Contents

1 Week 8 Module 6:

1.1 Digital Communication Systems

1.1.1 Introduction to digital communications

1. The success factors for digital communications
   (a) Power of the DSP paradigm
      - integers are easy to **regnerate**
      - good phase control
      - adaptive algorithms

   (b) Algorithmic nature of DSP is a perfect match with information therory:
      - Image Coding: JPEG's entropy coding
      - Encoding of of accustic or video informatiion: CD's and DVD's error correction
      - Communication Systems: trellis-coded modulation and Viterbi-code modulation

   (c) Hardware advancement
      - minituarization

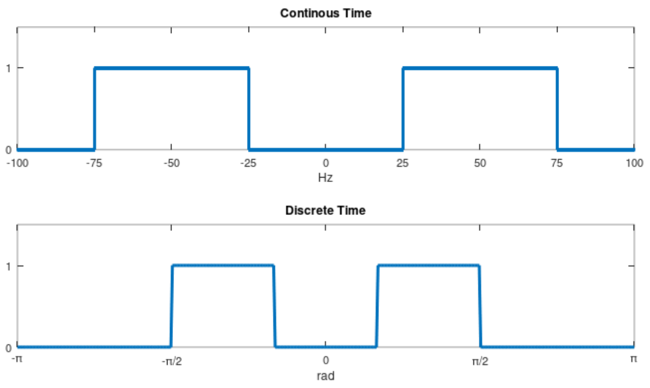
- general-purpose platforms
- power efficiency

2. The analog channel constraints

   - unescapable "limits" of physical channels:
     - **Bandwith:** the signal that can be send over a channel has a limited frequency band
     - **Power:** the signal has limited power over this band, e.g. due to power limit of the equipment
   - Both constraints will affect the final capacity of the channel.
   - The maximum amound of information that can be reliably delivered over a channel - bits per second -
   - Bandwidth vs. capacity:
     - small sampling period $T_s \Rightarrow$ high capacity
     - but the bandwidth signal grows as $\dfrac{1}{T_s} \Rightarrow \Omega_N = \dfrac{\pi}{T_s}$

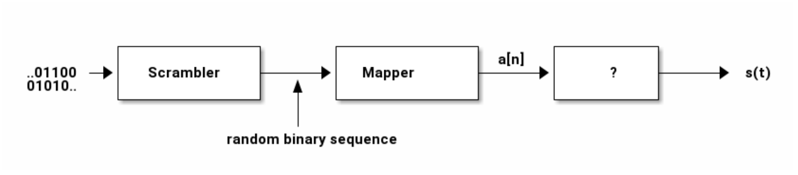
3. The design Problem

   - We are going to adapt the all-digital paradigm
   - Converting the specs to digital design

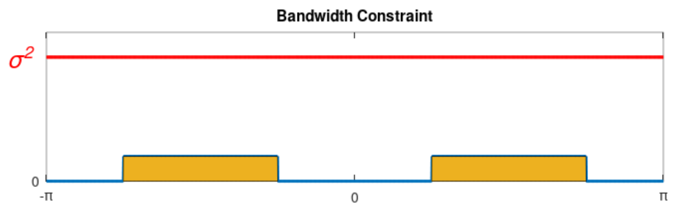


   - with:
     - Sampling Frequency $F_S \geq 2f_{max}$
     - Continuous Time $F_s/2$: Nyquist frequency

  - Maximum Frequency: $\frac{F_s}{2} \Rightarrow \pi$
  - Bandwidth: $\omega_{min,max} 2\pi \frac{f_{min,max}}{F_s}$
- Transmitter design
  - convert a bitstream into a sequence of symbols a[n] via a mapper
  - model a[n] as white random sequence $\Rightarrow$ add a scrambler
  - no need to convert a[n] into a continuous-time signal within the constraints



If we assume that the data is randomized and therefore the symbol sequence is a white sequence, we know that the power spectral density is simply equal to the variance. And so the power of the signal will be constant over the entire frequency band. But we actually need to fit it into the small band here as specified by the bandwidth constraint.



