Práctica 2
Análisis de Fourier y Síntesis de música

Contenidos de la práctica

1. Análisis de Fourier
    1. Respuesta en frecuencia
    2. Transformada de Fourier en Tiempo Discreto (DTFT)
2. Síntesis de música
    1. Conceptos previos
        1. Conversión Digital-Analógica
        2. Teoría de muestreo
        3. Teclado de un piano
    2. Síntesis de notas musicales
        1. Para Elisa
        2. Ajustes musicales

1.1 Respuesta en frecuencia

- **Autofunción** de un sistema LSI:
  - Secuencias que al introducirse en un sistema solo producen a la salida un cambio complejo en su amplitud

$$y(n) = \lambda x(n)$$

donde λ se denomina **autovalor**.

  - Las exponenciales complejas puras son autofunciones para ω constante. El autovalor de dicha función se conoce como respuesta en frecuencia.

$$H\left(e^{j\omega}\right) = \sum_{k=-\infty}^{\infty} h(k)e^{-j\omega k}$$

1.1 Respuesta en frecuencia

- Se puede expresar en términos de amplitud y fase:

$$H\left(e^{j\omega}\right) = |H\left(e^{j\omega}\right)| e^{j\varphi(\omega)}$$

- Esta función continua es aplicable a sistemas y caracteriza su efecto sobre la señal de entrada.

- Escribe una función para representar la respuesta en frecuencia de un sistema representado mediante su respuesta a impulso *h(n)*.
  - Para ello, crear una función con la siguiente cabecera:

```
function H = resp_frec(h, precision)
```

donde **h**: Respuesta a impulso del sistema

precision: Paso de frecuencia ($\Delta\omega$)

1.2 Transformada de Fourier en Tiempo Discreto (DTFT)

- La DTFT de una secuencia $x(n)$ se define como:

$$X\left(e^{j\omega}\right) = \sum_{k=-\infty}^{\infty} x(k)e^{-j\omega k}$$

  - Por analogía se puede observar que la respuesta en frecuencia corresponde a la DTFT de la respuesta al impulso unitario.
  - Para que exista la DTFT de una secuencia $x(n)$:

$$\sum_{n=-\infty}^{\infty} |x(n)| = S < \infty$$

1.2 Transformada de Fourier en Tiempo Discreto (DTFT)

- Crear una función en Matlab que calcule la DTFT de una señal de entrada
  - Para dicha función utilizar la siguiente cabecera

```
function X = DTFT(x, precision)
```

Donde:

`x` : Señal de entrada

`precision` : Paso de frecuencia ($\Delta\omega$)

- Representar el módulo y la fase de la DTFT obtenida
  - Utilizar la función **`subplot`** para dichas representaciones.

1.2 Transformada de Fourier en Tiempo Discreto (DTFT)

- Teorema de la convolución:
  - La DTFT de la convolución de dos secuencias es igual al producto de sus DTFTs particulares:

$$y(n) = h(n) * x(n)$$

$$Y(e^{j\omega}) = H(e^{j\omega}) \cdot X(e^{j\omega})$$

- Comprobar la propiedad de convolución (opcional)
  - Utilizar la función desarrollada en la práctica 1 para realizar la convolución de dos señales finitas.
  - Utilizar la función **`DTFT`** para obtener las DTFTs particulares de $h(n)$, $x(n)$ e $y(n)$.
  - Comprobar que se cumple la segunda expresión.

1.2 Transformada de Fourier en Tiempo Discreto (DTFT)

- Propiedad del desplazamiento:
  - Desplazar una secuencia en el tiempo tiene como resultado la multiplicación de su DTFT por una exponencial compleja

$$x(n-n_0) \overset{DTFT}{\Longleftrightarrow} e^{-jn_0\omega} X(e^{j\omega})$$

- Comprobar la propiedad de desplazamiento (opcional)
  - Utilizar la función desarrollada en la práctica 1 para realizar el desplazamiento de una señal. Hacer un desplazamiento de medio periodo.
  - Hacer una representación comparativa entre los módulos y las fases de la DTFT de las señales original y desplazada.

2. Síntesis de música

- En esta práctica sintetizaremos ondas compuestas de sumas de sinusoidales de la forma

$$x(t) = A \cdot \cos(\omega_0 t + \phi)$$

- Utilizaremos combinaciones de la sinusoide básica para sintetizar las siguientes señales:
  - Ondas sinusoidales de una frecuencia concreta reproducidas en un conversor D/A.
  - Sinusoides que crean una versión sintetizada de *Para Elisa*.
  - Una canción cualquiera que deberá ser sintetizada.

2.1 Conceptos previos

- En esta práctica se crearán ondas senoidales con intención de ser reproducidas por un altavoz.

- Para ello, es necesario convertir muestras digitales a la forma de onda de tensión que será reproducida por los altavoces

- La disposición de un teclado será también explorada, de modo que tengamos una fórmula que nos dé la frecuencia de cada tecla.

2.1.1 Conversión Digital-Analógica

- En su forma más sencilla, la conversión D/A viene determinada exclusivamente por el tiempo de muestreo (T_s), que nos determina el espaciado temporal de las muestras almacenadas digitalmente.

- Se debe corresponder con la frecuencia con la que se este usando el hardware D/A.

- En Matlab la salida del sonido se realiza a través del conversor D/A de la tarjeta de sonido mediante la función **`sound`**.

