UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

"DESARROLLO DE HERRAMIENTA DE ANÁLISIS PARA LA CODIFICACIÓN Y SÍNTESIS PARAMÉTRICA DE SEÑALES DE VOZ Y APLICACIÓN DE EFECTOS"

TRABAJO FINAL DE GRADO

Autor/a:

Carlos Pérez García

Tutor/a:

Miguel Ferrer Contreras

GANDIA, 2018

Resumen

La codificación paramétrica de las señales de voz es una de las más empleadas en comunicaciones por su capacidad de compresión y su facilidad para re-sintetizar la información en el receptor a partir de la extracción y transmisión de unos pocos parámetros. Se propone desarrollar una herramienta en Matlab que permita analizar la calidad de la señal de voz en función de las características de la codificación paramétrica empleada, así como realizar efectos o cambios en la sintetización de la misma en función de la manipulación de dichos parámetros.

Palabras clave: Codificación paramétrica, Filtros de predicción Lineal, Vocoder, Pitch, Síntesis de Voz.

Abstract

The parametric coding of the voice signals is one of the most used in communications for its compression capacity and its facility to re-synthesize the information in the receiver from the extraction and transmission of a few parameters. It is proposed to develop a tool in Matlab that allows to analyze the quality of the voice signal depending on the characteristics of the parametric coding used, as well as to perform effects or changes in the synthesization of the same depending on the manipulation of said parameters.

Keywords: Parametric coding, Lineal prediction filters, Vocoder, Pitch, Voice synthesis.

Contenido

Capítulo 1 INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción.

En el presente trabajo se han desarrollado herramientas para codificar, sintetizar y realizar efectos de voz usando codificación paramétrica, utilizando filtros de predicción lineal.

A su vez se ha estudiado la influencia de los parámetros derivados de la codificación paramétrica empleada a la hora de codificar y sintetizar las señales de voz.

Estas herramientas han sido desarrolladas bajo el entorno de trabajo facilitado en el paquete matemático, Matlab.

1.2 Objetivos principales

- Implementar un codificador de voz paramétrico configurable (señal de excitación, tamaño de trama, solape, etc.) en Matlab.
- Implementar un sintetizador de voz o decodificador paramétrico en Matlab usando diferentes variantes (pitch fijo, pitch recibido…).
- Analizar la influencia en la calidad de la voz de los parámetros que definen la codificación.
- Proponer efectos en función de la influencia detectada en el cambio de los parámetros.

1.3 Objetivos secundarios

- Realizar transformaciones tiempo-frecuencia, y compararlas con los efectos realizados mediante la modificación paramétrica.
- Extender el análisis y aplicación de efectos a señales de audio en general.

Capítulo 2 INTRODUCCIÓN TEÓRICA

2.1 Formación de la voz

El sonido vocal se produce por una señal de excitación producida en nuestra garganta (el aire de nuestros pulmones) que al chocar con nuestras cuerdas vocales ajusta el timbre y tono de nuestra voz y a su vez es modificada por diferentes resonancias, debido a las formas que puede adquirir nuestro tracto vocal [1].

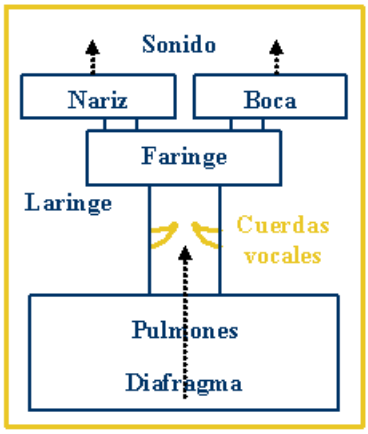

Ilustración 1:Formación de la voz

El sistema vocal humano puede dividirse en tres partes:

- Aparato respiratorio
- Aparato de fonación
- Aparato resonador

2.1.1 Aparato respiratorio

Es donde se almacena y circula el aire. Nariz, tráquea, pulmones y diafragma.

En la inspiración el diafragma se contrae y desciende, por lo cual la cavidad torácica se amplía y el aire entra en los pulmones. En la espiración o exhalación, el diafragma se relaja y sube, la cavidad torácica disminuye de tamaño provocando la salida del aire de los pulmones hacia el exterior.

El aparato respiratorio determina la intensidad con la que se inhalará y exhalará el aire [2].

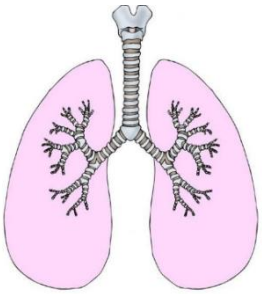

Ilustración 2: Pulmones

2.1.2 Aparato de fonación

Hay 3 mecanismos básicos de producción de voz [3]:

- Vibración de las cuerdas vocales, que produce los sonidos sonoros (vocales, semivocales, nasales, etc.).
  Estos sonidos son de mayor amplitud y periodicidad en el tiempo.

- Interrupción (total o parcial) en el flujo de aire que sale de los pulmones, que da lugar a los sonidos sordos (fricativas sordas, oclusivas sordas, etc.).
  Los sonidos sordos son de menor amplitud y naturaleza ruidosa en el tiempo.

- Combinación de vibración e interrupción, como las oclusivas sonoras (en español /b/, /d/ y /g/).

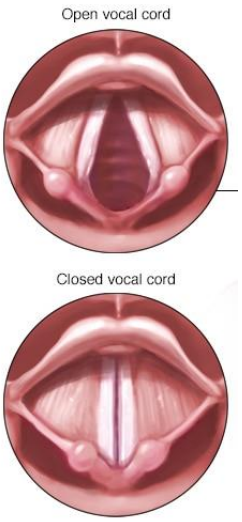

Ilustración 3: Cuerdas Vocales

2.1.3 *Aparato resonador*

El sonido producido en las cuerdas vocales es muy débil, por ello, debe ser amplificado.

Esta amplificación tendrá lugar en los resonadores nasal, bucal y faríngeo, donde se producen modificaciones que consisten en el aumento de la amplitud de ciertas frecuencias y la desvalorización de otras.

La voz humana, una vez que sale de los resonadores, es moldeada por los articuladores (paladar, lengua, dientes, labios y glotis), transformándose en sonidos del habla: fonemas, sílabas, palabras, etc.

La posición concreta de los articuladores determinará el sonido que emita la voz [4].

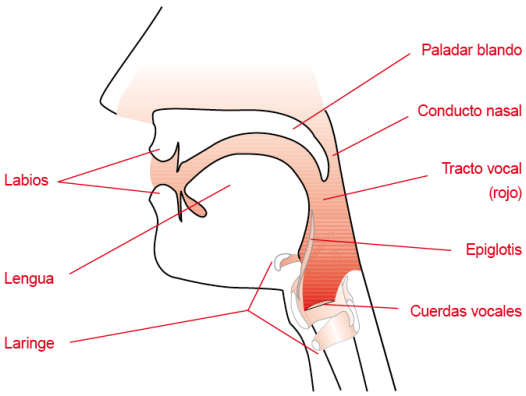

Ilustración 4: Aparato resonador

2.1.3.1 *Formantes de voz*

Al estar cambiando constantemente la posición de los articuladores cuando hablamos, estamos modificando constantemente los formantes de la voz.

Un formante es el pico de intensidad en el espectro de un sonido, se trata de concentración de energía (amplitud de onda) que se da en una determinada frecuencia.

En el habla se determinan por el proceso de filtrado por resonancia que se produce en el tracto vocal por la configuración de los articuladores.

Técnicamente los formantes son bandas de frecuencia donde se concentra la mayor parte de la energía sonora de un sonido [5].

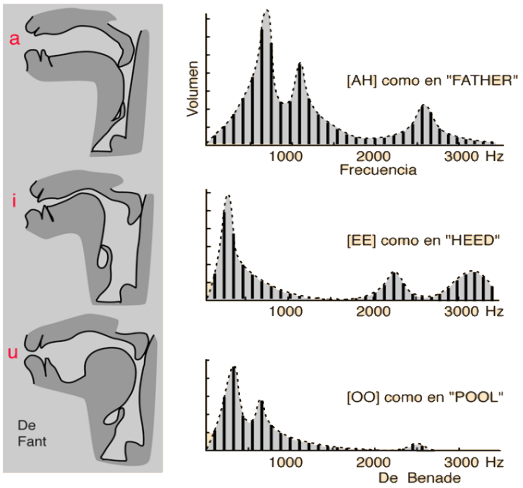

Ilustración 5: Formantes de voz

2.2 Clasificación de los codificadores de voz

En la actualidad existen una gran variedad de codificadores de voz y de audio, entre los más comunes, para la codificación de voz, podemos encontrar los siguientes [6]:

2.2.1 Codificadores de forma de onda:

Los codificadores de forma de onda intentan representar de manera compacta la forma de onda de la señal, independientemente del origen que tenga, por lo tanto, este tipo de codificadores se pueden utilizar para codificar cualquier tipo de señal (audio, vídeo, comunicaciones) o cualquier tipo de datos.

Los codificadores de forma de onda pueden trabajar tanto en el dominio temporal como en el dominio frecuencial. En ambos casos, intentan eliminar la redundancia que hay en la señal de entrada para reducir la tasa de bits.

Estos codificadores son de alta calidad (16-64kbps), pueden llegar a ser sin pérdidas.

2.2.2 Vocoders (Voice CODER):

Como hemos visto anteriormente, la voz humana consiste en sonidos generados por la apertura y cierre de la glotis (cuerdas vocales), lo que produce una onda periódica con muchos sonidos armónicos. Este sonido básico es entonces filtrado por la nariz y la garganta (un complicado sistema resonante conocido como el tracto vocal) de forma controlada, creando la amplia variedad de timbres del habla. Hay otro conjunto de sonidos, conocidos como sordos, que no son generados por la vibración de las cuerdas vocales.

El vocoder examina el habla encontrando su onda básica, que es la frecuencia fundamental, y mide cómo cambian en el tiempo las características espectrales, es decir los formantes.

Esto da como resultado una serie de parámetros que podemos utilizar para sintetizar de nuevo la señal original.

Al hacer esto, el vocoder reduce en gran medida la cantidad de información necesaria para almacenar el habla.

Para recrear el habla, el vocoder simplemente revierte el proceso, creando la frecuencia fundamental en un oscilador electrónico y pasando su resultado por una serie de filtros basado en la secuencia original de símbolos.

Aprovechan las particularidades de la señal de voz para optimizar la codificación.

En general este tipo de codificadores suele ser de baja/media calidad (1,2-4,8kbps).

Pueden ser con pérdidas.

2.2.3 Codificadores híbridos:

Es una mezcla de los dos anteriores.

Los codificadores híbridos son de calidad media/alta (2,4-16 kbps).

En este proyecto nos vamos a centrar en los Vocoders, concretamente en los Vocoders LPC.

