Fundamentals of information transmission

Bits, information, and signals

Elementary operation of communication: send bits or information as *signal* on physical medium from A to B.

- physical media—copper wire, optical fiber, air/space
- signals—voltage and currents, light pulses, radio waves, microwaves

  → electromagnetic wave

  → aka "light"

  → Einstein was intrigued by it

  → engineering aspect: well-understood

  → other aspects of light remain a mystery (physics) even today (2007)

Other types of signal:

- smoke signals (cowboy movies)
- sound (acoustic waves)
- other?

All signals of practical interest have one thing in common:

$\longrightarrow$ signal *strength*

Ex.:

- light: brightness (or intensity)
- sound: loudness (or volume)

$\longrightarrow$ signal strength can be measured

$\longrightarrow$ e.g., dB for sound

Another common feature of signals:

⟶ signal strength can *vary over time*

⟶ now: quiet, 1 sec later: loud, 2 secs later: loud

⟶ use it to send 3 bits: how?

⟶ what's the throughput (bps)?

⟶ is it a good solution?

What else can one do to increase throughput?

Is there a way to do much better?

⟶ yes

⟶ at the heart of today's high-speed networks

⟶ wireless networks: for free!

⟶ the main goal of the following discussion

Recap: signal has

- strength
- and signal strength can vary over time

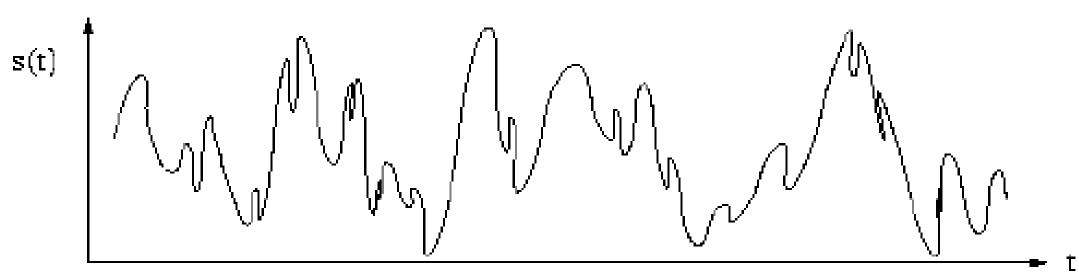
$\longrightarrow$ computer networks: light (electromagnetic waves)

Denote signal as $s(t)$ where:

- t: time (discrete or continuous)
- s(t): indicates signal strength at time t

$\rightarrow$ also called magnitude or amplitude

Cartoon picture of some signal $s(t)$:

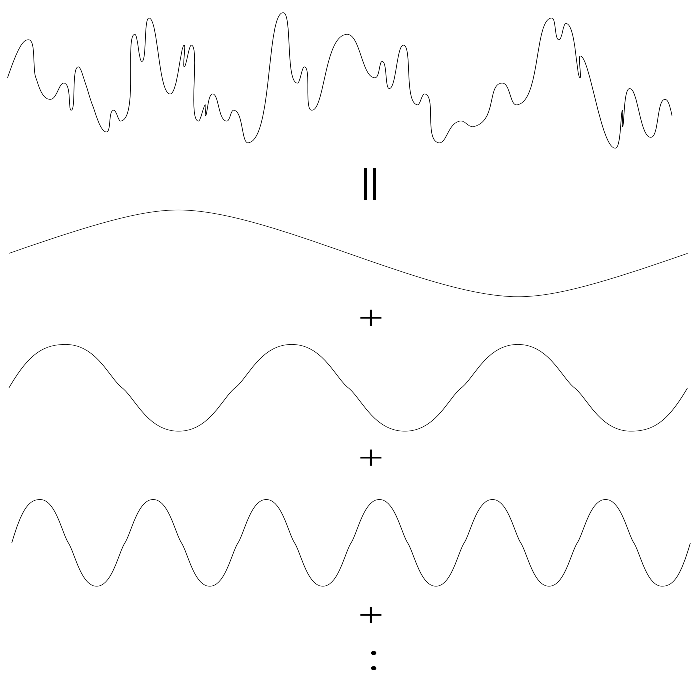


The most important feature of signals: *complicated looking signals are just sums of very simple signals*