2.1.1 Conversión Digital-Analógica

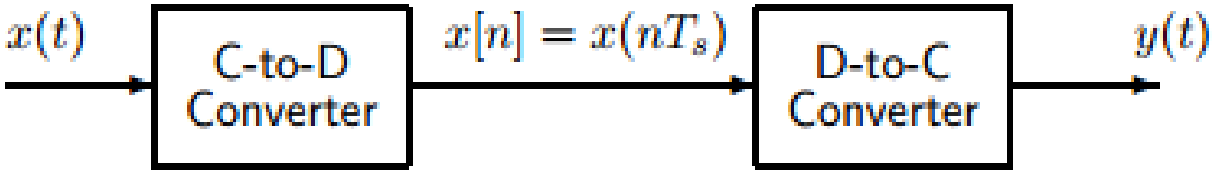
```
>> help sound
 SOUND Play vector as sound.
    SOUND(Y,FS) sends the signal in vector Y (with sample frequency
    FS) out to the speaker on platforms that support sound. Values in
    Y are assumed to be in the range -1.0 <= y <= 1.0. Values outside
    that range are clipped.  Stereo sounds are played, on platforms
    that support it, when Y is an N-by-2 matrix.

    SOUND(Y) plays the sound at the default sample rate of 8192 Hz.

    SOUND(Y,FS,BITS) plays the sound using BITS bits/sample if
    possible.  Most platforms support BITS=8 or 16.

    Example:
      load handel
      sound(y,Fs)
```

2.1.1 Conversión Digital-Analógica

- Algunas tasas de muestreo D/A habituales son:
  - 8.000 Hz
  - 11.025 Hz
  - 44.100 Hz
- Las notas de un piano tienen frecuencias bajas por lo que se podrá considerar una tasa de muestreo menor
- Es de gran utilidad disponer de un afinador con el fin de verificar que la onda que se está reproduciendo se ajusta a la frecuencia y por tanto a la nota que deseamos

2.1.2 Teoría de muestreo

- En la siguiente figura se esquematiza el muestreo y la reconstrucción de una señal a partir de sus muestras



- Un conversor D/C ideal toma las muestras e interpola una curva suave entre ellas para generar la salida

- Si *x(t)* es una suma de senos, entonces *y(t)* será igual a *x(t)* si : $f_s > 2f_{max}$

2.1.2 Ejercicios de muestreo

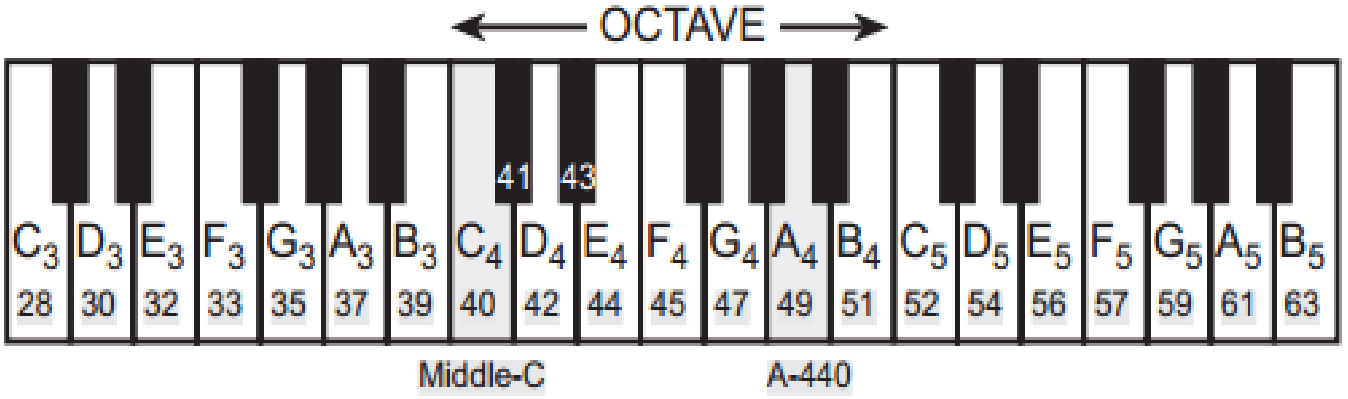
- Calcula un vector $x_1(n)$ de muestras de una señal sinusoidal con:
  - $A = 100$
  - $\omega_0 = 2\pi(1100)$
  - $\phi = 0$
  - $f_s = 8\ kHz$
- Calcula un número de muestras total equivalente a 2 segundos y reproduce la señal reconstruida con estas muestras mediante el conversor D/A de la tarjeta de sonido de tu computador.

2.2.1 Ejercicios de muestreo

- Calcula un vector $x_2(n)$ de muestras de una señal sinusoidal con :
  - $A = 100$
  - $\omega_0 = 2\pi(1650)$
  - $\phi = \pi/3$
  - $f_s = 8\ kHz$
- Repite el ejercicio anterior.
- Reproduce de nuevo el vector obtenido, pero doblando la tasa de muestreo a $f_s = 16\ kHz$ **sin cambiar el vector**. Describe lo que has escuchado y explica por qué ha sucedido.