2.3 Vocoder LPC (Codificación por Predicción Lineal)

Es el vocoder más simple y a su vez el de peor calidad, generalmente muestrea la señal a 8000 Hz y utiliza tramas de 30 ms (240 muestras).

Este codificador analiza la señal de voz de la que extrae 3 tipos de parámetros:

- Coeficientes LPC (Relacionados con los formantes: F1, F2, …).
- Decisión de sonido sordo o sonoro, en caso de ser sonoro, frecuencia fundamental (F0 o pitch).
- Factor de ganancia.

Una vez extraídos los parámetros, se envían al decodificador para que sintetice la señal de voz [7].

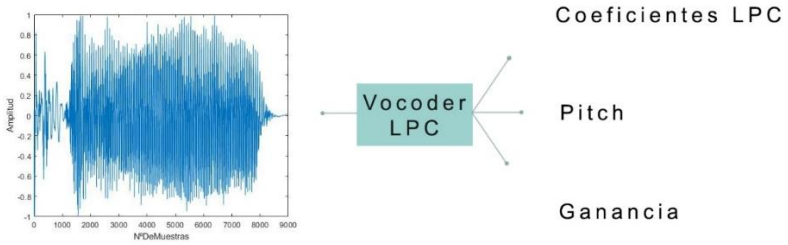

Ilustración 6: Vocoder LPC

2.4 Coeficientes LPC

Los coeficientes LPC se pueden obtener resolviendo un sistema de ecuaciones, y la herramienta “Matlab” define una función que facilita su cálculo.

La manera más sencilla de usar dicha función en Matlab es [a,e]=lpc(s, N); donde “a” son los coeficientes LPC, “e”, la potencia de la señal de error, “s” es un vector con la señal de la que se desea obtener los coeficientes del filtro de predicción lineal (en este caso, cada trama de la señal de voz), y “N” es el orden del filtro de predicción lineal.

El inverso del sistema LPC es el filtro que permitirá sinterizar la voz a partir de la señal de excitación [8].

2.4.1 Filtro Inverso LPC

Este filtro se trata de un filtro con todo polos, que modula el espectro de la señal de excitación, simulando las resonancias que producen la cavidad y el tracto vocal [9].

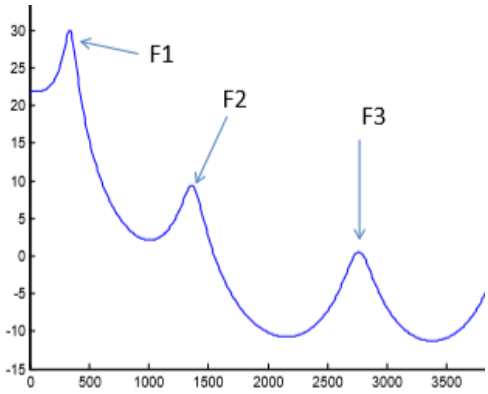

Ilustración 7: Filtro de síntesis

Se suelen apreciar ciertas resonancias que se asocian con las frecuencias de los formantes del habla.

F1 tiene una frecuencia más alta cuanto más baja está la lengua; es decir, cuanta mayor abertura tenga una vocal, mayor es la frecuencia en que aparece el F1.

El F2 tiene mayor frecuencia cuanto más hacia delante está posicionada la lengua, es decir, cuanto más anterior es una vocal, mayor es el F2.

2.5 Estimación del Pitch

El pitch se trata de la frecuencia fundamental, en este caso, de cada trama de sonido analizada.

La frecuencia fundamental es la frecuencia más baja de una forma de onda periódica.

La estimación de la frecuencia del pitch la realiza el codificador, analizando la señal de error de predicción (que se puede demostrar que es una señal periódica con la misma periodicidad de la señal de voz).

En caso de analizar un sonido sonoro, una sencilla estrategia para estimar el pitch es calcular donde se encuentra el máximo de la señal de autocorrelación del error de predicción. Este máximo será el periodo del pitch (T).

Teniendo en cuenta nuestra frecuencia de muestreo utilizada podremos obtener la frecuencia del pitch [9].

Normalmente esta frecuencia se encuentra entre: 60Hz y 300Hz.

La decisión de sonido sordo o sonoro se dará utilizando también la señal de autocorrelación del error de predicción, si el máximo de la autocorrelación normalizada supera valores de amplitud de 0.20, por recomendación, se considerará sonora, en caso contrario se considera sorda.

2.6 Factor de Ganancia

El codificador, se encarga también de calcular la ganancia a partir de la siguiente fórmula:

$$G = \sqrt{\frac{P_e}{P_{trenimpulsos}}} = \sqrt{\frac{P_e}{1/T}}$$

Esta ganancia será utilizada por el sintetizador para determinar la amplitud de los impulsos de la señal de excitación.

En caso de sonido sonoro, esta señal de excitación se trata de un tren de impulsos modulado en amplitud según la ganancia y cuyo periodo es igual al del pitch.

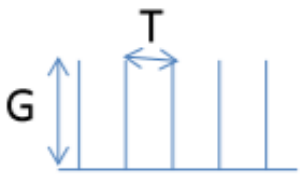

Ilustración 8: Señal de excitación

En caso de sonido sordo, la ganancia se calcula como:

$$G = \sqrt{P_e}$$

Y respecto a la señal de excitación no es más que un ruido blanco, pues al ser plano espectralmente se asemeja mucho a la función que realiza el aire de los pulmones, además de que el ruido blanco tiene la misma potencia que la señal de error de predicción.

2.7 Decodificador LPC

El decodificador o sintetizador LPC es el encargado de generar la señal de voz, recibiendo el periodo del pitch (T) en caso de sonido sonoro, la ganancia para compensar la energía de la señal de excitación a sintetizar y los coeficientes del filtro LPC.

El esquema para su construcción es el siguiente:

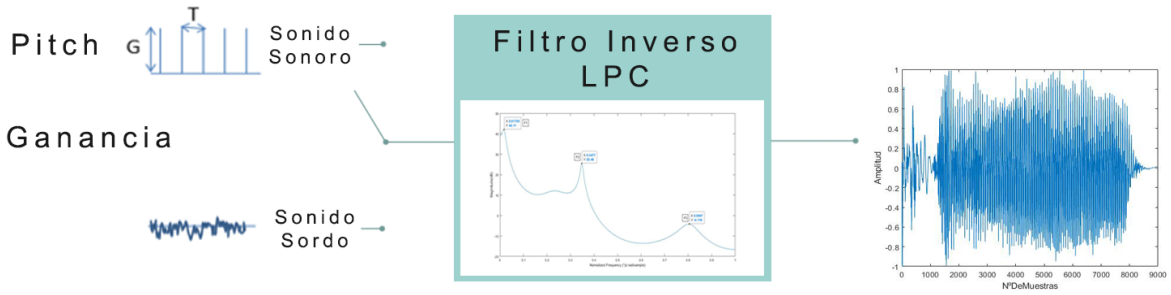

Ilustración 9: Decodificador LPC

Como vemos, el sistema es muy parecido a lo visto anteriormente con el sistema vocal.

En caso de sonido sonoro se recibe la ganancia para compensar la señal de excitación y se utiliza el periodo del pitch para sintetizar un tren de impulsos del mismo periodo que la señal vocal analizada.

En caso de sonido sordo, la ganancia será utilizada para ponderar el ruido blanco.

En ambos casos la señal pasará a través del filtro inverso LPC, el cual determina las resonancias a modo de tracto vocal, y finalmente se generará la voz sintetizada [8].

Capítulo 3: DESARROLLO PRÁCTICO

3.1 Codificador LPC

En primer lugar, antes de empezar el desarrollo del codificador, hay que tener claro los objetivos y necesidades de este, ya que los parámetros con los que trabaja ya los conocemos.

En este proyecto se ha propuesto implementar un codificador de voz paramétrico configurable en tamaño de trama y solape, el cual se encuentra en la carpeta de ficheros anexos y en Matlab se verá de la siguiente forma.

```
function [A,P,G] = C_Coder(s,TamTrama,Solape)
```

Donde A, P y G contienen los coeficientes LPC, el pitch y la ganancia respectivamente, y s, la señal del audio original.

Cabe mencionar que tanto el tamaño de trama como el solape se introducen en número de muestras (no en ms).

3.2.1 Tamaño de trama

Como hemos visto anteriormente, el vocoder LPC generalmente muestrea la señal a 8000 Hz y utiliza tramas de 30 ms (240 muestras).

Esto quiere decir que nuestro audio a codificar será dividido en tramas del tamaño indicado para así poder analizar cada una de ellas y obtener sus parámetros para la sintetización (pitch, ganancia y coeficientes LPC).

Por tanto, para poder calcular el número de tramas a codificar utilizaremos la siguiente expresión:

$$Nº De Tramas = \frac{Duración\ Del\ Audio}{Tamaño\ De\ Trama}$$

A continuación, vemos el ejemplo de la codificación de un audio de 2 segundos de duración (16000 muestras utilizando frecuencia de muestreo de 8000 Hz) utilizando tamaño de trama de 240 muestras.

La primera trama analizaría las primeras 240 muestras, la segunda trama de 241 a 480 y así sucesivamente.

Se utiliza enventanado hamming con el fin de minimizar la energía de los componentes tonales fuera del componente principal.

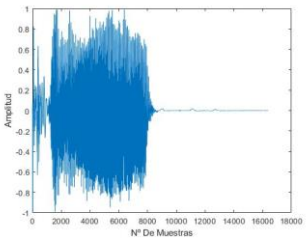

Ilustración 10: Señal original

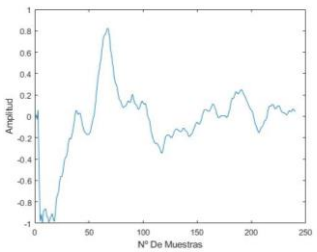

Ilustración 11: 1ª Trama (240 Muestras)

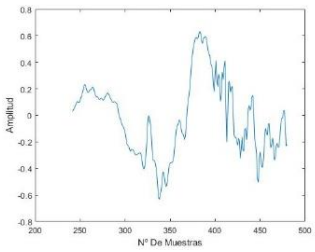

Ilustración 12: 2ª Trama (240 Muestras)

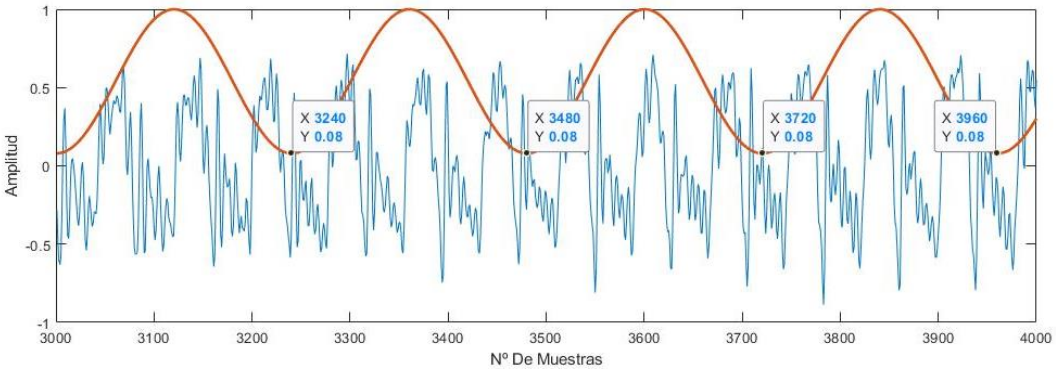

Ilustración 13: Codificación TamTrama 240

La razón de codificar el audio por tramas es la de obtener mucha más información para sintetizarlo con mayor calidad, pues cuando estamos hablando, tanto el tono como las resonancias o la fuerza del aire de los pulmones están cambiando constantemente, mientras que en la codificación tienen valores fijos para cada trama. Por tanto, con un tamaño de trama como 30 ms podemos conseguir que nuestros parámetros cambien cada poco tiempo simulando el habla con una calidad óptima.

