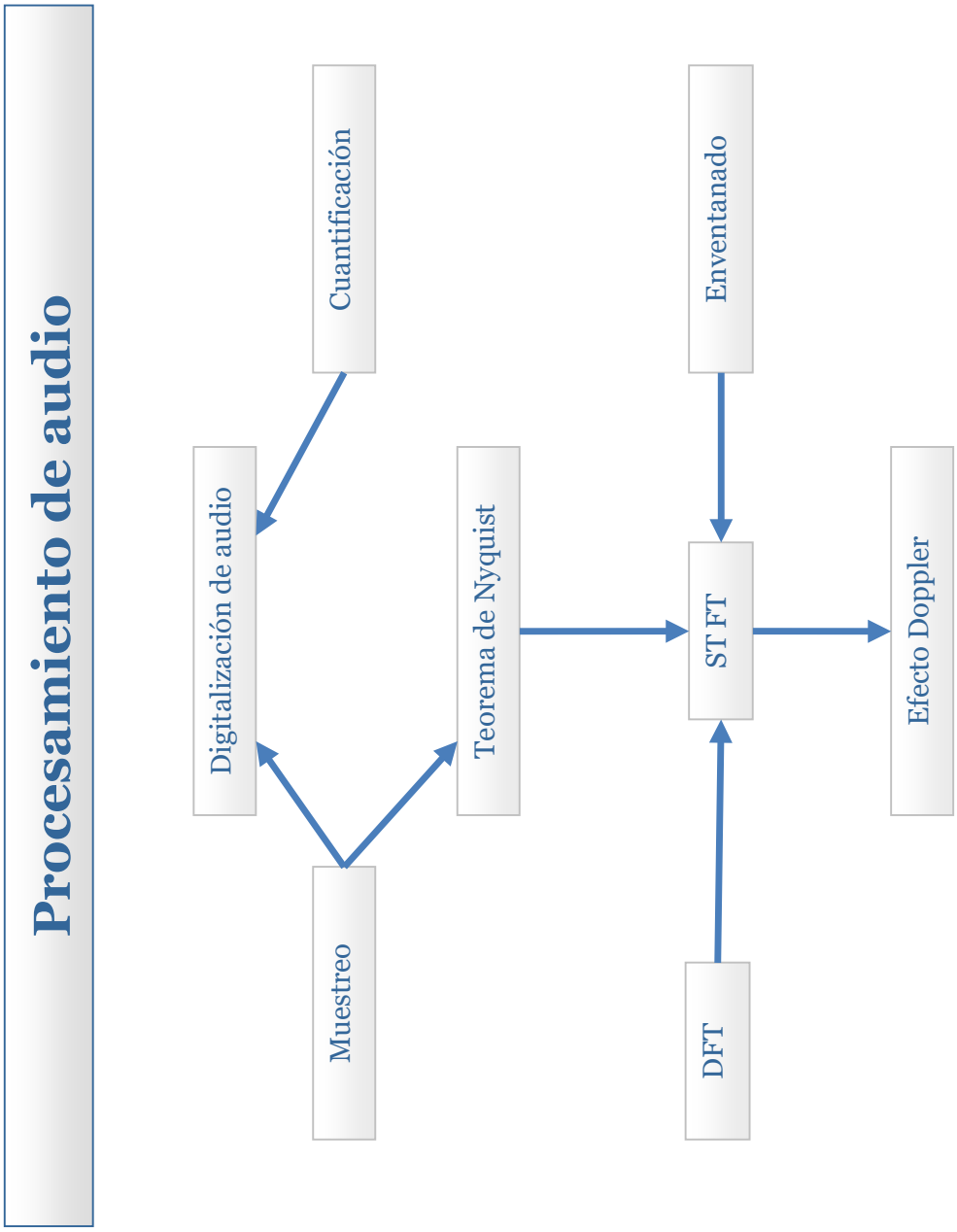
Procesamiento de audio

TEMA 10

Esquema



Ideas clave

10.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee las **Ideas clave** que encontrarás a continuación.

En este tema aprenderemos los principios de la digitalización de audio. Después veremos cómo los principios de la transformada de Fourier pueden aplicarse al análisis de señales de audio.

10.2. Digitalización de audio

El micrófono convierte una señal analógica donde el voltaje varía continuamente en muestras discretas mediante los procesos de muestreo y cuantificación que realiza un **Conversor Analógico Digital** (CAD). En concreto, el micrófono:

» **Muestrea**. Mide el voltaje a intervalos de tiempo, habitualmente equiespaciados. En el tema 2 vimos que se llama **periodo de muestreo** t_s al tiempo entre dos muestras (en segundos) y se llama **frecuencia cíclica de muestreo** f_s a al número de muestras que se toman en un segundo. También vimos el concepto de **frecuencia angular de muestreo** w_s, aunque en procesamiento de audio es más habitual trabajar con la frecuencia cíclica f_s. Matemáticamente estas magnitudes están relacionadas de la forma:

$$t_s = \frac{1}{f_s} = \frac{2\pi}{w_s} \qquad f_s = \frac{1}{t_s} \qquad w_s = \frac{2\pi}{t_s} \quad w_s = 2\pi f_s$$

» **Cuantifica**. Asigna un valor numérico discreto a la amplitud de cada muestra. En la práctica la cuantificación se hace con valores de 8 bits, 16 bits, o32 bits. Esto suele dar lugar a valores discretos en los rangos [-128..127], [-32768..32767] y [-2147483648.. 2147483647], respectivamente.

Para reproducir el audio en un altavoz el **Conversor Digital Analógico** (CDA) realiza el proceso contrario. Es decir, recibe la señal digitalizada y genera variaciones de voltaje continuas en el altavoz.

Generación de un audio sintético

Para demostrar el proceso de muestreo podríamos empezar grabando audio pero para entender mejor su relación con el periodo y la frecuencia de muestreo vamos a generar 3 tonos sinusoidales a frecuencias f_{s1}=250Hz, f_{s2}=1000Hz y f_{s3}=2000Hz en Octave y los vamos a sumar. La figura 1 muestra los tonos generados.

Dado que en procesamiento de audio se suele trabajar con frecuencia cíclica, las funciones que representan estos tres tonos aparecen multiplicadas por 2π, es decir:

$$x_1(t) = A_1 \cos(2\pi 250t)$$

$$x_2(t) = A_2 \cos(2\pi 1000t)$$

$$x_3(t) = A_3 \cos(2\pi 2000t)$$

Donde A_i es la amplitud de cada señal. Si el tiempo entre muestras es nt_s, podemos obtener los valores de la señal de tiempo discreto como $x[n] = x(nt_s)$, donde n es un entero que indexa las muestras.