2.1.3 Teclado de un piano

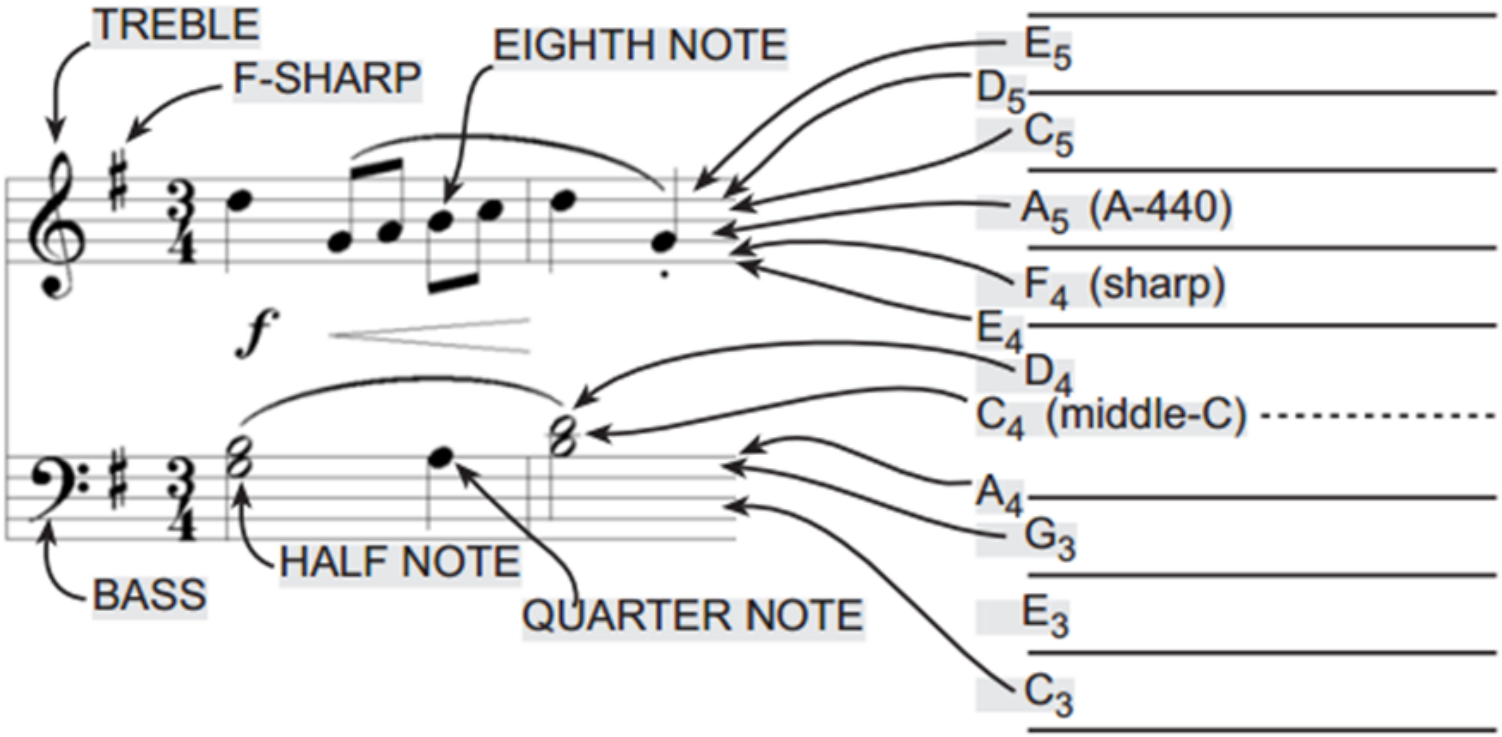
- La nota de referencia es A_4 (LA), normalmente llamada A-440 porque su frecuencia es de 440Hz. Esta es la notación anglosajona para las notas musicales.



- $2^{1/12}$ es la razón frecuencial entre cualesquiera dos teclas consecutivas del piano (semitono musical).

2.1.3 Teclado de un piano

- La notación musical muestra qué notas deben ser reproducidas y en qué momento, y cuál debe ser su duración.



2.1.3 Ejercicios

- Generar una sinusoide de 2 segundos de duración que represente la nota F4 (tecla 45). Elegir unos valores apropiados de T_s y f_s.

  - *NOTAS:*
  - Recuerda que f_s debería ser al menos el doble que la frecuencia de la sinusoide que estás generando.
  - Además, T_s y f_s deben "encajar" para que la nota se reproduzca correctamente en el conversor D/A.

2.1.3 Ejercicios

- Escribe una función llamada `nota.m` que genere una nota musical cualquiera a partir de su número de tecla
  - Se recomienda basar el cálculo de la frecuencia en la nota de referencia A-440.

```
function tono = nota(tecla,dur,fs)
% function tono = nota(tecla,dur,fs)
% NOTA Produce una onda sinusoidal correspondiente a la
% tecla de piano solicitada
%
% tono = onda de salida
% tecla = numero de tecla de piano a reproducir
% dur = duracion (en segundos) de la nota de salida
% fs = tasa de muestreo

% Algoritmo
```

2.1.3 Ejercicios

- Escribe una función llamada `escala.m` que genere una escala musical completa.
  - Una escala comienza en una C y llega hasta la siguiente C, pasando por todas las teclas **blancas**

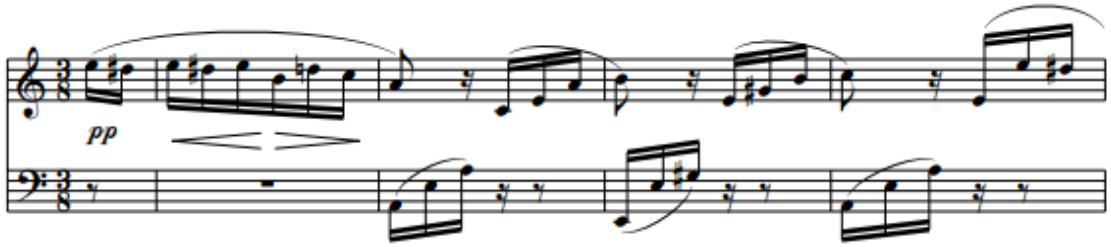
```
function sonido = escala(valor,dur,fs)
% function sonido = escala(valor,dur,fs)
% ESCALA Produce una sucesion de ondas correspondiente a la
% escala solicitada del piano
%
% sonido = onda de salida
% valor = indicativo de la escala solicitada (4 = escala media)
% dur = duration (en segundos) de cada nota de la escala
% fs = tasa de muestreo

% Algoritmo
```

2.2 Síntesis de notas musicales

- En el proceso de sintetizar la música se han de seguir estos pasos:
  - Determinar la frecuencia de muestreo
  - Determinar la duración de cada nota
  - Determinar la frecuencia en Hz para cada nota
  - Sintetizar la forma de onda como una combinación de sinusoides y reproducirla a través del altavoz del computador utilizando la función **`sound`**
  - **Un acorde se puede sintetizar sumando las sinusoides de cada nota**
  - Se pueden crear melodías al unísono mediante la suma de distintos vectores

2.2.1 Para Elisa

- Mapea cada nota a una tecla y a partir de ahí, sintetiza ondas sinusoidales para recrear la siguiente partitura.

