ELO 314 - Procesamiento Digital de Señales Laboratorio 5: Análisis y Compresión de Voz en MatLab

Preparado por Juan Aguilera e-mail: Juan.Aguileraca@sansano.usm.cl
Cristóbal Huidobro e-mail: cristobal.huidobro@sansano.usm.cl

I. Predicción Lineal y Síntesis de vocales

1) Se genera la función siguiente.

```
function X = exciteV (N, Np)
  for i=1:1:N
    if mod(i-1,Np) == 0
      X(i)=1;
    else
      X(i)=0;
    end
  end
 end
```

La cual se aplica de la siguiente manera para generar la señal con frecuencia fundamental de 100 Hz.

```
fs=8000;
N=fs;
Np=80;
X = exciteV(N,Np);
```

La magnitud del espectro resultante se muestra en la Figura 1. Se observa que efectivamente, la magnitud corresponden a pulsos separados por una frecuencia de 100 sps. Al oír la señal se escucha un sonido con una frecuencia musical sostenida, levemente parecido a un instrumento de viento.

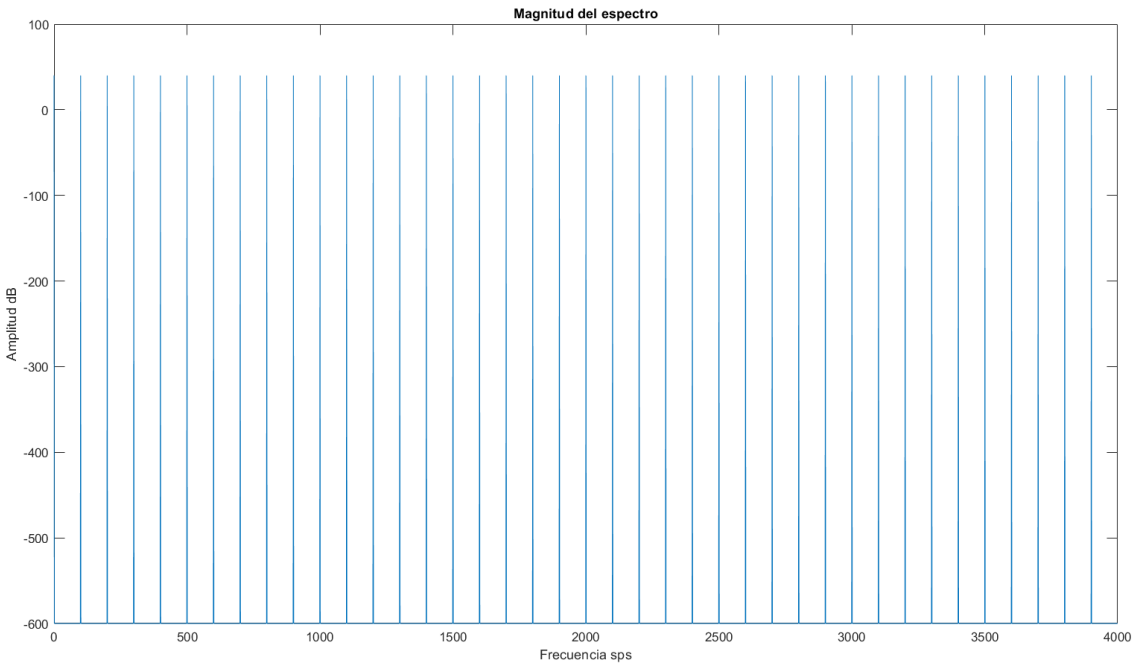

Figura 1: Magnitud del espectro de señal generada en dB.

2) Se genera cada filtro mediante el comando *lpc* según el siguiente código.

```
filtro_x = lpc(vowel_x,15);
X=0;
w = linspace(0,fs,N);
for i=1:1:16
    X=X+filtro_x(i).*exp(-1j*2*pi*w*(i-1));
end
```

donde x representa la vocal a elección, y X el nombre del filtro para dicha vocal. La magnitud del filtro para cada vocal se muestra en la Figura 2.
Se observa que cada vocal puede modelarse como un filtro con una respuesta en frecuencia específica.

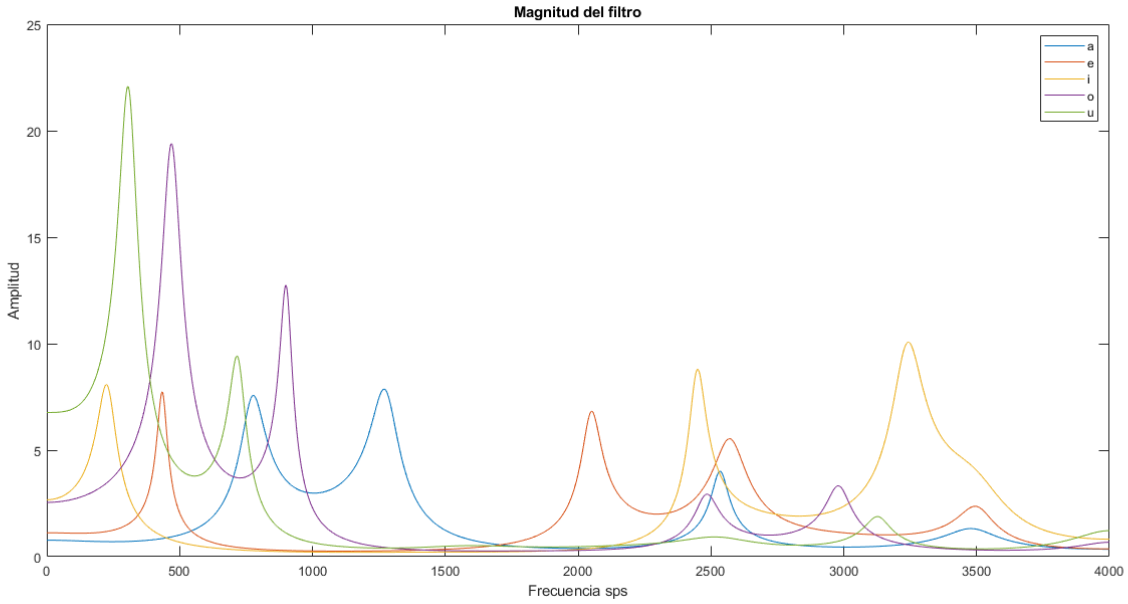

Figura 2: Magnitud del espectro del filtro para cada vocal.

3) Utilizando los filtros diseñados con el comando *lpc*, se generan vocales sintetizadas filtrando la señal de impulsos generada en 1).
El código utilizado para aquello es el siguiente:

```
    fs=8000;
    N=fs;
    Np=80;
    X = exciteV(N,Np);
    v = lpc(vowel_v,15);
    v_sint = filter(1,v,X);
```

Donde $v = a, e, i, o, u$ corresponde a la vocal a sintetizar. v_sint corresponde a la vocal sintetizada.
La magnitud en dB del espectro de las señales filtradas se muestra en la figura 3. Al oír las señales se identifican claramente las vocales, a pesar de que el timbre se la señal no corresponde exactamente al de la voz humana.

