104\text{KHz}$
- $N = 256$
- 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: 14.9Mbps (downstream)

ADSL specs

- $F_{\max} = 1104\text{KHz}$
- $N = 256$
- 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: 14.9Mbps (downstream)

ADSL specs

- $F_{\max} = 1104\text{KHz}$
- $N = 256$
- 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: 14.9Mbps (downstream)

ADSL specs

- $F_{\text{max}} = 1104\text{KHz}$
- $N = 256$
- 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: 14.9Mbps (downstream)

END OF MODULE 9.6

END OF MODULE 9