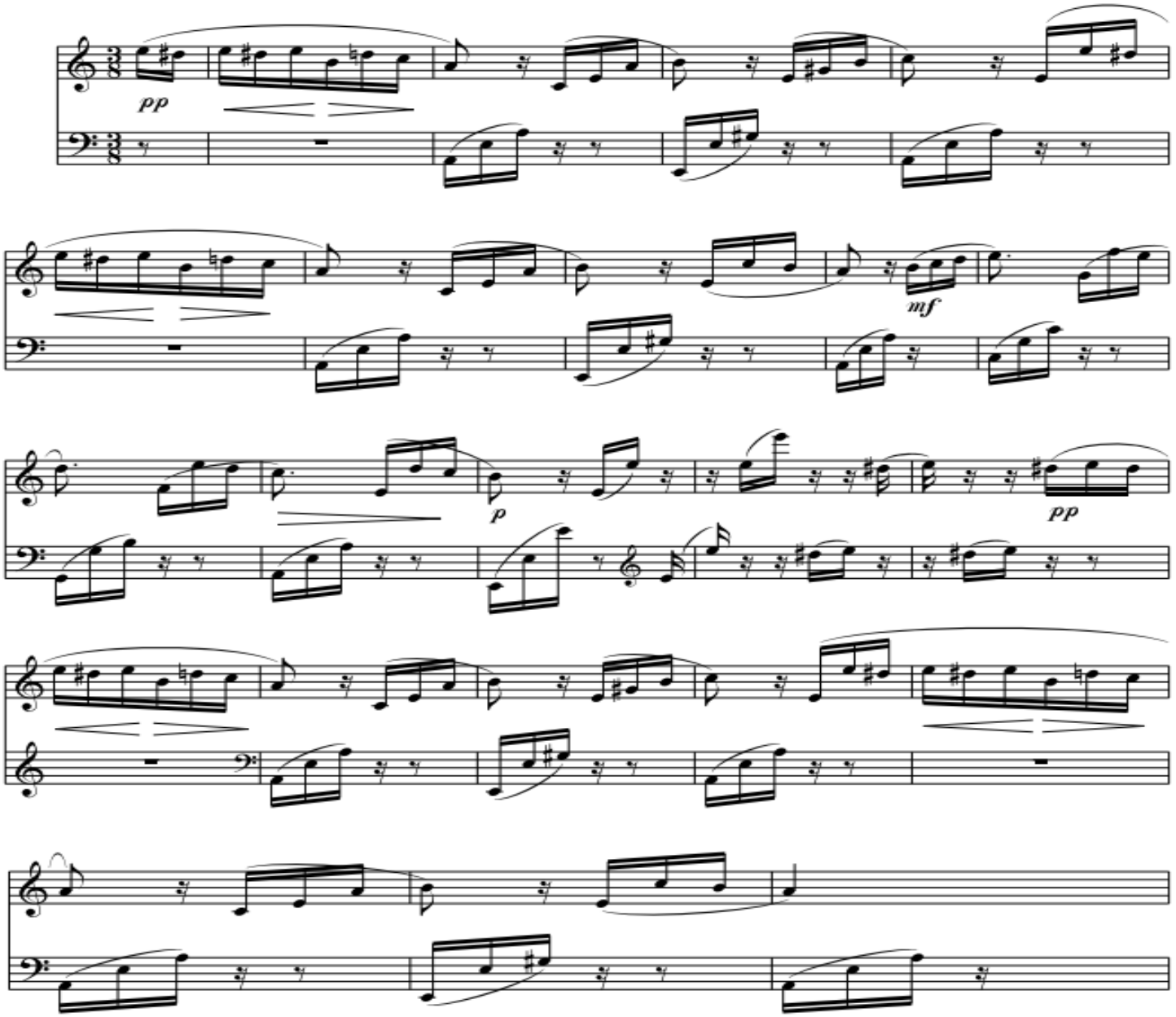


  - Utiliza ondas senoidales muestreadas a:
    - 8 KHz para UNIX
    - 11,025 KHz para Windows o Mac

Opcional

Fur Elise

L. van Beethoven



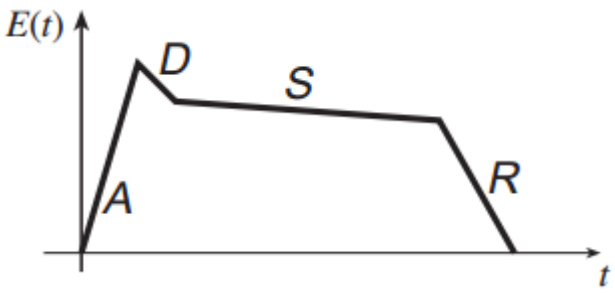
2.2.2 Ajustes musicales

- Como habrás comprobado las melodías suenan muy artificiales al sintetizarlas con ondas senoidales puras

