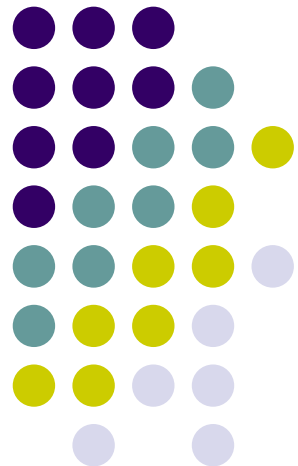
Señales discretas

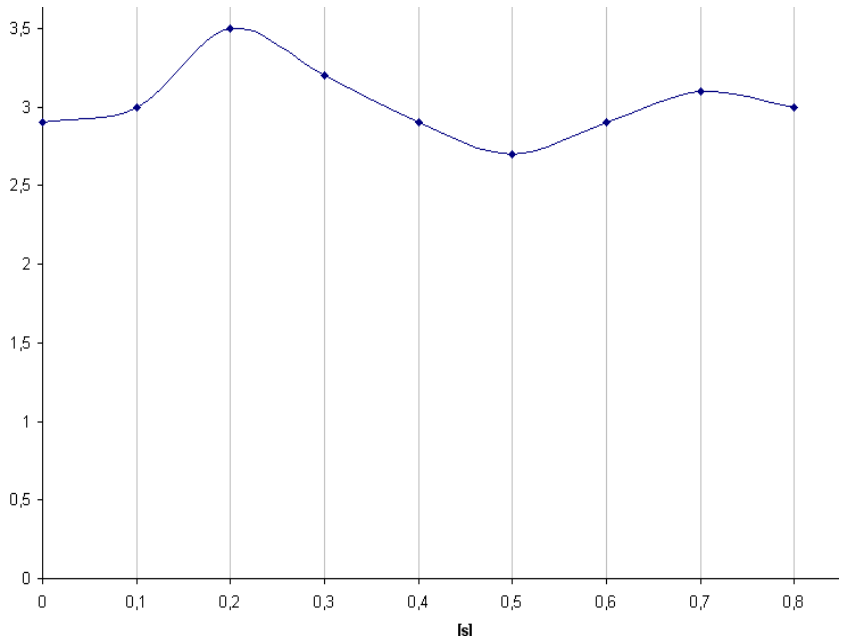
Patricio Loncomilla



Muestreo

- Consiste en tomar puntos de una señal, separados por un cierto periodo de muestreo. De este modo se puede obtener una señal discreta en vez de una continua.