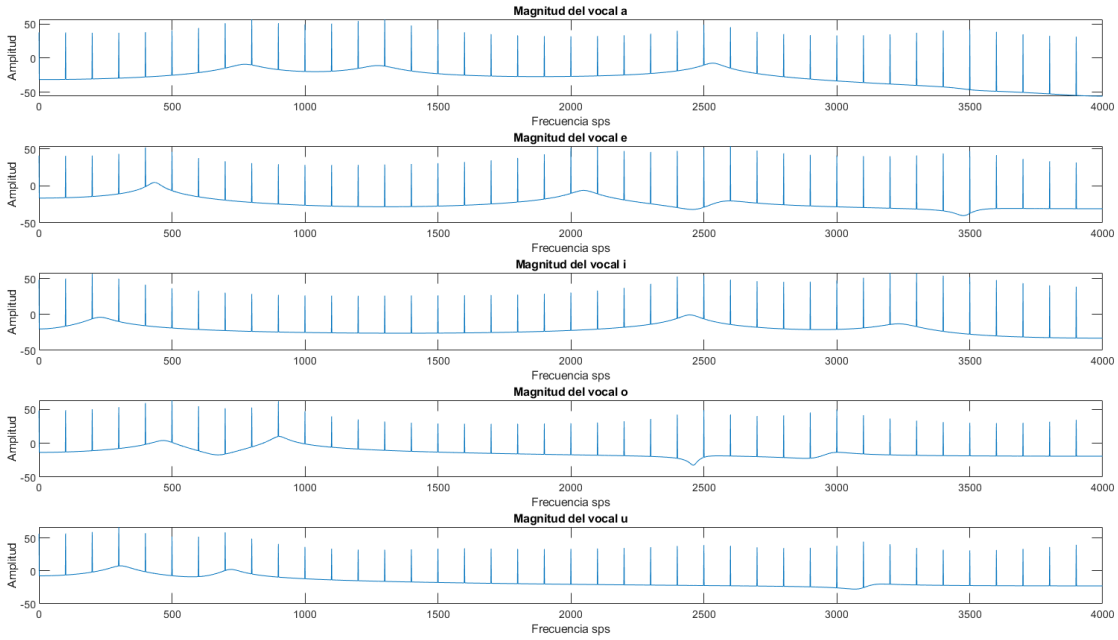

Figura 3: Magnitud del espectro de cada vocal en dB.

4) La función $mylpc$ diseñada se muestra a continuación. Cabe notar que se debe agregar el primer coeficiente (que es 1) a los parámetros obtenidos por la covarianza y la matriz de Toeplitz.

```
function a = mylpc(x,p)
rx = xcorr(x);
largo_rx = length(rx);
rx1 = rx(ceil(0.5*largo_rx)+1:ceil(0.5*largo_rx)+p);
rx2 = rx(ceil(0.5*largo_rx):ceil(0.5*largo_rx)+p-1);
Rx = toeplitz(rx2);
if  det(Rx) == 0
    a = zeros([1,16]);
else
   a = Rx^(-1)*rx1;
   a = [1; -a];
end
end
```

Los resultados obtenidos se observan en las figuras 4 y 5. Se observa que los filtros y las señales sintetizadas son iguales con el algoritmo de lpc de matlab y el algorítmo $mylpc$ diseñado. Corroborando el funcionamiento correcto de este último.

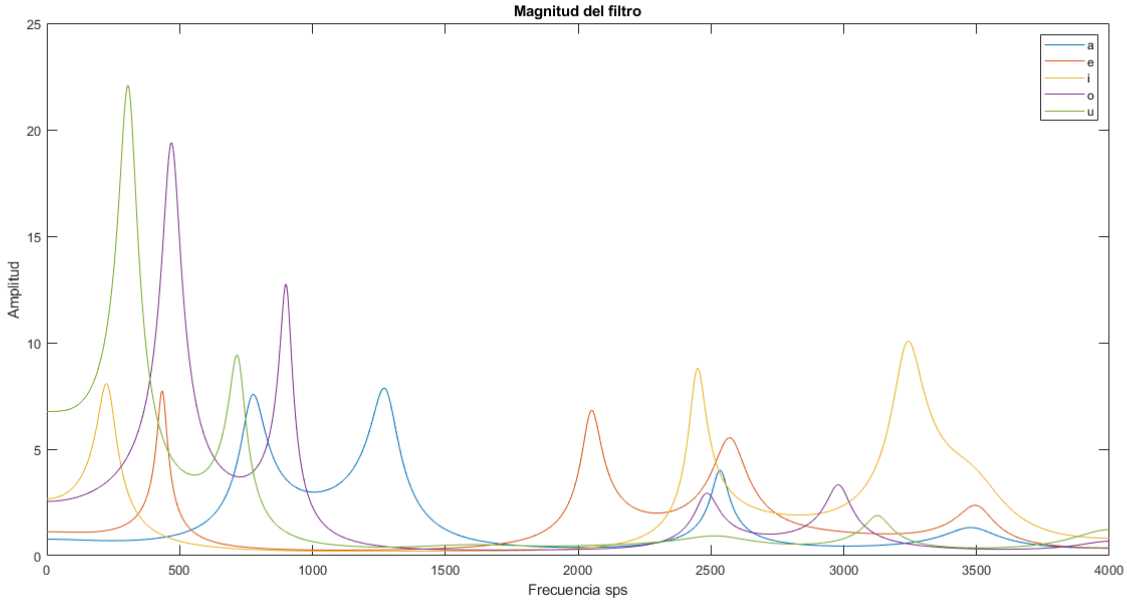

Figura 4: Magnitud del espectro del filtro para cada vocal.

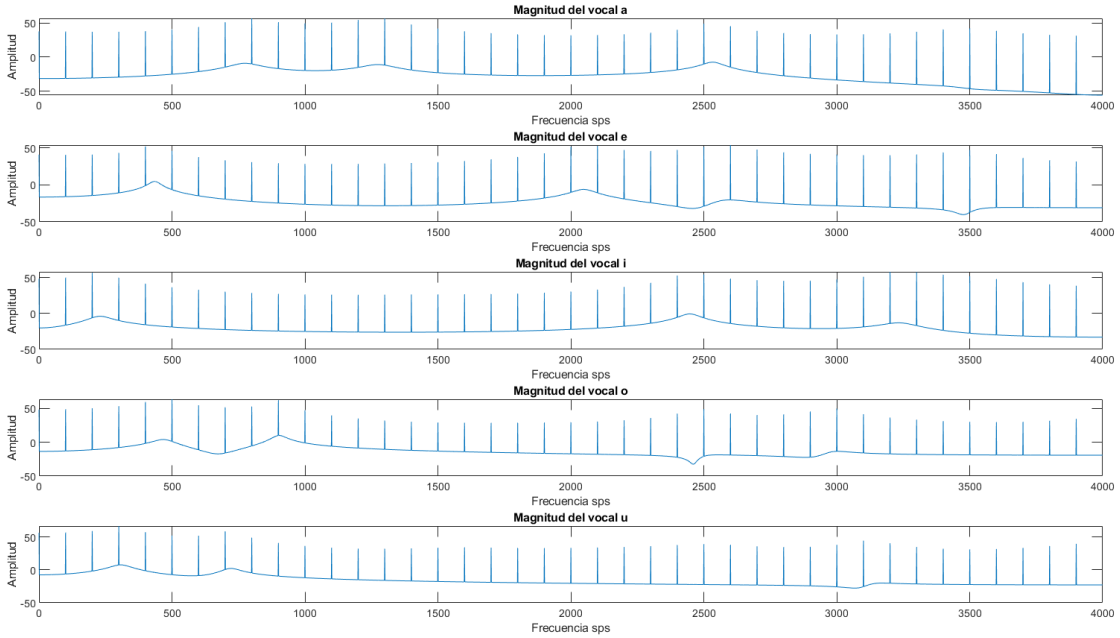

Figura 5: Magnitud del espectro de cada vocal en dB.

II. Clasificación de segmentos VUS

1) Se carga la señal *training_signal* para realizar el entrenamiento de clasificación de segmentos VUS.
   En primer lugar, se utiliza el comando *ginput* para seleccionar la señal en distintos fragmentos correspondientes a vocales, consonantes y silencio. El gráfico obtenido según aquello se muestra en la figura6.