- Es posible aplicar envolventes que aporten más calidad al sonido, por ejemplo:

$$x(t) = E(t)\cos(2\pi f_0 + \phi)$$

- A: Ataque
- D: Demora
- S: Mantenimiento
- R: Salida



2.2.2 Ajustes musicales

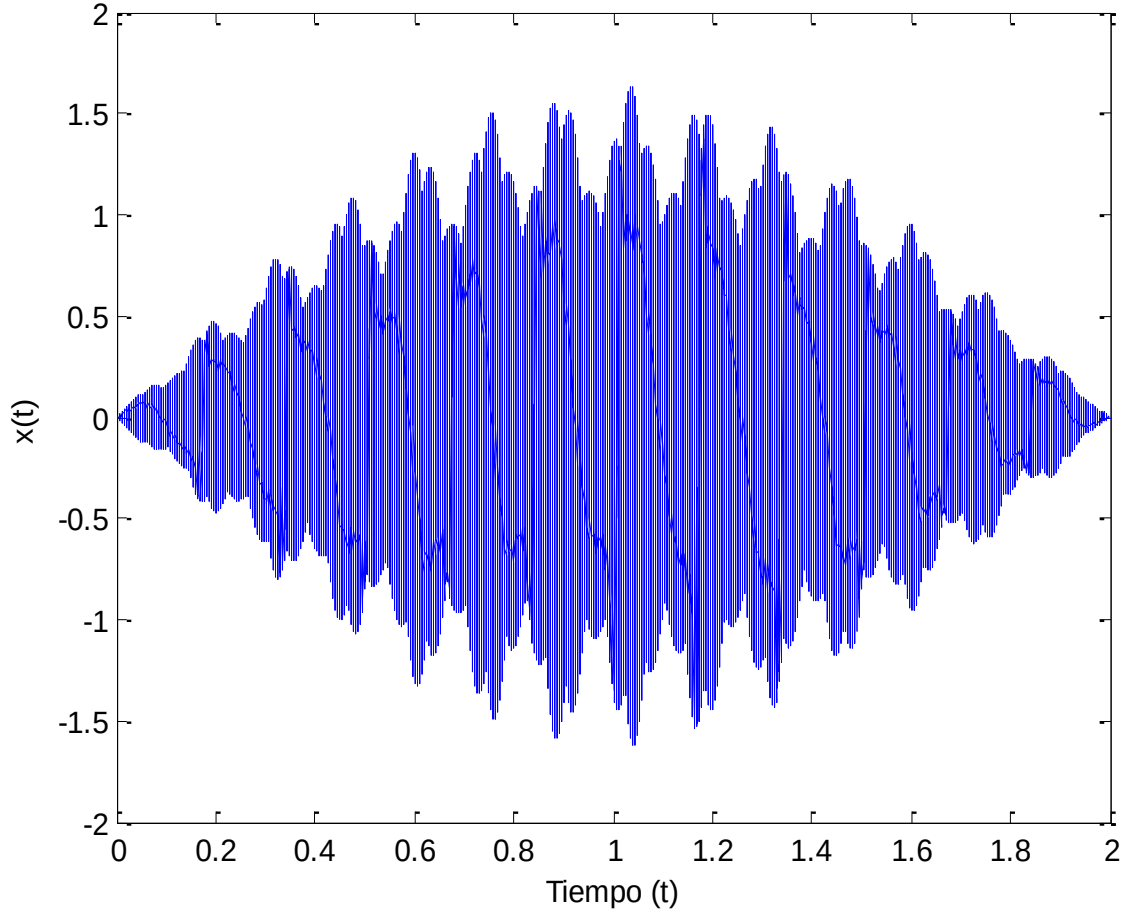
- También se puede mejorar el sonido mediante la introducción de armónicos.

- Se pueden obtener timbres característicos de distintos instrumentos introduciendo armónicos de frecuencia y amplitud adecuada.

- Se puede modificar la función `note.m` para aceptar parámetros adicionales de entrada o para añadir una envolvente o armónicos.

