Señales y Sistemas
Laboratorio 1 (4h)
Exponenciales complejas: síntesis de notas musicales

Profesor responsable: Luis Vielva

Curso 2002/2003

Materiales: (a) El alumno debe traer unos auriculares similares a los de un reproductor portatil. (b) Todos los ordenadores están equipados con Matlab y tarjeta reproductora de sonido.

Objetivos: (a) Explorar las exponenciales complejas genéricas, un elemento básico de la disciplina de Señales y Sistemas, y su particularización a las señales sinusoidales. (b) Familiarizarse con la generación y reproducción de señales sonoras en el entorno Laboratorio–Windows–Matlab. (c) Experimentar el fenómeno del aliasing, tanto visualmente como mediante la audición de señales sinusoidales muestreadas. (d) Establecer la conexión entre notas musicales, frecuencias y sinusoides. (e) Tratar de añadir otras características a la síntesis que mejoren la calidad subjetiva del sonido.

Introducción

Este laboratorio consta de dos partes. En la primera se exploran las exponenciales complejas genéricas, un elemento básico de la disciplina de Señales y Sistemas, y su particularización a las señales sinusoidales. En la segunda parte se realiza un proyecto de síntesis de música con señales sinusoidales.

En la primera parte se generan señales exponenciales complejas; abordando las exponenciales reales, las sinusoidales y las sinusoidales amortiguadas. Las señales complejas se representan como parte real e imaginaria; utilizando representaciones cartesianas y fasoriales. Se generan secuencias discretas obtenidas muestreando sinusoides continuas, observando dos fenómenos importantes: que no todas las sinusoides discretas son periódicas y el proceso de aliasing. Este último se analiza tanto en la representación gráfica (superponiendo muestras de señales distintas) como en la audición de los tonos correspondientes.

En la segunda parte, se utilizan combinaciones lineales de señales sinusoidales para construir: (a) Tonos monocromáticos de frecuencia especificada. (b) Una versión preliminar de las primeras notas de la Quinta Sinfonía de Beethoven. (c) Una versión mejorada de la misma.

También se explora un teclado de piano, asignando la frecuencia correspondiente a cada tecla. Se introduce la relación entre la escala y frecuencias musicales, explicando la forma de calcular las frecuencias correspondientes a las teclas de un piano con afinación equitemperada. Se explica la notación del pentagrama, resaltando sus propiedades temporales y frecuenciales, y relacionándolo con el espectrograma que implementa Matlab. El alumno debe generar una función que produzca una señal sinusoidal que se corresponda a la altura de una nota, para ello se le proporciona el ordinal de la tecla del piano y la duración. A continuación el alumno debe generar una melodía que le resulte conocida utilizando la función anterior con los parámetros adecuados de altura y duración, prestando atención a los silencios entre notas. Se sugiere al alumno la posibilidad de implementar acordes y líneas melódicas distintas. Se relaciona el timbre de los instrumentos musicales con la estructura armónica de sus notas.

Almacenamiento de los ficheros

Todos los ejemplos y ejercicios que se proponen en este laboratorio deben crearse en ficheros y almacenarse en el disco duro del alumno. Para ello, se debe crear una carpeta con el nombre `lab1`.

Parte I: Exponenciales complejas

Dibujo de funciones continuas mediante `ezplot`: la función `ezplot` admite como parámetros una cadena de caracteres que representa una función matemática y un intervalo en el que hacer variar la variable independiente. Dibuja exponenciales reales continuas $x(t) = e^{-\alpha t}$ mediante el siguiente código que almacenarás en el fichero `ejemplo1.m`:

```
close all;
ezplot('exp(-t)', [-2 2]');
hold on;
ezplot('exp(0.5*t)', [-2 2]');
```

Una vez escrito y almacenado el fichero, puedes ejecutarlo desde la línea de comandos de Matlab mediante la orden:

```
>> ejemplo1;
```

Dibujo de funciones continuas mediante `plot`: la función plot admite como parámetros vectores, que pueden interpretarse como las coordenadas correspondientes a muestras de una función. Por tanto, no se pueden dibujar realmente funciones continuas mediante `plot`, aunque si se disponen de suficientes muestras, la unión de puntos mediante rectas que realiza `plot` se parece bastante a la función continua. Prueba estos aspectos mediante el siguiente ejemplo que almacenarás en el fichero `ejemplo2.m`:

```
close all;
for nFigura = 1:2
    subplot(2, 1, nFigura);
    ezplot('sin(2*pi*t)', [0 1]);
    hold on;
    t = linspace(0, 1, 10^nFigura); x = sin(2*pi*t); plot(t, x, 'r');
end
```

