Part 4
Signal Processing Applications

0368.3464

עיבוד ספרתי של אותות

Digital Signal Processing for Computer Science

AKA

Digital Signal Processing – Algorithms and Applications

Signal similarity

Signal similarity

We often need to know how *similar* two signals ***x*** and ***y*** are
for example, to detect the appearance of a single in noise

In radar we transmit a known signal
the signal propagates in space until it is reflected by a target
it then travels back and is received by the radar receiver

By accurately detecting the exact time that the signal is received
and subtracting the exact time it was transmitted
we can deduce the distance to the target

However, due to

- the $1/r^2$ propagation loss
- the minute amount of energy reflected

the signal received is very weak
while the noise may be large

We need a very sensitive way to detect the known signal in noise

Signal similarity

To compare two signals
we can compute the difference signal $\boldsymbol{\delta} = \boldsymbol{x} - \boldsymbol{y}$
and define similarity as its energy E_{δ} being *small*

But that doesn't work if (with respect to $\boldsymbol{x}$)

- $\boldsymbol{y}$ has some gain
- $\boldsymbol{y}$ is shifted in time

For example, if $\boldsymbol{y} = g\,\boldsymbol{x}$ then the difference is $\boldsymbol{\delta} = (1-g)\,\boldsymbol{x}$
which is only small if $g \approx 1$

We can derive something better from the energy E_{δ}

$$E_{\delta} = \sum_n (x_n - y_n)^2$$
$$= \sum_n x_n^2 - 2 \sum_n x_n y_n + \sum_n y_n^2$$
$$= E_x - 2\,C_{xy} + E_y$$

C_{xy} is called the (cross)correlation between $\boldsymbol{x}$ and $\boldsymbol{y}$

Correlation

$E_\delta = E_x - 2\,C_{xy} + E_y$

where E_x and E_y are the (constant) energies of the 2 signals

Note that when E_δ is *minimal*, C_{xy} is *maximal*

so we say the signals are *correlated* when C_{xy} is *large*

C_{xy} is still large even if ***y*** has arbitrary gain with respect to ***x***

since the gain is captured in the energy component

For example, if $\boldsymbol{y} = g\,\boldsymbol{x}$ (and thus $E_y = g^2\,E_x$)

then $C_{xy} = \sum_n \boldsymbol{x}_n\,\boldsymbol{y}_n = g \sum_n \boldsymbol{x}_n\,\boldsymbol{x}_n = g\,E_x$

What do we mean by large?

If $\boldsymbol{y} = g\,\boldsymbol{x}$ then $C_{xy} = \sqrt{E_x E_y}$

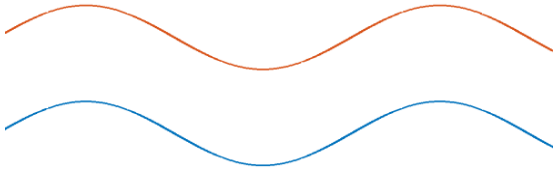
so a simple way to gauge the size of cross-correlation

is to compare it with $\sqrt{E_x E_y}$ (it won't be equal because of noise!)

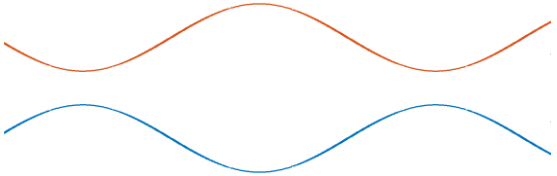
However, C_{xy} will not indicate similarity if ***y*** is shifted in time

Correlation

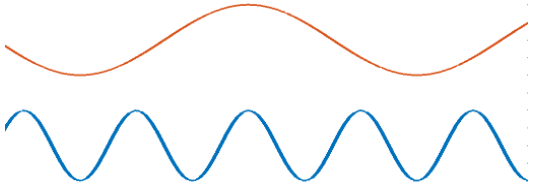
If C_{xy} is positive and large
then x and y are *correlated* (similar)



If C_{xy} is negative and large
then x and y are *anticorrelated*



If C_{xy} is zero (or very small)
then x and y are *uncorrelated* (different)



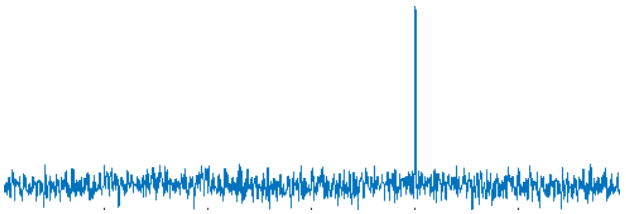
Correlation with lags

To take care of time shifts with define (m is called the correlation *lag*)

$$C_{xy}(m) = \sum_n x_n y_{n-m}$$

(m is called the correlation *lag*)

We now look for its maximum value (its peak)



This peak corresponds to the optimal time shift

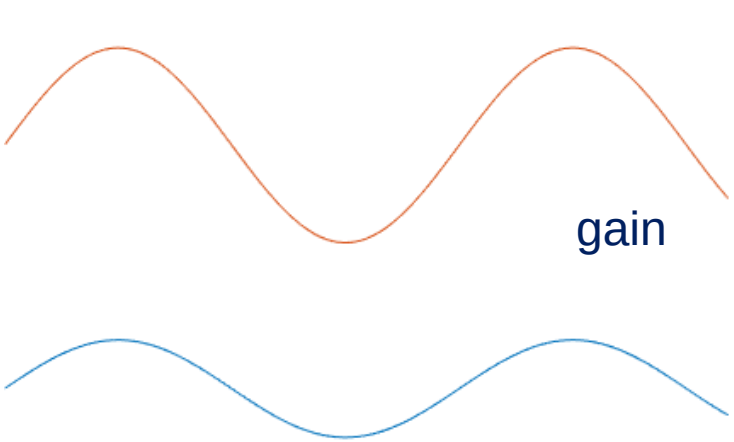
if there the correlation is significant (using the same criterion as before)

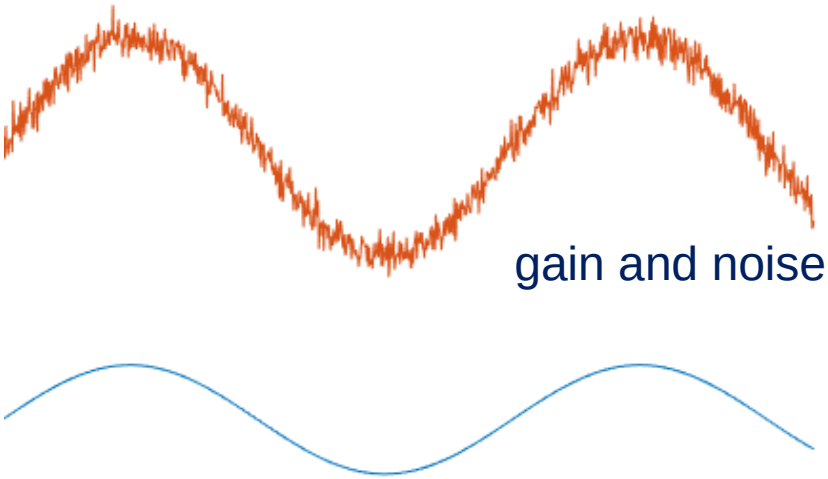
then we have found our similar signal

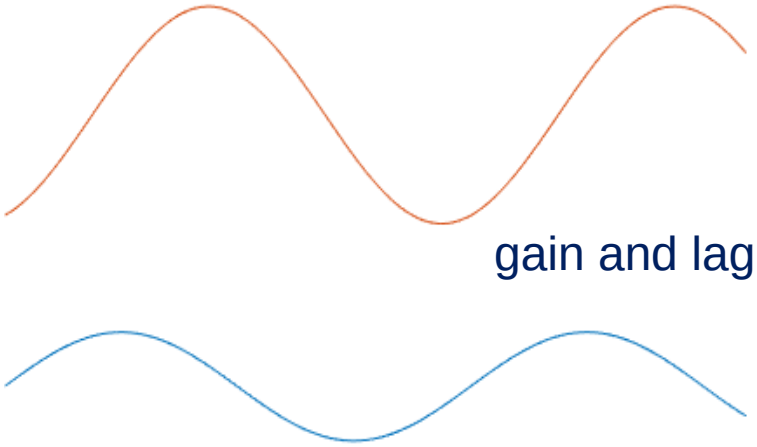
Note the difference between *correlation* and *convolution*

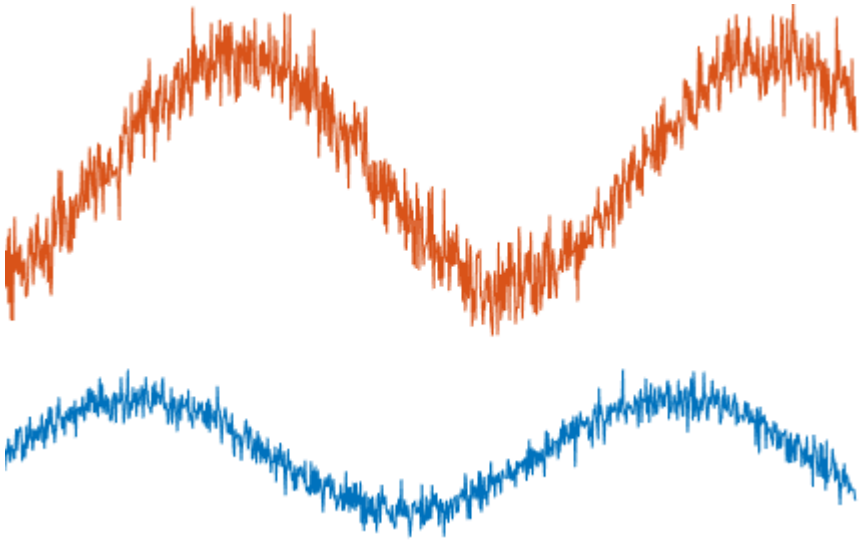
- in *correlation* both indexes increase
- in *convolution* one increases one decreases

Highly correlated signals

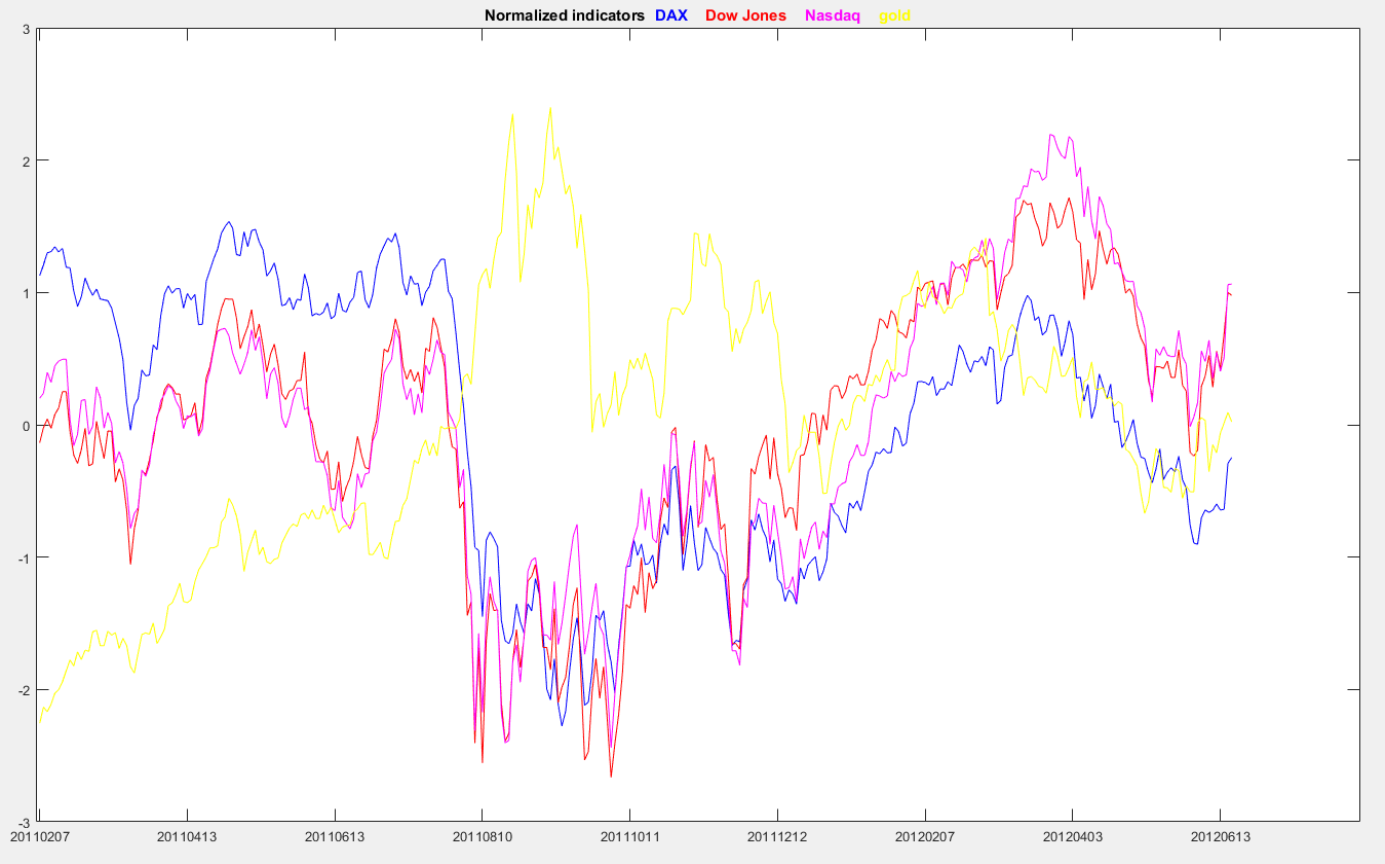








Financial indicators



DAX, Dow Jones, Nasdaq and gold *prices* from 2011-2016

Which signals are correlated? Anticorrelated?

Can you use this for prediction?

Autocorrelation

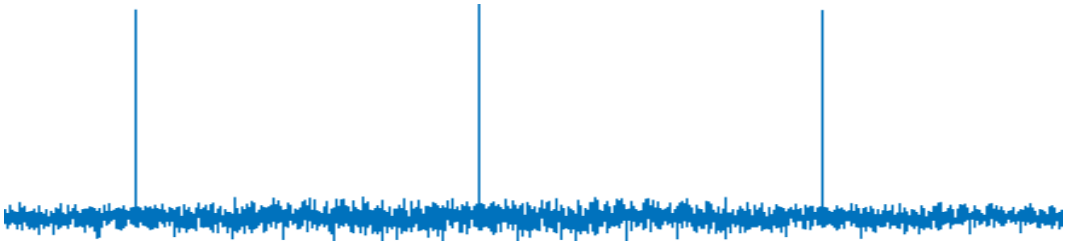
We can define the *auto*correlation $C_{xx}(m) = \sum_n x_n x_{n-m}$

which tells us how similar the signal is to itself

Of course a signal is always similar to itself!

But we use autocorrelation to see how similar a signal is to a time shifted version of itself

A periodic signal has peaks at lags that are multiples of its period!



Since $|C_{xx}(m)| \leq E_x$ it is useful to define the *normalized* autocorrelation

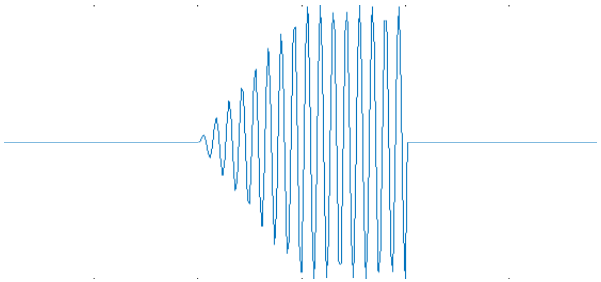
$$c_{xx}(m) = C_{xx}(m) / E_x$$

where $|c_{xx}| \leq 1$

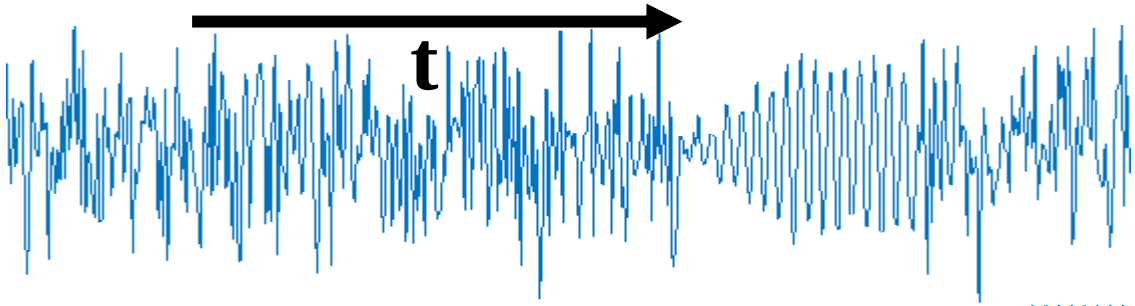
Why can autocorrelation be more accurate than using the FFT?

Wiener filter

Pulse radars transmit a short (and thus low energy) *distinctive* pulse



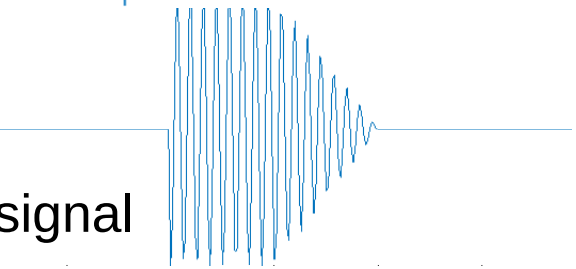
and receive a delayed, weak, noisy copy after time $t = 2 * c * R$



The Wiener filter finds t by building an MA filter

with coefficients equal to the time reversed signal

The MA's *convolution* actually performs a *correlation*



Prediction

Wiener's famous paper was named

Interpolation, extrapolation and smoothing of stationary time series

Where in modern terminology:

- smoothing = low-pass filtering
- interpolation = resampling (sampling at different time instants)
- extrapolation = prediction

We previously saw how Yule predicted sun-spots via an AR process

$$s^*_n = \sum_m^M b_m \, s_{n-m} \quad \text{where } s^*_n \text{ is the estimate of } s_n$$

Let's directly derive the coefficient b for the simplest case (M=1)

We want to minimize the energy of the error signal e = y*-y

$$\sum e^2 = \sum (s_n - s_n^*)^2 = \sum (s_n - b_1 \, s_{n-1})^2 = (1+b_1^2)E_s - 2b_1 C_s(1)$$

Differentiating and setting equal to zero we find $b_1 = C_s(1) / E_s = c_s(1)$

So, there is a connection between prediction and autocorrelation!

More generally, the coefficients are given by Yule Walker equations!

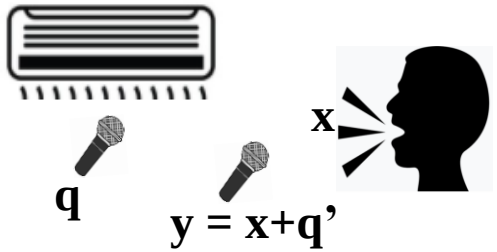
Adaptive filters

We often would like to remove noise from a noisy signal

Sometimes we can access a signal that is highly correlated to the noise

For example, assume 2 microphones

1. air conditioner noise q_n
2. speech x_n + filtered air conditioner noise q'



In the simplest case

the filtering is a *single-tap* MA filter which adds a q_{n-m}

so $y_n = x_n + a\, q_{n-m}$ where a and m are unknown

The idea is to find a and m by adapting our estimates of them

so this is called an *adaptive filter*

If a and q change over time

or even when we simply alter our estimates of them

then this is not a *filter* since it is not time invariant

Adaptation

We can assume that

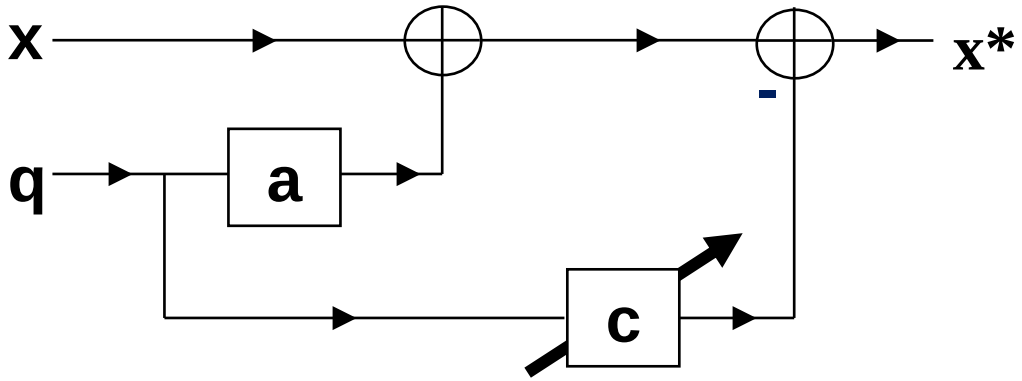
- the desired signal x and
- the noise

are *uncorrelated*

By finding the maximum cross-correlation
between q and y=x+q' (which only looks for q!)
we can determine *m* and thus q_{n-m}

But how do we find *a* ?

Estimate $x^*_n = y_n - c\, q_{n-m} = x_n + (a-c)\, q_{n-m}$



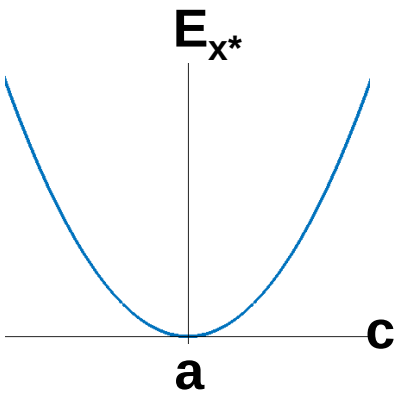
The energy parabola

$x^*_n = x_n + (a-c)\, q_{n-m}$

So $E_{x^*} = \langle x^{*2} \rangle = \langle x^2 \rangle + (a-c)^2 \langle q^2 \rangle + 2\,(a-c) \langle x\, q \rangle$

Since x and q are uncorrelated the last term is zero

The energy is parabolic in c with a single minimum

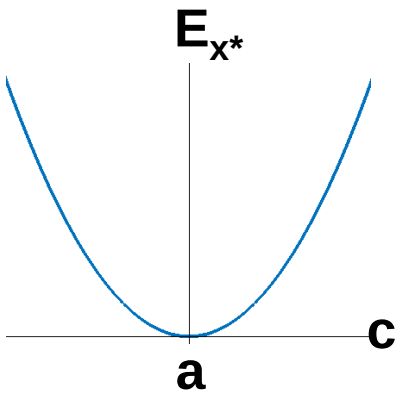


Adaptation

Starting from $x^*_n = x_n + (a-c)\, q_{n-m}$

The global minimum is for c=a and there $x^*_n = x_n$

To find c we need to minimize the energy of x^*_n as a function of c



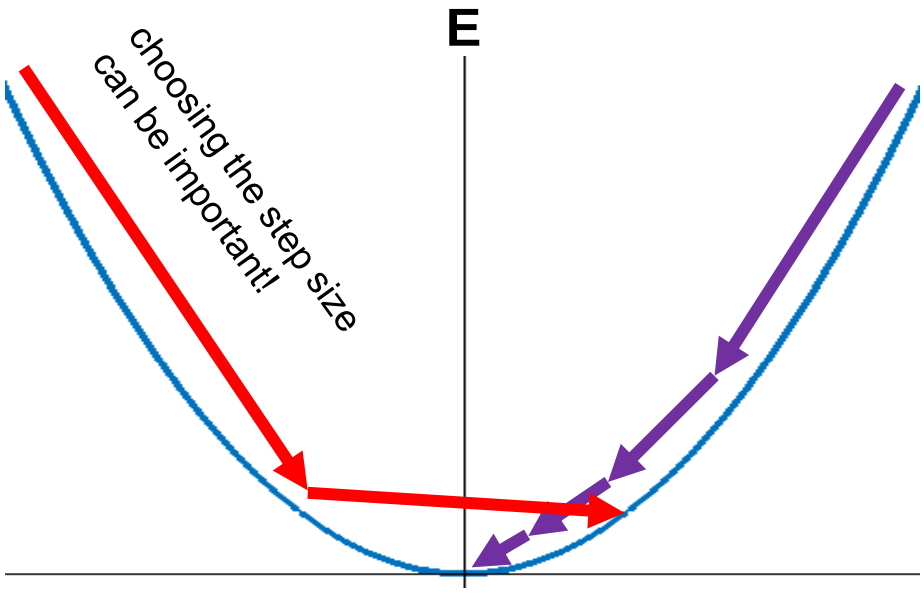
This minimization is typically done via *gradient descent* (AKA steepest descent)

What if we receive the noise and echoes of it? $y_n = x_n + \Sigma_m a_m q_{n-m}$

The same thing can be done for a M-tap filter
using M-dimensional gradient descent

Gradient descent

In gradient descent we choose a *step size* λ
at each step we estimate the gradient (derivative of E) δ
and correct c at each step by $c \leftarrow c - \lambda \delta$
The gradient points in the direction that E increases
so we move in the opposite direction (minus sign)
The larger the gradient the further we move

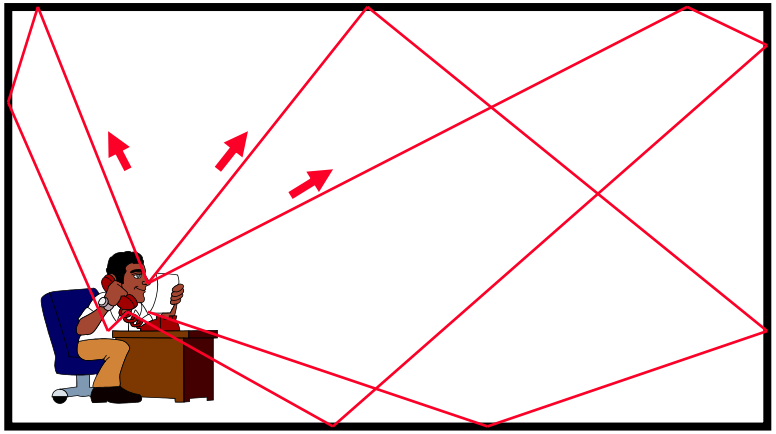


for more complex problems
the energy is not parabolic
and may contain local minima

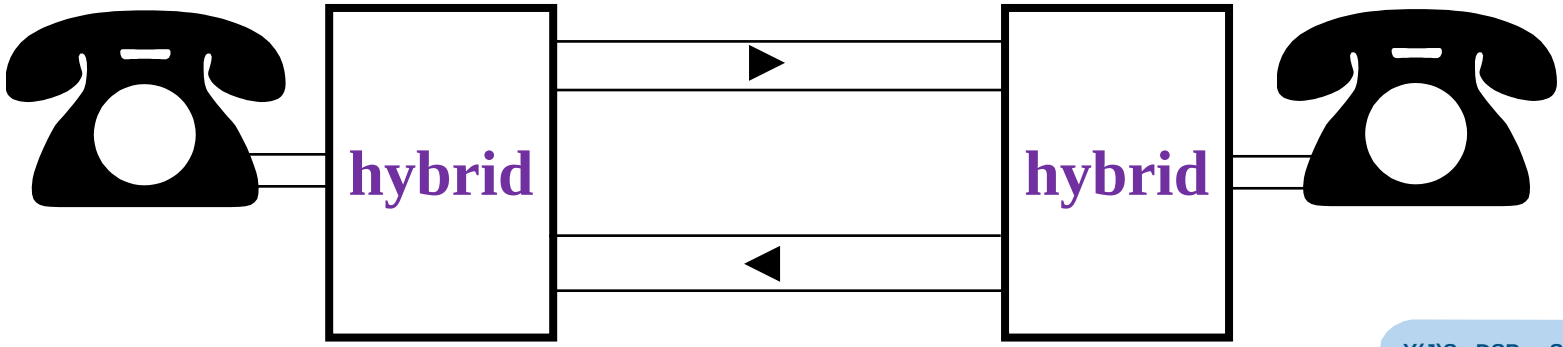
Echo cancellation

Another common use of adaptive filters is *echo cancellation*

- **A**coustic **E**cho **C**ancellation
  e.g., for car hands-free units



- **L**ine **E**cho **C**ancellation



Echo suppressors

Telephony hybrids are not perfect
leaking some amount of echo is sent back to the speaker
If the echo delay is < 20 ms then this is not noticed
but round-trip delay for US coast-to-coast is > 50 ms
and for satellite conversations the delay is > ½ second