So how do we do this? Well, in order to do that, we need to introduce a new technique called upsampling, and we will see this in the next module.

We are talking about digital communication systems and in this lesson we will talk about how to fulfill the bandwidth constraint. The way we are going to do this is by introducing an operation called upsampling and we will see how upsampling allows us to fit the spectrum generated by the transmitter onto the band allowed for by the channel.

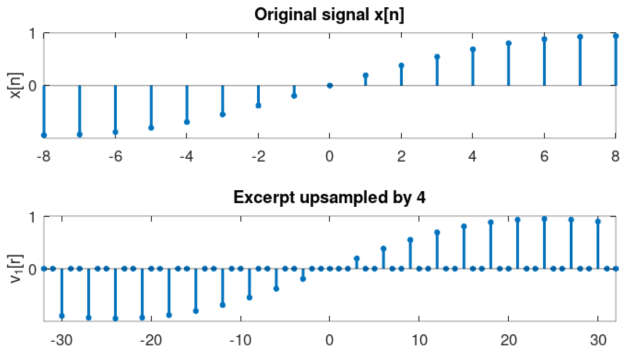
1.1.2 Controlling the bandwidth

Shaping the bandwidth Remember that our assumption is that the signal generated by the transmitter is a wide sequence and therefore its power spectral density will be constant and full band. In order to meet the bandwith constraint, we need to shrink the support of the power spectral density.

- the answer is multirate techniques

1. Upsampling

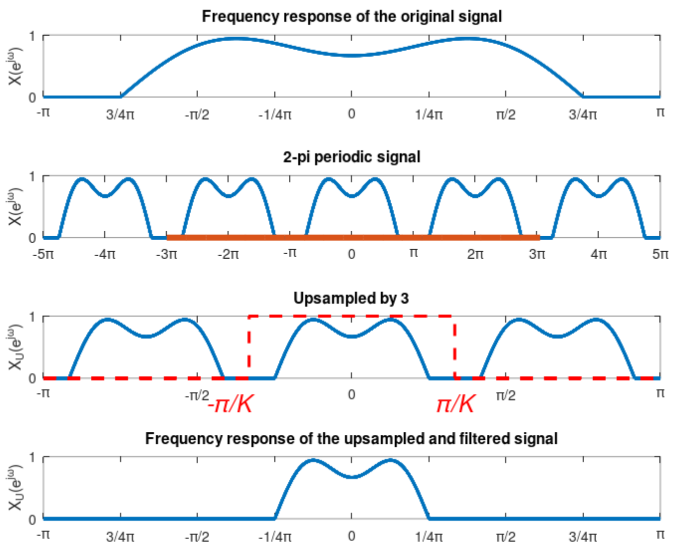
   - Our Problem
   - bandwith constraint requires us to control the spectral support of a signal
     - we need to be able to shrink the support of a full-band signal
   - Upsampling can be obtained by interpolating a discrete time sequence to get a continuous time signal. And resample the signal with a sampling period which is k-times smaller than the original interpolation sample.
   - Or we do it entirly digitally.
     (a) we need to "increase" the number of samples by k
     (b) obviously $x_U[m] = x[n]$ when m multiple of K
     (c) for lack of better strategy, put zeros elsewhere
   - Upsampling in the time domain



   - Upsampling in the digital domain: Frequency Domain

$$\begin{aligned} X_U(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} x_U[m]e^{-j\omega m} \text{ with } x_U = 0 \text{ if } m \neq nK \\ &= \sum_{m=-\infty}^{\infty} x[n]e^{-j\omega nK} \\ &= X(e^{j\omega K}) \end{aligned}$$

This is simply a scaling of the frequency axis by a factor of K. Graphical interpretation: since we are multiplying the frequency axis by a factor of K, there will be a shrinkage of the frequency axis.



