

UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS

INGENIEROS DE TELECOMUNICACIÓN



PROYECTO FIN DE CARRERA

SIMULACIÓN DEL SISTEMA DE CODIFICACIÓN de voz en GSM

Departamento de Ingeniería Electrónica
Área de Teoría de la Señal y Comunicaciones

Autor: **Rafael Morillo García**
Director: **Sergio Antonio Cruces Alvarez**

Junio de 2000

E.T.S. INGENIEROS DE TELECOMUNICACIÓN
UNIVERSIDAD DE SEVILLA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
ÁREA DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

Título del Proyecto : Simulación del Sistema de Codificación de voz en GSM
Autor: Rafael Morillo García
Director : Sergio Antonio Cruces Alvarez

Miembros del Tribunal Calificador :

- Presidente :
- Secretario :
- Vocal :

Calificación :

A mi madre Soledad y a mi novia Iki

ÍNDICE

Capítulo 1 .

Introducción

1.1 Objetivos.....1.1
1.2 Organización de la Memoria.....1.1

Capítulo 2 .

Tratamiento de la Señal de Voz

2.1 Estudio de la voz.....2.1
2.1.1 Modelado de la voz.....2.1
2.1.2 Características de la voz.....2.4
2.1.3 Problemas al trabajar con la voz.....2.4
2.1.4 Modelado de la voz II.....2.4
2.1.5 Modelado del oído.....2.6
2.1.6 Características del oído.....2.7
2.1.7 Fenómenos asociados al oído.....2.9
2.2 Análisis de la señal de voz.....2.9
2.2.1 Enventanado.....2.10
2.2.2 Extracción de parámetros.....2.10
2.2.3 Análisis en tiempo.....2.11
2.2.4 Análisis en frecuencia.....2.12

Capítulo 3 .

Descripción Funcional del Codificador RPE-LTP

3.1 Sección de Preprocesado.....3.3
3.1.1 Eliminación de la componente de continua.....3.3
3.1.2 Preénfasis.....3.4
3.2 Sección LPC.....3.4
3.2.1 Segmentación.....3.4
3.2.2 Autocorrelación.....3.4
3.2.3 Recursión de Schur.....3.4
3.2.4 Transformación de los coef. de reflexión a LAR.....3.6
3.2.5 Cuantización y Codificación de los LAR.....3.7
3.3 Sección STP.....3.7
3.3.1 Decodificación de los LAR.....3.7
3.3.2 Interpolación de los LAR.....3.7
3.3.3 Transformación de los LAR en coef. de reflexión.....3.8
3.3.4 Filtro que elimina la envolvente.....3.8
3.4 Sección LTP.....3.9
3.4.1 Subsegmentación.....3.9
3.4.2 Cálculo de los parámetros LTP.....3.9
3.4.3 Codificación de los retrasos LTP.....3.10
3.4.4 Codificación de las ganancias LTP.....3.10
3.4.5 Decodificación de los retrasos LTP.....3.10
3.4.6 Decodificación de las ganancias LTP.....3.10
3.4.7 Filtrado de análisis LTP.....3.11
3.4.8 Filtrado de síntesis LTP.....3.11
3.5 Sección RPE.....3.11
3.5.1 Filtrado de ponderación.....3.11
3.5.2 Diezmado de las muestras 3.12
3.5.3 Cuantización APCM.....3.12
3.5.4 Cuantización inversa APCM.....3.13
3.5.5 Reconstrucción de la secuencia RPE.....3.13

Capítulo 4 .

El Detector de Actividad Vocal

4.1 Introducción.....4.1
4.2 Descripción del Algoritmo.....4.2
4.2.1 Filtrado Adaptativo y Cálculo de la energía.....4.3
4.2.2 Media de los ACF.....4.3
4.2.3 Cálculos de los valores de predicción.....4.3
4.2.4 Comparación espectral.....4.4
4.2.5 Detección de Periodicidad.....4.4
4.2.6 Adaptación de Umbral.....4.5
4.2.7 Decisión del VAD.....4.7
4.2.8 Adición de hangover.....4.7
4.3 Montaje del VAD.....4.8
4.4 Tramas SID.....4.8
4.5 Ruido Conformador.....4.9

Capítulo 5 .

Codificación de Canal

5.1 Introducción.....5.1
5.2 Reordenación.....5.2
5.3 Código Cíclico.....5.3
5.3.1 Fundamento Teórico.....5.4
5.4 Entrelazado.....5.5
5.5 Código Convolucional.....5.6
5.5.1 Fundamento Teórico.....5.7
5.6 Particionamiento.....5.8
5.7 Entrelazado diagonal de bloques.....5.8
5.8 Entrelazado de trama.....5.9

Capítulo 6 .

El Decodificador RPE-LTP

6.1 Sección de Decodificación RPE.....6.1
6.1.1 Cuantización Inversa.....6.2
6.1.2 Reconstrucción de la secuencia RPE.....6.2
6.2 Sección LTP.....6.3
6.2.1 Decodificación de los retrasos LTP.....6.3
6.2.2 Decodificación de las ganancias LTP.....6.3
6.2.3 Filtrado de Síntesis LTP.....6.3
6.3 Sección STP.....6.4
6.3.1 Decodificación de los LAR.....6.4
6.3.2 Interpolación de los LAR.....6.4
6.3.3 Transformación de los LAR en coef. de reflexión.....6.4
6.3.4 Filtro que restaura la envolvente.....6.4
6.3.5 Postprocesado.....6.5

Capítulo 7 .

Resultados

7.1 Introducción.....7.1
7.2 Análisis de una vocal.....7.2
7.2.1 Codificador y Decodificador.....7.2
7.2.2 Detector de Actividad Vocal.....7.11
7.2.3 Codificación de Canal.....7.14
7.3 Análisis de una consonante.....7.18
7.3.1 Codificador y Decodificador.....7.19
7.3.2 Detector de Actividad Vocal.....7.25
7.3.3 Codificación de Canal.....7.27
7.4 Análisis de una palabra.....7.30
7.4.1 Codificador y Decodificador.....7.30
7.4.2 Detector de Actividad Vocal.....7.35
7.4.3 Codificación de Canal.....7.36
7.5 Análisis de una par de palabras simples.....7.38
7.5.1 Codificador y Decodificador.....7.39
7.5.2 Detector de Actividad Vocal.....7.42
7.5.3 Codificación de Canal.....7.43

Capítulo 8 .

Conclusiones

8.1 Introducción.....8.1
8.2 Conclusiones.....8.1

Anexo A

Bibliografía

Recursos en Internet

Capítulo 1 .
INTRODUCCIÓN

1.1 OBJETIVOS

El continuo crecimiento actual de las comunicaciones de voz debido a la aparición de nuevos servicios implica, a su vez, un desarrollo paralelo de las técnicas de procesado digital de señales de voz. Claro ejemplo de ello fue la aparición del sistema GSM que hoy en día lidera el marco de las comunicaciones móviles, que implicó que se desarrollara para ello un estándar de codificación de voz adecuado.

Los recursos tecnológicos de los que se disponían, ancho de banda, retardos, tasa de error del canal, etc. desembocaron en la aparición en 1986, tras la elección entre seis distintas propuestas, del estándar RPE-LTP para codificar voz en GSM.

Este codificador fue el que recibió una nota subjetiva de opinión (MOS) más elevada. Para ello fue sometido a diferentes tests y pruebas en diferentes entornos (ruidosos, silenciosos, con interferencias, etc.), por ello vamos en este proyecto a realizar la implementación de dicho codificador en MATLAB y realizaremos sobre él distintas pruebas con el fin de medir su robustez y eficiencia.

1.2 ORGANIZACIÓN DE LA MEMORIA

En primer lugar haremos una introducción sobre las técnicas de tratamiento digital de señales de voz, en la que veremos las características propias de este tipo de señales, las cuales desembocan en un tratamiento peculiar de ellas. Para ello será preciso conocer cómo se forman estas señales, es decir, cómo el cuerpo humano forma la voz así como también cómo las capta a través del sistema auditivo. Y es así porque de ninguna otra manera sería posible crear un codificador de voz más eficaz sino estudiando la generación y captación de estas señales por parte del cuerpo humano, ya que así conoceremos mejor las propiedades de la voz y podremos eliminar partes que sabremos, a priori, que no son importantes para una óptima recepción por parte del usuario final.

Posteriormente, ya en el tercer capítulo, entraremos ya propiamente en el codificador de voz de GSM, recogido del estándar de ETSI, en el que explicaremos paso a paso, cómo, a partir de una señal digital muestreada a 8 KHz y cuantificada con 13 bits, podemos extraer los parámetros más importantes de la señal de voz, como son la envolvente del espectro, el pitch y la excitación residual, todos ellos ligados íntimamente con la generación de esta señal por parte del cuerpo humano. Veremos cómo se codifican estos parámetros para formar una trama de salida de 260 bits cada 20ms.

En el siguiente capítulo, explicaremos un sistema también muy ligado con el sistema auditivo del cuerpo humano, y que es el detector de actividad vocal, con ello lograremos un ahorro sustancial en la transmisión, descartando el envío de parámetros de la voz cuando ésta no se detecte.

A continuación veremos un capítulo dedicado a la codificación de canal, y que es algo propio de todas las comunicaciones digitales. Debido a las características del canal de transmisión, que introduce errores al propagarse la señal radioeléctrica por él, debemos proteger nuestras señales de estos errores, incluyendo unas redundancias y protecciones, tal como códigos convolucionales, reordenaciones, entrelazados, etc. Veremos la codificación de canal propia de GSM, en la que se da una mejor protección a unos parámetros que a otros, ya que gracias al estudio de la señal de voz, se ha llegado a la conclusión de esta importancia relativa de algunos parámetros.

Ya en el sexto capítulo veremos el decodificador, en el que no entraremos mucho, ya que muchas de sus partes están también incluidas en el codificador. Esto es debido a que cuando codifiquemos los parámetros en GSM, al mismo tiempo los decodificaremos, para ver su efecto en el decodificador.

A continuación, dedicaremos un amplio capítulo a distintas simulaciones con el programa realizado en MATLAB, con distintos tipos de señales de voz, desde ambientes silenciosos hasta ruidosos, y medir así la eficiencia de este codificador, umbrales de ruido, robustez frente a errores, etc.

Para finalizar haremos un capítulo de conclusiones, a las que habremos llegado tras las simulaciones y resultados obtenidos en el capítulo anterior.

Capítulo 2 .

TRATAMIENTO DE LA SEÑAL DE VOZ

2.1 ESTUDIO DE LA VOZ.

Pretendemos, en este apartado[8], estudiar la formación de las señales de voz por parte del cuerpo humano y modelar esta formación por módulos. También veremos las características y propiedades de este tipo de señales, para así poder explotarlas para conseguir una codificación más óptima.

