Manual de prácticas para la detección automática de melodías usando la transformada de Fourier de ventana corta y la transformada wavelet

PRÁCTICA 1:

Objetivo: Familiarizar los estudiantes con las escalas musicales y con los conceptos relacionados con la naturaleza del sonido.

Esta primera práctica consiste en generar las escalas musicales pitagórica y temperada. Para la escala pitagórica se implementa el código para la ecuación 1 tal como se describe a continuación.

$$F_n = \left(\frac{3}{2}\right)^n F_0 \tag{1}$$

Con n un número entero desde 0 hasta 12 para cada octava generada, F_n la frecuencia generada a partir de la frecuencia de referencia F_0 que para nuestra escala actual se tomó la nota LA=110 Hz. Con las siguientes líneas de código en MATLAB, se puede crear un vector con la escala cromática pitagórica, es decir que contiene los 12 semitonos entre octava.

```
%% ----------------------ESCALA PITAGORICA-------------------------------
---

%------------------parámetros iniciales----------------------------------
--
A=110;%--------------------------------nota LA como referencia
fs=44100;%----------------------------frecuencia de muestreo para audio
duracion=0.5;%-------------------------duración de la cada nota
t=linspace(0,duracion,fs*duracion);%---vector de tiempo
N_octavas=3;%--------------------------se elige el número de octavas a
generar
pitagorica=zeros(1,12*N_octavas);%-----inicia un vector con ceros con una
%longitud variable dependiendo del número de octavas


for i=1:N_octavas
    for j=(12*(i-1)+1):(12*(i-1)+12)
        %se implementa la ecuación (1)
        pitagorica(j)=A*(3/2)^(j-1);
        %garantiza que la nota generada esté dentro de la octava
        while (pitagorica(j)>=A*2^i)
               pitagorica(j)=pitagorica(j)/2;
        end
    end
end
```

```
%ordena las frecuencias de la escala en orden ascendente
pitagorica=sort(pitagorica);
%Selecciona sólo las frecuencias usadas en la escala natural
%(1=DO 3=RE 5=MI 6=FA 8=SOL 10=LA 12=SI)
Natural=[1 3 5 6 8 10 12];

for i=1:N_octavas
    for j=1:length(Natural)
        %genera el vector de audio
        y=sin(2*pi*pitagorica(Natural(j))*t);
        %multiplica el vector por una exponencial negativa para simular
la
        %caída del sonido tal como ocurre en instrumentos de cuerda
reales
        y=(exp(-(5/duracion)*t).*y);
        sound(y,fs)
        pause(duracion)
    end
    Natural=12+Natural
end
```

En el código anterior se creó un vector “Natural”, con el fin de seleccionar únicamente la escala natural tradicional (DO-RE-MI-FA-SOL-LA-SI) con la cual estamos mas familiarizados, esto con el fin de comparar auditiva y gráficamente la escala pitagórica con la escala temperada que se implementa a partir de la ecuación 2.

$$F_n = 2^{n/12} F_0 . \tag{1}$$

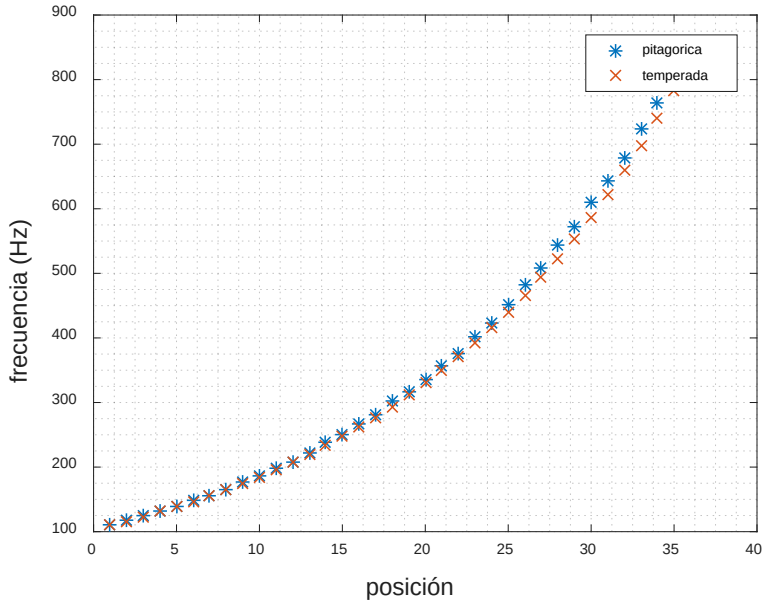
El código en MATLAB para la anterior ecuación se puede implementar como se muestra a continuación:

```
%% ----------------------ESCALA TEMPERADA-------------------------------
--
temperada=zeros(1,12*N_octavas);%-----inicia un vector con ceros con una
%longitud variable dependiendo del número de octavas

for i=1:12*N_octavas
    %se implementa la ecuación (2)
    temperada(i)=A*2^((i-1)/12);
end
Natural=[1 3 5 6 8 10 12];
for i=1:N_octavas
    for j=1:length(Natural)
        y=sin(2*pi*temperada(Natural(j))*t);
        y=(exp(-(5/duracion)*t).*y);
        sound(y,fs)
        pause(duracion)
    end
    Natural=12+Natural
end
```