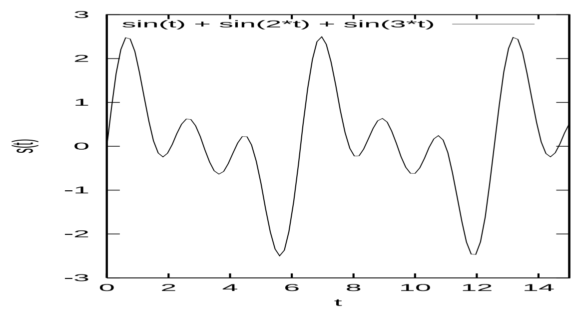
$\longrightarrow$ simple signals: waves

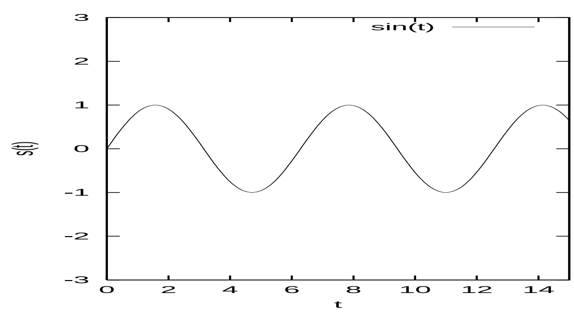
$\longrightarrow$ sine curve (why "simple"?)

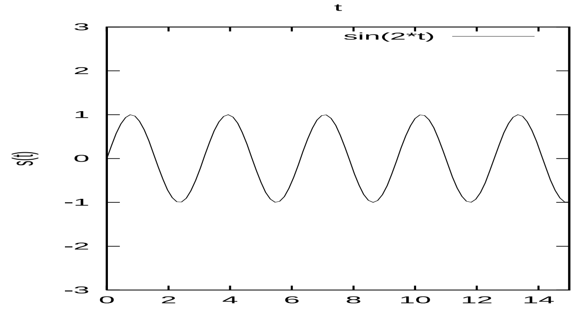
Cartoon picture of this "decomposition" principle:

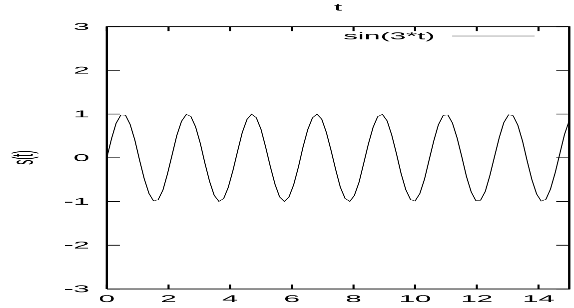


Real example:

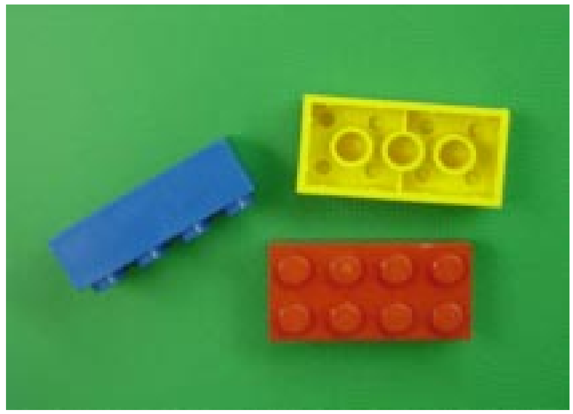


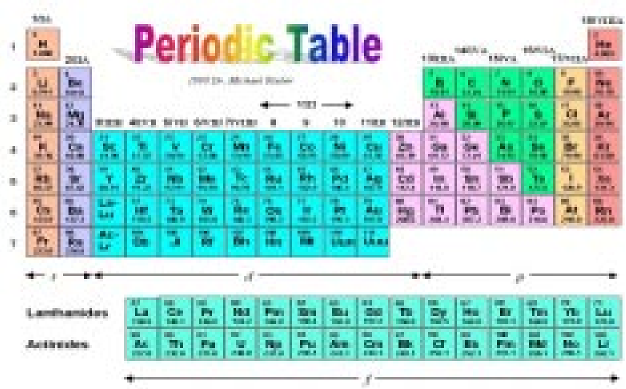






Other examples (man-made & nature):





⟶ pretty much everything obeys this principle

⟶ what about 2-D image?

Question: why is this the case?

A connection to linear algebra . . .

Actually, linear algebra comes to the rescue.

$\longrightarrow$ yes, there was a reason for studying linear algebra

$\longrightarrow$ university is a good place after all

$\longrightarrow$ where is my linear algebra textbook?

Simple signals (sine waves): building blocks of more complicated signals

Analogous to "basis" in linear algebra

other elements (vectors) can be expressed as sums of simple elements (basis vectors)

Ex.: in 3-D

$\longrightarrow$ $\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$ form a basis

$\longrightarrow$ $(7,2,4) = 7 \cdot (1,0,0) + 2 \cdot (0,1,0) + 4 \cdot (0,0,1)$

$\longrightarrow$ coefficients: 7, 2, 4

$\longrightarrow$ more precisely: *bases may have to be multiplied*

$\longrightarrow$ called linear combination

Coefficients are *very* important:

$\longrightarrow$ even have special name: spectrum

Note: bases need not be $\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$

$\longrightarrow$ $\{(2,0,0),\ (0,4,0),\ (0,0,5)\}$ is fine too

$\longrightarrow$ what's the spectrum of $(7,2,4)$?

$\longrightarrow$ is $\{(11,6,3),\ (2,500,7),\ (31,44,1)\}$ valid basis?

$\longrightarrow$ in general, to qualify as a basis ...

$\longrightarrow$ how many elements in a basis set?

Is there anything special about the basis set

$$\{(1,0,0),\ (0,1,0),\ (0,0,1)\}?$$

Yes, $\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$ is orthogonal:

$\longrightarrow$ $(1,0,0) \circ (0,1,0) = 0$

$\longrightarrow$ $(1,0,0) \circ (0,0,1) = 0$

$\longrightarrow$ $(0,1,0) \circ (0,0,1) = 0$

where "$\circ$" is the dot product

$$(x_1, x_2, x_3) \circ (y_1, y_2, y_3) = x_1 y_1 + x_2 y_2 + x_3 y_3$$

Furthermore,

$\longrightarrow$ $(1,0,0) \circ (1,0,0) = 1$

$\longrightarrow$ $(0,1,0) \circ (0,1,0) = 1$

$\longrightarrow$ $(0,0,1) \circ (0,0,1) = 1$

OK, so what's the big deal?

$\longrightarrow$ why is orthogonality relevant

Allows us to calculate coefficients of basis

$\longrightarrow$ algorithm for finding spectrum

Given basis set $\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$

$\longrightarrow$ $(7,2,4) = 7 \cdot (1,0,0) + 2 \cdot (0,1,0) + 4 \cdot (0,0,1)$

$\longrightarrow$ spectrum: 7, 2, 4

$\longrightarrow$ "reading off": cheating!

$\longrightarrow$ what is the general principle?

To compute spectrum of $(1,0,0)$ for $(7,2,4)$:

$\longrightarrow$ take dot product: $(7,2,4) \circ (1,0,0) = 7$

$\longrightarrow$ why does it work?