2.1.1 MODELADO DE LA VOZ.

En primer lugar, cuando hablamos, ponemos en funcionamiento la función viento, en la que intervienen los pulmones, generando el viento necesario para provocar las ondas de presión que será finalmente lo que salga al exterior. Esta función viento generaría lo que en el codificador se llamará excitación residual. A continuación intervienen las cuerdas vocales, y lo que hacen es añadir a esa excitación residual el pitch, periodo de pitch o frecuencia de pitch, que será propio de la persona y que la hace indistinguible de otra persona. Ahora ya tenemos un tren de deltas equiespaciados por el periodo de pitch. A continuación lo que haremos es darle la forma a la onda, modularla con el fonema que queramos transmitir. Eso lo hace el tracto vocal, compuesto por los dientes, la lengua y los labios. La señal ya modulada sale al exterior como ondas de presión, y se transmite por el aire a través de variaciones locales de presión. A la vez, como en muchos sistemas, se dispone de una realimentación a través de los oídos, oímos lo que escuchamos, y eso nos hace subir más o menos el volumen de lo que estamos hablando y así adecuarlo al oyente. El diagrama de bloques de la formación de la voz podemos observarlo en la figura 2.1.

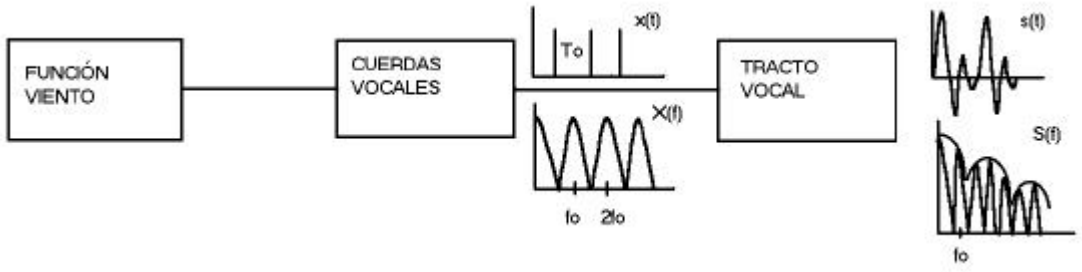


Fig 2.1 Diagrama de bloques de la formación de la voz.

Fonemas

La representación sonora de los fonemas se llama fono. Los fonemas lo estudia la fonología mientras que los fonos lo estudia la fonética. Entre los fonemas y los fonos no hay una relación 1:1, pues un fonema puede tener más de un fono. Veamos una clasificación de los fonos.

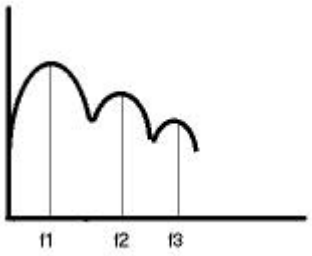
- Vocales. Tienen mucha energía. No hay nada que las obstruyen en el tracto vocal. Tiene el espectro muy claro. La amplitud varía mucho de unas vocales a otras. Ej. /a/
- Líquidas. La lengua interviene en la formación del sonido. Ej. /l/
- Consonantes. Hay algo que obstruye su salida. Hay tres tipos a su vez:
  - Nasales. La boca se cierra y el aire sale por la nariz. Ej. /n/
  - Oclusivas. Tiene un periodo de silencio tras el cual sube mucho la amplitud. Ej. /p/, /b/
  - Fricativas. Son muy ruidosas. Ej. /s/, /f/

Dentro de los sonidos hay también otra clasificación.

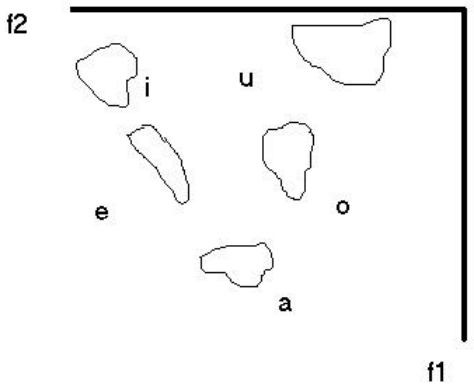
- Sonoros. Vibran las cuerdas vocales, tienen el pitch y los formantes claros. Tiene la energía del espectro desplazada a bajas frecuencias. Ej /b/, /d/ y todas las vocales.
- Sordos. No hacen vibrar las cuerdas vocales, el espectro es más ruidoso y tiene menos energía. Ej. /p/, /t/

Carta de formantes (Vocales)

Las vocales son la parte de la voz que tiene más energía, y por tanto la más importante. Tienen este espectro:



f1 es el pitch, mientras que f2 y f3 son los formantes. Vamos a ver sólo el primer y segundo formantes para distintas vocales.



Podemos observar que la /i/ es la más aguda, después la /e/, a continuación la /a/, después la /o/ y por último la /u/. Se suele poner como un triángulo vocálico. El oído va a detectar en la posición en la que está. Es muy importante en la comunicación la resonancia, que es lo que detecta el oído. En entornos ruidosos es difícil entenderse en inglés, porque los formantes se solapan en la carta y es más compleja.

Rasgos Prosódicos.

El pitch, la entonación y la intensidad varían de una persona a otra y también varían con los rasgos emocionales y con los estados físicos. En los hombres el periodo de pitch es más alto, por lo que la frecuencia de pitch es más baja y es una voz más grave. En las mujeres sucede al contrario, es una voz más aguda. En el caso de identificación de personas no se escoge la voz, puesto que es una cosa que se puede falsear.

2.1.2 CARACTERÍSTICAS DE LA VOZ.

La voz es una señal no estacionaria. No podemos esperar a que se acabe. Debemos procesarla poco a poco. Debemos hacer un análisis localizado de la señal de voz, tal que los trozos sean localmente estacionarios, con una duración de 20 a 25 ms. Si muestreamos a 8 KHz, corresponde a un número de muestras entre 160 y 200.

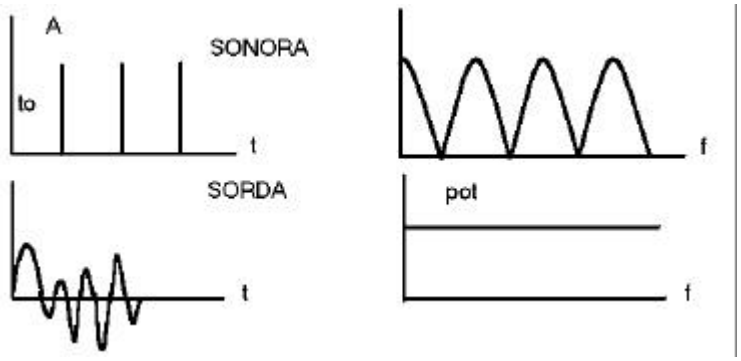
Hay tres tipos de tramas: Sordas, sonoras y de transición. Dentro de una palabra, un fono está influenciado por los fonos adyacentes. A ese mecanismo de cambiar un fono en torno a sus adyacentes se denomina coarticulación. Son tramas difíciles de localizar en las tramas de transición.

2.1.3 PROBLEMAS AL TRABAJAR CON LA VOZ.

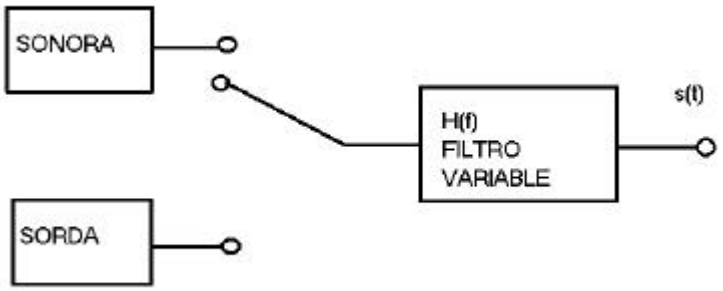
La voz tiene un gran rango dinámico, una gran diversidad de amplitudes y ruido. Otro problema añadido es la longitud de la trama, pues no todos los fonos tienen la misma longitud.

2.1.4 MODELADO DE LA VOZ II.

Si la excitación es sonora, la voz se puede modelar como un tren de deltas separados por el pitch. Necesitaremos *to* y *A*. Si la excitación es sorda, no tiene periodo ni amplitud definidos. Es un ruido con una cierta potencia.



Ponemos un conmutador a un filtro variable, que modela al tracto vocal.



Las resonancias en el espectro son los formantes que identifican esa voz. El filtro para sordas es más aplanado e incluso crece a alta frecuencia. Lo importante en la voz serán las resonancias, los picos, que es lo que el oído capta, esa potencia en los picos. Los pares *(Ai , fi)* sirven para detectar lo que estamos diciendo. Las cuatro primeras resonancias son las más importantes, las que nos identifican la vocal o consonante que estamos diciendo. Esto nos puede servir para codificar. A continuación, veamos dos ejemplos de codificadores:

Codificador de voz (Vocoder).

Es un codificador paramétrico, extrae parámetros de la señal y los envía. Hacemos un modelo ARMA para el filtro variable H(f).

$$H(z) = \frac{1 - \sum bk\, z^{-k}}{1 - \sum ak\, z^{-k}}$$

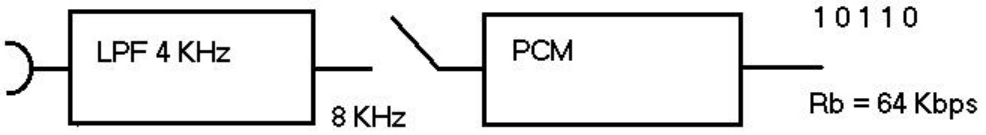
Los polos modelan bien la resonancia, y los ceros modelan bien los valles. La voz es poco sensible a los valles, por lo que nos quedamos sólo con la parte todo polo, y nos queda un modelo AR.

$$H(z) = \frac{G}{1 - \sum ak\, z^{-k}}$$

Hay un modelo de onda progresiva que modela el tracto vocal como una serie de tubos de diferentes anchuras, estudiándose frecuencias de resonancias, reflexiones, etc. Cada tubo tiene asociado un par de polos complejos conjugados. Con ocho tubos se modela muy bien la voz, pero con cuatro tubos se modela ya bien. Pero no con menos de cuatro. Ocho tubos corresponden a dieciséis polos y cuatro tubos a ocho polos o coeficientes. El modelo de onda progresiva se parece mucho al modelo de los coeficientes de reflexión. El filtro AR le mete las redundancias a la voz. Con predicción lineal se puede construir el filtro AR. El filtro va a capturar la forma de onda del espectro. Los coeficientes del filtro parametrizan la envolvente del espectro, por lo que puedo localizar las resonancias y sus amplitudes *(Ai ,Fi).* El vocoder transmite los ocho coeficientes *(Ai,Fi) i= 1 ... 4,* o directamente los ak *k=1 ... 8,* o también los coeficientes de reflexión asociados a los ak. Estos coeficientes tienen unos más importancia que otros, por lo que al cuantizarlos, se usan más bits para unos que para otros, y se le añade una protección contra el error. Cada 20 ó 25 ms, se manda el modelo del tracto vocal de cada trozo de voz. Cuando detectemos, va a haber un bit que nos indica si es sorda o sonora, si es sonora enviamos *A* y *to,* y si es sorda, enviamos la potencia del ruido. El vocoder es el codificador que más comprime, pero el que peor suena. La tasa del vocoder es de 2.4 Kbps. Cuando llega al otro extremo, el receptor reconstruye el canal y según el bit que nos indica si es sorda o sonora, le metemos el tren de deltas o el ruido al filtro reconstruido y generamos la salida, que en tiempo es totalmente distinta a la original, pero que se escucha y se entiende.