$$T_s = 0.1$$
$$f_s = 10$$



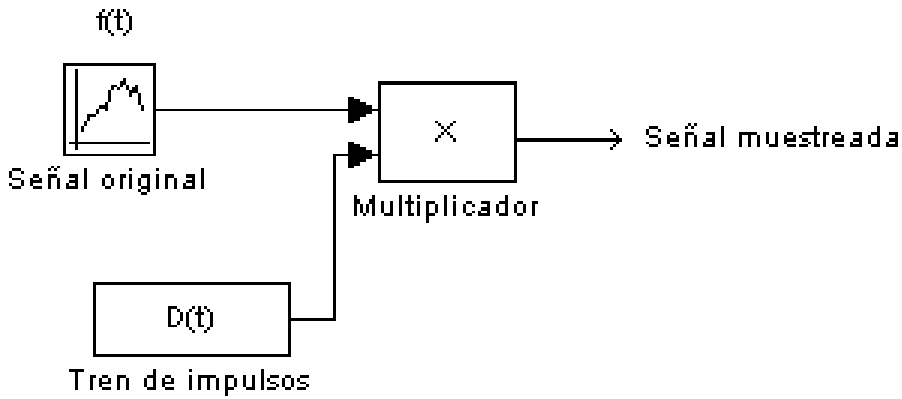
$f(t)$: señal continua

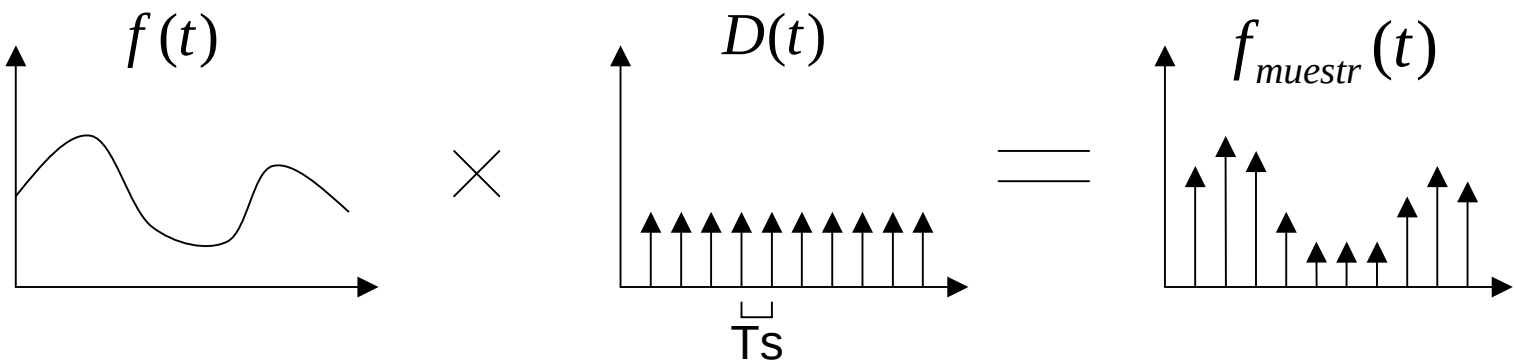
$x(n)$: señal discreta

$$x(n) = f\left(n \times T_s\right)$$

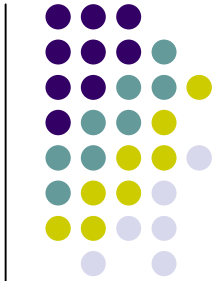
Muestreo ideal

- La señal es multiplicada por un tren de impulsos.





Espectro de la señal muestreada



El espectro de la señal original es :

$$F(\omega) = \Im\left(f(t) \right)$$

$$F_{muestr}(\omega) = \Im\left(f(t) \times D(t) \right)$$

$$F_{muestr}(\omega) = \Im\left(f(t) \times \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \right)$$

$$F_{muestr}(\omega) = \Im\left(f(t) \times \frac{1}{T_s} \sum_{n=-\infty}^{\infty} e^{jn\omega_0 t} \right) \text{con } \omega_0 = \frac{2\pi}{T_s}$$