- $\dfrac{\pi}{K}$: Filter Cut-Off Frequency
- The bandwidth of the signal was shrinked by factor K=3: from $\dfrac{3}{4}\pi$ to $\dfrac{1}{4}\pi$
- Upsampling in the digital domain: Time Domain
  (a) insert K-1 zeros after every sample
  (b) ideal lowpass filtering with $\omega_c => \dfrac{\pi}{K}$

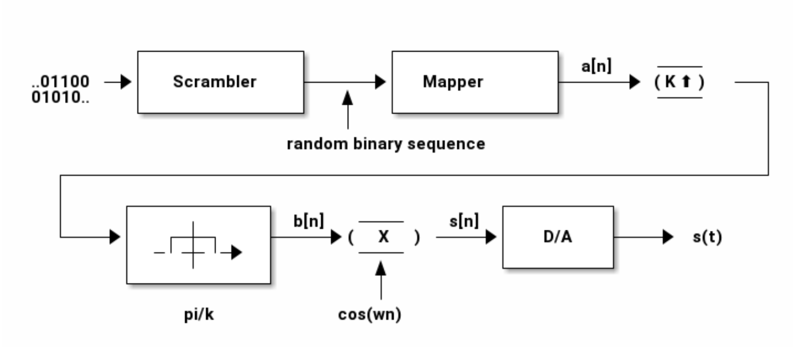
$$\begin{aligned} x[n] &= x_U(n) * sinc(n/K) \\ &= x_U[i] sinc\left(\frac{n-i}{K}\right) \\ &= x[m] sinc\left(\frac{n}{K} - m\right), \text{ with } i = mK \end{aligned}$$

Which is exactely the same formula when using an interpolator and a sampler.

2. Fitting the transmitter spectrum The bandwidth constraint says that only frequencies between F_{min} and F_{max} are allowed. To translate it to the digital domain, follow the preceeding steps:
   - let $W = F_{max} - F_{min}$
   - pick F_s so that:
     - $F_s > 2F_{max}$
     - $F_s = KW$, $k \in \mathbb{N}$
   - $\omega_{max} - \omega_{min} = 2\pi \frac{W}{F_s} = \frac{2\pi}{K}$
   - we can simply upsample by K

Bandwith constrainth

And so we can simply upsample the sample sequence by K, so that its bandwidth will move from 2pi to 2pi/K, and therefore, its width will fit on the band allowed by the channel.



Scrambler Randomizes the data, ensures the the resulting bit-stream is equiprobable.

Mapper Segments the bit-stream into consecutive groups of M bits. And this bits select one of 2^M possible signaling values. The set of all possible signaling values is called the alphabet.

a[n] The actual discrete-time signal. The sequence of symbols to be transmitted.

K The uppsampler narrows the spectral occupancy of the symbol sequence. The following low pass filter is known as the shaper, since it determines the time domain shape of the transmitted symbols.

b[n] The baseband signal.

s[n] The passand signal. $s[n] = Re\{c[n]\} = Re\{b[n]e^{j\omega_c n}\}$ The signal which is fed to the D/A converter is simply the real part of the complex bandpass signal. With $\omega_c = \frac{\omega_{max} - \omega_{min}}{2}$

Data Rates

- up-sampling does not change the data rate
- we produce (and transmitt) W symbols per seconds
- W is sometimes called the Baud Rate of the system and is equal to the available bandwidth.

Raised Cosine

1.1.3 Controlling the power

1. Noise and probability of error

   - Transmitter reliability
     - transmitter sends a sequence of symbols a[n]
     - receiver obtains 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 with respect to the power of the signal

- decoding strategy
- alphabet of transmission symbols

(a) Signaling alphabets

- we have a (randomized) bitstream coming in
- we want to send some up-sampled and interpolated samples over the channel
- how do we get from bit-stream to samples: How does the mapper works
- mapper:
  - split incoming bitstream into chunks
  - assign a symbol a[n] from a finite alphabet A to each chunk.
- slicer:
  - receive a value $\hat{a}$[n]
  - decide which symbol from A is "closest" to $\hat{a}[n]$

(b) Example: two-level signaling

- mapper:
  - split incoming bitstream into **single bits**
  - a[n] = G of bit is 1, a[n] = -G if bit is 0
- slicer:

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

- Hypothesis With the following hypothsis we can calculate the probability of error:
  - $\hat{a}[n] = a[n]Q\eta[n]$
  - bits in bitstream are equiprobable: zero and one appear with probability 50% each
  - noise and signal are indepenedent
  - noise is additive white Gaussian noise zero mean and variance σ_0
- Porbability Error

$$\begin{aligned}
P_{err} &= P\left[\eta[n] < -G \middle| \text{ n-th bit is 1}\right] + P\left[\eta[n] > G \middle| \text{ n-th bit is 0}\right] \\
&= \left(P\left[\eta[n] < -G\right] + P\left[\eta[n] > G\right]\right)/2 \\
&= P\left[\eta[n] > G\right] \\
&= \int_G^\infty \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau, \text{ with the PDF for the Gaussian Distribution with the kno} \\
&= erfc(G/\sigma_0), \text{ Numerical Packages: The Error Function}
\end{aligned}$$

Error Function erfc Integral from G to inifity of the PDF for the guassion distribution with the known variance $_0$. As available in most numerical packages

Probability Error

Is some function of the ratio between the amplitude of the signal and the standard deviation of the noise.

- transmitted power

$$\begin{aligned}
\sigma^2 &= G^2 P\left[\text{ n-th bit is 1}\right] + G^2 P\left[\text{ n-th bit is 0}\right] \\
&= G^2
\end{aligned}$$

Probability of Error

$$P_{err} = erfc(\sigma_s/\sigma_0) = erfc(sqrtSNR)$$

And since we are in a log log scale, we can see that the probability of error decays exponentially with the signal to noise ratio. Absolute rate of decay might change, in terms of the linear constants involved in the curve. The trend will stay the same, even for more complex signaling schemes

(c) 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.

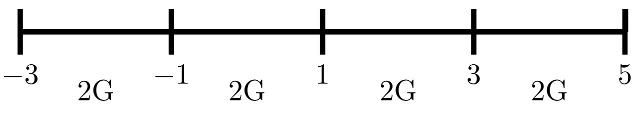
2. Multilevel signaling

- binary signaling is not very efficient (one bit at a time)
- to increase the throughput we can use multilevel signaling

the general idea We take now larger chunks of bits, and therefore, we have alphabets, that have a higher cardinality. So more values in the alphabet, means more bits per symbol, and therefore a higher data rate. But not to give the ending away, we will see that the power of the signal will also be dependent of the size of the alphabet, and so in order not to exceed the certain probability of error, given the channel's power of constraint. We will not be able to grow the alphabet indefinitely, but we can be smart in the way we build this alphabet. And so we will look at some examples.

(a) Pulse Ampltitude Modulation PAM

- mapper:
  - split incoming bitstream into chunks of M bits
  - chunks define a sequence of integers k[n] ∈ {0,1,2..2^{M}-1}
  - a[n] = G((-2^{M} +1) + 2k[n]) odd integers around zero
    - with M=2 and G=1: a[n]=-3.-1, 1, 3
- slicer:
  - $a'[n] = argmin[|\hat{a}[n] - a|]$



- PAM with M=2, G=1
- distance between points is 2G
- using odd integers creates a zero-mean sequence
- probability error analysis for PAM is analog the lines of binary signaling
- can we increase the throughput of PAM even further
- here's a wild idea, let's use complex numbers

(b) Quadratur Amplitude Modulation QAM

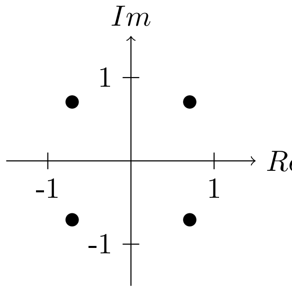
- mapper:
  - split incoming bitstream into chunks of M bits, M even

    - 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] Aj$_i$[n])
- slicer:
    - \$a'[n] = arg min[$|\hat{a}[n] - a|$]

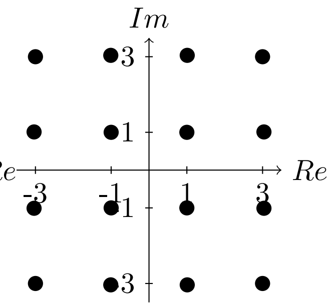
So the transition alphabet a, is given by points in the complex plane with odd valued coordinates around the origins. The receiver deslicer works by finding the symbol in the alphabet which is closest in Euclidian distance to the relieved symbol.