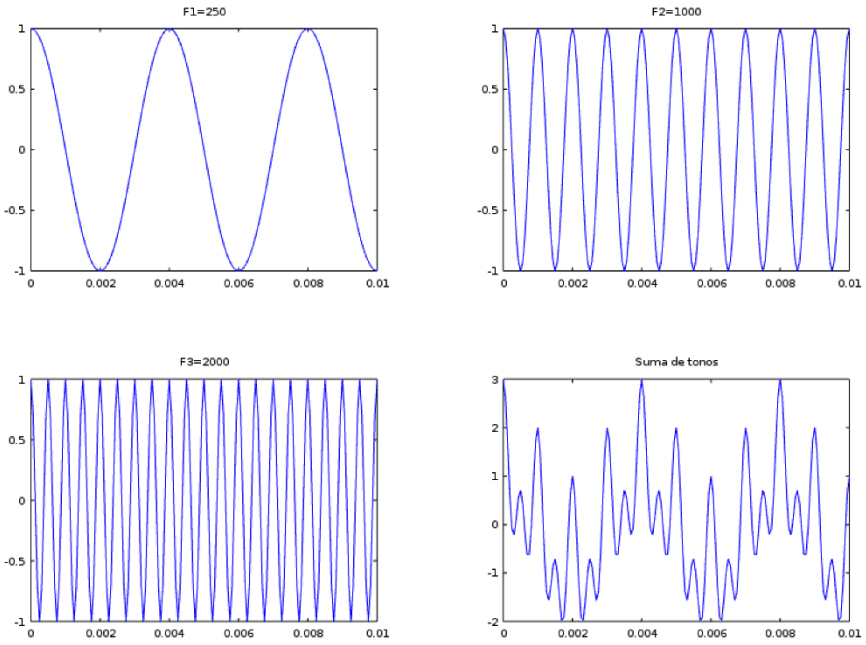


Figura 1: tonos sintéticos

Como frecuencia de muestreo vamos a usar f_s=16000Hz, con lo que el periodo de muestreo será de $T_s = 1/f_s = 0.0000625$ seg. Es decir, haremos:

```
Fs = 16000; inct = 1/Fs;
```

Si queremos representar tonos de **duración** d=0.05 seg de duración y conocemos que el periodo de muestreo t_s (inct) es de 0.0000635 seg, sabemos que necesitamos almacenar un total de $n=d/t_s$=0.01/0.0000625 = 160 muestras, distanciadas entre sí por el periodo de muestreo t_s=0.0000625. Es decir, en Octave podemos obtener los valores de t en los que generar las muestras como:

```
d = 0.01;
t = [0:inct:d-inct];
```

Aplicando la ecuación del tono sinusoidal para amplitud 1 y las tres frecuencias de tono propuestas tendríamos:

```
A = 1;
F1 = 250; F2= 1000; F3=2000;
y1 = A*cos(2*pi*F1*t)';
y2 = A*cos(2*pi*F2*t)';
y3 = A*cos(2*pi*F3*t)';
y = y1+y2+y3;
```

Donde y es el resultado de sumar las muestras de los 3 tonos, tal como se muestra en la figura 1. Para representar audio Octave acostumbra a usar vectores columna. El apóstrofe (') que aparece al final del cálculo de y1, y2, y3 convierte la matriz fila en una matriz columna.

Finalmente podemos dibujar y etiquetar los tonos con los comandos:

```
subplot(2,2,1), plot(t,y1), title('F1=250');
subplot(2,2,2), plot(t,y2), title('F1=1000');
subplot(2,2,3), plot(t,y3), title('F1=2000');
subplot(2,2,4), plot(t,y), title('Suma de tonos');
```


Reproducción de los tonos

Para reproducir un sonido en Octave podemos usar el comando *sound(x,Fs,nBits)* el cual recibe en los parámetros:

- *x*. Las muestras a reproducir. Tradicionalmente Octave ha usado vectores columna para el audio pero si se los pasamos el un vector fila también los acepta.
- *Fs*. La frecuencia de muestreo de la señal de audio.
- *nBits*. La señal en *x* debe estar representada con valores en coma flotante entre -1.0 y 1.0. Sin embargo estos valores deben ser cuantificados a 8, 16 o 32 bits al enviarlos al dispositivo de salida. *nBits* indica el número de bits de cuantificación. Si no se proporciona este valor, por defecto se usa 8 bits.

Vamos a volver a generar la señal anterior pero en este caso con duración de 1seg, es decir:

```
d = 1;
t = 0:inct:d;
y1 = A*cos(2*pi*F1*t)';
y2 = A*cos(2*pi*F2*t)';
y3 = A*cos(2*pi*F3*t)';
y = y1+y2+y3;
```

El otro problema con el que nos vamos a encontrar ahora es que la señal y no está normalizada en el rango [-1.0,1.0]. Para normalizarla podemos hacer:

```
y = y ./ max(abs(y));
```

Ahora podemos reproducir esta suma de tonos con el comando:

```
sound(y,Fs);
```

Señales mono y estéreo

En Octave 1 vector columna representa una señal mono y 2 vectores columna representa una señal estéreo. Por ejemplo, podemos generar una señal de audio que en el lado

izquierdo genere un tono de 250Hz y en el lado derecho un tono de 2000Hz con los comandos:

```
estereo = [y1 y3];
sound(estereo,Fs);
```

Una forma sencilla de convertir este sonido estéreo en mono es promediando las muestras de lado izquierdo y derecho:

```
mono = sum(estereo,2)./2;
```

Grabar y leer ficheros de audio

Para guardar las señales de audio generadas podemos usar el comando *audiowrite*, el cual tiene las siguientes formas de ejecutarse:

```
audiowrite(filename y,)
audiowrite(filename y,Fs,)
```

El nombre del fichero *filename* y las muestras a grabar y son parámetros obligatorios.

» El parámetro opcional que podemos no proporcionar es a frecuencia de muestreo que por defecto vale Fs=8000.

Los valores de las muestras de *y* deben de estar en rango -1,0 a 1.0. De lo contrario la operación las normaliza antes de generar el fichero.

Podemos guardar los tonos generados en la subsección anterior usando el comando *audiowrite* de la forma:

```
audiowrite('tono1.wav',y1,Fs);
```

Análogamente, podemos usar el comando *audioread* para leer un fichero de audio de la forma:

```
[y,Fs] = audioread('tono1.wav');
```


Donde además de las muestras y, se nos devuelve la frecuencia muestreo del fichero Fs y la profundidad de bits nBits de las muestras.

También podemos indicar un número de muestras N a leer:

```
[y,Fs,N] = audioread('tono1.wav',N);
```

O incluso indicar un rango de muestras a leer:

```
[y,Fs,N] = audioread('tono1.wav',[N1 N2]);
```

10.3. Frecuencia de muestreo

Intuitivamente parece claro que a mayor frecuencia de muestreo, mayor será la fidelidad de la señal de audio capturada y reproducida. Lo que no queda claro es cuál debe ser la frecuencia de muestreo para poder reproducir señales de audio audibles. En esta sección vamos a determinar estas frecuencias y vamos a ver qué ocurre cuando no se cumplen

El criterio de Nyquist

El **criterio de Nyquist** nos dice que la frecuencia de muestreo f_s debe ser al menos el doble que la señal con mayor frecuencia que queramos digitalizar.

El oído humano es sensible a señales acústicas entre 20Hz y 22000Hz. Si aplicamos el criterio de Nyquist podemos determinar que la frecuencia de muestreo que nos permite representar todos los sonidos audibles debería ser un poco mayor a 2*22000Hz=44000Hz. Por esta razón los formatos de digitalización de audio de alta fidelidad utilizan como frecuencia de muestreo 44100Hz.

Aunque 44100Hz es la frecuencia de muestreo que nos permite registrar todos los sonidos audibles, muchos formatos de audio reducen esta frecuencia para reducir la cantidad de información a transmitir. Valores típicos son 22050Hz, 16000Hz, 11025Hz y 8000Hz. Tradicionalmente los sistemas de telefonía han usado 8000Hz, lo cual dificulta el escuchar letras aisladas por el teléfono o transmitir música de calidad.

También existen formatos con frecuencias de muestreo mayores. Un valor típico es 48000Hz, aunque existen formatos con valores mayores.

