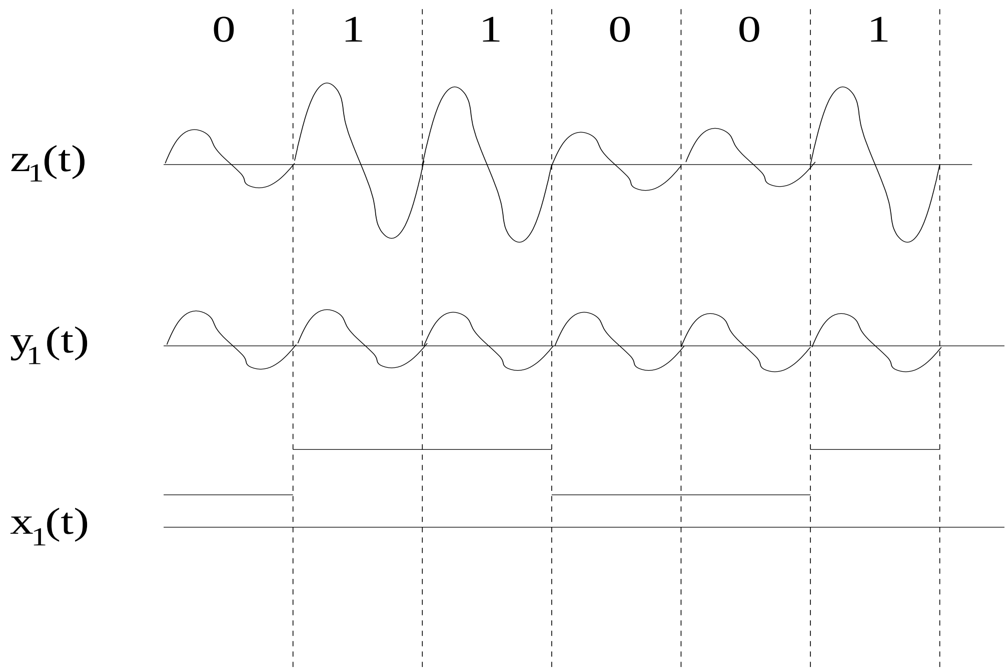
A global ("big picture") view of carrier separation in FDM with AM



→ hence $z_1(t) = x_1(t) \times y_1(t)$

→ what is the Fourier transform of sinusoid $y_1(t)$ of frequency f_1?

→ what is the Fourier transform of amplitudes $x_1(t)$?

Call the Fourier transform of $x_1(t)$, $F_{x_1}(f)$

$\rightarrow$ $F_{x_1}(f)$ gives the weight of sinusoid with frequency f

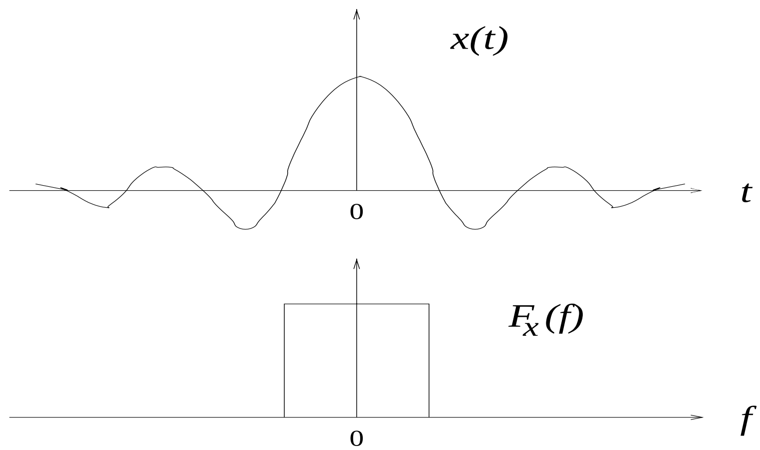
Useful fact:

The Fourier transform of $x_1(t) \times y_1(t)$ is just $F_{x_1}(f)$ shifted by f_1

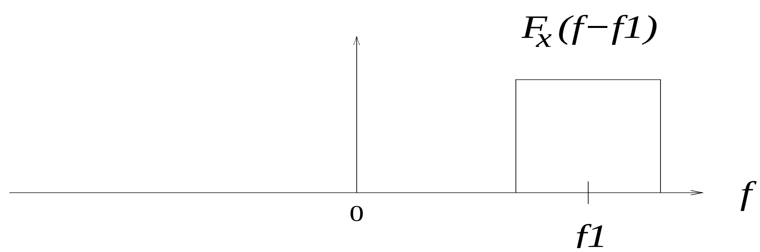
$\rightarrow$ $F_{x_1}(f - f_1)$

$\rightarrow$ modulation theorem

Example: Suppose $x_1(t)$ is the sinc function



Then Fourier transform of $x_1(t) \times y_1(t)$ is

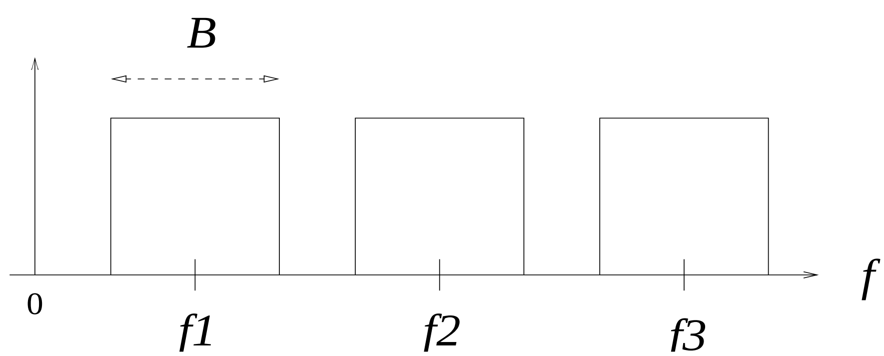


Back to FDM:

Suppose $y_2(t)$ is sinusoid of frequency f_2 and $x_2(t)$ is the amplitude signal

→ same for $y_3(t)$ and $x_3(t)$

If both $x_2(t)$ and $x_3(t)$ are sinc functions (same as $x_1(t)$) then the Fourier transform of all three is