- Some QAM Constellations with G=1

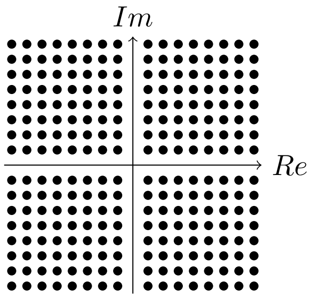
M=2



M=4



M=4



3. Summery In our communication system design problem, we introduced a mapper. We did not say much about this operation so far. We only said that this block maps the scrambled input into a sequence of symbols belonging to a certain alphabet. In this lesson, we have explored in greater details this operation and how it is related to the problem of satisfying the power constraints.

   The sequence received at the receiver inevitably contains some form of noise. For each symbol, if the noise level is high, the receiver wrongly interpret the symbol for another one in the alphabet. It makes a decoding error. The probability of decoding error depends on three factors

   the signal-to-noise ratio, i.e., the power of the signal with respect to the power of the noise, SNR is expressed in dB. Through SNR, the power constraints of the channel enters in the design problem and we cannot operate at an arbitrarily high SNR

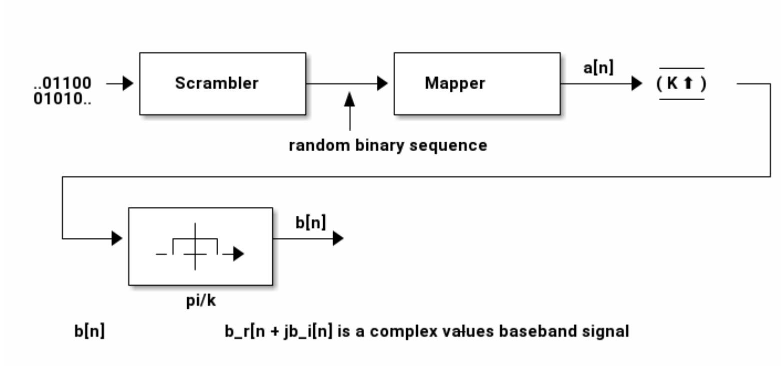
   the decoding strategy, i.e., how smart we are at circumventing the effect of noise the choice of alphabet. If we increase the size of the alphabet, we can transmit more information per symbol but symbols are closer in the alphabet and the probability of error increases.

We have also studied two encoding schemes, pulse amplitude modulation (PAM) and quadrature amplitude modulation (QAM) and analyzed their probability of error. In the case of QAM, the choice of constellation size MMM can be picked as to match a desired probability of error and SNR imposed by the channel constraints. The final throughput of the system is MWMWMW. Our analysis of QAM is based on the assumption that we transmit complex numbers over a real channel. How to do this in practice is the topic of the next lesson.

1.1.4 Modulation and Demodulation

1. Intrdoduction Welcome to Lesson 6.4 of Digital Signal Processing. In the previous module, we saw an interesting signaling scheme that allows to increase the data rate while keeping the same probability of error for a given power constraint. The problem is that communication alphabet that we devised is complex-valued whereas we know that physical channel can only handle real values. So in this lesson, we will see how to transmit and recover a complex-valued symbol stream over a real-valued channel. We will follow this with a concrete design example for the telephone channel and finally we will compare the performance of the system with the ultimate in data rate that is given us by the channel capacity formula.