3.2 Ajustes musicales

- Nota A4 de una violín

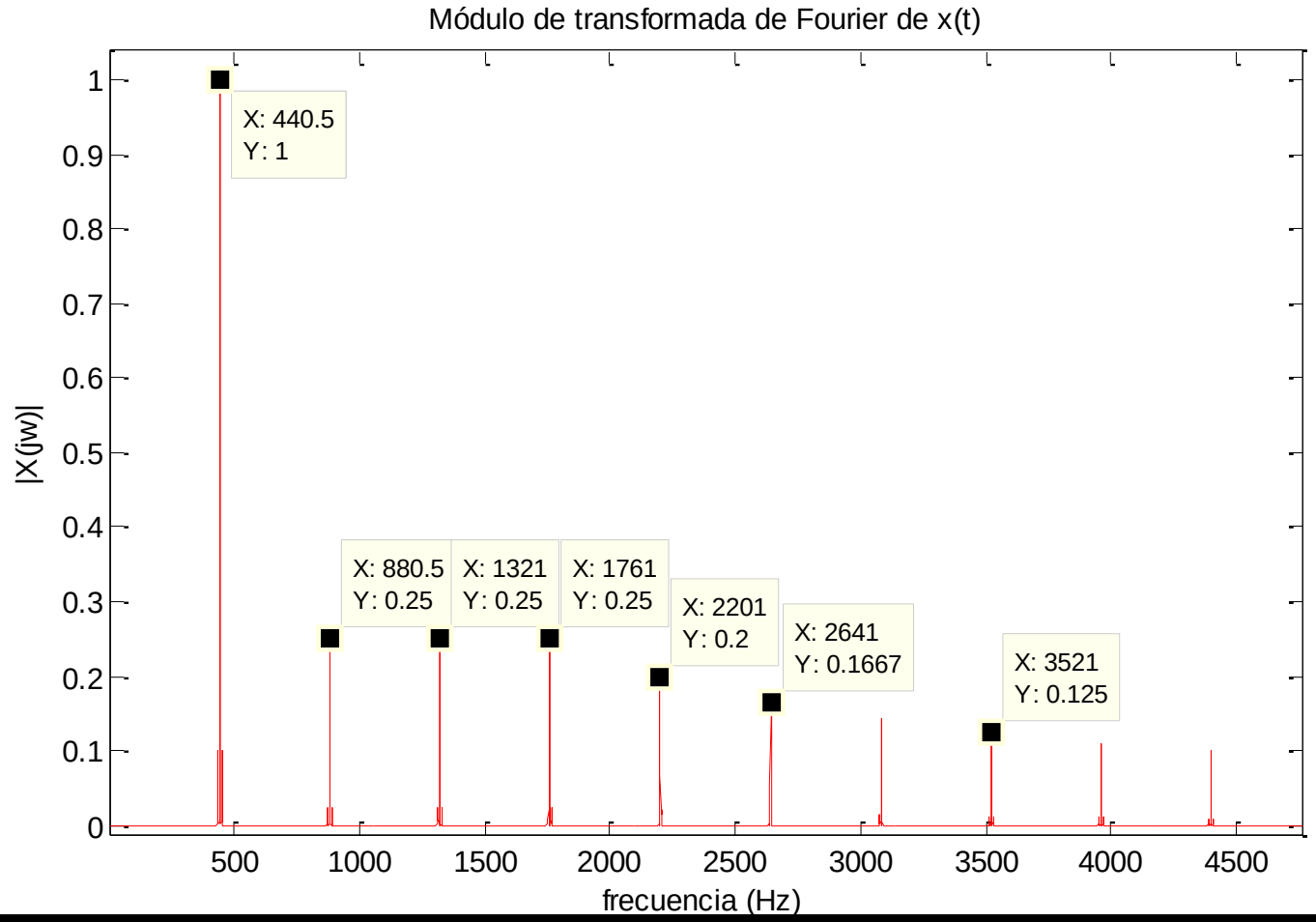



```
nota =
(sin(2*pi*t*vibrato)+5)/6
 .*(1.8*sin(pi*t/dur))
 .*(0.6*
 (sin(2*pi*f*t)
  +sin(2*pi*f*t*2)/4
  +sin(2*pi*f*t*3)/4
  +sin(2*pi*f*t*4)/4
  +sin(2*pi*f*t*5)/5
  +sin(2*pi*f*t*6)/6
  +sin(2*pi*f*t*7)/7
  +sin(2*pi*f*t*8)/8
  +sin(2*pi*f*t*9)/9
  +sin(2*pi*f*t*10)/10
 )
);
```