Aliasing

El ***aliasing*** es un artefacto que se produce cuando la frecuencia de muestreo es insuficiente con respecto a la frecuencia máxima de la señal (p.e. 22000Hz). Recordar que la frecuencia fundamental f_0 es la frecuencia más baja de la señal, en audio se suele llamar ***pitch*** y corresponde al tono más bajo percibido.

Dada una señal analógica $x(t)$, sabemos que el muestreo de una señal analógica $x_p(nt_s)$ crea una señal digital $x[n]$ cuya DTFS expresa una señal de tiempo discreto con frecuencias equiespaciadas en F_m=[0,1) donde:

$$F_m = m\frac{1}{N} \quad \text{con} \quad m = 0, \ldots, N-1$$

También sabemos que podemos relacionar las frecuencias de tiempo discreto de la DTFS (F_m) con las frecuencias de tiempo continuo (f_m) mediante la relación:

$$f_m = m\frac{f_s}{N} \quad \text{con} \quad m = 0, \ldots, N-1$$

Esto quiere decir que la señal analógica muestreada en frecuencia $X_p(f)$ es una superposición de infinitas copias de la FT de la señal original donde cada copia está centrada en una posición múltiplo de f_s. La figura 2 (a) muestra la representación espectral $X(f)$ de la señal analógica $x(t)$. La figura 2 (b) muestra la señal muestreada en frecuencia $X_p(f)$ en el caso en el que la frecuencia de muestreo f_s supera el criterio de Nyquist. La figura 2 (c) muestra el solapamiento en frecuencia que se produce en $X_p(f)$ cuando la frecuencia de muestreo es insuficiente y en consecuencia se produce el *aliasing*.

Ejemplo 1: generación de tonos.

Podemos usar Octave para estudiar lo que ocurre con nuestra suma de tonos a frecuencias f_1=500Hz, f_2=1000Hz, f_3=2000Hz, cuando la muestreamos a f_{s1}=3000 muestras/seg y f_{s2}=5000 muestras/seg:

$$x(t) = \cos(2\pi 500t) + \cos(2\pi 1000t) + \cos(2\pi 2000t)$$

El criterio de Nyquist dice que la frecuencia de muestreo debe ser al menos el doble de la señal de más frecuencia. En nuestro ejemplo 2000·2=4000 muestras/seg.

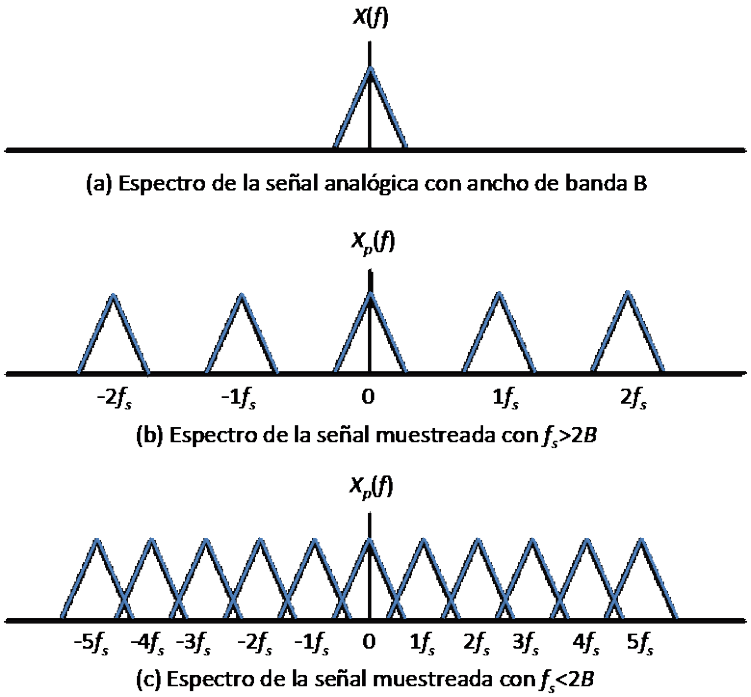
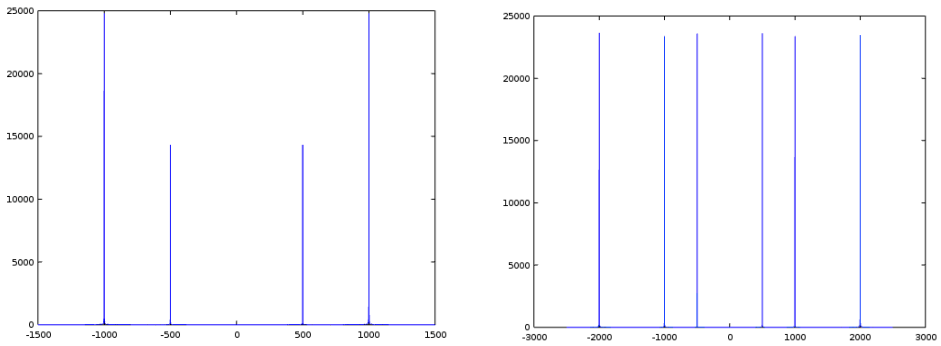


Figura 2: representación espectral del *aliasing*



(a) f_{s1}=3000 muestras/seg (b) f_{s1}=5000 muestras/seg

Figura 3: representación espectral de tonos a diferentes frecuencias de muestreo

La figura 3 (a) muestra la representación espectral del muestreo a f_{s1}=3000 muestras/seg. Podemos ver el *aliasing* donde el pulso a 2000Hz se suma al pulso a -1000Hz (y el pulso a -2000Hz se suma al pulso a 1000Hz). La figura 3 (a) muestra la representación espectral del muestreo a f_{s1}=5000 muestras/seg, donde no existe *aliasing*.

Para generar estas representaciones hemos usado los siguientes comandos:

```
Fs = 3000;
inct = 1/Fs;
t = [0:inct:10];
x = cos(2*pi*500*t) + cos(2*pi*1000*t) + cos(2*pi*2000*t);
X = fft(x);
N = length(X);
incf = Fs/N;
f = [-0.5*Fs:incf:0.5*Fs];  f = f(1:end-1);
plot(f,abs(fftshift(X)));
```

10.4. Enventanado

La señal de audio es una señal por naturaleza no estacionaria, es decir, que varía con el tiempo. Por esta razón, en la mayoría de las aplicaciones de análisis de audio la señal se divide **ventanas** de corta duración (típicamente de 10ms a 50ms) y se analiza la señal por ventana. Se llama **enventanado** a la operación de multiplicar una señal $x[n]$ por una ventana $w[n]$ de duración definida W_L:

$$w[n] = \begin{cases} 1, & 0 \leq n \leq W_L - 1 \\ 0, & \text{resto} \end{cases}$$