Ejercicio 1: Crea un fichero mediante el editor de Matlab que se llame `amor.m` y que construya una exponencial amortiguada $x(t) = Ce^{at}$, donde $C = |C|e^{j\theta}$ y $a = r + j\Omega_0$. Según esta definición de parámetros, la señal puede expresarse como $x(t) = |C|e^{rt}e^{j(\Omega_0 t+\theta)}$. La función admite tres parámetros de entrada:

1. Un vector `t` que indica los puntos en los que se evalúa la exponencial compleja.
2. Un número complejo `C` que es la amplitud compleja de la exponencial amortiguada.
3. Un número complejo `a` que indica el amortiguamiento y la frecuencia.

La función devuelve como argumento

1. Un vector complejo `x` que indica el valor de la exponencial compleja en los puntos del vector de entrada `t`

El fichero debe tener la siguiente estructura:

```
function x = amor(t, C, a)
...
...
```

donde los puntos suspensivos indican las líneas necesarias para implementar la función.

Comprueba el funcionamiento de tu función mediante el siguiente ejemplo que debes guardar en el fichero `ejemplo3.m`:

```
close all;
t = linspace(0, 20, 1000);
x = amor(t, 1, 0.1 + j*pi/2);
figure(1); plot(t, real(x), 'r', t, imag(x), 'b', t, abs(x), 'g');
figure(2); plot(real(x), imag(x)); axis equal;
figure(3); comet(real(x), imag(x));
```

Sinusoidales discretas: vamos a comprobar que no todas las exponenciales discretas son periódicas.

Ejercicio 2: Construye una función que evalúe la señal $x[n] = e^{j\omega_0 n}$ y almacenalá en el fichero `sinper.m`. La sintáxis de la función debe ser la siguiente:

```
function [x, EsPeriodica, N, m, wf] = sinper(n, w0)
...
```

La función tiene dos argumentos de entrada:

1. `n` es un vector con los índices en los que se debe evaluar la sinusoidal.
2. `w0` es la frecuencia angular discreta

y devuelve cinco valores:

1. El vector `x` con las muestras de la sinusoide.
2. Un valor booleano que vale `1` si la señal es periódica y `0` si no lo es.
3. El periodo de la señal, `N` (que sólo tiene sentido si `EsPeriodica` vale `1`).
4. El número de periodos de la señal continua necesarios para que se repita la señal discreta `m` (que sólo tiene sentido si `EsPeriodica` vale `1`).
5. La frecuencia fundamental de la señal discreta periódica (que sólo tiene sentido si `EsPeriodica` vale `1`).

Recuerda que para que una señal discreta sea periódica, debe verificar que

$$\frac{\omega_0}{2\pi} = \frac{m}{N} \in \mathbb{Q};$$

es decir, que $\omega_0/(2\pi)$ sea un número racional. Para realizar esta comprobación puedes utilizar la función de Matlab `[n, d] = rat(x)`, que devuelve como aproximación racional al número `x` el cociente `n/d`. Si este cociente es igual al número original, éste será racional. Es decir, podrías utilizar el siguiente código

```
[m, N] = rat(w0/2/pi);
EsPeriodica = (m/N == w0/2/pi);
```

Comprueba tu función con el siguiente ejemplo que debes almacenar en el fichero `ejemplo4.m`:

```
close all;
w0 = [2 0.2*pi 6*pi/35];
nf = [10 40 70];
for k=1:length(w0)
    n = 0:nf(k);
    [x, EsPeriodica, N, m, wf] = exper(n, w0(k));
    k, EsPeriodica, N, wf
    t = linspace(0, nf(k), 1000);
    figure(k); plot(t, real(amor(t, 1, j*w0(k))), 'r'); hold on;
    stem(n, real(x));
end
```

Aliasing de señales sinusoidales: ya sabemos que una señal discreta puede interpretarse como la obtenida tomando muestras de una señal continua. Realiza el siguiente ejemplo que debes almacenar en el fichero `ejemplo5.m`:

```
close all;

tf = 3;
W0 = 2*pi;
t = linspace(0, tf, 1001);
plot(t, sin(W0 * t), 'r'); hold on;

T = 0.2;
n = 0:tf/T;
x = sin(W0*n*T);
stem(n*T, x);

W = W0 - 2*pi/T;
plot(t, sin(W * t), 'g');
```

Observa que las dos señales sinusoidales continuas comparten el mismo conjunto de muestras (secuencia discreta) para el periodo de muestreo escogido. Por lo tanto, cuando muestreamos así, ambas señales se confunden, una es idéntica a la otra, son *alias* la una de la otra. Este es el fenómeno de aliasing aplicado a las señales sinusoidales. El resultado clave es que, para que una señal continua esté bien muestreada, debemos tomar al menos dos muestras por periodo de la señal. Si la señal está compuesta por varias señales sinusoidales simultáneamente, hay que utilizar una frecuencia de muestreo $f_s > 2f_{\max}$, donde $f_{\max}$ es la mayor de las frecuencias presentes en la señal.