2. Modulation and Demodulation



So let's review where we stand in terms of transmitter design. We have the user's bitstream that comes into the system. This is sent through a scrambler that makes sure that the resulting bitstream is equiprobable. The mapper will split the bitstream into m-bit chunks. And each chunk will be associated to a complex-valued symbol. This will create a complex value sequence a[n]. And to fit that over the bandwidth prescribed by the channel, we have to upsample it, which means inserting K minus 1 0's after each symbol of the sequence and then low passing the sequence with a filter with cutoff frequency pi over K.

(a) The passband signal

The Passband Signal

To transmit complex value over the real channel, we first modulate the signal b[n] with the frequency at the carrier frequency and take the real part of the passband signal.

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

So we have a cosine carrier, and a sine carrier. Now, these two carriers are orthogonal because they're shifted by a phase of 90 degrees. Now, when two things are 90 degrees apart, they're said to be in quadrature

$b_r[n]cos\omega_c n$	In phase Part
$b_i[n]sin\omega_c n$	Quadrature Part

(b) **TODO** Complex baseband signal Spectrum

(c) Recovering the baseband signal let^s try the usual method (multiplying by the carrier, see Module 5.5.)

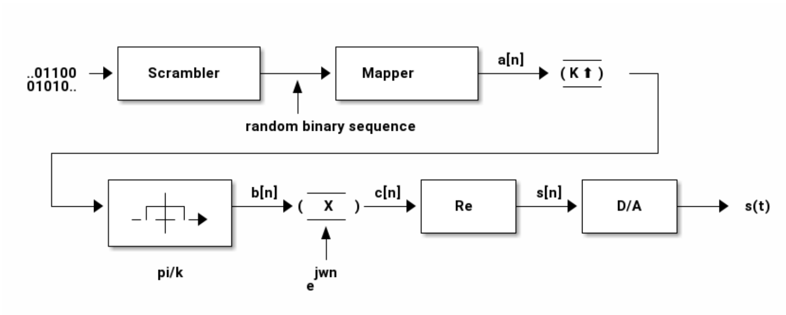
Real Part:

$$\begin{aligned} s[n]cos\omega_c n &= b_r[n]cos^2\omega_c n - b_i[n]sin\omega_c n \\ &= b_r[n]\frac{1+cos2\omega_c n}{2} - b_i[n]\frac{sin2\omega_c n}{2} \\ &= \frac{1}{2}b_r[n] + \frac{1}{2}(b_r[n]cos2\omega_c n - b_i[n]sin2\omega_c n) \end{aligned}$$

To get rid of the spurios components we need to low pass filter the so received signal. We have a matched filter confiugration if we use the same low-pass filter at the receiver side as we have used at the transmitter side.
Quadrature Part:

$$\begin{aligned} s[n]sin\omega_c n &= b_r[n]cos\omega_c n - b_i[n]sin^2\omega_c n \\ &= \frac{1}{2}b_r[n] + \frac{1}{2}(b_r[n]sin2\omega_c n - b_i[n]cos2\omega_c n) \end{aligned}$$

3. Design Example



(a) **TODO** Scetch the QAM Receiver
(b) Example: the V.32 voiceband modem
  - Bandwith Constraint
    - analog telephone channel: F_{min} = 4500 Hz, F_{max} = 2850 Hz
    - usable bandwith: W = 2400Hz, center frequency F_{c} = 1650Hz

    - pick: $F_s = 3 \cdot 2400Hz = 7200Hz$ so that K = 3
    - $\omega_c = 0.458\pi$
  - Power Constraint:
    - maximum SNR: 22dB (telephone line)
    - $P_{err} = 10^{-6}$
    - using QAM, we find the size of the alphabet resp. bits per symbol

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

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

  Final data rate is WM = 9600 bits pers seconds

  - WM: Baude Rate × bits per symbols = 2400Hz × 4
  - Baude Rate: identical with bandwith

(c) 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 ≈ 175000 bps
  - the gap can be narrowed by more advanced coding techniques

1.1.5 Receiver Design

- What is goining on by the sound made by a V.34 modem, while connecting to the internet?