Con el fin de comparar las dos escalas, se realiza una gráfica de los valores de frecuencia en función de cada semitono de la escala cromática

```
plot(pitagorica,'*')
xlabel('posición','FontSize',16)
ylabel('frecuencia (Hz)','FontSize',16)
grid minor
hold on
plot(temperada,'x')
legend('pitagorica','temperada')
```



En esta gráfica se puede observar que las escalas pitagórica y temperada son muy similares para la primera octava (12 semitonos), no obstante, a medida que aumentan las octavas, la escala pitagórica tiende a alejarse de los valores de la escala temperada, recordando que la octava se alcanza cuando la frecuencia F_n es un múltiplo entero de la frecuencia F_0, ($2F_0$, $3F_0$…nF_0). Esto no ocurre exactamente en la escala pitagórica tal como se vio en la sección teórica de la práctica, debido a la llamada coma pitagórica. Una vez comprendida la diferencia entre las escalas pitagórica y temperada, las siguientes prácticas se realizarán usando la implementación de la segunda, pues es la escala usada en la mayor parte de la música actual.

Tal como se explicó en la sección teórica, el timbre o color del sonido es la característica que nos permite distinguir entre instrumentos musicales o las voces de las personas. Para comprender mejor el concepto de timbre o color musical, se propone añadir armónicos al vector de audio, y realizar variaciones de sus parámetros de amplitud y frecuencia tal como se muestra en el siguiente código:

```
%% ----------------------ESCALA TEMPERADA-------------------------------
--
%-----------------parámetros iniciales---------------------------------
--
A=220;%--------------------------------nota LA como referencia
fs=44100;%-----------------------------frecuencia de muestreo para audio
duracion=0.5;%-------------------------duración de la cada nota
t=linspace(0,duracion,fs*duracion);%---vector de tiempo
N_octavas=1;%--------------------------se elige el número de octavas a
generar
pitagorica=zeros(1,12*N_octavas);%-----inicia un vector con ceros con una
%longitud variable dependiendo del número de octavas

temperada=zeros(1,12*N_octavas);%-----inicia un vector con ceros con una
%longitud variable dependiendo del número de octavas

for i=1:12*N_octavas
    %se implementa la ecuación (2)
    temperada(i)=A*2^((i-1)/12);
end
%Selecciona sólo las frecuencias usadas en la escala natural
%(1=DO 3=RE 5=MI 6=FA 8=SOL 10=LA 12=SI)
Natural=[1 3 5 6 8 10 12];

%----------amplitud y frecuencia de los armónicos-----------------------
--
%al variar los valores de amplitud y frecuencia, cambia el timbre del
%sonido
A1=1;
f1=2;
A2=0.6;
f2=6;
A3=0.8;
f3=12;
%-------------------------------------------------------------------------
--
decaimiento=7;%parámetro para la atenuación del sonido

for i=1:N_octavas
    for j=1:length(Natural)
        y=A1*sin(2*pi*f1*temperada(Natural(j))*t)+...
          A2*sin(2*pi*f2*temperada(Natural(j))*t)+...
          A3*sin(2*pi*f3*temperada(Natural(j))*t);
        y=(exp(-(decaimiento/duracion)*t).*y);
        sound(y,fs)
        pause(duracion)
    end
    Natural=12+Natural
end
```

los parámetros A1, A2 y A3 serán modificados con diferentes valores entre cero y uno (mayores a 1 pueden generar saturación en la función sound de MATLAB), mientras que los parámetros f1, f2, f3 tomarán valores enteros mayores a uno.