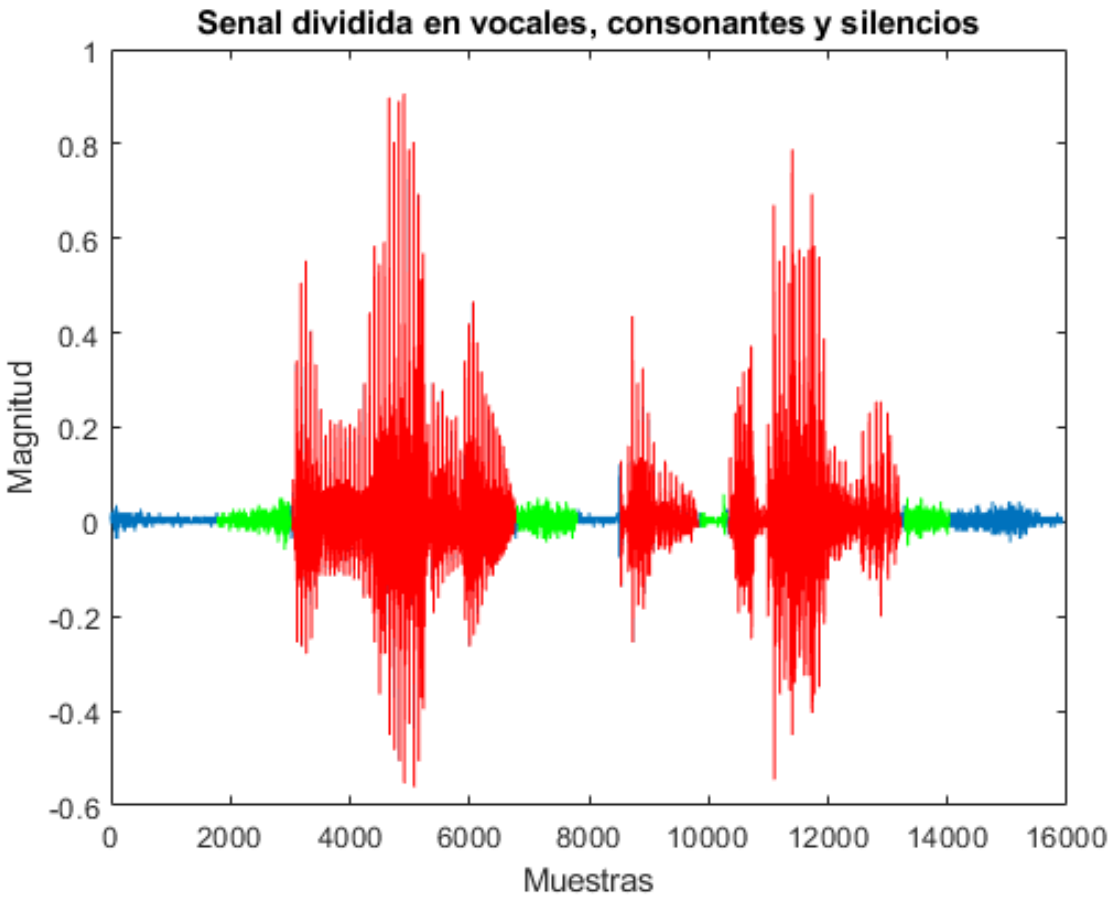

Figura 6: Señal dividida en silencio (azul), consonantes (verde) y vocales (rojo)

Para un análisis mas riguroso, se decide separar la señal en cada letra, calculando para cada una el valor RMS, y si cantidad de cruces por cero por segundo, la cual está dada por la siguiente función.

```
function y = cruces_cero(x)
largo_senal = length(x)/8000*1000;
y=0;
 for i=2:length(x)
   if (x(i)*x(i-1))<0
     y=y+1;
   end
 end
 y=y/largo_senal;
end
```

Se separan las letras en vocales, y se utiliza un *scatter plot* para obtener sus valores RMS vs sus cruces por cero por segundo. La gráfica obtenida se muestra en la figura 7.
Se observa que el valor RMS parecería ser un mejor criterio que la cantidad de cruces por cero para identificar si una señal es vocal, consonante o silencio. Sin embargo resulta más conveniente utilizar una combinación de ambos criterios para decidir si la señal es vocal o consonante.

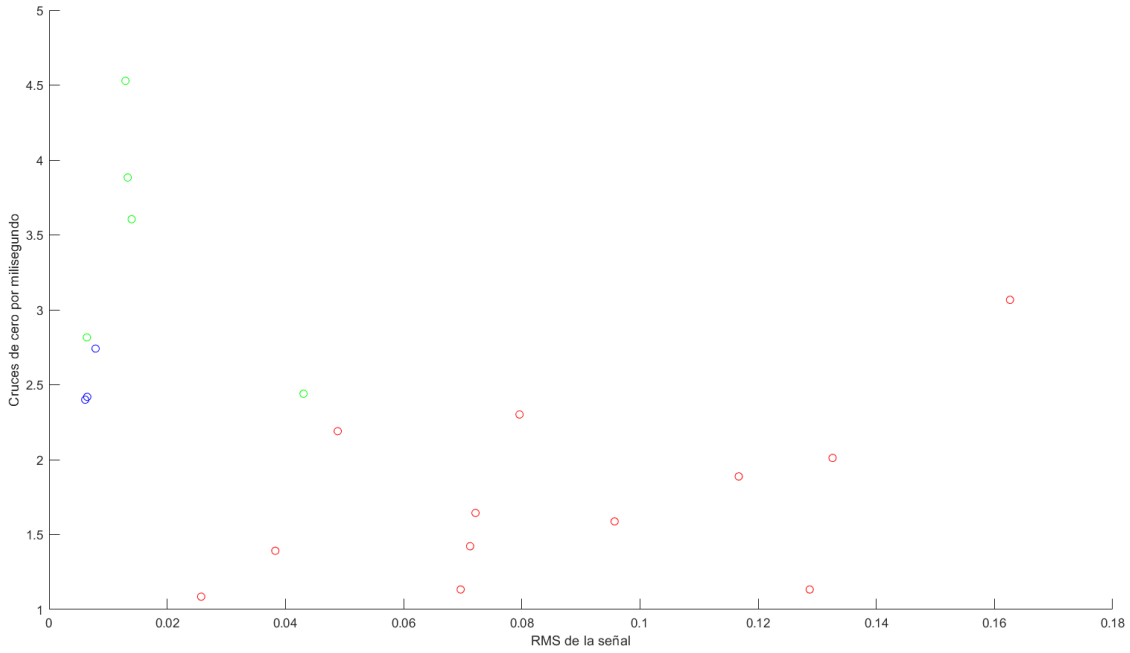

Figura 7: RMS y cruces de la señal.

2) A partir de los resultados del gráfico del punto anterior, se utiliza el criterio descrito en la siguiente función para clasificar si la señal es vocal, consonante o silencio. Donde $y = 0$ representa los frames donde hay silencio, $y = -1$ representa los frames donde hay una consonante y $y = 1$ representa los frames donde hay una vocal.

```
function y = find_VUS(x)
if(rms(x)<0.02)
    y = 0;
else
    if(cruces_cero(x)>2.4 && rms(x)<0.06)
        y = -1;
    else
        y = 1;
    end
end
end
```

Se calcula el valor VUS de cada frame de la señal de prueba y se almacena en un vector. Se grafica la señal junto a la variable VUS en la figura 8.
A simple vista los valores obtenidos parecen ser consistentes con los sonidos vocales (secciones donde se identifica una señal periódica), consonantes (secciones más similares a ruido blanco) y silencios (secciones donde no hay nada) en la señal.

