Guía de Trabajos Prácticos VIII – Identificación de sistemas

1. En este ejercicio se generó una señal aleatoria con distribución uniforme y media 0, se la pasó por el sistema $y[n]=0{,}3.y[n-1]-0{,}4.y[n-2]+0{,}2.y[n-3]+x[n]$ para luego tratar de identificar el sistema con el método de auto correlación con distintos criterios de determinación del orden. El algoritmo que se implementó genera el sistema de Wiener-Hopf a partir de la señal de salida del sistema a identificar y lo resuelve con el algoritmo de Levinson-Durbin. A continuación se presenta el algoritmo que se implementó:

```
function [a,G] =WienerHopf(s, plim, tol)
    r = zeros(plim,1);
    r0 = (s' * s);

%Estas lineas son para hacer un shift NO cíclico
    sd = shift(s,1);
    sd(1) = 0;

    r(1) = s' * sd;
    k = -r(1)/r0;
    a = [ k ];
    E = (1 - k*k) * r0;
    I = log(E) + (2)/length(s);
    for t = 2 : plim
        sd = shift(sd,1);
        sd(1) = 0;
        r(t) = s' * sd;
        k = -(r(t) + a * r((t-1):-1:1)) / E;
        a = [ a+k*a(t-1:-1:1), k ];

%Acá se pueden ver los métodos de determinación del orden
        En = E * ( 1 - k*k );
        In = log(En) + (2*t)/length(s);
        if ((1-En/E < tol) || (tol<0 && In > I))
            break;
        endif;

        E=En;
        I=In;
    endfor
    a = [1, a];
    G=sqrt(E);
endfunction;
```

Los criterios de determinación del orden que se probaron fueron:

- Error de predicción final:

$$1-\frac{E_{p+1}}{E_p} < \gamma \qquad \gamma\text{: tolerancia}$$

- El de Akaike:

$$I_p = \log(E_p) + \frac{2p}{N_e}$$

$$I_p < I_{p+1}$$

Como se puede ver arriba, el sistema original es de orden 3. Los resultados que se obtuvieron fueron los siguientes:

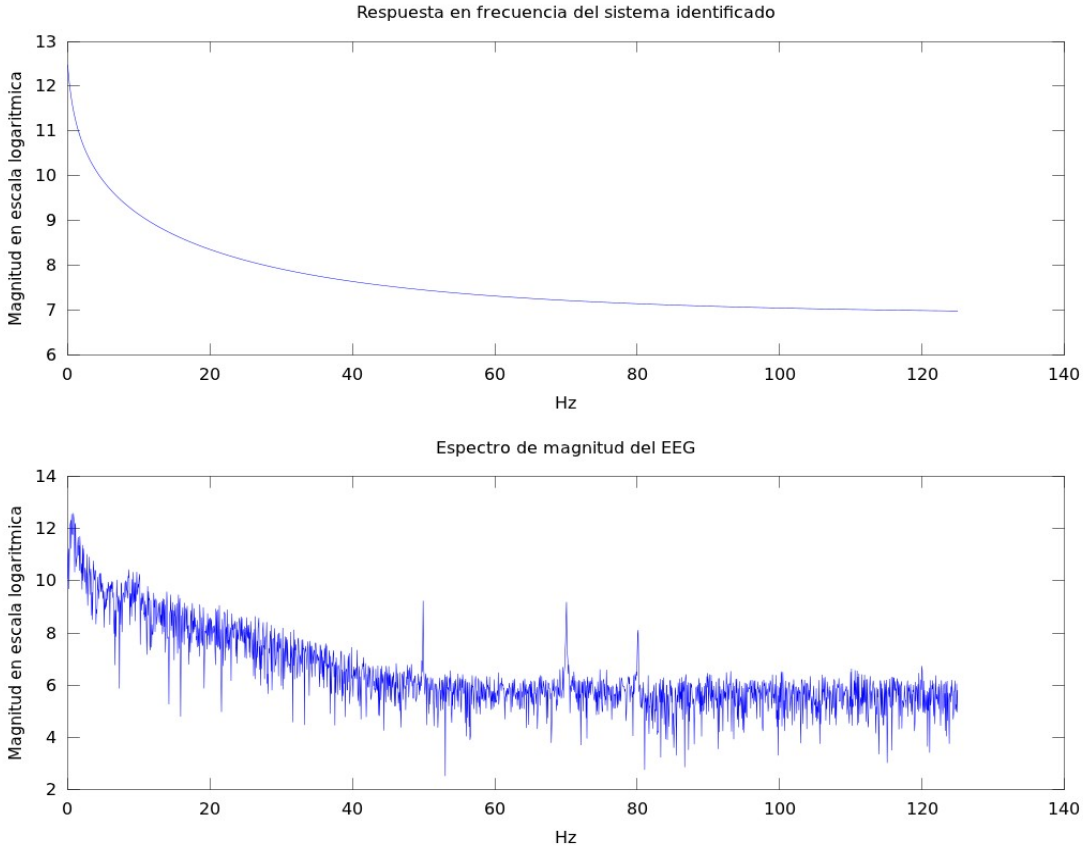
Orden original	3
Criterio de Akaike	4

Criterio de Error de predicción final (tol=0,5)	2
Criterio de Error de predicción final (tol=0,1)	3
Criterio de Error de predicción final (tol=0,05)	3
Criterio de Error de predicción final (tol=0,01)	4