Codificador PCM.

Es un codificador de forma de onda, al contrario que el anterior, que era un codificador paramétrico. Ha de tener una tasa elevada de transmisión para que tenga calidad. Tiene una tasa de 64 Kbps, mucho mayor que los 2.4 Kbps que la del vocodec.



2.1.5 MODELADO DEL OÍDO.

El oído se comporta de una forma no lineal. Tiene tres partes: Oído Externo, Oído Medio y Oído Interno.

Oído Externo.

Tiene dos partes: oreja, y conducto auditivo. Tenemos dos oídos, que son dos sensores acústicos. Forman un array. El cerebro es capaz de detectar la dirección de la que nos viene el sonido. Escuchamos de una forma directiva. Si el sonido tiene una frecuencia alta (tiene mayor atenuación), detectamos la dirección del sonido por intensidad, porque en un oído está más atenuado que en otro. Si la frecuencia es baja, detectamos la dirección gracias al desfasaje entre las dos ondas de presión.

Escuchamos frecuencias comprendidas entre 20 Hz y 20 KHz. Sin embargo, la voz tiene un rango de 20 Hz a 7 KHz. Una vez pasada por un canal telefónico tiene de 20 Hz a 4 KHz y se entiende muy bien. Las orejas son las antenas. El conducto auditivo es un resonador en $\lambda/4$. Es un filtro que amplifica 15 dB entre 2 KHz y 5 KHz. La velocidad del sonido es de 360 m/s. De estos datos podemos obtener la longitud del conducto auditivo, tomando una frecuencia media de 3KHz y sabiendo que:

$$v = \lambda/T = \lambda * f = 360\ m/s = \lambda * 3000 \Rightarrow \lambda/4 = 3\ cm$$

En realidad tiene un poco menos, unos 2 cm. En la figura 2.2 podemos observar un diagrama de bloques del oído.

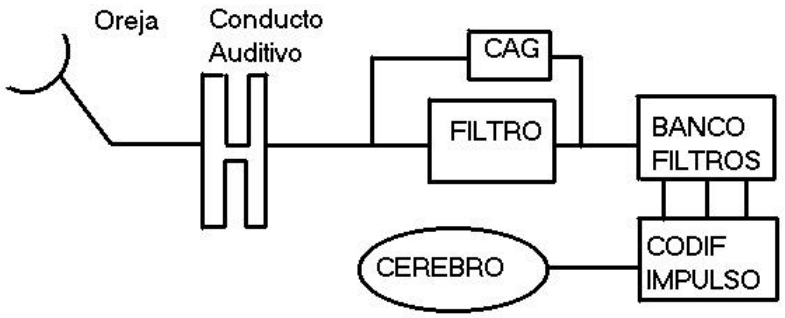


Fig 2.2 Diagrama de bloques del oído.

Oído Medio.

El tímpano es un transductor, al llegarle la onda de presión, pasa de una vibración acústica a una vibración mecánica. Hace de adaptador de impedancias de un medio a otro, y transmite esa vibración a una cadena de huesecillos (yunke, martillo, tambor). La cadena de huesecillos actúa como un filtro y a la vez como un Controlador Automático de Ganancia (CAG). Intenta no pasar mucha energía hacia adentro, si es muy alta. Evita que nos quedemos sordos si estalla una bomba. La inercia mecánica de la cadena de huesecillos es pequeña y ante impulsos de energía cortos, no los transmite bien y previene una variación muy brusca, no la deja pasar.

Oído Interno.

El último de los huesecillos, el martillo, golpea el caracol. Es un conducto óseo en forma de caracol. Si lo alargamos, veríamos que hay terminaciones nerviosas, cada una de ellas se excita con una cierta frecuencia. Se puede modelar como un banco de filtros de factor de calidad constante. Hay 24 bandas críticas. Las propiedades con que se escucha depende de la banda en la que estemos. El oído es capaz de discriminar entre bandas a baja frecuencia, posee una mayor resolución a bajas frecuencias. Todas las bandas tienen la misma anchura logarítmicamente. Eso explica que las notas en el pentagrama no estén equiespaciadas. Lo último es codificar esos impulsos para mandarlos al cerebro, aunque eso no se sabe muy bien cómo lo hace.

2.1.6 CARACTERÍSTICAS DEL OÍDO.

Umbral de audición.

La curva de 0 fonios define el umbral de audición. Si transmitimos un tono cuya amplitud no llega al umbral de audición, el oído no lo escucha. El oído tiene un umbral de audición distinto para cada frecuencia. Por ello, algunos despertadores están cerca de los 4 KHz, donde es más sensible. En torno a los 120 fonios está el umbral del dolor. La amplitud de las curvas cambia dependiendo de la duración del fono, pues el oído integra, promedia.

Discriminación temporal.

Mide cómo de cerca han de estar dos tonos de la misma frecuencia separados en tiempo para poderlos diferenciar. Está en torno a 3 ms. Que a 8 KHz serían 24 muestras.

Discriminación en frecuencia.

Mide cómo han de estar separados dos tonos en frecuencia que se reciben en el mismo instante de tiempo para poderlos diferenciar. Por debajo de 1KHz, esa separación es de 1 a 3 Hz. Cuando la frecuencia sube, también sube esa separación, a 8 KHz ha de ser de 100 Hz.

Distinguimos en total 1600 frecuencias y 350 intensidades, que son aproximadamente 300.000 tonos distintos. Esa capacidad de distinguir es para tonos con una duración mayor a 200 ms. Si el tono dura menos de 200 ms, ya sólo podemos distinguir 150.000 tonos.

Enmascaramiento en tiempo.

El hecho de que nos llegue un tono hace que el umbral de audición suba. Escuchamos un tono si está por encima del umbral de audición, pero si nos llega un tono y hace que ese umbral suba, puede que el siguiente tono que nos llegue esté por debajo y no lo escuchemos. Eso es interesante para codificaciones perceptuales, no codifican tonos que no se vayan a escuchar.

Enmascaramiento en frecuencia.

Si transmitimos dos tonos de distinta frecuencia al mismo tiempo, cada uno de los tonos sube el umbral de audición y puede pasar que uno suba tanto el umbral que haga que el otro no se escuche. Entonces no tiene sentido que se transmita, y eso lo aprovechan los codificadores de banda. Se codifica la primera banda, y cuando el error de cuantización caiga por debajo del umbral, los bits restantes se utilizan para la siguiente banda, y así sucesivamente. Si al final de todas las bandas sobran bits, se reparten entre todas las bandas.

Enmascaramiento tono-ruido.

El tono ha de estar por encima del ruido 25 dB para que se escuche. El ruido ha de estar 6 dB por encima del tono para que no escuche. Eso es porque el tono tiene una pequeña banda, y el oído integra para promediar la energía. El ruido tiene más ancho de banda, y con poca amplitud ya tiene más energía que el tono. El tono ha de subir mucho su amplitud para poderse escuchar.

2.1.7 FENÓMENOS ASOCIADOS AL OÍDO.

Inhibición.

Cuando estamos escuchando mucho tiempo lo mismo, el cerebro corta y dejamos de escuchar eso.

Saciedad Verbal.

Cuando escuchamos muchas veces la misma palabra, la escuchamos extraña.

Sumador Verbal.

Si escuchamos una palabra incorrecta, la entendemos gracias a que las vocales tienen más energía.

Comportamiento biaural.

Ponemos un tono a 400 Hz en el oído derecho y otro a 800 Hz en el otro oído. En un instante de tiempo se conmutan las frecuencias y el oído no percibe bien ese cambio.

Ocultación del oído.

Si nos ponemos unos auriculares cambia la función de transferencia del oído, y parece que el sonido nos surge de dentro.

Efecto Fiesta.

Si estamos en una fiesta hablando con alguien, hay más fuentes emitiendo, que son independientes e incorreladas. Pasan al canal aéreo y al oído con una cierta función de transferencia. Oímos sólo lo que nos interesa y el resto lo anulamos.

2.2 ANÁLISIS DE LA SEÑAL DE VOZ.

Hay dos formas de operar: en tiempo y en frecuencia. En tiempo son más simples pero en frecuencia son mejores. Hemos visto que los parámetros que caracterizan la voz son: energía, pitch y forma del tracto vocal. Como la voz no es estacionaria, para analizarla hemos de enventanarla, habrá un compromiso en la longitud de la ventana. Por un lado, ha de ser lo suficientemente corta como para que el parámetro que queramos medir sea estacionario en esa ventana, y también ha de ser lo sufucientemente larga para que contenga el suficiente número de muestras, y reduzca la varianza del error al estimar un parámetro. Con una ventana larga, se optimiza la tasa de transmisión, se envían menos parámetros. El tamaño de la ventana también nos determina la resolución que tenemos, si la ventana es corta tenemos gran resolución en tiempo pero menor en frecuencia. Si la ventana es larga justo al contrario.

2.2.1 ENVENTANADO.

Tenemos una señal *s[n]* muy larga y la multiplicamos por una ventana con una cierta forma. Eso en frecuencia sería la convolución de las dos señales, y modificamos el espectro en frecuencia. Vamos a ver dos tipos de ventanas.

Ventana Rectangular.

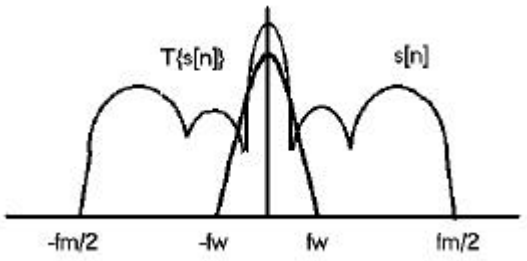
Es la ventana que da el lóbulo principal más estrecho, da mayor resolución en frecuencia, pero los lóbulos secundarios tiene más nivel. La caída del lóbulo principal al secundario es de 13 dB, por lo que componentes de distintas frecuencias afectan a otras frecuencias, el espectro se difumina y cambia la forma de la señal. Otra ventaja es que un cambio brusco en la señal lo detecta la ventana rectangular.

Ventana Hamming.