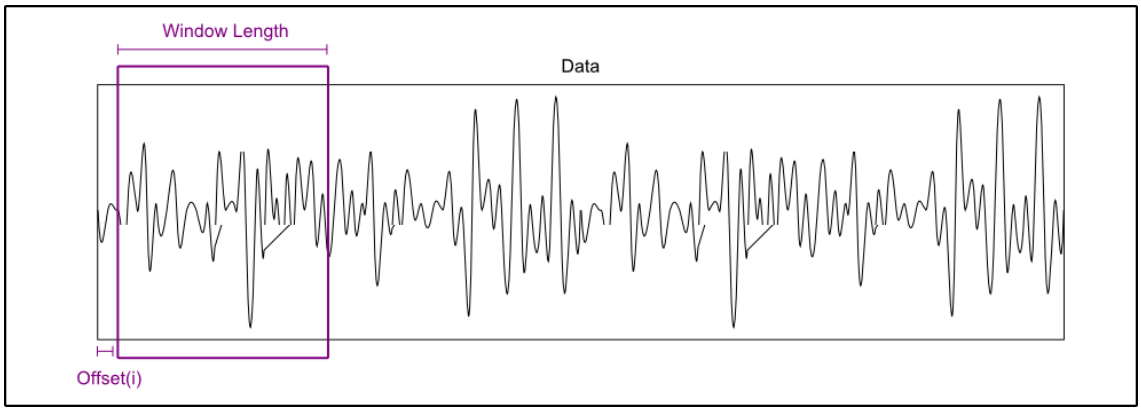


Figura 4: conceptos de enventanado

Fuente: https://kevinsprojects.wordpress.com/2014/12/13/short-time-fourier-transform-using-python-and-numpy/

La figura 4 muestra el concepto de enventanado. En concreto, en cada iteración multiplicamos la señal de audio $x[n]$ por una versión desplazada de la ventana. El resultado es la señal enventanada $x_i[n]$:

$$x_i[n] = x[n]w[n - iW_L]$$

Compromiso entre resolución frecuencial y espacial

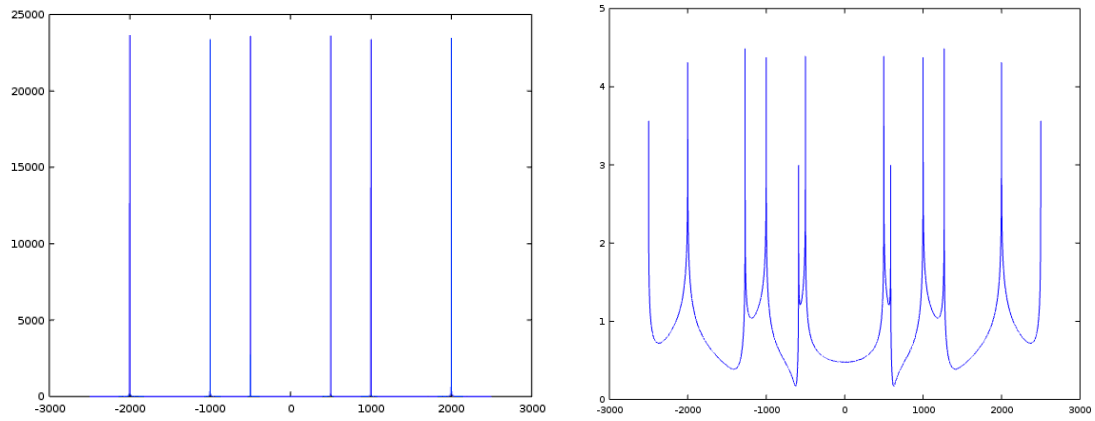
En el tema 9 vimos que secuencias temporales $x[n]$ con un N más grande daban lugar a una DFT con mayor resolución frecuencial. Sin embargo, el incremento de la resolución frecuencial es válido siempre que la señal permanezca estacionaria. Si las frecuencias que componen la señal cambian con el tiempo, el incremento de N da lugar a un promediado de las frecuencias en diferentes instantes de tiempo.
Esto quiere decir que existe un compromiso entre la resolución frecuencial y la resolución temporal. Experimentalmente se ha comprobado que los mejores análisis de señales de audio se producen con ventanas de entre 10 ms y 50 ms.

Goteo espectral

La limitación en el número de muestras que, por razones prácticas, puede procesar la DFT da lugar a un artefacto llamado **goteo espectral** en el dominio de la frecuencia por el cual la señal se desparrama por frecuencias distintas a la de la señal.

El goteo espectral se produce incluso con señales sencillas como son los tonos. En la figura 5 (a) hemos repetido el espectro que obtuvimos en el ejemplo 1, donde hicimos

una representación espectral de tres tonos. A simple vista no se observa el goteo espectral con frecuencia de muestreo suficiente como para que no exista *aliasing* (f_{s2}=5000 Hz).



(a) Escala lineal (b) Escala logarítmica

Figura 5: goteo espectral en tonos a frecuencias f_1=500Hz, f_2=1000Hz, f_3=2000Hz

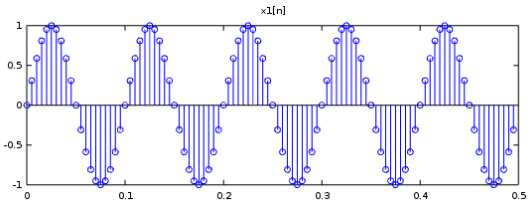
Para observar el goteo espectral resulta útil hacer una representación logarítmica de la frecuencia modificando el programa anterior de la forma:

```
plot(f,abs(fftshift(log10(X))));
```

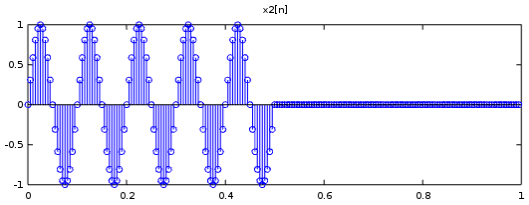
La figura 5(b) muestra el resultado de usar esta escala donde vemos que se produce un goteo espectral, especialmente en torno a la frecuencia de los tonos.

En el tema 9 también vimos que una forma de mejorar la resolución espectral sin aumentar el número de muestras (resolución temporal) es utilizar *padding*.
La figura 6 (a) muestra una sinusoidal y la figura 6 (b) muestra el resultado de añadirla *padding*. En la figura 6 (d) podemos observar que el *padding* ha introducido goteo espectral en la representación frecuencial de la señal. De hecho, en este caso no hemos necesitado usar una escala logarítmica para detectar el goteo espectral.

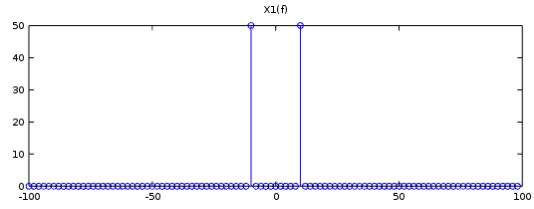
De nuevo existe un compromiso en el número de muestras *N* para las que es recomendable usar *padding*. En el tema 9 vimos que el *padding* mejoraba la resolución espectral de una señal con pocas muestras (*N*=5 en aquel ejemplo). En este ejemplo lo que vemos es que cuando el número de muestras es suficiente (*N*=100 en este ejemplo) el *padding* no mejora la resolución espectral y además introduce *padding*.