Vibrato es un n° entero
En el ejemplo vibrato=7

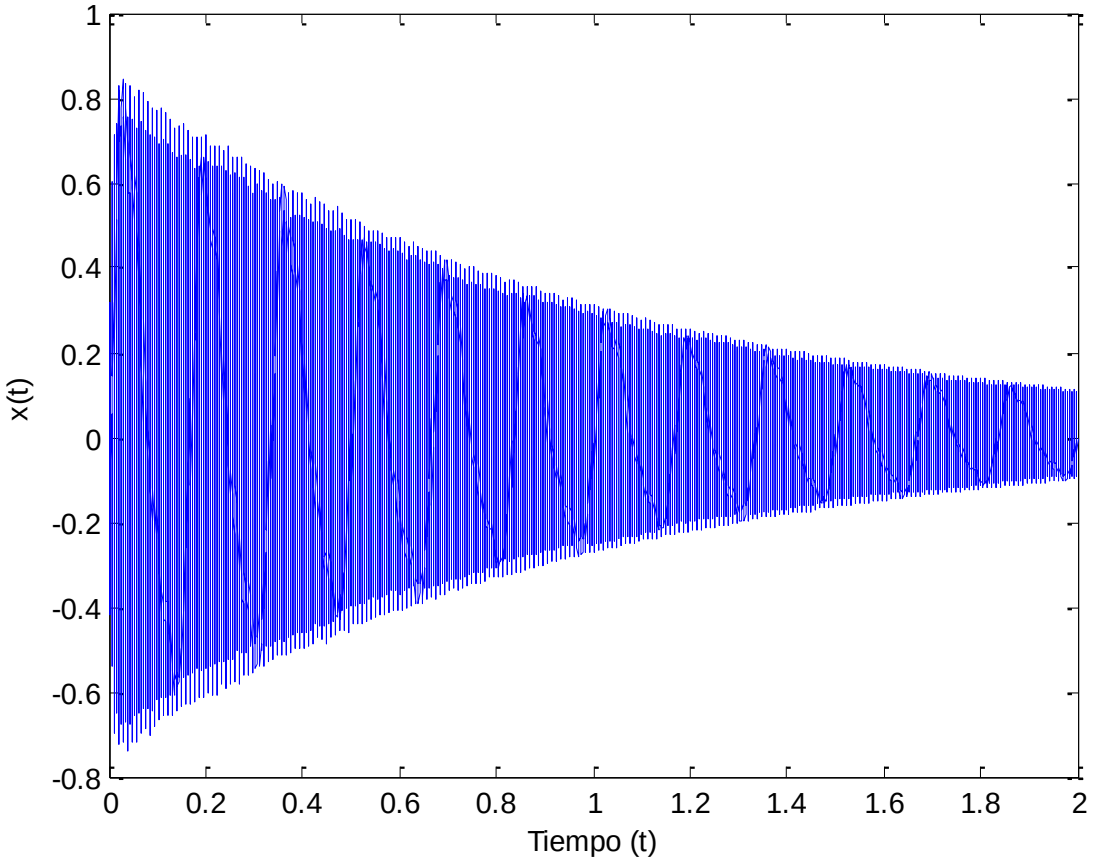
2.2.2 Ajustes musicales

- Nota A4 de una violín



2.2.2 Ajustes musicales

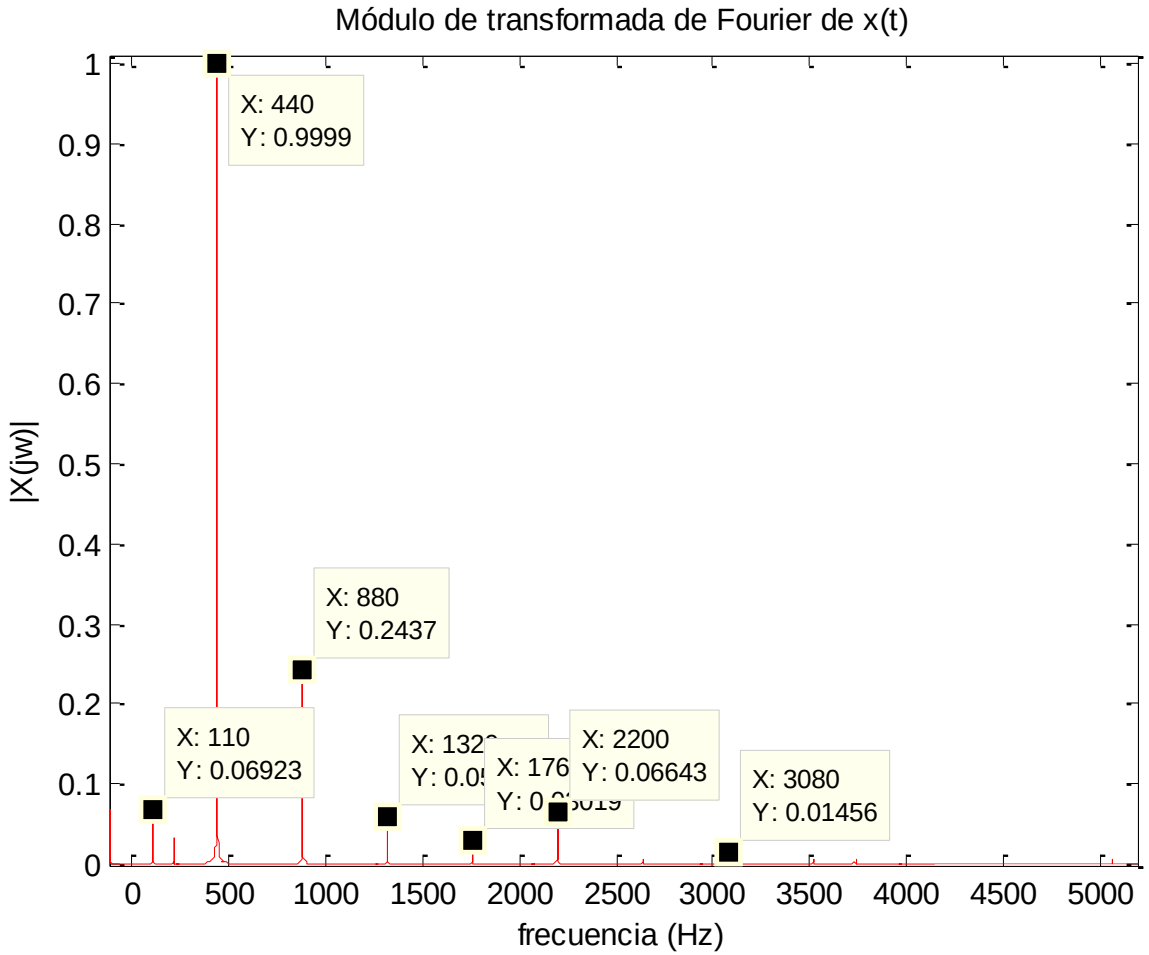
- Nota A4 de un clavicordio




```
Envolvente =
((sin(2*pi*t*5)+2)/3)
  .*(1-exp(-200*t))
  .*(exp(- 4*t/(dur)
)
```