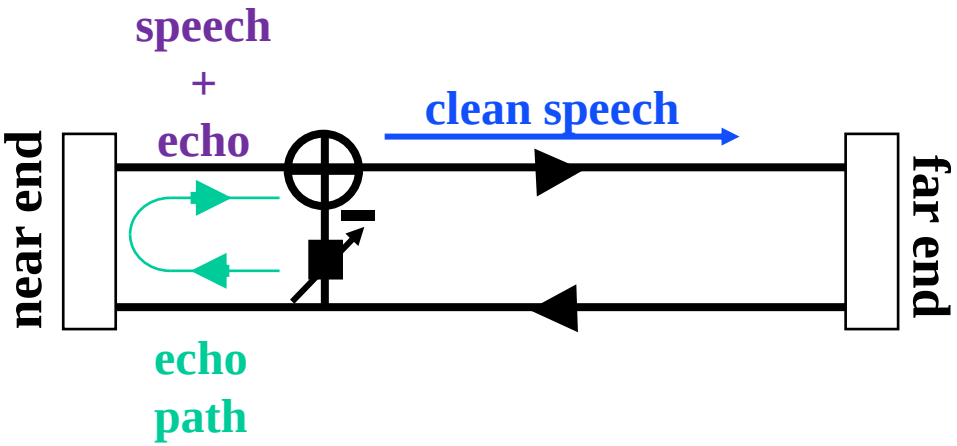
Before the advent of DSPs echo was removed by *echo suppressors*
that chose the louder direction and blocked the opposite direction
effectively making telephone conversations half-duplex
(this still sometimes happens with simple office phones)
For non-speech (fax, modems) echo suppressors are turned off
by sending a 2100 Hz tone
Echo suppression is a waste of full-duplex infrastructure

- conversation is unnatural
- hard to break in
- speaker hears dead line (so telephones artificially add *sidetone*)

LEC was one of the first applications of DSPs

Echo cancellers

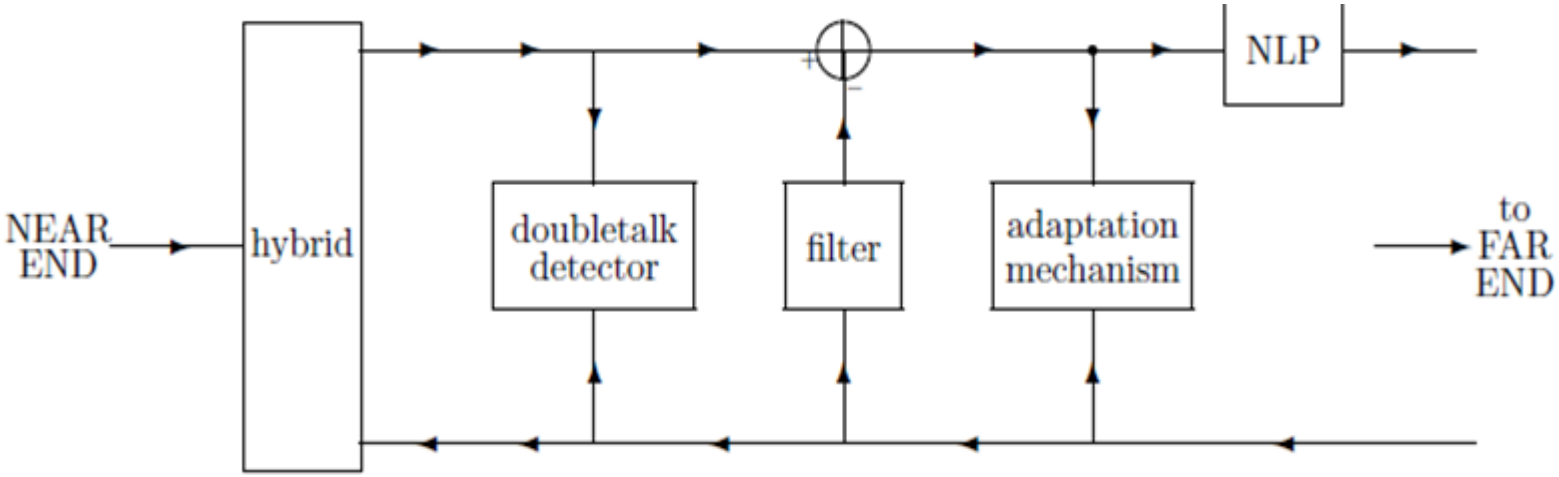
In a simplistic model of LEC the near end simply estimates the echo and subtracts it to send clean (echo-less) speech to the far end (of course the far end does the same)



Real LECs have additional elements, including

- double-talk detectors (Geigel algorithm)
  to freeze adaptation when both sides are speaking
- nonlinear processing (center clipping) to remove residual echo

Realistic LEC



LEC in action

The LEC mechanism

- places samples received from the far-end into the X buffer
- convolves them with the current filter (called the H register) obtaining the echo estimate
- subtracts the echo estimate from the near-end samples

How is the filter adaptation done?

Assume that only the far-end subscriber is talking
then the signal at the input to the subtracter is unwanted echo generated by the near-end hybrid and telephone

The adaptation mechanism varies the filter coefficients
minimizing the energy at the output of the subtracter

If the far-end is quiet or double-talk is detected
the adaptation algorithm automatically stops updating

Equalization

A problem that arises in data communications

We can model the channel distortion as

- filtering $Y(\omega) = H(\omega)\, X(\omega)$ i.e., $y_n = \sum h_l\, x_{n-l}$
- adding noise $y_n = \sum h_l\, x_{n-l} + v_n$

Recovering the original signal

requires applying the *inverse* of the channel filter (equalization)

$X(\omega) = G(\omega)\, Y(\omega)$ i.e., $x_n = \sum g_l\, y_{n-l}$

For this we need to

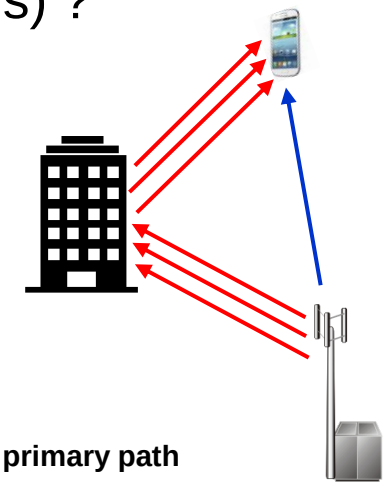
- find the channel filter coefficients h_l
- invert the filter g_l

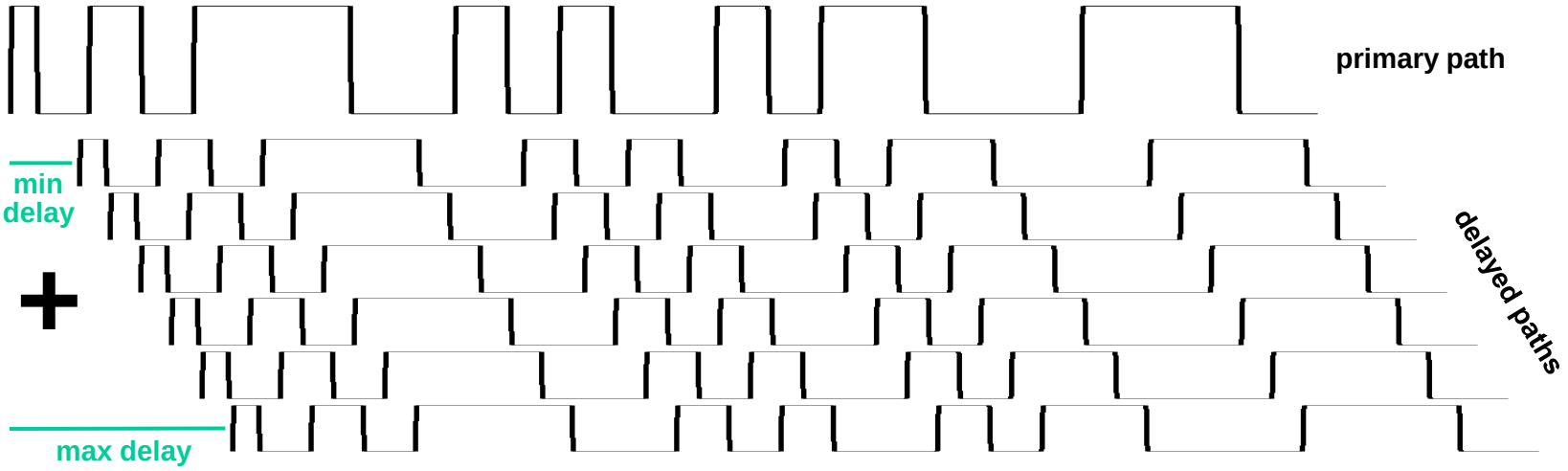
Equalizers are also used in stereo audio systems. Why ?

Multipath

What happens when a signal is received over a primary (shortest) path
and also over delayed paths (e.g. reflections off buildings) ?

The composite signal contains echoes
and is thus an MA filtered version of the original





Finding the equalizer

If we start the transmission with a known signal
then finding the channel filter is a system identification problem

In particular, if we assume that the channel filter

- is MA then we need to solve Wiener-Hopf equations
  and the inverse filter will be AR why?
- is AR then we need to solve Yule-Walker equations
  and the inverse filter will be MA why?

Sometimes we can't transmit a special known sequence - why?
for example, receiver may turn on at any time
then we need *blind* equalization

And even if we can, what if the channel changes over time? why?
for example, a *mobile* receiver

Then we need to adapt the equalizer over time
if we can derive a measure of received SNR
we can use gradient descent

Using the equalizer

There are two ways to apply equalization G to compensate for H

- at the receiver $x = G\ y = G\ H\ x = H^{-1}\ H\ x$
- at the transmitter (Tomlinson precoding) : transmit $x' = G\ x$
  and receiver automatically sees $y = H\ G\ x = H\ H^{-1}\ x$

If the channel filter is not AR then the second way is better – why?

Because at frequencies where the filter has zeros
the channel frequency response is not invertible

That is, if at ω_0 we have $H(\omega_0)=0$
then no $G(\omega_0)$ can obey $G(\omega_0)\ H(\omega_0) = 1$
and so information there is lost

Even if $H(\omega_0)$ is not exactly zero, but very small $H(\omega_0) \approx 0$
$G(\omega_0)$ will have to be very large
and will lead to *noise amplification*
(we have been neglecting the additive noise until now ...)

However, Tomlinson requires a *back-channel*!

Speech signal processing

Application: Speech

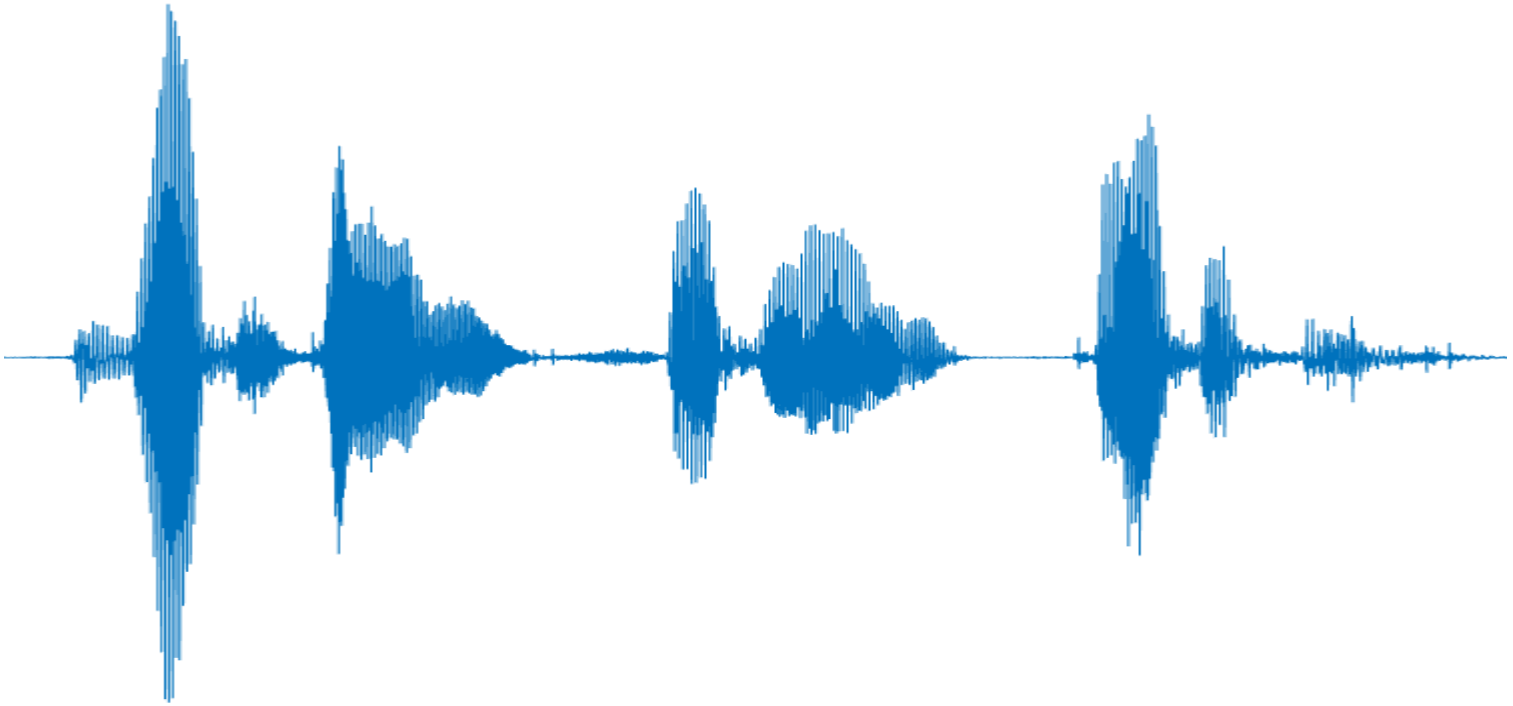
Speech is a wave traveling through space
    at any given point in space it is a signal in time

The speech values are pressure differences (or molecule velocities)

There are many reasons to process speech, for example

- speech storage / communications
- speech compression (coding)
- speed changing, lip sync
- text to speech (speech synthesis)
- speech to text (speech recognition)
- translating telephone
- speech control (commands)
- speaker recognition (forensic, access control, spotting, …)
- language recognition, speech polygraph, …
- voice fonts

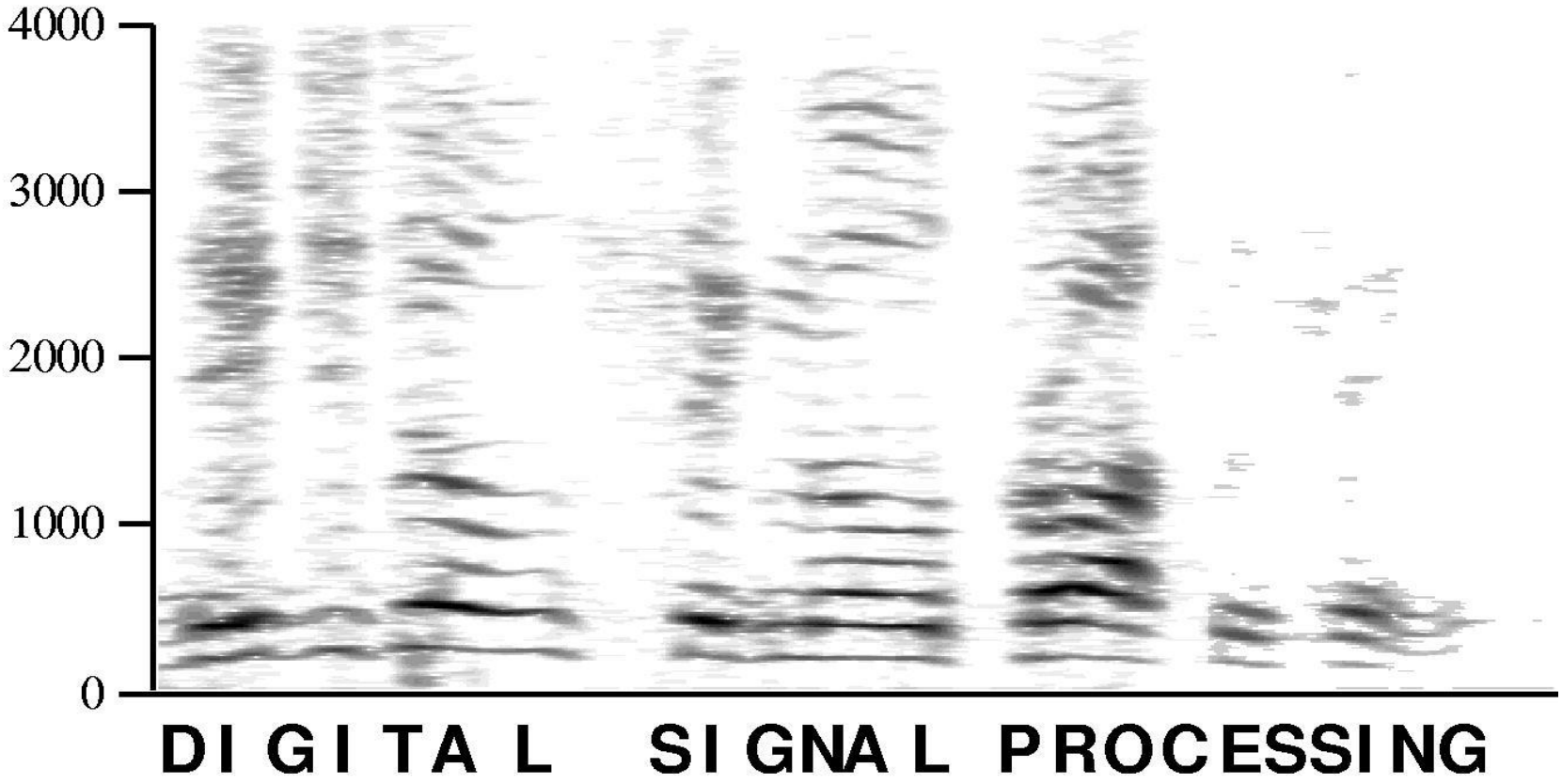
Speech in the time domain



D IGI T A L S I GNA L PROCESSING

Why don’t we see the letters?

Speech in the frequency domain



Why don’t we see the letters?

Phonemes

A phoneme is defined to be the smallest acoustic unit
that can change meaning

Different languages have different phoneme sets
and their speakers *hear* sounds differently

- in Hebrew there is no TH phoneme, so speakers substitute Z
- in Arabic there is no P phoneme, so speakers substitute B
- in English there is no ɔ phoneme, so speakers substitute K
- in Japanese there is a single phoneme
somewhere between English R and L

Some languages have many phonemes

- Danish has 25 different vowels
- Taa has over 150 consonants
- Xhosa has 3 different *click* sounds

Some have very few

- Piraha has 3 vowels and 8 consonants
- Hawaiian has only 3 consonants

Some phoneme types

Vowels

- front (heed, hid, head, hat)
- mid (hot, heard, hut, thought)
- back (boot, book, boat)
- dipthongs (buy, boy, down, date)

Semivowels

- liquids (w, l)
- glides (r, y)

Consonants

- nasals (murmurs) (n, m, ng)
- stops (plosives)
  - voiced (b,d,g)
  - unvoiced (p, t, k)
- fricatives
  - voiced (v, that, z, zh)
  - unvoiced (f, think, s, sh)
- affricatives (j, ch)
- whispers (h, what)
- gutturals (ח ,ע)
- clicks
- etc. etc. etc.

Speech sampling

Telephony speech is from 200 to 4000 Hz
- this is not enough for all sounds we can hear
  - not really even enough for speech (e.g., S - F)
- high quality audio is from 20 Hz to over 20 kHz

By the sampling theorem we need 8000 samples per second
- high quality audio typically sampled at 44.1 / 44 / 48 kHz

If we sample at 8 bits per sample
- we perceive significant quantization noise (see next slide)

So, let's assume we should sample at 16 bits per sample
- actually 13-14 bits is enough

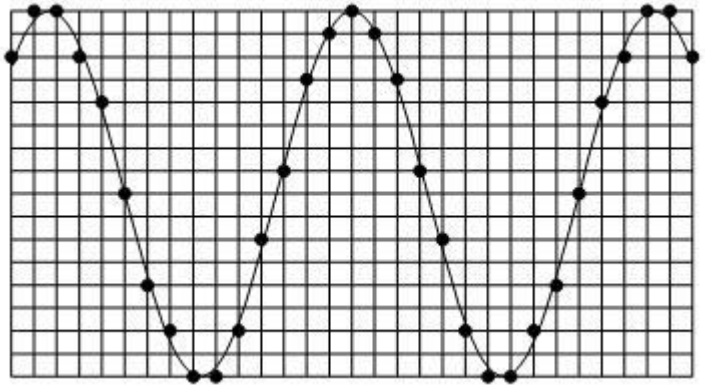
So we need 8000 samples/sec * 16 bits/sample = 128 kbits/sec
- to faithfully capture telephony quality speech

Information theory tells us that speech actually only carries
- a few bits per second

So we should be able to improve this
- but we need to learn more about the speech signal!

What's quantization noise?

When we quantize a signal value s_n to the closest integer (or rational) we add an error value $x_n = s_n + q_n$



Assuming the q_n samples are uncorrelated the quantization noise is white noise

We hear white noise as a hiss

$q_n \varepsilon [-½\ 2^{-n}, ½\ 2^{-n}]$

(or $[-½\ 2^{-n}, ½\ 2^{-n}]$ if we assume the signal is in [-1 , 1])

So the more bits we use the better the **S**ignal to **N**oise **R**atio

Each additional bit reduces the SNR by a factor of 2 (3 dB)

Speech biology

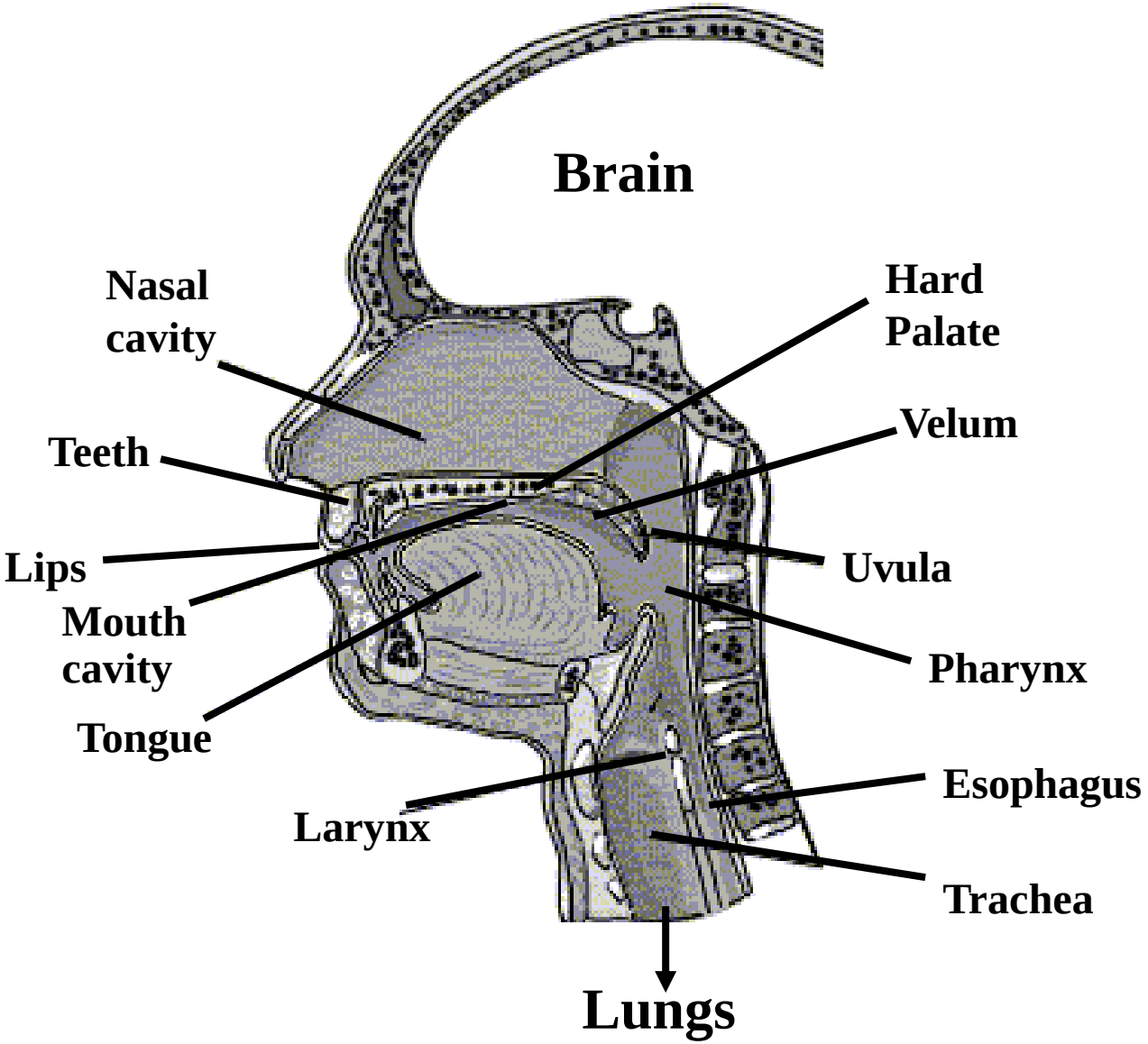
To understand the speech signal
we need to learn some biological signal processing

There are two separate systems of interest

- speech generation system
  - lungs
  - windpipe
  - vocal folds (cords)
  - vocal tract (mouth cavity, tongue, teeth, lips, uvula)
  - Broca's area (in left hemisphere)
- speech recognition system
  - outer ear
  - ear drum and hammer
  - cochlea, organ of Corti (cilia)
  - auditory nerve
  - medial geniculate nucleus (in thalamus)
  - auditory cortex and Wernicke's area (in superior temporal gyrus)

These two systems are not well matched

Speech Production Organs



Speech Production

Air from **lungs** is exhaled into **trachea** (windpipe)

Vocal cords (folds) in larynx can produce periodic pulses of air by opening and closing (glottis)

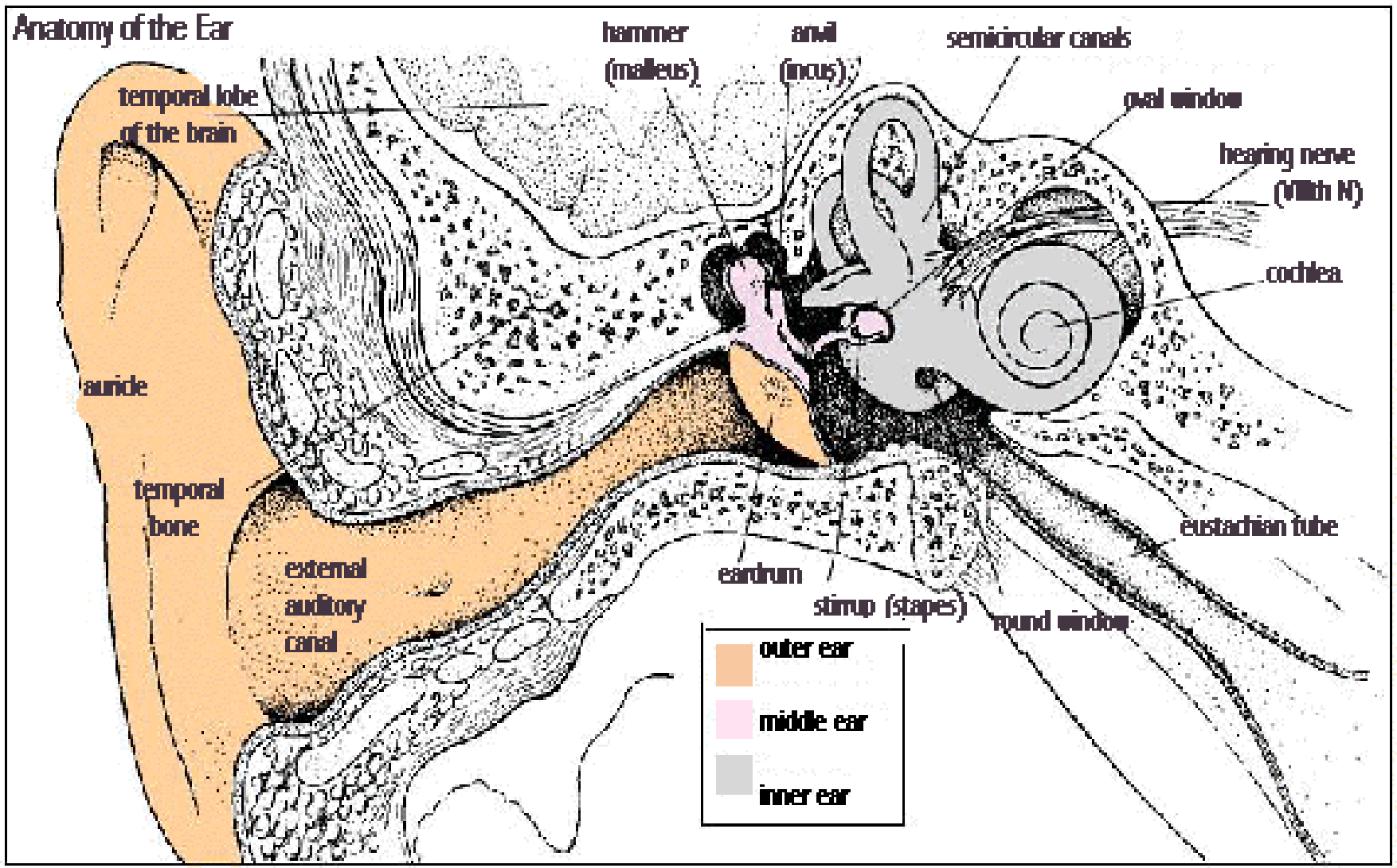
Throat (pharynx), mouth, tongue and nasal cavity *modify* air flow

Teeth and **lips** can introduce turbulence

Basic function – filter an excitation signal

(we will see that the filter can be modeled as an AR filter)

Hearing Organs



Hearing Organs

Sound is air pressure changing over time (and propagating through space)

Sound enters the **external ears** which

- collect as much sound energy as possible (like a satellite dish)
- differentiate between right/left and front/back
- even differentiate between up and down

Sound waves impinge on **outer ear** enter **auditory canal**

Amplified waves cause **eardrum** to vibrate

Eardrum separates **outer ear** from **middle ear**

The **Eustachian tube** equalizes air pressure of middle ear

Ossicles (hammer, anvil, stirrup) amplify vibrations

Oval window separates middle ear from inner ear

Stirrup excites oval window which excites liquid in the cochlea

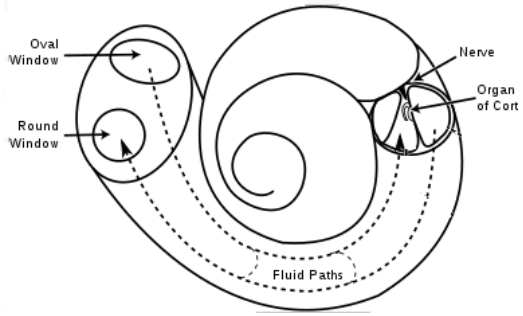
The **cochlea** is curled up like a snail

The **basilar membrane** runs along middle of cochlea

The **organ of Corti** transduces vibrations to electric pulses

Pulses are carried by the **auditory nerve** to the **brain**

Cochlea



- Cochlea has 2 1/2 to 3 turns (for miniaturization) were it straightened out it would be 3 cm in length
- The basilar membrane runs down the center of the cochlea as does the organ of Corti
- 15,000 cilia (hairs) contact the vibrating basilar membrane and when it vibrates they release neurotransmitters stimulating 30,000 auditory neurons
- Cochlea is wide (1/2 cm) near oval window and tapers towards apex and is stiff near oval window and flexible near apex
- hence high frequencies cause vibrations near the oval window while low frequencies cause section near apex to vibrate

Basic function – Fourier Transform
(by overlapping bank of bandpass filters)

Voiced vs. Unvoiced Speech

We saw that air from the lungs passes through the vocal cords

When open the air passes through unimpeded

When laryngeal muscles close them glottal flow is in bursts

When glottal flow is periodic we have *voiced speech*

- basic interval/frequency called the **pitch** (f_0)
- pitch frequency is between 50 and 400 Hz

You can feel the vocal cord vibration
by placing your fingers on your larynx

A laryngeal microphone directly perceives the sound at this point
even opening you mouth very wide doesn't work very well

Vowels are always voiced (unless whispered)

Consonants come in voiced/unvoiced pairs

Exercise

Which are voiced and which are unvoiced ?