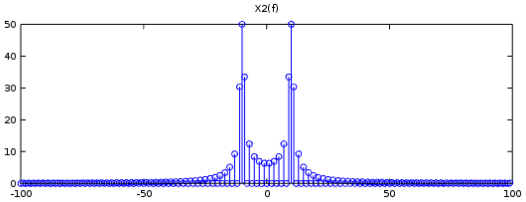
(a) Señal sin padding



(c) Señal con padding



(b) Representación frecuencial de (a)



(d) Representación frecuencial de (c)

Figura 6: el *padding* aumenta la resolución frecuencial y el goteo espectral

Los comandos usados para generar $x_1[n]$ y $X_1(f)$ han sido:

```
F0=10; % Frecuencia de la sinusoide
Fs=200; % Frecuencia de muestreo
t1=[0:1/Fs:0.5-1/Fs]; % 0.5 seg
x1=sin(2*pi*F0*t1); % Sinusoide muestreada
N1=length(x1); % Longitud de la DFT igual a longitud de la señal
X1 = fftshift(fft(x1,N1));
f1=(-N1/2:1:N1/2-1)*Fs/N1;
subplot(2,1,1), stem(t,x1), title('x1[n]');
subplot(2,1,2), stem(f1,abs(X1)), title('X1(f)');
```

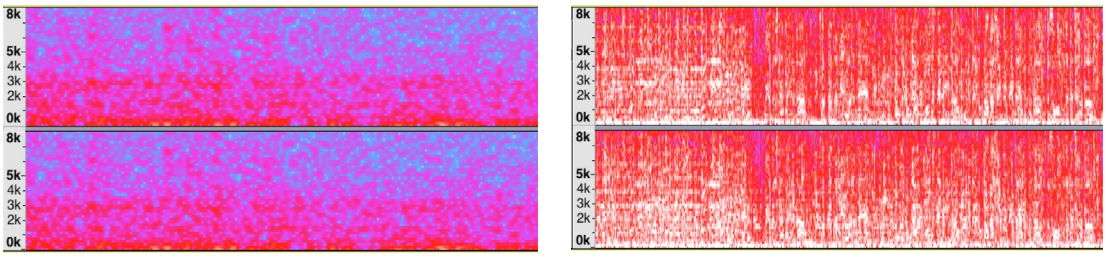
Y para generar $x_2[n]$ y $X_2(f)$:

```
t2 = [0:1/Fs:1-1/Fs]; % 1 seg
x2 = [x1 zeros(1,N1)]; % Sinusoide muestreada con padding
N2 = length(x2);
X2 = fftshift(fft(x2,N2));
f2=(-N2/2:1:N2/2-1)*Fs/N2;
subplot(2,1,1), stem(t2,x2), title('x2[n]');
subplot(2,1,2), stem(f2,abs(X2)), title('X2(f)');
```


10.5. La STFT

La STFT (*Short Time Fourier Transform*) es una representación frecuencia que se utiliza con señales que varían en el tiempo. La STFT consiste en dividir la señal en ventanas y calcular la DFT de cada ventana. Como hemos visto arriba, la longitud de la ventana juega un papel importante: ventanas más largas dan una mejor resolución espectral a cambio de reducir la calidad de la resolución temporal.

Un **espectrograma** es una imagen que representa la evolución de una señal en los dominios del tiempo y de la frecuencia. Para generarlo, los coeficientes DFT de cada ventana se colocan en una columna de una matriz. Las filas indican las diferentes frecuencias.



(a) Voz humana (hombre) (b) Música instrumental

Figura 7: espectrograma de voz humana frente a música instrumental

El espectrograma permite observar la evolución de las altas y bajas frecuencias de una señal de audio. En la figura 7 (a) vemos el espectrograma de una voz humana generado con Audacity para una señal estéreo.

Podemos ver que la mayoría de la energía se concentra en torno a los 2000Hz. En la figura 7 (a) vemos el espectrograma de música instrumental donde vemos que la mayoría de la energía se concentra en torno a los 7000hz y las bajas frecuencias aparecen vacías.

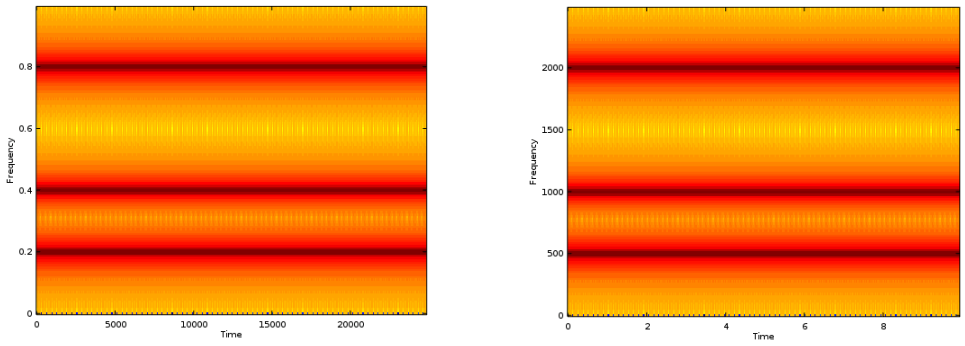
En Octave podemos calcular el espectrograma de una señal usando el comando *specgram(x,N,Fs)*, donde *N* indica el tamaño de la ventana y *Fs* la frecuencia de muestreo. Por ejemplo, para generar el espectrograma de los tres tonos del ejmplo 1 podemos ejecutar el comando:

```
specgram(x);
```


La figura 8 (a) muestra el espectrograma obtenido. Por defecto el comando asume f_s=2Hz, con lo que es muy recomendable pasar siempre la frecuencia a la que está muestreada la señal. Dado que nuestra señal está muestreada a f_s=5000Hz, la forma correcta de ejecutarlo es:

```
specgram(x,256,5000);
```

El resultado obtenido se muestra en la figura 8 (b) donde ahora sí podemos ver en el eje de coordenadas que la energía se concentra en torno a las frecuencias fundamentales de los tonos. Además, el eje de abscisas ha sido escalado para que la distancia entre muestras sea $t_s = 1/f_s = 1/5000$ seg.



(a) Espectrograma por defecto (b) Espectrograma con f_s=5000Hz

Figura 8: espectrograma de tres tonos

Por defecto el comando *specgram* usa una ventana de N=256 muestras. El ancho de la ventaja determina si tenemos mayor resolución frecuencial o temporal. La figura 9 muestra la forma que adoptan los coeficientes DFT del espectrograma en función de este parámetro.

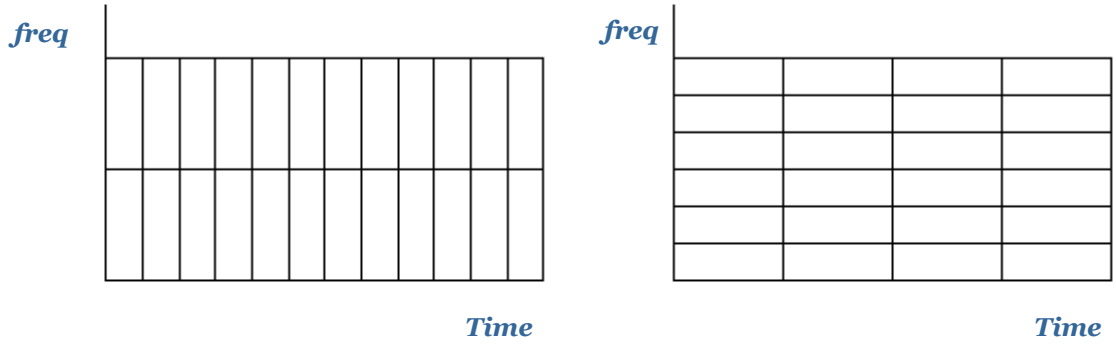
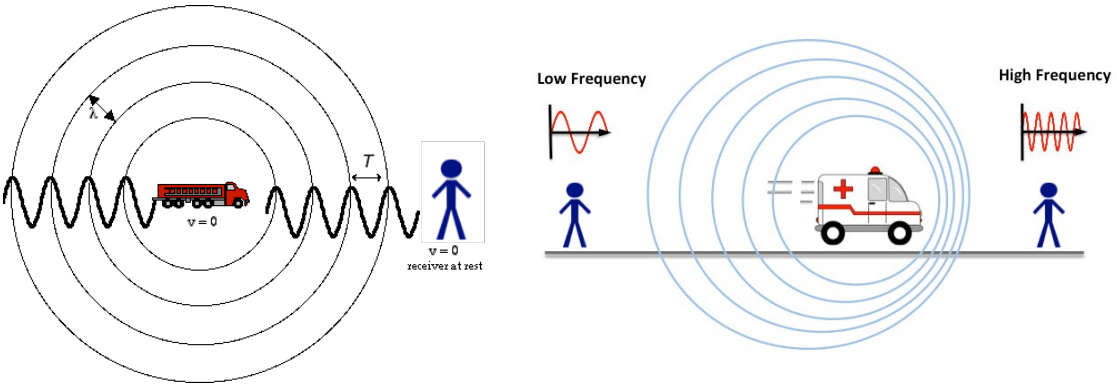


