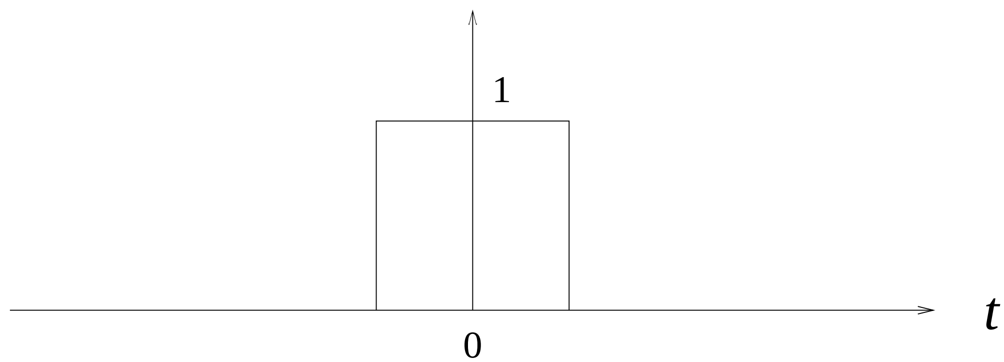
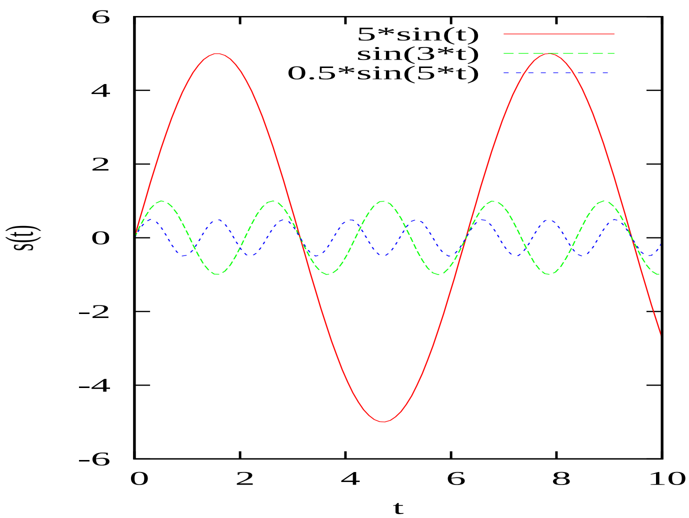
Example of spectra: square wave

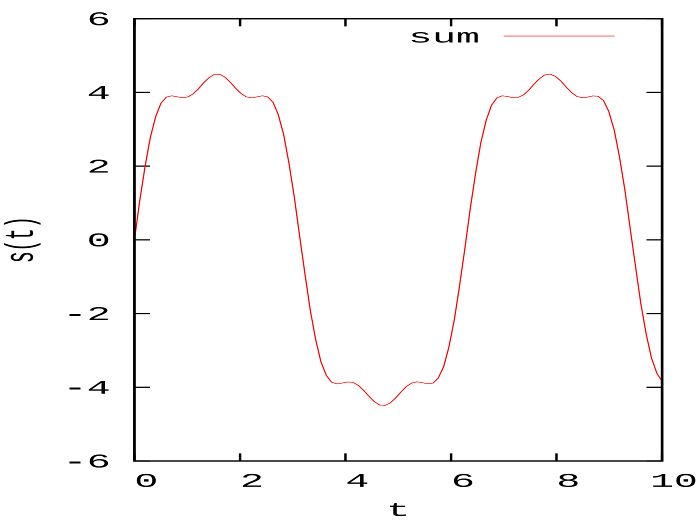


→ one of the signals considered difficult to synthesize using sinusoids

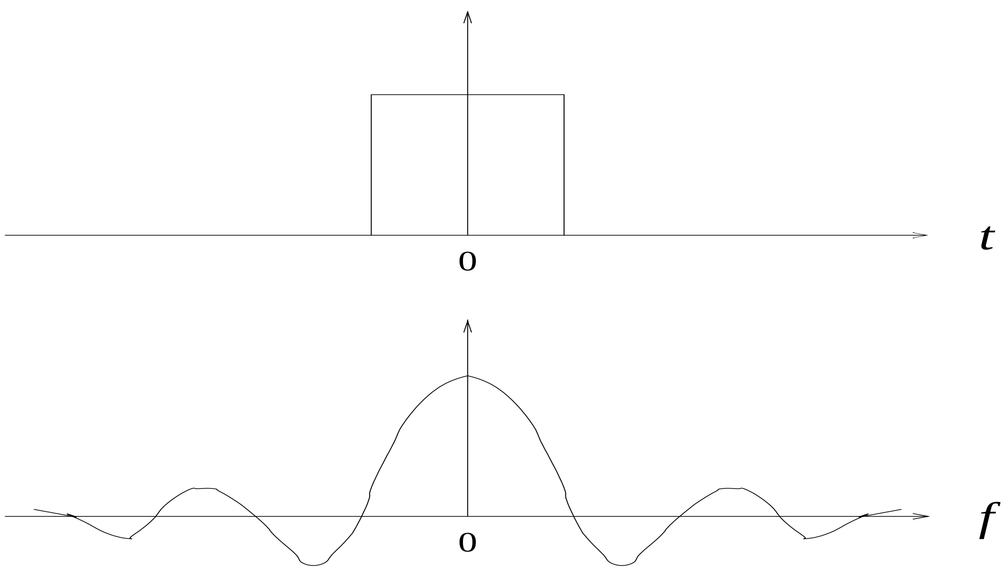
→ due to sharp transition or edge

Three sine curves and their sum:





Spectrum of square wave:

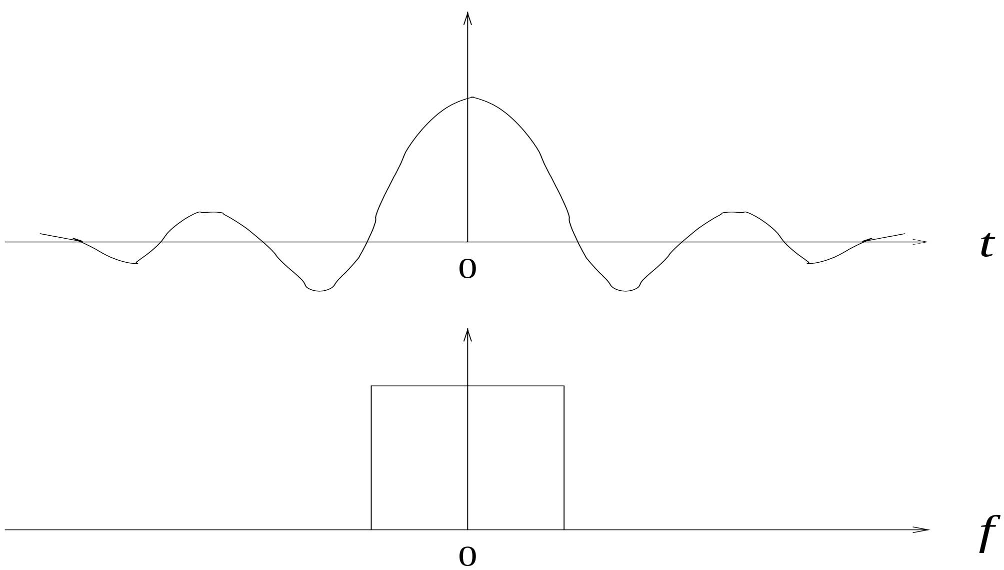


→ high frequency sine curves are less important but still needed

→ don't become zero (close to zero for large f)

→ infinite spectrum (i.e., not bandlimited)

Bandlimited signal:



→ take opposite from before

→ bandlimited

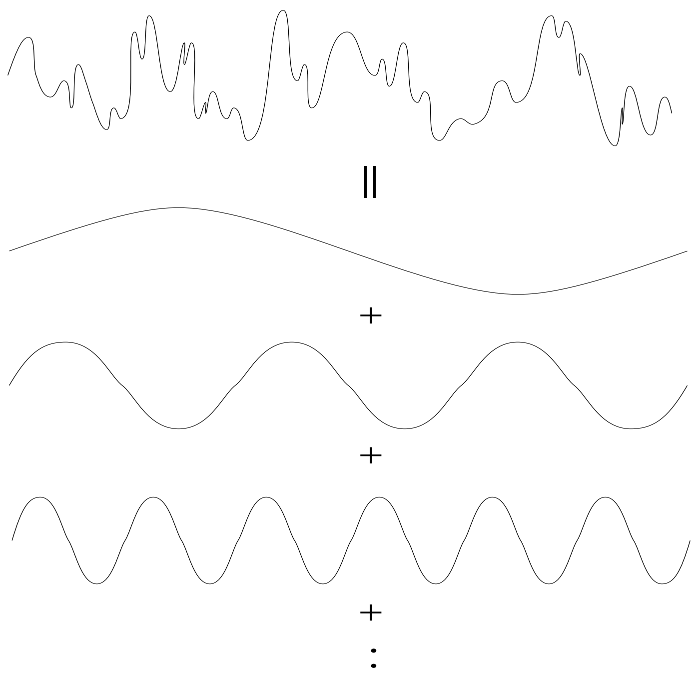
→ but signal is not timelimited

Spectrum of flat signal?

Spectrum of random signal?

Common notations to note:

signal $s(t)$ is just sum of weighted sine curves:



sine curve of frequency f: $\sin ft$

its weight (i.e., amplitude): α_f

$\rightarrow$ spectrum

Reminder:

- The value of the weight α_f of sinusoid f indicates how important sinusoid f is
- For example, if $\alpha_f = 0$ then sinusoid of frequency f is not needed at all for creating $s(t)$
- The weights α_f shown for all frequencies f as a table or graph is called the spectrum of signal $s(t)$

  $\rightarrow$ the "genes" of $s(t)$

Two important notations:

1. signal $s(t)$ is weighted sum of sinusoids:

$\rightarrow$ concept translated into math symbols

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \alpha_f e^{ift} df$$

$\rightarrow$ called Fourier expansion

$\rightarrow$ integral "$\int$" is just continuous sum

$\rightarrow$ recall: $e^{ift} = \cos ft + i \sin ft$

$\rightarrow$ Euler's formula

$\rightarrow$ why complex sinusoids involving $i = \sqrt{-1}$?