El motivo de que 30 ms sea óptimo para utilizarlo como tamaño de trama es debido a que la voz se considera estacionaria en este periodo [11].

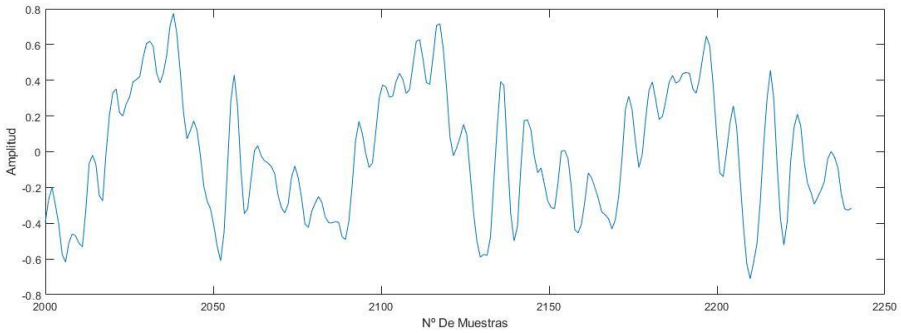

Ilustración 14: Periodicidad de la voz

Se puede llegar a conseguir una mejora en la calidad de la sintetización de la voz utilizando un tamaño de trama menor entre 15-30ms, sin embargo, si se lleva al límite utilizando tamaños de trama menores a 15ms el codificador no tiene suficiente

información para tomar las decisiones correctamente y empieza a generar errores, produciendo una señal completamente distorsionada.

Todo esto lo veremos más adelante en el punto 3.4.

3.2.2 Solape

El solape entre tramas indica la cantidad de muestras con la que las tramas se solaparán para el análisis.

El objetivo del solapado es el de mejorar simultáneamente la resolución temporal y frecuencia del análisis, es decir, conseguir una mayor calidad en la sintetización de la voz, pues se analiza más información múltiples veces en el codificador para obtener unos parámetros más precisos, mientras que en el sintetizador se sintetizan menos muestras con estos parámetros.

Para calcular el número de tramas a codificar teniendo en cuenta el solapado, utilizaremos la siguiente expresión.

$$Nº De Tramas = \frac{Duración\ Del\ Audio - Solape}{Tamaño\ De\ Trama - Solape}$$

En la ilustración 15 tenemos el ejemplo de una codificación con tamaño de trama de 240 muestras (30ms) con un solape del 50% (120 muestras).

Como podemos observar en la codificación las tramas analizan 240 muestras, con un desplazamiento temporal de la ventana de 120 muestras.

De cada trama de 240 muestras se obtendrán los 3 parámetros necesarios para la síntesis, que serán utilizados para sintetizar 120 muestras de voz.

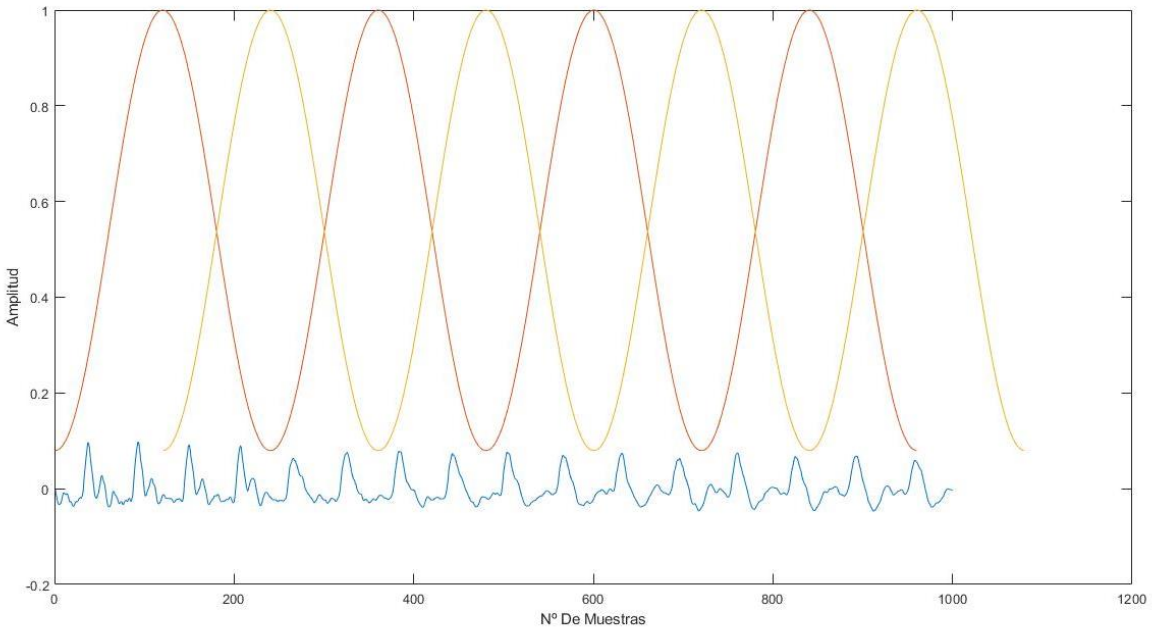

Ilustración 15: Solape 120 muestras (50%)

3.2.3 Coeficientes LPC

Una vez hemos determinado el número de tramas a codificar, podemos empezarnos a plantear como obtener los parámetros necesarios para cada una de ellas.

Esto se realiza mediante un bucle for, que en Matlab podría verse de la siguiente forma:

```
for TramaActual=1:N°DeTramas
```

A continuación, es imprescindible obtener las muestras correspondientes a dicha trama, para así analizarla con el fin de calcular todos los parámetros.

```
entrada=s(1+(TamTrama-Solape)*(TramaActual-1):(TamTrama-Solape)*(TramaActual-
1)+TamTrama);
```

Como hemos visto anteriormente para calcular los coeficientes LPC es necesario resolver un sistema de ecuaciones, pero Matlab nos lo facilita con la siguiente función:

```
[a,e]=lpc(entrada,10);
```

Donde “a” son los coeficientes del filtro LPC y “e” la potencia de la señal del error de predicción.

Hay que decir que se utilizan filtro de orden 10 debido a que normalmente, por cada 1 Khz de ancho de banda se produce una resonancia, considerando un ancho de banda de 4KHz (fs=8KHz), tendremos 4 resonancias en las frecuencias positivas y 4 más en las frecuencias negativas, por tanto, se utiliza 10 como orden del filtro dejando un par de resonancias como margen de libertad.

En este momento ya disponemos de los coeficientes LPC para poder realizar el filtro inverso que es el que nos permitirá sintetizar la trama de voz posteriormente.

Para poder visualizarlo lo podemos hacer de la siguiente manera:

```
[H,W]=freqz(1,a,512);
plot(W/pi*4000,20*log10(abs(H)))
```

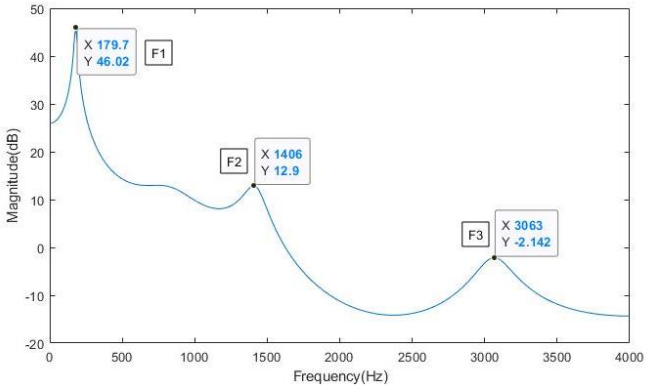

Ilustración 16: Filtro de síntesis

Debemos poder apreciar cómo se forman ciertas resonancias correspondientes a los formantes de la voz.

3.2.4 *Pitch*

Como ya sabemos, para poder estimar el Pitch de cada trama se necesita la señal de error de predicción, la cual la podemos obtener de la siguiente forma:

```
errordeprediccion=filter(a,1,entrada);
```

Para poder determinar el periodo de esta señal de una manera sencilla utilizamos la correlación normalizada:

```
C=xcorr(errordeprediccion,133);
Cxx=C(133+1:2*133+1)/C(134);
```

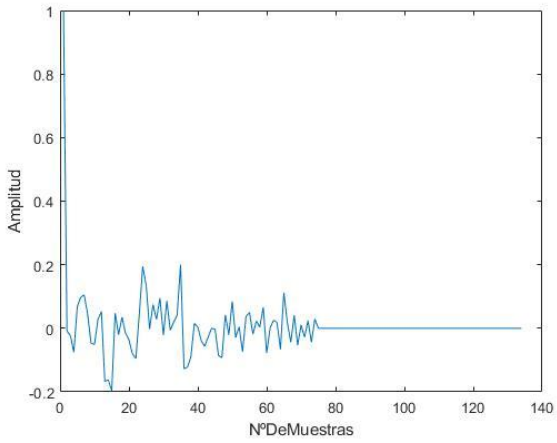

Ilustración 17: Correlación

Como el pitch se considera que está entre 60 y 300 Hz (es decir, periodos entre 134 muestras y 26 muestras) tanto las muestras de antes como las de después no son representativas de ningún pitch y por eso no se consideran.

```
Cxx(1:26)=0;
```

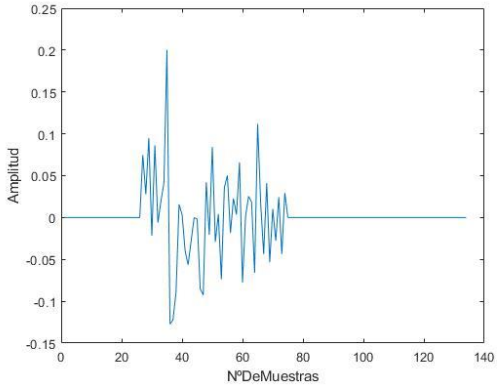

Ilustración 18: Correlación

Obtenemos el máximo y la posición del mismo, M lo utilizaremos para umbralizar y determinar si la trama de sonido es sorda o sonora y I corresponderá al periodo del pitch.

```
[M,I]=max(Cxx);
```

Con este umbral determinamos si la señal es sonora.

```
if (M>0.20)
```

En ese caso obtenemos el periodo del pitch con el método que estamos utilizando.

```
T=(I-1);
```

Finalmente, para calcularlo en unidades de frecuencia (Hz), tendremos que dividir la frecuencia de muestreo utilizada (en este caso 8000 Hz) entre dicho periodo.

```
P(TramaActual)=8000/T;
```

En caso de no superar el umbral predeterminado, la trama de sonido se considera sorda y no es necesario realizar nada más al respecto, ya que el pitch se codificará como 0 indicándonoslo.