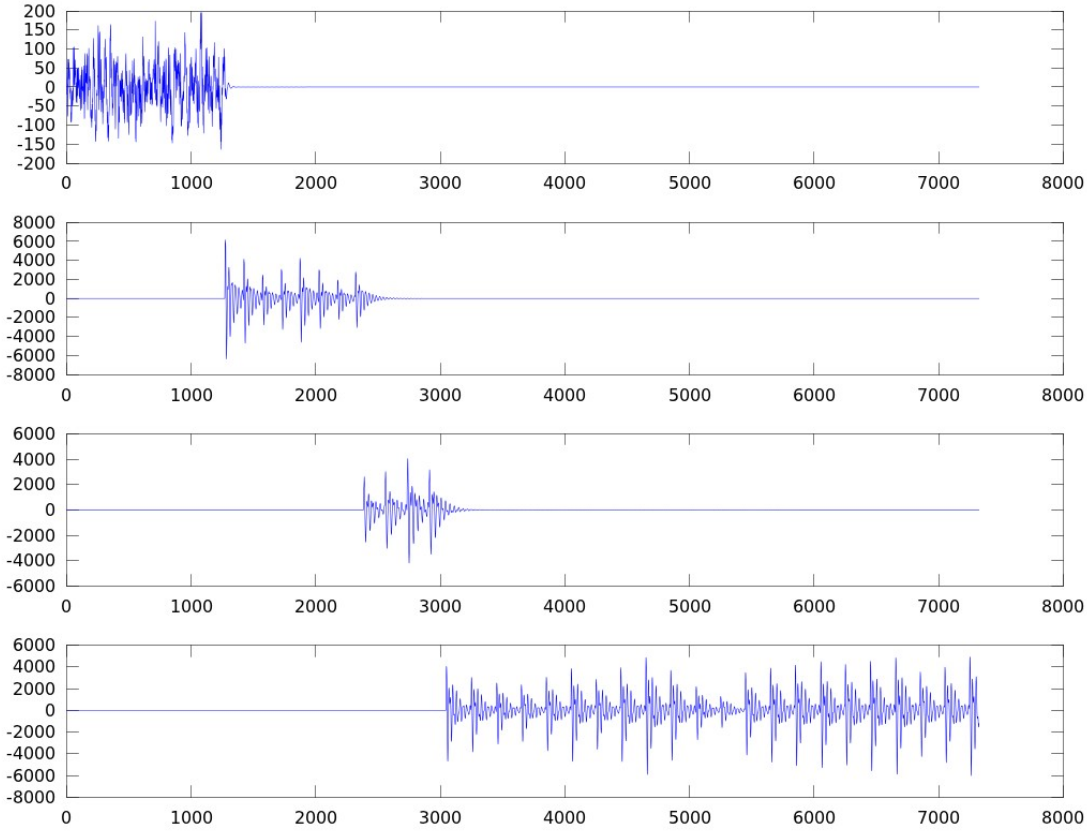
2.En este ejercicio se utilizó el algoritmo implementado en el punto anterior para identificar el sistema que generó un Electroencefalograma, suponiendo que la entrada fue ruido blanco. Una vez identificado los coeficientes del sistema se obtuvo la respuesta en frecuencia del sistema y se la comparó con el espectro de magnitud del EEG. El script que se utilizó para realizar este ejercicio es:

```
clear all;
y=load('eeg.txt');
[a,G]=WienerHopf(y,8,0);
n=length(a);
H=@(z) G / (1 + sum( a .* z.^(-1:-1:-n) ));
n=length(y);
z=exp(0:2*pi/n:pi);
n=length(z);
rf=zeros(n,1);
for k=1:n
        rf(k)=H(z(k));
end

figure(1);
subplot(2,1,1);
plot( 0:(125/(length(rf)-1)):125 ,log(rf));
title("Respuesta en frecuencia del sistema identificado");
xlabel("Hz");
ylabel("Magnitud en escala logaritmica");

subplot(2,1,2);
plot(0:(125/(length(rf)-3)):125, log(abs(fft(y)(2:end/2))));
title("Espectro de magnitud del EEG");
xlabel("Hz");
ylabel("Magnitud en escala logaritmica");

print(1, "Unidad_8-Ejer2.png", "-dpng");
```

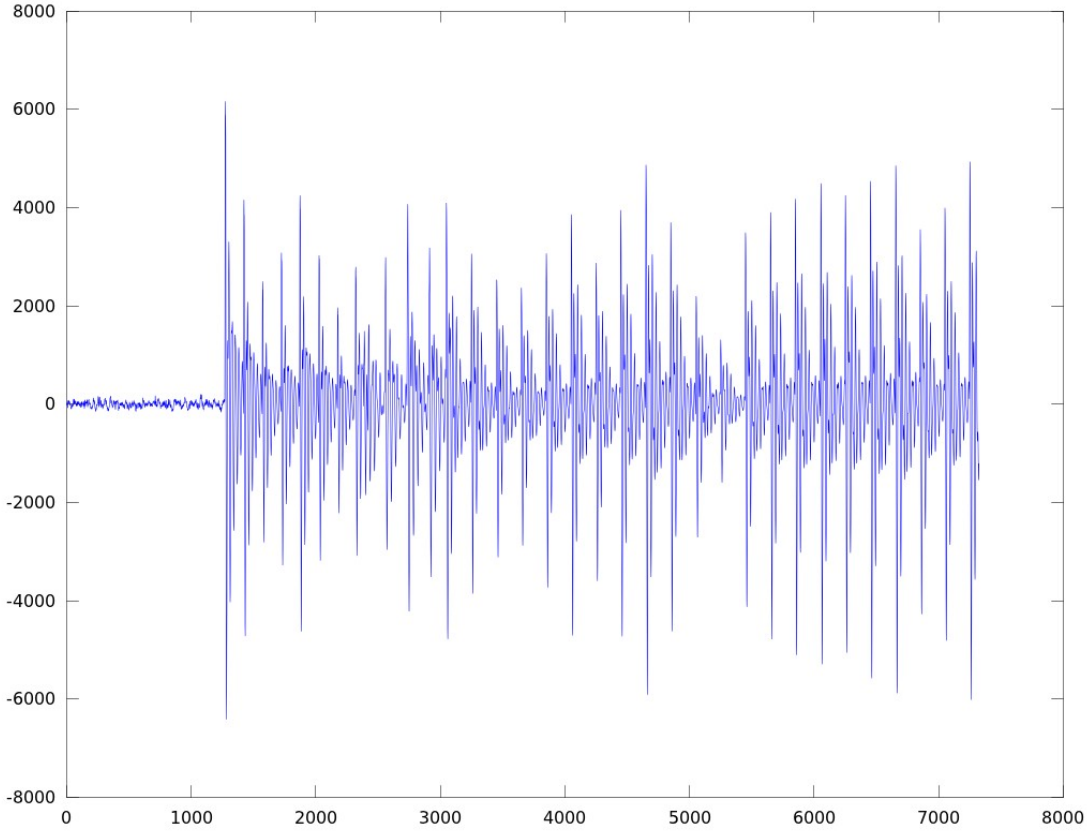
El gráfico que se obtuvo como resultado es el siguiente. Como puede verse los dos espectros de magnitud son considerablemente similares:



3. En este ejercicio la idea es identificar el sistema del aparato fonador al producir la palabra “hello”, para luego sintetizar nuevamente esta palabra. Para ello contamos con una grabación de dicha palabra y los límites de cada fonema, estos limites son necesarios ya que el sistema es variante en el tiempo.
Lo primero que realizamos fue tomar la parte de la señal correspondiente a cada fonema y multiplicarla por una ventana. Luego se utilizó el algoritmo implementado en el punto 1 para identificar 14 coeficientes del sistema que generaron cada fonema. Por otro lado, se generaron 4 señales para utilizar de entrada al sistema (una por cada fonema) y así poder sintetizar la voz, de las cuales la primera es aleatoria (fonema sordo), con distribución normal y media 0 y las otras 3 son trenes de impulsos con frecuencia de 100 Hz(fonemas sonoros). Aplicamos los sistemas definidos por los coeficientes antes obtenidos a estas señales de entradas para obtener así 4 señales, cada una correspondiente a cada fonema. Estas se pueden ver a continuación:



Al sumar estas 4 señales obtendríamos la señal deseada:



Algoritmo utilizado:

```
clear all;
y=load('hello.txt');
n=length(y);
x1=[randn(1270,1); zeros(n-1270,1)];
x2=zeros(n,1);
x3=zeros(n,1);
x4=zeros(n,1);
sp=320;

for k=1270:(sp-160):2379
    x2( k + round(randn*2) )=0.8+0.2*randn;
end
for k=2380:(sp-80):3046
    x3( k + round(randn*2) )=0.8+0.2*randn;
end
for k=3047:sp:n
    x4( k + round(randn*2) )=0.8+0.2*randn;
end

a1=zeros(14,1);
a2=zeros(14,1);
a3=zeros(14,1);
a4=zeros(14,1);
G=0;
s=y(1:1270) .* hamming(1270);
[a1, G]=WienerHopf(s,14,0);
```

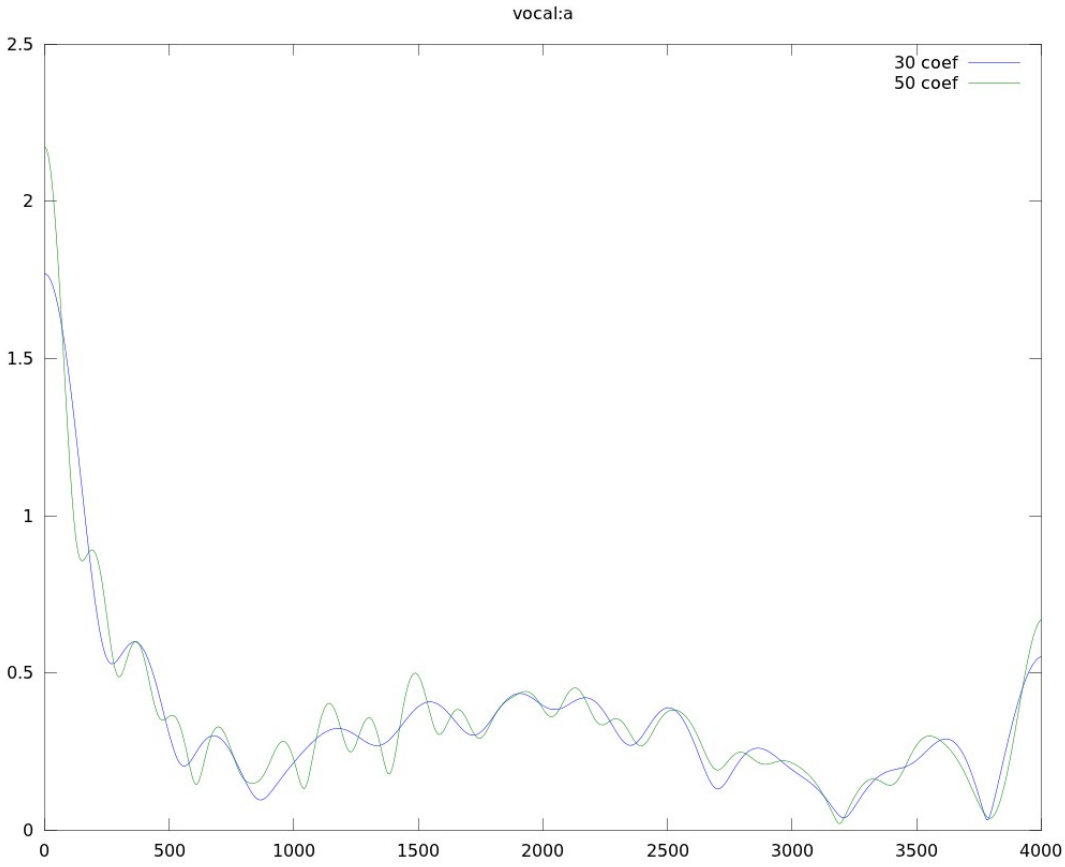
```
s=y(1271:2379) .* hamming(1109);
[a2, G]=WienerHopf(s,14,0);
s=y(2380:3046) .* hamming(667);
[a3, G]=WienerHopf(s,14,0);
s=y(3047:end) .* hamming(4277);
[a4, G]=WienerHopf(s,14,0);

yg1=filter([1],a1,x1);
yg2=filter([1],a2,x2);
yg3=filter([1],a3,x3);
yg4=filter([1],a4,x4);

yg1 *= norm(y(1:1270)) / norm(yg1);
yg2 *= norm(y(1271:2379)) / norm(yg2);
yg3 *= norm(y(2380:3046)) / norm(yg3);
yg4 *= norm(y(3047:end)) / norm(yg4);

yg=yg1+yg2+yg3+yg4;

figure(1); plot(y);
figure(2); plot(yg);

print(2, "Unidad_8-Ejer3-2.png", "-dpng");
print(3, "Unidad_8-Ejer3-3.png", "-dpng");
wavwrite(y, 16000, 16,'original.wav')
wavwrite(yg, 16000, 16, 'sintetizado.wav')
```