Es la más usada. Normalmente se suele usar con el doble de duración de la ventana rectangular, porque tiene partes muy atenuadas. El lóbulo principal es mayor pero los secundarios están más atenuados, por tanto, distorsiona un poco la anchura de los pulsos pero no la envolvente del espectro, y lo que importa es la envolvente del espectro. Con la ventana Hamming se hace necesario el solapamiento porque atenúa mucho las partes extremas. Se usa un solapamiento del 50% para suavizar y detectar mejor los cambios en la señal.

2.2.2 EXTRACCIÓN DE PARÁMETROS.

Para extraer un parámetro a partir de una señal *s[n]* se le aplica una transformada, por ejemplo, si el parámetro es la energía se eleva la señal al cuadrado. Después esa transformada se le aplica un enventanado. Si queremos extraer el parámetro de varias ventanas, se aplica el proceso anterior con sucesivas ventanas. La ventana es una especie de filtro LP. *Q[n]* es el parámetro *Q* de la ventana colocada en la n-ésima posición. *Q[n] = T{ s[n] } * w[n]*.



Vemos que el parámetro varía más lentamente que la señal, es como un promediado, tiene menos ancho de banda. El parámetro es redundante, está sobremuestreado. Lo que se hace es diezmarlo para coger sólo las muestras precisas. A partir del parámetro *Q[n]* generamos el parámetro diezmado *Q[nD]*.

Vamos a calcular el factor de diezmado *D*. Si usamos una ventana de 20 ms, la frecuencia de corte del parámetro, *fw = 50 Hz*. Si muestreamos la voz a *fm = 8 KHz*, entonces *D* es:

$$D = fm/2\ fw = 80$$

Nos basta con quedarnos con 1 de cada 80 muestras porque el parámetro varía muy lentamente. En la práctica, lo que se hace es directamente diezmarlo, saltando la ventana por el factor de diezmado.

2.2.3 ANÁLISIS EN TIEMPO.

Energía

Para obtener un parámetro de la voz aplicábamos una transformación y pasábamos por una ventana que promediaba. Podríamos diezmar ya que lo anterior era como filtrar LP. En el caso de la energía era la señal al cuadrado. Para el cálculo de la energía interesa ventanas rectangulares para no falsear el resultado. La energía nos puede servir para detectar cuando empieza y termina una palabra. La /s/ puede confundirse con ruido. Por ello emplearemos un preprocesado y necesitaremos una técnica adicional, el Vocal Activity Detector (VAD), que detecta si hay voz o no. Usa la energía y otros parámetros.

Tasa de cruces por cero.

Un parámetro que usa el VAD es la tasa de cruces por cero.

$$T\{s[n]\} = ZCR = ½ \mid signo\{s[n]\} - signo\{s[n-1]\}\mid$$

Si lo promediamos con una ventana nos da el número de cruces por cero debajo de la ventana. Teniendo el número de cruces por ventana y sabiendo la duración de la ventana, tenemos el número de cruces por segundo. Cuando hay mucho ruido, la tasa de cruces por cero es elevada. La ZCR se acerca mucho a la frecuencia del primer formante, sobre todo en señales de voz sonoras. Para señales sordas no está tan claro esa relación.

Autocorrelación.

Elimina la fase y deja el módulo. La fase no es importante. Las propiedades de la autocorrelación es que es máxima en cero, es par y sobre todo, que preserva la periodicidad. Localizando dos máximos locales consecutivos en tiempo, su diferencia será el periodo de pitch. Cuando enventanamos, esas propiedades dejan de ser ciertas. Hay que tener cuidado por ello a la hora de estimar el periodo de pitch. Para señales sordas es más difícil calcular el pitch con la autocorrelación. Las ventanas usadas suelen ser largas, de al menos 300 muestras para que como mínimo haya dos periodos de pitch. La autocorrelación sirve también para calcular la redundancia del espectro.

2.2.4 ANÁLISIS EN FRECUENCIA.

Suelen ser mejores que los análisis en tiempo, pues el oído es más sensible a frecuencia. Pero son métodos más costosos. Se suele trabajar con bancos de filtros, cada uno en una banda de frecuencias. Se suele recurrir a espectrogramas, que son análisis de Fourier localizados. Hay dos tipos de espectrogramas

Espectrograma de Banda Estrecha.

Si el tamaño de la ventana es muy elevado, tiene mucha resolución, puedo detectar cambios de muy pequeña frecuencia.

Espectrograma de Banda Ancha.

Si localizamos mucho en tiempo, perdemos resolución en frecuencia. Veamos cómo podemos calcular el pitch. Contamos el número de lóbulos que hay hasta 2 KHz y dividimos 2 KHz entre el número de lóbulos que nos haya dado. Lo hacemos así y no cogemos el primer pico porque nos puede dar un error mayor.

Veamos otra técnica que se aplica normalmente antes de comprimir la voz.

Preénfasis.

Antes de comprimir la voz, se enfatizan las altas frecuencias con el filtro:

$$y[n] = s[n] - A\, s[n-1]$$

A típicos suelen ser del orden de 0.4.Y después de descomprimirla se aplica un deénfasis.

Deénfasis.

$$y[n] = s[n] + B\, s[n-1]$$

Valores de B típicos suelen estar comprendidos entre 0.1 y 1.

Capítulo 3 .

DESCRIPCIÓN FUNCIONAL DEL CODIFICADOR RPE-LTP.

La entrada al codificador RPE-LTP[1] es una señal muestreada a 8 KHz y cuantizada con 13 bits. Por tanto, la tasa de entrada al codificador son 104 kbits/s. El codificador RPE-LTP es un codificador paramétrico, es decir, codificamos parámetros de una señal en contraposición a los codificadores no paramétricos, que codifican cada una de las muestras. Aquí vamos a codificar la envolvente del espectro de la señal, su pitch y la excitación residual que queda tras eliminar la envolvente del espectro y el pitch, todos ellos calculados en un periodo de tiempo de 20 ms donde se considera que una señal de voz es estacionaria. Cada término lo codificamos con un cierto número de bits, distintos de unos términos a otros. En total, para una trama de 20 ms tenemos los siguientes términos con sus correspondientes bits:

Parámetros codificados	*Número de bits*
8 LARc	*36*
4 bc	*4x2=8*
4 Nc	*4x7=28*
4 Mc	*4x2=8*
4 xmaxc	*4x6=24*
4x13=52 xmc	*52x3=156*
Número total de bits	*260*
Velocidad binaria	*13 kbits/s*

Tabla 3.1 Parámetros de salida del codificador RPE-LTP.

El diagrama de bloques del codificador RPE-LTP se ilustra en la figura 3.1. Los bloques individuales se describen en las siguientes secciones.

La sección de preprocesado del codificador RPE-LTP comprende a su vez los dos siguientes subbloques:

- Eliminación de la componente de continua. (3.1.1)
- Preénfasis. (3.1.2)

La sección LPC del codificador RPE-LTP comprende a su vez los cinco siguientes subbloques:

- Segmentación. (3.2.1)
- Autocorrelación. (3.2.2)
- Recursión de Schur. (3.2.3)
- Transformación de los coeficientes de reflexión a LAR. (3.2.4)
- Cuantización y Codificación de los LAR. (3.2.5)

La sección STP comprende a su vez los cuatro siguientes subbloques:

- Decodificación de los LAR. (3.3.1)
- Interpolación de los LAR. (3.3.2)
- Transformación de los LAR en coeficientes de reflexión. (3.3.3)
- Filtrado STP. (3.3.4)

La sección LTP comprende los siguientes cuatro subbloques:

- Subsegmentación. (3.4.1)
- Cálculo de los parámetros LTP. (3.4.2)
- Codificación de los retrasos LTP (3.4.3) y las ganancias LTP (3.4.4)
- Decodificación de los retrasos LTP (3.4.5) y las ganancias LTP (3.4.6)
- Filtrado de análisis LTP (3.4.7) y filtrado de síntesis LTP (3.4.8)

La sección RPE comprende cinco diferentes subbloques:

- Filtrado de Weight. (3.5.1)
- Diezmado de las muestras. (3.5.2)
- Cuantización APCM. (3.5.3)
- Cuantización inversa APCM. (3.5.4)
- Reconstrucción de la secuencia RPE. (3.5.5)

En cada sección tenemos una parte para codificar y enviar parámetros y otra parte para decodificar lo que ya hemos codificado, con el fin de conocer lo que le llegará al receptor y poder así actuar eliminándolo de la señal de entrada.

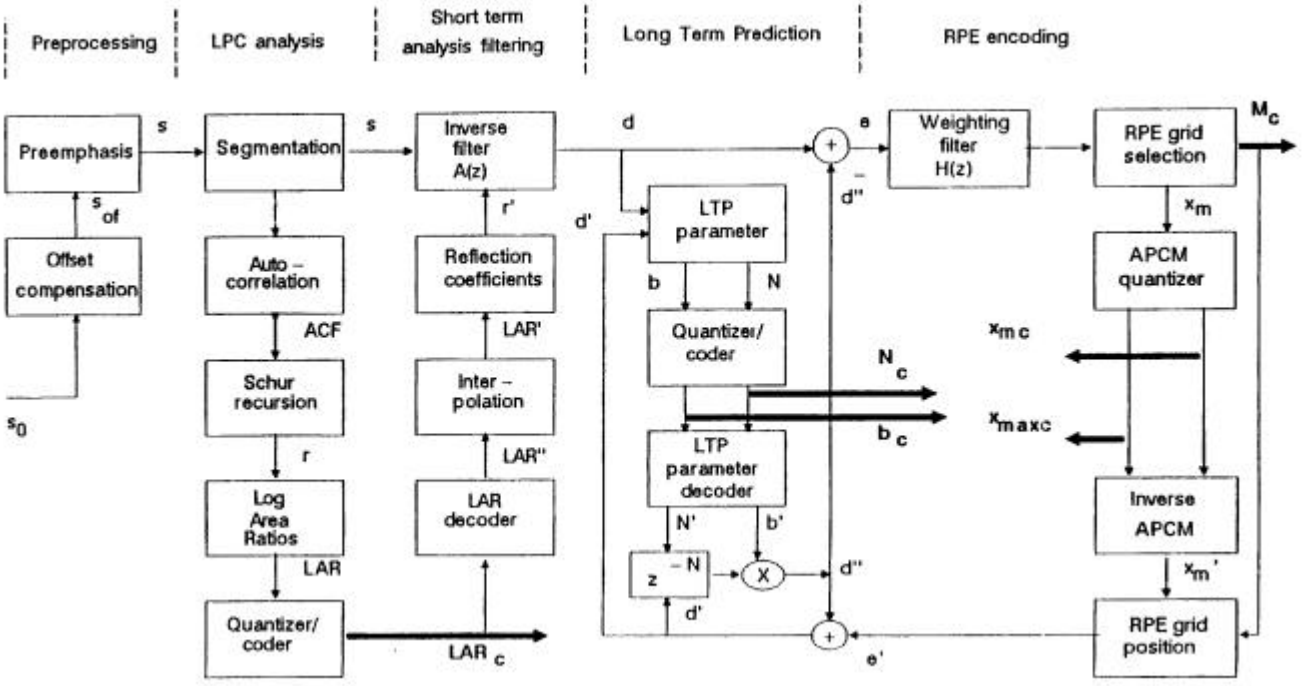


Fig 3.1 Diagrama de bloques del codificador RPE-LTP.