3.2.5 Ganancia

Para calcular la ganancia de cada trama utilizamos la siguiente expresión, como ya hemos visto:

$$G = \sqrt{\frac{P_e}{P_{trenimpulsos}}} = \sqrt{\frac{P_e}{1/T}}$$

En Matlab se vería de la siguiente manera (teniendo en cuenta el nombre de las variables anterior):

```
G(TramaActual)=sqrt(e)/sqrt(1/T);
```

Esto sería en el caso de que la trama fuera sonora, si fuera sorda, se calcularía de la siguiente forma:

```
G(TramaActual)=sqrt(e);
```

3.3 Sintetizador LPC

Una vez el codificador ha obtenido todos los parámetros necesarios para sintetizar la voz, vamos a ver como se realiza el proceso.

Vamos a poder encontrar el sintetizador desarrollado en este proyecto en la carpeta anexos y se verá de la siguiente manera.

```
function [s] = C_Sinte(A,P,G,TamTrama,Solape)
```

Como ya sabemos, el sintetizador utiliza una señal de excitación, a modo de flujo de aire.

Esta señal de excitación no se trata más que de un tren de impulsos cuyo periodo es igual al del pitch, en caso de ser una trama sonora.

```
aux=[ones(1,TamTrama-Solape);zeros(T-1,TamTrama-Solape)];
e=G(TramaActual)*aux(1:TamTrama-Solape);
```

Para tener una mayor calidad y evitar efectos de robotización en la voz, podemos considerar el offset entre tramas para la generación del tren de impulsos de la siguiente manera:

```
aux=[zeros(ofset,1);aux(:)];
ofset=ofset+(T-rem((TamTrama-Solape),T));

if ofset>T
ofset=ofset-T;
end
```

De esta forma, tendremos en cuenta la fase del tren de impulsos.

Si fuese sorda, al ser un sonido principalmente ruidoso, se utilizaría ruido blanco como señal de excitación. Esto lo podríamos lograr de la siguiente manera:

```
e=G(TramaActual)*randn(TamTrama-Solape,1)
```

Podemos determinar si una trama es sonora o sorda, simplemente analizando los valores del Pitch, si es 0 será una trama sorda, y si tiene cualquier otro valor será sonora.

```
if P(TramaActual)==0
else
```

Una vez tenemos la señal de excitación, para sintetizar la señal de voz, se filtra esta señal por el filtro inverso LPC.

```
[y,zf]=filter(1,A(:,TramaActual),e,zf);
```

Cabe mencionar la utilización de los estados del filtro para garantizar el correcto filtrado entre las transiciones entre tramas.

Por último, ponemos las muestras sintetizadas en su sitio correspondiente teniendo en cuenta el tamaño de trama y el solape utilizado.

```
s(1+(TamTrama-Solape)*(TramaActual-1):(TamTrama-Solape)*TramaActual)=
                                    y;
```

3.4 Calidad Señal Sintetizada

En este punto vamos a comparar de forma objetiva la calidad entre la señal original y la señal sintetizada, así como los problemas o mejoras que suceden al aumentar o disminuir el tamaño de trama y solape.

En primer lugar, hay que saber que los Vocoders LPC no son los de mejor calidad, pues generalmente, muestrea la señal a 8000 Hz.

A continuación, en las siguientes imágenes vamos a ver el parecido entre dichas señales en el tiempo.

Para ello se ha utilizado una codificación y síntesis con 240 muestras como tamaño de trama (30 ms) y 0 muestras como solape (0ms).

Se puede realizar con los archivos de Matlab adjuntos a este proyecto de la siguiente forma:

```
[A,P,G]=C_Coder(audio,240,0);
sintesis=C_Sinte(A,P,G,240,0);
```

Y graficando los resultados.

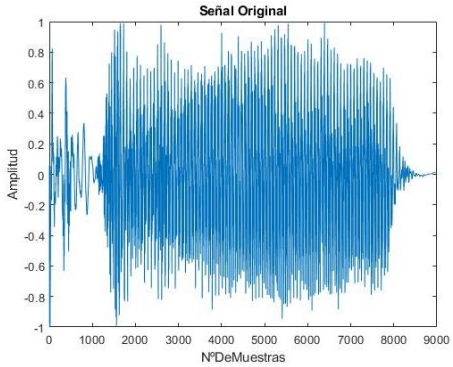

Ilustración 19: Señal original

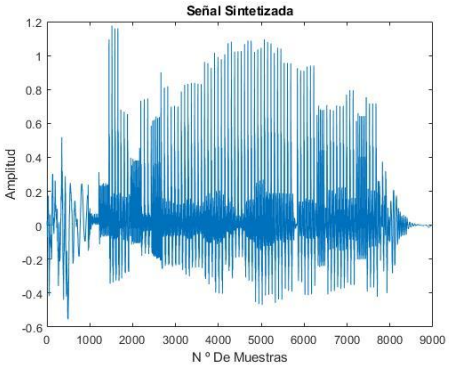

Ilustración 20: Señal sintetizada

Destacamos que la señal sintetizada no es realmente parecida a la original, pero esto es debido a que los Vocoder LPC no se especializan por replicar la señal perfectamente en el tiempo, sino que más bien, tratan de que se parezcan sus componentes frecuenciales.

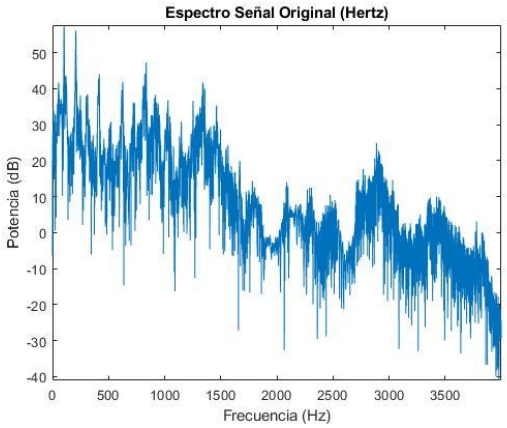

Ilustración 21:Espectro señal original

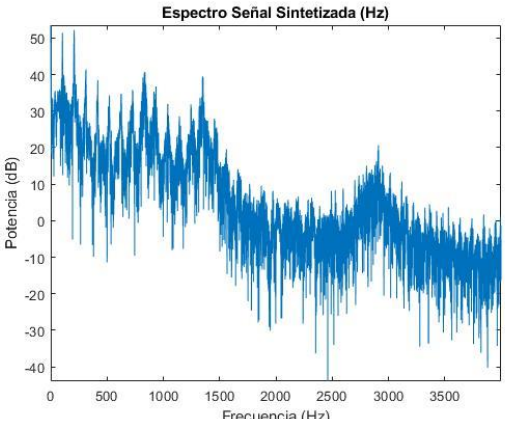

Ilustración 22: Espectro señal sintetizada

Como vemos las señales en frecuencia ya tienen mucho más parecido una con otra, ya que utilizan el Pitch y las resonancias LPC, las cuales vemos que también se parecen.

Cabe mencionar que los espectros se corresponden a la respuesta en frecuencia de la señal total.

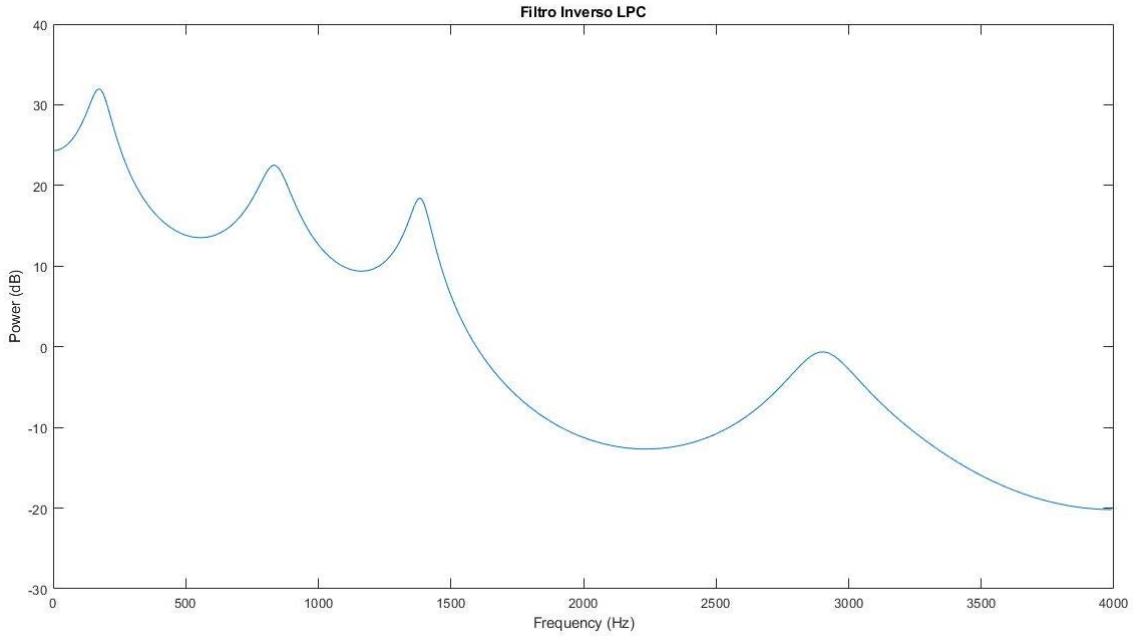

Ilustración 23: Filtro Inverso LPC

Podemos concluir con que con esta configuración se obtiene una calidad bastante buena de la voz sintetizada teniendo en cuenta las propias limitaciones de los Vocoder LPC.

En los ficheros anexos podremos escuchar este audio como “lafabTamTrama240.wav”.

Se puede mejorar la calidad de la sintesis de audio al reducir un poco el tamaño de trama, pero ahora vamos a comprobar que ocurre en los dos casos extremos, cuando el tamaño de trama es muy pequeño y cuando es muy grande.

Para este primer ejemplo hemos utilizado un tamaño de trama de 60 muestras (7,5ms) y un solape de 0 muestras (0ms).