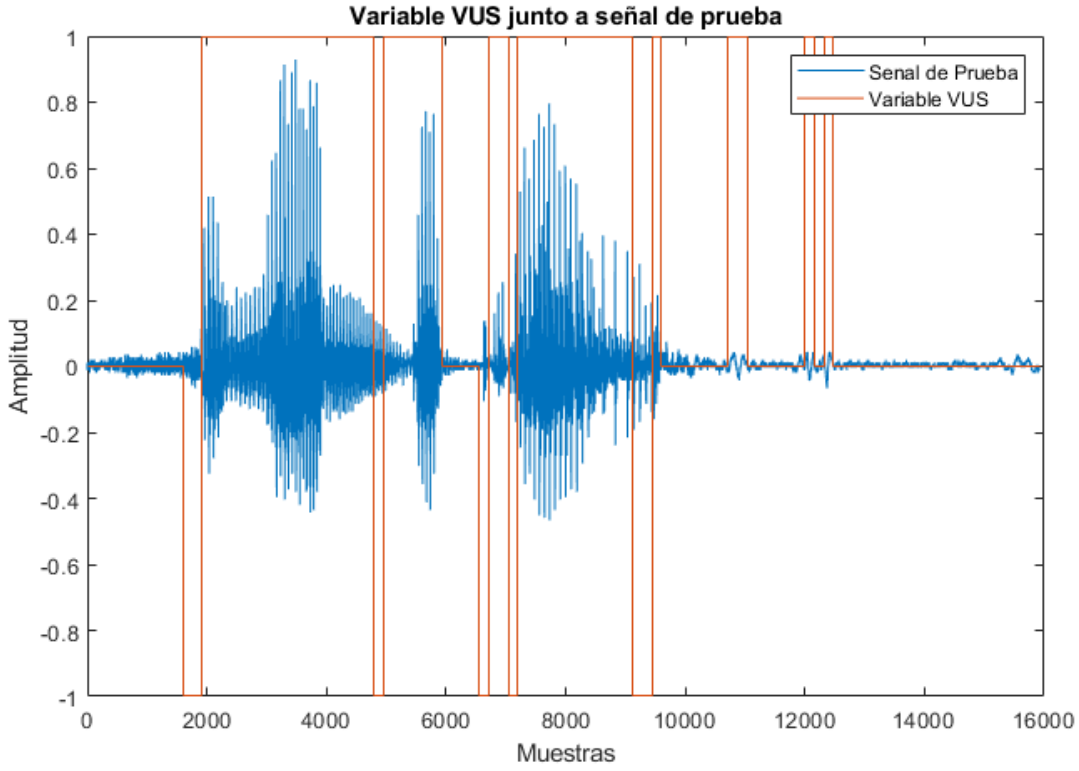

Figura 8: Valor VUS obtenido para cada frame de la señal

III. Síntesis de voz hablada

1) A partir de la señal a comprimir se genera un vector con los coeficientes VUS de cada frame, un vector con el valor RMS de cada frame y una matriz con los coeficientes LPC de cada frame

```
for i=1:100
    i2 = 0.02*fs*(i-1)+1;
    frame = test_signal(i2:i2+min(0.02*fs-1,length(test_signal)-i2));
    VUS_vector(i)=find_VUS(frame);
    RMS_vector(i)=rms(frame);
    lpc_matrix(:,i) = mylpc(frame,15);
end
```

Esos valores son los datos que el emisor debe enviar al receptor en lugar de la señal completa.

Luego, se genera la señal sintetizada a partir de los vectores (que en caso de comunicación real son los datos que debieron haber llegado al receptor). Donde la señal a filtrar depende del flag VUS correspondiente a cada frame.

```
senal_sintetizada = zeros([1,length(test_signal)]);
for j=1:100
    j2 = 0.02*fs*(j-1)+1;
    if VUS_vector(j)==0
        senal_sintetizada(j2:j2+min(0.02*fs-1,length(test_signal)-j2-1)) =
                    zeros(1,min(0.02*fs,length(test_signal)-j2));
    else
        if VUS_vector(j)==1
            X = exciteV(min(0.02*fs,length(test_signal)-j2),80);
            rmsfactor = RMS_vector(i)/rms(filter(1,lpc_matrix(:,j),X));
            senal_sintetizada(j2:j2+min(0.02*fs-1,length(test_signal)-j2-1)) =
                        filter(1,lpc_matrix(:,j),X)*rmsfactor;
        else
            X = rand([1,min(0.02*fs,length(test_signal)-j2)]);
            rmsfactor = RMS_vector(i)/rms(filter(1,lpc_matrix(:,j),X));
            senal_sintetizada(j2:j2+min(0.02*fs-1,length(test_signal)-j2-1)) =
                        filter(1,lpc_matrix(:,j),X)*rmsfactor;
        end
```

```
    end
end
```

La señal sintetizada obtenida junto a la señal original se muestran en la Figura 9.
Al escuchar la señal creada, se identifica correctamente la frase del audio original, y el tono es relativamente similar a la voz humana, aunque sigue siendo apreciable que es una señal generada por computador.

Para mejorar la síntesis, se podrían considerar los cambios de frecuencia que tiene la voz humana al hablar, agregar mayor orden al filtro (aunque eso implica enviar más datos por el canal), además de quizás cambiar la señal de base para el filtrado al residuo calculado con el filtro inverso de voces reales.

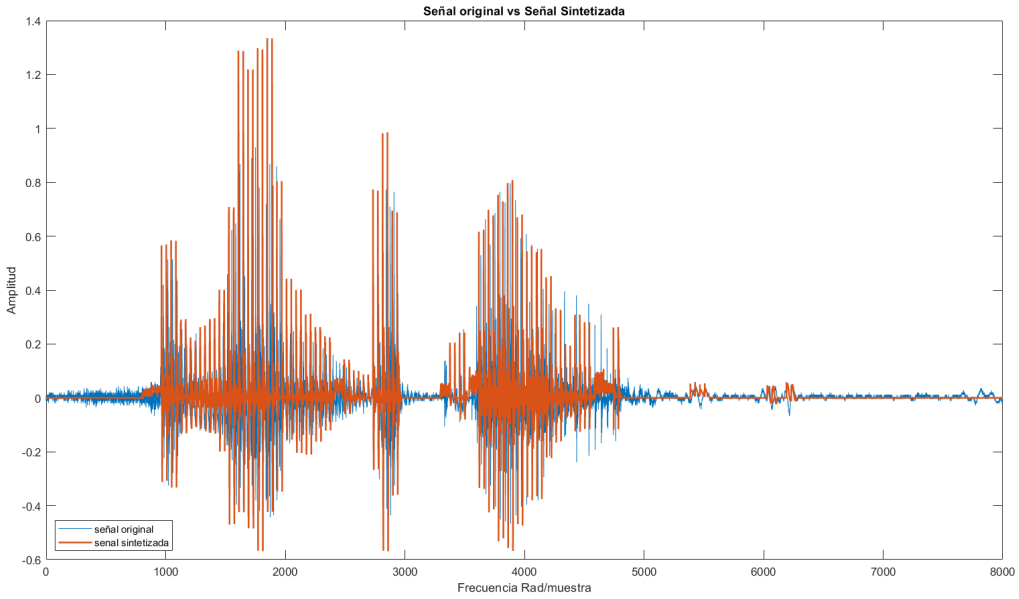

Figura 9: Señal de prueba vs señal sintetizada.

2) La cantidad de bytes necesarios para almacenar la señal original es de 127360.
La cantidad de bytes necesarios para almacenar el vector de flags VUS es 800, la cantidad para el vector de valores RMS es 800 y la cantidad para la matriz de coeficientes lpc es 12800. Por lo tanto los bytes necesarios para almacenar la señal comprimida son 14400.
Por lo tanto la razón de compresión es de 8,84.