2. Given $s(t)$, how to find weight α_f of sinusoid f:

$$\alpha_f = \int_{-\infty}^{\infty} s(t) e^{-ift} dt$$

$\rightarrow$ called Fourier transform

$\rightarrow$ another weighted sum

$\rightarrow$ algorithm to compute Fourier transform quickly: fast Fourier transform (FFT)

$\rightarrow$ see algorithms book

$\rightarrow$ can implement in software, chip firmware, or chip hardware (DSP)

$\rightarrow$ what receiver NIC has to do to decode bits sender has sent

$\rightarrow$ what sender does to ship bits: manipulating amplitude (AM) is referred to as IFFT (inverse FFT)

Traditional way to combat ICI in FDM: use guard bands

→ insert sufficient gaps between carrier frequencies

→ overhead can be significant

→ reduces how many carrier frequencies can be squeezed into a given frequency band

→ low spectral efficiency

Modern approach to packing more carrier frequencies within a given frequency band

$\rightarrow$ orthogonal FDM

$\rightarrow$ neighboring spectra can overlap without causing ICI

$\rightarrow$ the state-of-the-art in wired/wireless communication systems

Conceptual similarity to linear algebra

$\rightarrow$ get to make use of linear algebra!

$\rightarrow$ also get CDMA for free!

Simple facts from linear algebra:

Take 3-D space:

Given two vectors $x = (x_1, x_2, x_3)$ and $y = (y_1, y_2, y_3)$, they are orthogonal—i.e., perpendicular to each other—if, and only if,

$$x \circ y = x_1 y_1 + x_2 y_2 + x_3 y_3 = 0$$

$\rightarrow$ called dot product (or inner product)

$\rightarrow$ 3-D: $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$ are orthogonal

$\rightarrow$ also basis of 3-D

$\rightarrow$ called orthonormal if dot product with itself is 1

Lots of other orthogonal basis vectors

For example: $(5, 2, 0)$, $(2, -5, 0)$, $(0, 0, 1)$ are mutually orthogonal

$\rightarrow$ but not orthonormal

$\rightarrow$ how to make them orthonormal?

Relevance to networking:

→ CDMA (code division multiple access)

In CDMA—for example, used by Sprint and Verizon for wireless cellular in the U.S.—$(5, 2, 0)$, $(2, -5, 0)$, $(0, 0, 1)$ are called codes

→ one code per user

→ 3-D codes: 3 users (say Bob, Mira, Steve)

Suppose each user wants to send a single bit

$\rightarrow$ Bob: 1, Mira: 0, Steve: 0

Bob's cell phone: send $1 \times (5, 2, 0)$ to base station (cell tower)

Mira's cell phone: send $-1 \times (2, -5, 0)$ to base station

Steve's cell phone: send $-1 \times (0, 0, 1)$

$\rightarrow$ common convention: 1 for bit 1, -1 for bit 0

Base station receives: $(3, 7, -1)$

$\rightarrow \big(1 \times (5, 2, 0)\big) + \big(-1 \times (2, -5, 0)\big) + \big(-1 \times (0, 0, 1)\big)$

How can base station find what bit Bob has sent?

Base station: compute the dot product of what it has received, $(3, 7, -1)$, and the code of Bob, $(5, 2, 0)$

$\rightarrow (5, 2, 0) \circ (3, 7, -1) = 15 + 14 + 0 = 29$

$\rightarrow$ positive: hence bit 1

$\rightarrow$ what's special about 29?

To find out what Mira has sent:

$\rightarrow (2, -5, 0) \circ (3, 7, -1) = 6 - 35 + 0 = -29$

$\rightarrow$ negative: hence bit 0

To find out what Steve has sent:

$\rightarrow (0, 0, 1) \circ (3, 7, -1) = 0 + 0 + 1 = -1$

$\rightarrow$ negative: hence bit 0

$\rightarrow$ why does this work?

Base station decoding Bob's bit: $(5,2,0) \circ (3,7,-1)$

Since $(3,7,-1) = \big(1 \times (5,2,0)\big) + \big(-1 \times (2,-5,0)\big) + \big(-1 \times (0,0,1)\big)$

$(5,2,0) \circ (3,7,-1)$ equals

$$\begin{aligned}\big(1 \times (5,2,0) \circ (5,2,0)\big) \ &+ \ \big(-1 \times (5,2,0) \circ (2,-5,0)\big) \\ &+ \ \big(-1 \times (5,2,0) \circ (0,0,1)\big)\end{aligned}$$

which equals $\big(1 \times (5,2,0) \circ (5,2,0)\big)$

$\rightarrow$ the two interference terms are nullified

$\rightarrow$ orthogonality!

Same holds when computing Mira's bit and Steve's bit.