Universidad Técnica Federico Santa María

DEPARTAMENTO DE INGENIERÍA ELECTRONICA

ELO 314 - Laboratorio de procesamiento Digital de Señales

Análisis y compresión de voz en MatLab

Estudiante	ROL
Rodrigo Graves	201621009-1
Ricardo Mardones	201621036-9

Paralelo: 1

Profesor

Gonzalo Carrasco

Ayudante

Jaime Guzmán

Fecha : 23 de Julio de 2021

Índice

Índice de figuras

Índice de cuadros

1. Predicción lineal y síntesis de vocales

1.1. Generación de Tren de Impulsos

Se crea la función $X = exciteV(N, Np)$ que representa el sonido de las cuerdas vocales mediante un tren de impulsos, donde N es el largo de la señal en muestras y Np el período en muestras. El código de la función generada se muestra a continuación:

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

Posteriormente se genera un tren de impulsos de 100 Hz de duración 1 s muestreada a 8 kHz, para luego escucharla y graficar su espectro en frecuencia. Lo anterior usando el script `p1_1.m`, adjunto a la entrega.

La sección de código donde se genera la señal de tren de impulsos corresponde a:

```
fs = 8000;
t =1;
f = 100;
T = fs/f;
y = exciteV(t*fs,T);
Y =   fft(y);
Ymod = abs(Y);
Ydb = 20*log10( 0.000000001 + Ymod(1:fs/2));
plot([0:fs/2-1],  Ydb(1:fs/2));
```

El espectro obtenido para la señal generada se muestra en la figura 1

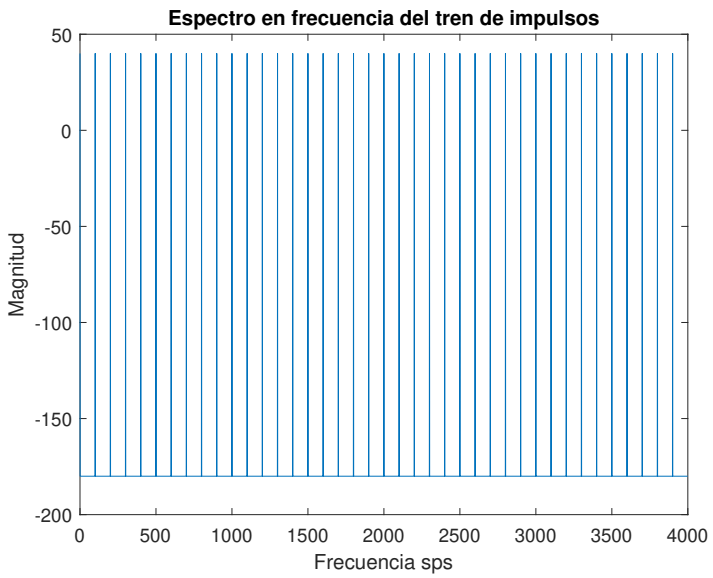

Figura 1: Espectro del tren de impulsos generada con la función $exciteV$

1.2. Diseño de filtros AR óptimos usando $y = lpc(x, p)$

Para esta parte se utilizó el código `p1_2.m`, adjunto a la entrega.

La sección de código donde se diseñan los filtros que simulan el comportamiento del tracto vocal para cada vocal se muestra a continuación:

```
load vowels

% Diseno de filtros
a_vowel_a = lpc(vowel_a,P);
a_vowel_e = lpc(vowel_e,P);
a_vowel_i = lpc(vowel_i,P);
a_vowel_o = lpc(vowel_o,P);
a_vowel_u = lpc(vowel_u,P);
```

donde $P = 15$, el cual corresponde al orden del filtro que se desea generar.

Las respuestas en frecuencia de cada vocal se muestran en las figuras 2 - 6, donde además se destaca el primer y segundo formante.

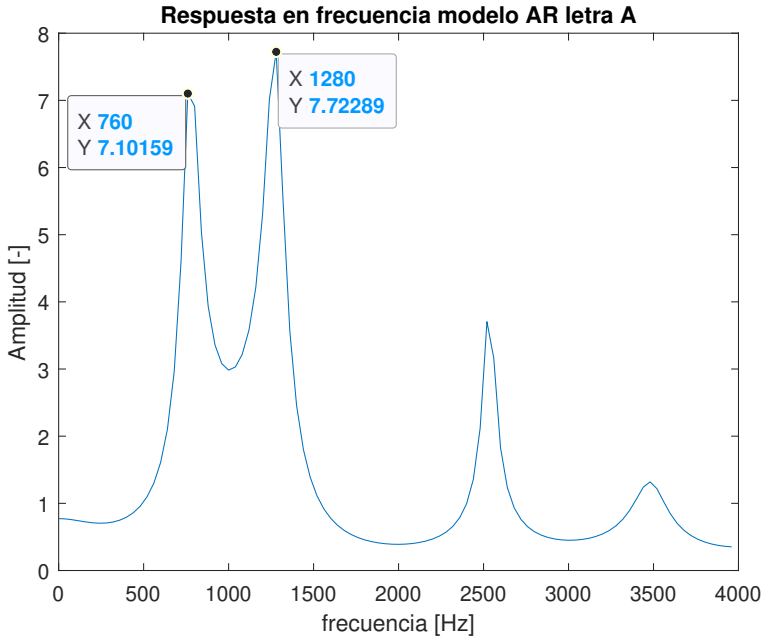

Figura 2: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra a. Se destaca el primer y segundo formante encontrado.

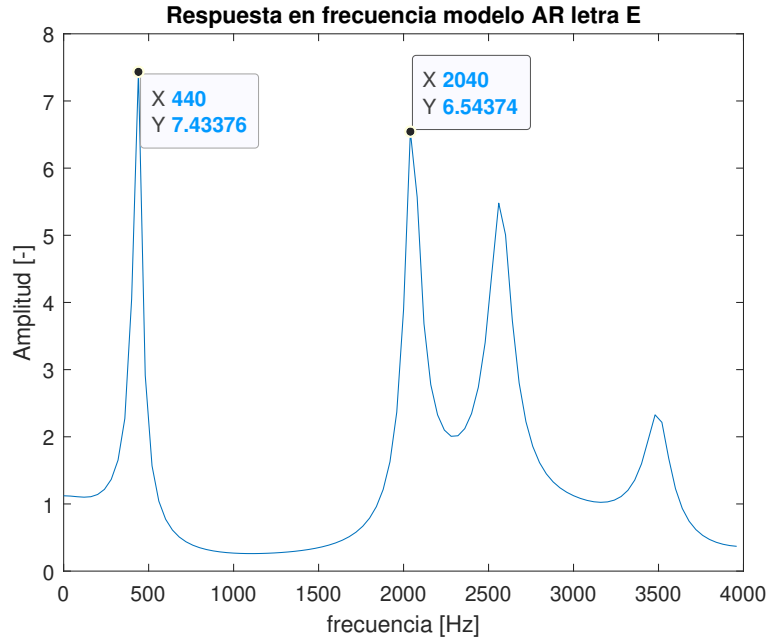

Figura 3: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra e. Se destaca el primer y segundo formante encontrado.

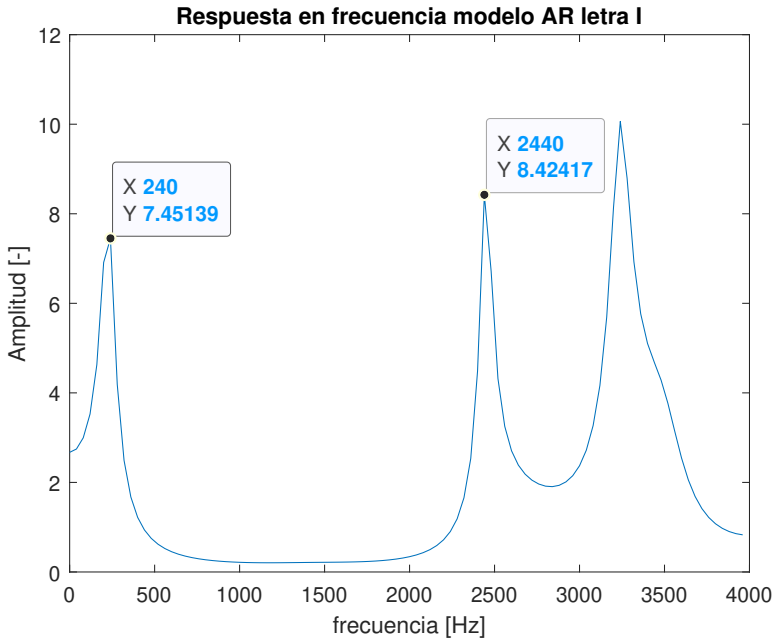

Figura 4: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra i. Se destaca el primer y segundo formante encontrado.

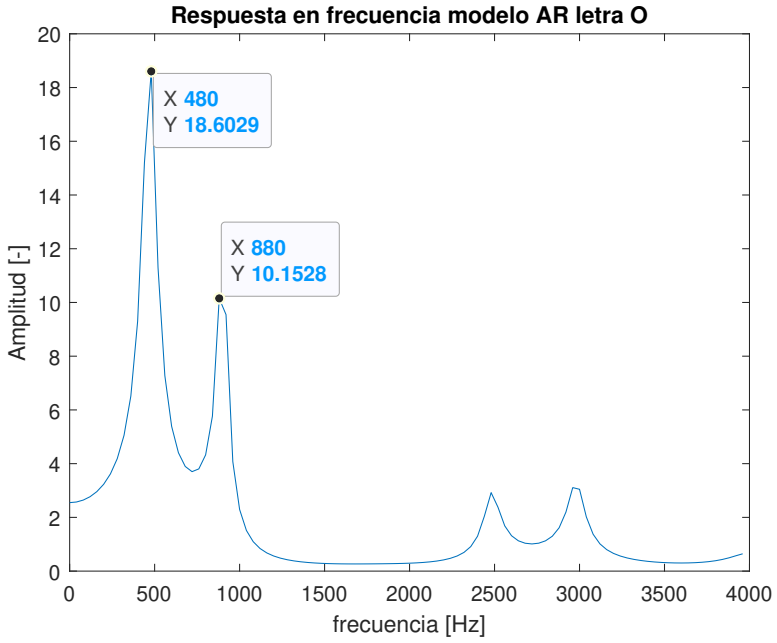

Figura 5: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra o. Se destaca el primer y segundo formante encontrado.