IV. Filtro Autoregrecivo

1) La función *positiveSpectrum* se muestra a continuación.

```
function Y = positiveSpectrum(X)
Spectrum = abs(fft(X));
Y = Spectrum(1:floor(length(X)/2)+1);
end

function X = exciteV (N, Np)
  for i=1:1:N
    if mod(i-1,Np) == 0
      X(i)=1;
    else
      X(i)=0;
    end
  end
 end
```

El espectro de la señal $Vowel\_a$ se muestra en la Figura 10, donde se reconocen aproximadamente 5 formantes.

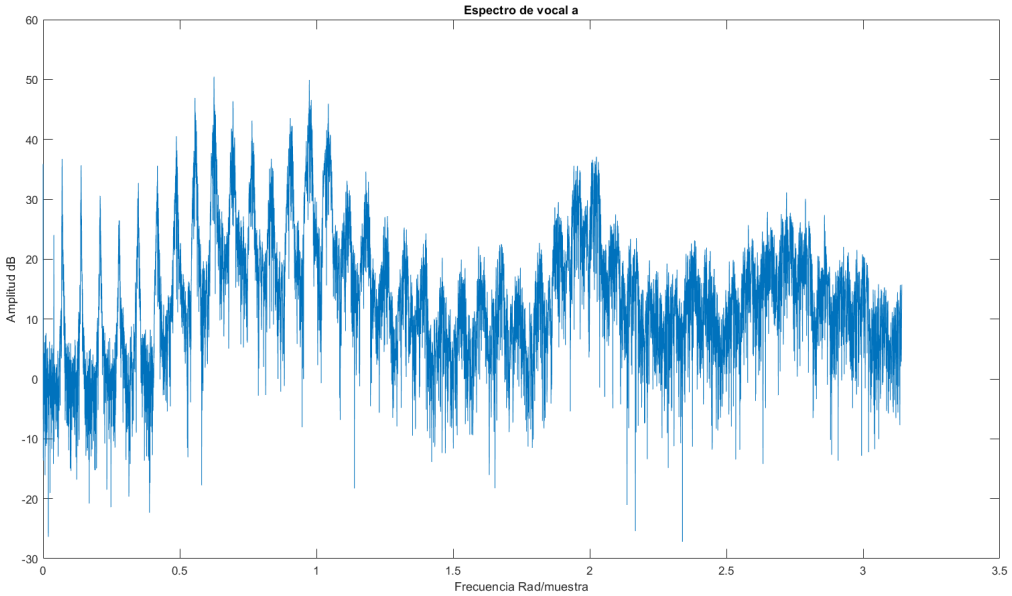

Figura 10: Espectro de la Vocal A.

2) Se probarón filtros lpc de orden 2 y 15 para las vocales a y u.
   Los filtros probados se muestran junto al espectro de las señales en las Figuras 11 y 12.

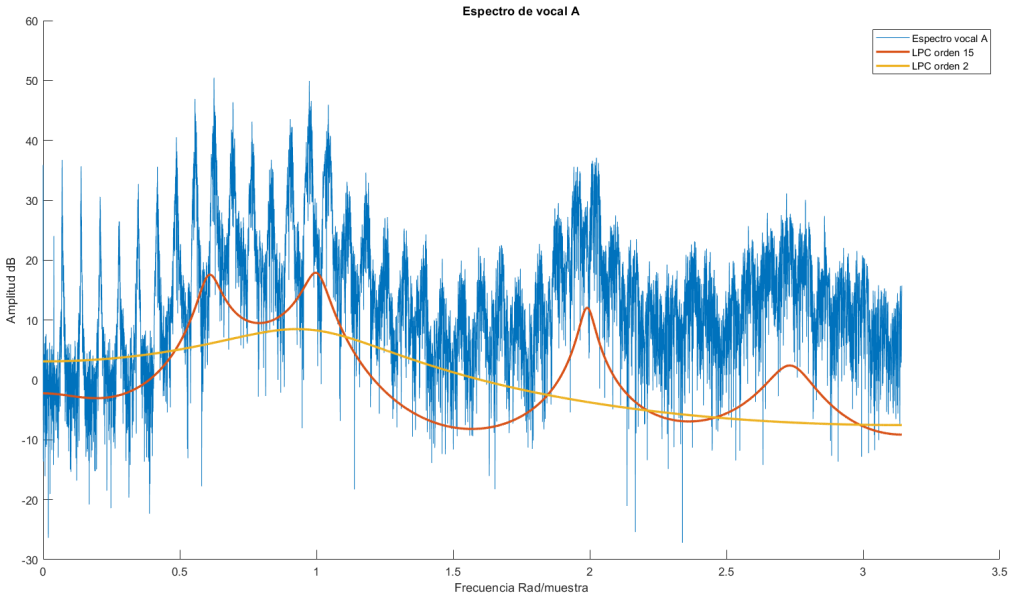

Figura 11: Espectro y LPC de vocal A.

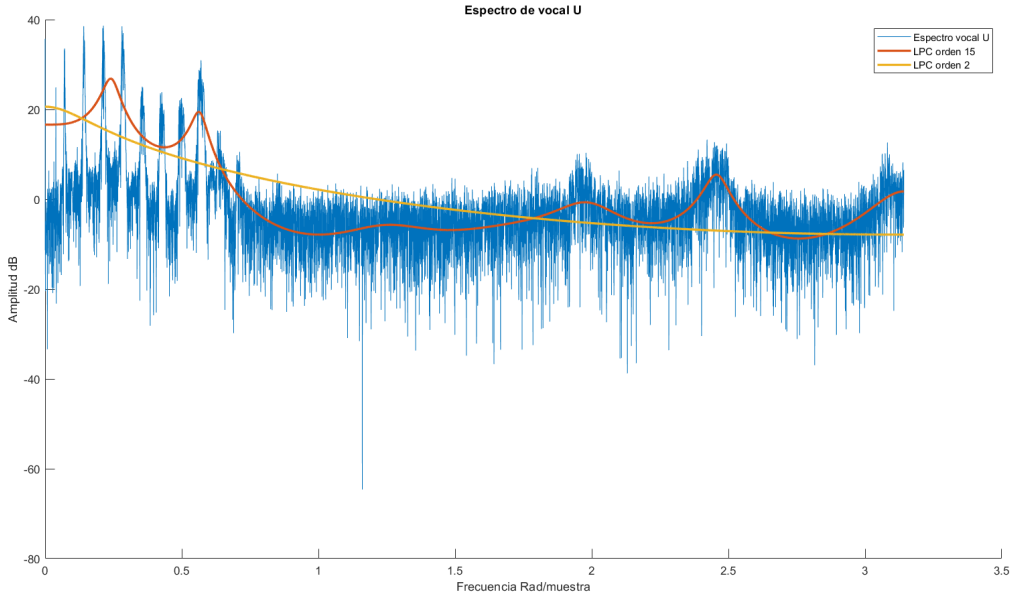

Figura 12: Espectro y LPC de vocal U.

En la Figura 13 se muestra el plano Z para los filtros de distintos ordenes.