PRÁCTICA 2:

Objetivo: Crear melodías musicales a partir de la escala temperada previamente implementada

En clase se vieron los conceptos fundamentales relacionados con la notación musical, el solfeo, las figuras rítmicas, las diferencias entre armonía y melodía. En esta práctica se implementa un generador de melodías basado en la escala temperada previamente implementada, en donde se crean dos vectores de datos, uno asociado a la nota musical el cual hemos llamado “notas” y otro asociado a la duración de la nota que hemos llamado “tempo”. Al ser la escala temperada generada la escala cromática, es decir que contiene 12 notas por octava, se pueden crear melodías en todas las tonalidades y no sólo en las escalas naturales. En cuanto al vector de tiempo, se han creado las figuras rítmicas más comunes, la redonda con duración de 4 tiempos con (referencia a la negra), la blanca con duración de 2 tiempos, la negra con duración de un tiempo, la corchea con duración de ½ de tiempo y la semicorchea con duración de ¼ de tiempo, omitiendo por lo pronto las figuras de fusa y semifusa, así como patrones rítmicos mas complejos basados en puntillo, síncopas entre otros, pero que para prácticas futuras se puede implementar sobre el presente código.

Para realizar la práctica, se proponen 3 melodías que se pueden seleccionar por medio de la variable “melodía” del `switch` en el código. Se propone dentro de la práctica que el estudiante varíe a su gusto los vectores de notas y tempo, creando sus propias melodías.

```
%---------------------parámetros iniciales-------------------------------
C=130;%--------------------------------nota DO como referencia
fs=44100;%-----------------------------frecuencia de muestreo para audio
duracion=0.5;%---------------------------duración de la cada nota
t=linspace(0,duracion,fs*duracion);%---vector de tiempo
N_octavas=2;%--------------------------se elige el número de octavas a
generar
temperada=zeros(1,12*N_octavas);%-----inicia un vector con ceros con una
%longitud variable dependiendo del número de octavas

temperada=zeros(1,12*N_octavas);%-----inicia un vector con ceros con una
%longitud variable dependiendo del número de octavas

for i=1:12*N_octavas
    %se implementa la ecuación (2)
    temperada(i)=C*2^((i-1)/12);
end
%% -----------------notas y tiempos para la melodía----------------------
-

%Para el vector de notas
 %1=DO  2=DO#  3=RE  4=RE#  5=MI  6=FA  7=FA#  8=SOL  9=SOL# 10=LA 11=LA#
12=SI
%13=DO 14=DO# 15=RE 16=RE# 17=MI 18=FA 19=FA# 20=SOL 21=SOL# 22=LA 23=LA#
24=SI
```

```
%Para el vector de tempo
%1=redonda
%2=blanca
%3=negra
%4=corchea
%5=semicorchea

%notas=[ 1, 5,12, 5,12,13,17, 3, 8,13,12,13,15,13,10, 8, 5, 3, 1];
%tempo= [ 4, 4, 4, 3, 4, 4, 4, 3, 3, 4, 4, 3, 5, 5, 5, 5, 2, 3, 1];

bpm=120;%-----------------------------Bit Por Minuto

melodia=3;
switch melodia
    case 1%escala de DO MAYOR natural
        %la escala de DO MAYOR es una escala que no presenta ninguna
        %alteración, es decir ninguna nota con sostenidos ni bemoles
        %DO=1 RE=3 MI=5 FA=6 SOL=8 LA=10 SI=12
        notas= [ 1, 3, 5, 6, 8, 10, 12, 10, 8, 6, 5, 3, 1];
        tempo= [ 3, 3, 3, 3, 3,  3,  3,  4, 4, 4, 4, 4, 2];
    case 2%esccala de blues
        %la escala blues consiste en una escala pentatónica con un cuarto
        %grado aumentado. Para la escala de LA menor la
        %LA=10 DO=13 RE=15 RE#=16 MI=17 SOL=20 LA=22
        notas= [10,13,15,16,17,20,22];
        tempo= [ 3, 4, 4, 4, 3, 3, 2];
    case 3%figuras ritmicas
        %en este caso se hacen variaciones ritmicas basadas en figuras
        %musicales de corcheas=4 y semicorcheas=5 en el vector de tempo
        notas= [ 1, 1, 1, 3, 3, 3, 5, 5, 5, 5, 3, 3];
        tempo= [ 4, 5, 5, 4, 5, 5, 5, 5, 5, 5, 4, 4];
end
v_notas=zeros(1,length(notas));
for i=1:length(notas)
    v_notas(i)=temperada(notas(i));
end

LY=0;
for i=1:length(v_notas)
    switch tempo(i)
    case 1
        t=0:1/fs:4*60/bpm;%----------Redonda (4 tiempo de negra)
    case 2
        t=0:1/fs:2*60/(bpm);%--------Blanca (2 tiempos de negra)
    case 3
        t=0:1/fs:60/(bpm);%----------Negra (1 tiempo)
    case 4
        t=0:1/fs:60/(2*bpm);%--------Corchea(1/2tiempo de negra)
    case 5
        t=0:1/fs:60/(4*bpm);%--------Semicorchea(1/4 tiempo de negra)
    end

duration=length(t)/fs;
```

```
%----------amplitud y frecuencia de los armónicos-----------------------
--
%al variar los valores de amplitud y frecuencia, cambia el timbre del
%sonido
A1=-1;  f1=2;
A2=0.7;  f2=6;
A3=0.5;  f3=12;
A4=0.3;  f4=12;
%-------------------------------------------------------------------------
--
a=5;%parámetro para la atenuación del sonido

y=A1*sin(2*pi*f1*v_notas(i)*t)+...
  A2*sin(2*pi*f2*v_notas(i)*t)+...
  A3*sin(2*pi*f3*v_notas(i)*t)+...
  A4*sin(2*pi*f4*v_notas(i)*t);
y=0.5*(exp(-(a/duration)*t).*y);

Ly=length(y)
Y((LY+1):LY+Ly)=y;
LY=length(Y)
end
figure()
plot(Y)
sound(Y,fs)
```