B, D, F, G, J, K, P, S, T, V, W, Z,

Ch, Sh, Th (the), Th (theater), Wh, Zh

Which unvoiced phoneme matches the voiced one?

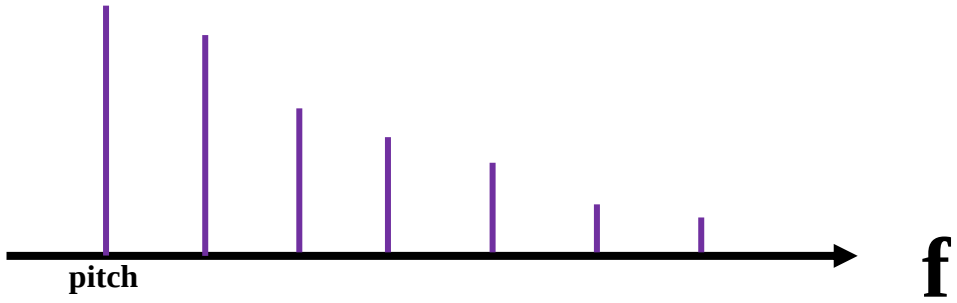
- B
- D
- G
- V
- J
- Th (the)
- W
- Z
- Zh

Excitation spectra

Voiced speech is periodic and so has line spectrum

Pulse train is not sinusoidal due to short pulses of air

and is rich in harmonics (amplitude decreases about 12 dB per octave)



Unvoiced speech is not periodic

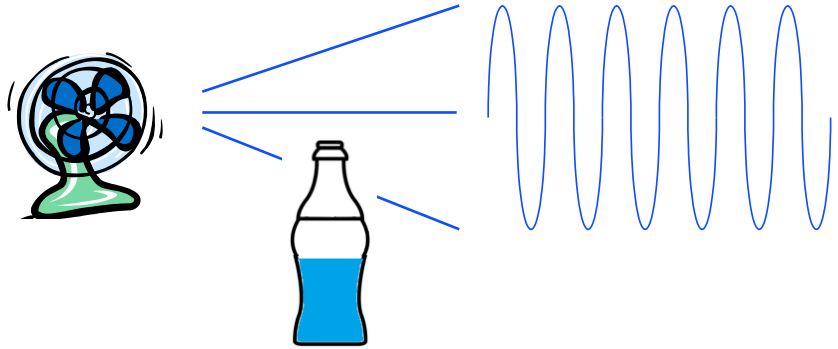
Common assumption : white noise (turbulent air)

Effect of vocal tract

So what is the difference between all the (un)voiced sounds?

The sound exiting the larynx enters the mouth cavity and is filtered

Mouth and nasal cavities have ***resonances*** (poles)
similar to blowing air over a bottle



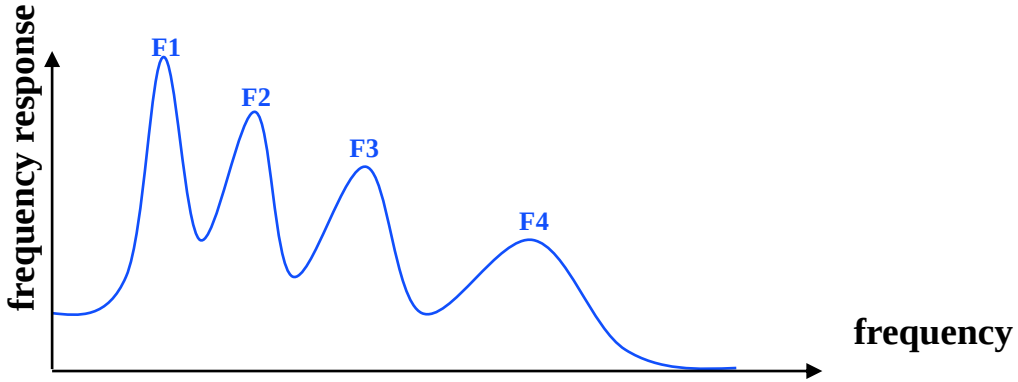
Thus the sound exiting the mouth is periodic (due to periodic excitation)
but with some harmonics strong and some weak

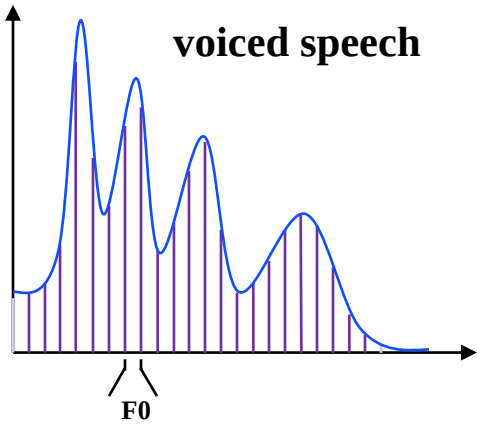
Resonant frequencies depend on geometry
in particular mouth opening, tongue position, lip position

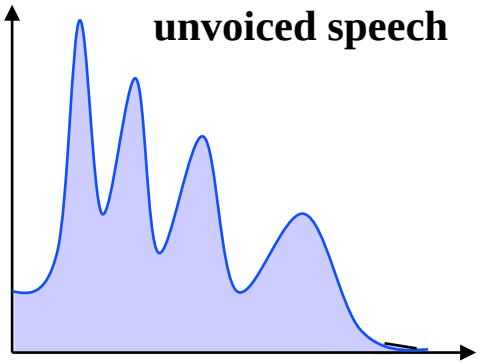
Effect of vocal tract - cont.

Sound energy at resonant frequencies is amplified

Frequencies of peak amplification are called **formants**

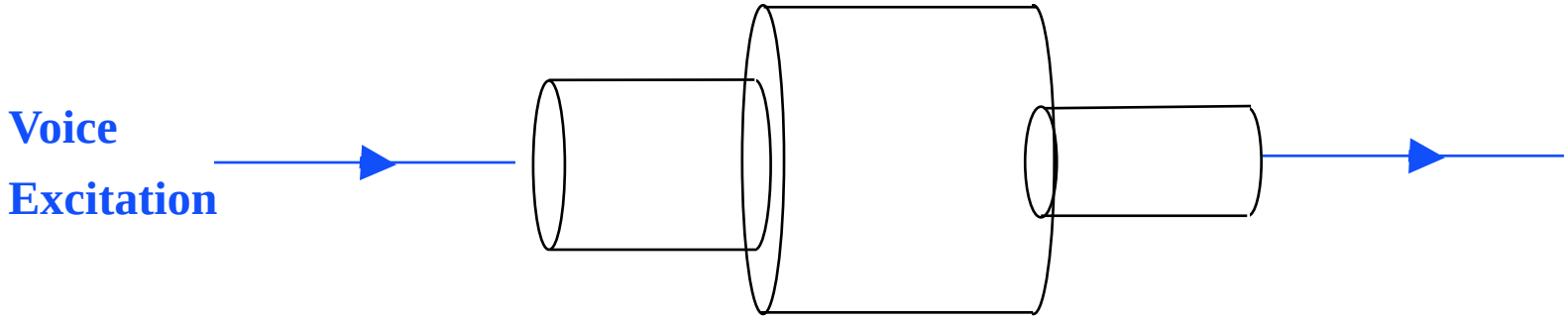




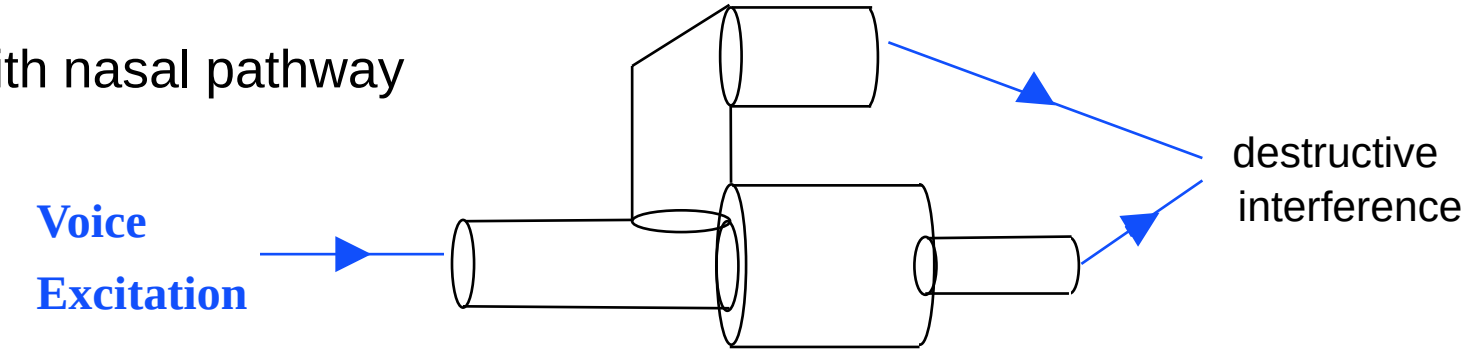


Cylinder model(s)

Rough model of throat and mouth cavity (without naval pathway)

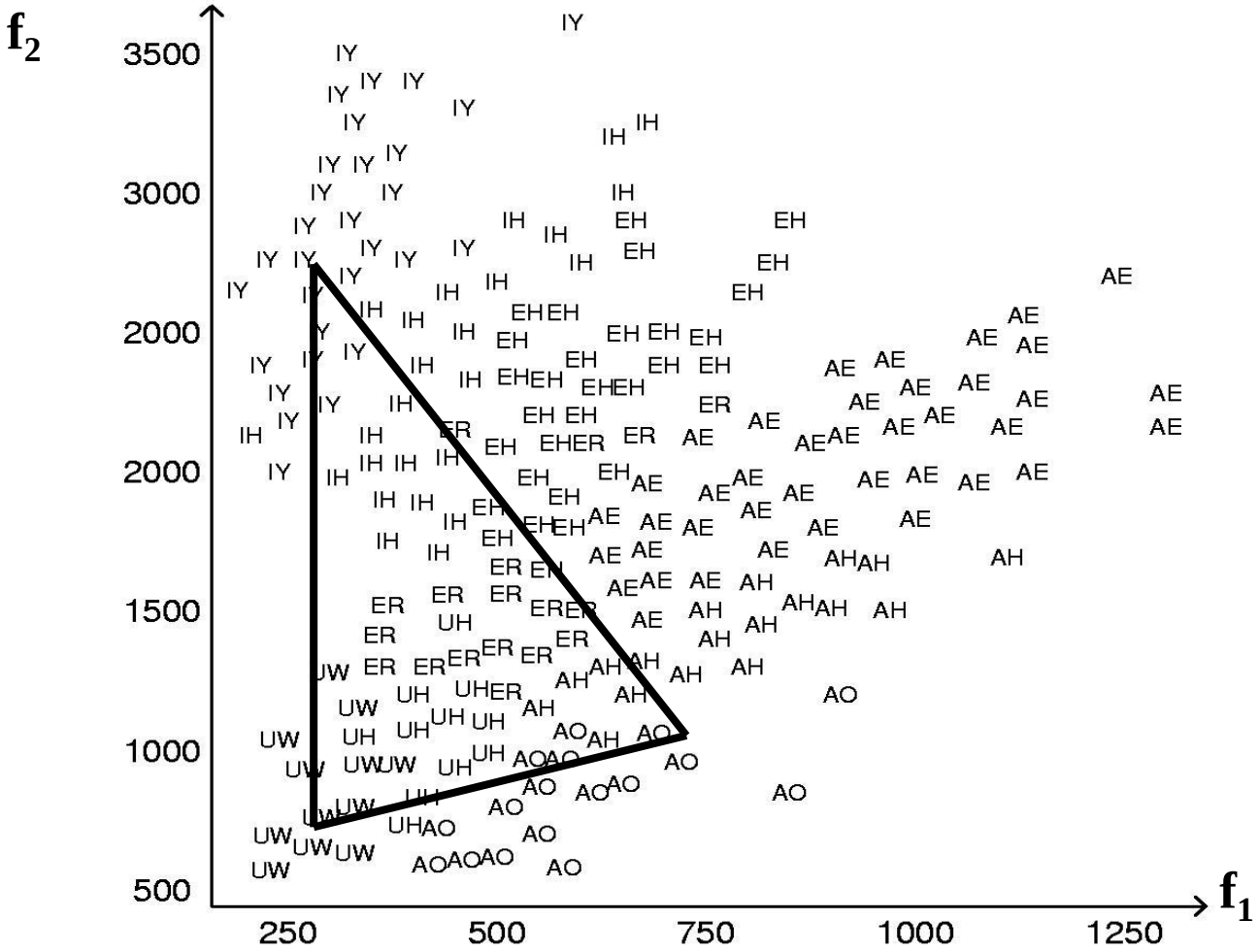


With nasal pathway



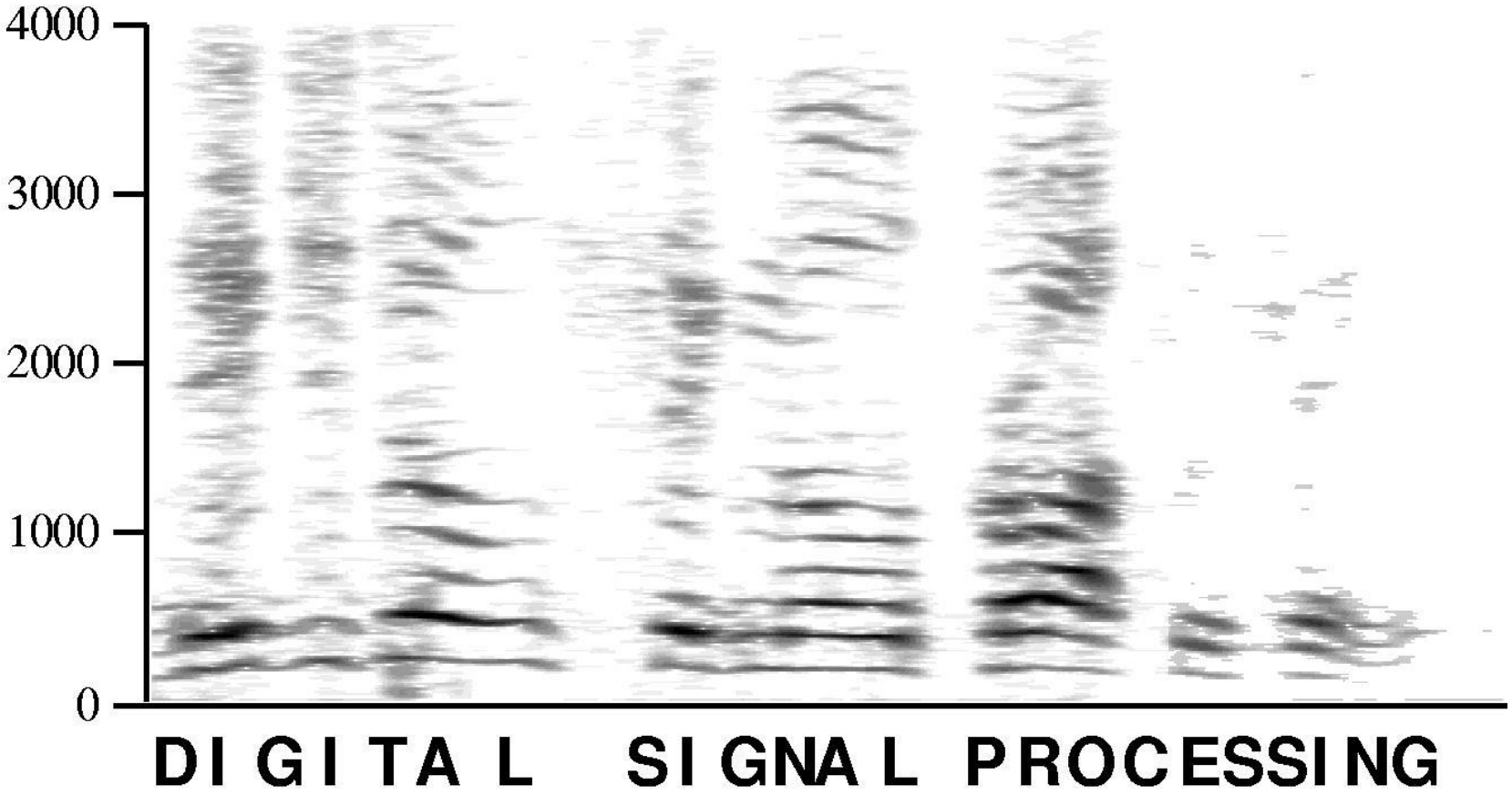
Formant frequencies

- Peterson - Barney data (note the “vowel triangle”)



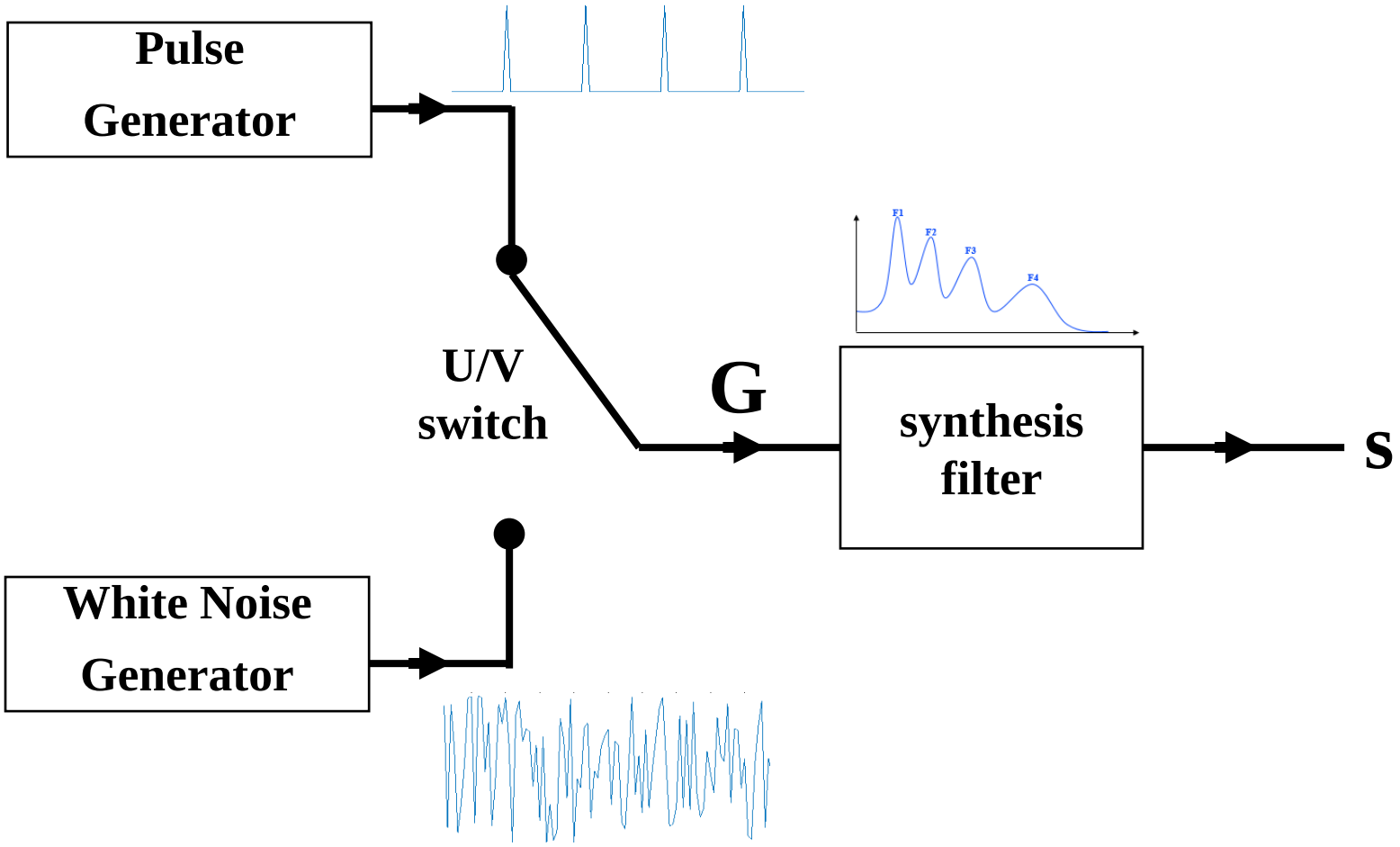
AE = hat
AH = hut
AO = ought
EH = head
ER = hurt
IH = hit
IY = heat
UH = hood
UW = who

Sonograms



Which sounds are voiced?
Where is the pitch?
Where are the formants?

Simple model for speech generation



LPC Model

This model was invented by Bishnu Atal (Bell Labs) in 1960s

- pulse generator produces a harmonic rich periodic impulse train
- white noise generator produces a random signal
- U/V switch chooses between voiced and unvoiced excitation
- varying gain allows to speak loudly or softly
  (typically placed before the filter)
- LPC filter amplifies formant frequencies
  (no zeros but peaks - all-pole or AR IIR filter)
  Note: standard LPC doesn't work well for nasals
  destructive interference creates zeros in the frequency response
- output resembles true speech to within modeling error

Can LPC compress speech?

Let's estimate the number of bits/second required to capture speech (for transmission or storage) using the LPC model

- we need to model several times per phoneme
  - no-one can produce more than a few phonemes per second
  - 20-100 frames per second is reasonable, let's assume 25
- we need to specify the pitch (between 50 and 400) in Hz
  - 1 byte is more than enough (no-one hears the difference of 1 Hz!)
- we need 1 bit for the U/V switch
  - but don't have to waste a bit since we can encode as pitch with zero frequency
- we need to capture the gain
  - 1 byte is enough (our ear is not that sensitive to small gain changes)
- the AR filter has 4 formants
  - so we need 8 values (4 frequencies and 4 amplitudes or 8 poles) and once again will assume 1 byte for each

Altogether we need 25 * (10 bytes * 8 bits/byte) = 2000 bits/sec much better than 128,000 bits/sec

LPC ?

Why is this model called **L**inear **P**redictive **C**oding ?

The **synthesis filter** performs

$$\check{s}_n = \sum_m b_m s_{n-m}$$

which predicts the next signal sample
based on a linear combination of previous samples

Most of the time we can forget about the glottal excitation
but this introduces an error e_n

So we define

$$\check{s}_n = e_n + \sum_m b_m s_{n-m}$$

This defines a classic AR model, solvable using Yule-Walker

How do we do it?

Given a frame of speech samples (x_0, x_1, x_2, ... x_{N-1})
how do we find the LPC parameters?

- the gain is easy to find – it requires calculating the energy
- U/V can be found by observing the spectrum
  if lines then voiced, if continuous then unvoiced
- the pitch can be determined by
  - lowest spectral peak or frequency difference between peaks
  - autocorrelation
- since the input to the AR filter can now be created
  and the output is the speech samples
  we now have an AR filter system identification problem
  so the filter can be found by solving the Yule-Walker equations

Does it work?

An early commercial use of LPC
was the Texas Instruments **Speak** & Spell chip
which used LPC-10 (10 AR coefficients)

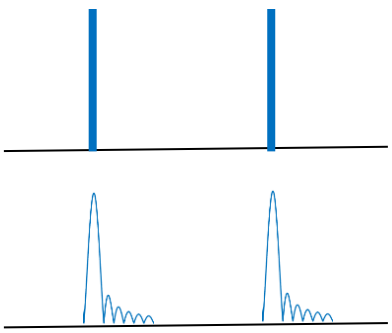
An early software implementation was Klattalk

Pure LPC speech sounds robotic (Steven Hawking speech) because

- we need to add prosodic modeling
- we need to post process to clean up estimation errors
  e.g., pitch doubling
- we need to add frame-to-frame processing
  e.g., pitch needs to change smoothly

but most importantly

- pitch pulses are not deltas or on-off rectangles
  they have waveforms that influence the sound



Better speech compression

Compressing telephony quality speech
was once a tremendously important problem

The **I**nternational **T**elecommunications **U**nion set goals
to reduce speech transmission rates by factors of 2

We can thus compare:

- 128 kbit/sec – linearly quantized speech
- 64 kbit/sec – Pulse Code Modulation (logarithmic quantization)
- 32 kbit/sec – Adaptive Delta PCM
- 16 kbit/sec was never standardized
- 8 kbit/sec – Code Excited LPC

But we now need to understand some **psychophysics**

Psychophysics

Psychophysics is the subject that combines psychology and physics

Its fundamental question is the connection between

- *external* physical values (light, sound, etc.)
- *internal* psychological perception

Ernst Weber was one of the first to investigate of the senses

In a typical experiment

a subject is asked in which hand there are more coins

- 1 coin in 1 hand and 2 coins in the other is easily noticeable
- 10 coins in 1 hand and 11 coins in the other is just noticeable
- 41 coins in 1 hand and 42 in the other is not noticeable

But how could this be?