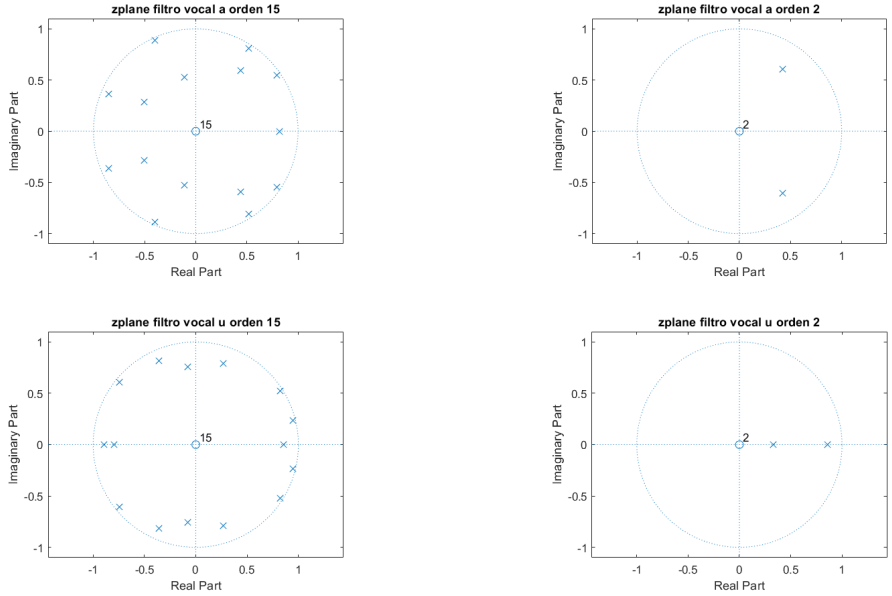

Figura 13: Planos Z de vocales A y U para distintos ordenes.

La respuesta en frecuencia obtenida por el algoritmo *lpc* de orden 2 no alcanza a imitar el comportamiento de las vocales, ya que estas tienen una cantidad de formantes mayores a aquel orden. Se observa que en el caso de la vocal *a*, los polos de esta aproximación son complejos conjugados, mientras que para la vocal *u* son dos polos reales.

3) La comparación de los espectros de la vocal *a* original y sintetizada con ordenes 9 y 15 se muestra en la Figura 14. El plano Z para dichos filtros en la Figura 15.
No se aprecian diferencias significativas en ambos casos ya que el orden 9 es suficiente para la cantidad de formantes del espectro de la vocal *a*.

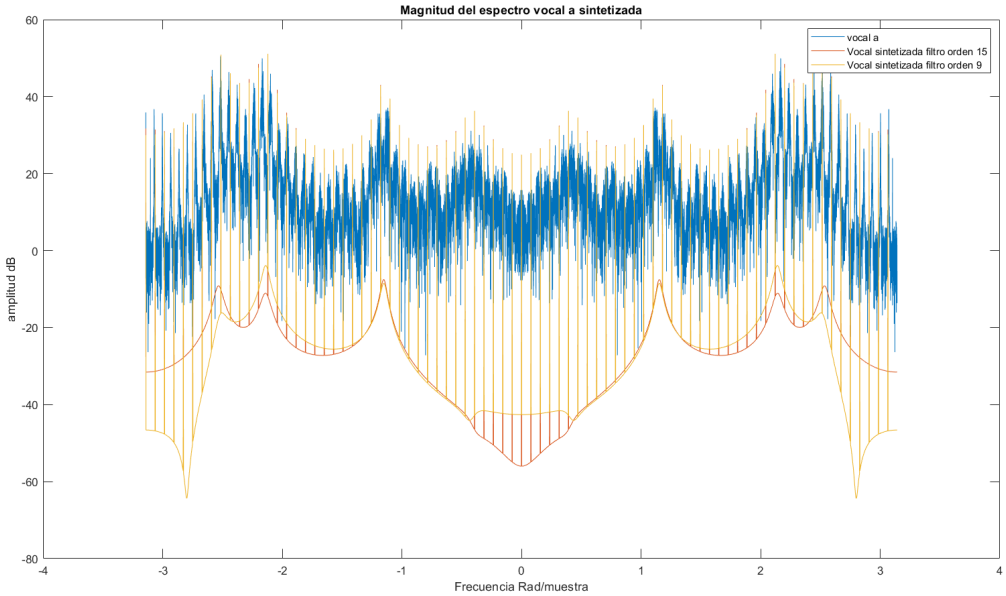

Figura 14: Magnitud de la vocal ’a’ origínenla y sintetizada.

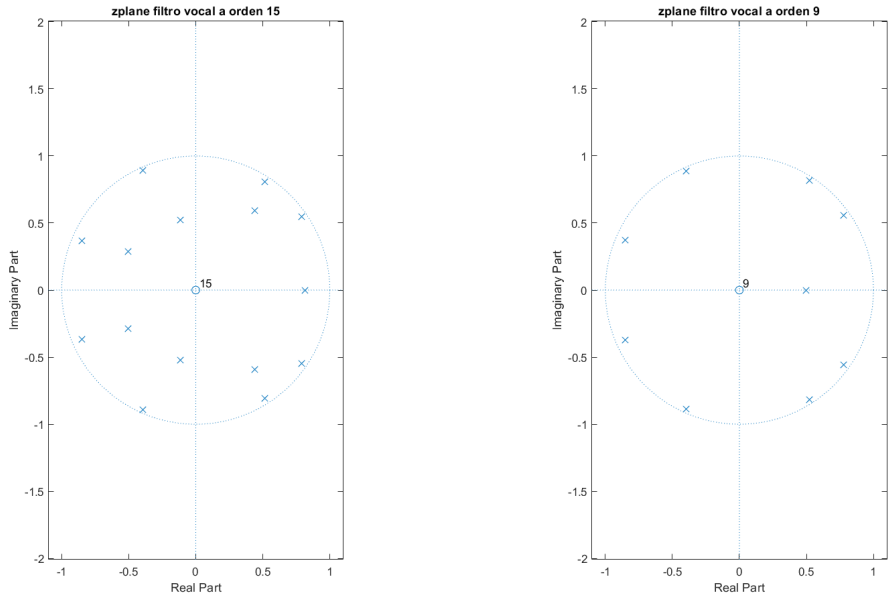

Figura 15: Planos Z de vocal ’a’ para distintos ordenes de filtro.

Para la vocal *u*, el mínimo orden del filtro para el cual no se aprecia diferencia respecto al filtro de orden 15, es 12. La comparación de los espectros de la vocal original, y la vocal sintetizada con ambos ordenes se muestran en la Figura 16, y el plano Z para dichos filtros en la Figura 17.

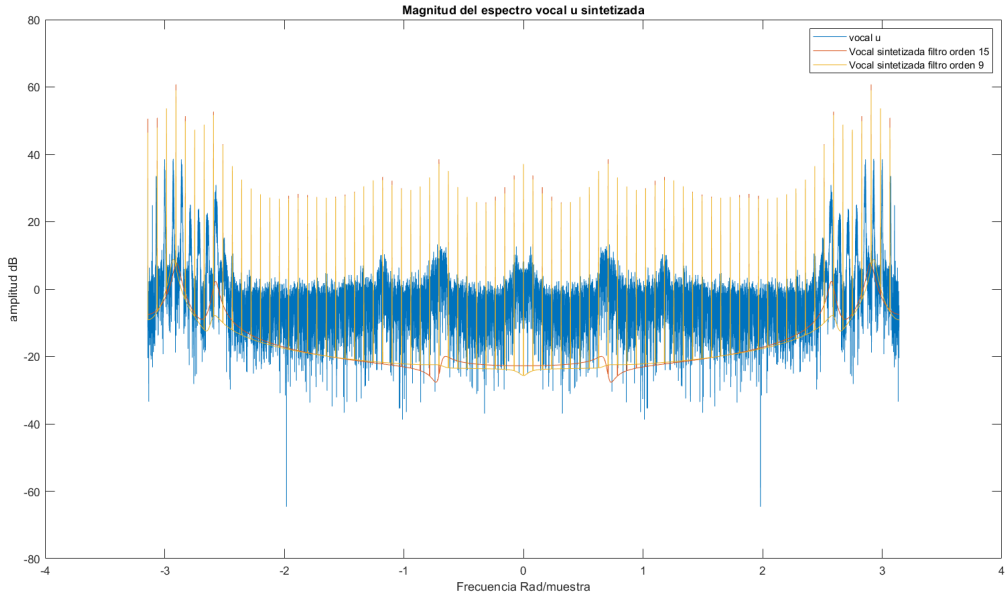

Figura 16: Magnitud de la vocal 'u' original y sintetizada.