Parte II: Síntesis de notas musicales

Sintetizaremos formas de onda compuestas por señales sinusoidales de la forma

$$x(t) = A\cos(\Omega_0 t + \phi), \tag{1}$$

las muestrearemos, y las reconstruiremos para poderlas oir. Utilizaremos combinaciones lineales de señales del estilo de (1) para construir:

1. Tonos monocromáticos de frecuencia especificada.
2. Una versión preliminar de las primeras notas de la Quinta Sinfonía de Beethoven.
3. Una versión mejorada de la misma.
4. Alguna otra composición musical elegida individualmente.

Conversión D-A: En matlab, se utiliza la función `sound(x, fs)`, que admite especificar distintas tasas de muestreo. En esta práctica utilizaremos 8000 u 11025 muestras por segundo. La función `soundsc(x, fs)` se comporta análogamente, pero hace un escalado previo de la magnitud.

Teoría de muestreo: Consideraremos muestreo ideal, $x[n] = x(nT_s)$, donde T_s es el periodo de muestreo. La frecuencia de muestreo es $f_s = 1/T_s$. Recordemos que el criterio de muestreo establece que $f_s > 2f_{\max}$.

1. Construye un vector `x1` de muestras de una señal sinusoidal con $A = 100$, $\Omega_0 = 2\pi(1100)$, y $\phi = 0$. Utiliza $f_s = 8000$, y calcula el número total de muestras equivalentes a dos segundos de duración. Utiliza `sound()` para reproducir el vector y escucha el resultado. Crea el fichero `ejemplo6.m`.

2. Calcula un vector `x2` de muestras (otros dos segundos) para $A = 100$, $\Omega_0 = 2\pi(1650)$, y $\phi = \pi/3$. Escucha la señal. ¿Qué relación existe con la del primer caso?. Construye un nuevo vector `xx = [x1 zeros(1,2000) x2];`. Escucha la señal y comenta lo que oyes. Crea el fichero `ejemplo7.m`.

3. Reproduce `xx` de nuevo, pero doblando la frecuencia de muestreo indicada a `sound()`. No recalcules la muestras de `xx`. Describe lo que escuchas. Observá y explica los cambios en duración y entonación de la señal. Crea el fichero `ejemplo8.m`.

Teclado de piano: Posteriormente sintetizaremos las notas de una pieza musical conocida. Como estas señales utilizan tonos sinusoidales para representar las notas de un piano, haremos una pequeña incursión en las características básicas de éste.

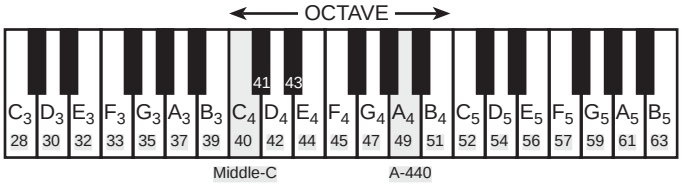

Figura 1: Subconjunto de teclas de un piano.

Consideraremos que la nota La-4 (A-4) tiene una frecuencia de 440 Hz, y que se ajusta al afinamiento equitemperado; es decir, que entre cada tecla del piano y la siguiente (contando tanto las teclas negras como las blancas) la frecuencia aumenta en un factor $2^{1/12}$. Observa que según esta relación, de una nota a la homónima de la siguiente octava se aumenta la frecuencia al doble.

Pentagramas: Las notas que aparecen en una partitura, gracias a su aspecto y su posición, permite definir simultáneamente tres parámetros:

- La posición vertical de la nota define su altura (aguda o grave). Las notas situadas más arriba en el pentagrama son más agudas.
- La posición horizontal de la nota define el momento de ejecución. El eje horizontal del pentagrama define una escala de tiempo creciente de la izquierda hacia la derecha. Dos notas colocadas en la misma columna indican simultaneidad: un acorde.
- La forma de la nota define su duración. Cada tipo de nota es dos veces más larga que la siguiente: redonda, blanca, negra, corchea, semicorchea, fusa, semifusa, ...