se pueden guardar las melodías usando función audiowrite:

```
audiowrite('melodia_1.wav',Y,fs);
```

Esta melodía será usada en las prácticas siguientes.

PRÁCTICA 3:

Objetivo: Realizar el análisis en frecuencia de las melodías generadas mediante la transformada de Fourier

En esta práctica se utilizan las melodías generadas en la práctica anterior, con el fin de realizar el análisis en frecuencia correspondiente

```
[y,fs]=audioread('C:\Users\ingel\Documents\MEGA\4. PAPER Chords detection
Using STFT and Wavelet\codigo matlab\melodia_2.wav');
Ly=length(y);
ti=0.01;
tf=Ly/fs
y=y((fs*ti):fs*tf,1);
sound(y,fs)
```

```
t=ti:1/fs:tf;%vector de tiempo
figure(1)
plot(t,y)
grid on
title('Señal en el tiempo')
xlabel('Tiempo (s)','FontSize',14)
ylabel('Amplitud (V)','FontSize',14)
```

Para hacer el análisis en frecuencia se puede usar la función FFT de Matlab, con las siguientes líneas de código

```
%% ----------------------------------FFT--------------------------------
----
Fy=fft(y,Ly/2);%--------------------calcula la transformada de Fourier
Fy = abs(Fy(1:fix(end/2)));%--------obtiene el valor absoluto y toma sólo
la mitad
Fy = 2*Fy/Ly;%----------------------normaliza la intensidad
LFy=length(Fy)
f=linspace(1,fs/2,LFy);%------------crea el vector de frecuencia
figure(2)
plot(f(1:10000),Fy(1:10000))
grid on
title('FFT')
xlabel('Frecuencia(Hz)','FontSize',14)
```