3.1 SECCIÓN DE PREPROCESADO.

3.1.1 ELIMINACIÓN DE LA COMPONENTE DE CONTINUA.

Previamente a la codificación, se aplica una eliminación de componente de continua mediante un filtro notch para eliminar el posible offset de la señal de entrada y producir una señal sin offset a su salida.

$$sof(k) = so(k) - so(k-1) + alpha * sof(k-1) \qquad alpha = 32735*2^{-15}$$

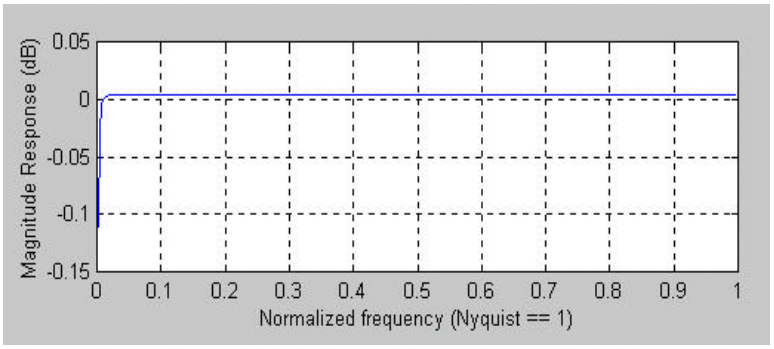


Fig 3.2 Filtro de eliminación de la componente de continua.

3.1.2 PREÉNFASIS.

La señal sin offset es aplicada como entrada a un filtro FIR de primer orden produciendo la señal de entrada a la sección LPC.

$s(k) = sof(k) - beta * sof(k-1)$ $\qquad beta = 28180*2^{-15}$

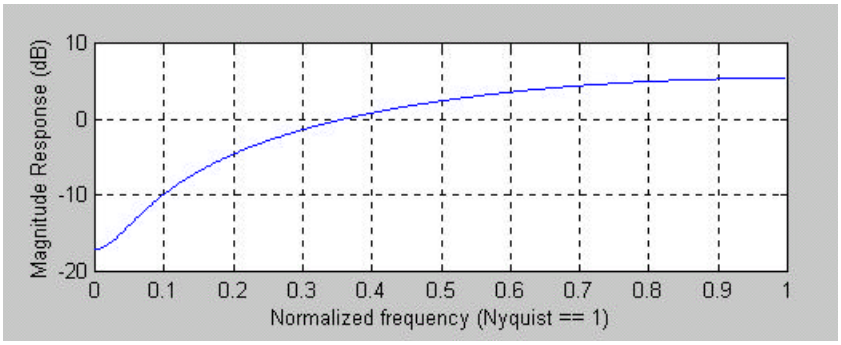


Fig 3.3 Filtro de preénfasis.

3.2 SECCIÓN LPC.

3.2.1 SEGMENTACIÓN.

La señal *s(k)* es dividida en tramas sin overlapping con un periodo de duración de To = 20 ms (160 muestras). Un análisis LPC de orden p=8 es aplicado a cada trama.

3.2.2 AUTOCORRELACIÓN.

Los primeros p+1=9 valores de la función de autocorrelación son calculados mediante la siguiente fórmula:

$$ACF(k) = \sum_{i=k}^{159} s(i)\, s(i-k),\ k = 0,1,\ldots 8$$

3.2.3 RECURSIÓN DE SCHUR.

Los coeficientes de reflexión *r(i)* son calculados a partir de los *ACF* tal como muestra la figura 3.3 usando el algoritmo de la recursión de Schur. La recursión de Schur es esencialmente la misma que la recursión de Levirson-Durbin. La recursión de Levirson Durbin parte de un conjunto de datos de una señal, calcula los coeficientes de autocorrelación de esa señal, y a partir de ellos, calcula los coeficientes de reflexión y posteriormente los coeficientes del filtro del que se ha formado la señal. La recursión de Schur aprovecha este paso intermedio y se centra en la transformación de los coeficientes de autocorrelación en coeficientes de reflexión, por lo que dichas

ecuaciones son las mismas, pero modificadas. Las ecuaciones que implementan la recursión de Schur las tenemos en forma de diagrama de flujo en la figura 3.4.

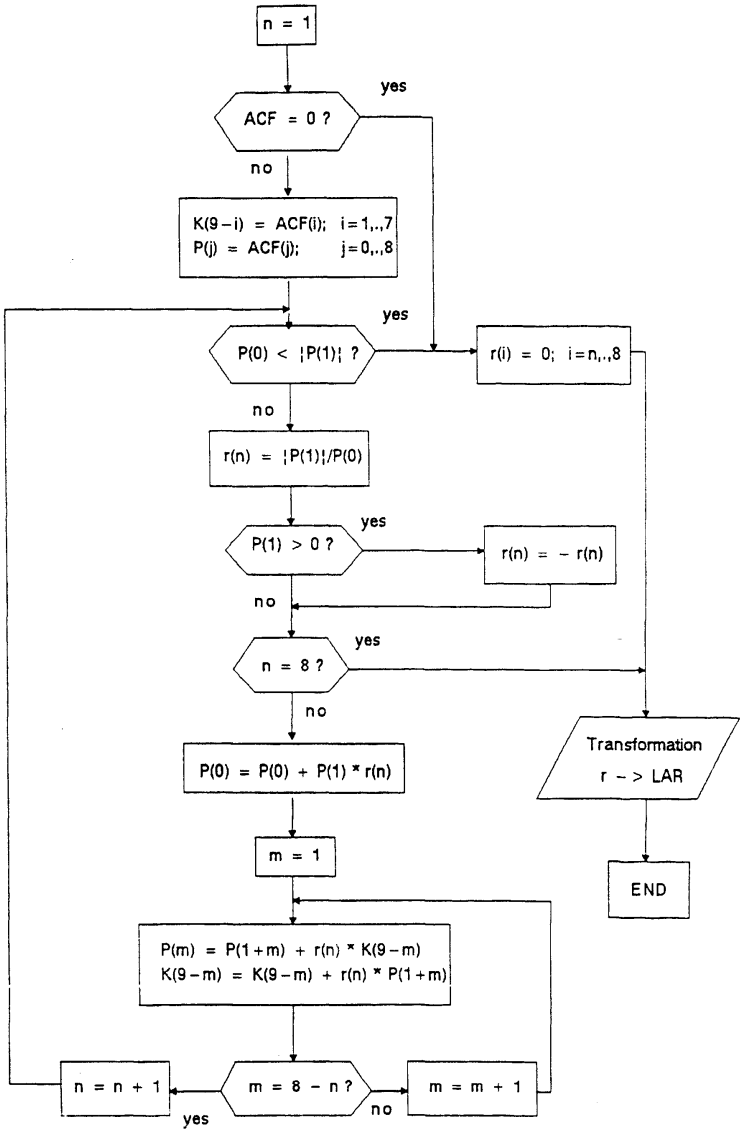


Figura 3.4 Recursión de Schur.

3.2.4 TRANSFORMACIÓN DE LOS COEFICIENTES DE REFLEXIÓN A LAR.

Los coeficientes de reflexión *r(i)* , *(i=1 ... 8)* calculados con la recursión de Schur están en el rango:

$$-1 <= r(i) <= +1$$

Debido a las características favorables de cuantización que así lo permiten, los coeficientes de reflexión son convertidos a *LAR* según la siguiente fórmula:

$$LAR(i) = \log_{10} \frac{1 + r(i)}{1 - r(i)}$$

Debido al carácter discreto de una cuantización, esta curva característica se aproxima por tramos y queda la siguiente fórmula:

$$LAR(i) = \begin{cases} r(i) & |r(i)| < 0.675 \\ sign(r(i)) * (2|r(i)| - 0.675) & 0.675 \le |r(i)| < 0.950 \\ sign(r(i)) * (8|r(i)| - 6.375) & 0.950 \le |r(i)| \le 1.000 \end{cases}$$

Veamos en la siguiente gráfica la curva de cuantización de los LAR en función de los coeficientes de reflexión.

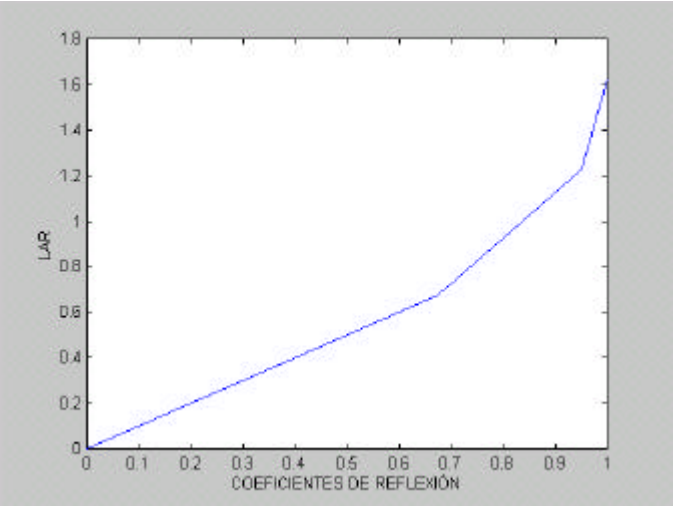


Fig 3.5 Curva de cuantización de los LAR

3.2.5 CUANTIZACIÓN Y CODIFICACIÓN DE LOS LAR.

Los *LAR* tienen diferentes rangos dinámicos y diferentes distribuciones de densidad. Por esta razón, cada *LAR* tiene una codificación distinta, de acuerdo con la siguiente ecuación:

$$LARc(i) = round\,[\,A(i) * LAR(i) + B(i)\,] \qquad i = 1 \ldots 8$$

En la tabla 3.2 observamos los valores de *A (i)* y *B (i)* así como los valores extremos de cada *LARc (i).* Cuando un *LARc (i)* sobrepasa su rango se sustituye por su valor máximo o mínimo según corresponda. Los *LARc (i)* es lo que finalmente se envía, usando una codificación en CA2, y con distinto número de bits, 6, 6, 5, 5, 4, 4, 3 y 3 respectivamente para cada *LAR(i) i=1 ... 8.*

LAR Nº i	*A (i)*	*B (i)*	*Valor mínimo*	*Valor máximo*
1	*20.000*	*0.000*	*-32*	*+31*
2	*20.000*	*0.000*	*-32*	*+31*
3	*20.000*	*4.000*	*-16*	*+15*
4	*20.000*	*-5.000*	*-16*	*+15*
5	*13.637*	*0.184*	*-8*	*+7*
6	*15.000*	*-3.500*	*-8*	*+7*
7	*8.334*	*-0.666*	*-4*	*+3*
8	*8.824*	*2.235*	*-4*	*+3*

Tabla 3.2 Codificación de los LAR.