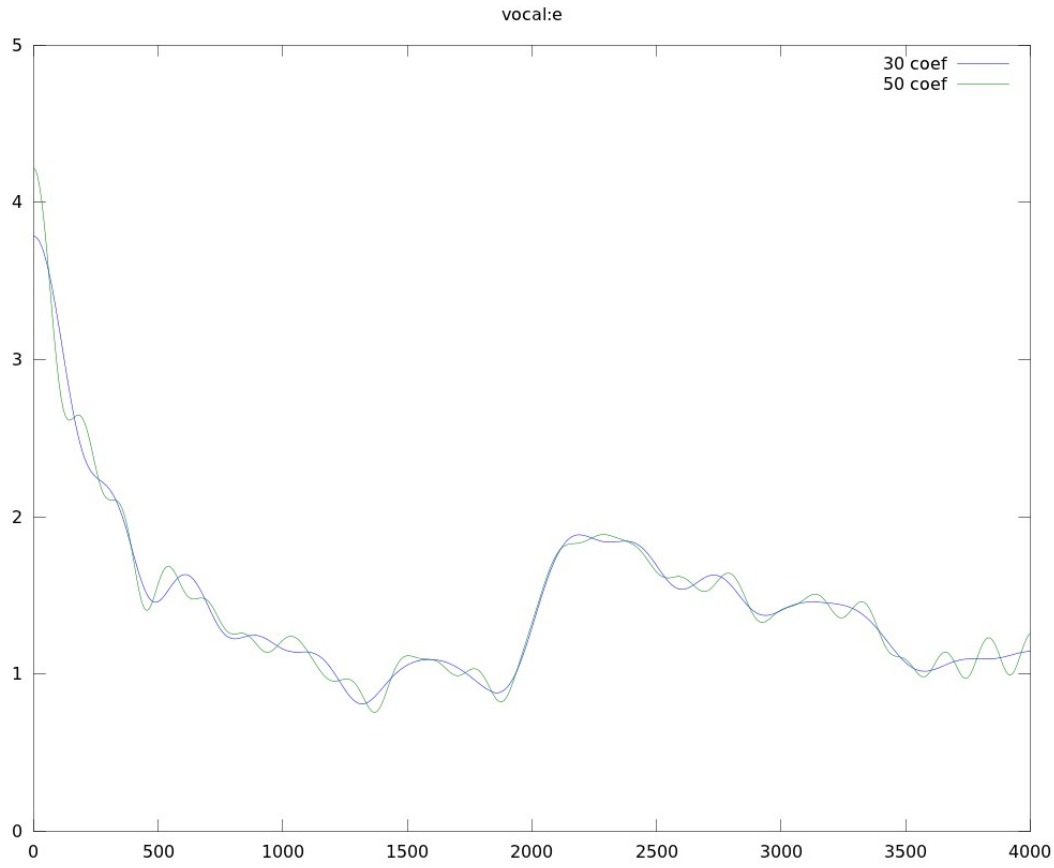
Mejoras aplicadas al algoritmo original

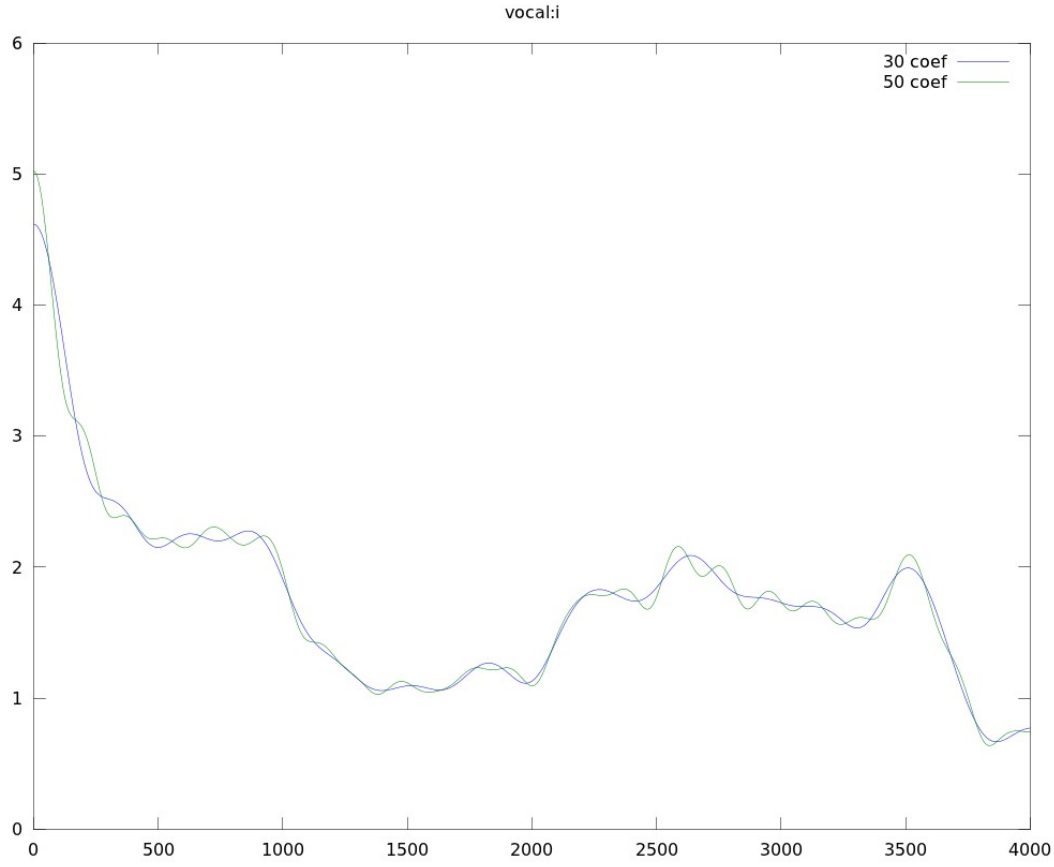
- La señal sintetizada como se detalló anteriormente sonaba “robótica”, lo cual se debe a que la frecuencia de pitch es siempre la misma. Para evitar esto se hizo que la frecuencia de los trenes de impulso varíen entre cada fonema, utilizando 100 Hz para el primero ('e'), 150 Hz para el segundo('ll') y 200Hz para el tercero('ou').
- Se generaron pequeñas variaciones aleatorias en los impulsos para obtener un sonido más natural. Se modificó tanto la magnitud como la posición (se los desplazo hasta una muestra de la que le corresponde). Estas dos mejoras se pueden observar en el código.
- Para imitar la evolución del volumen en la elocución de la palabra, una vez generado cada fonema, se lo dividió por su norma y se lo multiplicó por la norma del fonema de la grabación original.