En los ficheros anexos podremos escuchar este ejemplo como “lafabTamTrama60.wav”.

```
[A,P,G]=C_Coder(audio,60,0);
sintesis=C_Sinte(A,P,G,60,0);
```

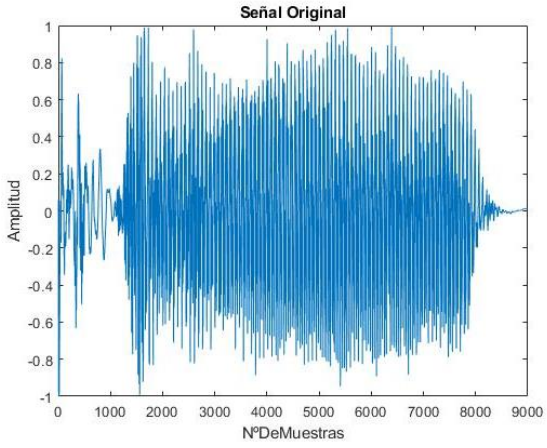

Ilustración 24: Señal original

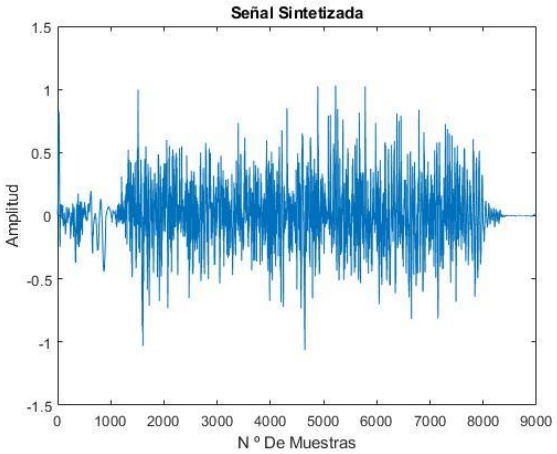

Ilustración 25: Señal sintetizada

En un primer momento ya podemos observar cómo se han perdido muchas componentes, pues el codificador empieza a tener poca información para codificar la voz de una forma correcta perdiendo resolución frecuencial.

De forma objetiva se puede escuchar como la voz sigue siendo entendible, pero su sonido ha sido completamente distorsionado, así como la pérdida de su tonalidad.

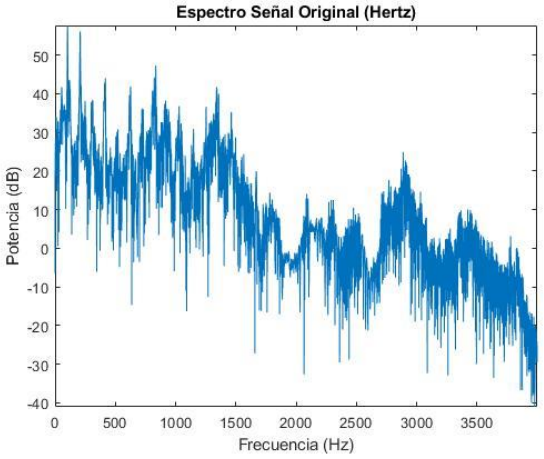

Ilustración 26: Espectro señal original

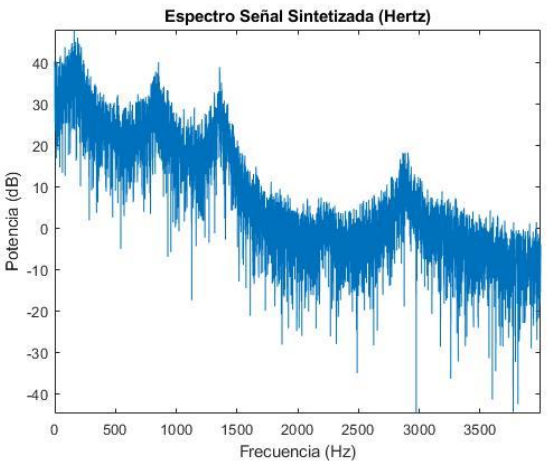

Ilustración 27: Espectro señal sintetizada

Podemos apreciar como el espectro sintetizado sigue teniendo similitud con el original pero es cada vez más ruidoso y menos definido.

En el segundo caso extremo, consideramos que el tamaño de trama sea muy grande, para este ejemplo se ha utilizado un tamaño de trama de 960 muestras (120ms) y un solape de 0 muestras (0ms).

En los ficheros anexos lo podremos encontrar como “lafabTamTrama960.wav”.

```
[A,P,G]=C_Coder(audio,960,0);
sintesis=C_Sinte(A,P,G,960,0);
```

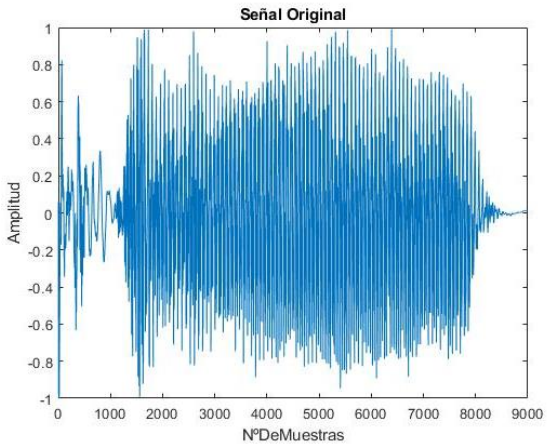

Ilustración 28: Señal original

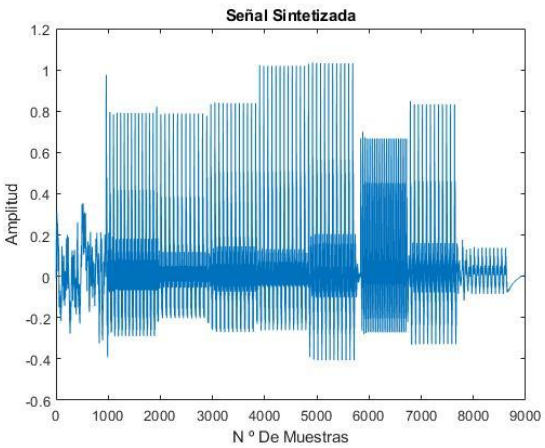

Ilustración 29: Señal sintetizada

Se ve claramente en la señal sintetizada que sus componentes tienen forma escalonada, aproximadamente del mismo tamaño que las tramas.

Esto es debido a que durante esas 960 muestras (120ms) todos los parámetros tendrán un valor fijo, el correspondiente a la codificación de la trama, mientras que los parámetros de la voz humana cambian en todo momento, perdemos la resolución temporal de la señal de voz.

No es de extrañar que también por ese motivo se logren entender pocas palabras, pues el cambio no es suficientemente rápido como para definirlas correctamente.

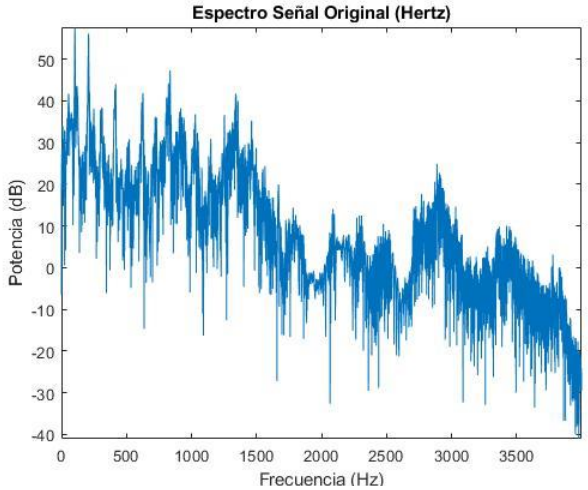

Ilustración 30: Espectro señal original

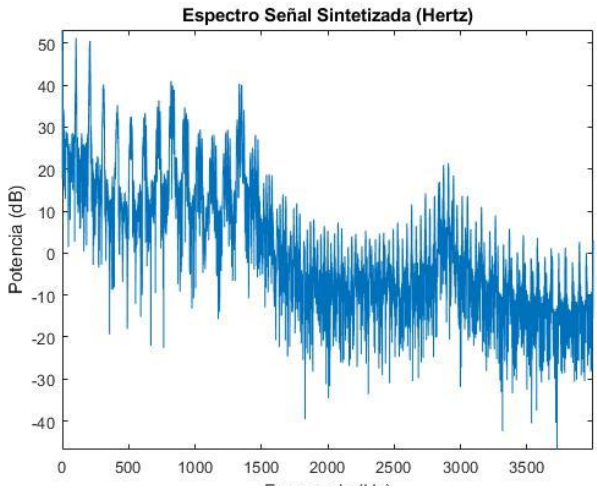

Ilustración 31: Espectro señal sintetizada

En cuanto al espectro, principalmente se observa como las frecuencias del pitch y los formantes están bastante afectadas, y el espectro se vuelve más ruidoso.

Como efecto objetivo se aprecia que la voz se vuelve claramente robótica.

Ahora vamos a ver qué ocurre con el solape:

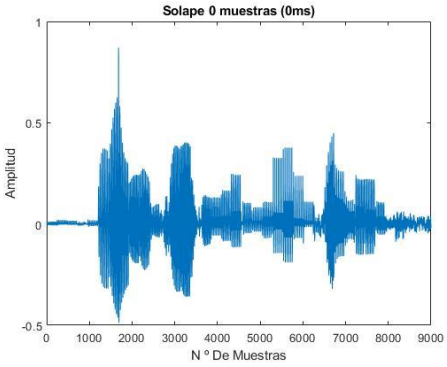

Ilustración 32: Solape 0 muestras ("lafabSolape0.wav")

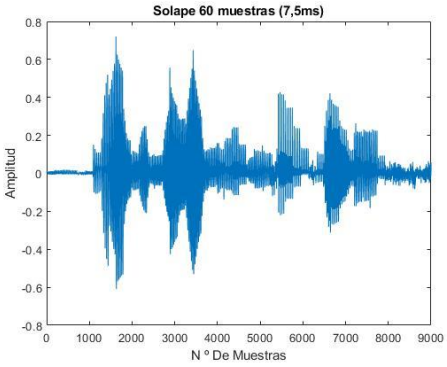

Ilustración 33: Solape 60 muestras ("lafabSolape60.wav")

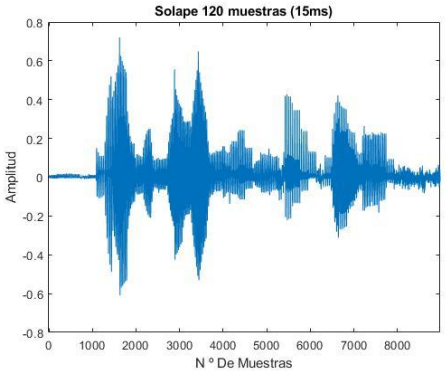

Ilustración 34: Solape 120 muestras ("lafabSolape120.wav")

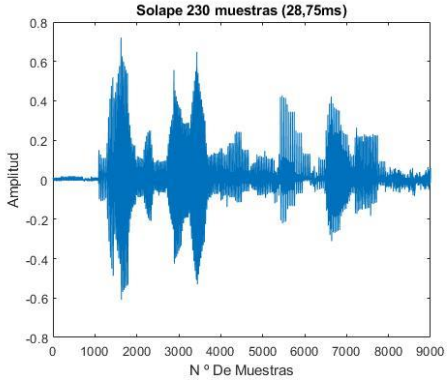

Ilustración 35: Solape 230 muestras ("lafabSolape230.wav")

Como podemos observar claramente, al aumentar el solape la señal gana definición, reduciendo en gran medida los cambios bruscos, facilitando las transiciones entre sonidos sordos y sonoros.