También existen silencios de un duración determinada. Como las notas, están organizados en longitudes decrecientes por un factor de dos. A la longitud de la nota redonda corresponde el silencio llamado pausa, a la blanca la media pausa, a la negra el suspiro, a la corchea el silencio de corchea, a la semicorchea el silencio de semicorchea, ...

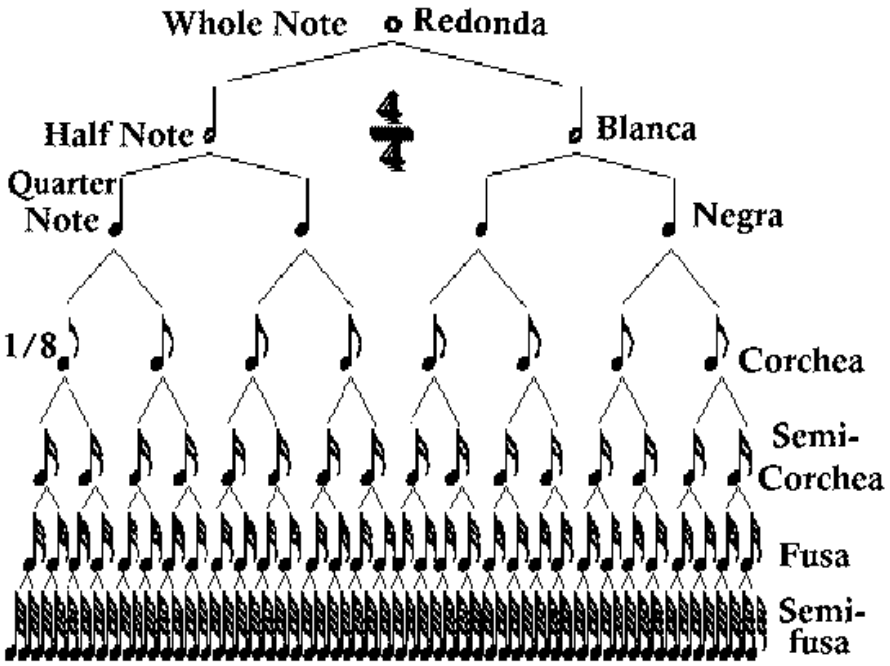

Figura 2: Duración de las notas.

Redonda	W		Silencio de redonda	w	
Blanca	H		Silencio de blanca	h	
Negra	Q		Silencio de negra	q	
Corchea	E		Silencio de corchea	e	
Semicorchea	S		Silencio de semicorchea	s	
Fusa	T		Silencio de fusa	t	

Los pentagramas están divididas en compás, separados por líneas verticales dispuestas regularmente en el pentagrama. Los compases dividen los pentagramas en intervalos de tiempo iguales.

A la izquierda de cada pentagrama se coloca una clave que permite definir varios parámetros:

- La forma de la clave indica la correspondencia entre las líneas de la pentagrama y la altura de las notas correspondientes.

  La clave de sol (pentagrama alto) indica que la línea inferior corresponde a un Mi, espacio entre las dos líneas inferiores a un Fa, la línea encima a un Sol, y así sucesivamente en el orden: Mi, Fa, Sol, La, Si, Do, Re, Mi, Fa, Sol, ...

  

  En clave de Fa, la línea inferior corresponde a un Sol grave, y subiendo, La, Si, Do, Re, Mi, Fa, ...

  

- A su derecha existen dos números. Aquí 4 y 4 indican la signatura tiempo, es decir la longitud de cada compás. 4/4 significa que un compás corresponde a la longitud de 4 negras, es decir de una redonda, de dos blancas, de ocho corcheas, ...

En un pentagrama se muestra qué notas deben tocarse y su duración relativa. La figura muestra la correspondencia entre las notas en el piano y en el pentagrama.

1. Genera una sinusoide de dos segundos que represente la nota Mi-5 (tecla número 56). Escoge valores apropiados para T_s y `fs`. Construye para ello el código `Mi5.m`.