Can one notice the difference of 1 coin or not?

Weber

Weber defined the concept of the **J**ust **N**oticeable **D**ifference (JND)
the minimal change produces a noticeable difference

Weber's important discovery was that JND varied with signal strength

In fact, Weber found

the sensitivity of a subject to X is in direct proportion to the X itself

that is

one notices adding a specific *percentage not* an *absolute* value

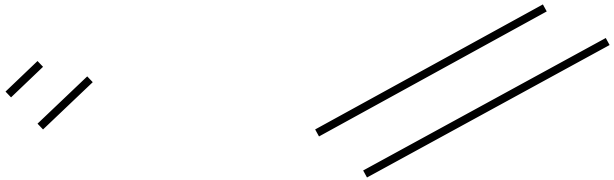
$$\Delta I = k I$$

Thus

- the number of noticeable additional coins
  is proportional to the number of coins

and the same is true for

- saltiness of salty water
- length of lines
- strength of sound



Fechner's law

Gustav Theodor Fechner was a student of Weber

While viewing the sun in order to discover JNDs he was blinded

He later regained his sight and took this as a sign
that he would solve the psychophysical problem

Simplest assumption: **JND is single internal unit**

Weber's law says we perceive external values multiplicatively

Fechner concluded that internal unit is the log of the external one:

$$Y = A \log I + B$$

People celebrate the day he discovered this
as Fechner Day (October 22 1850)

Dynamic range

Logarithms are *compressive*

Fechner's law explains the fantastic ranges of our senses

Sight (retinal excitation)

- minimum: single photon
- maximum (harm threshold): direct sunlight
- ratio: 10^{15}

Hearing (eardrum movement)

- minimum: < 1 Angstrom 10^{-10} meters
- maximum (harm threshold): >1 mm (behind a jet plane)
- ratio: 10^{8}

Alexander Graham Bel defined to be $\log_{10}$ of power ratio

A **decibel** (dB) one tenth of a Bel

$$d(dB) = 10 \log_{10} P_1 / P_2$$

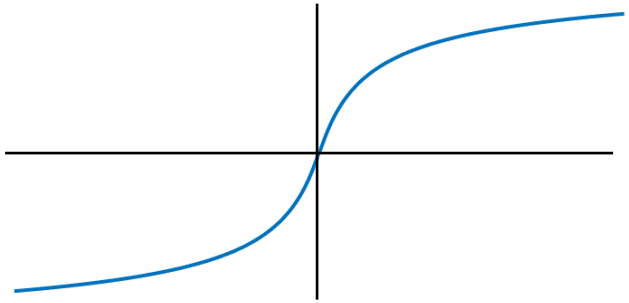
Fechner's law for sound *amplitudes*

According to Fechner's law, we are more sensitive at lower amplitudes

We perceive small differences when the sound is weak
but only perceive large differences for strong sounds

Companding is the compensation
for the logarithmic nature of speech perception

We need to adapt the logarithm function
to handle positive/negative signal values



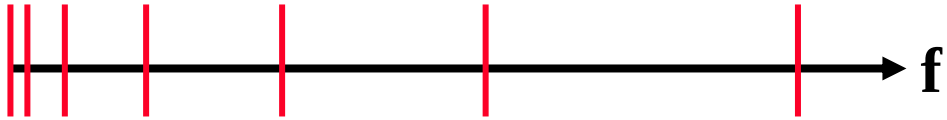
If we could sample non-evenly
then this can be exploited for speech compression

Fechner's law for sound *frequencies*

octaves, well tempered scale

Critical bands

Frequency warping



Melody 1 KHz = 1000, JND afterwards $M \sim 1000 \log_2 (1 + f_{KHz})$

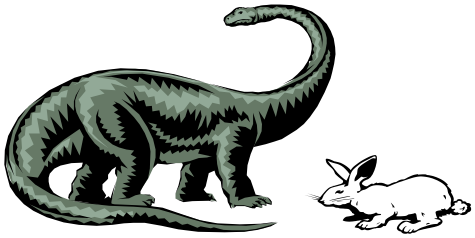
Barkhausen can be simultaneously heard $B \sim 25 + 75 (1 + 1.4 f^2_{KHz})^{0.69}$

excite different basilar membrane regions

2 more psychological laws

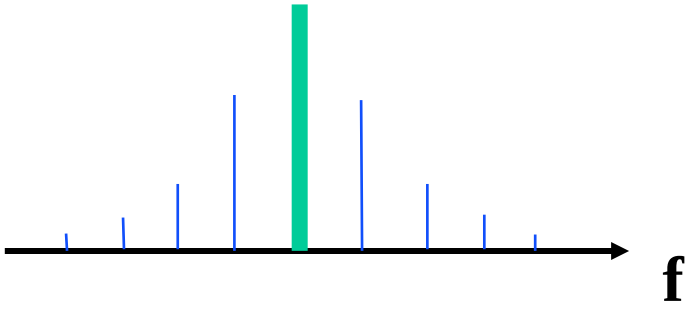
Respond to changes

Our senses respond to changes – not to constant values



Masking

Strong tones block weaker ones at nearby frequencies



PCM

8 bit linear sampling (256 levels) is noticeably noisy

But due to

- prevalence of low amplitudes in generated speech
- logarithmic response of ear

we can use 8-bit *logarithmic sampling* (with sgn(s) * log(|s|))

G.711 gives 2 different logarithmic approximations

μ-law $\check{s} = \operatorname{sgn}(s)\,\check{s}_{max}\,\frac{1+\mu\frac{|s|}{s_{max}}}{1+\frac{|s|}{s_{max}}}$ **North America μ = 255**

A-law $\check{s} = \operatorname{sgn}(s)\,\check{s}_{max}\begin{cases}\frac{A\frac{|s|}{s_{max}}}{1+\ln(A)} & 0 < \frac{|s|}{s_{max}} < \frac{1}{A} \\ \frac{1+\ln(A\frac{|s|}{s_{max}})}{1+\ln(A)} & \frac{1}{A} < \frac{|s|}{s_{max}} < 1\end{cases}$ **Rest Of World A = 87.56**

Although very different looking they are nearly identical

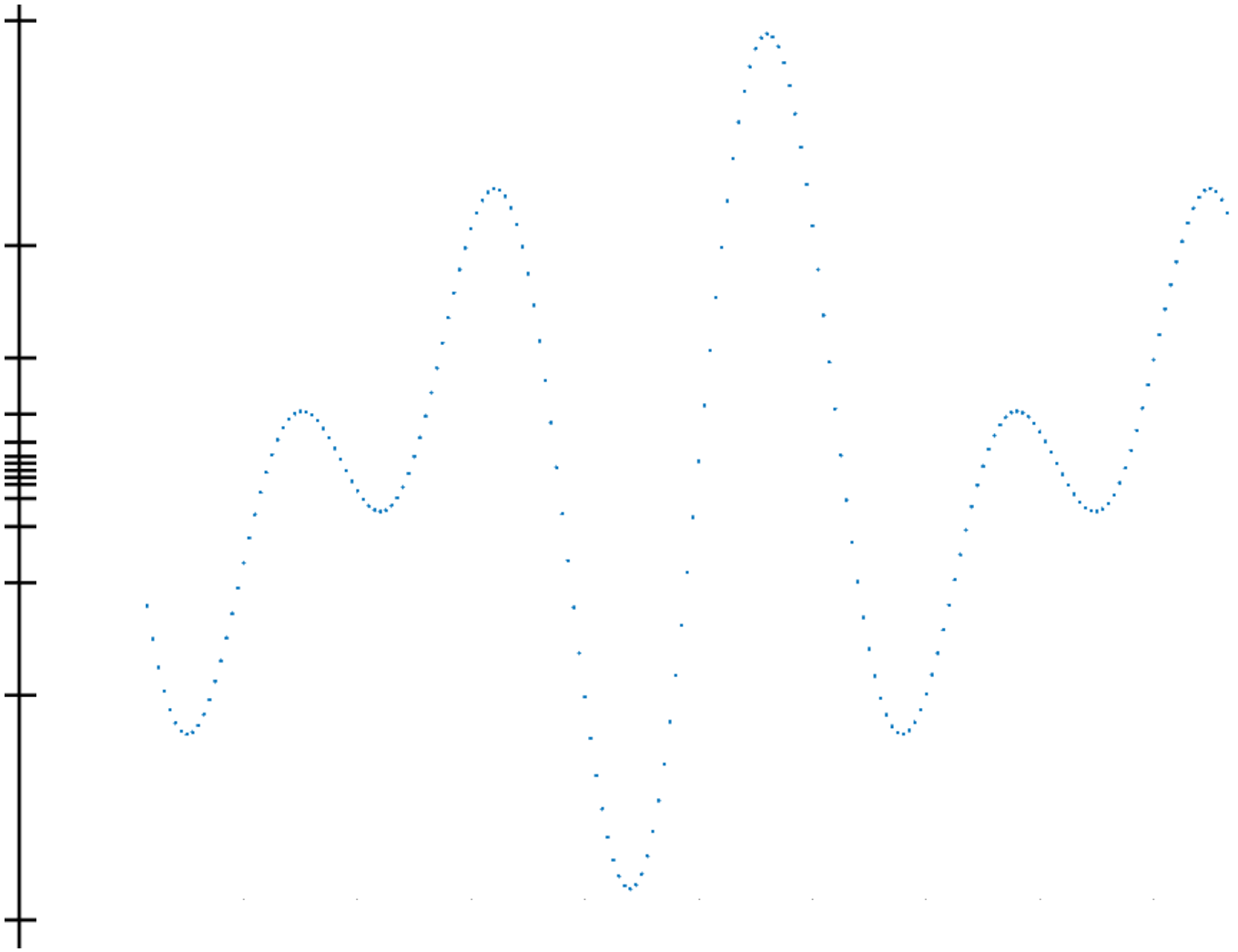
G.711 standard further approximates these expressions

by 16 staircase straight-line segments (8 negative and 8 positive)

Note that μ-law has horizontal segment through the origin (plus and minus zero)

while A-law has a vertical segment

Logarithmic sampling



DPCM

Due to low-pass character of speech excitation

differences are usually much smaller than signal values

and hence require fewer bits to quantize

This would not be the case for white noise!

Simplest Delta-PCM (DPCM) : quantize first difference signal Δs

Delta-PCM : it is even better to quantize
the difference between the signal and its *prediction*

$$\check{s}_n = p\,(\, s_{n-1}\,,\, s_{n-2}\,,\, \ldots\,,\, s_{n-N}\,) = \sum_i p_i\ sn_{-i}$$

Since we predict using linear combination
this is a simple type of *linear prediction*

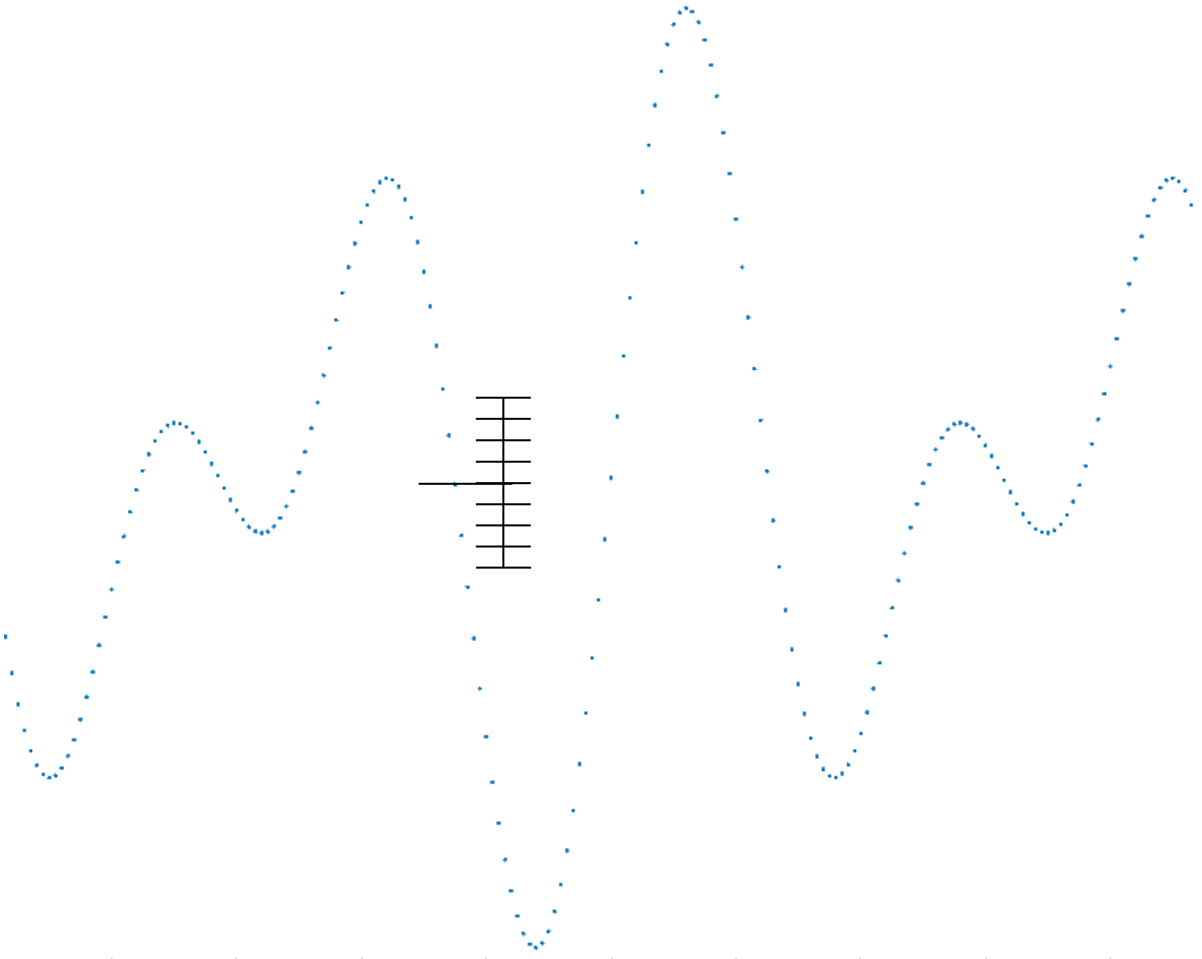
Delta-modulation (DM) : use only the sign of difference (1bit DPCM)

Sigma-delta (1bit) sampling : oversample, DM, trade-off rate for bits

Deltas are usually small

The first difference requires fewer bits

if 4 bits is enough then we need 32 kb/s



DPCM with prediction

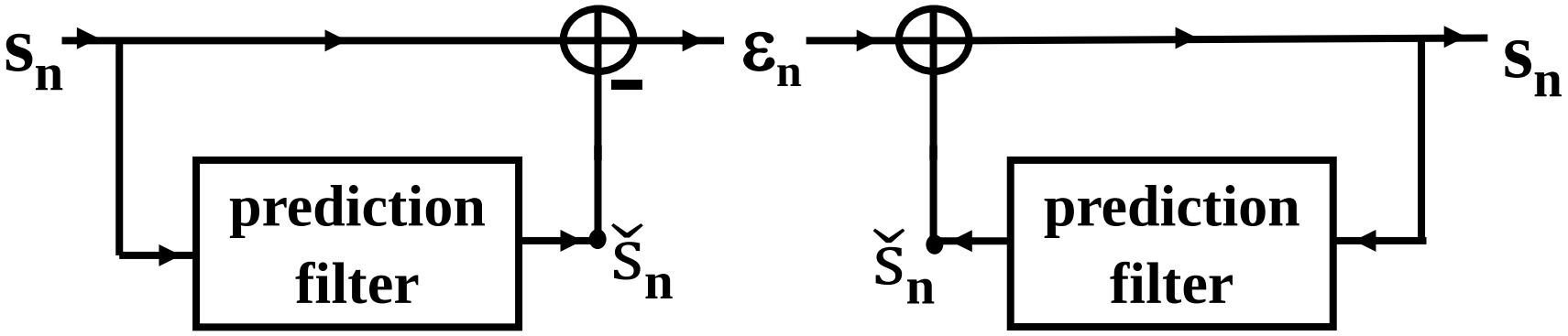
If the linear prediction works well, then the **prediction error**

$$\varepsilon_n = s_n - \check{s}_n$$

will be ***lower in energy*** and ***whiter*** than s_n itself !

Only the error is needed for reconstruction,

since the predictable portion can be predicted $s_n = \check{s}_n + \varepsilon_n$!



Open loop prediction

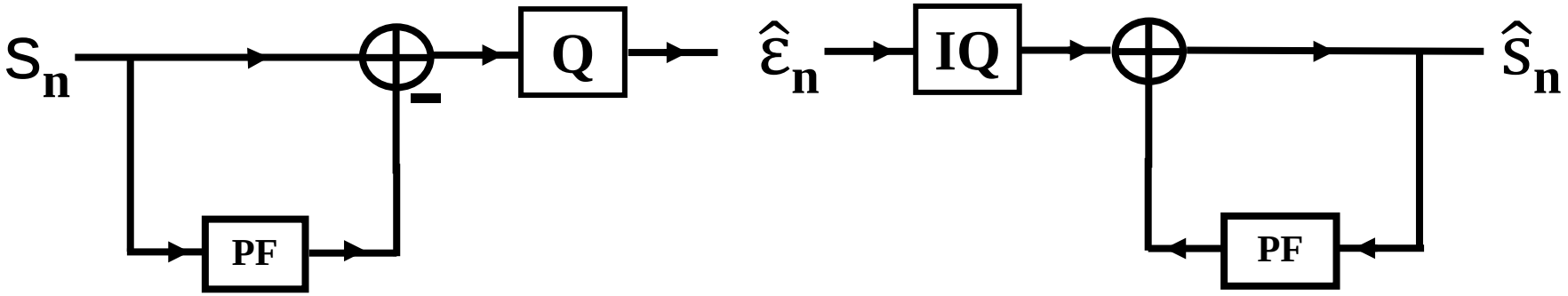
The encoder (linear predictor) is present in the decoder

but there runs as **feedback**

The decoder's predictions are accurate with the precise error ε_n

but it gets the **quantized error** $\hat{\varepsilon}_n$

and as the error accumulates the signals diverge!



Side information

There are two ways to solve the error accumulation problem ...

The first way is to send the prediction coefficients

from the encoder to the decoder

and not to let the decoder derive them

The coefficients thus sent are called side-information

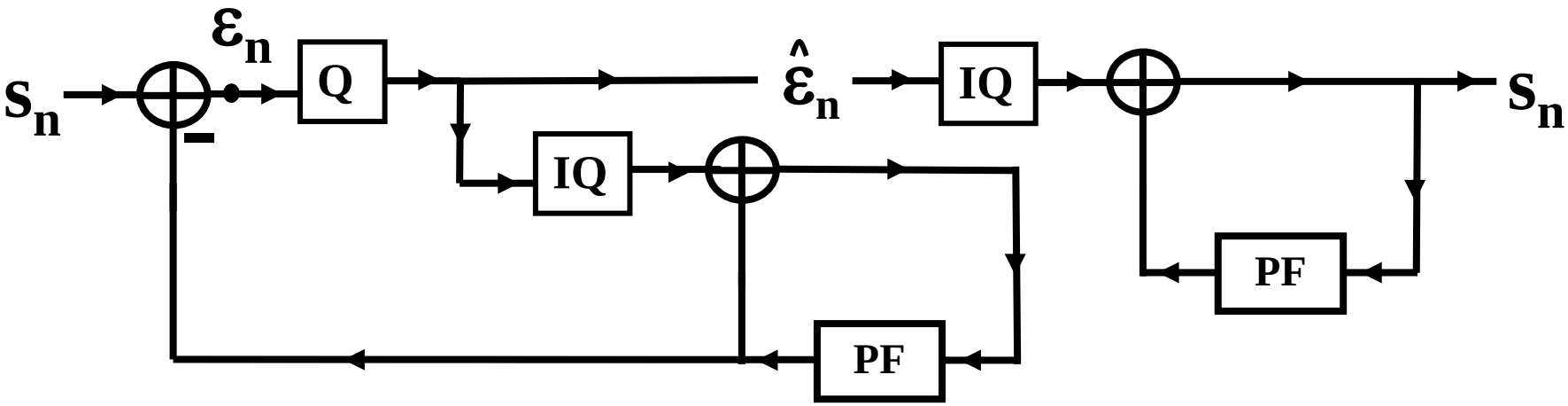
Using side-information means higher bit-rate
(since both $\hat{\varepsilon}_n$ and the coefficients must be sent)

The second way does not require increasing bit rate

Closed loop prediction

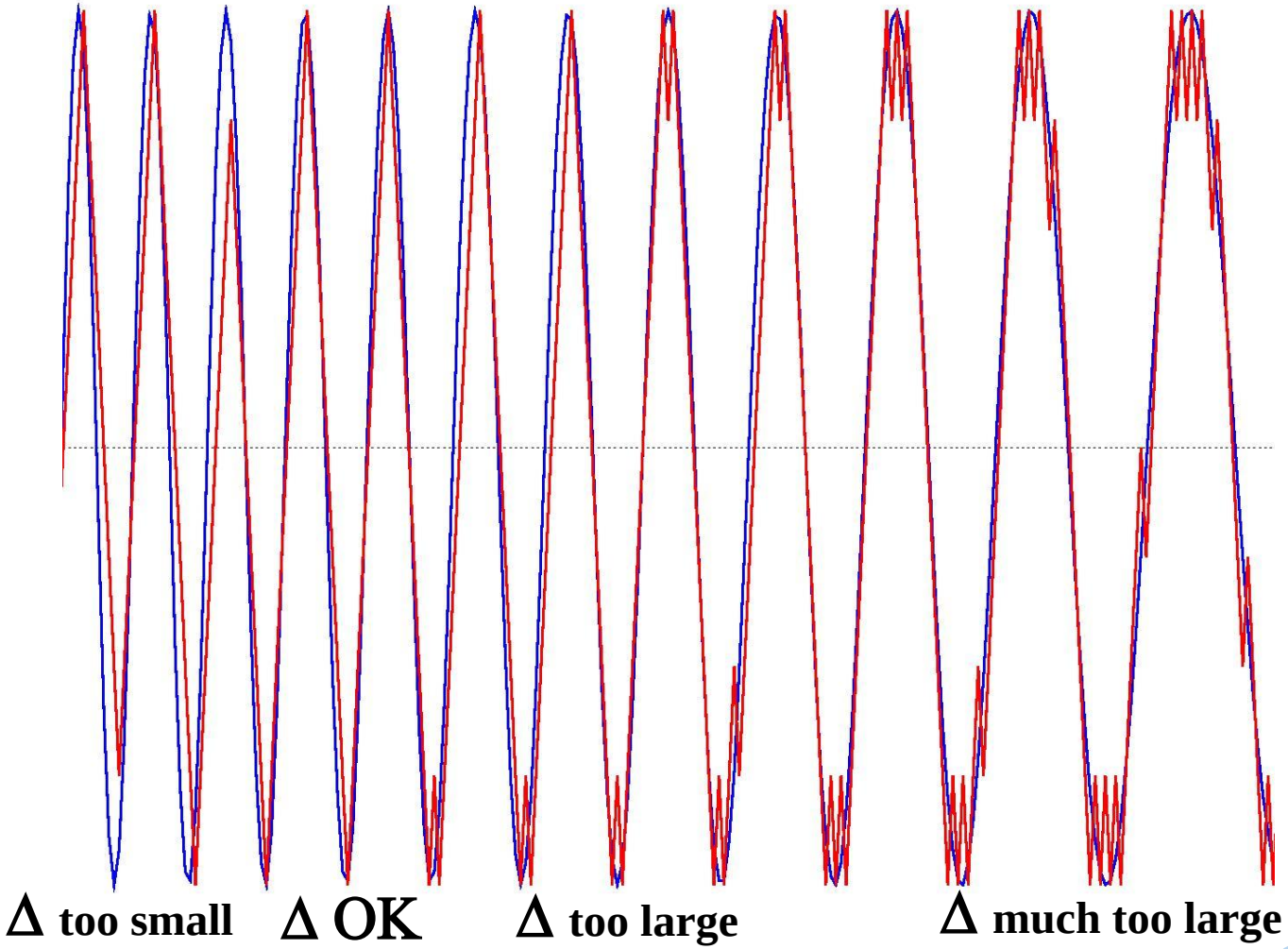
To ensure that the encoder and decoder stay “in-sync”
we put the entire decoder
including quantization and inverse quantization
into the encoder

Thus the encoder’s predictions are identical to the decoder’s
and no model difference accumulates



Two more forms of error

For DM there is another source of error (that depends on step size)



Adaptive DPCM

Speech signals are very nonstationary

We need to adapt the step size to match signal behavior
- increase Δ when signal changes rapidly
- decrease Δ when signal is relatively constant

Simplest method (for DM only):
- if present bit is the same as previous multiply Δ by K (K=1.5)
- if present bit is different, divide Δ by K
- constrain Δ to a predefined range

More general method :
- collect N samples in buffer (N = 128 … 512)
- compute standard deviation in buffer
- set Δ to a fraction of standard deviation
  - send Δ to decoder as side-information or
  - use backward adaptation (closed-loop Δ computation)

G.726

- G.726 (standard for international telephony) has
  - adaptive predictor
  - adaptive quantizer and inverse quantizer
  - adaptation speed control
  - tone and transition detector
  - mechanism to prevent loss from tandeming
- computational complexity relatively high (10 MIPS)
- 32 kbps toll quality
- 24 and 16 Kbps modes defined, but not toll quality

G.727 same rates but *embedded* for packet networks

ADPCM only exploited general low-pass characteristic of speech

What is the next step?

AVQSBC

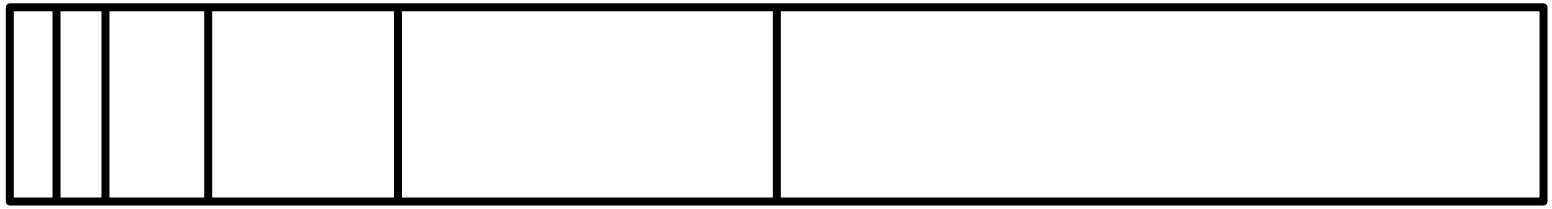
A good quality 16 kb/s speech encoder is obtained

by exploiting Fechner's law for sound frequencies

The idea is to use ***S**ub-**B**and **C**oding*

We divide the signal in the frequency domain
using perfect-reconstruction band-pass filters

The frequency widths are small at low frequencies
but large at high frequencies



Each subband is separate quantized using **V**ector **Q**uantization

This technique was never standardized for speech
but SBC was standardized for music as *MP3 audio*

LPC10

For military-quality voice

180 sample frame (44.4 frames per second)
are encoded into **54 bits** as follows:

- Pitch + U/V (found using AMDF) **7 bits**
- Gain **5 bits**
- 10 reflection coefficients
  - first two coefficients converted to log area ratios
  - L_1, L_2, a_3, a_4 5 bits each
  - a_5, a_6, a_7, a_8 4 bits each
  - a_9 3 bits a_{10} 2 bits **41 bits**
- 1 sync bit **1 bit**

54 bits 44.44 times per second results in 2400 bps

By using VQ one could reduce bit rate to under 1 Kbps!

LPC-10 speech is intelligible, but synthetic sounding
and much of the speaker identity is lost !

CELP

The true s_n is obtained by adding back the *residual* error signal

$$s_n = \check{s}_n + \varepsilon_n$$

So if we send ε_n as *side-information* we can recover s_n

ε_n is smaller than s_n so may require fewer bits !
but ε_n is *whiter* than s_n so may require many bits!

Can we compress the residual?