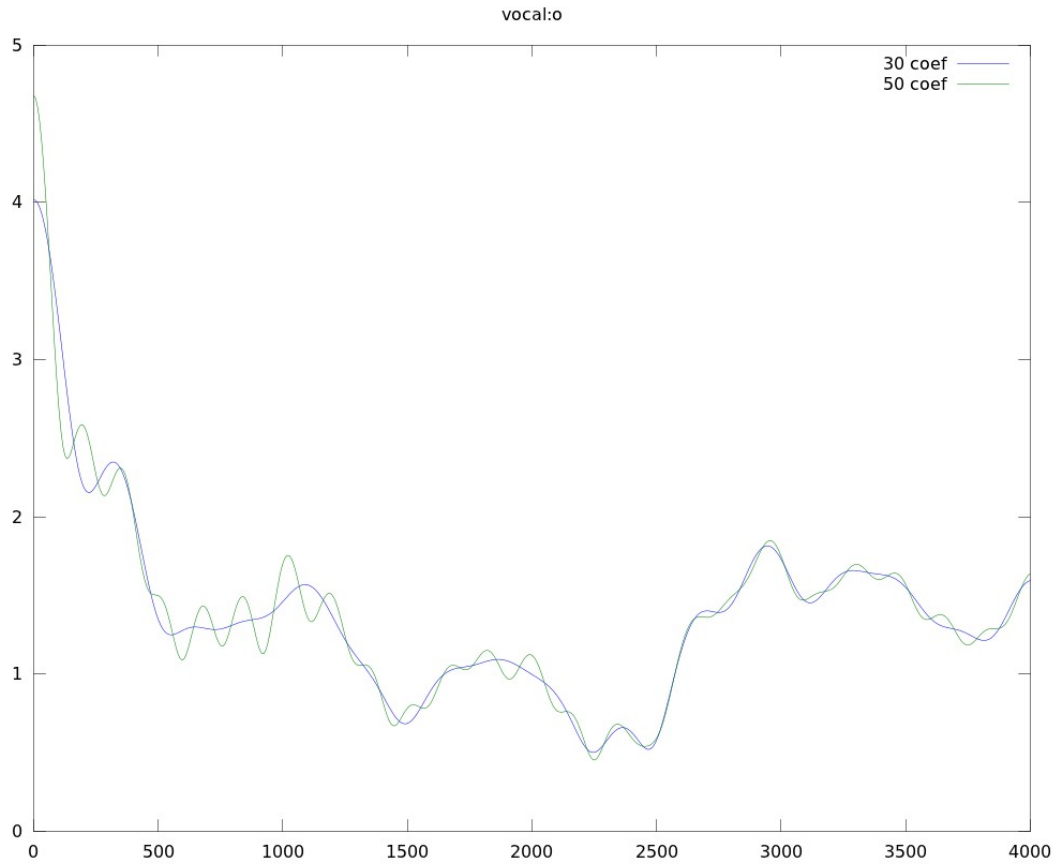
Guía de Trabajos Prácticos IX – Procesamiento Digital de Voz

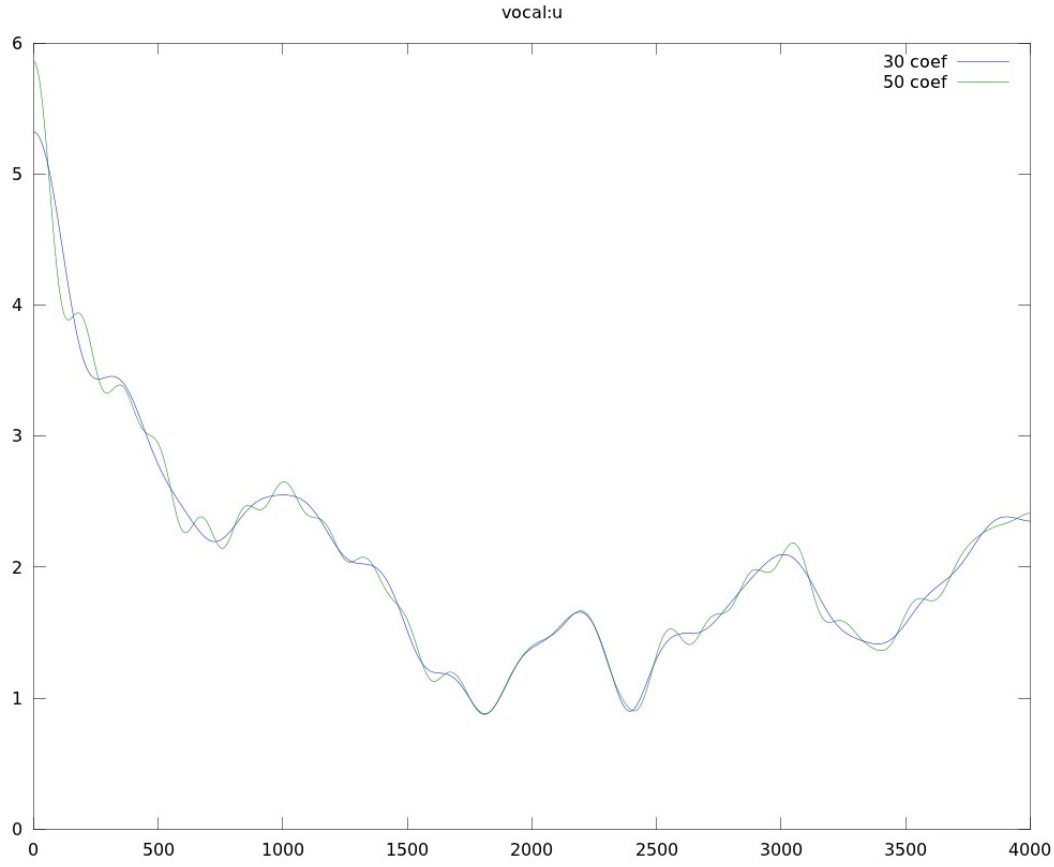
1. En este ejercicio se tomó una señal de voz registrada durante la emisión de las 5 vocales, se separó cada fonema y se le aplicó un liftrado. De esta forma se pudo separar la señal de excitación de la respuesta del tracto vocal. Luego se tomaron los primeros N coeficientes en el dominio de las cuefrencias y se le calculó el espectro de magnitud para, de esta forma, conseguir la respuesta en frecuencias del tracto vocal. A continuación vemos los resultados al tomar N=30 y N=50:




vocal:e
30 coef
50 coef
0
1
2
3
4
5
0
500
1000
1500
2000
2500
3000
3500
4000


vocal:i
30 coef
50 coef
0
1
2
3
4
5
6
0
500
1000
1500
2000
2500
3000
3500
4000


vocal:o
30 coef
50 coef
0
1
2
3
4
5
0
500
1000
1500
2000
2500
3000
3500
4000


vocal:u
30 coef
50 coef
0
1
2
3
4
5
6
0
500
1000
1500
2000
2500
3000
3500
4000

Lo que se hizo a continuación fue estimar las 2 primeras frecuencias formantes y la frecuencia de pitch. Simplemente se buscaron los primeros 3 máximos locales en las repuestas en frecuencias de las vocales. Como primera medida se intentó hacer un algoritmo que lo haga automáticamente, pero no se obtuvieron buenos resultados al estimar F1 y F2. Esto puede deberse a que la señal de excitación interfiere con la respuesta del tracto ocasionando que falle el algoritmo. Por lo tanto se optó por buscar los máximos correspondientes a F1 y a F2 a simple vista. Esto arrojó los siguientes resultados:

vocal: a
F0:145.454545 Hz
F1:800 Hz
F2:1100 Hz
vocal: e
F0:140.350877 Hz
F1: 600 Hz
F2: 2200 Hz
vocal: i
F0:142.857143 Hz
F1: 400 Hz
F2: 2200 Hz
vocal: o
F0:145.454545 Hz
F1: 400 Hz
F2:1100 Hz
vocal: u
F0:131.147541 Hz
F1: 300 Hz
F2: 1000 Hz

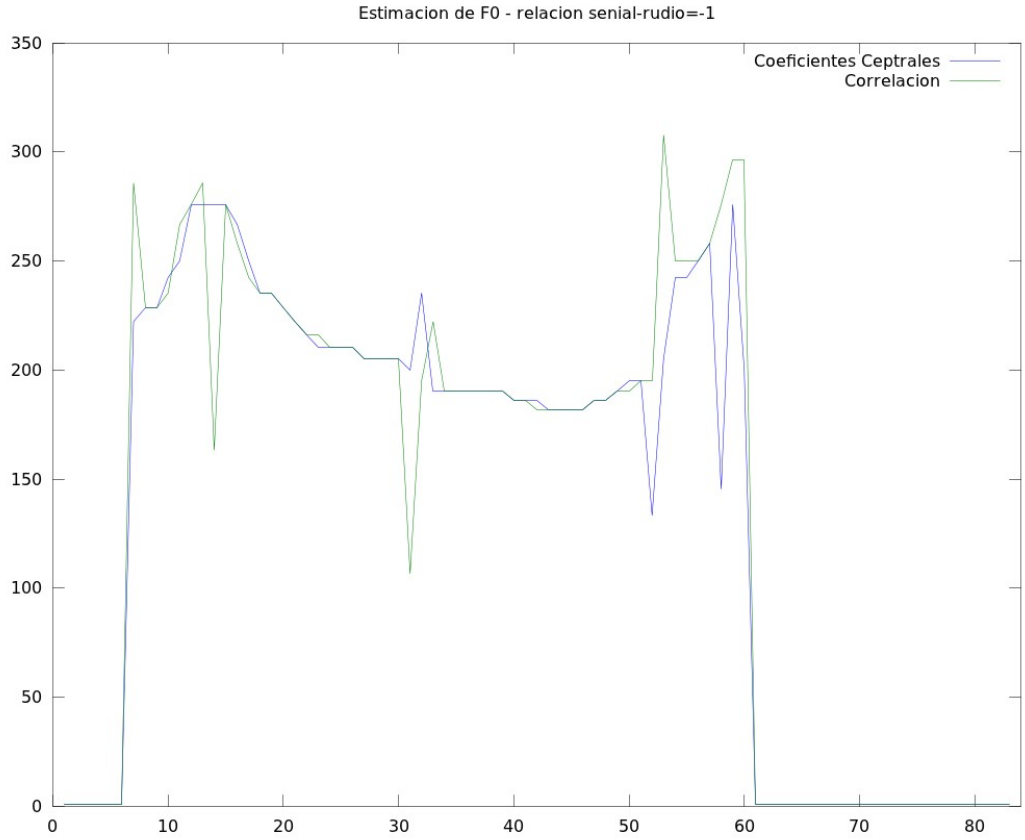
El algoritmo utilizado para realizar todo el ejercicio es le siguiente:

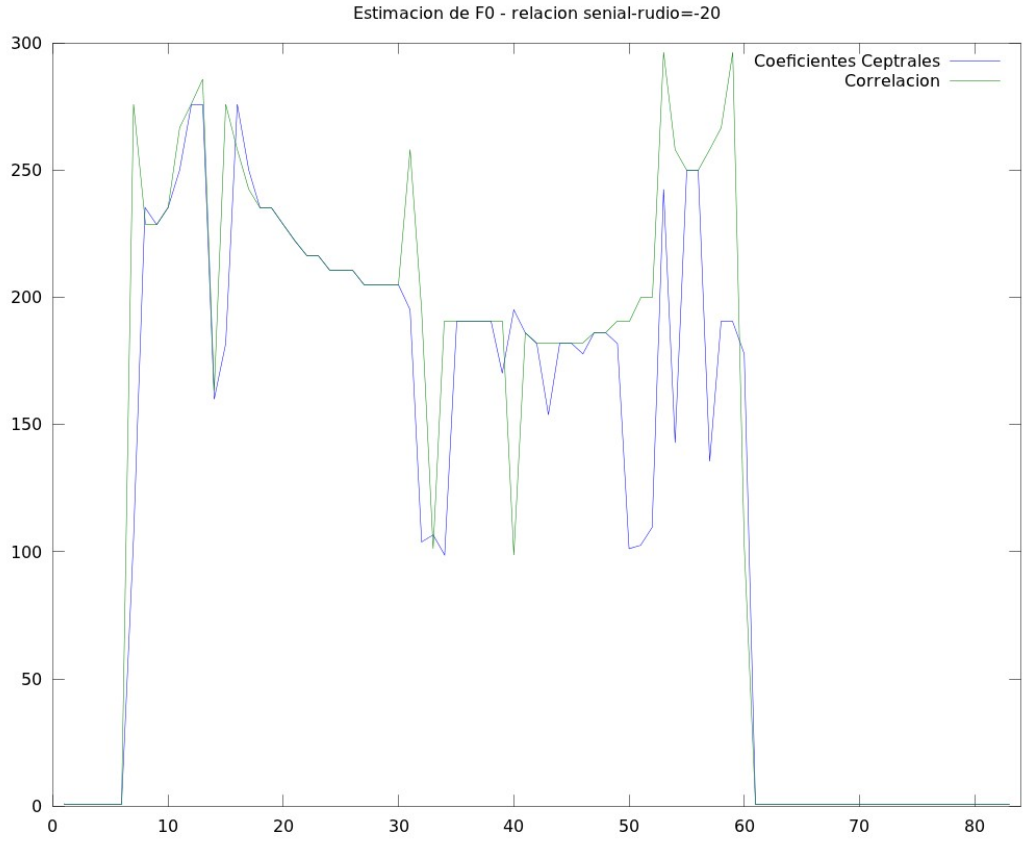
```
clear all;
clf;
letra=['a' 'e' 'i' 'o' 'u'];

#y=load('aeiou_fem.txt');
y=load('aeiou_mas.txt');

py = y.*y;
py = real(ifft( fft(py)(1:round(length(py)/125)) ));
py /= max(py);
l=1;
med=0.1;

vent0=[ones(31, 1); zeros(1024-31, 1)];
vent1=[ones(31, 1); zeros(1024-31, 1)];
vent2=[ones(51, 1); zeros(1024-51, 1)];
ejef=0:4000/511:4000;

F1=F2=zeros(1,5);
for k=1:5
        l1=l2=1;
        while( l<length(y) && py( ceil(l/125) ) < med  ) l++; end;
        l1=l;
        while( l<length(y) && py( ceil(l/125) ) > med ) l++; end;
        l2=l;
        l1 = round((l1+l2)/2 - 512);
        l2 = l1 + 1023;
```