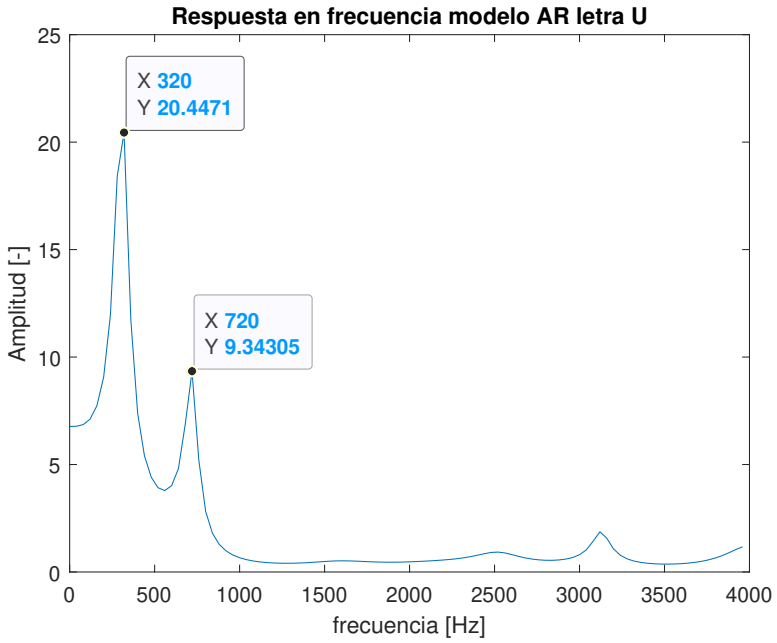

Figura 6: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra u. Se destaca el primer y segundo formante encontrado.

De las figuras de las respuestas en frecuencia del tracto vocal por cada vocal se aprecia que los formantes de cada vocal están donde corresponden según el gráfico de formantes mostrado en la figura 7. Por otro lado en algunas vocales se aprecia un menor número de peaks, por lo que un valor menor de P dependiendo la vocal podría ser suficiente (vocales i y u).

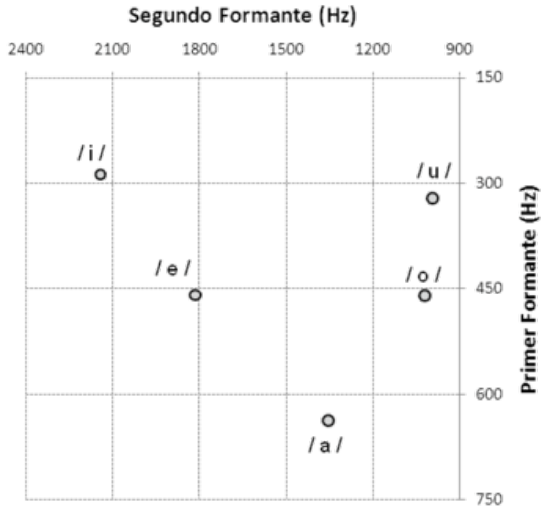

Figura 7: Gráfico de formantes de vocales en habla hispana.

1.3. Uso de filtros diseñados para síntesis de vocales

Para esta sección se utiliza el código `p1_3.m`, el cual se adjunta a la entrega.

Se utilizan los filtros AR diseñados en el punto anterior junto a la señal de tren impulsos generada en el parte 1.1 como entrada a los filtros para sintetizar archivos de audio.

La sección de código donde se genera la señal discreta que imita a un muestreo de una señal de audio de cada vocal se muestra a continuación:

```
%%% 3. Generacion de archivos de audio (filtrado)
y_a = filter(1, a_vowel_a, x);
y_e = filter(1, a_vowel_e, x);
y_i = filter(1, a_vowel_i, x);
y_o = filter(1, a_vowel_o, x);
y_u = filter(1, a_vowel_u, x);
```

Con respecto a los archivos de audio generados, si es posible distinguir que vocal corresponde a cada una. Sobre la calidad de audio, todos los sonidos tienen un tono "robótico", siendo la letra u la que tiene menos calidad, al menos desde el parlante disponible al momento de escucharla. Las vocales se adjunta al informe en el formato `matlab_vowel_x` siendo `x` la vocal respectiva.

Para mejorar la calidad del audio se podría filtrar la señal generada con el sistema actual por un filtro pasa bajos a diseñar mediante algún criterio. Lo anterior se plantea debido a que se pueden percibir frecuencias altas en el audio que no son naturales en la voz humana a esa potencia.

Los gráficos de los espectros en frecuencia de cada vocal sintetizada se muestran en las figuras 8 - 12. Con respecto a las formas obtenidas, como era de esperarse, corresponden a un tren de impulsos con envolvente la respuesta en frecuencia del filtro de la vocal respectiva.

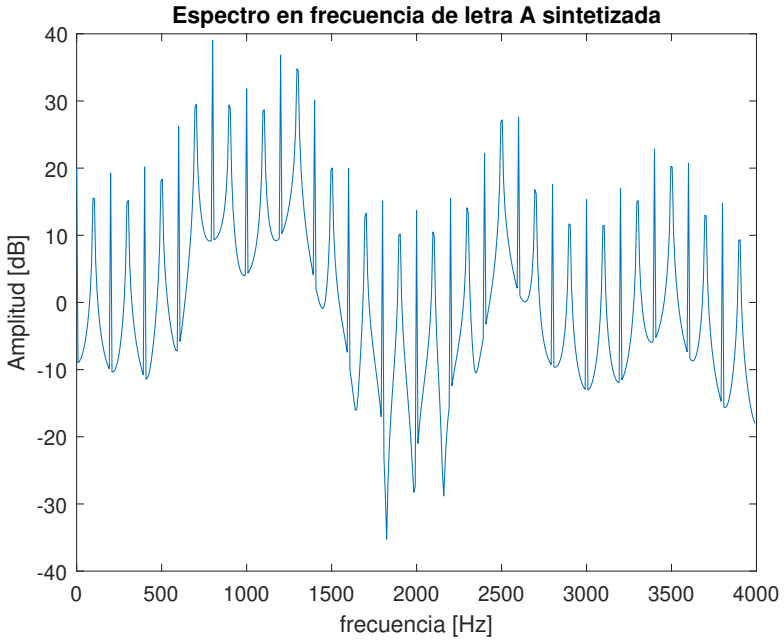

Figura 8: Espectro en frecuencia de letra A sintetizada.

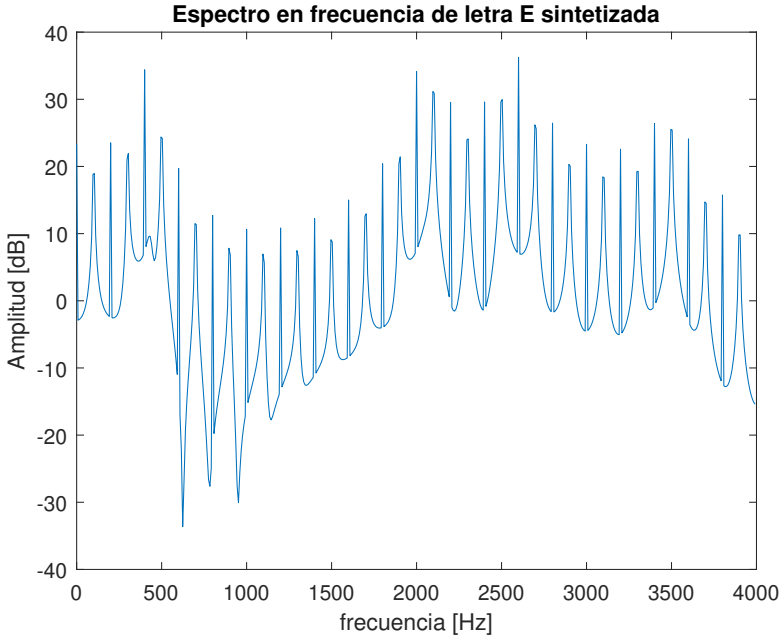

Figura 9: Espectro en frecuencia de letra E sintetizada.

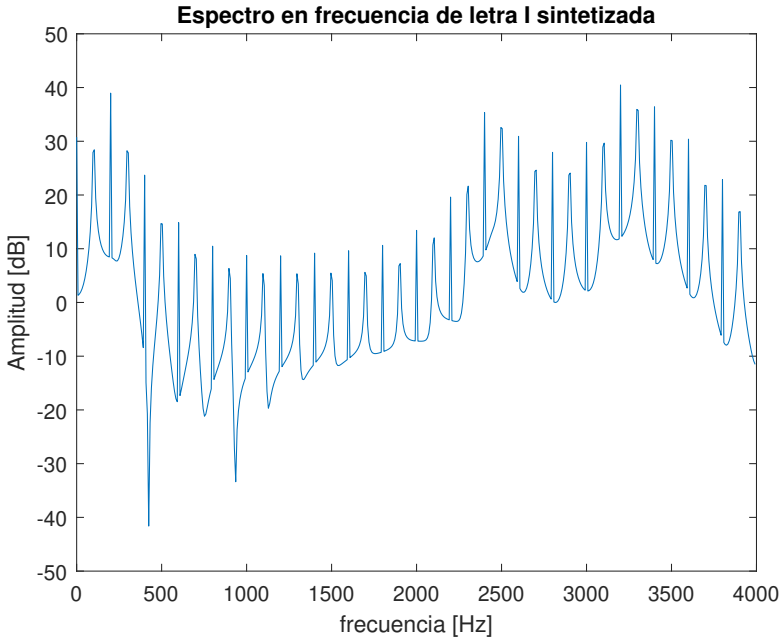

Figura 10: Espectro en frecuencia de letra I sintetizada.

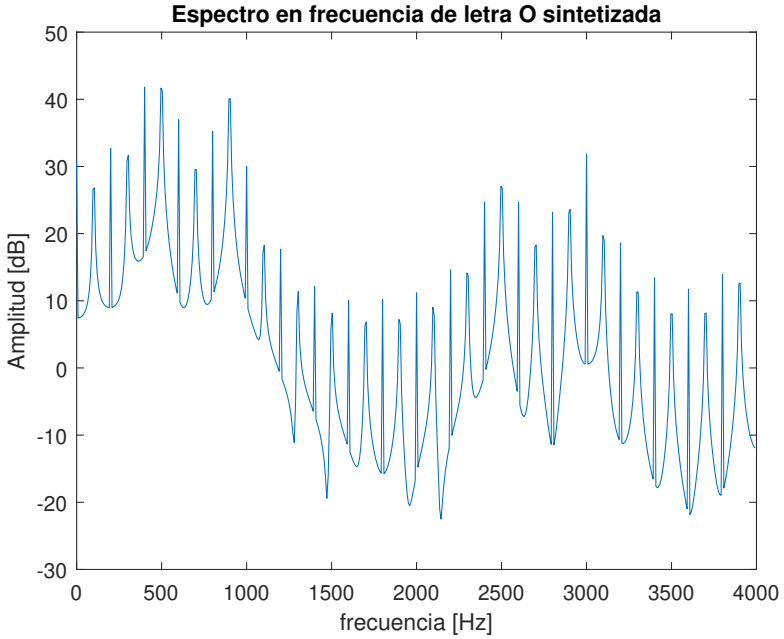

Figura 11: Espectro en frecuencia de letra O sintetizada.