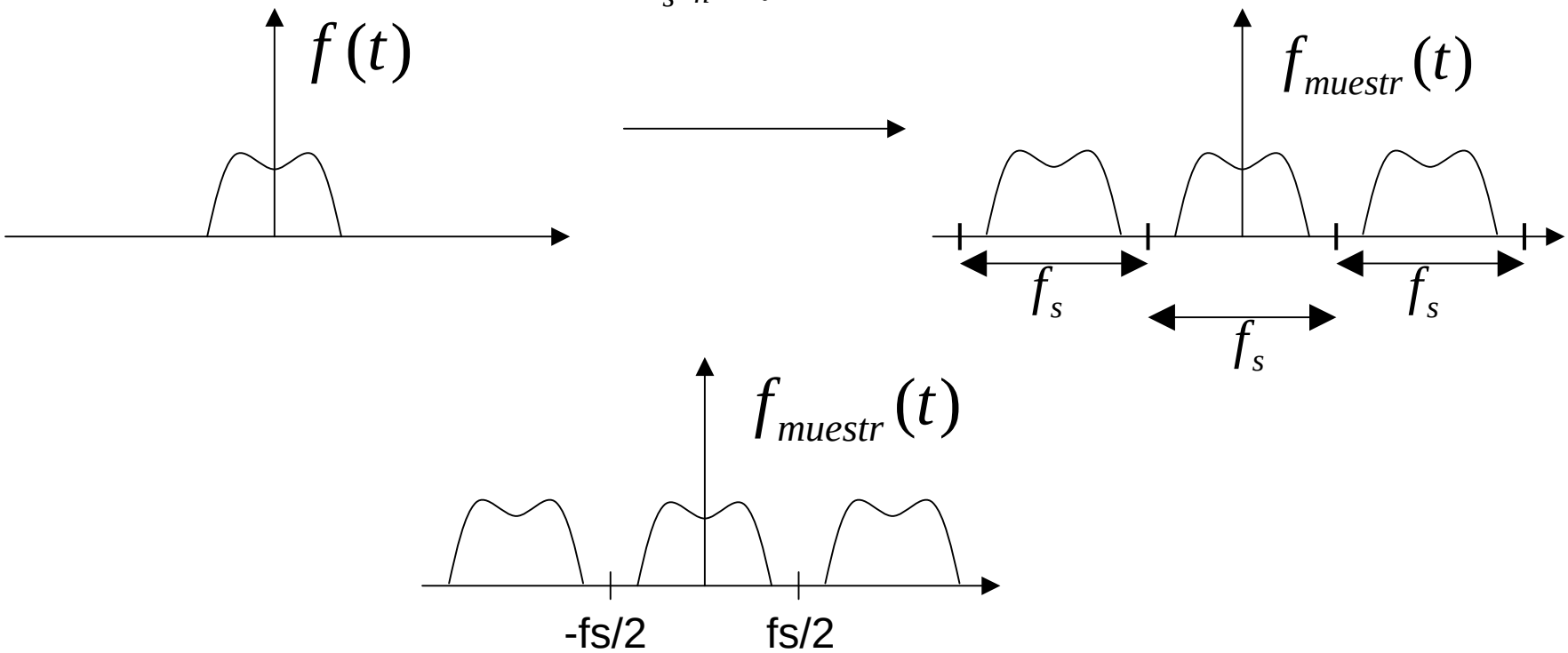
$$F_{muestr}(\omega) = \Im\left(\frac{1}{T_s} \sum_{n=-\infty}^{\infty} f(t) e^{jn\omega_0 t} \right)$$

$$F_{muestr}(\omega) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} F(\omega - n\omega_0)$$

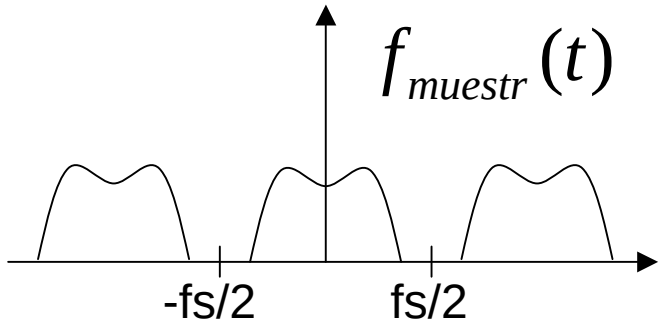
Teorema del Muestreo

$$F_{muestr}(\omega) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} F(\omega - n\omega_0) \text{ para } \omega$$

$$F_{muestr}(f) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} F(f - nf_s) \text{ para } f$$



Teorema del muestreo



- El espectro de la señal muestreada se obtiene sumando varios espectros de la señal original que están desplazados
- Si la señal original contiene frecuencias mayores a fs/2, se va a producir un traslape (aliasing) al sumar los espectros. Esto va a impedir recuperar la señal original.

Teorema del Muestreo

- Una señal se puede muestrear y luego reconstruir únicamente y sin sufrir alteraciones si su espectro está entre –fs/2 y fs/2