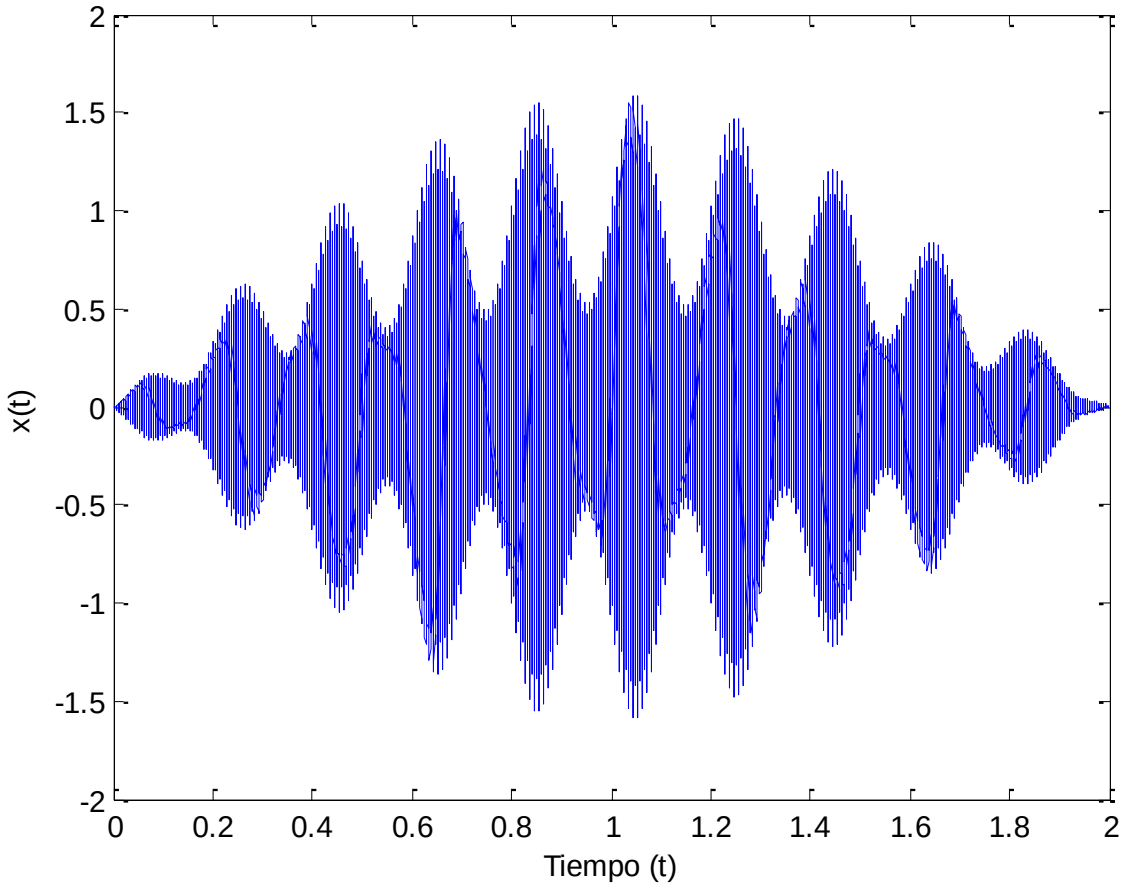
2.2.2 Ajustes musicales

- Nota A4 de un clavicordio



2.2.2 Ajustes musicales

- Nota A4 de una flauta

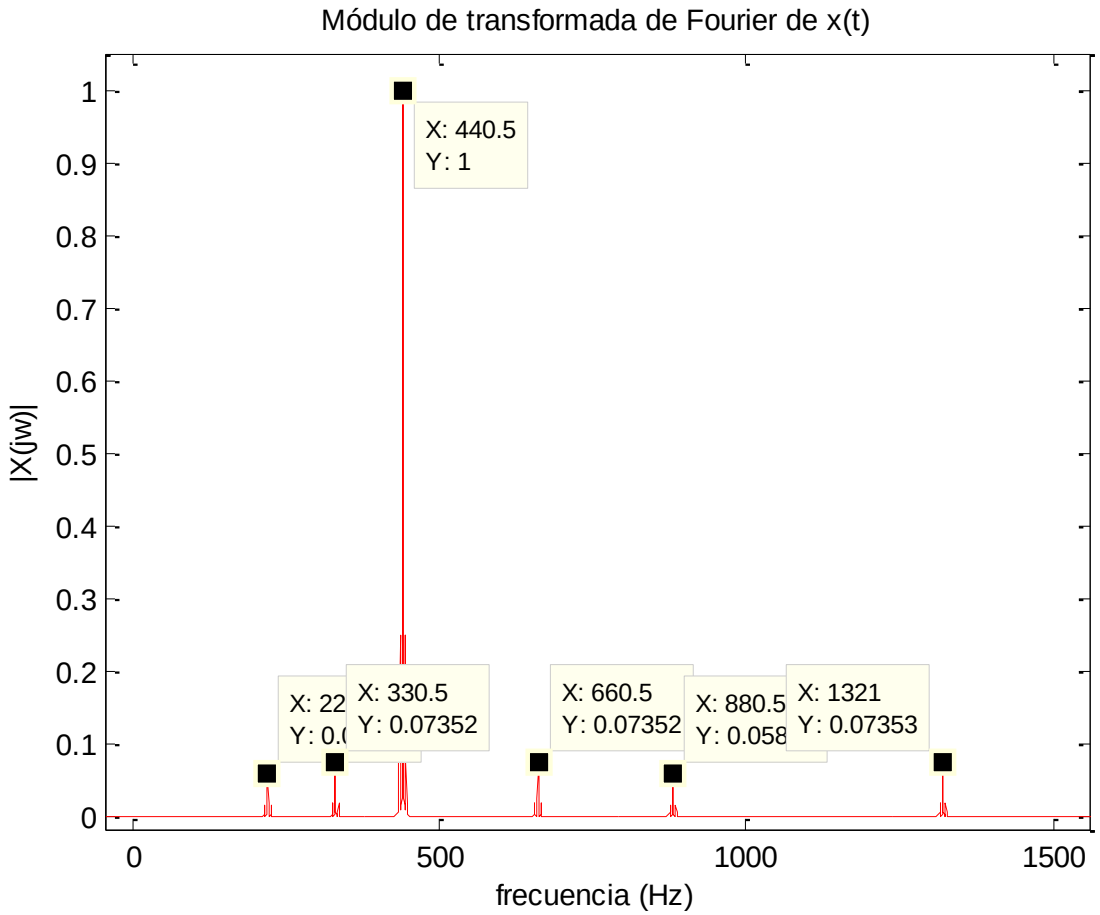



```
nota =
sqrt(2)*sin(pi*t/dur)
.*(sin(2*pi*t*vib)+2)/3
.*(0.96*sin(2*pi*f*t)
+0.075*(sin(2*pi*f*0.75*t)
        +sin(2*pi*f*1.5*t)
        +sin(2*pi*f*3*t))
+0.06*(sin(2*pi*f*2*t)
       +sin(2*pi*f*2*t/2)
       +sin(2*pi*f*2*t/4)
       )
   )
```

vib es un n° entero
En el ejemplo vib=7

2.2.2 Ajustes musicales

- Nota A4 de una flauta



2.2.2 Ajustes musicales

- Es posible obtener el espectro de cualquier instrumento mediante el uso de la FFT.
- Para realizarlo con Matlab es necesario disponer del tono de audio en un archivo *.wav* (este archivo se puede leer con el comando *audioread*).

- Realizar el ajuste necesario para imitar el sonido de un instrumento musical.
- Sintetizar cualquier tema musical a elegir por el alumno (opcional).