Note that the residual error is actually the LPC filter's excitation

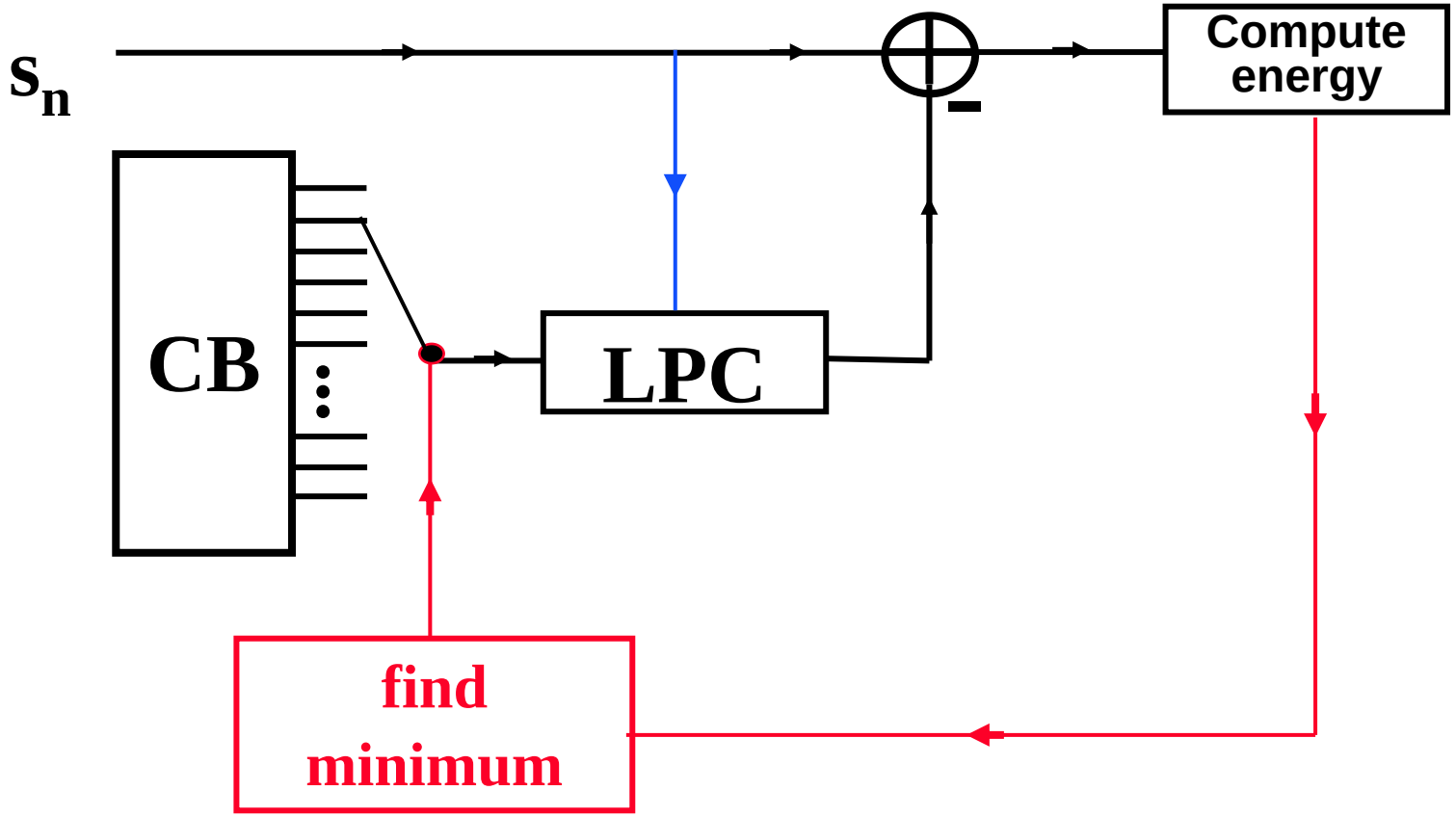
The idea behind **C**ode **E**xcited **L**inear **P**rediction
is to encode possible *excitations* in a codebook of waveforms
and to send the index of the best codeword

Analysis By Synthesis

To find the best code word we try them all (exhaustive enumeration)

although algebraic tricks save computation

We choose the code word with the least error



Perceptual weighting

But we don't want the approximation
    that *looks* the most like the real signal

We want the approximation
    that *sounds* the most like the real signal

Now is the time to use psychophysics

Instead of computing the energy of the difference
    we use a perceptual measure of similarity
        that takes masking, etc. into account

G.729

The standard speech compression in cellular
and very popular in Internet and other applications

Encodes 10 ms frames (100 frames per second) into 80 bits
resulting in 8000 bits per second

filter coefficients	**18 bits**	pitch	**8 bits**		
gain CB	**14 bits**	adaptive CB	**5 bits**		
pulse positions	**26 bits**	pulse signs	**8 bits**	parity check	**1 bit**

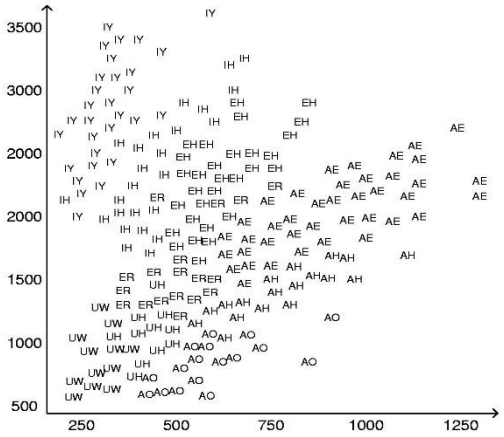
Speech recognition

You might think that speech recognition could be performed thus:

- extract speech features (e.g., LPC, Cepstrum) for speech frames
- classify several successive frames into phonemes
- combine phonemes into words

Unfortunately, that doesn't work for many reasons

- phonemes are pronounced differently depending on context
- phonemes are pronounced differently depending on accent
- phonemes may not be pronounced at all in rapid speech
- nonlinear time warping



Backtracking

The lower levels of the human auditory tentatively identify
the most probable phoneme

But as we continue listening
the higher levels continuously backtrack
and select lower probability ones if needed

To understand this backtracking we can use an analogy
from even higher levels of speech understanding

What does the word FIRE mean in the following sentence?

THE MAN SHOUTED

FIRE !

FIRE THE GUN !

FIRE THE GUN MAKER !

Why don’t we notice this?

All of this happens unconsciously in the background

- but involves higher levels of the cortex

Example 1

- People who do not understand a language
  - can not reliably transcript phonemes in that language

Example 2

- People who understand a language well
  - can successfully understand in very low SNR

Example 3

- What am I saying?
- HOW TO RECOGNIZE SPEECH
- HOW TO WRECK A NICE BEACH

Levenshtein distance

Automatic typing correction mechanisms and spelling correction
find the closest known word to an unknown one

How do we define closeness?

The Levenshtein distance between 2 strings (not necessarily equal length)
is defined as the minimal cost to transform 1 string to the other

Where each of the following has a defined cost

- deletion digital → digtal
- insertion signal → signall
- substitution processing → prosessing
  - in typing the cost may depend on how close the keys are
  - in spelling correction the cost may depend on the sounds

What is the unity weighted Levenshtein distance
between digital and dijtal?

How do we find the minimum cost?

Levenshtein distance (unity weighting)

What is the Levenshtein distance between **prossesing** and **processing** ?

		p	r	o	c	e	s	s	i	n	g
p											
r											
o											
s											
s											
e											
s											
i											
n											
g											

Rules:

1 enter square
from left (deletion) cost = 1

2 enter square
from under (insertion) cost = 1

3a enter square
from diagonal
and same letter cost = 0

3b enter square
from diagonal
and different letter
(substitution) cost = 1

4 Always use minimal cost

Levenshtein distance - cont.

Start with 0 in the bottom left corner

g	9									
n	8									
i	7									
s	6									
e	5									
s	4									
s	3									
o	2									
r	1									
p	0	1	2	3	4	5	6	7	8	9
	0	p	r	o	c	e	s	s	i	n	g

Levenshtein distance - cont.

Continue filling in table

g	9									
n	8									
i	7									
s	6									
e	5									
s	4	3	2	2	2					
s	3	2	1	1	2					
o	2	1	0	1	2					
r	1	0	1	2	3					
p	0	1	2	3	4	5	6	7	8	9
0	p	r	o	c	e	s	s	i	n	g

Note that only local computations and decisions are made

Levenshtein distance - cont.

Finish filling in table

		p	r	o	c	e	s	s	i	n	g
	0	1	2	3	4	5	6	7	8	9
p	1	0	1	2	3	4	5	6	7	8
r	2	1	0	1	2	3	4	5	6	7
o	3	2	1	1	2	2	3	4	5	6
s	4	3	2	2	2	2	2	3	4	5
s	5	4	3	3	2	3	3	3	4	5
e	6	5	4	4	3	2	3	4	4	5
s	7	6	5	5	4	3	3	3	5	5
i	8	7	6	6	5	4	4	4	3	4
n	9	8	7	7	6	5	5	5	4	3
g										

The global result is 3 !

So the Levenshtein distance is 3

Levenshtein distance - end

Backtrack to find path actually taken

		p	r	o	c	e	s	s	i	n	g
	0	1	2	3	4	5	6	7	8	9
p	1	0	1	2	3	4	5	6	7	8
r	2	1	0	1	2	3	4	5	6	7
o	3	2	1	1	2	2	3	4	5	6
s	4	3	2	2	2	2	2	3	4	5
s	5	4	3	3	2	3	3	3	4	5
e	6	5	4	4	3	2	3	4	4	5
s	7	6	5	5	4	3	3	3	5	5
i	8	7	6	6	5	4	4	4	3	4
n	9	8	7	7	6	5	5	5	4	3
g										

Remember:

The question is always how we got to a square

We see that the distance is 3

Since, e.g.,

1. **substitution**
2. **insertion**
3. **deletion**

Generalization to DP

If not all substitutions are equally probable
then we add the cost function instead of 1

We can also have costs for insertions and deletions

$$D_{i\,j} = \min\left(D_{i-1\,j} + C_{i-1\,j;\,I\,j};\quad D_{i-1\,j-1} + C_{i-1\,j-1;\,I\,j};\quad D_{i\,j-1} + C_{i-1\,j;\,I\,j} \right)$$

And even more general rules are often used

This algorithm is called Dynamic Programming
and many other names

- Viterbi algorithm
- Levenshtein distance
- **D**ynamic **T**ime **W**arping

but they are all basically the same thing!

The algorithm(s) are computationally efficient
since they find a **global** minimum based on **local** decisions

DTW

DTW uses the Dynamic Programming technique
for matching spoken words

The input is a feature vector (e.g., LPC, cepstrum)
and is separately matched to each dictionary word feature vector
and the closest word is the winner!

The cost for each substitution needs to represent
how similar the two sounds *sound*

In isolated word recognition systems
energy contours are used first to isolate the words
linear time warping is then used to normalize the duration
but special mechanisms are used for endpoint location flexibility

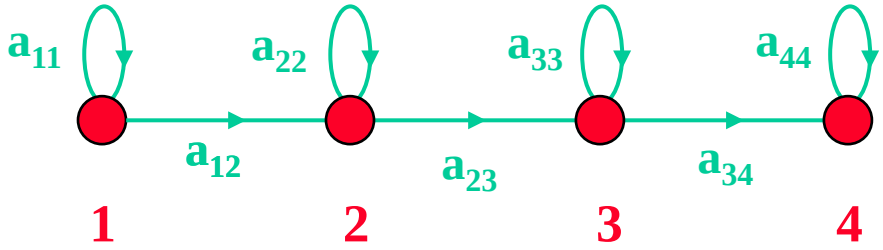
In connected word recognition systems
the endpoint of each recognized utterance is used
as a starting point for searching for the next word

In speaker-independent recognition systems
multiple templates are used for each reference word

Markov Models

An alternative to DTW is based on **Markov Models**

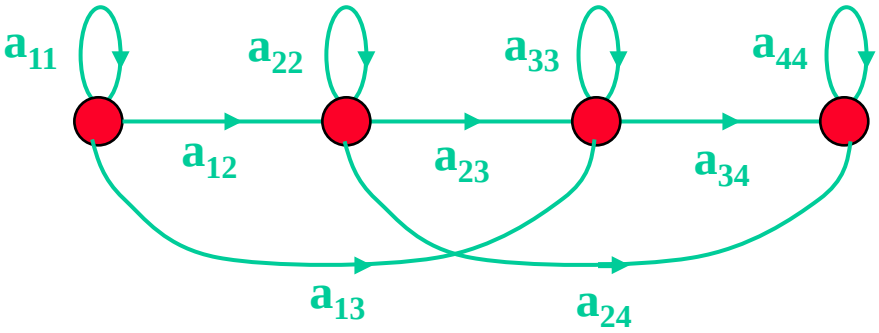
A discrete-time left-to-right first order Markov model



probabilities

$a_{11} + a_{12} = 1,$
$a_{22} + a_{23} = 1,$
etc.

A DT LR second order Markov model

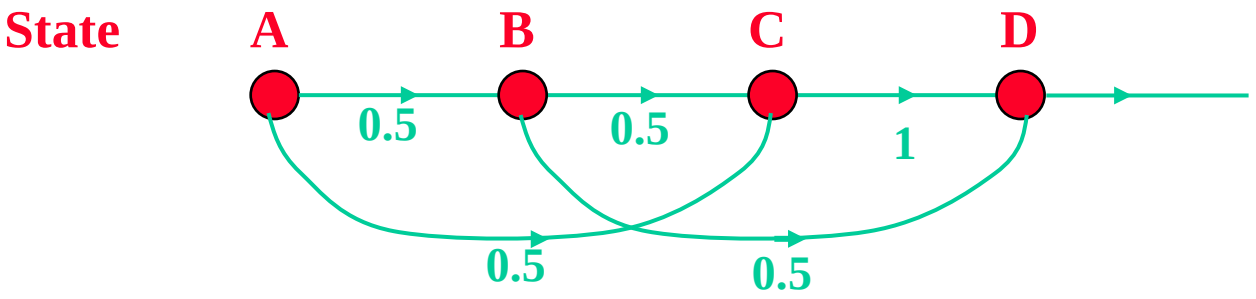


$a_{11} + a_{12} + a_{13} = 1,$
$a_{22} + a_{23} + a_{24} = 1,$
etc.

In Markov models future states depend only on the current state

What is the probability of remaining stuck in a state?

Example

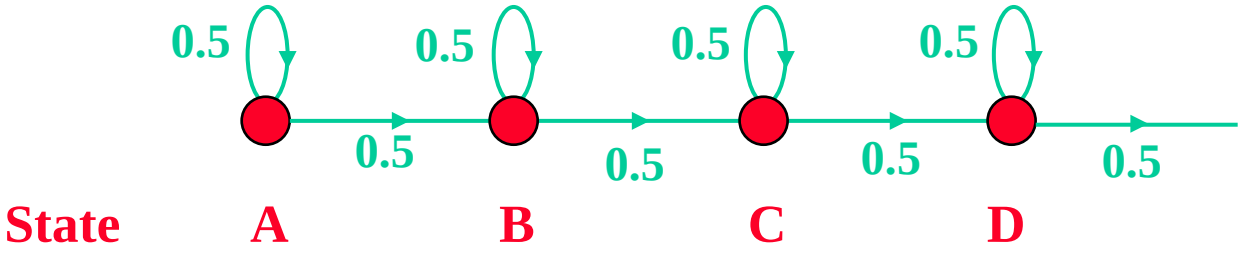


Which strings are possible? What are their probabilities?

- ABCD p = 0.5*0.5*1 = 0.25
- ABD p = 0.5 * 0.5 = 0.25
- ACD p = 0.5 = 0.5

Note that the probabilities sum to 1 (of course!)

Example



Which strings are possible?

- ABCD
- AABBCCDD
- ABBBBBBBCD
- ACBD
- ABBCDCCD
- AAAAA....

Which string is most probable?

What is its probability?

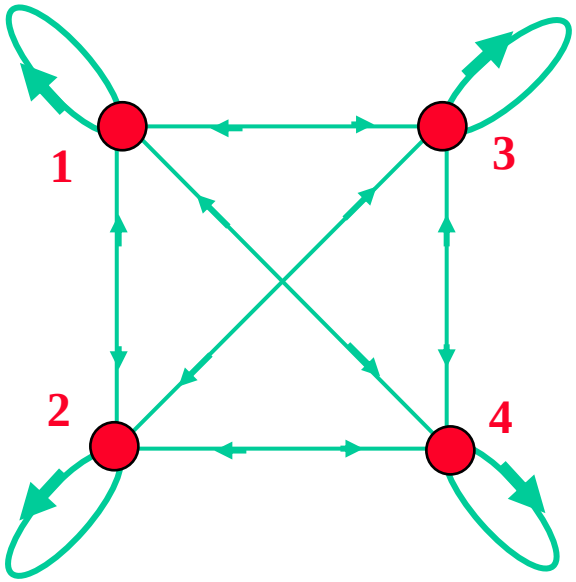
Which strings are 2nd most probable?

What are their probability?

What is the probability of an infinite string?

Markov Models - cont.

General DT Markov Model



Model jumps from state to state with given probabilities

e.g. 1 1 1 1 2 2 3 3 3 3 3 3 3 4 4 4

or 1 1 2 2 2 2 2 2 2 2 2 4 4 4

Markov Models - cont.

Why use Markov models for speech recognition?

- states can represent phonemes (or whatever)
- different phoneme durations (but exponentially decaying)
- phoneme deletions using 2nd or higher order

So time warping is automatic !

How do we use it?

We first build a Markov model for each word

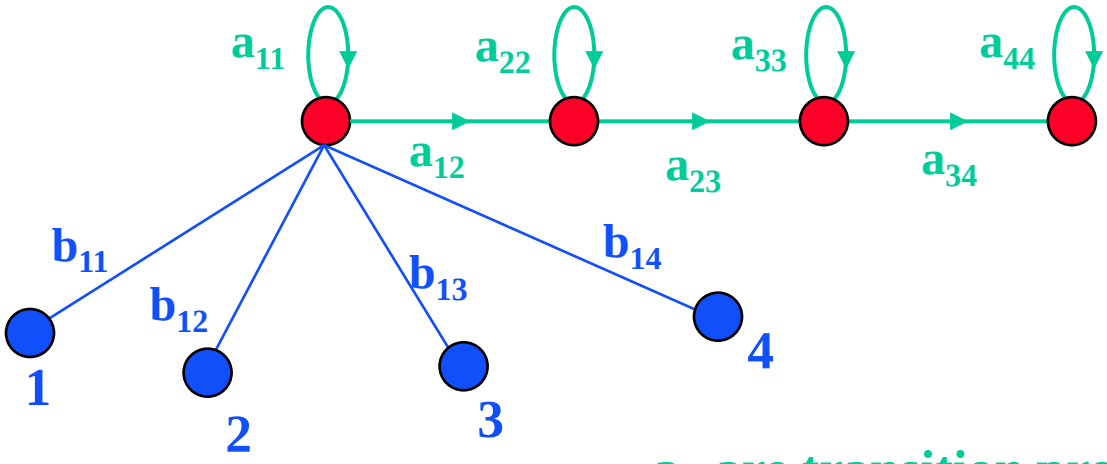
Then, given an utterance

we select the most probable word

HMM

But the same phoneme can be said in different ways

so we need a ***H**idden* **M**arkov **M**odel



acoustic phenomenon

a_{ij} are transition probabilities

b_{ik} are observation (output) probabilities

$b_{11} + b_{12} + b_{13} + b_{14} = 1,$

$b_{21} + b_{22} + b_{23} + b_{24} = 1,$

etc.

HMM - cont.

For a given state sequence $S_1\ S_2\ S_3\ \ldots S_T$
the probability of an observation sequence $O_1\ O_2\ O_3\ \ldots O_T$
is $P(O|S) = b_{S_1O_1}\ b_{S_2O_2}\ b_{S_3O_3} \cdots b_{S_TO_T}$

For a given hidden Markov model $M = \{ \text{states}, a, b \}$
the probability of the *state* sequence $S_1\ S_2\ S_3\ \ldots S_T$
is (the initial probability of S_1 is taken to be p_{S_1})

$$P(S|M) = p_{S_1}\ a_{S_1S_2}\ a_{S_2S_3}\ a_{S_3S_4} \cdots a_{S_{T-1}S_T}$$

So, for a given hidden Markov model M
the probability of an *observation* sequence $O_1\ O_2\ O_3\ \ldots O_T$
is obtained by summing over all possible state sequences

HMM - cont.

$$P(O|M) = \sum P(O|S)\, P(S|M)$$

$$= \sum \pi_{s_1} b_{s_1 o_1} a_{s_1 s_2} b_{s_2 o_2} a_{s_2 s_3} b_{s_2 o_2} \cdots$$

So for an observation sequence we can find

the probability of any word model

(but this can be made more efficient using the forward-backward algorithm)

How do we train HMM models?

The full algorithm (Baum-Welch) is an EM algorithm

An approximate algorithm is based on the Viterbi (DP) algorithm

(Sorry, but beyond our scope here)

The new world

DTWs and HMMs were the state of the art until recently

DTWs have little training time but long run times

HMM have long training times but short run times

Recently Deep Neural Networks have taken over many applications

DNNs

- use essentially no knowledge about the speech signal
- require astronomical amounts of data to train
- have very long training times
- provide no reasoning as to their decisions
- can frequently outperform classical methods
  but can also fail catastrophically
  and can be subject to *adversarial attacks*

Communications signal processing

Application: Data Communications

Communications is moving information from place to place

Information is the amount of surprise, and can be quantified!

Communications was originally analog – telegraph, telephone

There are many forms of digital communications that we use every day

- Mobile (cellular)
  - Internet
  - from 4G *voice* is simply packet data
- Television
  - DVB (Idan plus)
  - Cable TV (DOCSIS)
- Home Internet access
  - DSL (ADSL, VDSL)
  - Cable modem (DOCSIS again)
- Internet of Things
  - smart home / smart city
  - Industry 4.0

Digital Communications

All physical channels
- have limited bandwidth (BW)
- add noise (so that the signal to noise ratio SNR is finite)

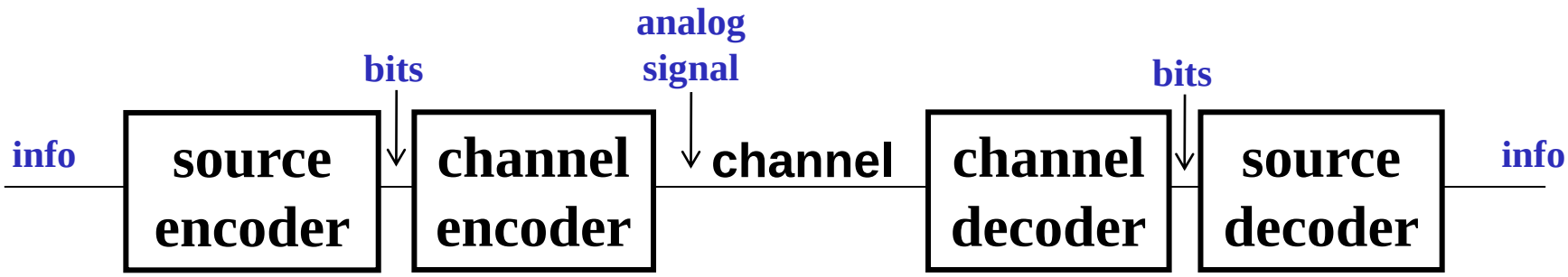
so analog communications always degrades
and there is no way to completely remove noise

In analog communications the only solution to noise
is to transmit a stronger signal
since later amplification amplifies N along with S

Communications has become digital
- digital communications is all or nothing
perfect reception or no data received

Shannon's Theorems

1. **Separation Theorem**



2. **Source Encoding Theorem**

   Information can be quantified (in bits)

3. **Channel Capacity Theorem**

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

Warning: Shannon's Laws are not constructive
they only give us targets to aim for!

Modulation

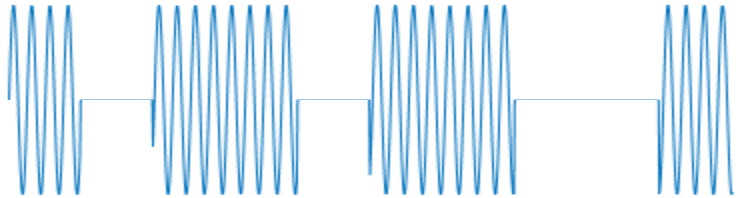
We already learned that modulation means
changing parameters of a signal in order to carry information

We previously saw AM and FM in which
the signal carried analog information

Shannon leads us to ask how a signal can carry digital information

A simple example would be to modulate the amplitude of a DC signal
allowing it to be 0 or 1

We could also modulate the amplitude of a sinusoid (like in AM, but 0/1)



Since we will often need bidirectional transfer of information
it is convenient to package the modulator and demodulator
in a single box called a modem

(A similar box for analog modulation is called a transceiver)

Digital communications

Shannon's theorems taken together
explain why all communications has become digital

Analog communications always have quality degradation

Vinyl records are very limited in spectral and dynamic range
and add a tremendous amount of noise

A Compact Disk is essentially undistinguishable form the original

When recording from tape (e.g., cassette) to tape
each successive recording is noisier, until only noise remains

Copying CD to CD the 1000^{th} copy is indistinguishable from the 1^{st}

Scratching a CD/DVD, either there is no effect
or nothing works at all (in video there can be missing *squares*)

The simple reason

Digital communications inherit a fundamental property of the bit

it is either 0 or 1 – on of off – perfect or nothing

Either you receive exactly what was sent

or you receive nothing (complete garbage)

But why is it perfect and not always nothing?

If I transmit a voltage of 3.14159... volt, but you receive 2.71828... volt

even if I tell you that there was noise and that isn't what I sent

you can't possibly figure out the right value

If I transmit a bit 1 and we obey Shannon's laws

you should receive a bit 0

But even if you don't and I tell you that you received the wrong value

then you know that the correct answer is 1 (error correction)

Separation Theorem

The separation theorem is logically the first
even if chronologically the last

The theorem states that the optimal method of transporting
any kind of information (analog or digital) involves

- *properly* converting it into digital format (Shannon's 2nd theorem)
- *properly* modulating an analog signal (Shannon's 3rd theorem)
- recovering digital information from the received analog signal
- recovering original format information from the digital information

Note that

- the source decoder is the inverse of the source encoder
  input with the digital format it reproduces the original information
- the channel decoder is the inverse of the channel encoder
  input with the undistorted transmitted signal
  it reproduces the digital format information
- the channel encoder is designed with knowledge of the channel

Channels

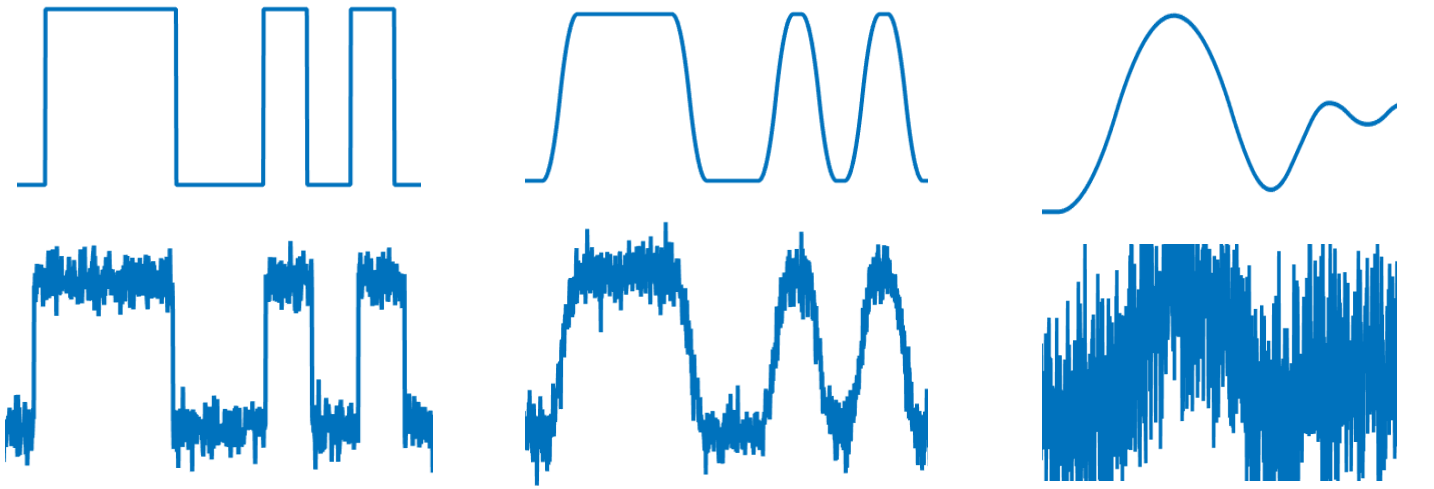
The problem with communications is that all physical channels distort the signals traversing them

So that the signal received is not the same as the signal transmitted

Were it not for this distortion communications would be trivial and there would be no limit on the possible (bit-)rate

From physics we know that all channels:

- add noise (at least some dependent on temperature)
- limit bandwidth (may be low-pass or bandpass)



bandwidth limited

with noise

Information

What is *information* anyway?