Respecto al audio escuchamos una señal con mayor calidad de sonido y menos robotizada.

Para sintetizar una trama de voz se necesita cierta estacionariedad, y si el solape es muy grande, la síntesis se realiza con pocas muestras, y como consecuencia es más variante de lo que realmente es la voz.

Lo normal para conseguir una buena calidad trabajando a 8 KHz, es usar tramas de análisis de 30ms (240 muestras) y un solape entre 7,5ms (60 muestras) y 20ms (160muestas).

3.5 Efectos

En este proyecto se han desarrollado dos efectos que se encuentran en la carpeta de archivos de Matlab.

Estos efectos consisten en una modificación intencionada del pitch de la voz haciendo que suba y baje durante un determinado tiempo y un modificador de las resonancias principales que caracterizan la voz humana.

3.5.1 Efectos con el Pitch

Uno de los efectos programados, realiza una subida y una bajada del pitch a partir de un incremento, la subida se realiza por defecto hasta la mitad de la longitud del audio, en este momento sucede la bajada del pitch.

Este incremento se aplica a cada trama sonora y se va acumulando para ser utilizado por las tramas futuras.

En los archivos adjuntos de Matlab lo podemos utilizar de la siguiente manera:

```
[Pitch] = C_FxP(A,P,G,2)
```

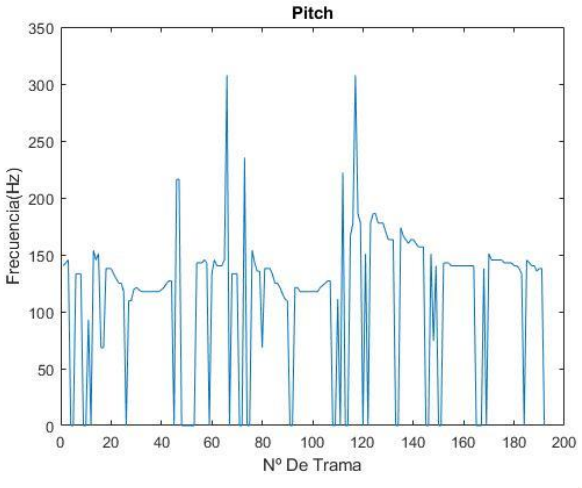

Ilustración 36: Pitch sin efecto

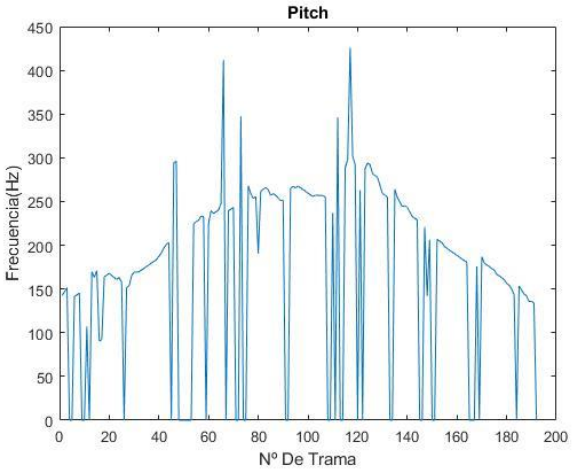

Ilustración 37: Pitch con efecto

Como vemos en la ilustración 36, el pitch corresponde al de la voz sintetizada y tras pasar por el módulo de efecto, se produce un incremento continuo del mismo hasta la mitad del audio, donde empieza a reducirse hasta llegar al pitch original sin modificación.

Esto simplemente produce que a medida transcurre el tiempo, el tono de la voz es modificado sonando más agudo y luego más grave.

Podremos encontrar un ejemplo utilizando este efecto en los ficheros anexos como “lafabFxP.wav”

Parece algo sencillo, y lo es, pero es la base de efectos más complejos y con tiempo de desarrollo, como podría ser Autotune, o infinidad de afinadores vocales.

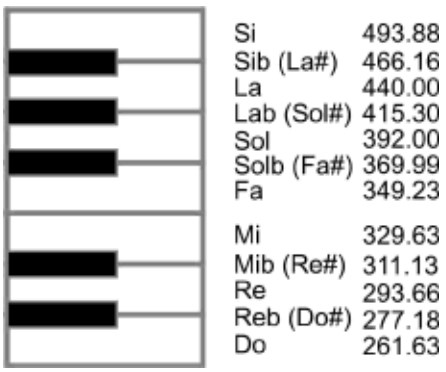

Ilustración 38: Frecuencia de las notas musicales (4ª octava)

Pues si esta frecuencia de pitch, en vez de modificarla aleatoriamente, la modificamos en base a la frecuencia de las notas musicales, podemos crear melodías con cualquier voz que no tenga musicalidad en sí misma.

También se podría utilizar en voces de cantantes de forma que teniendo en cuenta la escala musical en la que se ejecute su cante, autocorrija su voz si no ha llegado a entonar bien alguna de las notas.

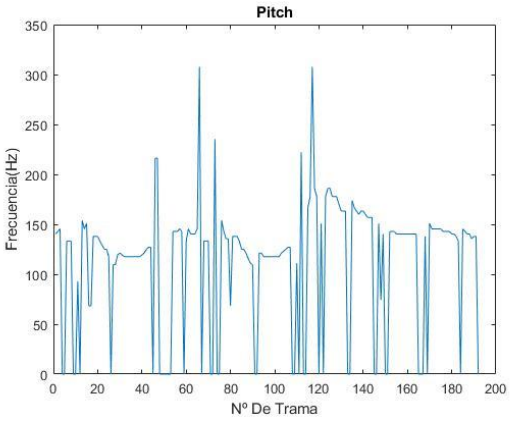

Ilustración 39: Pitch sin efecto

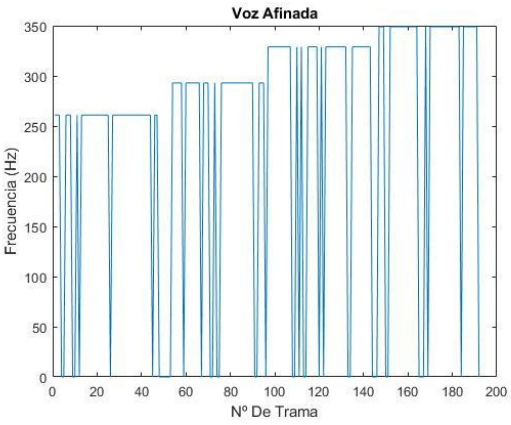

Ilustración 40: Pitch afinado

En la ilustración 39 vemos como sería el pitch de la voz normal, mientras que en la 40 se ha afinado en función de la frecuencia del Do (261.63 Hz), Re (293.66 Hz), Mi (329.63Hz) y Fa (349.23 Hz).

Este efecto lo podemos lograr utilizando el fichero de Matlab anexo a este proyecto de la siguiente forma:

```
[Pitch] = C_FxMelodia(A,P,G, 261.63, 293.66, 329.63, 349.23)
```

Esto genera el típico efecto Autotune con el cual la voz suena completamente afinada a las notas musicales pero robótica a su vez, pues no es algo natural.

Podremos encontrar un ejemplo en los ficheros anexos como “lafabDoReMiFa.wav”.

También se ha programado el sintetizador con funcionalidad de pitch fijo, modificando el pitch de todas las tramas a un único valor frecuencial.

Esto lo podemos realizar de la siguiente manera:

```
[s] = C_Sinte(A,293,G,TamTrama,Solape)
```

Se puede ver como al modificar el pitch de la señal a Re (293.66 Hz, en este caso), el pitch se desplaza, creando con ello harmónicos múltiplos a esta frecuencia.

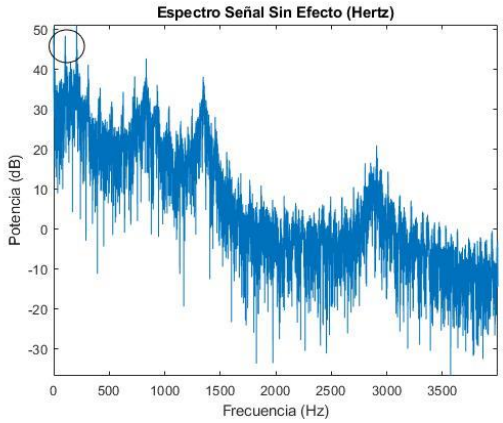

Ilustración 41: Espectro señal sin efecto

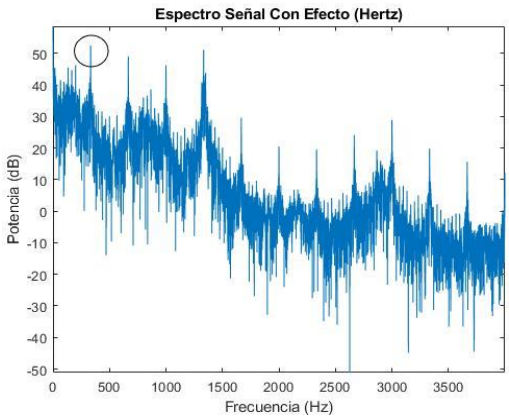

Ilustración 42: Espectro señal con efecto

Este efecto lo encontraremos como “lafabPitchFijo293.wav”.

El cambio de pitch a una señal de voz se suele utilizar con la finalidad de modificar su tonalidad para poder conseguir que las voces, sonidos vocales o sonidos en general, puedan asemejarse a distintas melodías sin tener que ser cantadas o interpretadas por algún músico.

Otra forma de modificar el pitch de una forma más natural es aplicándole un incremento que se sume o reste al pitch de la señal, subiendo o bajando el tono de la voz.

Se puede realizar con la siguiente función.

```
[Pitch] = C_ModificarPitch(A,P,G,10)
```

Esto provoca un efecto parecido al pitch fijo pero menos robotizado.

Este tipo de efectos, junto con modificaciones de los coeficientes LPC, se suelen utilizar para convertir una voz en aguda o grave, con el fin de que sea irreconocible respecto a la original, por ejemplo, para dar anonimato al testimonio grabado de una persona.

Podremos encontrar este último ejemplo como “lafabModPitch.wav”.