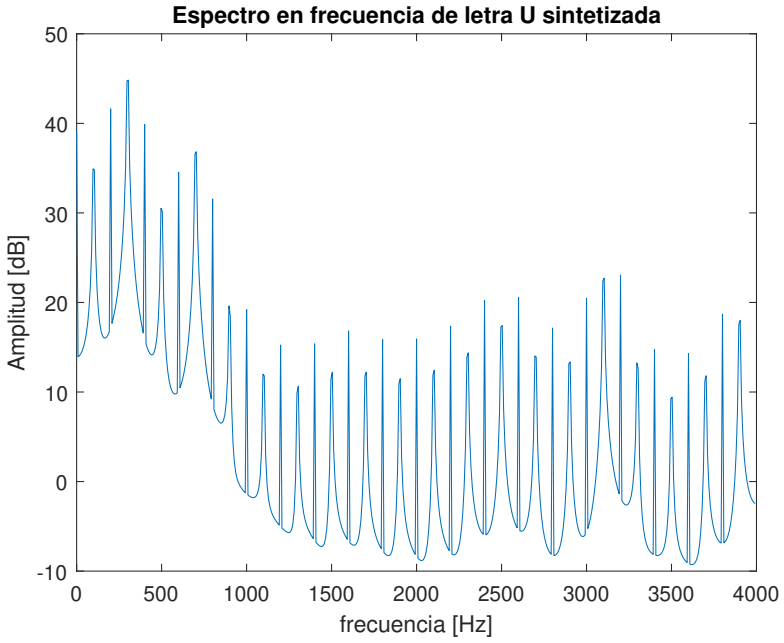

Figura 12: Espectro en frecuencia de letra U sintetizada.

1.4. Generación de función $y = mylpc(x, p)$ y comparación con puntos anteriores

En primer lugar se escribe la función $y = mylpc(x, p)$ la cual corresponde a una implementación del comando $lpc(x, p)$ usado anteriormente. El código se muestra a continuación:

```
function y = mylpc(x,p)
    rx = xcorr(x);
    rx_toeplitz  = rx(length(x):length(x)+p-1);
    rx = rx(length(x)+1:length(x)+p);

    Ra = toeplitz(rx_toeplitz);

    a = (Ra\rx)';
    y = [1 -a];
end
```

donde, obteniendo R_x y r_x a partir de estimaciones de la autocorrelación se resuelve $R_x a = r_x$, para finalmente retornar los parámetros del filtro AR diseñado.

Posteriormente se utiliza la nueva función para repetir los puntos 1.2 y 1.3. Los coeficientes AR obtenidos con el comando *lpc* y *mylpc* coinciden para cada vocal. Los gráficos de la respuesta en frecuencia de los filtros diseñados con *mylpc* se muestran en las figuras 13 - 17, coincidiendo con las obtenidas en 1.2.

Los archivos de audio resultantes se adjuntan a la entrega con el formato `mylpc_vowel_x` siendo `x` la vocal respectiva.

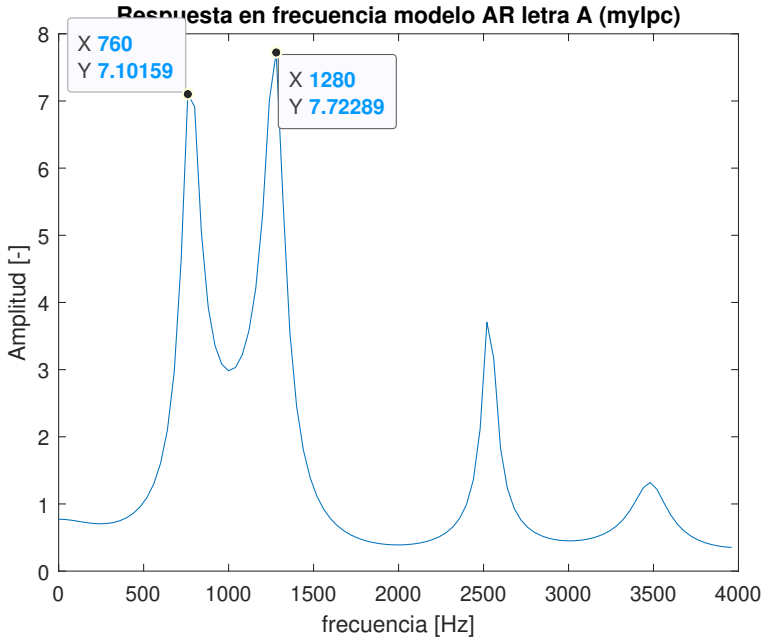

Figura 13: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra a. Se destaca el primer y segundo formante encontrado (*mylpc*).

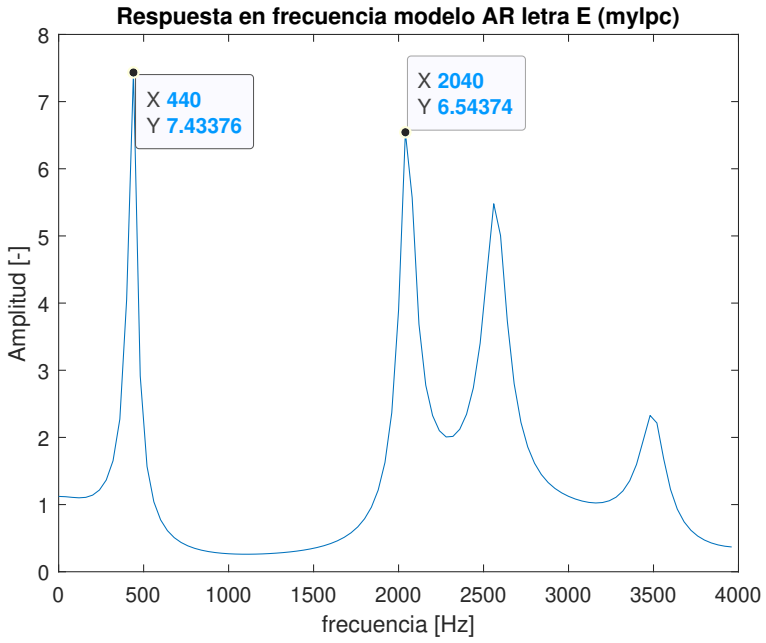

Figura 14: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra e. Se destaca el primer y segundo formante encontrado (*mylpc*).

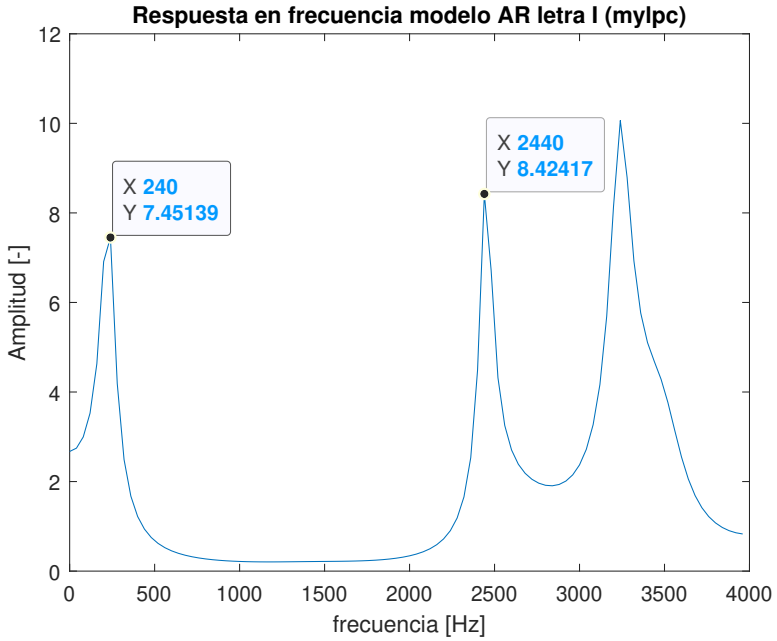

Figura 15: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra i. Se destaca el primer y segundo formante encontrado (*mylpc*).

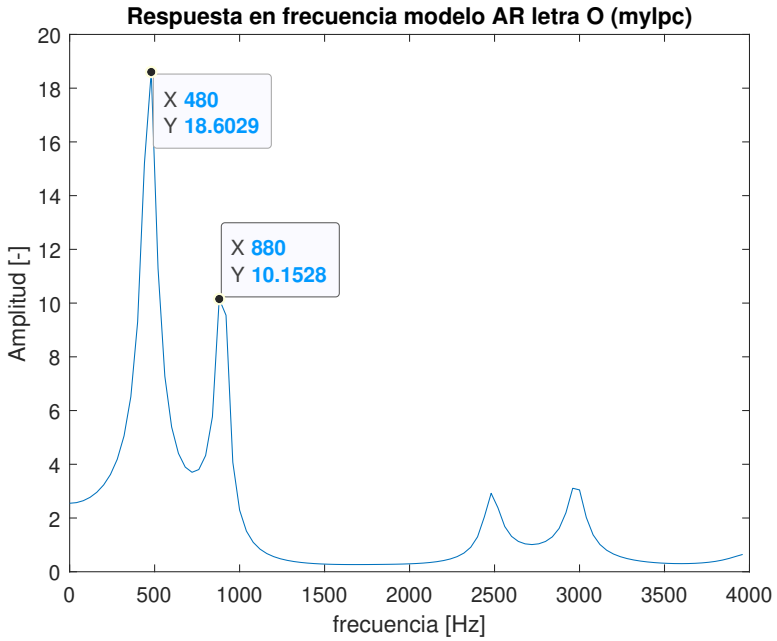

Figura 16: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra o. Se destaca el primer y segundo formante encontrado (*mylpc*).

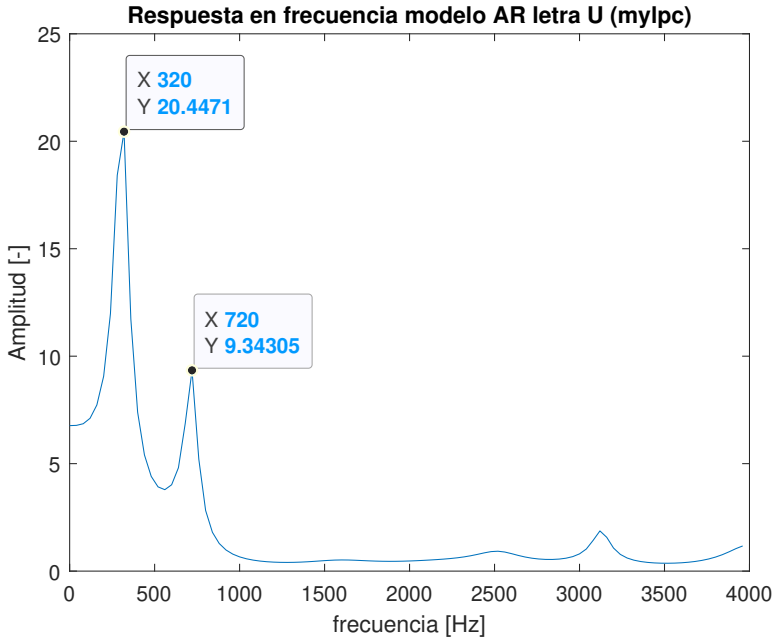

Figura 17: Respuesta en frecuencia de filtro estimado para simular tracto vocal haciendo letra u. Se destaca el primer y segundo formante encontrado (*mylpc*).