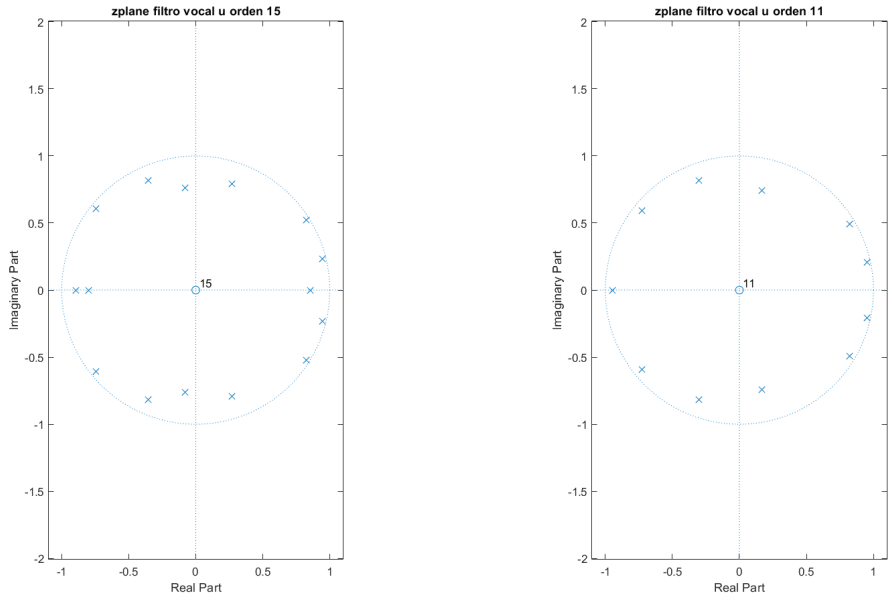

Figura 17: Planos Z de vocal 'u' para distintos ordenes.

Una estrategia para determinar el orden de un filtro AR para LPC es obtener el espectro positivo de la señal a sintetizar. Observar el número n de formantes (sin incluir la frecuencia 0). Multiplicar el número por 2 para obtener n pares de polos complejos conjugados, y agregar un polo en 0. Por lo tanto el orden debe ser de $2n + 1$.

4) ▪ El comando `tfsos` de matlab obtiene los coeficientes de los $n/2$ filtros biquad que al conectarse en cascada, entregan la función de transferencia de orden n ingresada. En el caso que n sea impar, el comando entregará además el filtro de orden 1 restante.
Las filas de la matriz `sos` corresponden a cada filtro biquad cuyos coeficientes están dados por las columnas. Por lo tanto la matriz siempre tendrá 5 columnas mientras que el número de filas depende del orden de la función de transferencia ingresada.

La variable entregada `g` corresponde a la ganancia por la cual multiplicar los filtros biquad para obtener la transferencia deseada.

- Se utilizó el comando `tfsos` para obtener los filtros biquad que obtienen el filtro lpc de orden 15 para la vocal *a* en cascada. En este caso se utilizarán los 7 filtros biquad, ignorando el filtro de orden 1 restante.
  La magnitud de la vocal 'a' con los diferentes filtros biquad se presenta en la Figura 18. Donde se observa que cada filtro tiene polos complejos conjugados correspondientes a la frecuencia de una de las formandes de la vocal.

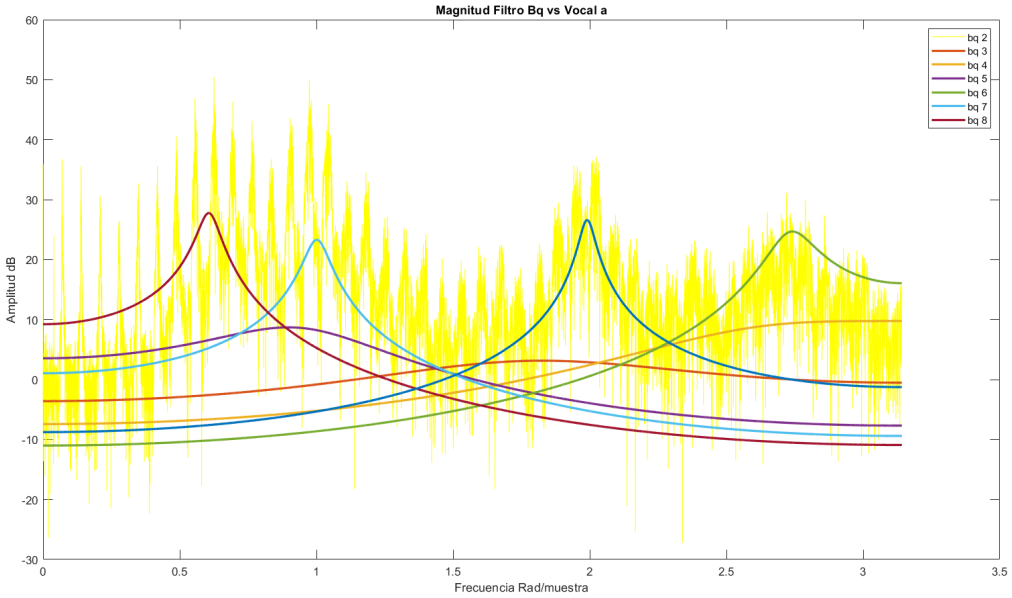

Figura 18: Magnitud de la vocal 'a' y diferentes filtros bq.

V. Observando el residuo de predicción

1) La respuesta en frecuencia del filtro inverso vs el filtro AR de orden 15 se muestra en la Figura 19.

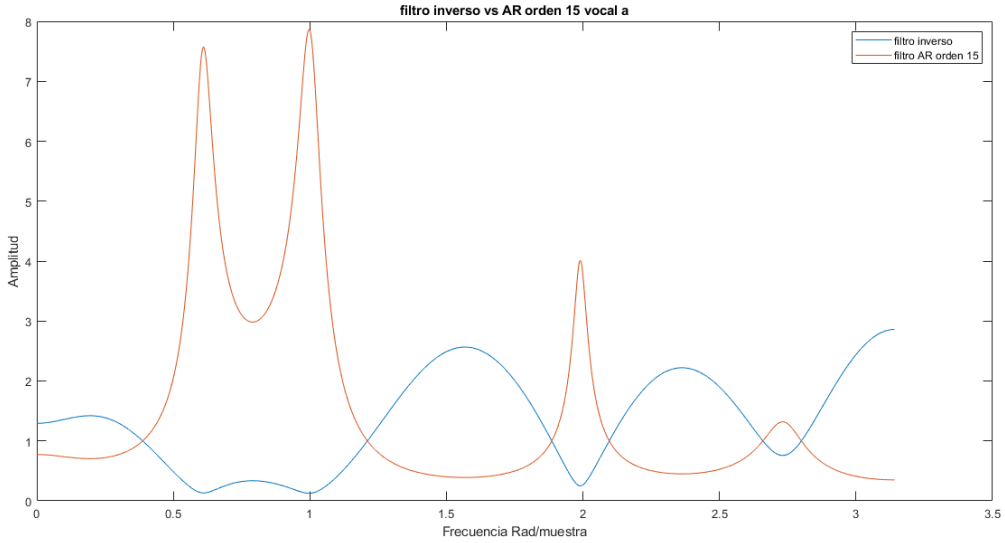

Figura 19: Filtro AR vd filtro inverso.