3.5.2 Efectos con el filtro LPC

Llegados a este punto se ha realizado un efecto que modifica los polos del filtro LPC, es decir, modifica la frecuencia de las resonancias de la voz.

Con este efecto podemos modificar las características de la voz y funciona de la siguiente manera:

Si la trama es sonora, en un primer momento se obtienen las raíces del polinomio LPC.

```
raices=roots(A(:,TramaActual));
```

Entonces obtenemos sus módulos y fases.

```
modulos=abs(raices);
fases=angle(raices)
```

En este momento se busca la primera resonancia para poder modificar su fase, con lo que estamos modificando la frecuencia de la primera resonancia del filtro LPC.

```
M=min(fases(fases>0))
I=find(fases==M)
```

En este momento ya podemos aplicar una modificación sobre la fase.

```
fases(I)=fases(I)+incrementofase;
fases(I+1)=fases(I+1)-incrementofase;
```

El funcionamiento para el resto de las resonancias es similar, solo que esta vez la posición en frecuencia de la segunda resonancia tendrá que ser mayor a la anterior, pues no es la primera.

```
M2=min(fases(fases>M))
I2=find(fases==M2)
```

Por último, una vez aplicadas las modificaciones en fase necesarias hay que volver a generar el polinomio LPC y almacenarlo para poder utilizarlos en la síntesis, que es donde se generará la voz con el efecto deseado.

```
senal=modulos.*exp(i*fases);
pol=real(poly(senal));

LPC(:,TramaActual)=pol;

audio=C_Sinte(LPC,P,G,240,0);
```

Al igual que podemos visualizar el espectro del filtro LPC como ya hemos visto, también se puede graficar sus polos y ceros utilizando la siguiente función:

```
zplane(1,a);
```

En las siguientes imágenes graficamos de ambas formas los coeficientes del filtro LPC, en las primeras podemos ver como serían sin aplicarles ninguna modificación.

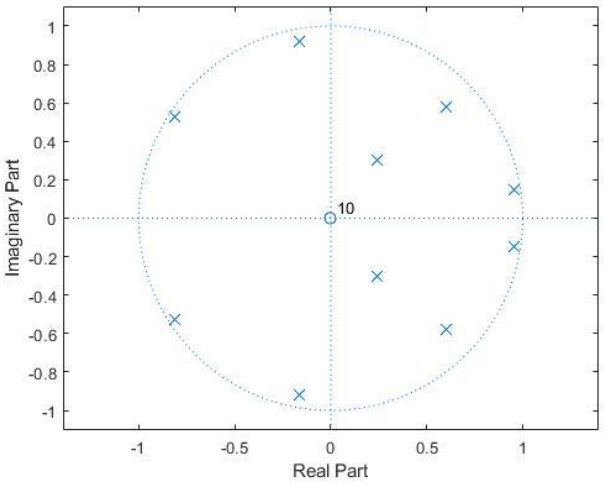

Ilustración 43: Plano Z sin modificación

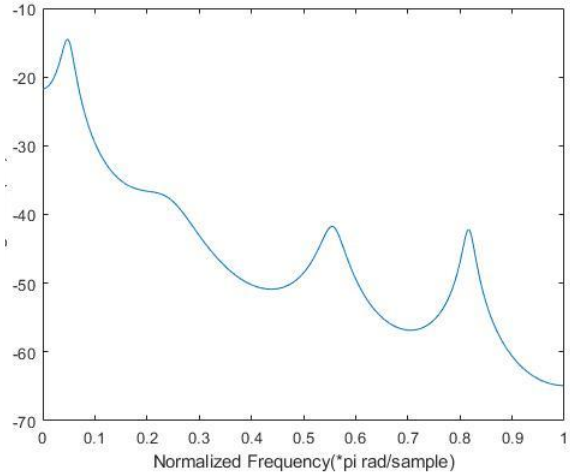

Ilustración 44: Filtro LPC sin modificación

Tras obtener el módulo y la fase de cada polo en plano Z y poder modificarlos, se ha aplicado un desplazamiento al primero de ellos, provocando que los Formantes 1 y 2 de la voz estén más unidos.

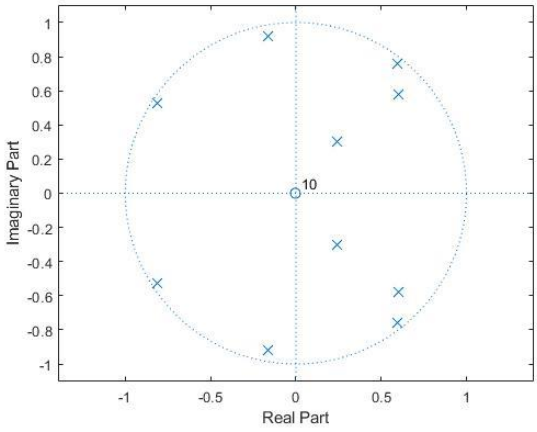

Ilustración 45: Plano Z con modificación

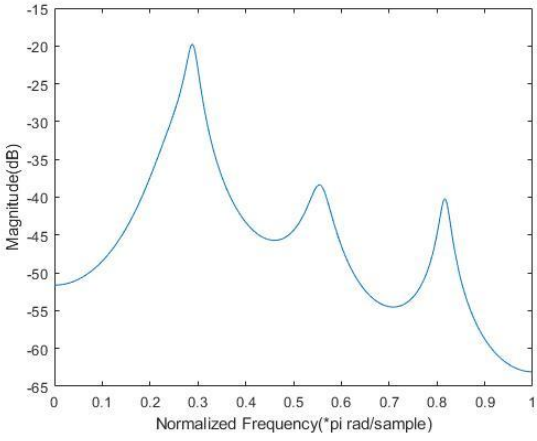

Ilustración 46: Filtro LPC con modificación

Cabe destacar que este proceso se aplica a cada trama, pero estamos graficando solo una de ellas para poder verlo y entenderlo mejor.

Como podemos esperar si graficamos el espectro en frecuencia de la señal sin efecto y con efecto, vamos a poder ver claramente un desplazamiento de los formantes de la voz modificados en función del nuevo filtro LPC, así como otras diferencias.

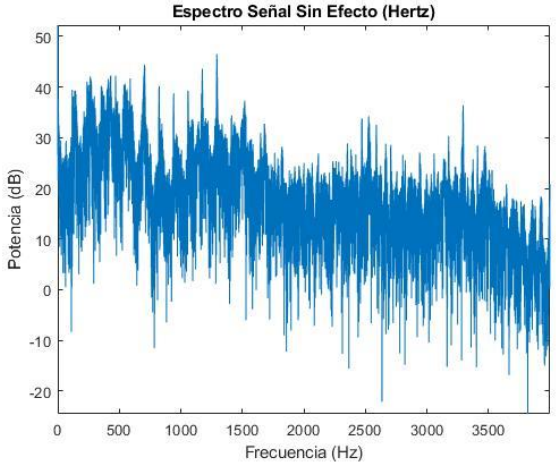

Ilustración 47: Espectro de la señal sin efecto

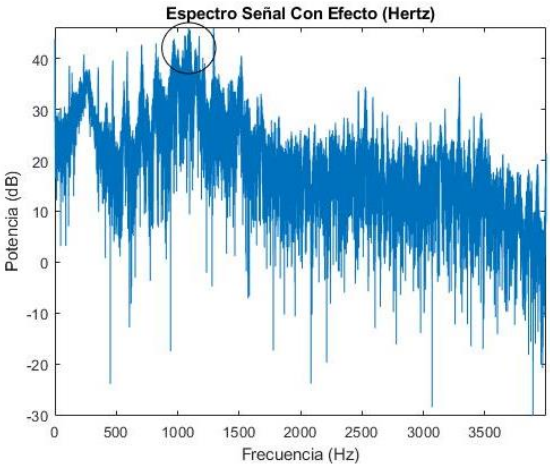

Ilustración 48: Espectro de la señal con efecto

Esto aumenta las características del primer formante volviendo la voz un poco más aguda si se realiza de una forma leve.

Podemos encontrar un ejemplo de la utilización de este efecto en ficheros anexos como “lafabModPolo1.wav”

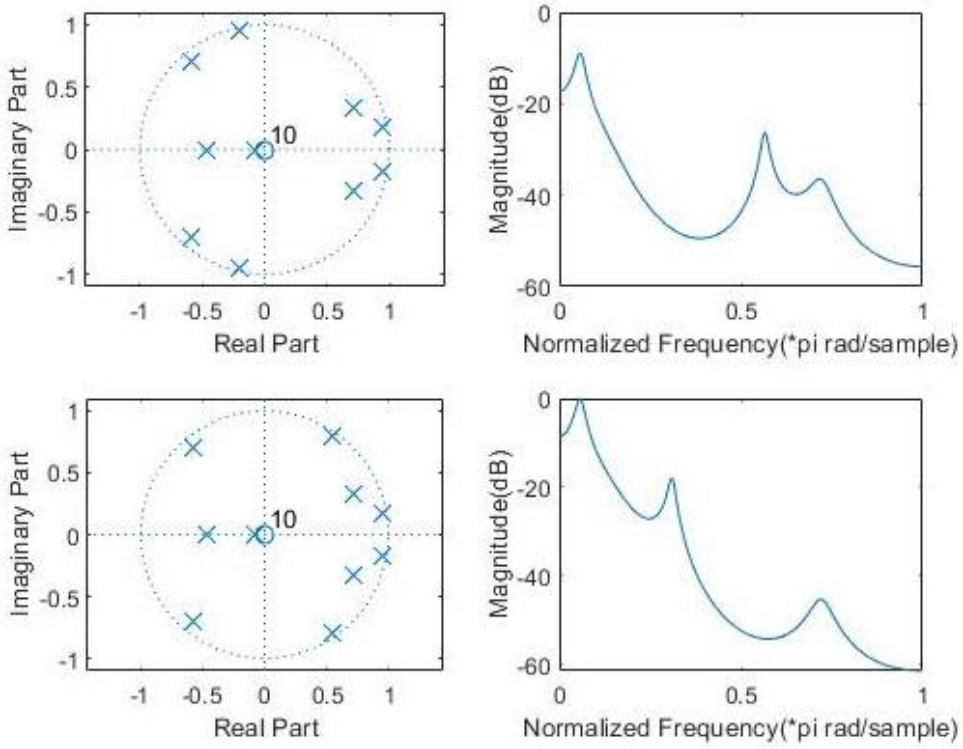

Ilustración 49: Modificación de la segunda resonancia (“lafabModPolo2.wav”)

Si modificamos la frecuencia del segundo formante y lo llevamos más cerca del primero, notamos como la voz suena más grave y con más cuerpo, pero también llegando a provocar distorsión en la voz.

Con los dos primeros formantes se pueden llegar a obtener infinidad de efectos si se busca un objetivo artístico, pues son los más importantes en la formación de la voz.