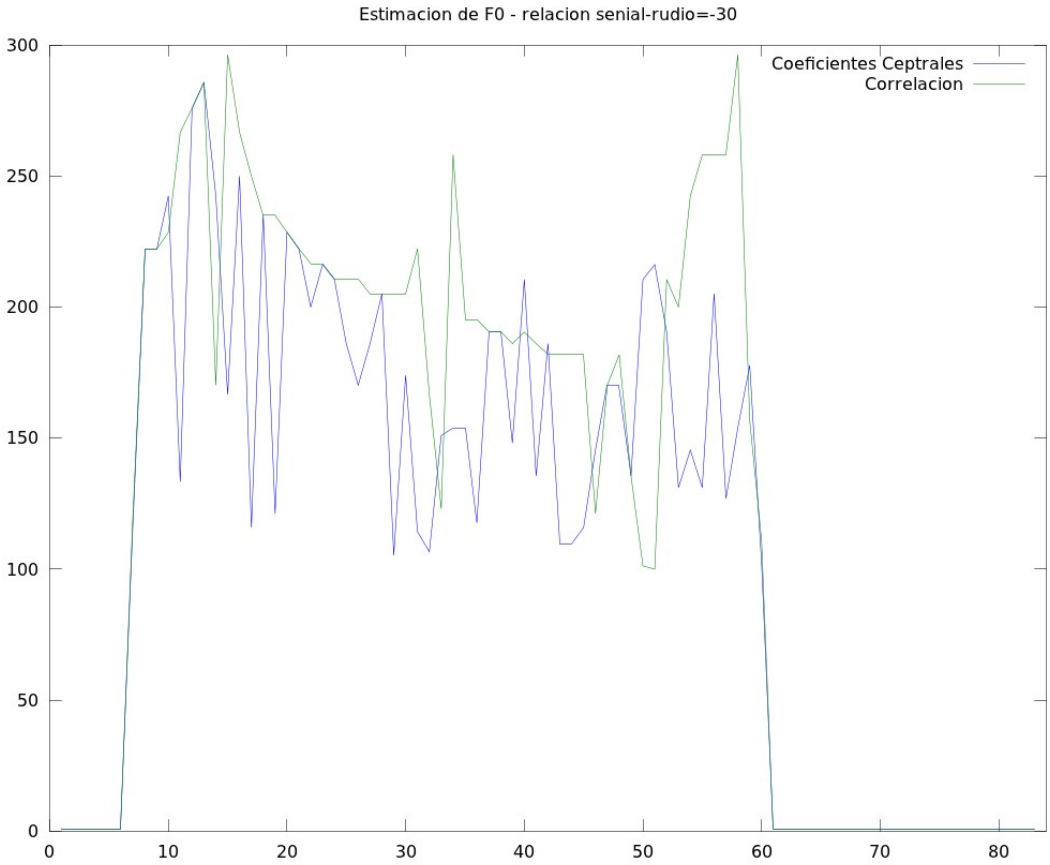
```
        s=y(l1:l2) .* hamming(1024);
        cep=abs(ifft(log(abs(fft(s)))));
        [foo,iF0]=max(cep(25:110));
        F0 = (8000/(iF0+25));

        resp0=abs(fft(cep .* vent0));
        resp1=abs(fft(cep .* vent1));
        resp2=abs(fft(cep .* vent2));

        figure(k);
        plot(ejef, resp1(1:512), ejef, resp2(1:512) );
        legend('30 coef', '50 coef');
        title( strcat('vocal: ', letra(k) ));
        print(k, strcat(strcat('vocal: ', letra(k) ),'.png'), "-dpng");

        m = round(400 * length(resp0)/8000);
        l1 = round(1000 * length(resp0)/8000);
        l2 = round(4000 * length(resp0)/8000);
        while (m<l1 && !(resp0(m)>resp0(m-1) && resp0(m)>resp0(m+1)) ) m++; end;
        F1(k)=m*4000/512;
        m=max(round(750 * length(resp0)/8000), m+20);
        while (m<l2 && !(resp0(m)>resp0(m-1) && resp0(m)>resp0(m+1)) ) m++; end;
        F2(k)=m*4000/512;
        printf('vocal: %c\n F0:%f Hz\n F1:%f Hz\n F2:%f Hz\n',letra(k), F0,F1(k),F2(k));
endfor

figure(6);
plot(F1(1), F2(1),'x',F1(2), F2(2),'x',F1(3), F2(3),'x',F1(4), F2(4),'x',F1(5), F2(5),'x');
legend('a','e','i','o','u');
title( 'distribucion de las vocales en el espacio F1-F2');
print(6, 'distribucion de las vocales en el espacio F1-F2.png', "-dpng");
```