Figura 9: comparación entre la resolución espacial y temporal del espectograma

10.6. El efecto Doppler

Cuando está en movimiento el emisor de una onda (p.e. sonido, luz) se produce una diferencia entre la frecuencia a la que el emisor produce las ondas f_e y la frecuencia a la que el receptor las recibe f_r. Este fenómeno recibe el nombre de **efecto Doppler**, en honor a su descubridor, el físico Australiano Christian Doppler (1842).



(a) Frentes de onda (b) Efecto Doppler

Figura 10: efecto Doppler

Fuente: http://www-inst.eecs.berkeley.edu/~eegsa/or/sound.html

La figura 10 (a) muestra lo que ocurre cuando el emisor está en reposo. En la figura se muestran los sucesivos **frentes de onda**: ondas concéntricas centradas en el emisor y separadas por un periodo T. La figura 10 (b) muestra que cuando el emisor se acerca al receptor la frecuencia percibida en el receptor f_r aumenta y cuando se aleja disminuye.

Longitud de onda

Partimos del caso en el que emisor y receptor están en reposo. Sabemos que la frecuencia en reposo de la señal emitida f_e es el número de ciclos que hace una onda en un segundo (ciclos/seg) y el periodo T_e de la onda es el tiempo que la onda tarda en completar un ciclo de vibración (seg/ciclo).

Llamamos **longitud de onda** λ_e a la distancia recorrida por la onda en un periodo de vibración. Dado que ni la frecuencia ni el periodo incluyen una magnitud de distancia, tenemos que introducir una nueva magnitud para calcularla. Esta magnitud suele ser la

velocidad de propagación de la señal c (metros/seg). Ahora podemos calcular la distancia recorrida por la onda en un periodo:

$$\lambda_e = c \cdot T_e = \frac{c}{f_e}$$

Lo que nos dice el efecto Doppler es que cuando el emisor se está acercando al receptor, la **longitud de onda percibida por el receptor** se reduce $\lambda_r<\lambda_e$. Esto también implica que el periodo percibido por el receptor se acorta $T_r<T_e$ y la frecuencia percibida por el receptor aumenta $f_r>f_e$.

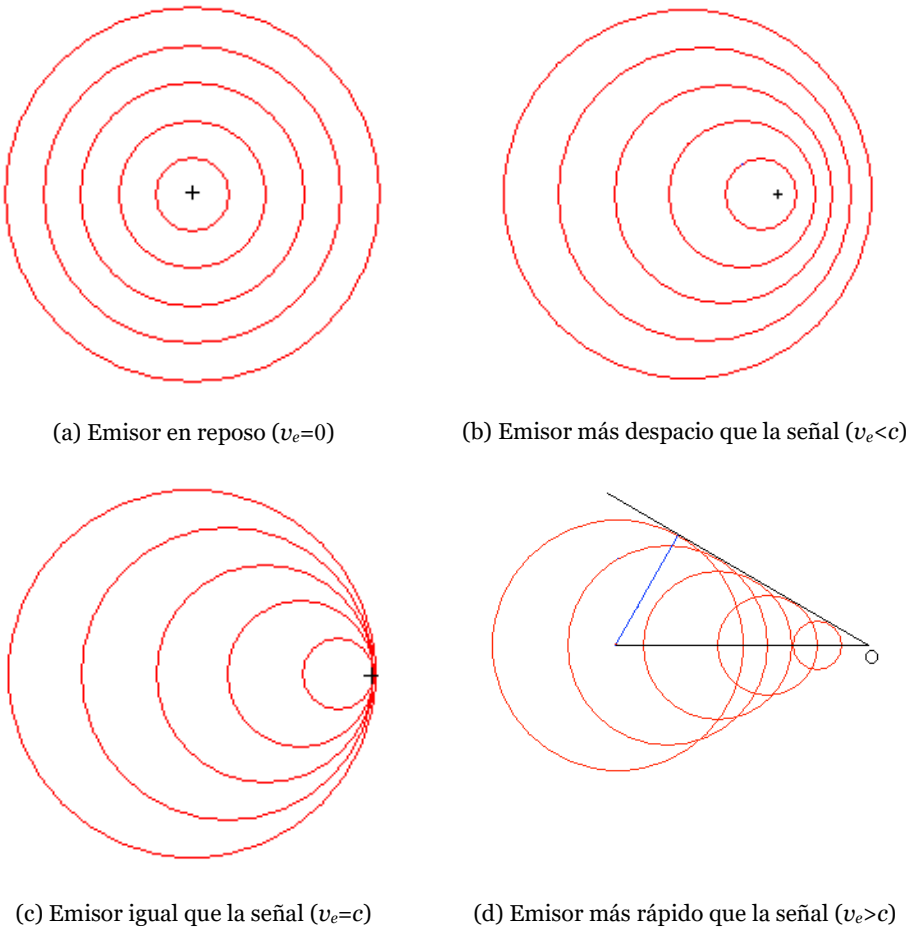
Frente de onda para el movimiento del emisor

El efecto Doppler se produce siempre que hay un movimiento relativo entre emisor y receptor. En esta sección vamos a centrarnos en estudiar cómo el receptor observa el frente de ondas solo en el caso en el que el receptor está en reposo (v_r=0) y es el emisor el que se mueve.

La figura 11 (a) muestra que cuando el emisor está en reposo (v_e=0) los frentes de onda observados por el receptor son concéntricos. En la figura 11 (b) el emisor empieza a moverse pero la velocidad del emisor v_e es menor a la de la onda c.

En este caso si el emisor se mueve al receptor situado a la derecha, la longitud de onda percibida por el receptor es menor y viceversa. La figura 11 (c) muestra que cuando la velocidad del emisor es igual a la velocidad de propagación de la señal (v_e=c) la longitud de onda medida por el receptor situado a la derecha es cero.

Finalmente, si la velocidad del emisor es mayor a la velocidad de propagación de la señal (v_e>c) se produce una onda cónica llamada **onda de choque**, como la que se muestra en la figura 11 (d), en la cual el receptor percibe una longitud de onda negativa. Esta onda es la que se produce cuando un avión supersónico se acerca a nosotros.