3.3 SECCIÓN STP.

3.3.1 DECODIFICACIÓN DE LOS LAR.

En este bloque, los *LARc* son decodificados según la siguiente ecuación:

$$LAR''(i) = [\,LARc(i) - B(i) / A(i)\,]$$

3.3.2 INTERPOLACIÓN DE LOS LAR.

Para evitar transitorios desagradables que pueden ocurrir cuando los coeficientes del filtro son modificados abruptamente, se interpola los *LAR* decodificados de la trama actual con los de la trama anterior. El filtro STP trabaja con cuatro conjuntos de *LAR* como podemos ver en la tabla 3.3. Para la primera trama los *LAR''* son cero.

K	*LAR'J (i)*
0 ... 12	*0.75 * LAR''J-1 (i) + 0.25 * LAR''J (i)*
13 ... 26	*0.50 * LAR''J-1 (i) + 0.50 * LAR''J (i)*
27 ... 39	*0.25 * LAR''J-1 (i) + 0.75 * LAR''J (i)*
40 ... 159	*LAR''J (i)*

Tabla 3.3 Interpolación de los LAR (J = segmento actual)

3.3.3 TRANSFORMACIÓN DE LOS LAR EN COEFICIENTES DE REFLEXIÓN.

Los coeficientes de reflexión son finalmente determinados usando la siguiente ecuación.

$$r'(i) = \begin{cases} LAR'(i) & |LAR'(i)| < 0.675 \\ sign(LAR'(i)) * (0.500 * |LAR'(i)| + 0.337500) & 0.675 \le |LAR'(i)| < 1.225 \\ sign(LAR'(i)) * (0.125 * |LAR'(i)| + 0.796875) & 1.225 \le |LAR'(i)| \le 1.625 \end{cases}$$

3.3.4 FILTRO QUE ELIMINA LA ENVOLVENTE.

La envolvente del espectro la tenemos codificada con los LARc. Con este filtro pretendemos eliminarla para así seguir codificando el resto de parámetros. El esquema de dicho filtro lo podemos ver en la figura 3.6 que se corresponde con las ecuaciones que mostramos a continuación:

$d_0 (k) = s (k)$
$u_0 (k) = s (k)$
$d_i (k) = d_{i-1} (k) + r'_i * u_{i-1} (k-1)$
$u_i (k) = u_{i-1} (k-1) + r'_i * d_{i-1} (k)$
$d (k) = d_8 (k)$

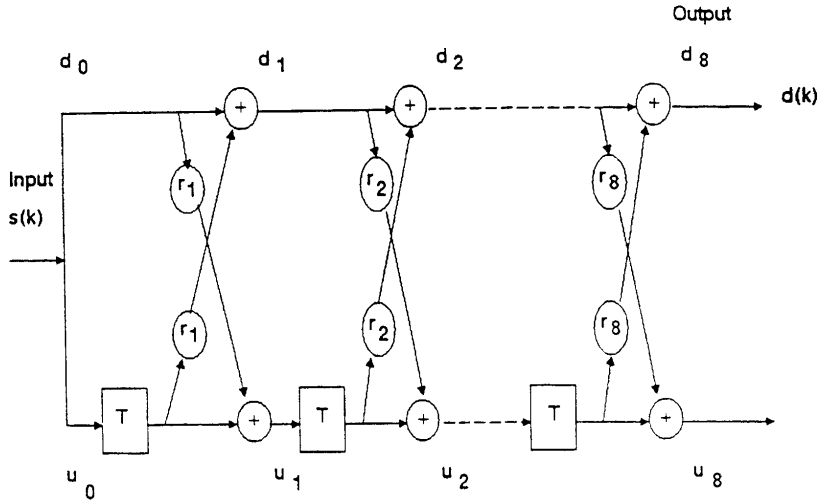


Fig 3.6 Esquema del filtro STP.

3.4 SECCIÓN LTP.

3.4.1 SUBSEGMENTACIÓN.

Cada trama de entrada tras el filtrado STP contiene 160 muestras, correspondientes a 20 ms. La correlación larga (LTP) es calculada cuatro veces por trama, con una duración, por tanto, de 20 ms. De aquí en adelante usaremos la siguiente notación, designaremos por *j = 0 ... 3* el subsegmento actual, por lo que las muestras pertenecientes al j-ésimo subsegmento de la señal sin envolvente se denotarán a partir de ahora por *d (kj + k)* donde *kj= ko + j*40* con *j = 0 ... 3 , k = 0 ... 39* y siendo *ko* la primera muestra de cada subsegmento.

3.4.2 CÁLCULO DE LOS PARÁMETROS LTP.

Para cada uno de los cuatro subsegmentos se calcula un retraso LTP designado por *Nj (j = 0 ... 3)* y una ganancia LTP asociada *bj*. Para cada subsegmento, la determinación de estos parámetros se implementa en tres pasos:

1) El primer paso es la evaluación de la correlación *Rj (lambda)* del actual subsegmento *d (kj + i)* y las 120 muestras anteriores reconstruidas *d'(kj + i)*.Para la primera trama *d'(kj + i)* son cero.

$$Rj(lambda) = \sum_{i=0}^{39} d(kj+i) * d'(kj+i-lambda); \quad \begin{matrix} j = 0 \ldots 3 \\ kj = ko + j*40 \\ lambda = 40 \ldots 120 \end{matrix}$$

Esta correlación es evaluada para retrasos mayores a 40 y menores a 120.

2) El segundo paso es encontrar el retraso *Nj*. Para ello se maximiza la función *Rj* en la variable *lambda.*

   $Rj(Nj) = max\{ Rj(lambda); lambda = 40 \ldots 120 \}$

   $j = 0 \ldots 3$

3) El tercer paso es la evaluación de la ganancia LTP. Ello se calcula de acuerdo con la siguiente ecuación.

   $bj = Rj(Nj) / Sj(Nj) \qquad j = 0 \ldots 3$

   siendo

$$Sj(Nj) = \sum_{i=0}^{39} [d'(Kj + i - Nj)]^2 \qquad j = 0 \ldots 3$$

3.4.3 CODIFICACIÓN DE LOS RETRASOS LTP.

Los retrasos LTP están comprendidos en el rango (40 ... 120) por lo que son codificados usando 7 bits. Son enviados al receptor.

Ncj = Nj

3.4.4 CODIFICACIÓN DE LAS GANANCIAS LTP.

Las ganancias LTP son codificadas usando 2 bits de acuerdo con el siguiente algoritmo:

if bj <= DLB (i) then bcj = 0; i = 0

if DLB (i-1) < bj <= DLB (i) then bcj = i; i = 1,2

if DLB (i-1) < bj then bcj = 3; i = 3

Donde *DLB (i), (i = 0 ... 2)* denota los niveles de decisión del cuantizador y *bcj* representa el valor de la ganancia codificado. Los niveles de decisión y cuantización se ilustran el la tabla 3.4. Las ganancias *bc* son enviadas al receptor.

i	*DLB (i)*	*QLB (i)*
0	*0.2*	*0.10*
1	*0.5*	*0.35*
2	*0.8*	*0.65*
3		*1.00*

Tabla 3.4 Tabla de cuantización LTP.

3.4.5 DECODIFICACIÓN DE LOS RETRASOS LTP.

Para ello se decodifican los siete bits y se recupera el valor del retraso LTP.

Nj' = Ncj'

3.4.6 DECODIFICACIÓN DE LAS GANANCIAS LTP.

Las ganancias LTP son decodificadas de acuerdo con la siguiente ecuación:

bj' = QLB (bcj)

donde *QLB (i)* representa el nivel de cuantización (véase tabla 3.4) y *bj'* la ganancia decodificada.

3.4.7 FILTRADO DE ANÁLISIS LTP.

La señal sin envolvente *d (ko + k)* es procesada en subsegmentos de 40 muestras. Para cada uno de los cuatro subsegmentos es estimada una señal reconstruida *d''(kj + k)* y calculado un error entre ambas señales, que será la excitación residual, tal como denotamos en las siguientes ecuaciones:

$$d''(kj + k) = bj' * d'(kj + k - Nj') \qquad j = 0 \ldots 3 \quad k = 0 \ldots 39 \quad kj = ko + j * 40$$

$$e(kj + k) = d(kj + k) - d''(kj + k) \qquad j = 0 \ldots 3 \quad k = 0 \ldots 39 \quad kj = ko + j * 40$$

3.4.8 FILTRADO DE SÍNTESIS LTP.

Esta excitación residual *e (kj + k)* resulta de eliminar de la señal original su envolvente y su pitch por lo que sólo nos quedamos con la excitación residual. La mandamos hacia el codificador RPE, el cual nos devuelve dicho residuo decodificado, el que recibiría el decodificador, y con ella actualizamos las 120 muestras anteriores reconstruidas necesarias para el análisis LTP de la siguiente subtrama. La ecuación que sigue es la siguiente:

$$d'(kj + k) = e'(kj + k) + d''(kj + k) \qquad j = 0 \ldots 3 \quad k = 0 \ldots 39 \quad kj = ko + j * 40$$

3.5 SECCIÓN RPE.

3.5.1 FILTRADO DE PONDERACIÓN.

A las 40 muestras del subsegmento de la excitación residual le aplicamos un filtro FIR de orden diez cuya respuesta impulsional viene dada por los coeficientes del filtro en la tabla 3.5.

i	*5*	*4 (6)*	*3 (7)*	*2 (8)*	*1 (9)*	*0 (10)*
$H(i) * 2^{13}$	*8192*	*5741*	*2054*	*0*	*-374*	*-134*

Tabla 3.5 Coeficientes del filtro de Ponderación.

La convolución de estas 40 muestras con un filtro de longitud 11 produce 50 muestras. Nos quedaremos con las 40 muestras centrales. La fórmula viene dada por:

$$x(k) = \sum_{i=0}^{10} H(i) * e(k+5-i) \qquad k = 0 \ldots 39$$

Nota: $e(k+5-i) = 0$ *para* $k+5-i<0$ *y* $k+5-i>39$

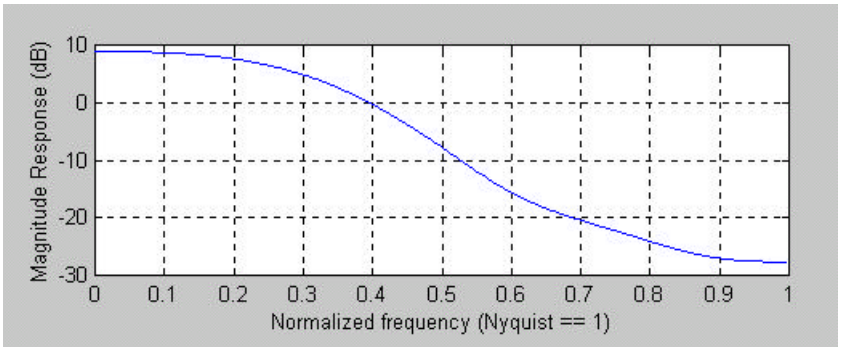


Fig 3.7 Filtro de Ponderación.