$$f_M < \frac{f_s}{2}$$

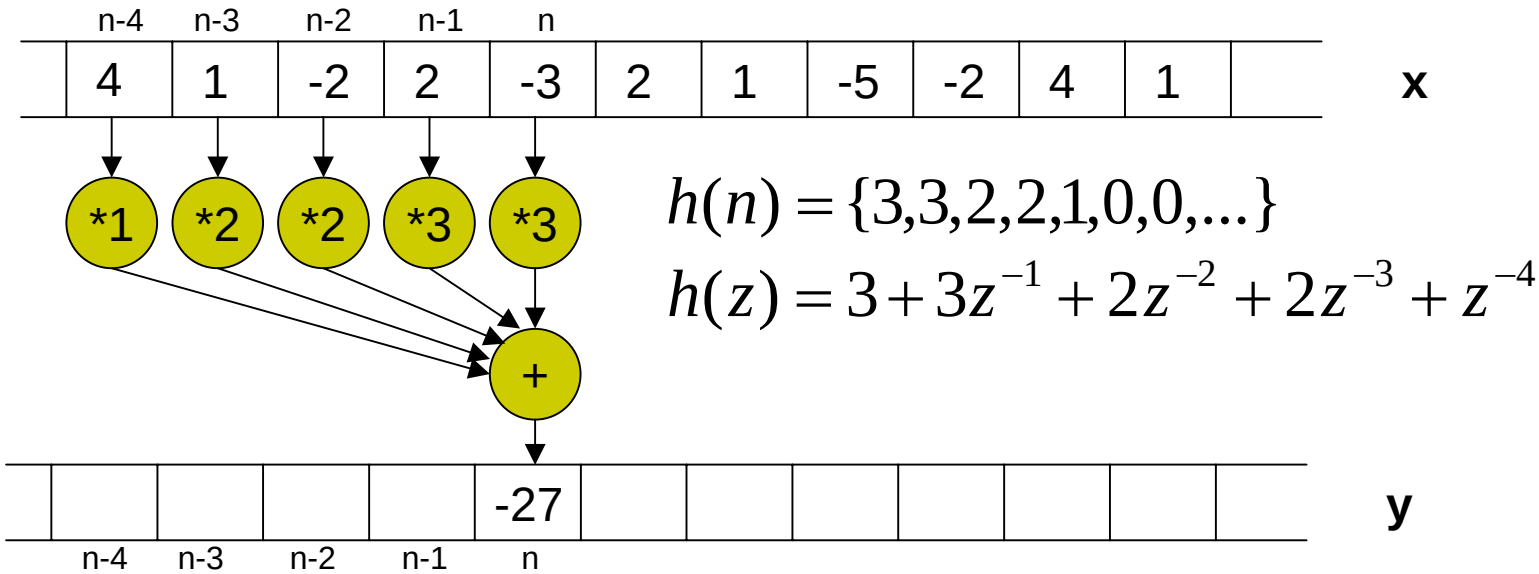
Mayor frecuencia contenida en la señal

Frecuencia o tasa de muestreo

Filtros discretos lineales

- FIR: la salida es una combinación lineal (una suma) de la entrada actual y las anteriores. Si el filtro es no causal, también pueden entrar a la suma entradas posteriores a la actual. Ejemplo:

$$y(n) = 3x(n) + 3x(n-1) + 2x(n-2) + 2x(n-3) + x(n-4)$$



$$h(n) = \{3,3,2,2,1,0,0,\ldots\}$$

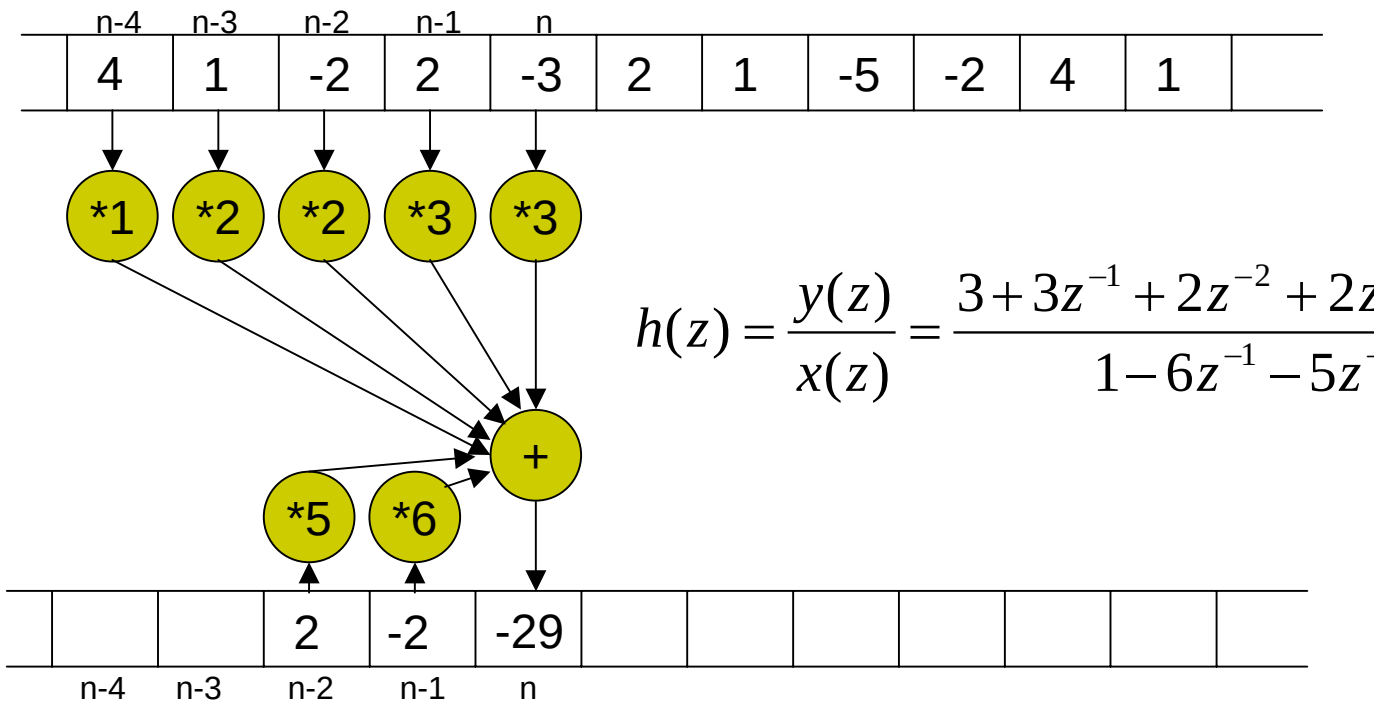
$$h(z) = 3 + 3z^{-1} + 2z^{-2} + 2z^{-3} + z^{-4}$$

Filtros discretos lineales

- IR: La salida es una combinación lineal de la entrada actual, entradas anteriores (y futuras) y salidas anteriores. Requieren condiciones iniciales. Ejemplo:

$$y(n)-6y(n-1)-5y(n-2)=3x(n)+3x(n-1)+2x(n-2)+2x(n-3)+x(n-4)$$

$$\Rightarrow y(n)=6y(n-1)+5y(n-2)+3x(n)+3x(n-1)+2x(n-2)+2x(n-3)+x(n-4)$$



$$h(z)=\frac{y(z)}{x(z)}=\frac{3+3z^{-1}+2z^{-2}+2z^{-3}+z^{-4}}{1-6z^{-1}-5z^{-2}}$$

DFT, FFT

- DFT: Transformada discreta de Fourier: Nace de hacer la integral numéricamente. También hay una DFT inversa.

$$F(\omega) = \int_{t=-\infty}^{\infty} f(t) e^{-j\omega t} dt; \quad f(t) = \frac{1}{2\pi} \int_{\omega=-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega$$

$$\Rightarrow X(k) = \sum_{n=0}^{N-1} x(n) W^{kn}; \quad x(n) = \sum_{k=0}^{N-1} X(k) W^{-kn} \text{ con } W = e^{-j\frac{2\pi}{N}}$$

Para k=0..N-1 Para k=0..N-1

- FFT: Algoritmo para calcular la DFT rápidamente (aprovecha mejor los valores ya calculados). Requiere que el número de puntos sea potencia de 2. Si no se cumple esto, se deben agregar ceros para que se cumpla.

DFT, FFT

Sea $W_N = e^{-j\frac{2\pi}{N}} \Rightarrow \frac{1}{N}$ de un ángulo de -360^o $(N = 4 \Rightarrow -90^o)$