(a) Emisor en reposo (v_e=0) (b) Emisor más despacio que la señal (v_e<c)

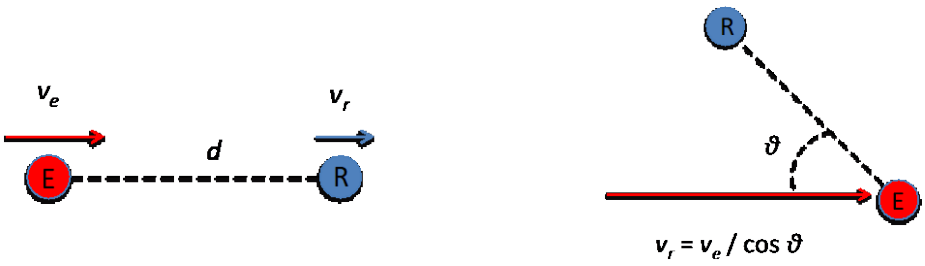
(c) Emisor igual que la señal (v_e=c) (d) Emisor más rápido que la señal (v_e>c)

Figura 11: frente de onda para el movimiento del emisor
Fuente: http://www.sc.ehu.es/sbweb/fisica/ondas/doppler/doppler.html

Fórmula del efecto Doppler

En esta sección vamos a deducir la fórmula del efecto Doppler. Por simplicidad vamos a asumir que emisor y receptor están en una línea recta, tal como muestra la figura 12 (a).

Si el receptor se encuentra en reposo y formando un ángulo θ con el emisor, como muestra la figura 12 (b) la velocidad del emisor que estimemos v_e deberemos dividirla entre cos θ para obtener la velocidad real del emisor v_r.



(a) Modelo lineal (b) Modelo general

Figura 12: modelo lineal y general con receptor en reposo

En la figura 12 (a) el emisor *E* y el receptor *R* se desplazan en la misma dirección y están separados por una distancia *d*, que no afecta al fenómeno en cuestión. Al estar en una línea recta, si el desplazamiento fuera en direcciones opuestas se mantendría este mismo modelo aplicando el correspondiente cambio de signo a sus correspondientes variables de velocidad v_e y v_r.

En la figura 13 (a) el emisor emite una señal en el primer frente de onda (que corresponde con un pico de la onda armónica) en el instante t=0 que es recibida por el receptor en el instante t_{r1}. La señal emitida tiene periodo T_e, con lo que el segundo frente de onda (que corresponde con el segundo pico) se emite en el instante T_e y se recibe en el instante t_{r2}. Esto quiere decir que la señal recibida tiene un periodo T_r, en general distinto al periodo de emisión T_e.

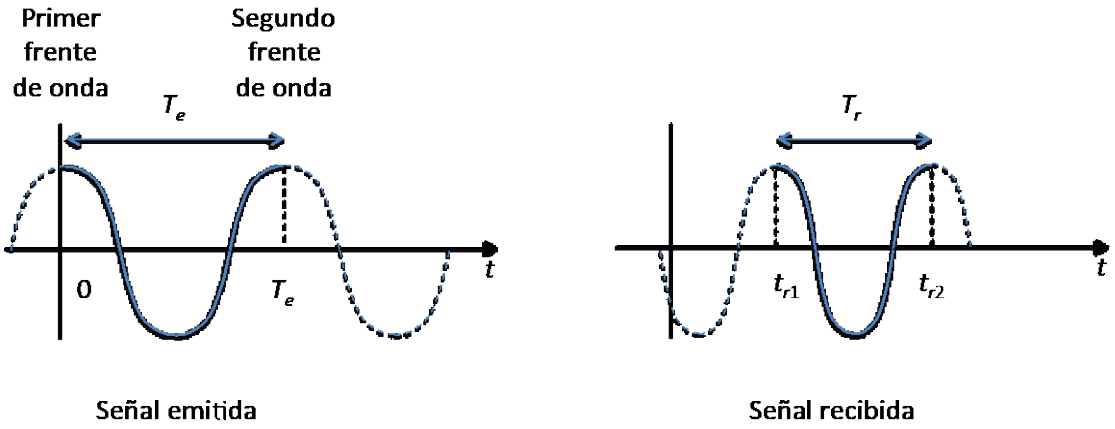
La figura 13 (b) muestra la posición de emisor y receptor cuando se emite y recibe el primer frente de onda. La señal se emite en t=0 y lleva al receptor en el tiempo $t=t_{r1}$ desplazándose el camino marcado en trazo grueso, por lo que podemos definir la ecuación:

$$ct_{r1} = d + v_r t_{r1} \qquad (1)$$

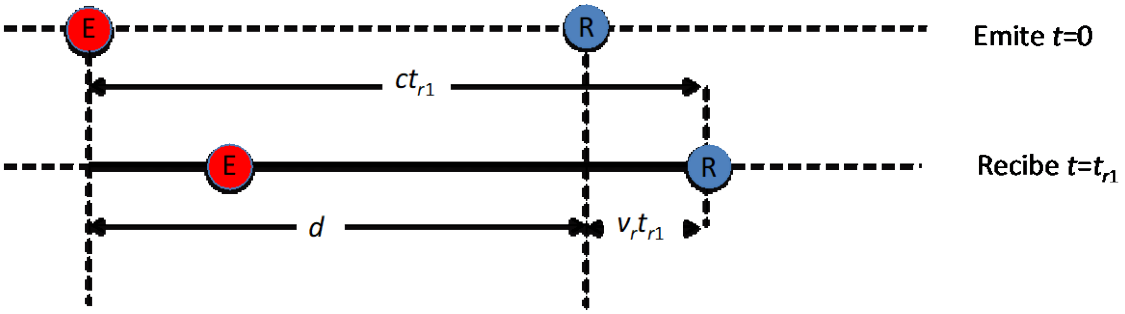
La figura 13 (c) muestra la posición de emisor y receptor cuando se emite y recibe el segundo frente de onda. La señal se emite en el instante T_e y se recibe en el instante t_{r2}.

En el momento en que se emite el segundo frente de onda el emisor se habrá desplazado $v_e T_e$ y el receptor $v_r T_e$. El segundo frente de onda recorre el camino marcado en trazo grueso desde que se emite hasta que se recibe, por tanto podemos definir la ecuación:

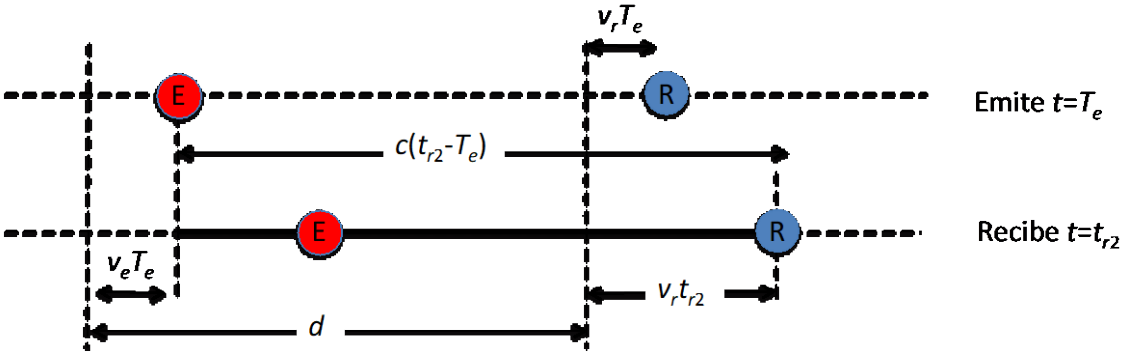
$$c(t_{r2} - T_e) = d - v_e T_e + v_r t_{r2} \qquad (2)$$



(a) Diferencia entre periodos



(b) Posiciones al emitir y recibir el primer frente de onda



(c) Posiciones al emitir y recibir el segundo frente de onda

Figura 13: cálculo de la diferencia entre periodos

Dado que nos interesa conocer el periodo de la segunda señal T_r=t_{r2}-t_{r1}, despejamos t_{r1} y t_{r2} de (1) y (2):

$$\begin{cases} t_{r1} = \dfrac{d}{c - v_r} \\ t_{r2} = \dfrac{cT_e + d - v_e T_e}{c - v_r} \end{cases}$$

Haciendo la diferencia tenemos la relación entre el periodo de la señal T el camino marcado en trazo grueso y el periodo percibido en el receptor T_r:

$$T_r = \frac{c - v_e}{c - v_r} T_e$$

O bien, teniendo en cuenta que la frecuencia es el inverso del periodo, obtenemos la **fórmula de Doppler** que nos relaciona la frecuencia de la señal f_e con la frecuencia percibida en el receptor f_r:

$$f_r = \frac{c - v_r}{c - v_e} T_e \qquad (3)$$

Ejemplo 2: cálculo de frecuencia percibida.