1. **TODO** Draw the receiver concept

   - Receiver has to cope with four potential sources of problem:
     - interference ⇒ handshake and line porbing
     - propagation delay ⇒ delay estimation procedure
     - linear distortion ⇒ adaptive equalization techniques

    - clock dirfts ⇒ timing recovery

- The 2 main problems

2. **TODO** Draw the chain... .... of events that occur between the transmission of the original digital signal and the beginning of the demodulation of the received signal, we have a digital to analog converter at the transmitter, this is the transmitter part of the chain

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

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

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

    Offsetting the bulk delay ($T_s = 1$) The bulk delay is determined by sending out an impulse δ[n] over the channel. The D/A converter will output a contious time signal that looks like a sink (interpolator function).

4. **TODO** Draw the sink propagation on the channel. The bulk delay is just the maximum value in the sequence of samples. Because of the interpolator function (sync) we know the real maximum is half a sample in either direction of the location of the maximum sample value.

    - So at the receiver to offset the bulk delay we will just set the nominal time n=0, to coincide with the location of the maximum value of the sample sequence.

    Remark Of course we're not using impulses to offset the bulk delay. Because impulses are fullband signals, and so they would be filtered out by the passband characteristic of the channel. The trick is to embed these continuities in pilot tones, and to recognize these discontinuities at the receiver. As we have seen in the animation at the beginning of this module, We use phaser reversals which are abrupt discontinuities in sinusoid to provide a recognizable instant in time for the receiver to latch on.

Estimate Fractional Delay For the fractional delay we use a sinusoid instead of a delta, so we build a baseband signal which is simply complex exponential at a known frequency:

- transmit b[b] = $\mathrm{e}^{\mathrm{j}\omega_0\ \mathrm{n}}$ (i.e. s[n] = cos((ω_c + ω_0)n))
- receive $\hat{s}[n]$ = cos((ω_c + ω_0)(n -b - τ))
- after demodulation and bulk delay offset $\hat{b}[n] = e^{j\omega_0(n-\tau)}$
- multiply by known frequency $\hat{b}[n]e^{-j\omega_0 n} = e^{j\omega_0\tau}$

Compensate for the fractional delay Now we have to bring back the signal to the original timing. The bulk delay is no problem, it's just an integer number of samples. What creates a problem is the fractional because that will shift the peaks with respect to the sample intervals. So, if we want to compensate for fractional delay we need to compute subsample values.

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

Compute Lagrange Coefficients Lagrange approximation (see Module 6.2)

(Offset the bulk delay)

Estimate the fractional delay

Compute the Lagrange coefficients

Filter with FIR filter with its Lagrange coefficients

Delay Compensation Algorithm

(a) estimate the delay τ

(b) compute the 2N + 1 Lagrangian coefficients

(c) filter with teh resulting FIR

The advantage of this strategie is that if the delay changes over time for, all we need to do is to keep the estimation running and update the coefficients

5. Adaptive Equalization

1.1.6 ADSL

1.1.7 Notes and Suplementatry materials

2 Week 8 Module 7:

2.1 Image Processing

3 Installation Prerequistis

Prerequisite dotEmacs

- sudo apt install fonts-firacode
- sudo apt install fonts-cantarlell
- isoeveka-etoil Download from github:
  - sudo mkdir /usr/local/share/fonts/iosevka-font
  - sudo cp iosevka-etoile.ttc *usr/local/share/fonts/iosevka-font*
  - sudo fc-cache -fv
- ditaa Download from sourceforge to ~/java ln -s ditaa0$_{9.jar}$ ditaa.jar
- sudo apt install default-jre // for ditaa
- sudo apt install texlive-xetex
- sudo apt install texlive-pstricks
- sudo apt -y install texlive-science

- sudo apt install dvipng // org-latex-preview
- sudo apt install imagemagick // display inline image
- all-the-icons (melpa)
- git config –global user.email "email@example.com"
- tree-sitter how to install for lsp server????

3.1 TODO Add to github repositiory

- File mode specification error: (json-readtable-error 47)
- Unable to read file "*home/duagon.*emacs.d/git-submodules/org-html-themes/org/theme-readtheorg.setup" [2 times]