Since $(7,2,4) = 7 \cdot (1,0,0) + 2 \cdot (0,1,0) + 4 \cdot (0,0,1)$,

we have

$$
\begin{aligned}
(7,2,4) \circ (1,0,0) \\
&= [7 \cdot (1,0,0) + 2 \cdot (0,1,0) + 4 \cdot (0,0,1)] \circ (1,0,0) \\
&= 7 \cdot (1,0,0) \circ (1,0,0) \\
&\qquad + 2 \cdot (0,1,0) \circ (1,0,0) \\
&\qquad\qquad + 4 \cdot (0,0,1) \circ (1,0,0) \\
&= 7 \cdot 1 + 2 \cdot 0 + 4 \cdot 0 \\
&= 7
\end{aligned}
$$

Note: works for *any* orthonormal basis

$$\{(x_1, x_2, x_3), (y_1, y_2, y_3), (z_1, z_2, z_3)\}$$

Vector spaces:

- finite dimensional

  $\longrightarrow$ e.g., 7-dimensional: $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$

  $\longrightarrow$ subject of linear algebra

- infinite dimensional: signal $s(t)$

  $\longrightarrow$ e.g., infinite number of components $(x_1, x_2, \ldots)$

  $\longrightarrow$ or continuously varies with t

  $\longrightarrow$ infinite number of bases

  $\longrightarrow$ bad news: cannot use linear algebra in either case

  $\longrightarrow$ good news: same basic principles

Why is knowing the coefficients (spectrum) important?

Two reasons:

- allows us to transmit bits faster

  $\longrightarrow$ the foundation of today's *high-speed* networks

- makes life a little easier

First reason: Allows us to transmit bits faster

How?

Two steps:

- Step 1: Encode bit in the coefficient

  → coefficient 1: bit 1

  → coefficient 0: bit 0

  → spectrum is important because it hides the bit

  → not much to it (step 2 is the interesting one)

- Step 2: To increase bps k-fold

  $\longrightarrow$ say from 1 bps to 100 bps if $k = 100$

  $\longrightarrow$ use k-dimensional orthonormal basis vectors

  $\longrightarrow$ call them $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$

  $\longrightarrow$ call k data bits: $a_1, a_2, \ldots, a_k$

  $\longrightarrow$ to be sent *simultaneously* (hence k-fold faster!)

  $\longrightarrow$ sender prepares $\mathbf{z} = a_1\mathbf{x}^1 + a_2\mathbf{x}^2 + \cdots + a_k\mathbf{x}^k$

  $\longrightarrow$ $\mathbf{z}$ is another k-dimensional vector ("scrambled")

  $\longrightarrow$ sender transmits $\mathbf{z}$ in one step

  $\longrightarrow$ receiver gets $\mathbf{z}$

  $\longrightarrow$ to recover first bit a_1, calculates $\mathbf{z} \circ \mathbf{x}^1$

  $\longrightarrow$ we established: $\mathbf{z} \circ \mathbf{x}^1 = a_1$

  $\longrightarrow$ in parallel: do the same for $\mathbf{z} \circ \mathbf{x}^i = a_i$

Powerful technique.

$\longrightarrow$ courtesy of linear algebra

What if we wanted to add security?

$\longrightarrow$ e.g., protect against eavesdropping?

The above linear algebra method (of course, simplified) is used by some cellular providers (e.g., Sprint) to carry k customer calls simultaneously

$\longrightarrow$ called CDMA

$\longrightarrow$ note: all voice calls are digital (transmit bits)

Second reason: makes life a little easier

$\longrightarrow$ broader implications than computer networks

$\longrightarrow$ laid back attitude

$\longrightarrow$ don't sweat the little things

$\longrightarrow$ in science & engineering jargon: let's approximate!

Focus on what's *important*.

Take $(7, 2, 4)$.

$\longrightarrow$ which building block is most important?

$\longrightarrow$ $(1, 0, 0)$ since it's multiplied by 7

$\longrightarrow$ least important: $(0, 1, 0)$

From an approximation angle

$\longrightarrow$ $(7, 2, 4)$ kind of looks like $(7, 0, 0)$

$\longrightarrow$ $(7, 0, 4)$ is pretty close

$\longrightarrow$ $(7, 2, 4)$ is 100% accurate

An aside:

In science & engineering: we almost never deal with exact things. (The same is true in mathematics.)