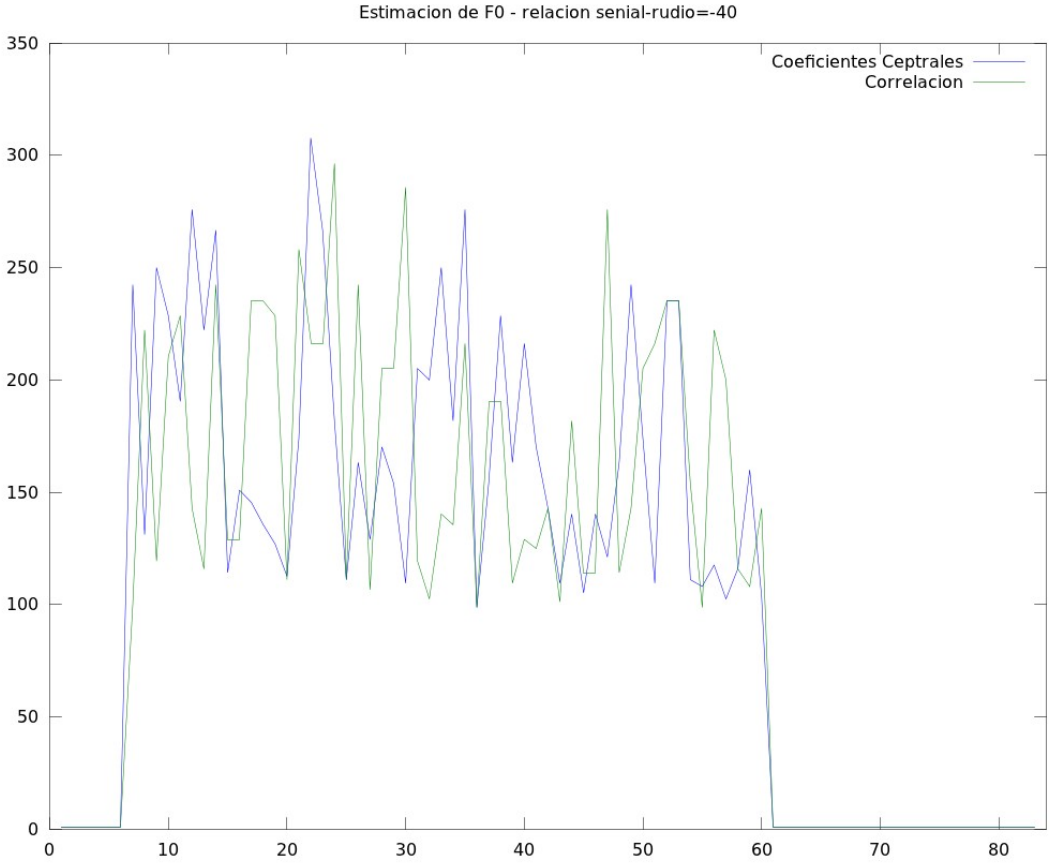
2. En este ejercicio se compararon dos técnicas de estimación del pitch de una señal de voz, la autocorrelación y los coeficientes ceptrales. Para llevar adelante esta comparación se tomó una señal de una grabación de voz, se la ensució con distintas cantidades de ruido, se separó en ventanas y se aplicó el método a cada una. Esto dio como resultado la evolución de F0 a lo largo de la frase.

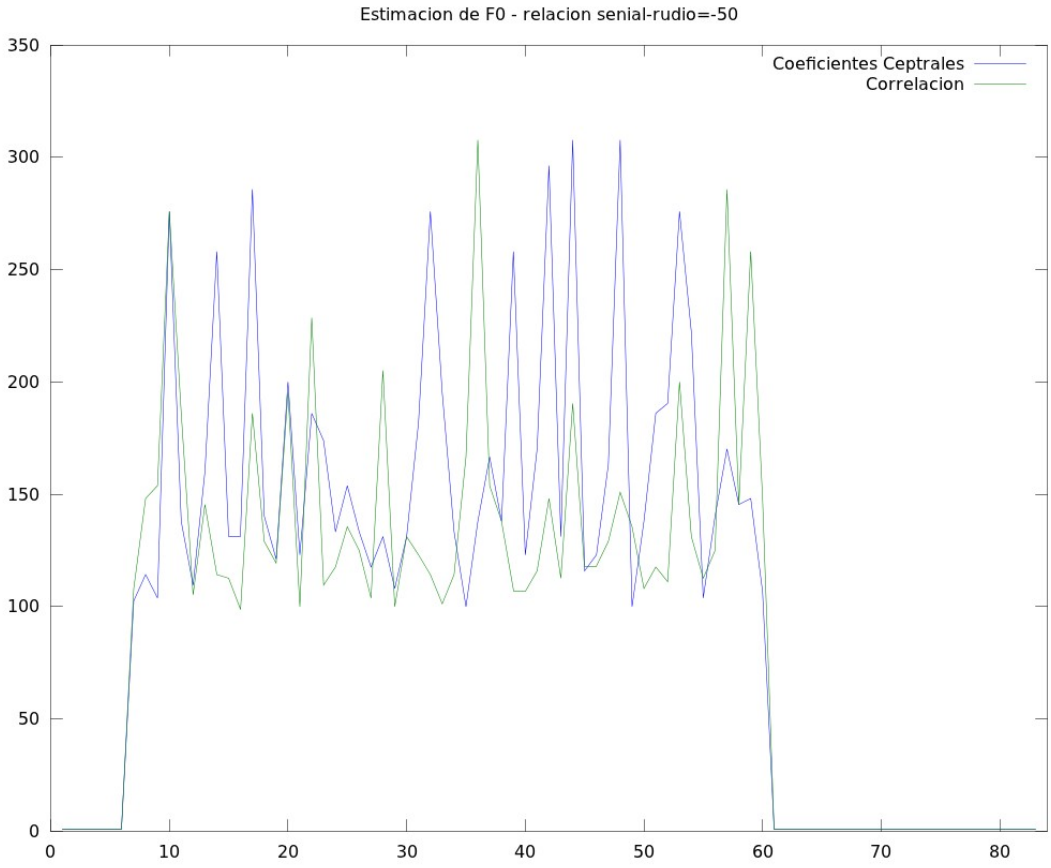
A continuación se pueden ver los resultados gráficamente:






Estimacion de F0 - relacion senial-rudio=-30
Coeficientes Ceptrales
Correlacion
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80


Estimacion de F0 - relacion senial-rudio=-40
Coeficientes Ceptrales
Correlacion
350
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80



Cuando las ventanas no alcanzaban un nivel mínimo de energía, no se las procesaba, y se definía F0 = 0. Por este motivo aparecen partes donde F0 es nula en las gráficas. Como era de esperarse, la técnica de coeficientes ceptrales es mucho más sensible al ruido que la de autocorrelación. Mientras que la segunda da resultados útiles hasta relaciones señal-ruido de -30db, la técnica de los coeficientes ceptrales a los 20db queda inutilizable.