2. Escribe una función Matlab de nombre `nota.m` que genere una nota especificada durante cierto tiempo. La función debe tener la siguiente forma:

   ```
   function tono = nota(NumeroTecla, Duracion)
   ```

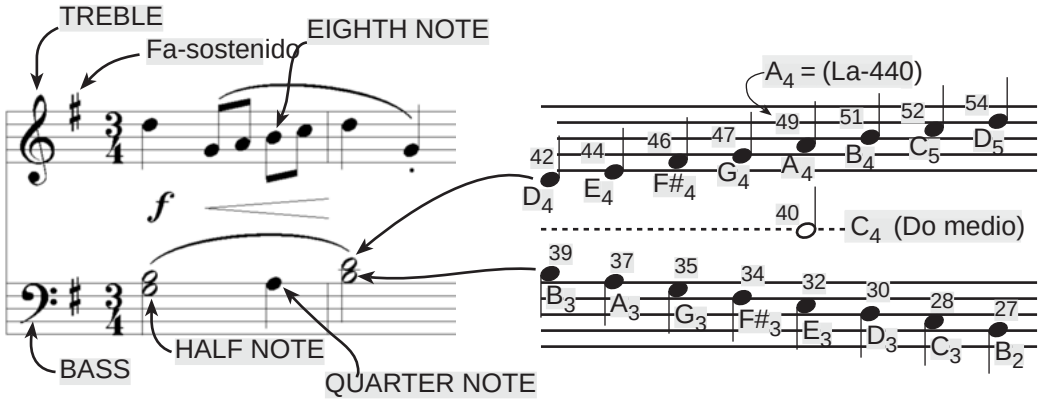

Figura 3: Las notas de un pentagrama representan un diagrama tiempo-frecuencia, dónde la posición vertical indica frecuencia de la nota que debe interpretarse.

```
% NOTA   Produce una sinusoide correspondiente a una tecla del piano.
%
% utilización:  tono = nota(NumeroTecla, Duracion)
%
%        tono = la salida sinusoidal
% NumeroTecla = indica la tecla del piano de la nota deseada
%    Duracion = Segundos de la nota generada
%
fs = 11025;
tt = 0:(1/fs):dur;
freq =
tono =
```

3. Genera una macro que reserve un vector `x` y que vaya rellenándolo mediante llamadas a `nota()` para generar una escala musical. Por último, debe llamar a `sound()` para reproducirlo. Almacena el resultado en el fichero `escala.m`.

Síntesis de notas musicales: Comenzaremos generando una melodía muy conocida[1]. Construye una señal que se corresponda a las siguientes notas `G4 G4 G4 Eb4 F4 F4 F4 D4` y duraciones `E E E H E E E H`. Reprodúcela y escúchala. Genera el fichero `quinta.m` con el código necesario.

[1]Según el autor, la descripción de las cuatro primeras notas es que "Así llama el destino a la puerta de los hombres ..."

Espectrograma de la música: la notación musical describe cómo una canción está compuesta de frecuencias distintas y cuándo deberían tocarse cada una. La partitura es por tanto una representación tiempo-frecuencia de la señal que sintetiza la canción. En Matlab podemos calcular representaciones tiempo-frecuenica mediante la función `specgram()`.

Utiliza esta función sobre la composición generada por la función `quinta.m`, comparalá con el pentragrama asociado.

Modificaciones y mejoras (opcional) La música probablemente suene muy artificial, porque se ha creado utilizando sinusoides puros. Para mejorarla pueden incorporarse algunas modificaciones. Por ejemplo, puede multiplicarse cada tono puro de la señal por una envolvente $E(t)$

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

La envolvente debería surgir rápido y desvanecerse más despacio. Un convenio habitual es dividir la envolvente en cuatro secciones: ataque (A), decaimiento (D), sostenimiento (S), y relajación (R). La siguiente figura muestra un perfil típico de ADSR Mejora la composición

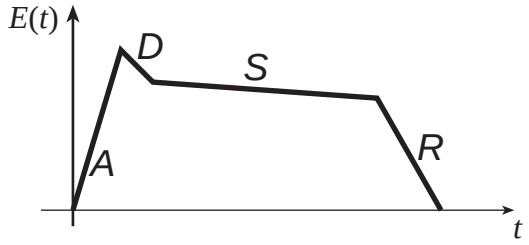

Figura 4: Perfil ADSR de una función envolvente.

de `quinta.m` con la envolvente.

Algunos otros aspectos que afectan a la calidad de la síntesis incluyen la temporización relativa de las notas, la duración correcta de tempo, reposos en los lugares apropiados, amplitudes relativas para enfatizar ciertas notas, y armónicos. El verdadero sonido del piano contiene varias componentes frecuenciales, como segundos y terceros armónicos. Esta modificación es sencilla, pero ten cuidado de hacer que las amplitudes de los armónicos sean menores que la de la frecuencia fundamental. Mejora la composición de `timbre.m` con la riqueza armónica.