Respecto al tercero y cuarto se suelen relacionar con cualidades del color de la voz y brillo, aunque en ciertos idiomas siguen siendo importantes para la caracterización de ciertos sonidos vocálicos.

Por último, se ha experimentado eliminando directamente las resonancias de la voz en este caso el primer formante.

Esto lo podemos conseguir haciendo que el módulo de dicha resonancia tengo valor 0, en Matlab se realiza de la siguiente manera:

```
modulos(I)=0;
modulos(I+1)=0;
```

Ya que ya hemos visto como encontrar la posición de cada resonancia.

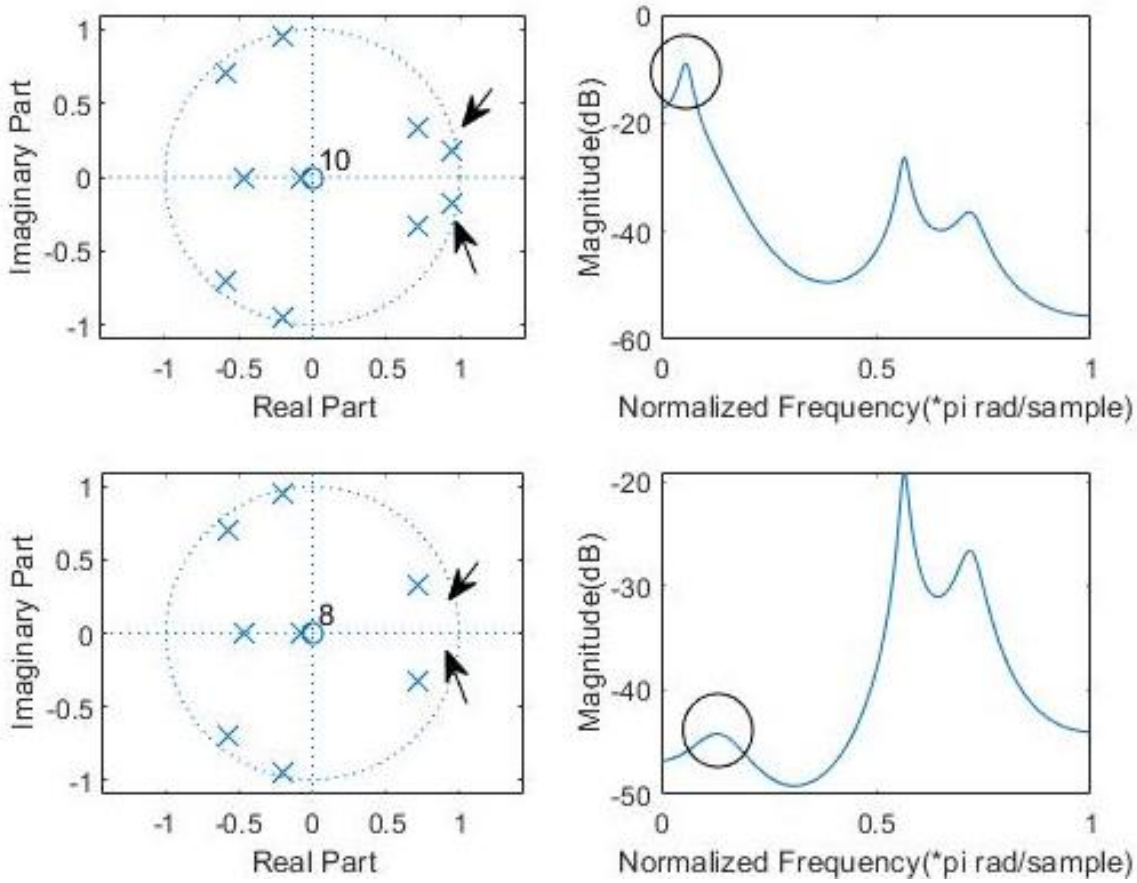

Ilustración 50: Eliminación de la primera resonancia ("lafabSinPolo1.wav")

Notamos como la voz pierde gran cantidad de cuerpo, y llega a tener un sonido casi telefónico, pues al eliminar la primera resonancia correspondiente al primer formante, también eliminamos todas las frecuencias graves que este formante aporta.

Al igual que se puede eliminar el primer formante se puede realizar el mismo proceso con el resto, por ejemplo, con el segundo:

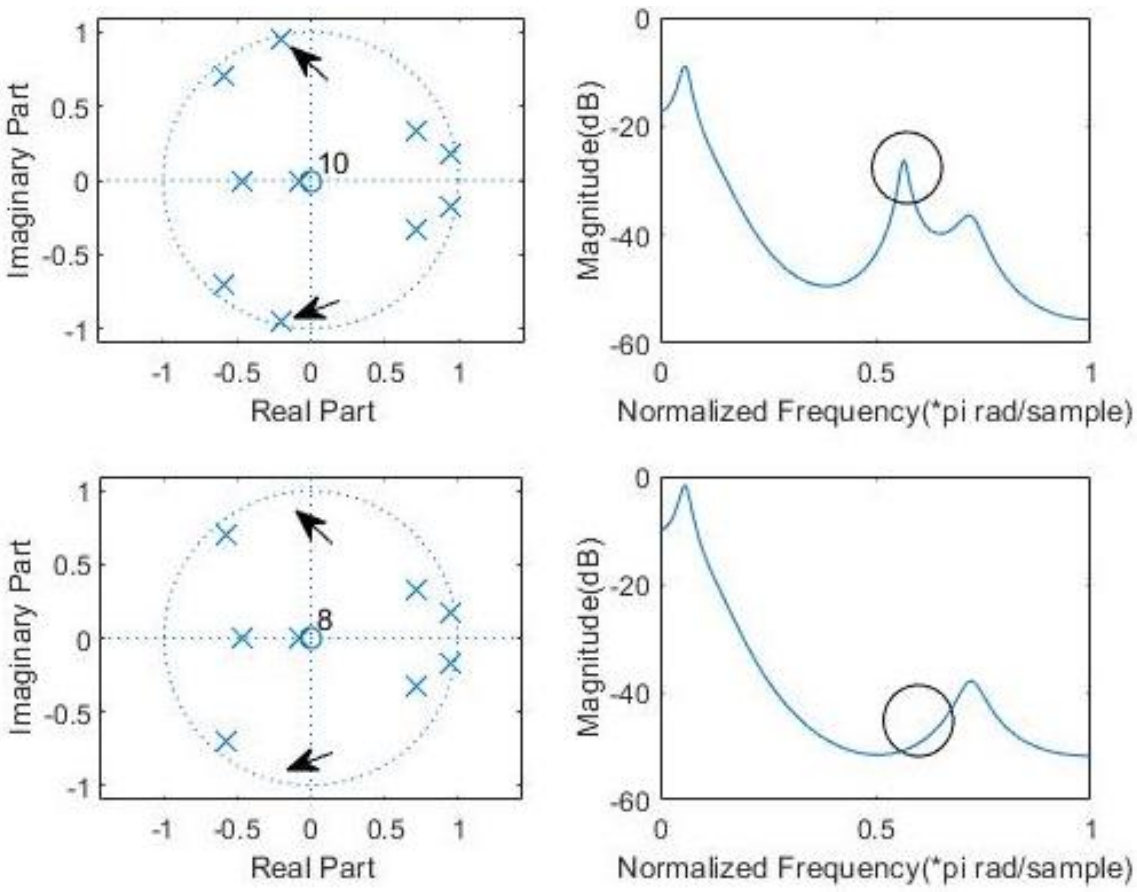

Ilustración 51: Eliminación de la segunda resonancia (“lafabSinPolo2.wav”)

Esto provoca en la voz una pérdida de brillo y inteligibilidad, llegando en ciertas ocasiones a provocar una pequeña distorsión.

Respecto al resto de resonancias no se han apreciado grandes cambios a la hora de aplicar los efectos.

CAPÍTULO 4: CONCLUSIONES

Como conclusión, se puede decir que se ha logrado cumplir los objetivos de este proyecto, se ha visto como desarrollar un Vocoder LPC para codificar y sintetizar voces de forma paramétrica y a su vez como realizar múltiples efectos con dichos parámetros, tales como modificaciones de pitch, afinaciones respecto a escalas musicales o modificaciones en las resonancias del filtro LPC.

Se ha comprendido y estudiado las limitaciones del sistema LPC y se ha analizado la influencia de sus parámetros en la calidad de la voz para conseguir un resultado óptimo a la hora de codificar.

Para todo esto se han aplicado conocimientos de múltiples asignaturas cursadas durante la ingeniería de telecomunicaciones, así como, “Tratamiento Digital del Audio”, “Tratamiento Digital de la Señal” o “Comunicaciones Digitales” y también conocimientos de investigación referenciados en la bibliografía.

Después de este proyecto entendemos mejor como funcionan las codificaciones LPC y como se relacionan directamente con los parámetros de la formación de la voz.

CAPÍTULO 5: BIBLIOGRAFÍA

[1] Sottovoce (2015). “¿Cómo se produce la voz?: la fonación”:

https://sottovoce.hypotheses.org/category/non-classe/voz-hablada/como-se-produce-la-voz-la-fonacion

[2] Educantabria. “Cómo funciona el aparato respiratorio”:

https://www.educantabria.es/docs/Digitales/Primaria/Cono_3_ciclo/CONTENIDOS/CUERPO%20HUMANO/DEFINITIVO%20RESPIRATORIO/Publicar/page5.html

[3] Cristina Herrera y Begoña Morante. “El aparato fonador”:

https://www.virtuniversidad.com/greenstone/collect/ingles/index/assoc/HASHf01b/45450734.dir/doc.pdf

[4] Noelia Avendaño (2012). “Aparato resonador y fonador”:

https://www.cantalirico.com.ar/2012/01/31/aparato-resonador-y-fonador/

[5] Marco Guzmán “Acústica del tracto vocal”:

https://www.logopediapsicologia.com/wp-content/uploads/acustica-del-tracto-vocal.pdf

[6] Marta Ruiz y Helenca Duxans. “Codificación del audio”:

https://www.exabyteinformatica.com/uoc/Audio/Procesamiento_de_audio/Procesamiento_de_audio_(Modulo_4).pdf

[7] Angie Katie. “Codificación de análisis síntesis LPC”:

https://www.academia.edu/28583512/Codificaci%C3%B3n_de_an%C3%A1lisis_s%C3%ADntesis._LPC

[8] Temario de Tratamiento Digital de Audio. “Codificación de Voz”

[9] Thierry Dutoit, Ferrán Marqués (2009). “Applied signal processing: a MATLAB-based proof of concept”. *Chapter 1: How is speech processed in a cell phone conversation?.*

[10] Udo Zölzer (2002). “DAFX digital audio effects”.

[11] Universidad las Palmas de Gran Canaria. “Características estadísticas de la señal de voz”:

https://www2.ulpgc.es/hege/almacen/download/23/23210/teoriavocoderlpc.pdf