3.5.2 DIEZMADO DE LAS MUESTRAS.

En la siguiente etapa, la señal filtrada *x* es diezmada por un factor de tres resultando tres secuencias entrelazadas de longitud 14, 13 y 13, que son de nuevo ordenadas en cuatro subsecuencias de longitud 13 de la siguiente manera:

$$xm(i) = x(kj+m+3*i) \qquad i = 0 \ldots 12 \quad m = 0 \ldots 3$$

donde m denota el número de la subsecuencia.

La subsecuencia seleccionada será la de máxima energía entre las cuatro, para ello se calcula la energía de las cuatro subsecuencias:

$$Em = \max_{m} \sum_{i=0}^{12} xm^2(i) \qquad m = 0 \ldots 3$$

El número *m* es codificado con 2 bits, se denomina *Mc* y se envía.

3.5.3 CUANTIZACIÓN APCM.

La subsecuencia seleccionada *xm(i)* (secuencia RPE) es cuantizada usando codificación APCM (Modulación por Pulsos Codificados Diferencial). Para cada secuencia RPE consistente en un conjunto de 13 muestras, el valor absoluto máximo de la secuencia | *xm (i)* | es seleccionado, cuantizado logarítmicamente y codificado con 6 bits como *xmaxc* para su envío al receptor. La tabla 3.6 nos ilustra el rango de valores.

Posteriormente los 13 valores de la secuencia seleccionada son divididos por el valor *x'max* que resulte.

$$x'(i) = xm(i) / x'max \qquad i = 0 \ldots 12$$

y los treces valores resultantes son cuantizados uniformemente y codificados como *xmc (i)* para su envío al receptor, tal como ilustra la tabla 3.7.

3.5.4 CUANTIZACIÓN INVERSA.

Los valores codificados *xmc (i)* son decodificados obteniendo *xm'(i)*, al igual que el valor *xmaxc*, obteniendo *x'max*, y denormalizando obtenemos así la secuencia RPE decodificada *x'm(i)*.

$$x'm(i) = xm'(i) * x'max$$

3.5.5 RECONSTRUCCIÓN DE LA SECUENCIA RPE.

La secuencia resultante es rellenada con ceros con un factor de tres y de acuerdo con las posiciones que indica el valor *Mc* decodificado, obteniendo así la excitación residual decodificada.

$$e'(kj+Mc+3*i) = x'm(i) \qquad i = 0 \ldots 12$$

$$e'(kj+Mc+3*i) = 0 \qquad \text{Para el resto.}$$

xmax	*x'max*	*xmaxc*	*xmax*	*x'max*	*xmaxc*
0 ... 31	*31*	*0*	*2048 ... 2303*	*2303*	*32*
32 ... 63	*63*	*1*	*2304 ... 2559*	*2559*	*33*
64 ... 95	*95*	*2*	*2560 ... 2815*	*2815*	*34*
96 ... 127	*127*	*3*	*2816 ... 3071*	*3071*	*35*
128 ... 159	*159*	*4*	*3072 ... 3327*	*3327*	*36*
160 ... 191	*191*	*5*	*3328 ... 3583*	*3583*	*37*
192 ... 223	*223*	*6*	*3584 ... 3839*	*3839*	*38*
224 ... 255	*255*	*7*	*3840 ... 4095*	*4095*	*39*
256 ... 287	*287*	*8*	*4096 ... 4607*	*4607*	*40*
288 ... 319	*319*	*9*	*4608 ... 5119*	*5119*	*41*
320 ... 351	*351*	*10*	*5120 ... 5631*	*5631*	*42*
352 ... 383	*383*	*11*	*5632 ... 6143*	*6143*	*43*
384 ... 415	*415*	*12*	*6144 ... 6655*	*6655*	*44*
416 ... 447	*447*	*13*	*6656 ... 7167*	*7167*	*45*
448 ... 479	*479*	*14*	*7168 ... 7679*	*7679*	*46*
480 ... 511	*511*	*15*	*7680 ... 8191*	*8191*	*47*
512 ... 575	*575*	*16*	*8192 ... 9215*	*9215*	*48*
576 ... 639	*639*	*17*	*9216 ... 10240*	*10239*	*49*
640 ... 703	*703*	*18*	*10241 ... 11263*	*11263*	*50*
704 ... 767	*767*	*19*	*11263 ... 12287*	*12287*	*51*
768 ... 831	*831*	*20*	*12288 ... 13311*	*13311*	*52*
832 ... 895	*895*	*21*	*13312 ... 14335*	*14335*	*53*
896 ... 959	*959*	*22*	*14336 ... 15359*	*15359*	*54*
960 ... 1023	*1023*	*23*	*15360 ... 16386*	*16383*	*55*
1024 ... 1151	*1151*	*24*	*16384 ... 18431*	*18431*	*56*
1152 ... 1279	*1279*	*25*	*18432 ... 20479*	*20479*	*57*
1280 ... 1407	*1407*	*26*	*20480 ... 22527*	*22527*	*58*
1408 ... 1535	*1535*	*27*	*22528 ... 24575*	*24575*	*59*
1536 ... 1663	*1663*	*28*	*24576 ... 26623*	*26623*	*60*
1664 ... 1791	*1791*	*29*	*26624 ... 28671*	*28671*	*61*
1792 ... 1919	*1919*	*31*	*28672 ... 30719*	*30719*	*62*
1920 ... 2047	*2047*	*31*	*30720 ... 32767*	*32767*	*63*

Tabla 3.6 Cuantización de las secuencias RPE.

x'(i)	*xm'(i)*	*xmc(i)*
-32768 ... –24577	*-28672*	*0 = 000*
-24576 ... –16385	*-20480*	*1 = 001*
-16384 ... –8193	*-12288*	*2 = 010*
-8192 ... –1	*-4096*	*3 = 011*
0 ... 8191	*4096*	*4 = 100*
8192 ... 16386	*12288*	*5 = 101*
16384 ... 24575	*20480*	*6 = 110*
24576 ... 32767	*28672*	*7 = 111*

Tabla 3.7 Cuantización de las muestras RPE normalizadas.

Por último, vamos a presentar una tabla en la que se recogen los bits de salida del codificador, en el orden en el que salen y a los parámetros a los que pertenecen. Son en total 260 bits cada 20 ms que corresponden a una tasa de 13 Kbps. Posteriormente se le añadirá una codificación de canal que modificará esa tasa, y por ello la importancia de esta tabla, ya que en el capítulo dedicado a la codificación de canal, veremos lo primero que se hace con estos 260 bits es reordenarlos por su importancia para posteriormente aplicarles una protección que dependerá de su importancia.

Parámetro	*Parámetro Nº*	*Nombre del parámetro*	*Nombre de la variable*	*Número de bits*	*Bit Nº (LSB-MSB)*
Parámetros STP	*1*	*LAR 1-8*	*LAR1*	*6*	*b1-b6*
	2		*LAR2*	*6*	*b7-b12*
	3		*LAR3*	*5*	*b13-b17*
	4		*LAR4*	*5*	*b18-b22*
	5		*LAR5*	*4*	*b23-b26*
	6		*LAR6*	*4*	*b27-b30*
	7		*LAR7*	*3*	*b31-b33*
	8		*LAR8*	*3*	*b34-b36*
Subtrama Nº 1					
Parámetros LTP	*9*	*Retraso LTP*	*N1*	*7*	*b37-b43*
	10	*Ganancia LTP*	*b1*	*2*	*b44-b45*
Parámetros RPE	*11*	*Posición RPE*	*M1*	*2*	*b46-b47*
	12	*Valor max*	*xmax1*	*6*	*b48-b53*
	13	*Pulso nº1*	*x1(0)*	*3*	*b54-b56*
	14	*Pulso nº2*	*x1(1)*	*3*	*b57-b59*
	...	*...*	*...*	*...*	*...*
	25	*Pulso nº13*	*x1(12)*	*3*	*b90-b92*
Subtrama Nº 2					
Parámetros LTP	*26*	*Retraso LTP*	*N2*	*7*	*b93-b99*
	27	*Ganancia LTP*	*b2*	*2*	*b100-b101*
Parámetros RPE	*28*	*Posición RPE*	*M2*	*2*	*b102-b103*
	29	*Valor max*	*Xmax2*	*6*	*b104-b109*
	30	*Pulso nº1*	*x2(0)*	*3*	*b110-b112*
	31	*Pulso nº2*	*x2(1)*	*3*	*b113-b115*
	...	*...*	*...*	*...*	*...*
	42	*Pulso nº13*	*x2(12)*	*3*	*b146-b148*
Subtrama Nº3					
Parámetros LTP	*43*	*Retraso LTP*	*N3*	*7*	*b149-b155*
	44	*Ganancia LTP*	*b3*	*2*	*b156-b157*
Parámetros RPE	*45*	*Posición RPE*	*M3*	*2*	*b158-b159*
	46	*Valor max*	*xmax3*	*6*	*b160-b165*
	47	*Pulso nº1*	*x3(0)*	*3*	*b166-b168*
	48	*Pulso nº2*	*x3(1)*	*3*	*b169-b171*
	...	*...*	*...*	*...*	*...*
	59	*Pulso nº13*	*x3(12)*	*3*	*b202-b204*
Subtrama Nº4					
Parámetros LTP	*60*	*Retraso LTP*	*N4*	*7*	*b205-b211*
	61	*Ganancia LTP*	*b4*	*2*	*b212-b213*
Parámetros RPE	*62*	*Posición RPE*	*M4*	*2*	*b214-b215*
	63	*Valor max*	*xmax4*	*6*	*b216-b221*
	64	*Pulso nº1*	*x4(0)*	*3*	*b222-b224*
	65	*Pulso nº2*	*x4(1)*	*3*	*b225-b227*
	...	*...*	*...*	*...*	*...*
	76	*Pulso nº13*	*x4(12)*	*3*	*b258-b260*

Tabla 3.8 Parámetros del codificador ordenados por orden de salida.

Capítulo 4 .

EL DETECTOR DE ACTIVIDAD VOCAL

4.1 INTRODUCCIÓN.

La función básica del Detector de Actividad Vocal[2], de aquí en adelante lo designaremos por sus siglas, VAD, es distinguir entre tramas con ruidos y voz presentes y tramas de ruido puro. La principal dificultad para detectar voz en una comunicación móvil es la pequeña SNR que es a menudo ofrecida al receptor. La eficiencia del VAD es elevada usando filtros para incrementar esta SNR antes de la decisión.