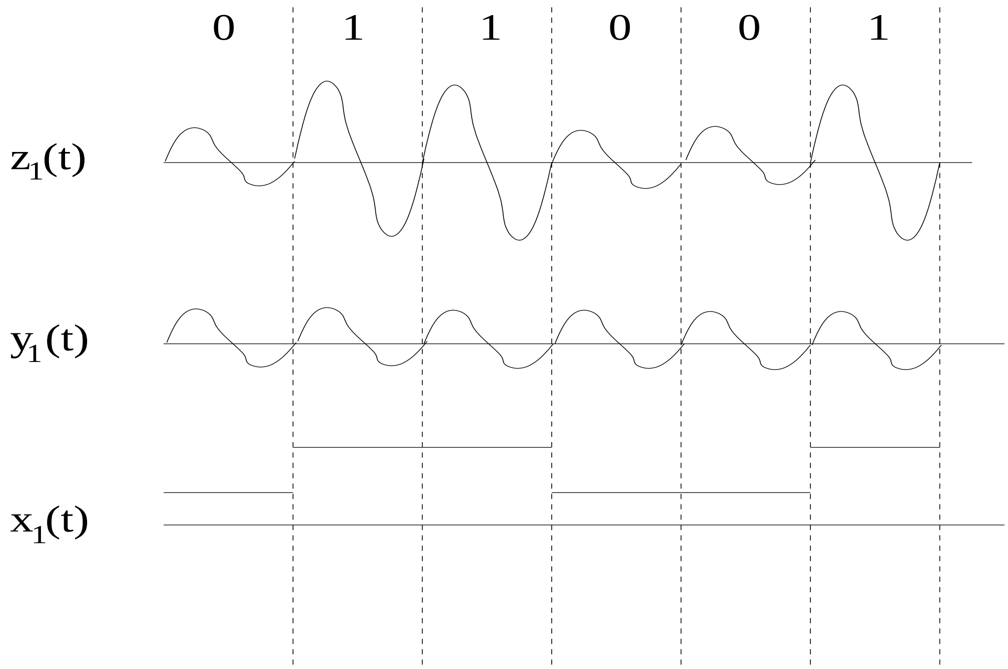


→ no inter-carrier interference for sufficiently large carrier separation

→ i.e., $f_2 - f_1 > B$ and $f_3 - f_2 > B$

where B is the bandwidth of $x_1(t)$, $x_2(t)$, $x_3(t)$

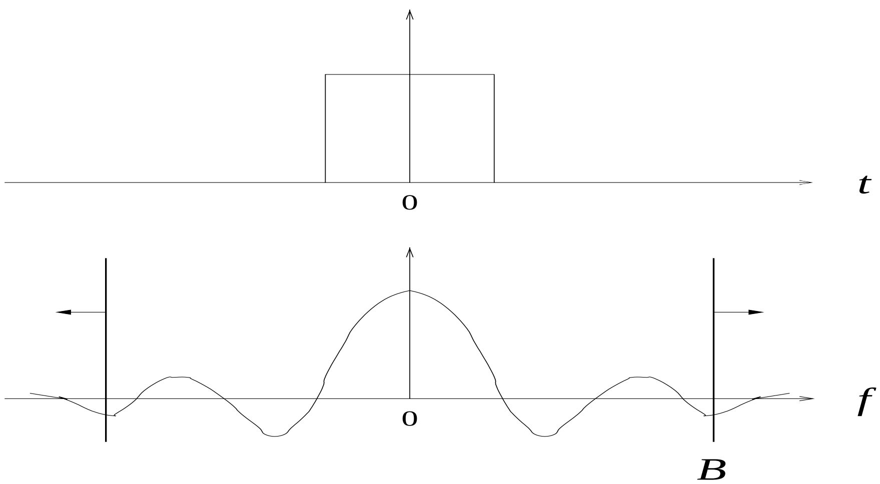
Of course, Fourier transform of $x_1(t)$



won't be simple square function and not even bandlimited

$\rightarrow$ how to deal with it?

For example: If $x_1(t)$ is square wave and its Fourier transform $F_{x_1}(f)$ a sync function



then treat $x_1(t)$ as bandlimited by ignoring frequencies greater than a cut-off threshold B

$\rightarrow$ then apply carrier frequency separation at least B plus guardband

$\rightarrow$ note: there is ICI (ignoring doesn't make it go away)

$\rightarrow$ goal: ICI at the level of minor noise

$\rightarrow$ bits decoded successfully with high likelihood

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

→ orthogonal FDM

Conceptual similarity to linear algebra

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,

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

→ called dot product (or inner product)

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

→ also basis of 3-D

→ 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

→ but not orthonormal

→ how to make them orthonormal?

Relevance to networking:

In CDMA (code division multiple access)—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

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

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

$\rightarrow$ CDMA has additional twists (discussed in wireless)

$\rightarrow$ but the above is essential idea

Back to orthogonal FDM (OFDM)