Information is the amount of surprise

there is no information in yesterday's newspaper

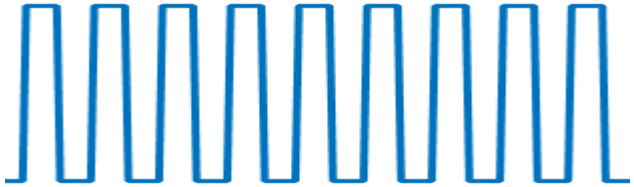
since we know everything and it doesn't surprise us any more

Similarly, a constant sinusoidal signal carries no information

since it is deterministic – there is no surprise

This deterministic On/Off signal taking only 1 of two values

carries no information



But this one *does* carry information (it is stochastic - it surprises us!)

Shannon's bits

Shannon taught us that information can be *quantified*

You can compare the amount of information in a book
to that in a piece of music or in a picture
just like you can compare the weight of lead and feathers

The unit often used is the *bit*

Beware, Shannon's bit is more general than Tukey's **bi**nary digi**t**

Tukey's bit is only useful for encoding numbers
while Shannon's bit can be used for any form of information

Consider the set of all items of interest
(e.g., all news items, all types of animals, all letters in an alphabet)

How much information is involved in selecting one item from this set?

Shannon's bit is defined as follows:

- ask questions with yes/no (binary) answers until you find out
- the minimum number of questions required
is the information content in bits

A simple example

Children play a game called **21 questions**

One player (the puzzler) thinks of something – say an elephant

The other player (the questioner) asks yes/no questions

If he finds out in 21 questions or less the questioner wins

(and proves that the puzzler's brain contains < 21 bits of information)

otherwise the puzzler wins

1. Is it an animal? YES
2. Is it bigger than a bread-box? YES
3. Does it live in the water? NO
4. Does it live in Africa? YES
5. Is it a lion? NO
6. Is it an elephant? YES

The questioner wins, but this is not the optimal set of questions

The optimal strategy asks questions

that divide the remaining possibilities in half!

(assuming equally probable items)

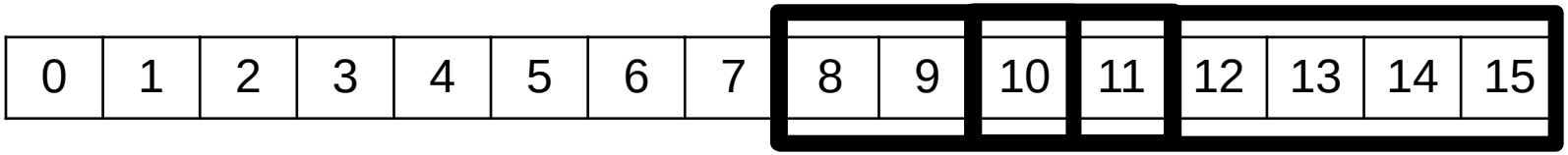
When both bits are the same

When are Tukey's bits the same as Shannon's?

Assume that the set of all possibilities consists of the numbers between 0 and 15

The puzzler thinks of the number 10

The questioner asks the optimal set of questions as follows:



1. Is greater than 7 ? YES = **1**
2. Is greater than 11? NO = **0**
3. Is it greater than 9 ? YES = **1**
4. Is it greater than 10? NO = **0**

So the Shannon information quantity is 4 bits
and the answers form the Tukey binary number for 10 – **1010**

What should the questioner do if the items are not equally probable?

Source Encoding Theorem

What does this have to do with communications?

Shannon tells us that we need to first encode the information
according to Shannon's 2nd law

Example: if the current letter in English text is Q
how much information is there in the fact
that the following letter is U?

Exactly zero! So we shouldn't waste bits on it!

Example: assume a language with 4 letters A, B, C, and D

If they are all equally probable
then we can encode 00, 01, 10, 11 – 2 bits per letter!

But if p(A) = 1/2 p(B) = 1/4 p(C)=p(D)=1/8
how should we encode the letters?

A=1 B=01 C=001 D=000

Average bits per letter = 1/2*__1__ + 1/4*__2__ + 1/8*__3__ + 1/8*__3__ = 1.75 bits

This is smaller – and the minimal amount!

Shannon information

We can generalize the last example

If we have N possible symbols, each of which has probability p_i

then the information per symbols is given by

$$< \log\left(\frac{1}{p_i}\right) > = -\sum_{i=1}^{N} p_i \log p_i$$

If we use the base-2 logarithm (physicists use ln and call this *entropy*)
we get the information in *bits* per symbol

If there are 2^m equally probable symbols (each with $p=2^{-m}$)
the information of selecting 1 symbol is (obviously) m bits

The information content of a string of K *independent* symbols
is K times the information of a single symbol

What is the entropy in bit/symbol of our previous example?

p(A) = 1/2 p(B) = 1/4 p(C)=p(D)=1/8

$1/2 * \log_2(2) + 1/4 \log_2(4) + 1/8 \log_2(8) + 1/8 \log_2(8) = 1.75$ bits/symbol

Shannon encoding

Shannon's 2nd law is not constructive –
it tells us how many bits are required to capture information
but doesn't teach us how to actually encode information

So generations of mathematicians and engineers have developed coding methods that approach Shannon's information *limit*

- Ziv and Lempel developed a digital file compression method that approaches Shannon's limit asymptotically (for large files)
- the speech compression methods we already learned are *lossy* methods of Shannon encoding for speech signals
- the **J**oint **P**hotographic **E**xperts **G**roup have developed a *lossy* mechanism to encode images
- the **M**otion **P**ictures **E**xpert **G**roup have developed many *lossy* mechanisms for encoding video

If Ziv-Lempel compression can compress a file to 1/5 its size
why can't we recompress that file to 1/25 the original?

Error detection/correction

Information theory teaches us how to optimally encode information
and it also teaches us how to protect information from errors

You all know about parity bits

For example, we can add 1 bit to each byte making the number of 1s even
this adds 12.5% overhead
and detects any single bit error
but can't correct any errors

01011001	0
01001010	1
00100000	1
01010011	0
01110100	0
01100101	0
01101001	0
01101110	1

Building on this we can make a simple error correction technique
that adds 16 bits to 64, i.e., 25% overhead
but can detect any two errors
and correct any single error

01011001	0
01001010	1
00100000	1
01010011	0
01110100	0
01100101	0
01101001	0
01101110	1

01110110

What happens if there two errors in the same row or column?
What happens if the error is in the parity check bits?

There are much more sophisticated error correction schemes!

To FEC or not to FEC

Shannon's separation theorem implies
that it is suboptimal to use *data-layer* **E**rror **C**orrection schemes

This is because an ECC always adds extra non-information bits
reducing the information bit rate

Yet, many communications systems use error detection/correction

- CRC (Cyclic Redundancy Code) in Ethernet
- turbo codes in cellular communications
- Reed Solomon FEC in ADSL and optical networks

This is because unlike the assumptions of the capacity theorem
the SNR is often not stationary due to *noise events* and *fading*
resulting in errors in the recovered information

Signal layer ECC methods (e.g., TCM)
don't contradict the separation theorem

Channel capacity theorem

Shannon's third theorem states that there is an upper limit
on the data rate in which bits can be transferred in a channel

This is the main reason a 100 Mbps Internet connection
costs more than a 20 Mbps one!

If there were no such limit then
why not give the customer as much as the routers can forward?

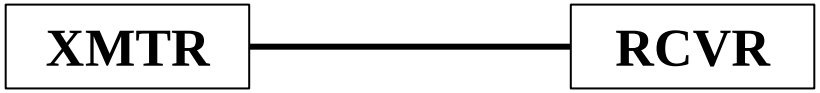
This capacity depends on

- the Signal to Noise ratio
  the amplitude of the modulated signal
  divided by the amplitude of the noise added by the channel
- the bandwidth of the analog channel (in Hz)
  which is why communications people frequently say
  bandwidth instead of *data-rate*!

Since this is a theorem in signal processing we are going to *prove* it
Well, actually our proof has a lot of holes
one day you should read Shannon's real proof – just for the beauty of it!

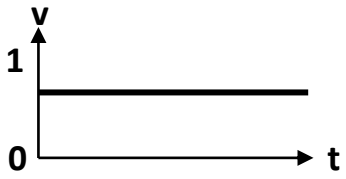
Channel Capacity (1)

Let's first prove that a channel with no noise has infinite capacity
even if its bandwidth is close to zero



Let's take some large amount of information

- source encode it into bits 10111010010111100010101000010...
- add "0." at the beginning 0.10111010010111100010101000010...
  to obtain a number between 0 and 1
- place precisely that voltage as DC on the *wire*
  connecting transmitter to receiver

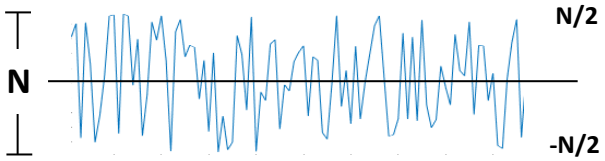


- since there is no noise precisely this voltage will be received
  even though the bandwidth is small (DC requires no bandwidth!)
- there is no physical limit on how fast this voltage can be measured

So we can transfer a large amount of data in negligible time
which means the capacity is infinite

Channel Capacity (2)

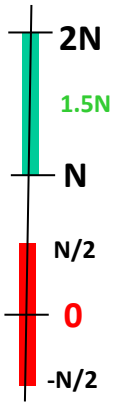
Now let's prove that a channel with infinite bandwidth
has infinite capacity even if it has noise of peak-to-peak value N



DC-less noise with uniform distribution

Let's take some large amount of information

- source encode it into bits 10111010010111100010101000010...
- for every 0 bit transmit 0 voltage for T seconds
  for every 1 bit transmit 1.5N voltage for T seconds
- for every 0 bit the receiver will see a voltage between –N/2 and N/2
  for every 1 bit it will see a voltage between N and 2N
- since there is no overlap the receiver can identify 0/1 with no error
- since the bandwidth is infinite, we can send T→0

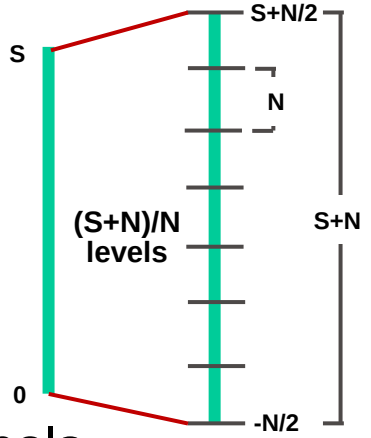


So we can transfer a large amount of data in negligible time
which means the capacity is infinite

Shannon's capacity theorem

If there is both noise and limited bandwidth:

- assume (*wolg*) that we transmit some signal with values between **0** and **S**
- assume that the channel adds DC-less noise with uniform distribution
  we'll call the peak-to-peak noise N
  so that the noise values are between –N/2 and +N/2
- the receiver always sees values between –N/2 and S+N/2
  so the receiver's dynamic range is S+N
- to maximize the information per symbol w/o overlap
  we space the signal levels by N
- so there can be (S+N)/N = S/N + 1 = SNR+1 different symbols
- hence each symbol contains $\log_2(\text{SNR} + 1)$ bits
- but there can be BW symbols per second



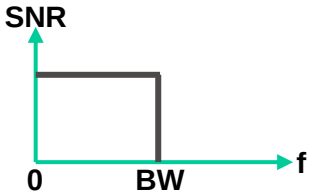
Hence, the maximum information rate $C = \text{BW} \log_2(\text{SNR} + 1)$ bit/s

The maximum spectral efficiency is $C/\text{BW} = \log_2(\text{SNR} + 1)$ bit/s/Hz

Capacity for frequency-dependent SNR

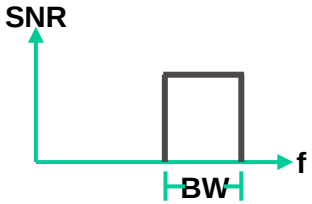
The capacity theorem assumed that the SNR was

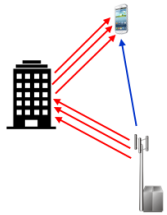
- constant from DC to BW
- zero over BW



or more generally

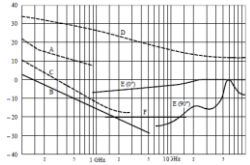
- constant in some passband
- zero outside the passband

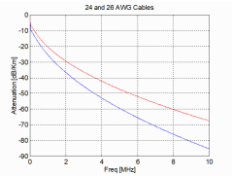




Which will not be the case if either

- the signal attenuation (including multipath cancellation)
- the noise (including interference)





or both, vary with frequency

The extension of the theorem is simple

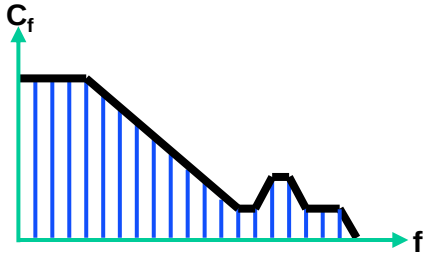
- divide the passband into channels of bandwidth Δf centered at f
- the capacity theorem states that for the channel

$$C_f \approx \log_2(\mathrm{SNR}(f) + 1)\,\Delta f$$

- the composite capacity is $\sum_f \log_2(\mathrm{SNR}(f) + 1)\,\Delta f$

The theorem becomes exact when we send $\Delta f \rightarrow 0$

and then $C = \int \log_2(\mathrm{SNR}(f) + 1)\,df$

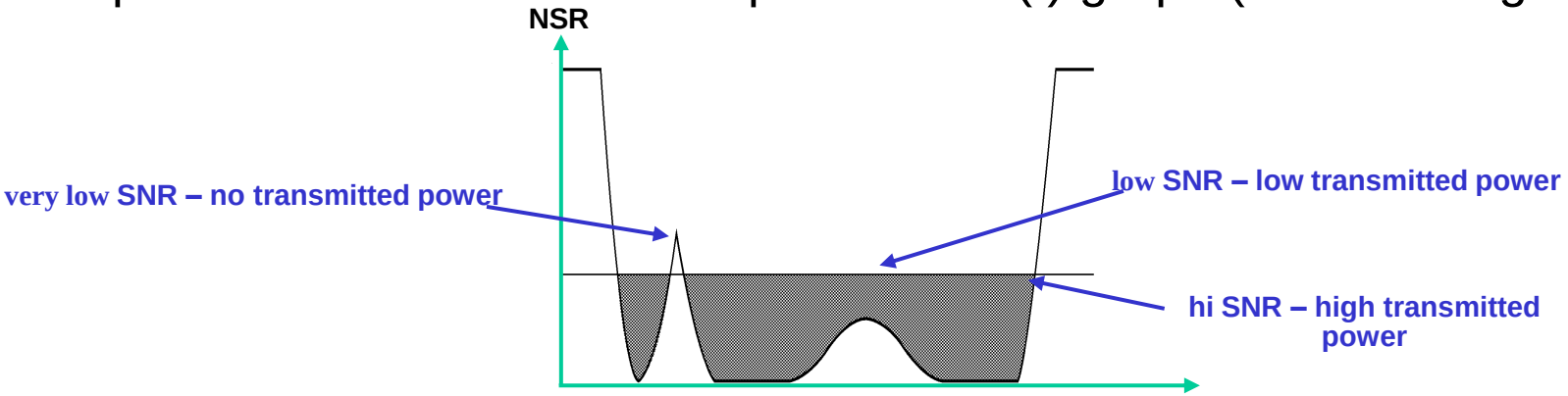


Water pouring theorem

The *converse* to Shannon's channel capacity theorem
is Gallager's water pouring theorem

The spectrum of the optimum (capacity-achieving) modulated signal
is given by the *water pouring* algorithm

- fill a pitcher with a volume of water
  representing the total power to be transmitted
- pour the water over the reciprocal SNR(f) graph (Noise to Signal Ratio)



This power spectrum can be achieved by

- spectral shaping
- explicit power loading

Modem design

Shannon's theorems are existence proofs
not constructive methods to design modulation techniques
that maximize capacity given channel characteristics

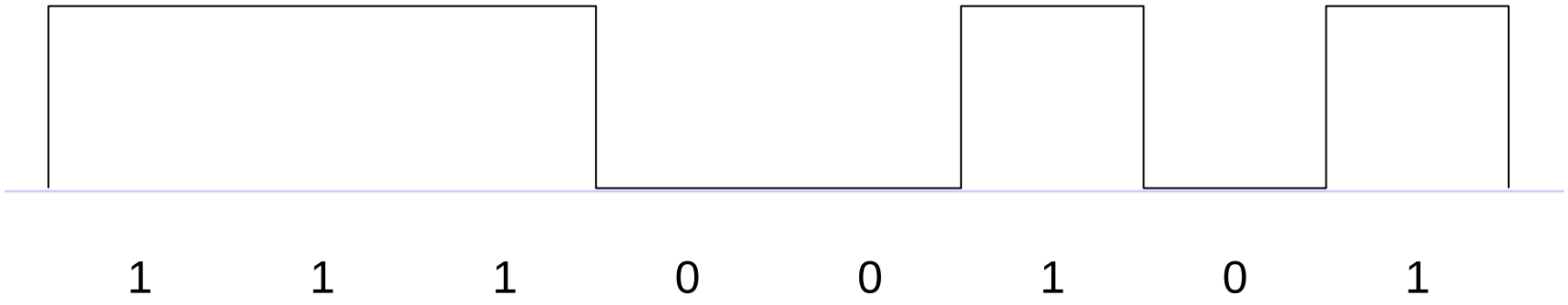
So we need to be creative to reach channel capacity

Over the years many kinds of modems have been designed :

- NRZ
- RZ
- PAM
- FSK
- PSK
- QAM
- OFDM

NRZ

Our first attempt is to simply transmit 1 or 0 (volts?)



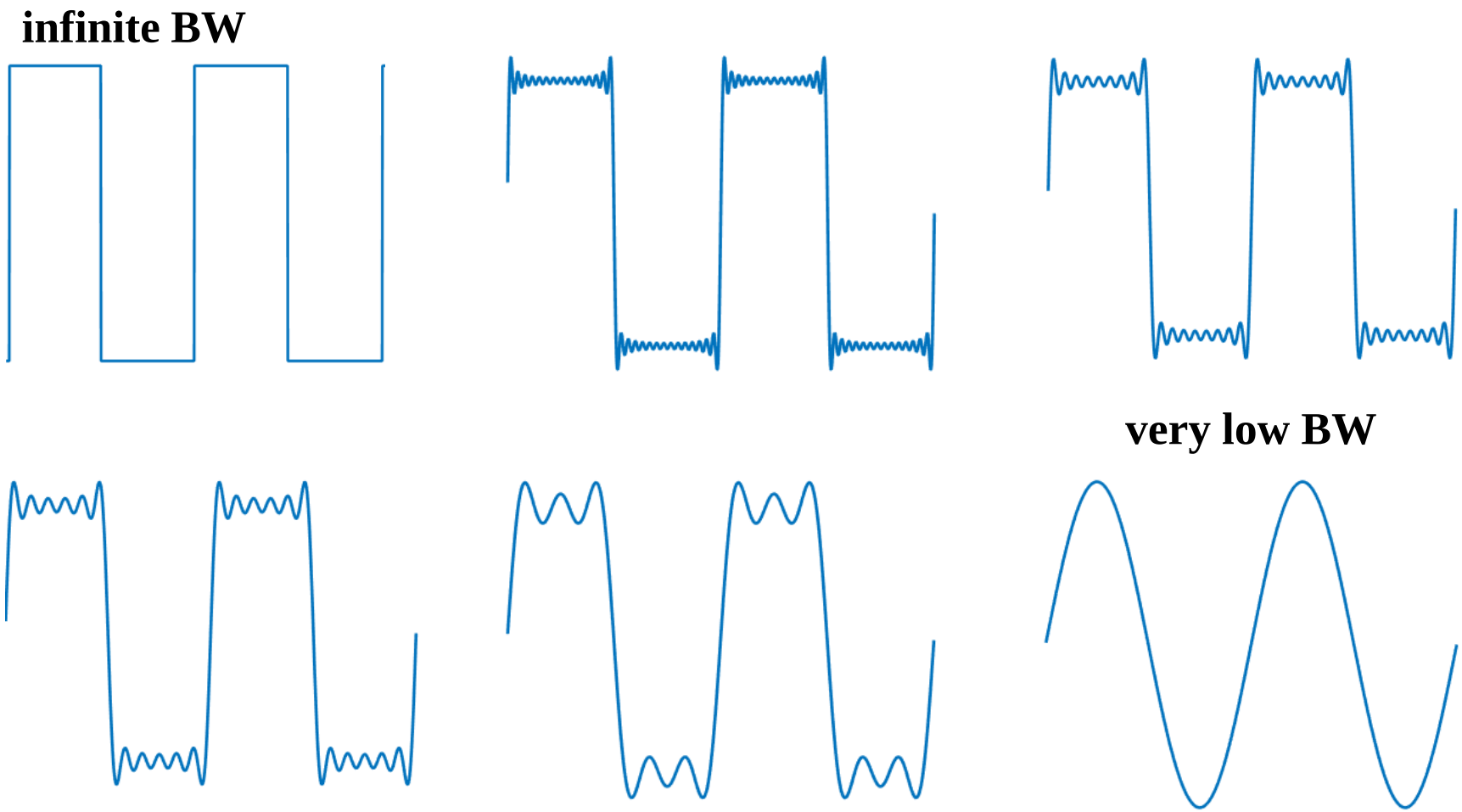
This is called **N**on **R**eturn to **Z**ero (i.e., NOT RZ)

Information rate = number of bits transmitted per second (bps)

But this is only good for short serial cables (e.g. RS232), because

- requires DC (doesn't work over radio or fiber)
- if DC blocked lose runs of 1s
- needs high bandwidth (sharp corners require high frequencies)
- strong **I**nter**S**ymbol **I**nterference
- timing recovery may not be possible (if long *runs* of 0 or 1)

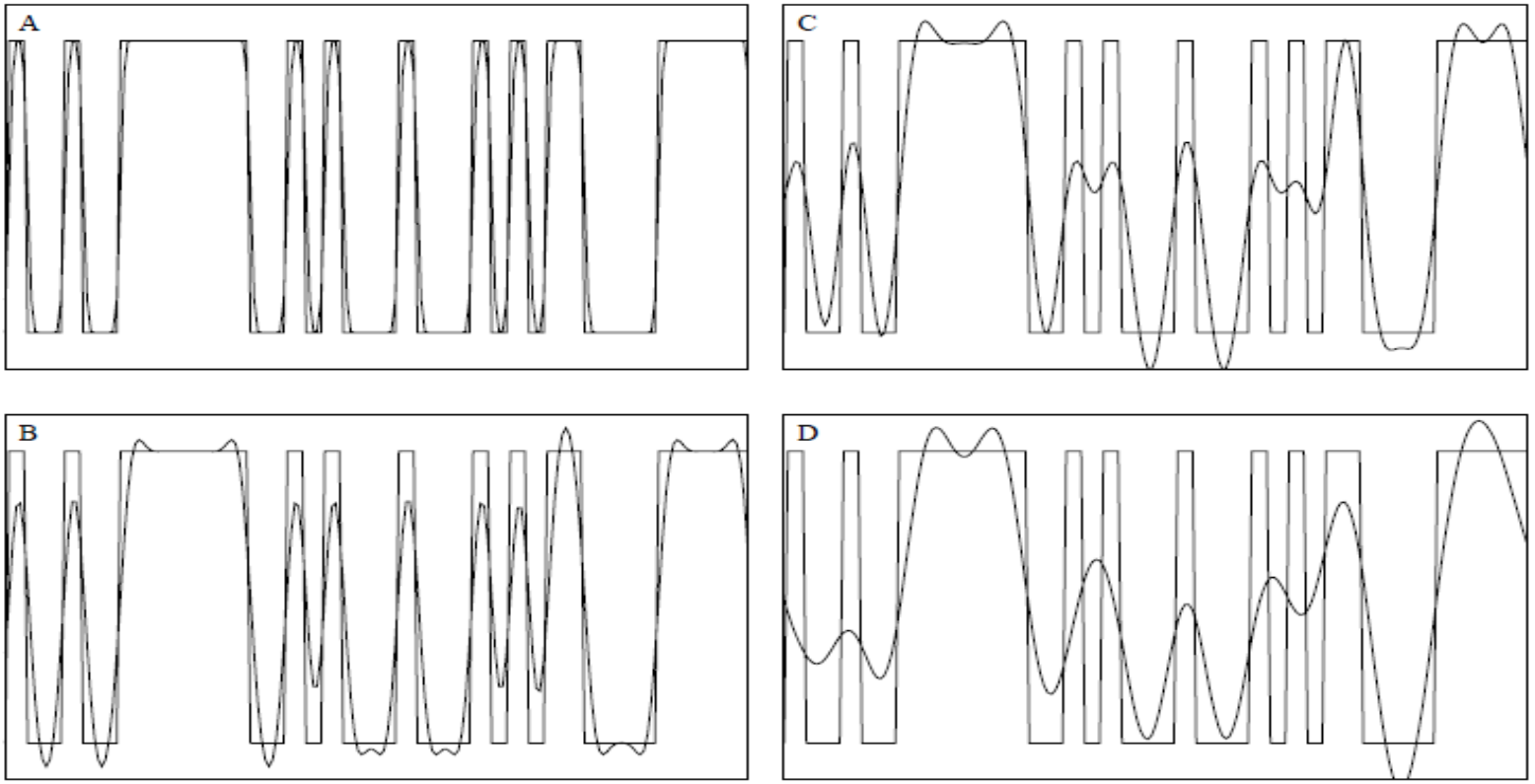
NRZ needs infinite bandwidth



and this is for 0101010101!
for random bits finite BW causes an additional problem!

NRZ InterSymbol Interference (ISI)

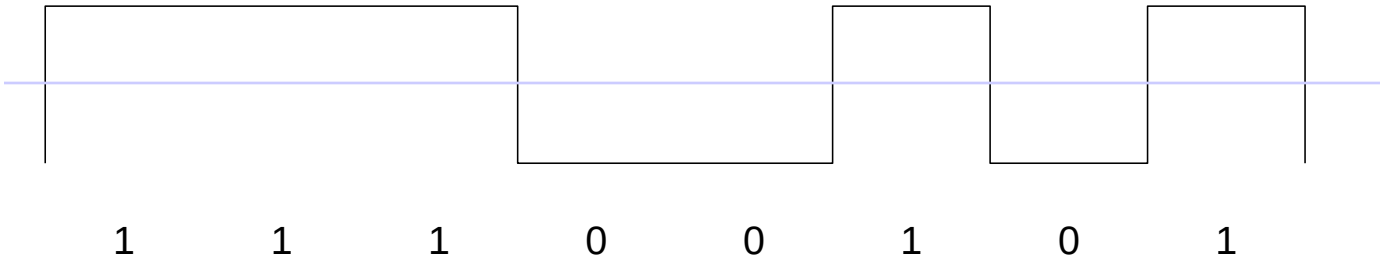
low-pass filtered signal
keeps up with bit changes



insufficient BW to keep up with bit changes

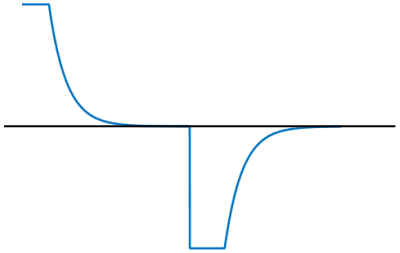
DC-less NRZ

So what about transmitting -1/+1?



This is better, but not perfect!