2. Clasificación de segmentos VUS

Se pide escribir función que para obtener el número de cruces por cero por milisegundo de una señal. El código escrito se muestra a continuación:

```
function n_ms = cruces_zero(x,fs)
    n = 0;

    for i = 1:length(x)-1
        if x(i+1)*x(i) < 0
            n = n+1;
        end
    end

    ms = 1000*length(x)/fs;
    n_ms = n/ms;
end
```

2.1. Obtención de Criterios de Clasificación

Para esta sección se utiliza el código `p2_1.m`, el cual se adjunta a la entrega.

En primer primer lugar se separa la señal *training_signal* por cada letra con la ayuda del comando *ginput*. Esta señal corresponde a alguien diciendo "señales temporales". La separación se muestra en la figura 18

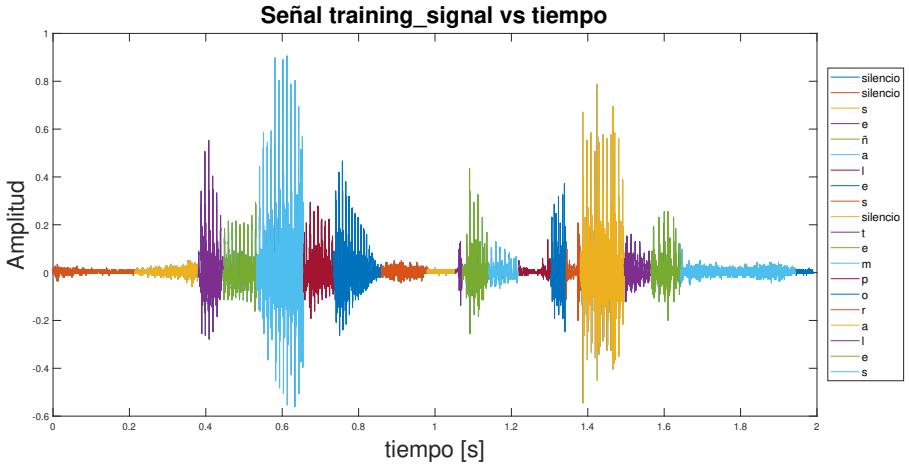

Figura 18: Separación manual por cada letra y silencios en señal *training_signal*.

En el cuadro 1 se muestra el valor RMS, la razón de Cruces por Cero por milisegundo y la clasificación VUS de los segmentos encontrados en *training_signal*.

Sonido	RMS	Ratio Cruces por Cero	Clasificación VUS
Silencio 1	0.0077	2.7242	S
s	0.0121	3.8855	U
e	0.0995	1.6557	V
ñ	0.0766	1.0613	U
a	0.1617	2.9960	V
l	0.0685	2.3987	U
e	0.0788	1.5984	V
s	0.0151	4.5381	U
Silencio 2	0.0061	2.4678	S
t	0.0359	1.7297	U
e	0.0714	1.4756	V
m	0.0336	0.8166	U
p	0.0161	2.0000	U
o	0.0911	1.5181	V
r	0.0394	1.7627	U
a	0.1525	1.9349	V
l	0.0412	1.4611	U
e	0.0544	1.9904	V
s	0.0136	4.3554	U
Silencio 3	0.0061	2.4632	S

Cuadro 1: Valor RMS, Ratio de Cruces por Cero por milisegundo y Clasificación VUS de segmentos encontrados en *training_signal*

Posteriormente se obtiene una nube de puntos para los segmentos VUS, donde el eje horizontal corresponde al valor RMS y el vertical a los cruces por cero. Dicho gráfico se muestra en la figura 19.

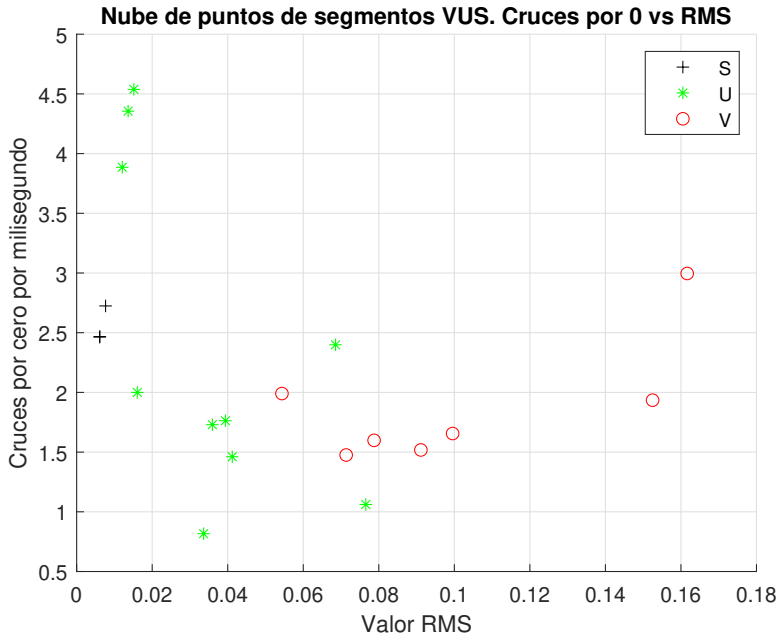

Figura 19: Nube de puntos para los segmentos VUS para una posterior clasificación.

A partir de la nube de puntos y una inspección visual sencilla se concluyen

los siguientes criterios de clasificación:

1. Valor RMS:
   - Si $RMS < 0{,}01$: Sonido tipo S.
   - Si $0{,}01 \leq RMS < 0{,}06$: Sonido tipo U.
   - Si $RMS \geq 0{,}06$: Sonido tipo V.
2. Cruces por cero (CZ):
   - Si $CZ < 2{,}25$: Sonido tipo V.
   - Si $2{,}25 \leq CZ < 2{,}8$: Sonido tipo S.
   - Si $CZ \geq 2{,}8$: Sonido tipo U.

Viendo la nube de puntos la decisión de clasificar los segmentos mediante umbrales en los cruces por cero no parece ser un método muy fiable. Por otro lado, umbrales en el valor RMS parece tener un mejor desempeño con respecto a la clasificación.

2.2. Comparación de criterios

Para esta sección se utiliza el código `p2_2.m`, el cual se adjunta a la entrega. Cómo método combinado se propone:

1. Valor RMS y Cruces por Cero (CZ):
   - Si $CZ < -250RMS + 5$: Sonido tipo S.
   - Si $CZ \geq -250RMS + 5$ y $RMS < 0{,}06$: Sonido tipo U.
   - Si $RMS \geq 0{,}06$: Sonido tipo V.

El cual se obtuvo a partir de delimitar la nube de puntos obtenida en la sección anterior por 2 rectas, las cuales se muestran en la figura 20.

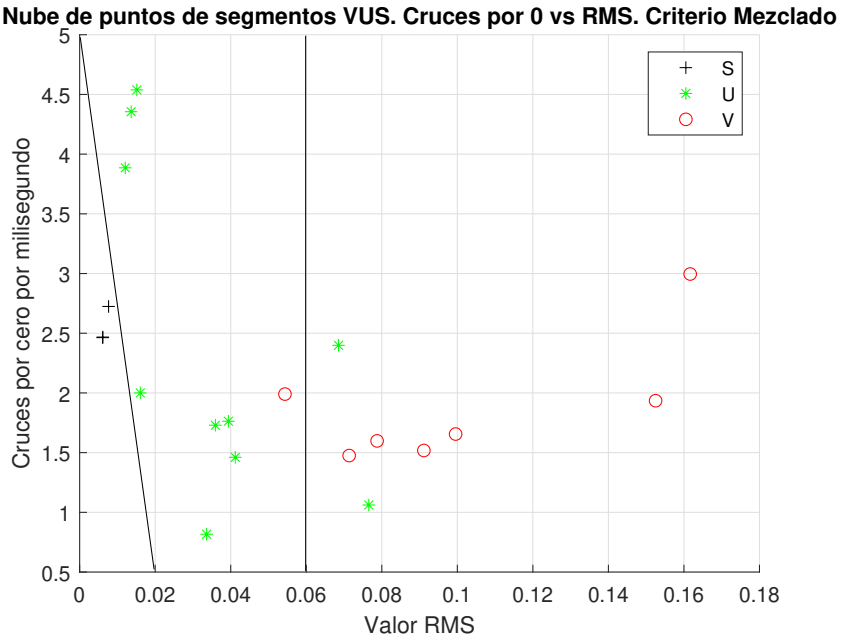

Figura 20: Nube de puntos para los segmentos VUS con rectas para clasificación con criterio combinado.

Al revisar los resultados de los 3 métodos, se aprecia que:

- El criterio de los Cruces por Cero presenta resultados no satisfactorios. Lo anterior era esperable debido a que no se apreciaba una correlación en la nube de puntos lo suficientemente fuerte como para aplicar umbrales.
- El criterio del valor RMS presenta resultados bastante mejores que el de cruces por cero. De la nube de puntos era esperable que umbrales en el valor RMS (separación del plano por líneas verticales) debía dar mejores resultados en la clasificación.
- El criterio combinado propuesto entrega resultados muy similares de clasificación de puntos. Lo anterior no es una sorpresa, ya que este criterio puede interpretarse como una leve inclinación de la recta vertical correspondiente al umbral más bajo del criterio RMS.

Por todo lo anterior se concluye que desde una perspectiva costo-efectivo el mejor criterio encontrado corresponde al del valor RMS.

Finalmente se grafica la señal *text_signal*, la variable VUS (1, -1 o 0) obtenida a partir de la estimación de RMS y el valor RMS de los segmentos. Dichos gráficos se muestran en la figura 21. Con respecto a los gráficos se puede comentar que:

- Como es de esperarse, el valor RMS se comporta como una especie de envolvente de la señal.
- Al ser la señal "sonidos de voz" es claro que el criterio presenta falencias en la clasificación.
- A partir de la evolución del valor RMS en el tiempo, podría mejorar la clasificación cambiando los umbrales.