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}$$ Separamos en n pares e impares :

$$X(k) = \sum_{\text{n pares}} x(n) W_N^{kn} + \sum_{\text{n impares}} x(n) W_N^{kn}$$

$$pares = \{0,2,\ldots, N-2\} \Rightarrow n = 2r \Rightarrow r \in \left\{0,1,\ldots,\frac{N}{2}-1\right\}$$

$$impares = \{1,3,\ldots, N-1\} \Rightarrow n = 2r+1 \Rightarrow r \in \left\{0,1,\ldots,\frac{N}{2}-1\right\}$$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_N^{2rk} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_N^{(2r+1)k}$$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) (W_N^2)^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) (W_N^2)^{rk}$$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_N^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_N^{rk}$$

$$X(k) = G(k) + W_N^k H(k)$$

DFT,FFT

sea $X(k) = FFT_k\{x0, x1, x2, x3, x4, x5, x6, x7\}$

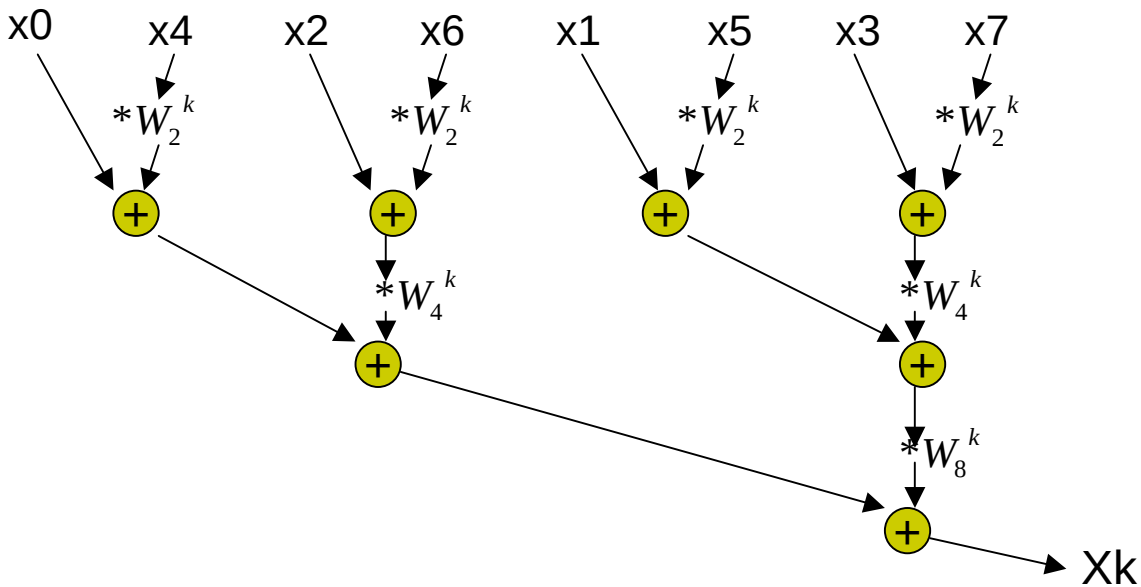
$$FFT_k\{x0, x1, x2, x3, x4, x5, x6, x7\} =$$

$$= FFT_k\{x0, x2, x4, x6\} + W_8^k FFT_k\{x1, x3, x5, x7\}$$

$$= FFT_k\{x0, x4\} + W_4^k FFT_k\{x2, x6\} + W_8^k FFT_k\{x1, x5\} + W_8^k W_4^k FFT_k\{x3, x7\}$$

$$= FFT_k\{x0\} + W_2^k FFT\{x4\} + W_4^k FFT\{x2\} + W_4^k W_2^k FFT\{x6\} + \ldots$$

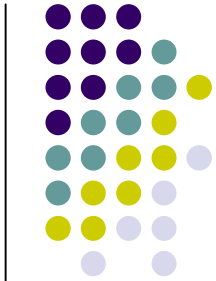
$$= x0 + W_2^k x4 + W_4^k x2 + W_4^k W_2^k x6 + \ldots$$



Ingreso de sonido a MATLAB

- [x,fs,nb]=wavread(‘nombre.wav’);
- wavwrite(x,fs,nb,’nombre.wav’);
  - x: vector con las muestras
  - fs: frecuencia de muestreo
  - nb: nº de bits con que se almacena la amplitud
  - ‘nombre.wav’: el nombre del archivo
- sound(x,fs); permite oír el sonido
- Otros comandos útiles: cd, dir, help, type, edit, fdatool