$\longrightarrow$ many times hard

$\longrightarrow$ most of the time: unnecessary

$\longrightarrow$ i.e., approximate answer is good enough

Thus science & engineering is about *managed* inaccuracy.

Some examples.

Ex.: computer science

- compression: JPEG, MPEG are all lossy

  $\longrightarrow$ disk space forces us to approximate

  $\longrightarrow$ luckily human eye or ear does the same

- caching: memory hierarchy

  $\longrightarrow$ cache $\mapsto$ RAM $\mapsto$ disk

  $\longrightarrow$ cache contains approximation of memory

  $\longrightarrow$ memory contains approximation of disk

  $\longrightarrow$ luckily it works

  $\longrightarrow$ because programs obey locality-of-reference

- many more

Back to continuous signals $s(t)$.

In high-speed networks, we do not use finite dimensional vectors but continuous signals.

$\longrightarrow$ instead of vectors, sine curves

$\longrightarrow$ basis set is now comprised of sine curves

$\longrightarrow$ an infinite number of them

$\longrightarrow$ linear algebra concepts carry over

Specifically: $s(t)$ is viewed as the integral (i.e., sum)

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{i\omega t} d\omega$$

$\longrightarrow$ signal $s(t)$ is a linear combination of the $e^{i\omega t}$

$\longrightarrow$ recall: $e^{i\omega t} = \cos \omega t + i \sin \omega t$

$\longrightarrow$ building block: sine curve

$\longrightarrow$ basically: weighted sum of sine curves

$\longrightarrow$ $S(\omega)$: coefficient of basis elements

$\longrightarrow$ like a_i in $\mathbf{z} = a_1 \mathbf{x}^1 + a_2 \mathbf{x}^2 + \cdots + a_k \mathbf{x}^k$

$\longrightarrow$ note similarity: $\mathbf{z}(t) = \sum_{i=1}^{k} a_i \mathbf{x}^i(t)$

$\longrightarrow$ called Fourier expansion

ω: cycles per second (Hz)

$\longrightarrow$ $\omega = 1/T$ where T is the period

$\longrightarrow$ called frequency

For the same reasons as before, coefficient $S(\omega)$ (i.e., spectrum) is important:

- allows us to transmit bits faster

  $\rightarrow$ high-speed simultaneous transmission

- makes life a little easier

  $\rightarrow$ approximation

Need to know how to compute $S(\omega)$

$\longrightarrow$ similar to dot product $\mathbf{z} \circ \mathbf{x}^i$ to get a_i

Formula to compute $S(\omega)$:

$$S(\omega) = \int_{-\infty}^{\infty} s(t) e^{-i\omega t} dt.$$

$\longrightarrow$ called Fourier transform

$\longrightarrow$ does it look like a "dot product"?

Note: $a_i = \mathbf{z} \circ \mathbf{x}^i$

$\longrightarrow$ keep in mind: dot product is sum of products

To send k bits simultaneously:

- pick k different frequencies $\omega_1, \omega_2, \ldots, \omega_k$

  $\longrightarrow$ in place of vectors $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$

  $\longrightarrow$ ω_i called carrier frequency

- encode k bits as high/low (e.g., 1 or 0) of the $S(\omega_i)$'s

- sender prepares $s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{i\omega t} d\omega$

- sender transmits "scrambled" signal $s(t)$

- receiver gets $s(t)$

- receiver, in parallel, recovers i'th bit by computing Fourier transform $S(\omega_i) = \int_{-\infty}^{\infty} s(t) e^{-i\omega_i t} dt$

- $\longrightarrow$ recall: bits cannot travel faster than SOL
- $\longrightarrow$ high-speed networks: parallel lanes
- $\longrightarrow$ different carrier frequencies ω_i: role of lanes
- $\longrightarrow$ more frequencies, more parallel transmission
- $\longrightarrow$ also called broadband networks