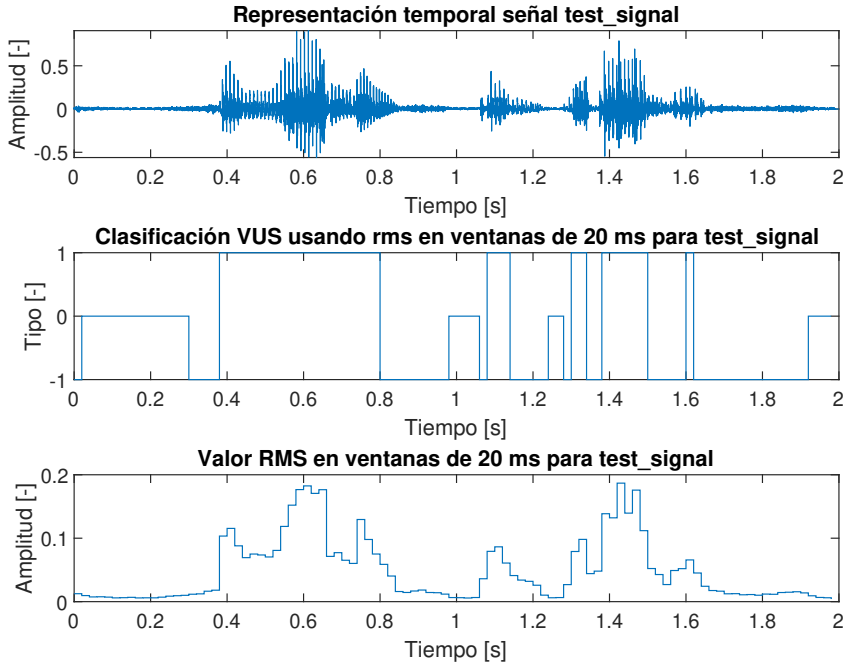

Figura 21: Señal *text_signal*, clasificación VUS (RMS) y valor RMS de segmentos clasificados.

3. Síntesis de voz hablada

3.1. Compresión y Síntesis de Señal de Voz

Se pide aplicar compresión y síntesis la señal de voz *test_signal*. Los pasos para realizar lo anterior corresponden a:

- Compresión:

  1. Subdividir la señal en segmentos de 20 ms.
  2. Obtener el valor RMS de cada segmento.
  3. Clasificar cada segmento como V, U o S con el criterio basado en RMS obtenido en la parte 2.
  4. Obtener los valores del filtro AR que simula el tracto vocal por medio de *mylpc* para los sonidos de tipo V y U.

- Síntesis:

  Se van generando los segmentos en orden

  - En el caso de que un segmento corresponda a un sonido S, generar segmento de silencio (ceros) de 20 ms.
  - En el caso de que el sonido corresponda a un sonido V, excitar filtro AR respectivo con tren de impulsos de 20 ms (en este caso de 100 Hz). Realizar corrección de RMS del segmento.
  - En el caso de que el sonido corresponda a un sonido U, excitar filtro AR respectivo con señal de ruido blanco de 20 ms. Realizar corrección de RMS del segmento.

  Finalmente se concatenan los segmentos generados

El código que realiza lo anterior corresponde a `p3_1.m` y se adjunta a la entrega.
El gráfico de la señal original y la sintetizada se muestran en la figura 22. Visualmente la síntesis parece correcta.

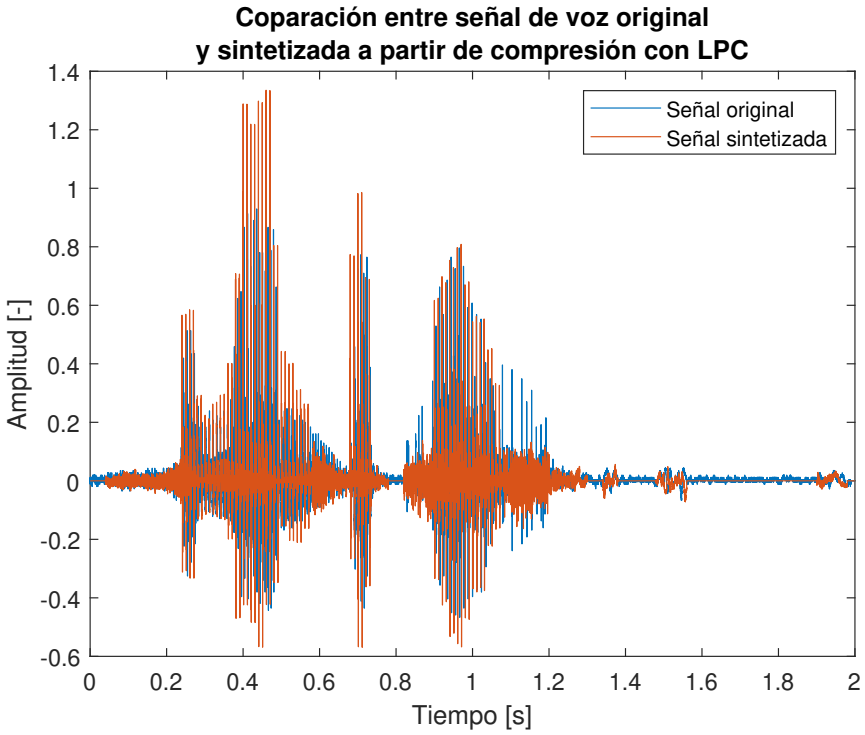

Figura 22: Comparación gráfica de señal de voz original y sintetizada.

Finalmente se guarda la señal sintetizada como *my_test_signal.wav*, la cual se adjunta a la entrega. Con respecto a la calidad del audio, si bien mantiene el timbre "robótico", se entiende lo que se esta diciendo.

Para mejorar la síntesis se podría reconocer la frecuencia fundamental de la voz en sonidos tipo V y U, con la idea de excitar el filtro AR con un tren de impulsos de frecuencia acorde al hablante, obteniendo una voz más natural.

3.2. Estimación de Compresión

Se estima la compresión del archivo de audio como la razón entre los bytes usados para almacenar los coeficientes del filtro, los flags VUS y el valor RMS con respecto a los bytes de la señal *test_signal*. Para lo anterior se utilizó el comando *whos*

Por lo tanto, la razón de compresión R corresponde a:

$$R = \frac{\text{bytes}(A) + \text{bytes}(flags) + \text{bytes}(RMSs)}{\text{bytes}(test\_signal)} = \frac{9088 + 100 + 100}{127360} \approx 0{,}079$$

4. Filtro AR

4.1. Función *positiveSpectrum*

Se implementa la función *positiveSpectrum(x)* que recibe como entrada una señal digital x, de largo N, la cual haciendo uso de la función *fft()* de Matlab pueda entregar el vector *amp* que corresponde de amplitudes de la DFT para el rango de frecuencias positivas y un vector w correspondiente a las frecuencias normalizadas en *rad/muestra*. El código de la función descrita se presenta a continuación

```
function [amp,w] = positiveSpectrum(x)
    X = fft(x);
    amp = X(1:floor(length(X)/2))
    w = linspace(0,pi,length(amp))
end
```

Para poner a prueba la función implementada se grafica usando el comando `plot()` el espectro en frecuencia positivo de la vocal A contenida en la señal *vowel_a*, obteniendo la gráfica presente en la figura 23

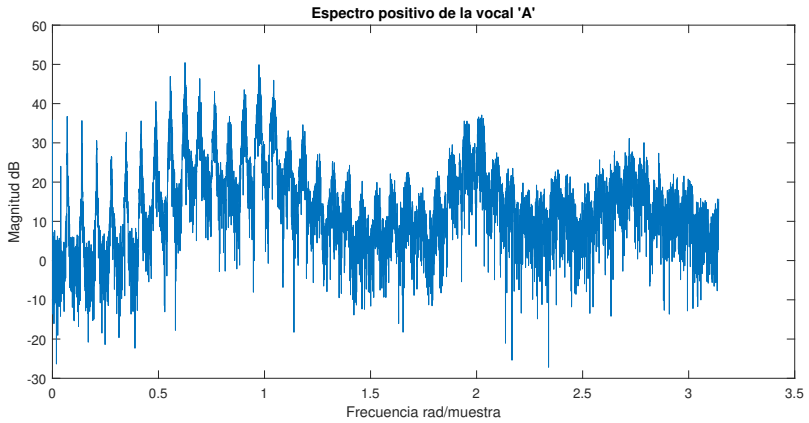

Figura 23: Espectro de la señal *vowel_a*

En el espectro obtenido se pueden diferenciar 4 formantes ubicadas aproximadamente en 0,6, 1,2, 2 y 2,7 *rad/muestra*

4.2. Experimentación con Órdenes del filtro AR en LPC

Para está parte se utiliza el código `p4_2.m`, el cual se adjunta a la entrega.

Inicialmente se gráfica de la respuesta en frecuencia del filtro AR de orden 15 y 2, junto al espectro de *vowel_a*. Los gráficos se muestran en la figura 24. Se aprecia que con 2 polos se tiende a formar una resonancia al medio de los 2 formantes de la vocal.

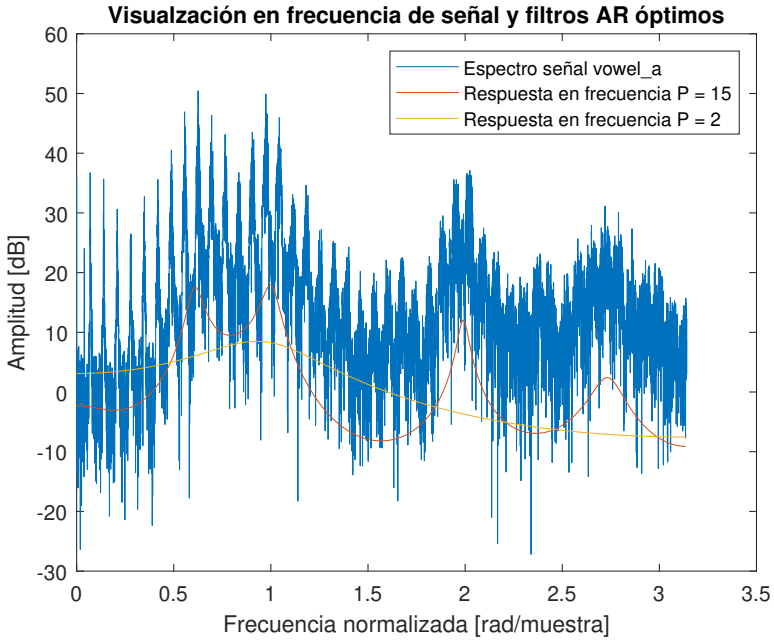

Figura 24: Espectro de *vowel_a* y respuesta en frecuencia del filtro AR de orden 15 y 2 obtenidos por LPC.