FFT con matlab



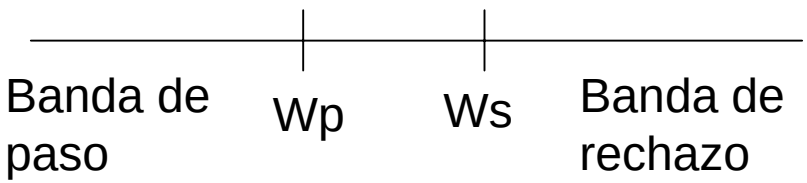
- yw=fft(y); calcula la fft de y, dejándola en yw
- y=ifft(yw); calcula la fft inversa de yw
- plot(abs(yw)); dibuja el espectro sin simetría
- plot(fftshift(abs(yw)); dibuja el espectro (sin unidades en el eje de frecuencia)
- N=length(yw); plot(((1:N)-floor(N/2))*fs/N, fftshift(abs(yw))); dibuja el espectro correctamente (coloca Hz en el eje x)
- figure; crea una nueva ventana de dibujo
- Plot(x,y1,x,y2);legend('y1(t)','y2(y)'); coloca nombres a las curvas
- Para mas usos de plot (colores, tipos de línea), ver help plot

Filtros con MATLAB

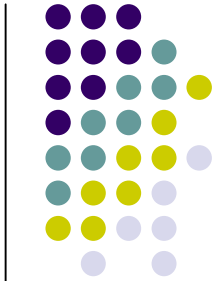
- FIR1
  - B=fir1(N,Wc,tipo); genera un filtro FIR pasaaltos o pasabajos usando ventana de Hamming.
    - N: nº de coeficientes
    - Wc = f_{corte} / (fs/2)
    - Tipo: ‘low’ para pasabajos, ‘high’ para pasaaltos
  - B=fir1(N,[W1 W2],’stop’); genera un rechazabandas, con ‘DC-0’ genera un pasabandas
    - N: nº de coeficientes
    - W1 = f_{corte1} / (fs/2) ; W2 = f_{corte2} / (fs/2)
- El filtro se puede aplicar sobre una señal x escribiendo:
  - y=filter(B,[1],x);

Filtros con MATLAB

- BUTTER, BUTTORD
  - [B,A]=butter(N,Wn,tipo); genera un filtro digital que aproxima a uno analógico de orden N (A y B tienen N+1 coeficientes). Wn y tipo ya se describieron para FIR1(Wn debe ser [w1 w2] para pasabandas)
  - [N,Wn]=buttord(Wp,Ws,Rp,Rs): Devuelve el orden N del filtro butter más pequeño que atenúa menos de Rp dB en la banda de paso y más de Rs dB en la de rechazo. Wp y Ws pueden ser vectores con varias frecuencias que definan bandas. El filtro se aplica con: y=filter(B,A,x);

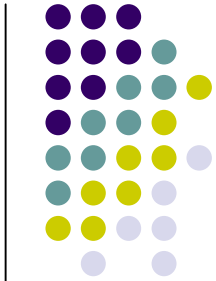


Filtros con MATLAB



- KAISERORD (ventanas Kaiser)
  - [N,Wn,BETA,TYPE]=kaiserord(F,A,DEV,fs); devuelve los parámetros para generar el siguiente filtro:
    - B = FIR1(N, Wn, TYPE, kaiser(N+1,BETA), 'noscale')
  - El vector F indica frecuencias de corte en Hz.
    - F=[f1 f2] => bandas (0,f1), (f2,fs/2)
    - F=[f1 f2 f3 f4] => bandas (0,f1), (f2,f3), (f4,fs/2)
  - El vector A indica la amplitud deseada en cada banda
  - El vector DEV indica la máxima amplitud del ripple en cada banda
  - fs es la frecuencia de muestreo

Filtros con MATLAB



- Visualizar la respuesta en frecuencia:
  - freqz(B,A,n,fs); muestra la respuesta en frecuencia de un filtro.
    - B: vector con coeficientes del numerador
    - A: vector con coeficientes del denominador
    - n: nº de puntos que tiene la señal a filtrar
    - fs: frecuencia de muestreo
- Para más detalles, ver el tutorial que está en la página