→ key idea: use carrier waves that are orthogonal

Dot product of two vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$

$$x \circ y = \sum_{i=1}^{n} x_i y_i$$

"Dot product" of two sinusoids $x(t) = \sin f_x t$ and $y(t) = \sin f_y t$

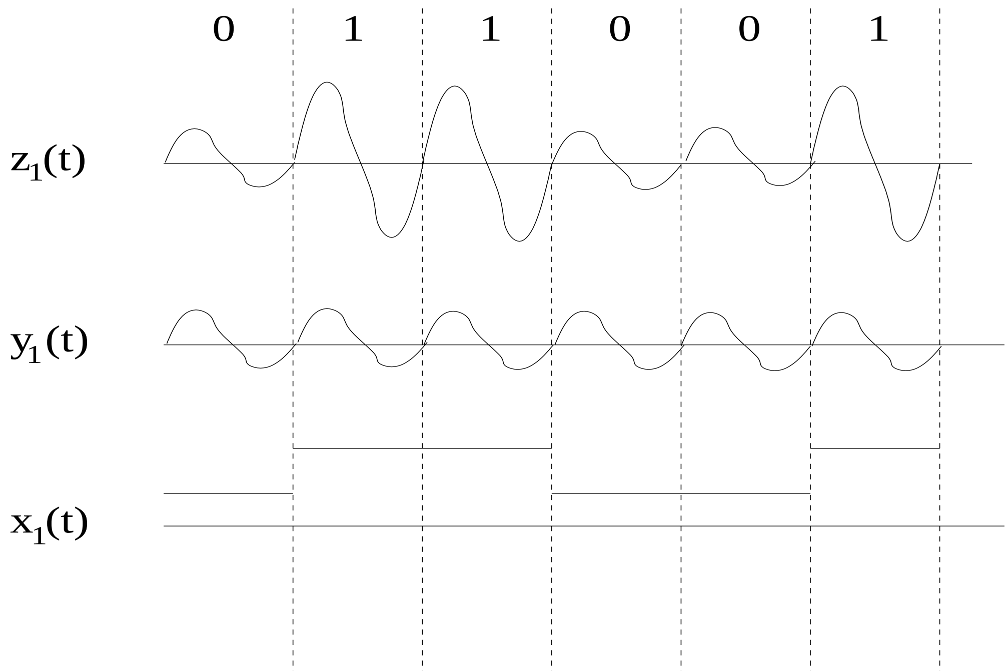
$$x(t) \circ y(t) = \int_{-\infty}^{\infty} (\sin f_x t)\,(\sin f_y t)\,dt$$

→ again: just a sum of products

More generally: $x(t) \circ y(t) = \int_{-\infty}^{\infty} e^{i f_x t} e^{-i f_y t} dt$

→ since Fourier transform involves complex sinusoids

User 1 uses carrier wave $y_1(t)$ to transmit bit stream (high and low) given by $x_1(t)$



Same for users 2 and 3

Suppose carrier waves $y_1(t)$, $y_2(t)$, $y_3(t)$ are orthogonal

Then receiver sees $z_1(t) + z_2(t) + z_3(t)$ which is

$$\sum_{k=1}^{3} x_k(t) y_k(t)$$

To decode what user 1 has sent, receiver computes dot product with $y_1(t)$

$$\begin{aligned} y_1(t) \circ \left(\sum_{k=1}^{3} x_k(t) y_k(t) \right) &= \sum_{k=1}^{3} x_k(t) \big(y_1(t) \circ y_k(t) \big) \\ &= x_1(t) \big(y_1(t) \circ y_1(t) \big) \\ &= x_1(t) \end{aligned}$$

→ last steps holds if also orthonormal

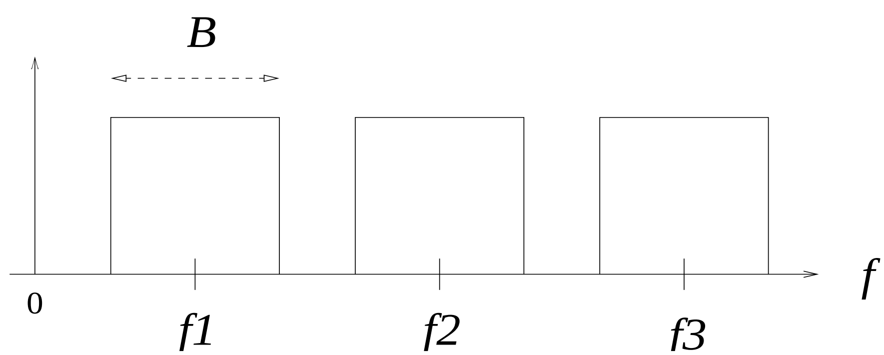
But look at Fourier transform formula (lecture notes, part 2):

→ just taking dot product!