Por otro lado se obtiene el diagrama de polos y ceros de los filtros diseñados para la vocal a. Dichos diagramas se muestran en la figura 25

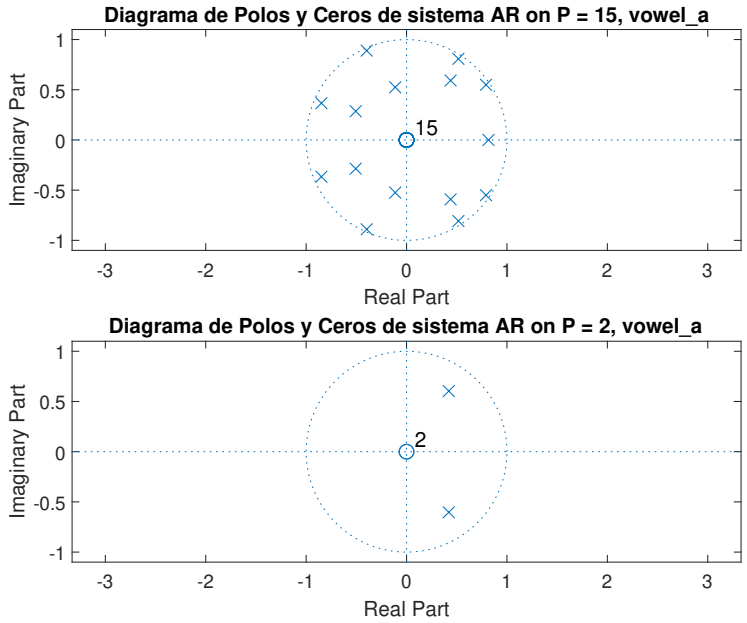

Figura 25: Diagrama de polos y ceros para filtros de orden 15 y 2 diseñados por LPC para la señal *vowel_a*.

Posteriormente se grafica de la respuesta en frecuencia del filtro AR de orden 15 y 2, junto al espectro de *vowel_u*, los cuales se muestran en la figura 26. Se aprecia que con 2 polos se tiende a formar una resonancia en la frecuencia 0. Lo anterior también se aprecia en la figura 27, donde se presenta el diagrama de polos y ceros para los filtros diseñados para la vocal u.

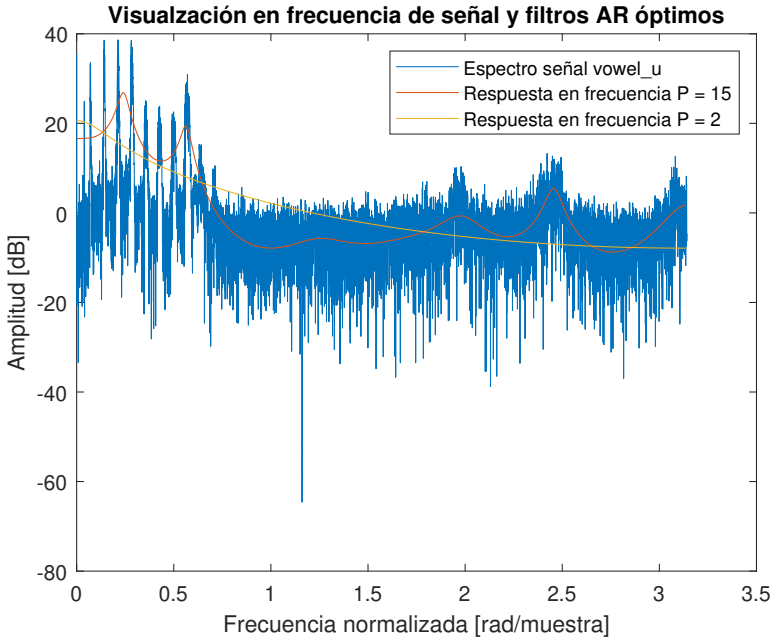

Figura 26: Espectro de *vowel_u* y respuesta en frecuencia del filtro AR de orden 15 y 2 obtenidos por LPC.

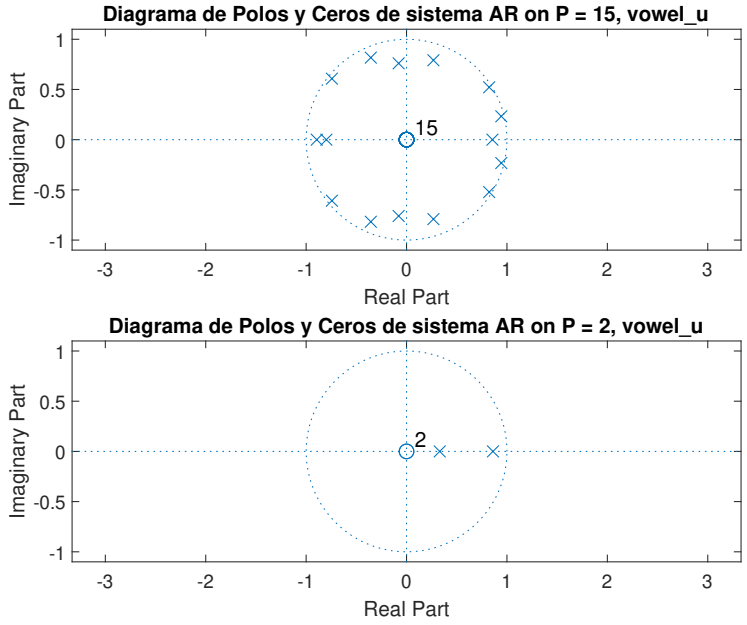

Figura 27: Diagrama de polos y ceros para filtros de orden 15 y 2 diseñados por LPC para la señal *vowel_u*.

4.3. Comparación de filtrado AR de distinto orden

Para estudiar las diferencias de resultados al filtrar una señal con filtros AR de distinto orden, primero se escoge una señal de vocal arbitraria para que sea filtrada, en este caso se escogió la señal *vowel_a* correspondiente a la vocal *A*.

Lo siguiente es definir los ordenes de los filtros que se usarán, se escoge como referencia un orden relativamente alto 15 y uno menos de 9. Se crea un tren

de impulsos de 100 Hz de frecuencia con 0,5 s de duración. Se obtienen los coeficientes a_k de cada filtro y se procede con el filtrado de la señal.

Los resultados obtenidos se muestran en las gráficas de la figura 28, se incluyen además los diagramas de polos y ceros en la figura 31, donde la gráfica superior corresponde al diagrama asociado al filtro de orden 15 y la inferior al filtro de orden 9.

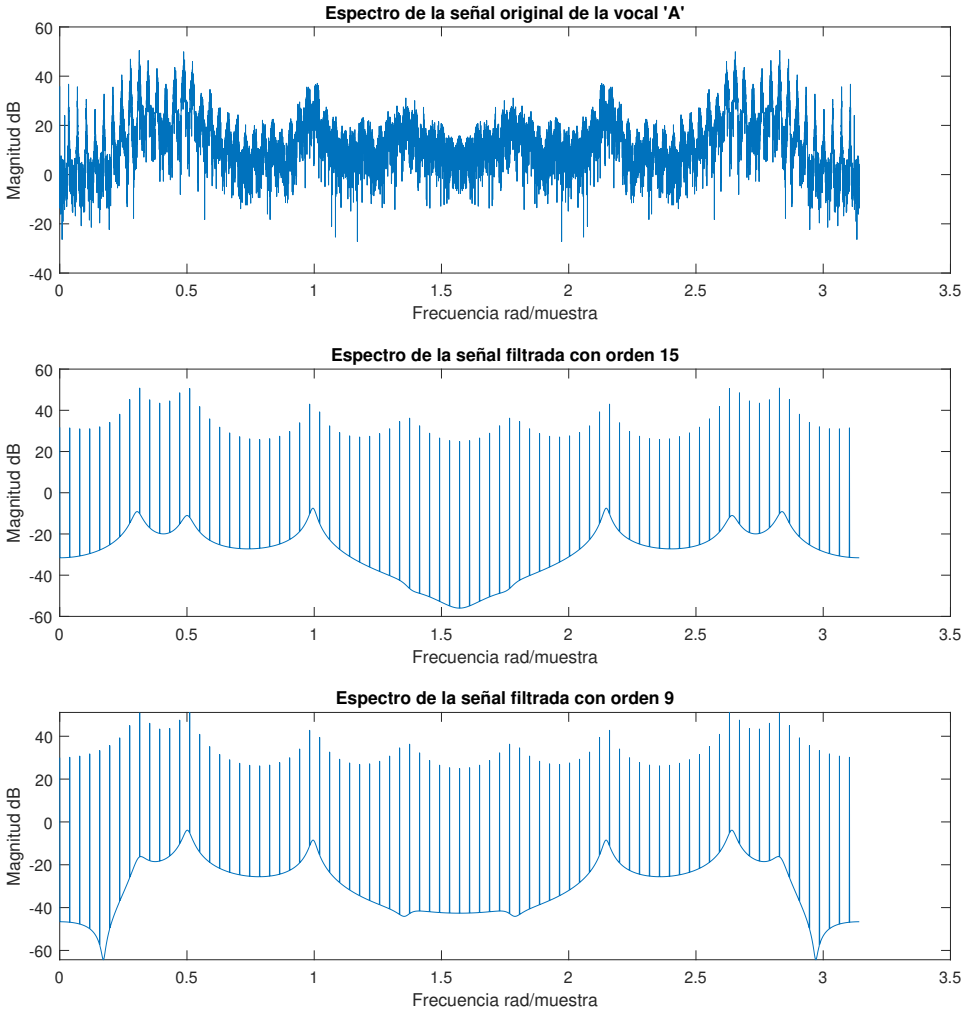

Figura 28: Espectro en frecuencia de la señal *vowels_a* y el resultado de filtrado de orden 15 y 9

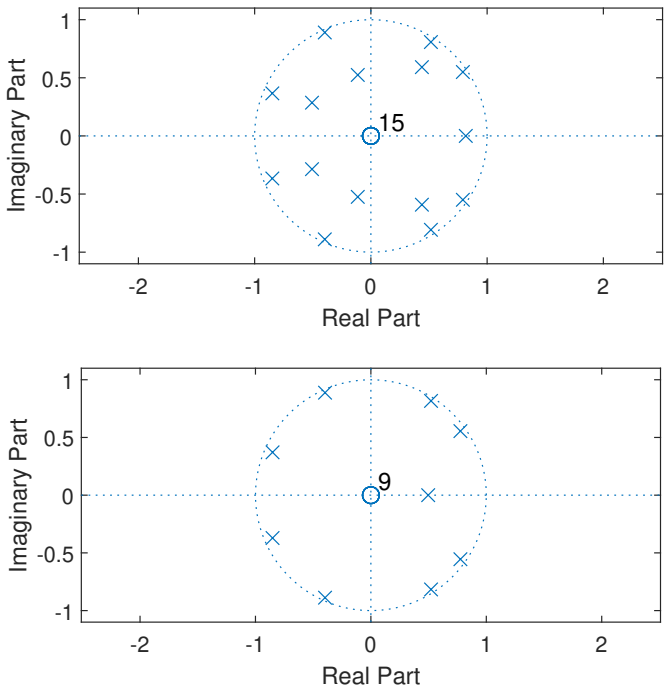

Figura 29: Diagramas de polos y ceros para los filtros AR de orden 15 y 9

Se pide además realizar el mismo ejercicio anterior comparando el filtrado de la señal *vowel_u* correspondiente a la vocal *U*, probando distintos ordenes de filtrado hasta encontrar el mínimo orden que no presente diferencias significativas en cuando a la percepción auditiva respecto al resultado obtenido con el filtrado de orden 15.