- DC isn't exactly zero
- if DC blocked worse than before
- still is modulation of DC and not for fiber or radio
- still has high bandwidth (sharp corners)
- even without decay, long runs ruin timing recovery



RZ

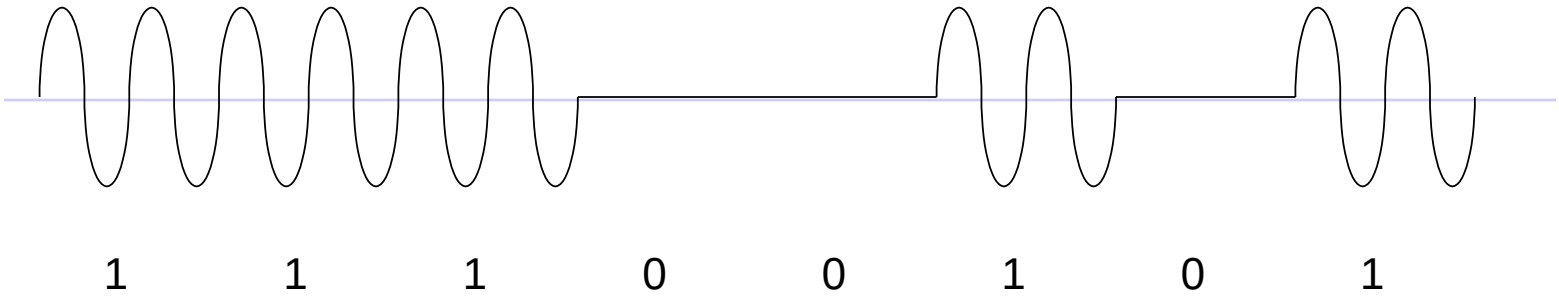
What about **R**eturn to **Z**ero ?

- never long + runs, so DC decay less important
- BUT half-width pulses
  means twice the number of changes per second
  which requires twice the bandwidth!

Shannon strikes !

OOK

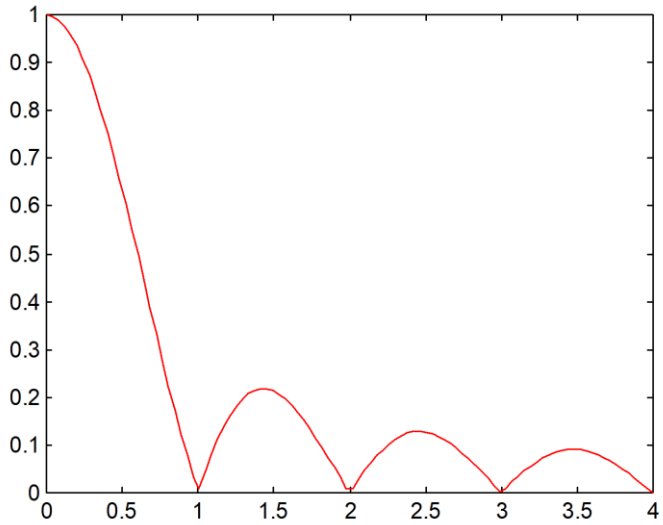
Even better - use OOK (On Off Keying)



- no signal discontinuity (but derivative discontinuities)
- absolutely no DC!
- based on desired sinusoid (“carrier”) frequency
  so suitable for radio and fiber
- can hear it (Morse code)

NRZ - Bandwidth

The PSD (Power Spectral Density) of NRZ is a sinc (sinc(x) = sin(x)/x)

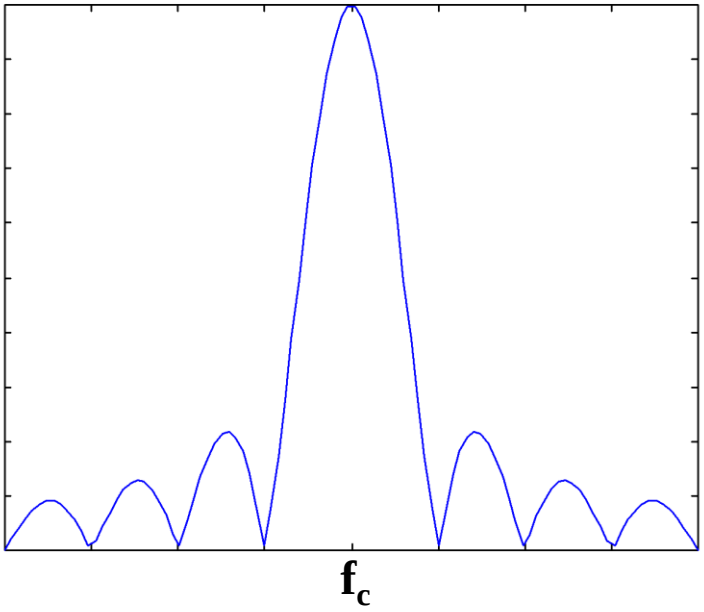


- the first zero is at the bit rate (from the uncertainty principle!)
- so channel bandwidth limits bit rate
- DC depends on level placement (may be zero or spike)

OOK - Bandwidth

PSD of -1/+1 NRZ is the same, except there is no DC component

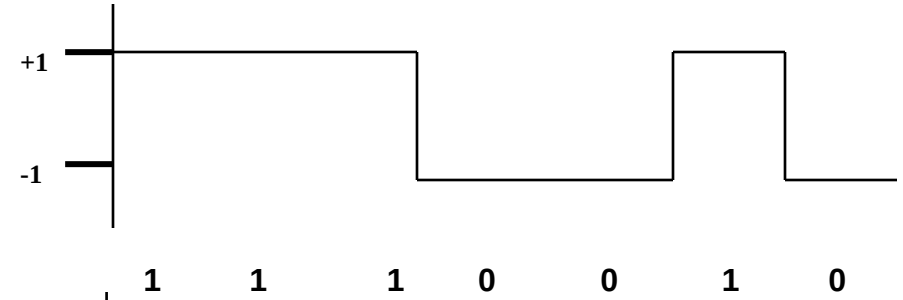
If we use OOK the sinc is mixed up to the carrier frequency



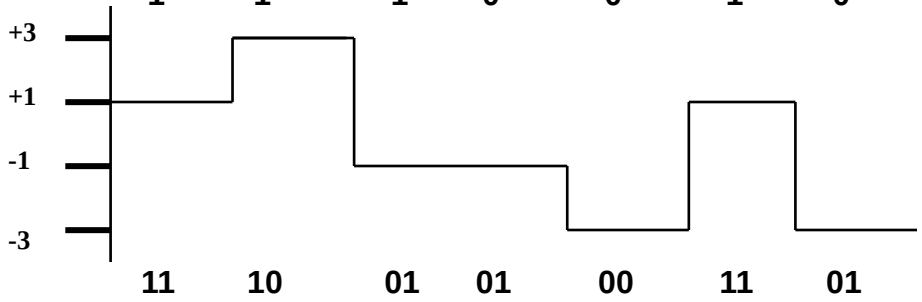
- (The spike helps in carrier recovery)

From NRZ to n-PAM

NRZ

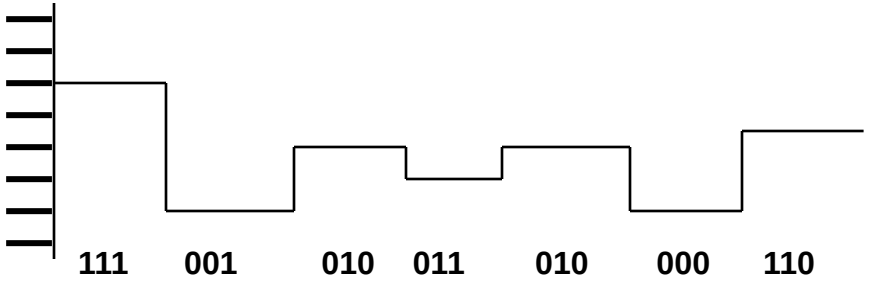


4-PAM
(2B1Q)



GRAY CODE
10 => +3
11 => +1
01 => -1
00 => -3

8-PAM

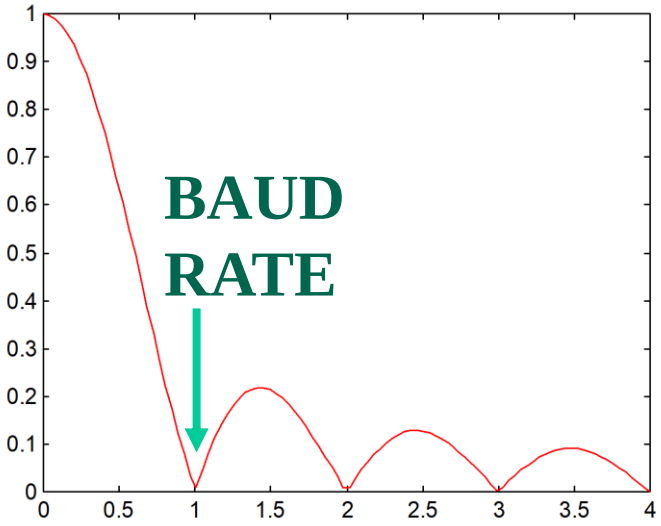


GRAY CODE
100 => +7
101 => +5
111 => +3
110 => +1
010 => -1
011 => -3
001 => -5
000 => -7

- Each level is called a *symbol* or *baud*
- Bit rate = number of bits per symbol * baud rate

PAM - Bandwidth

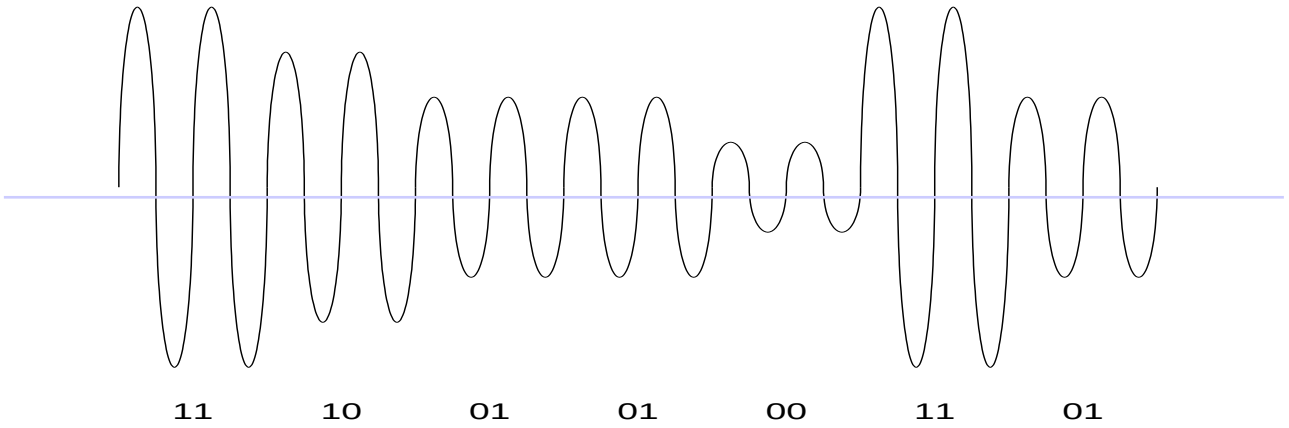
BW (actually the entire PSD) doesn't change
with the number of levels !



So we should use many bits per symbol
But then noise becomes more important
(Shannon strikes again!)

ASK

What about Amplitude Shift Keying - ASK ?



2 bits / symbol

- generalizes OOK in the same way that PAM generalized NRZ
- not widely used since hard to differentiate between levels when there is noise

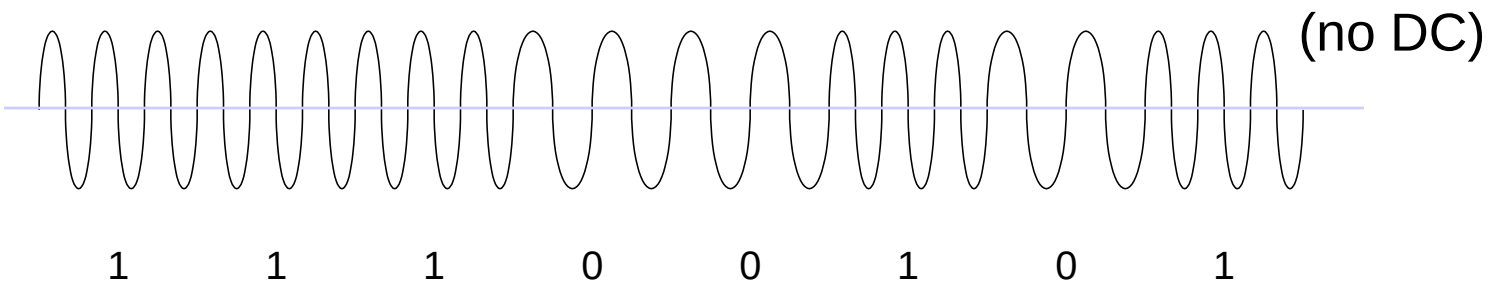
How do we find the amplitude of a sine?

FSK

What can we do about noise?

If we use *frequency diversity* we can gain 3 dB

Use two independent OOKs with the same information



This is FSK - Frequency Shift Keying

We maintain *continuous phase* for minimum bandwidth

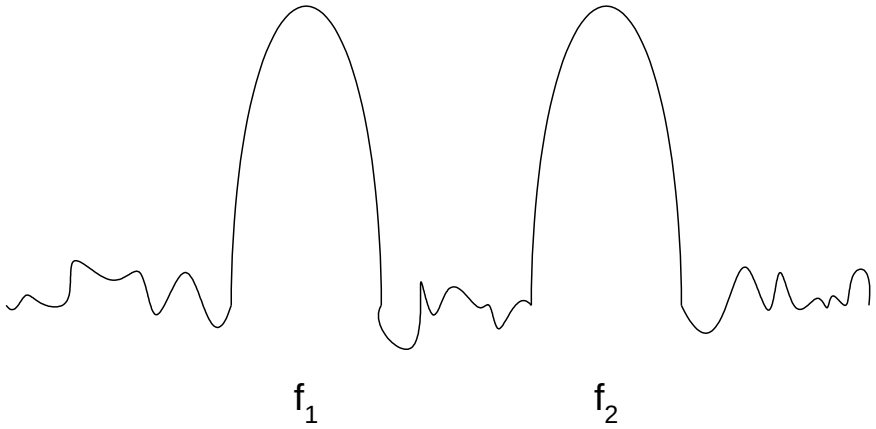
Note that sinusoids are orthogonal – but only over long times !

FSK in the frequency domain

FSK is based on orthogonality of sinusoids of different frequencies
make decision only if there *is* energy at f_1 but *not* at f_2

The uncertainty theorem says that the uncertainty in frequency
is inversely proportional to the time duarion

So when we leave a frequency for a short amount of time
its spectrum is not a line – but a sinc!



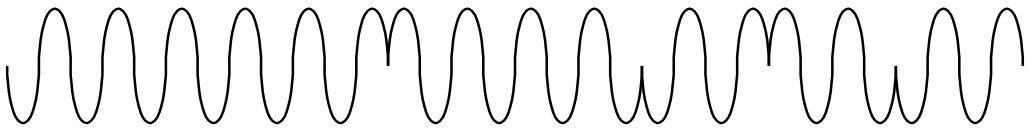
So FSK is robust but slow (Shannon strikes again!)

PSK

What about sinusoids of the same frequency but different phases?

Correlations reliable after a single cycle ($\int \sin(\omega t)\cos(\omega t)\,dt = 0$ on 1 cycle!)

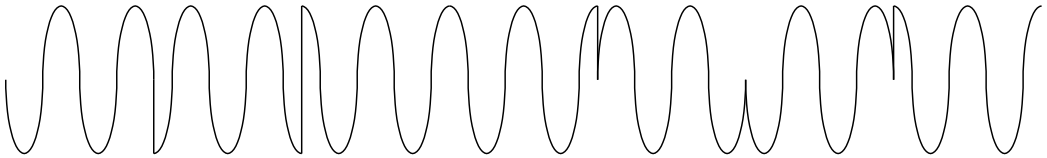
So let's try BPSK



1 bit / symbol

1 1 1 0 0 1 0 1

or QPSK



11 10 01 01 00 11 01

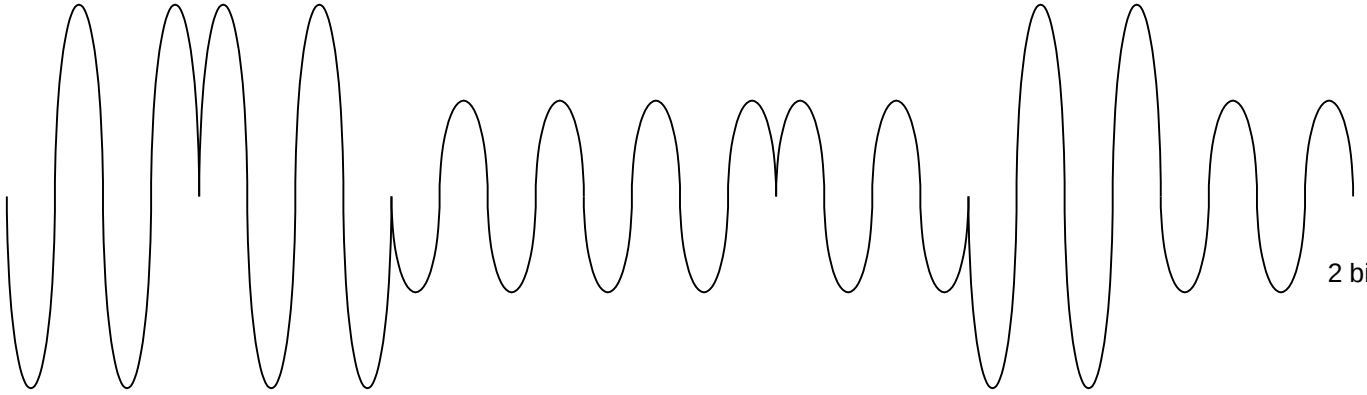
2 bits / symbol

Bell 212 2W 1200 bps

V.22

QAM

Finally, we can combine PSK and ASK (but not FSK)
by changing both the amplitude and the phase



This is called **Q**uadrature **A**mplitude **M**odulation

This is getting confusing

The secret math behind it all

Remember the Hilbert Transform?

$$x(t) = A(t) \cos \left(2 \pi f_c t + \phi(t) \right)$$

$$y(t) = A(t) \sin \left(2 \pi f_c t + \phi(t) \right)$$

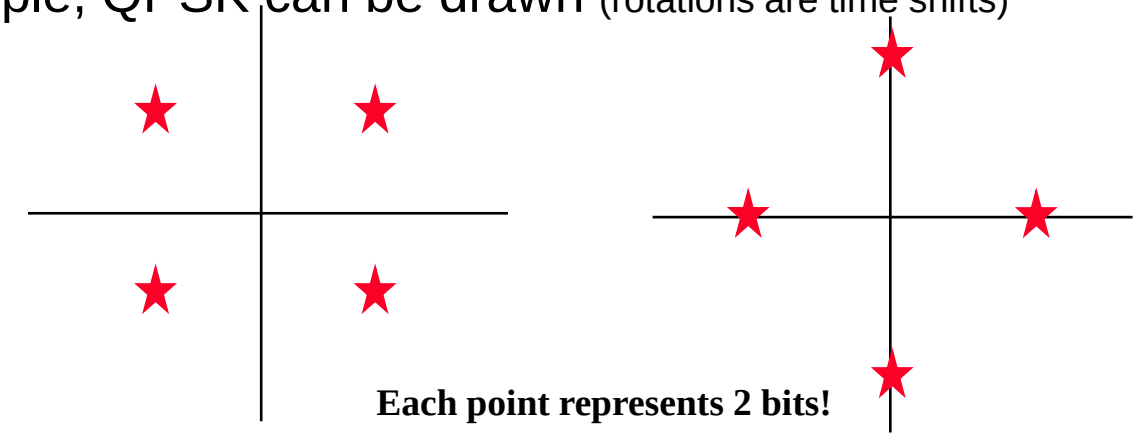
A(t) is the *instantaneous amplitude*, ϕ(t) is the *instantaneous phase*

This can be used to demodulate analog/digital communications signals

For digital modulations we draw constellation diagrams

- x and y as axes
- A is the radius, ϕ the angle

For example, QPSK can be drawn (rotations are time shifts)



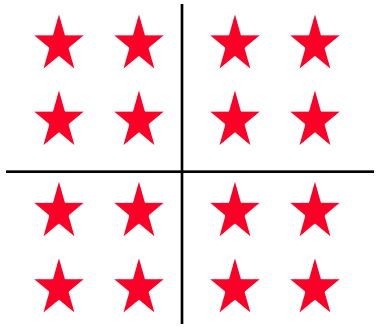
Each point represents 2 bits!

QAM constellations

16 QAM

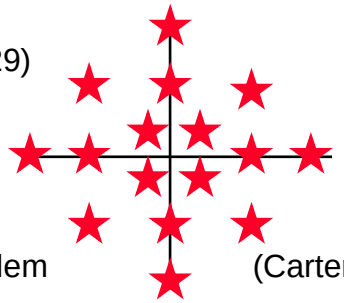
V.22bis 2400 bps

2W



V.29 (4W 9600 bps)

Codex 9600 (V.29)



first non-Bell modem

(Carterphone decision)

Adaptive equalizer

Reduced PAR constellation

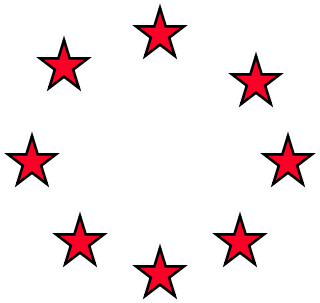
Today - 9600 fax!

8PSK

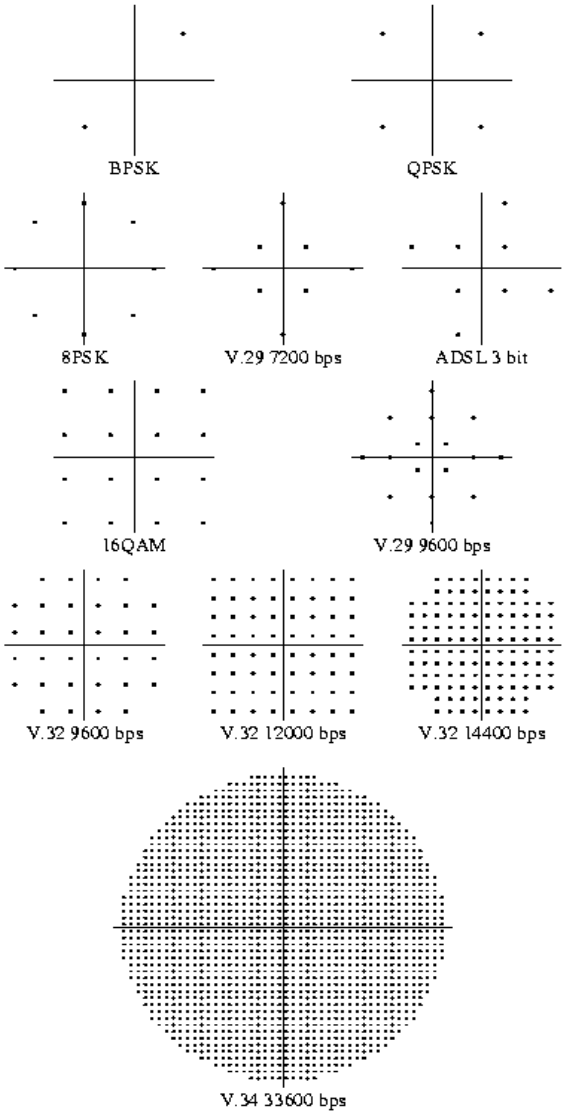
V.27

4W

4800bps



Voicegrade modem constellations



Spectral efficiency of modulations

We defined spectral efficiency as the bit/s per Hz

We can now compare the modulation techniques we have seen so far

modulation	bit/symbol	BW/symbol rate	spectral efficiency
NRZ	1	1	1
BPSK	1	2	0.5
QPSK	2	2	1
8PSK	3	2	1.5
16QAM	4	2	2
64QAM	6	2	3
256QAM	8	2	4

Note that we are using *raw* (DSB) efficiencies

values for PSK and QAM can be improved by a factor of 2 !

FDM

QAM (including PSK as a special case) is a very efficient modulation
approaching the Shannon capacity for simple bandpass channels

But the spectrum of a QAM signal is *inflexible*
like all discretely keyed modulations of a single carrier
it is a sinc centered at f_c with first zeros at $f_c \pm$ symbol-rate

The spectrum can be shaped using a Tomlinson precoder
but this requires a feedback channel and precomputing the filter

So, QAM does not lend itself to water-pouring

The trick is to use Frequency Domain Multiplexing

We saw FDM as a technique to mux different information sources

Here we divide a single information stream into blocks of bits
and mux them together using distinct sub-carriers

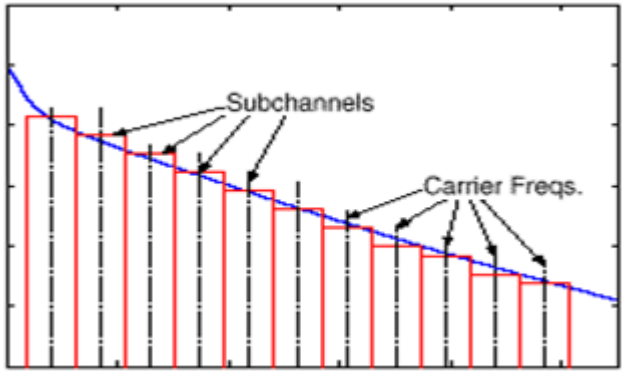
Each sub-carrier signal can

- have its own power level
- use its own modulation technique (PSK, QPSK, 16QAM, 64QAM, ...)

thus directly implementing water pouring

FDM combats ISI but creates ICI

FDM uses many subchannels, each with low bandwidth but low data rate



Since the data rate is low, there is essentially no ISI

And since each subchannel is localized in frequency
we can perform equalization in the frequency domain (FEQ)
i.e., simply multiply each frequency and shift its phase

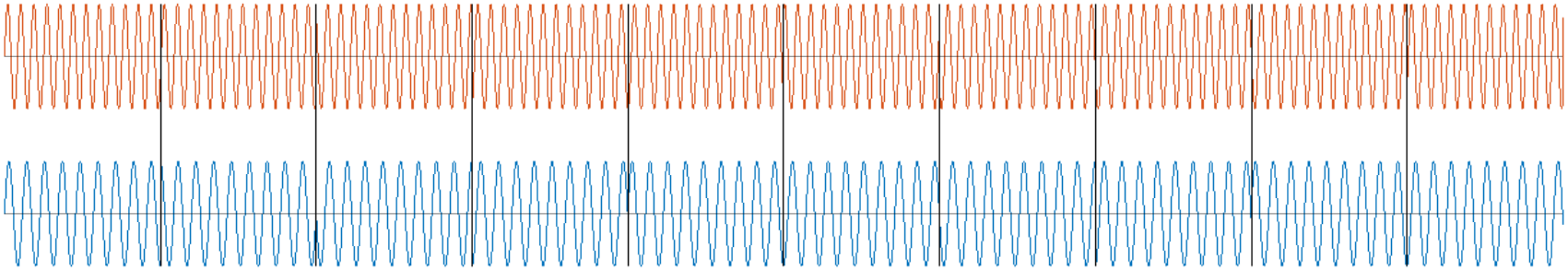
But, we need to space the sub-carriers far enough apart
to avoid **I**nter**C**hannel **I**nterference

This squanders bandwidth, distancing us from the Shannon capacity

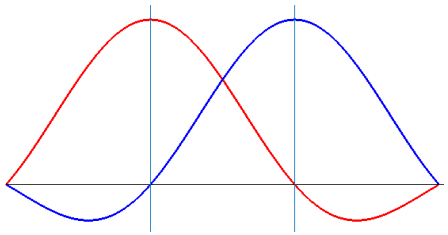
OFDM

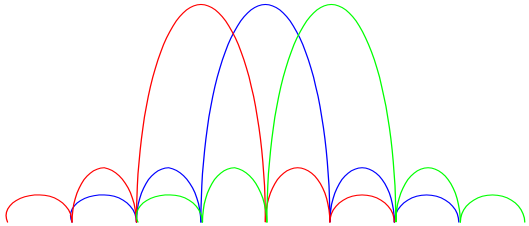
The solution is called Orthogonal FDM (OFDM)

- all sub-channels use the same symbol rate (even if different modulations)
- sub-carriers are spaced at precisely the symbol rate
- the sub-carriers are the precisely orthogonal and hence do not interfere with each other



- ICI is eliminated with no guard frequencies needed
- simple implementation based on the FFT



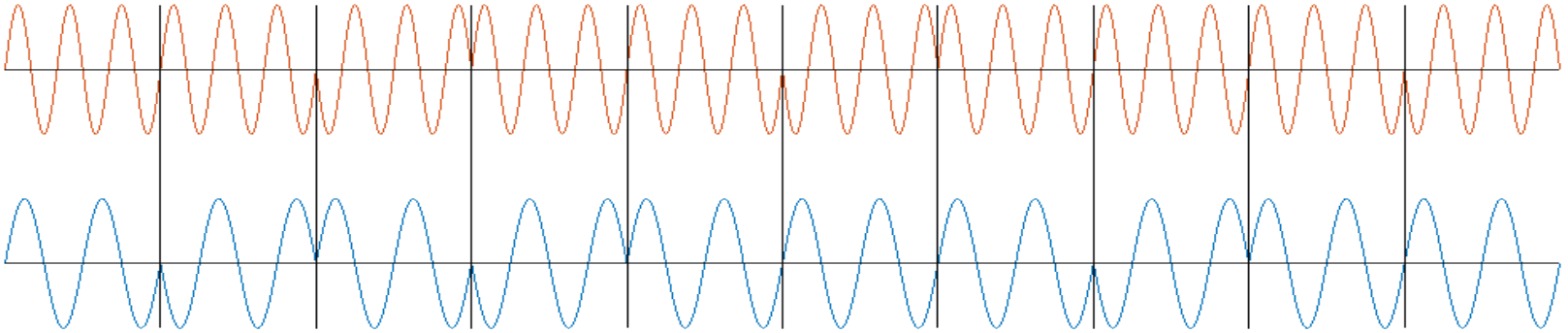


Why are the channels orthogonal?