2) Se obtiene el residuo luego de aplicar el filtro inverso a la vocal *a*.
   El gráfico del residuo se presenta en Figura 20. Se observa que este se asemeja bastante a un tren de impulsos, por lo que este resulta una aproximación adecuada. Sin embargo la señal no es exactamente igual a un tren, habiendo otros componentes entre los peaks, cuya inclusión podría mejorar la verosimilitud de las señales sintetizadas..

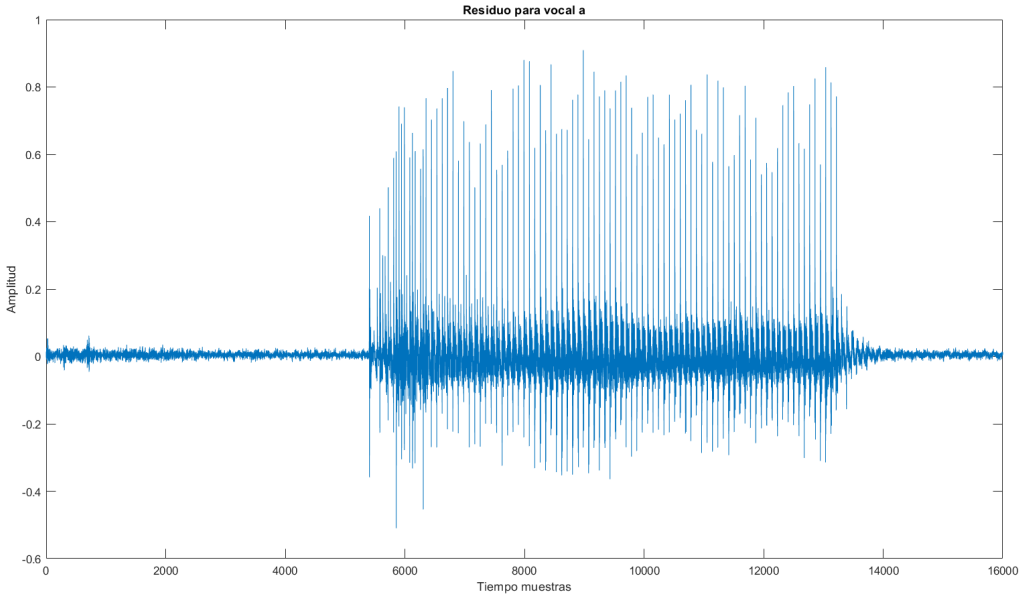

Figura 20: Residuo de la señal vocal 'a'.

3) El espectro en frecuencia del residuo de la vocal 'a' y de la señal vocal 'a' se presentan en Figura 21. Se observa que la mayoría del componente armónico de la señal es eliminada por el filtro inverso al obtener el residuo. Sin embargo, el espectro del residuo no es completamente plano, habiendo componentes mayores en frecuencias más bajas.
Se observa también que el ruido es más significativo en el residuo que en la vocal.

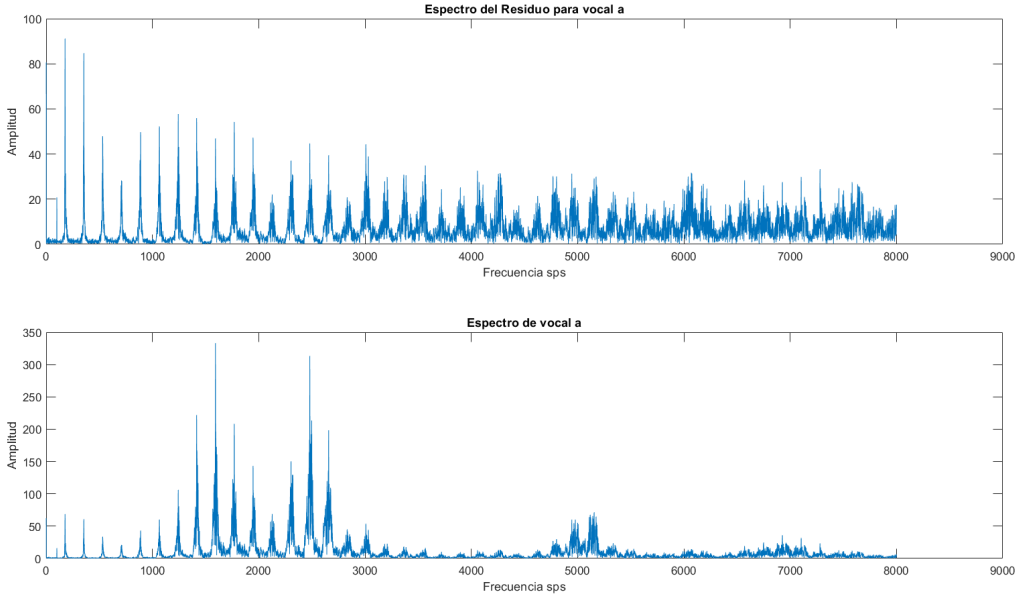

Figura 21: Residuo de la señal vocal 'a'.

4) La autocorrelación del residuo y de la vocal se muestran en la Figura 22.
Se observa que la frecuencia fundamental es obtenida muy facilmente de la autocorrelación del residuo de la señal.
En particular, para señales que no son solamente una vocal sostenida, los componentes entre frecuencias puede hacer confusa la obtención de las frecuencias, estos prácticamente desaparecen al calcular la correlación del residuo, por lo que el uso de esta simplificaría mucho más la obtención de la fundamental.

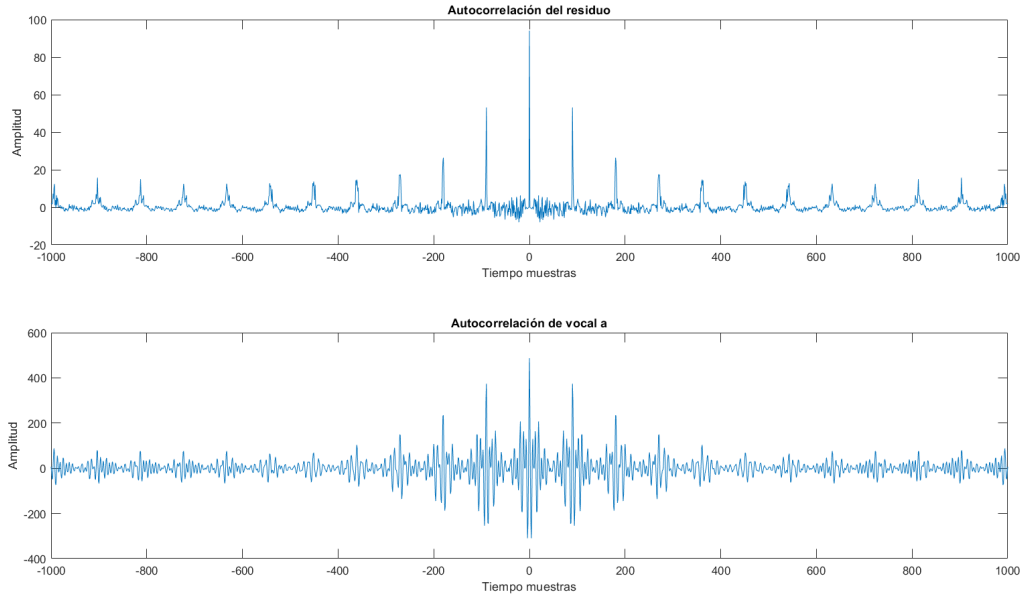

Figura 22: Autocorrelación entre residuo y vocal.