Se notó que haciendo uso de un filtro de orden 10 la señal ya se distorsiona lo suficiente para tener la impresión de que la vocal está siendo emitida por una persona distinta, por lo que se concluye que el orden mínimo para no tener diferencias significativas un filtro de orden 11. Los espectros en frecuencia obtenidos para filtros de orden 15 y 10 se presentan en la figura 30, se incluyen además los diagramas de polos y ceros en la figura **??**, donde la gráfica superior corresponde al diagrama asociado al filtro de orden 15 y la inferior al filtro de orden 10.

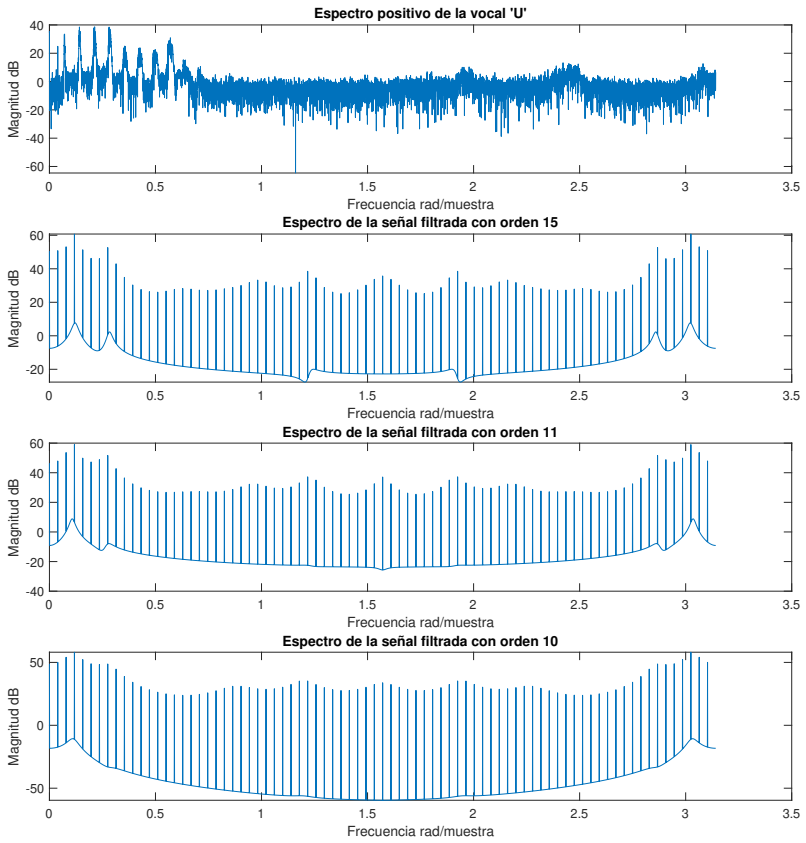

Figura 30: Espectro en frecuencia de la señal *vowels_u* y el resultado de filtrado de orden 15 y 10

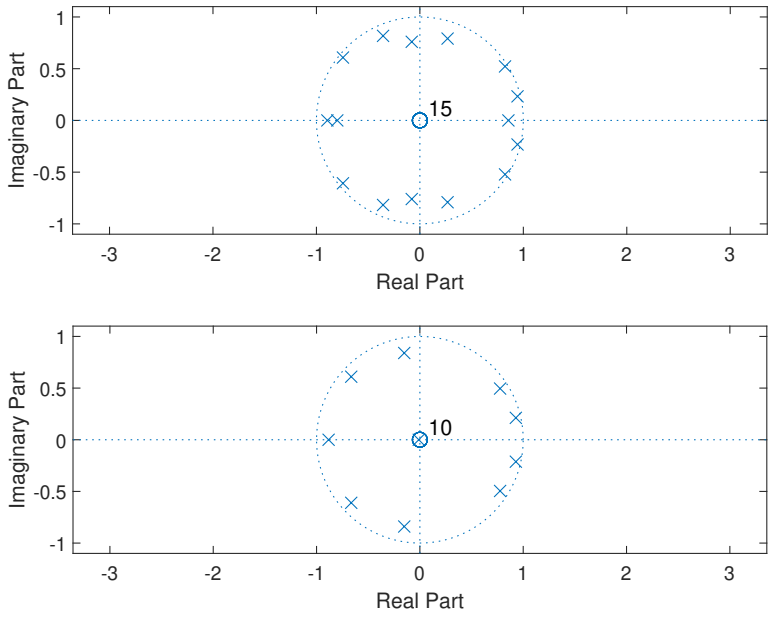

Figura 31: Diagramas de polos y ceros para los filtros AR de orden 15 y 10

De la figura 30 se puede ver que en el espectro de la vocal U se ven 5 formantes,

y el orden mínimo que permitía una síntesis de la señal sin diferencias significativas fue 11. Por otro lado en en la figura 23 de la sección 4.1 se encontraron 4 formantes y cuando se sintetizó la señal con un filtro de orden 9 no se notaron diferencias significativas respecto al filtrado de orden 15, con esto se puede concluir que un buen criterio para lograr una buena síntesis de este tipo de señales es encontrando el orden según $2n + 1$, siendo n el número de formantes de la señal a sintetizar.

4.4. Filtro AR con filtros *Biquad*

El procedimiento que hasta ahora se ha llevado a cabo haciendo uso de filtros AR de distintos ordenes se puede homologar mediante el procesamiento en cascada de la señal con múltiples filtros *Biquad* como los estudiados en experiencias anteriores. Matlab provee de la función `tf2sos(b,a)` que recibe como parámetros los coeficientes a_k y b_k de un filtro y entrega la matriz sos, que contiene en sus $n/2$ filas en caso de que n sea par y $n/2 + 1$, donde n corresponde al orden del filtro original. Cada fila contiene los coeficientes del numerador y denominador A_k Y B_k de los filtros *Biquad* necesarios para que, de en forma de cascada filtren una señal y obtengan el resultado del filtro original. El comando además entrega el parámetro g correspondiente a la ganancia que poseen los filtros *Biquad* entregados en la matriz.

Se experimenta el uso de la función antes descrita para un filtro AR de orden 15 obtenido al usar el comando `lpc` para la señal *vowel_a*, se obtiene la respuesta en frecuencia de cada filtro *Biquad* entregado en la matriz permitiendo analizar el aporte individual de cada una de ellas en la respuesta en frecuencia total. Se grafican dichas respuestas en frecuencia forma superpuesta como se puede ver en la figura 32.

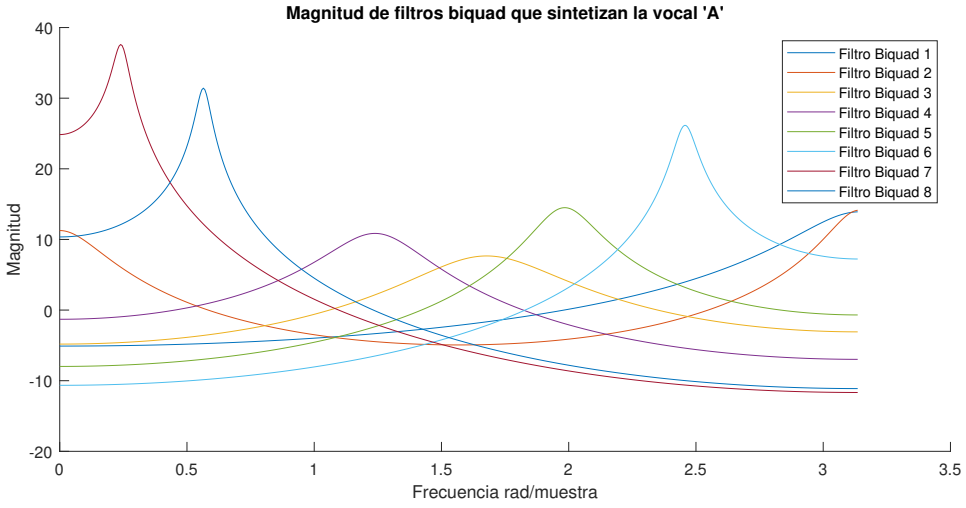

Figura 32: Magnitud en dB de la respuesta en frecuencia de los filtros *Biquad* entregados por la función `tf2sos(b,a)`

Se grafican además de forma superpuesta las magnitudes de las respuestas en frecuencia de los filtros junto al espectro de la señal *vowel_a* como se muestra en la figura 33

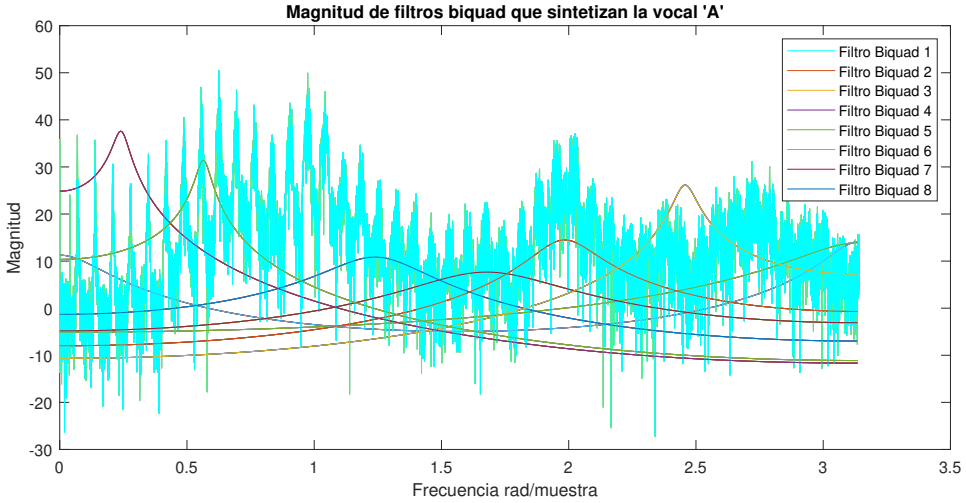

Figura 33: Magnitud en dB de la respuesta en frecuencia de los filtros *Biquad* entregados por la función tf2sos(b,a) junto al espectro de la vocal A