Los camiones *A*, *B* se desplazan en direcciones opuestas por una carretera, tal como muestra la figura 14. Sus respectivas velocidades son V_A=120 km/h, V_B=95 km/h. El camión *A* toca la bocina que emite un tono con una frecuencia de 500Hz. Asumiendo una propagación del sonido de 1224 km/h, ¿con qué frecuencia la recibirá el camión *B*?

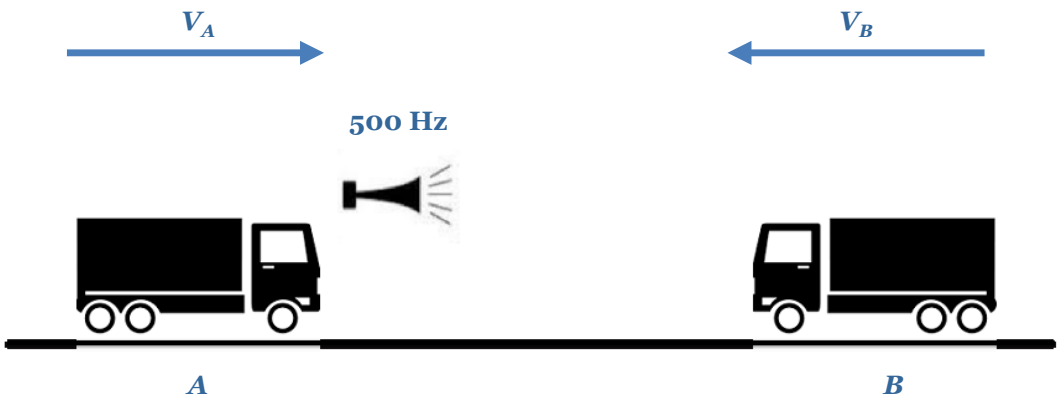


Figura 14: ejemplo efecto Doppler

Dado que las direcciones son opuestas las velocidades de emisión y recepción serán v_e=+120 km/h y v_r=-95 km/h, respectivamente. Aplicando la fórmula de Doppler (3) tenemos:

$$f_e = \frac{c - v_r}{c - v_e} f_e = \frac{1224 + 95}{1224 - 120} 500 \text{ Hz} = 597 \text{ Hz}$$

Lo + recomendado

Lecciones magistrales

Ecuaciones diferenciales lineales con coeficientes constantes

En esta lección magistral veremos cómo resolver un tipo de ecuación diferencial lineal muy habitual: cuando los coeficientes de la ecuación diferencial son constantes.

Accede a la lección magistral a través del aula virtual.

No dejes de leer…

¿Cómo explicar la transformada de Fourier en una sola frase?

En este artículo se incluye un gráfico de colores muy intuitivo para entender la DFT.

R-bloggers
R news and tutorials contributed by (580) R bloggers

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
https://www.r-bloggers.com/the-fourier-transform-explained-in-one-sentence/

Métodos de análisis acústico del habla

Este documento describe cómo se usan los espectrogramas para el análisis de la voz.

Métodos de análisis acústico del habla

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://liceu.uab.es/~joaquim/phonetics/fon_anal_acus/met_anal_acust.html

No dejes de ver...

Procesamiento de sonido en MatLab

Este tutorial da múltiples ejemplos de procesamiento de señal de audio con MatLab.

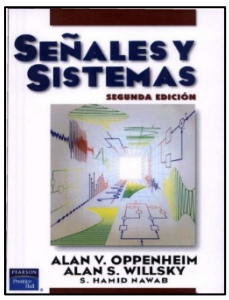
Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
https://www.youtube.com/watch?v=QApgWO4ytDQ

+ Información

A fondo

Señales y sistemas

Oppenheim, A. V., Willsky, A. S. y Hamid, S. (1998). *Señales y sistemas*. México: Pearson.



En las páginas 514-555 se describe el muestreo y los problemas de aliasing.

Accede al libro desde el aula virtual o a través de la siguiente dirección web:
https://books.google.es/books?id=g2750K3PxRYC&pg=PA514

Bibliografía

Benesty, J., Moham, M. y Huang, Y. (2008). *Springer Handbook of speech processing*. EE.UU.: Springer.

Chen, V. C. (2011). *The micro-doppler effect in radar*. London: Artech House.

Test

1. ¿Qué magnitud es más habitual utilizar en procesamiento de audio?

A. f_s.

B. w_s.

C. Ambas por igual.

D. Ninguna es habitual.

2. Para muestrear una señal a 3500Hz necesitamos hacerlo al menos a:

A. 1750 muestras/seg.

B. 3500 muestras/seg.

C. 7000 muestras/seg.

D. 14000 muestras/seg.

3. Si muestreamos a 5000 muestras/seg podemos representar señales de:

A. Hasta 2500 Hz.

B. Hasta 5000 Hz.

C. Hasta 10000 Hz.

D. Ninguna de las anteriores.

4. El pitch corresponde con:

A. La frecuencia más baja de la señal.

B. La frecuencia media de la señal.

C. La frecuencia más alta de la señal.

D. La frecuencia característica de la señal.

5. El muestreo de una señal analógica $x_p(nt_s)$ crea una señal digital x[n] con frecuencias equiespaciadas en:

A. F_m=[-0.5,0.5].

B. F_m=[-1.1].

C. F_m=[-π,π].

D. F_m=[-2π, 2π].

6. Una señal analógica muestreada en frecuencia $X_p(f)$ es una superposición de infinitas copias de la FT de la señal original separadas por:

A. π.

B. 2π.

C. t_s.

D. f_s.

7. ¿Cómo podemos aumentar el goteo espectral?

A. Aumentando la resolución frecuencial.

B. Aumentando el número de coeficientes de la DFT.

C. Aplicando *padding*.

D. Ninguna de las anteriores es correcta.

8. ¿Qué no muestra es espectrograma?

A. La evolución temporal de la señal.

B. La evolución de las ventanas.

C. El tamaño de las ventanas.

D. Muestra todos los anteriores.

9. Los frentes de onda están separados por:

A. El periodo *T*.

B. La longitud de onda λ.

C. Las dos anteriores.

D. Ninguna de las anteriores.

10. ¿Cuándo es máximo el efecto Doppler?

A. Cuando emisor y receptor están alineados.

B. Cuando emisor y receptor forman 45^o.

C. Cuando emisor y receptor forman 90^o.

D. Ninguna de las anteriores.