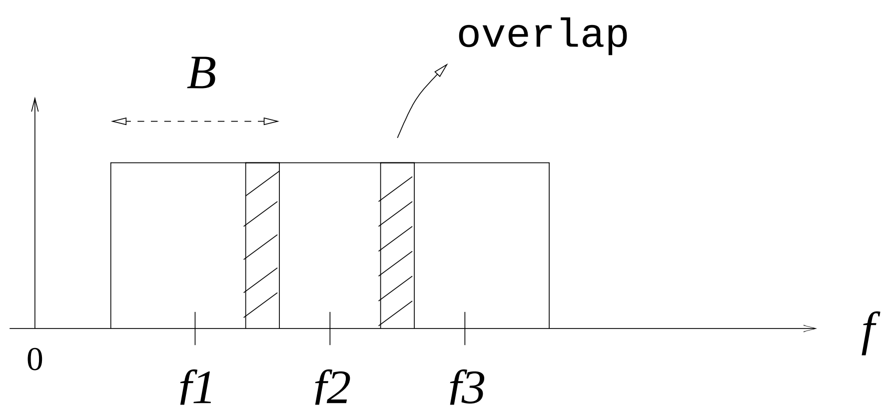
OFDM's advantage over FDM:

$\rightarrow$ don't need to worry about spectra of $x_1(t)$, $x_2(t)$, $x_3(t)$

FDM:



OFDM:



$\rightarrow$ spectra allowed to overlap

Can pack more carrier frequencies within given frequency band

→ called spectral efficiency

Technique known since the mid-1960s

→ only recently practically feasible

→ discrete Fourier transform (DFT)

→ implementation issue

Suppose usable frequency band is from f_a (Hz) to f_b (Hz)

Frequency band: $W = f_b - f_a$

$\rightarrow$ ex.: $f_a = 2.4$ GHz and $f_b = 2.5$ GHz, $W = 100$ MHz

To support N users or parallel bit streams set

$\rightarrow$ carrier spacing: $\bar{f} = W/N$

Then N carrier waves—called sub-carriers—are

$\rightarrow f_a, f_a + \bar{f}, f_a + 2\bar{f}, \ldots, f_a + (N-1)\bar{f}$

Easy to check they are orthogonal

$\rightarrow$ take dot product

Since this works for any N, does it mean we can support arbitrarily many parallel streams?

Not quite: physics of signal propagation imposes constraints

→ wireless: symbol period τ cannot be too short

→ multi-path propagation

→ leads to ISI (inter-symbol interference)

→ different from ICI (inter-carrier interference)

Symbol period (or time) determines carrier spacing $\bar{f}$

→ $\bar{f} = 1/\tau$

Thus: total number of sub-carriers N

→ $N = W/\bar{f}$

Example: $\tau = 3.2$ μs in IEEE 802.11g WLANs

→ $\bar{f} = 1/\tau = 312.5$ kHz

→ $W = 20$ MHz, $N = W/\bar{f} = 64$

Wireline: frequency spacing influenced by noise factors

$\rightarrow$ e.g., ADSL: $\bar{f} = 4.3125$ kHz

$\rightarrow$ ITU G.992.1 standard

$\rightarrow$ copper wire: UTP (unshielded twisted pair)

$\rightarrow$ frequency band: 0–1.104 MHz

$\rightarrow$ $N = W/\bar{f} = 256$

Frequency band 0–4 kHz used for voice

$\rightarrow$ also called POTS (plain old telephone service)

In the not-too-distant past, it was commonly held that UTP copper can support at maximum 30 kbps

$\rightarrow$ today's speeds: several Mbps range