5. Observando el residuo de predicción

Para esta parte se utiliza el script `p5.m`, el cual se adjunta a la entrega

Para lo que sigue de la sección, se considera $e[n]$ como la señal de entrada a filtro AR. Por lo tanto

$$X(z) = E(z) \cdot H(z)$$

5.1. Filtro inverso

Se puede obtener el filtro inverso al AR como el filtro $H(z)^{-}1$, el cual corresponde a un filtro FIR. A partir de lo anterior se la respuesta en frecuencia del filtro inverso, la cual se muestra en la figura 34

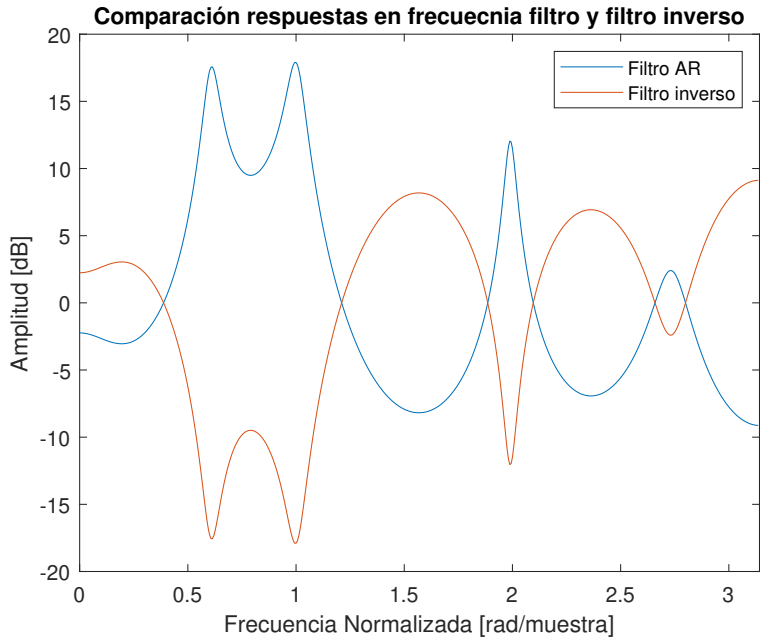

Figura 34: Comparación de respuesta en frecuencia de filtro y filtro inverso.

Como es de esperarse, suma de la respuesta en frecuencia del filtro inverso con el filtro AR es 0.

5.2. Obtención de la $e[n]$

Se obtiene $e[n]$ aplicando el filtro inverso a la señal *vowel_a*. La señal de entrada obtenida se muestra en la figura 35.

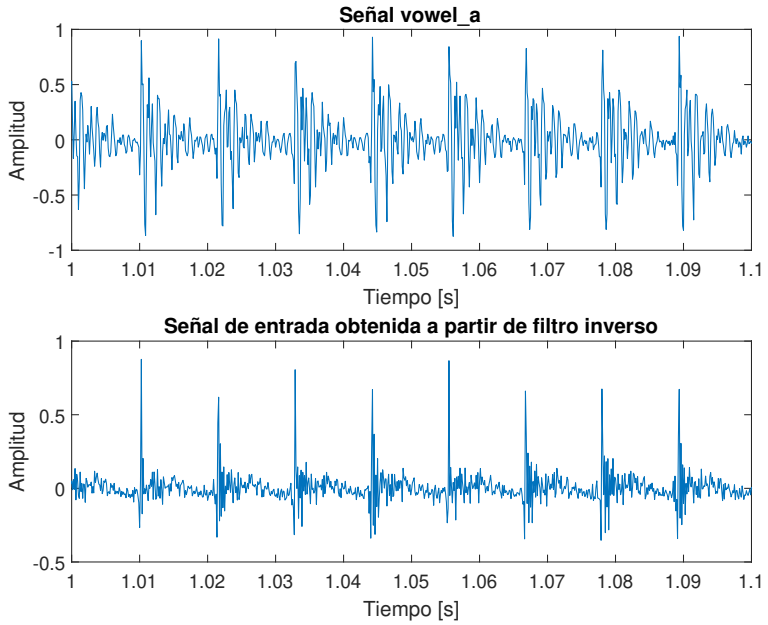

Figura 35: Señal *vowel_a* y entrada $e[n]$ según filtro inverso.

Con respecto a la forma de la señal $e[n]$ se pueden comentar que:

- Modelar la entrada como un tren de impulsos para síntesis no está tan alejado de la entrada "real", por lo menos visualmente.
- La entrada tiene cierta periodicidad, por lo que se podría modelar como un tren de impulsos convolucionado con otra señal. Lo anterior puede interpretarse como que esta otra señal podría ser parte de la respuesta a impulso que el filtro AR no fue capaz de capturar (respuesta a impulso del tracto vocal). Otra interpretación sería que corresponde a la respuesta impulso de un sistema que no corresponde al del tracto vocal, como podría ser del equipo con el que se grabó o del lugar.

5.3. Espectro en frecuencia de $e[n]$ y *vowel_a*

Se grafica el espectro en frecuencia de $e[n]$ y la señal *vowel_a*, los cuales se muestran en la figura 36. Con respecto a los gráficos obtenidos se puede comentar que:

- La señal $e[n]$ tiene un contenido en frecuencia aproximadamente plano.
- La respuesta en frecuencia del filtro AR de a empieza a parecer la envolvente del espectro de *vowel_a* debido a la naturaleza plana y ruidosa de $e[n]$.

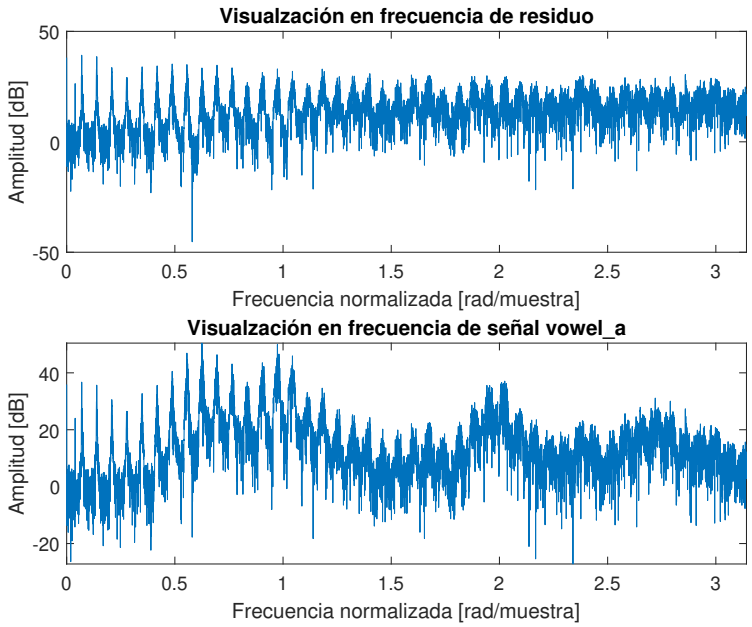

Figura 36: Espectro en frecuencia de $e[n]$ y *vowel_a*

5.4. Autocorrelación del Residuo y Frecuencia Fundamental

Se obtiene la autocorrelación del residuo $e[n]$ y de la señal *vowel_a*. Ambas señales se muestran en los gráficos presentes en la figura 37.

Para estimar la frecuencia fundamental la autocorrelación de $e[n]$ resulta bastante útil. Un peak en la autocorrelación significa que la señal tiene un gran parecido con respecto a dicho desfase de muestras, por lo que una señal que presente cierta periodicidad debiese tener peaks periódicos en su estimador de autocorrelación.

En el caso de la figura 37, se aprecian peaks periódicos cada un desfase D aproximado de 90 muestras, por lo que la frecuencia fundamental podría estimarse como:

$$f_0 = \frac{1}{D \cdot T_s} = \frac{1}{90 \cdot (1/8000)} \approx 88{,}9\ Hz$$

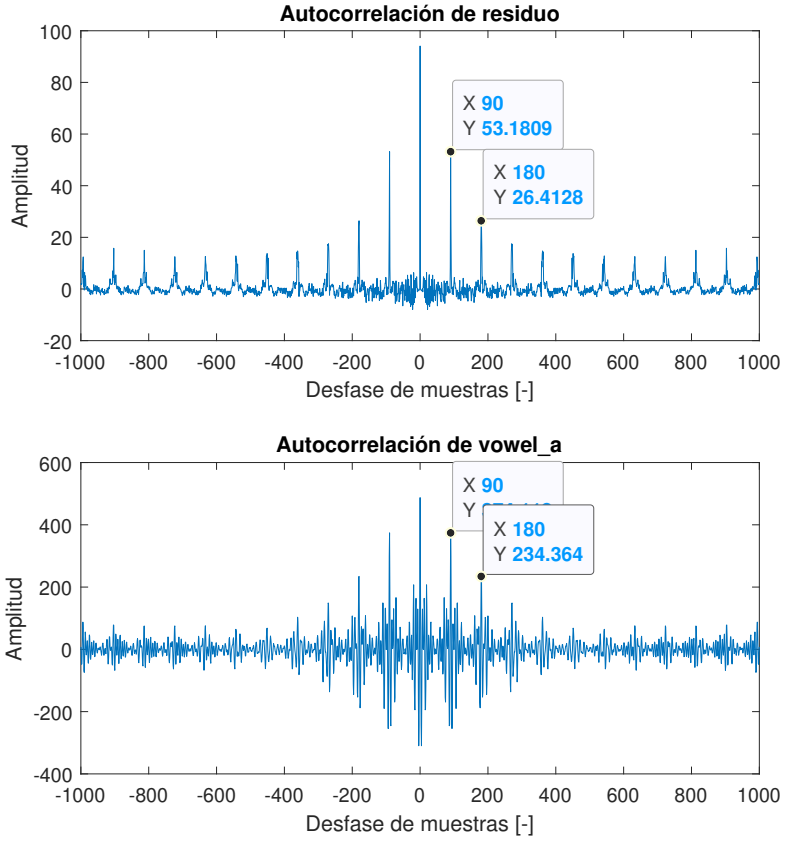

Figura 37: Autocorrelación de $e[n]$ y $vowel\_a$

Este método de estimación depende de que la señal este poco correlacionada en el entorno del desfase D respectivo a la fundamental. De no ser así complicaría encontrar el peak, lo cual se puede apreciar levemente en la autocorrelación del la señal $vowel\_a$, ya que si bien los peaks "importantes" para estimar la frecuencia fundamental aún son distinguibles, un filtro de respuesta a impulso más ancha haría la tarea más complicada.