Let's look at the baseband signal

where all sub-carriers are multiples of the symbol rate

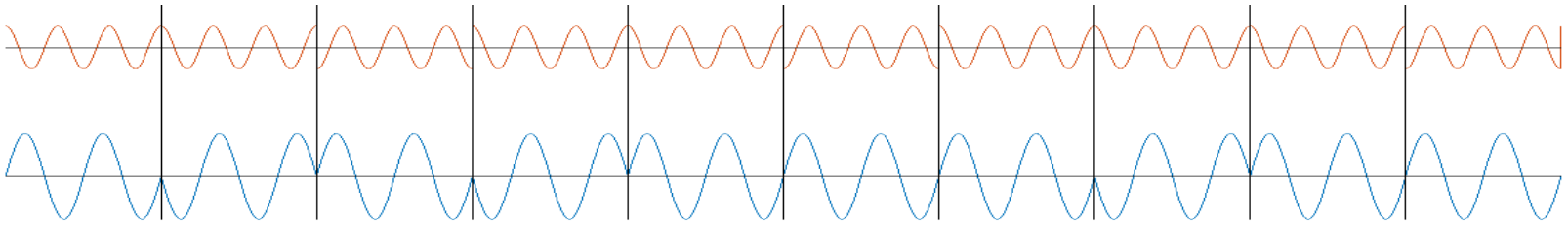
so that there are an integral number of sinusoid cycles in a symbol



Any two such signals are precisely orthogonal

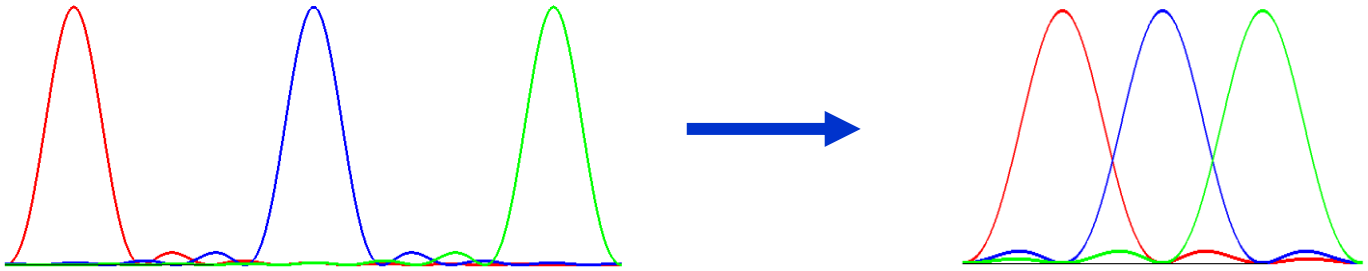
$$\int_0^{\frac{2\pi}{\omega}N} e^{in_1\omega t}\, e^{-in_2\omega t}\, dt = 0$$ (the only requirement is for a whole number of cycles of sin **Δω** t !)

and the same is true if we arbitrarily phase shift or amplify the signals



Spectral efficiency

OFDM provides the minimum possible sub-carrier spacing
and hence eliminates the need for *guard bands*

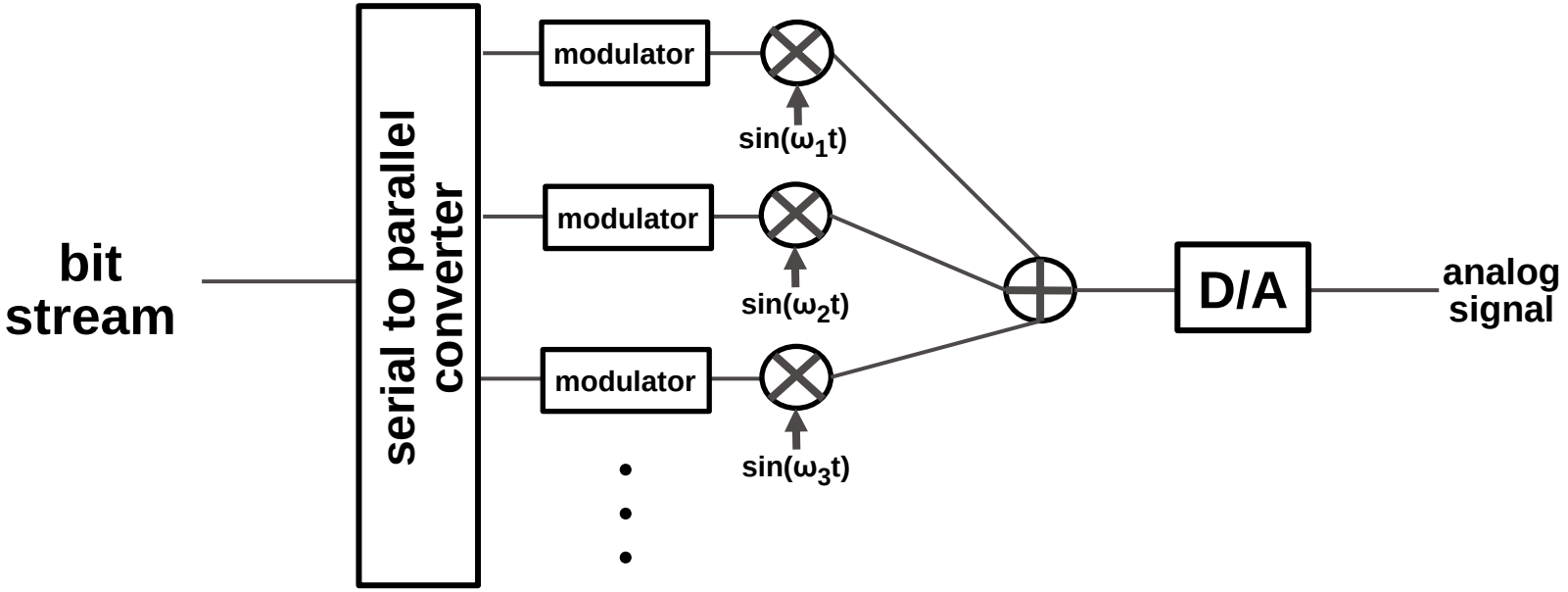


Now each sub-channel effectively occupies only symbol-rate of bandwidth
So the spectral efficiencies improve by a factor of 2 !

modulation	bit/symbol	BW/symbol rate	spectral efficiency
BPSK	1	1	1
QPSK	2	1	2
8PSK	3	1	3
16QAM	4	1	4
64QAM	6	1	6
256QAM	8	1	8

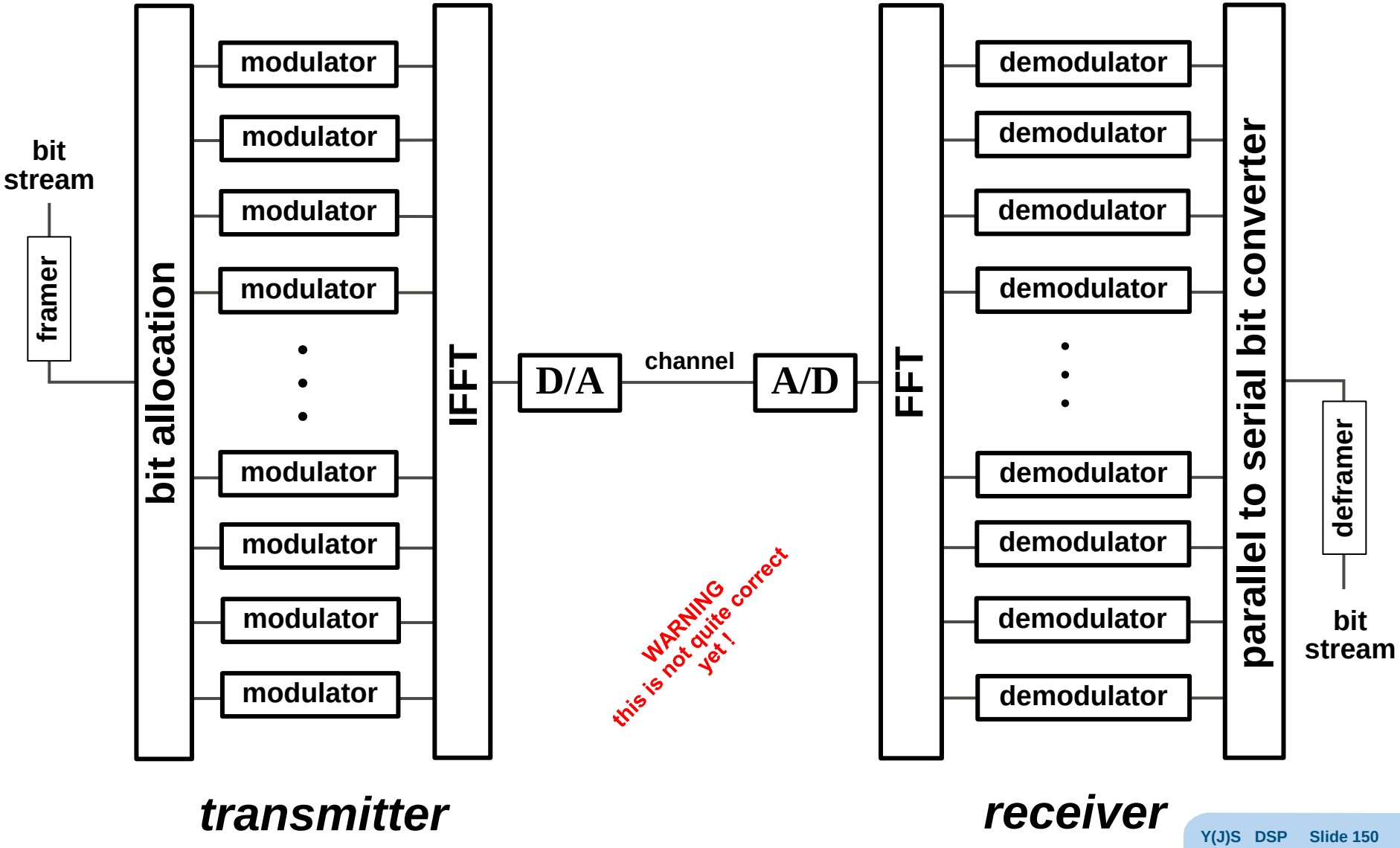
FFT

In order to transmit each sub-channel
we should modulate a baseband signal
up-mix each to the desired sub-carrier frequency (multiply by $e^{i\omega_c t}$)
and add the sub-channels together



The upmixing can be performed in parallel for all sub-carriers
by the inverse Fourier transform (Zimmerman and Kirsch 1967)
and the FFT does it quickly (Weinstein and Ebert 1969)

OFDM modem paradigm



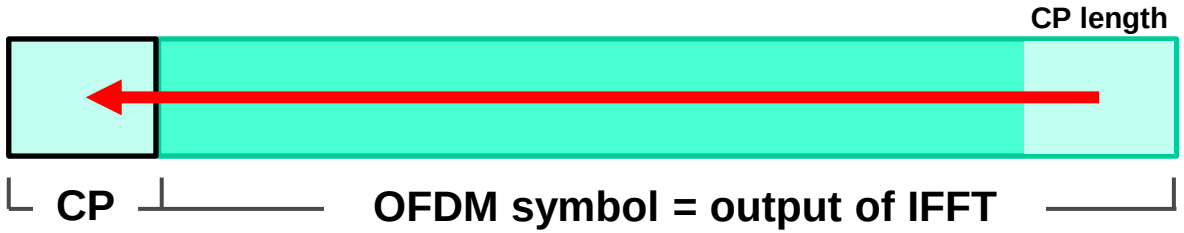
Cyclic prefix

What we previously derived *almost* works

For **analog digital** signal processing $Y(\omega) = H(\omega)\, X(\omega)$
linear cyclic convolution in the time domain **y = h * x**
is equivalent to multiplication in the frequency domain $Y_k = H_k\, X_k$

So, the linear convolution in the analog channel
has to be converted into cyclic convolution for the digital channel

This is done by adding a **C**yclic **P**refix to the signal



which basically acts as a *guard interval* to eliminate ISI

The CO duration tends to be from 1/32 up to 1/4 of the symbol duration
and has to be long enough to include the maximum ISI spread

For example, if ISI is due to multipath
then the CP must be long enough to incorporate the longest delayed path

CP at work

OFDM splits the signal in the time domain into *frames*

In order for the signal processing of each frame to be independent
the CP ensures that delay spread is within the received frame

As a simple example, assume a frame of 8 samples
a 2-sample multipath $y_n = x_n + h_1 x_{n-1} + h_2 x_{n-2}$ and CP of 2 samples

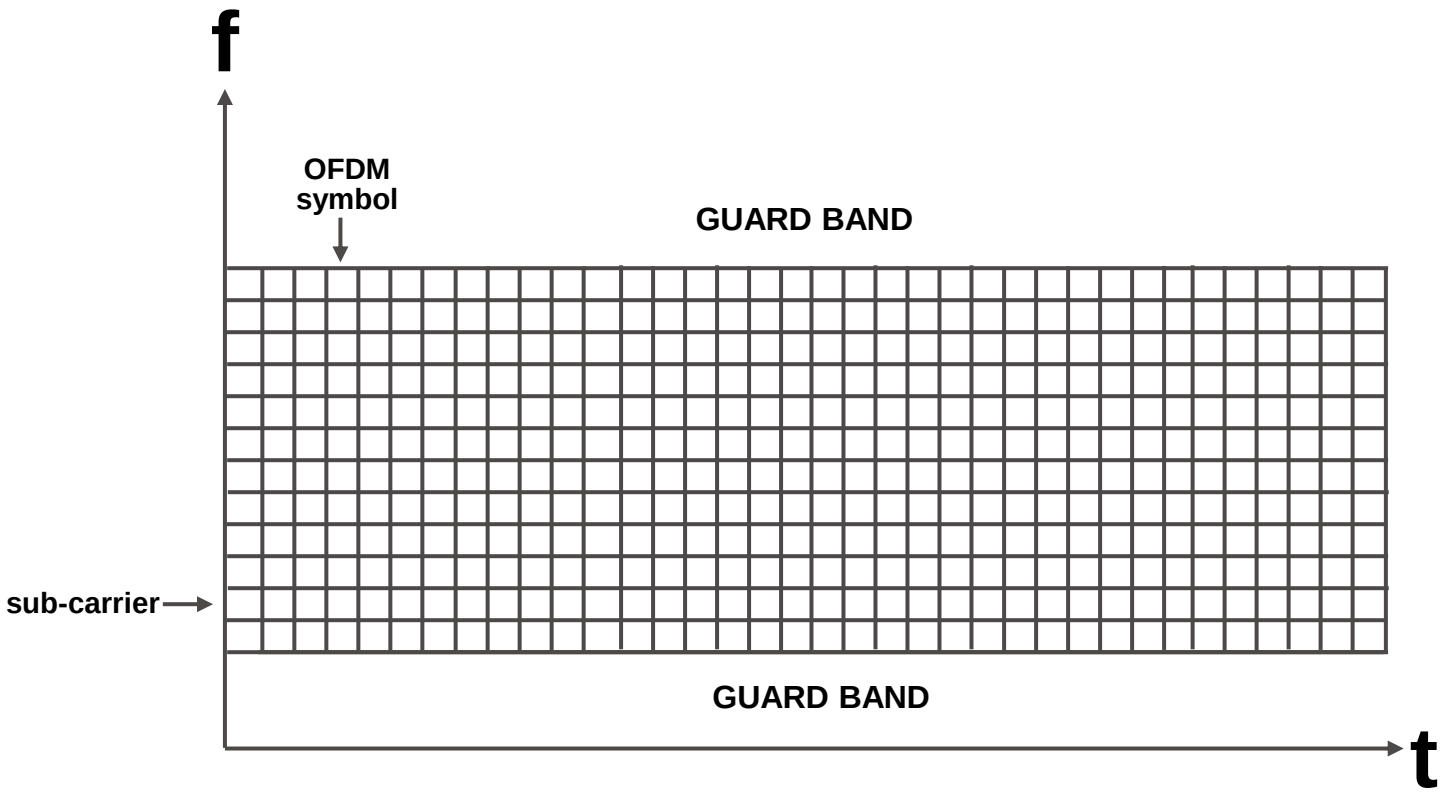
The OFDM frame before CP insertion is $x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7$
After CP insertion the xmted OFDM frame is $x_6\ x_7\ x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7$

The received OFDM signal is $y_6\ y_7\ y_0\ y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7$
where y_6 and y_7 contain ISI from the previous frame and are *discarded*
and

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & h_2 & h_1 \\ h_1 & 1 & 0 & 0 & 0 & 0 & 0 & h_2 \\ h_2 & h_1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_2 & h_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_2 & h_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_2 & h_1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_2 & h_1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_2 & h_1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}$$

All the information is present for equalization to recover the original $x_0 \cdots x_7$
and the matrix is *circulant* – which is solved in the frequency domain!

4G OFDM signal structure



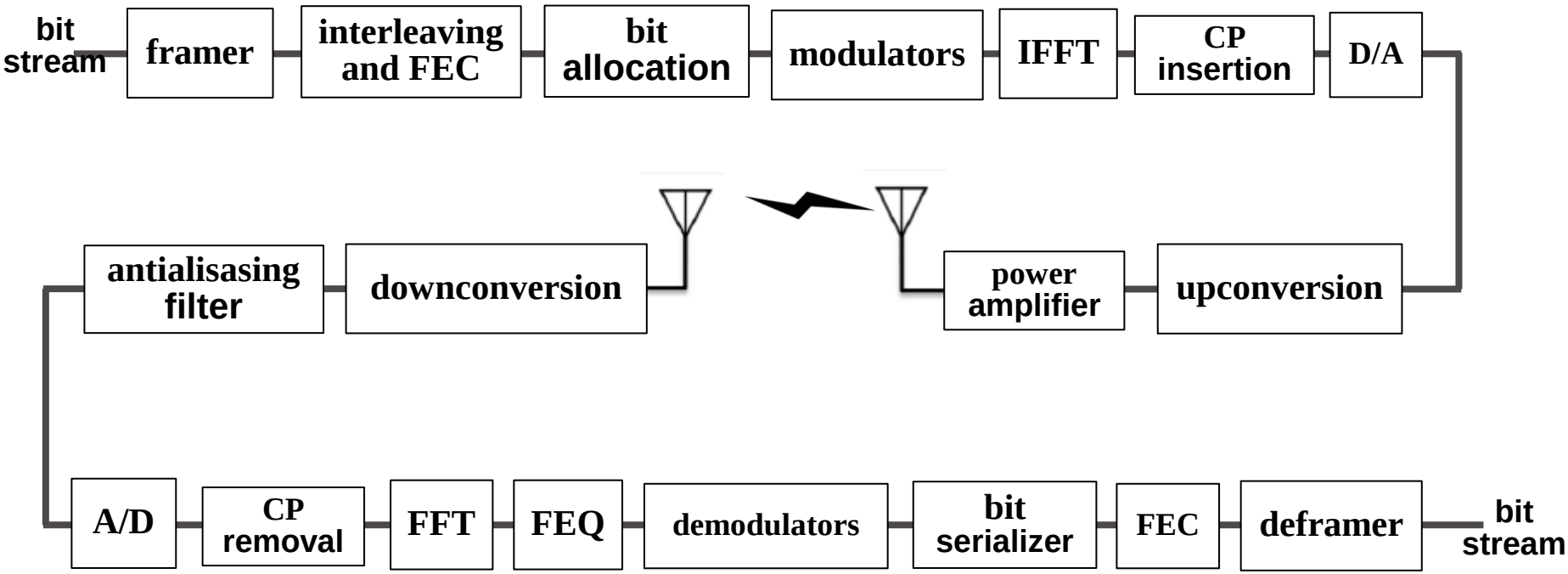
For 4G:

- **channel bandwidth ..., 5, 10, 15, or 20 MHz**
- **guard band overhead is 10%**
- **sub-carrier spacing = 15 kHz**
- **OFDM symbol duration = 1/15KHz = 66.67 µsec**
  - **short CP = 4.7 µsec so total duration = 71.367 µsec**
    - **1 *slot* = 7 symbols ≈ ½ msec***
  - **long CP = 16.7 µsec so total duration = 83.367 µsec**
    - **1 *slot* = 6 symbols ≈ ½ msec***

BW (MHz)	usable BW (MHz)	subchannels	FFT
5	4.5	300	512
10	9	600	1024
15	13.5	900	1536
20	18	1200	2048

*** CP durations are *adjusted* so that the slot is precisely ½ msec**

OFDM modem



Warning: this is somewhat over-simplified!

OFDMA

4G and 5G cellular are based on OFDM
but need **M**ultiple **A**ccess and Duplexing mechanisms too

For **MA** an orthogonal version of FDMA is used

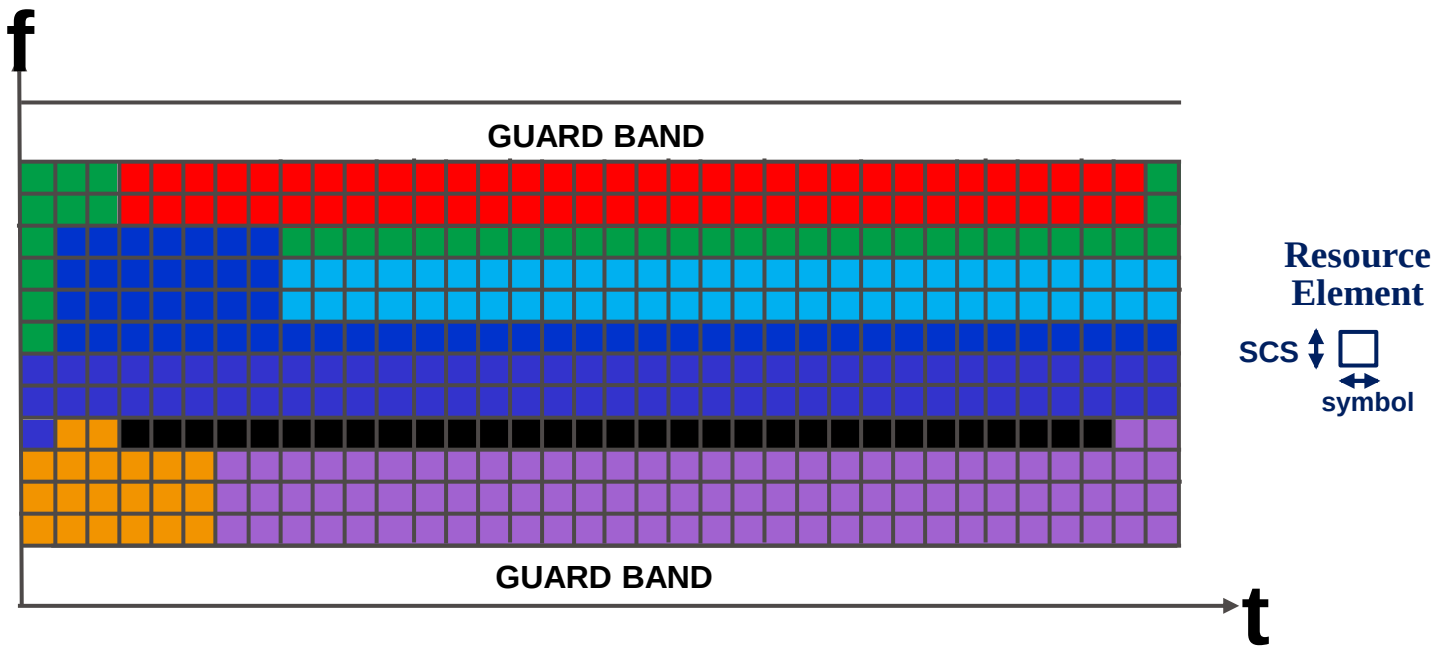
- signals from/to different users occupy different frequencies
- the **S**ub**C**arrier **S**pacing must be exactly the OFDM symbol rate

For **duplexing** there are two alternatives:

- Frequency Domain Duplexing
  different frequency bands are used for the UL and DL
  for example n1: UL 1920-1980 MHz, DL 2110-2170 MHz
- Time Domain Duplexing
  a single frequency band is used
  for example n38: 2570-2620 MHz for both UL and DL

OFDMA

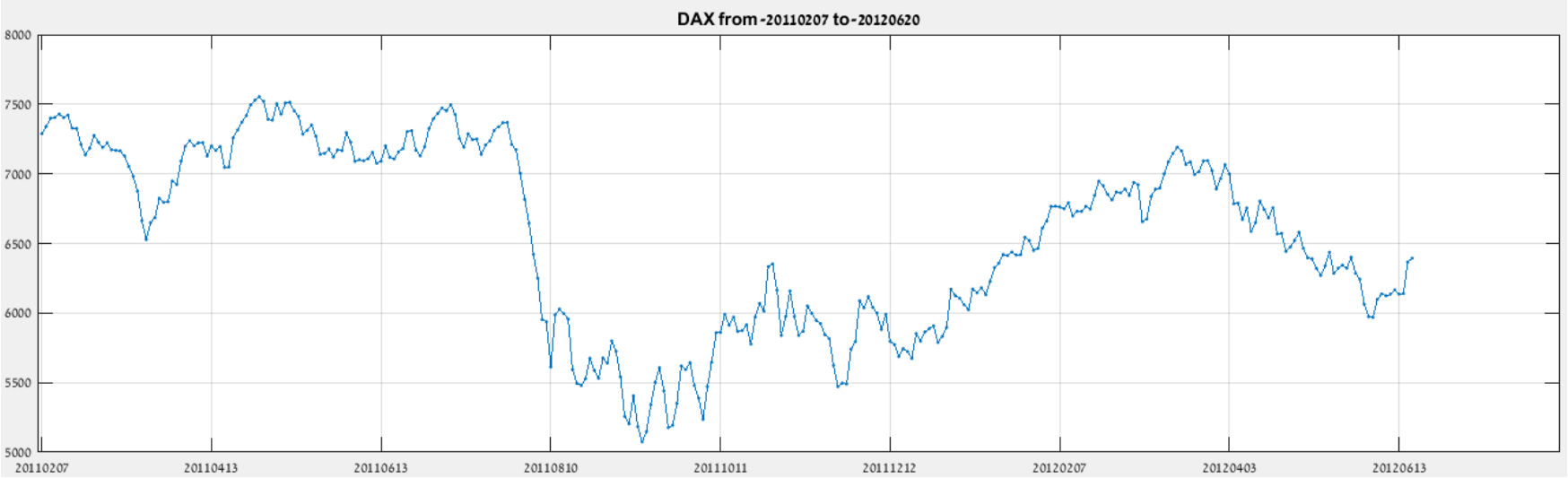
The basic OFDM paradigm can be readily extended to OFDMA
by allocating time-frequency **R**esource **E**lements to different users



In the DL direction the base-station transmits to *all* users
and each needs to know which REs it needs to extract
In the UL direction each UE transmits only in its REs
in order not to interfere with other UEs in the cell

Financial signal processing

Application : Stock Market



This signal is hard to predict (extrapolate)

- *self-similar* and *fractal* dimension
- polynomial smoothing leads to overfitting
- noncausal MA smoothing (e.g., Savitsky Golay) doesn't extrapolate
- causal MA smoothing leads to significant delay
- AR modeling works well
  - but sometimes need to bet the trend will continue
  - and sometimes need to bet against the trend