En una comunicación móvil, las peores SNR son encontradas en vehículos móviles. Se ha descubierto que el ruido es relativamente estacionario durante largos periodos, y esto es pernicioso para la detección puesto que una manera de detectar voz es detectar estacionaridad. Sin embargo, es posible usar filtros adaptativos con coeficientes obtenidos durante tramas de ruido, para ignorar el ruido de muchas fuentes externas.

El VAD es básicamente un detector de energía. La energía de la señal filtrada es comparada con un umbral, identificándose voz cuando no se supera el umbral.

Sin embargo, el ruido encontrado en comunicaciones móviles está constantemente cambiando de nivel. El espectro frecuencial del ruido también va cambiando, y es muy diferente de unos vehículos a otros. Es por ello que los coeficientes del filtro adaptativo y el umbral van cambiando constantemente. Para lograr una buena detección, el umbral debe estar lo suficientemente por encima del ruido para evitar que ruido sea identificado como voz pero no debe estar muy por encima para evitar que tramas de voz con un nivel bajo sean identificadas como tramas de ruido.

El umbral y los coeficientes del filtro son sólo actualizados durante periodos de ruido. Es especialmente peligroso que el VAD cambie estos valores antes de su decisión.

También es usado un hangover para eliminar los transitorios de ruido a voz y de voz a ruido y evitar así que el principio de una trama de voz sea confundido con ruido.

4.2 DESCRIPCIÓN DEL ALGORITMO.

El diagrama de bloques del VAD se ilustra en la figura 4.1. Los bloques individuales se describen en las siguientes secciones. Los coeficientes *ACF, N y sof* son calculados en el codificador.

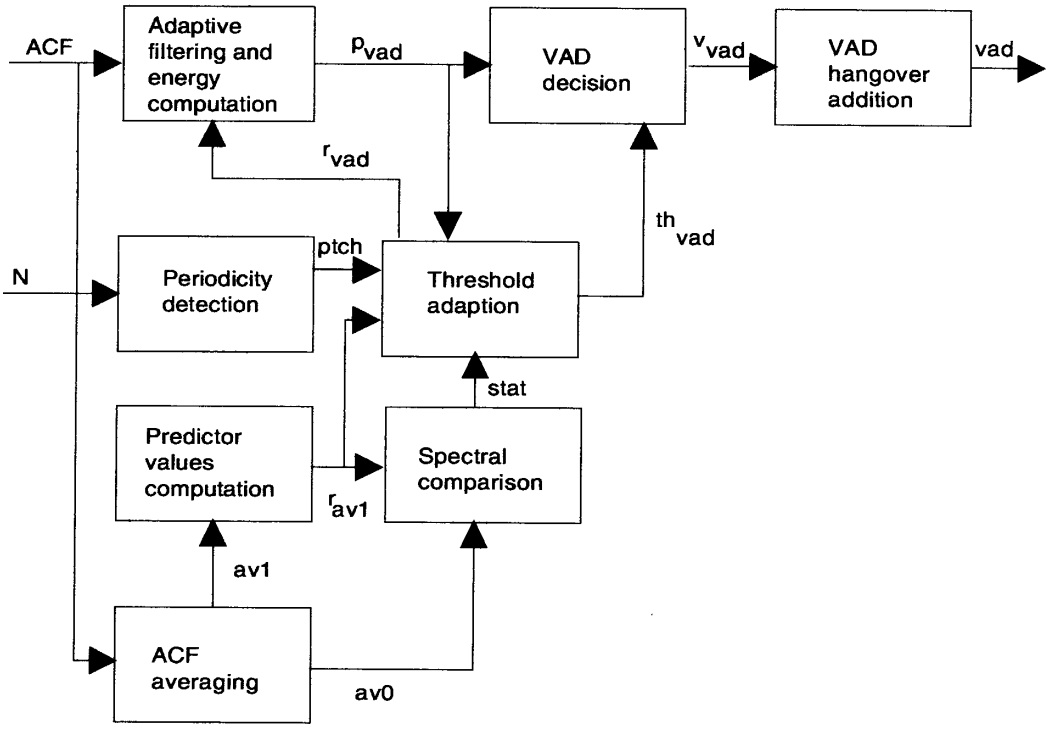


Fig 4.1 Diagrama de bloques del VAD.

Las variables globales que se muestran en esta diagrama son las siguientes:

- *ACF* son los coeficientes de autocorrelación calculados en el codificador.
- *av0* y *av1* son vectores de autocorrelación.
- *rav1* son valores obtenidos de av1.
- *rvad* son los valores para el filtrado adaptativo.
- *N* es el retraso LTP obtenido en el codificador.
- *ptch* indica que la señal tiene una componente periódica.
- *sof* es la señal sin offset calculada en el codificador.

4.2.1 FILTRADO ADAPTATIVO Y CÁLCULO DE LA ENERGÍA.

Pvad se computa como sigue:

$$Pvad = rvad[0]\,ACF[0] + 2\sum_{i=1}^{8} rvad[i]\,ACF[i]$$

4.2.2 MEDIA DE LOS ACF.

Las características espectrales de la señal de entrada han de ser obtenidas usando bloques más extensos que la duración de una trama que es de 20 ms. Esto se realiza mediante el cálculo de la media de los coeficientes de autocorrelación de varias tramas consecutivas de acuerdo con las siguientes ecuaciones:

$$av0\{n\}[i] = \sum_{j=0}^{frames-1} ACF\{n-j\}[i] \qquad ;i = 0 \ldots 8$$

$$av1\{n\}[i] = av0\{n - frames\}[i] \qquad ;i = 0 \ldots 8$$

Donde *n* representa la trama actual, *n-1* representa la trama anterior, etc. Las constantes y valores iniciales se dan en la tabla 4.1.

Constante	*Valor*	*Variable*	*Valor inicial*
frames	*4*	*ACF's anteriores* *Av0 y av1*	*Todos a 0*

Tabla 4.1 Constantes y valores iniciales para la media de los ACF.

4.2.3 CÁLCULO DE LOS VALORES DE PREDICCIÓN.

Los valores de predicción se obtienen a partir de los valores de autocorrelación *av1* de acuerdo con la siguiente ecuación:

$$\begin{Bmatrix} aav1(1) \\ aav1(2) \\ aav1(3) \\ aav1(4) \\ aav1(5) \\ aav1(6) \\ aav1(7) \\ aav1(8) \end{Bmatrix} = \begin{Bmatrix} av1(0) & av1(1) & av1(2) & av1(3) & av1(4) & av1(5) & av1(6) & av1(7) \\ av1(1) & av1(0) & av1(1) & av1(2) & av1(3) & av1(4) & av1(5) & av1(6) \\ av1(2) & av1(1) & av1(0) & av1(1) & av1(2) & av1(3) & av1(4) & av1(5) \\ av1(3) & av1(2) & av1(1) & av1(0) & av1(1) & av1(2) & av1(3) & av1(4) \\ av1(4) & av1(3) & av1(2) & av1(1) & av1(0) & av1(1) & av1(2) & av1(3) \\ av1(5) & av1(4) & av1(3) & av1(2) & av1(1) & av1(0) & av1(1) & av1(2) \\ av1(6) & av1(5) & av1(4) & av1(3) & av1(2) & av1(1) & av1(0) & av1(1) \\ av1(7) & av1(6) & av1(5) & av1(4) & av1(3) & av1(2) & av1(1) & av1(0) \end{Bmatrix}^{-1} * \begin{Bmatrix} av1(1) \\ av1(2) \\ av1(3) \\ av1(4) \\ av1(5) \\ av1(6) \\ av1(7) \\ av1(8) \end{Bmatrix}$$

Los valores de predicción se calculan finalmente de la siguiente forma:

$$rav1[i] = \sum_{k=0}^{8-i} aav1[k]\ aav1[k+i] \qquad ; i = 0 \ldots 8$$

4.2.4 COMPARACIÓN ESPECTRAL.

El espectro representado por los valores de predicción *rav1* y la media de los *ACF, av0,* son comparados usando la distorsión media *dm* definida más adelante. Esta variable produce un valor booleano cada 20 ms, tal como se describe en las siguientes ecuaciones:

$$dm = (rav1[0]\ av0[0] + 2 \sum_{i=1}^{8} rav1[i]\ av0[i]) / av0[0]$$

$$difference = | dm - lastdm |$$

$$lastdm = dm$$

$$stat = difference < thresh$$

Las constantes y valores iniciales se dan en la tabla 4.2

Constante	*Valor*	*Variable*	*Valor inicial*
thresh	*0.05*	*lastdm*	*0*

Tabla 4.2 Constantes y valores iniciales para la comparación espectral.

4.2.5 DETECCIÓN DE PERIODICIDAD.

El espectro en frecuencia del ruido móvil es relativamente estacionario sobre largos periodos. Los valores para el filtrado adaptativo son sólo actualizados cuando se detecta esta estacionaridad. Sin embargo, los sonidos de vocales también poseen esta estacionaridad, pero pueden ser excluídos detectando su periodicidad a través de los retrasos LTP (Nj) que son obtenidos del codificador. Los retrasos LTP consecutivos son comparados. Los casos en que un retraso LTP sea un factor de otro son excluídos, sin embargo los casos en que tengan un factor común no lo son. Este caso no es importante para una entrada de voz pero puede causar errores en la detección para ondas sinusoidales. La variable *ptch* es actualizada cada 20 ms y es verdadera cuando la periodicidad es detectada. Este cálculo se hace de acuerdo con las siguientes ecuaciones:

$$ptch = oldlagcount + veryoldlagcount >= nthresh$$

Las siguientes operaciones son realizadas antes de la decisión del VAD y cuando los retrasos LTP *(N0 ... N3)* están disponibles, esto reduce el retraso en la decisión del VAD. *(N (-1) = N3* del segmento anterior).

lagcount = 0

for j= 0 to 3 do
begin
smallag = maximun (N (j), N {j-1}) mod minimun (N (j), N {j-1})
if minimun (smallag , minimun (N (j),N{j-1} – smallag) < lthresh
then increment lagcount
end

veryoldlagcount = oldlagcount

oldlagcount = lagcount

Las constantes y valores iniciales se dan en la tabla 4.3.

Constante	*Valor*	*Variable*	*Valor inicial*
lthresh	*2*	*oldlagcount*	*0*
nthresh	*4*	*veryoldlagcount*	*0*
		N3	*40*

Tabla 4.3. Constantes y valores iniciales para la detección de periodicidad.

4.2.6 ADAPTACIÓN DE UMBRAL.

Cada 20 ms se realiza una comprobación para determinar si el umbral ha de ser modificado. Esta adaptación se realiza tal y como se describe en el diagrama de flujo de la figura 4.2. Las constantes y valores iniciales se dan en la tabla 4.4.

Constante	*Valor*	*Variable*	*Valor Inicial*
ptch	*300000*	*adapcount*	*0*
plev	*800000*	*thvad*	*1000000*
fac	*3.0*	*rvad[0]*	*6*
adp	*8*	*rvad[1]*	*-4*
inc	*16*	*rvad[2]*	*1*
dec	*32*	*rvad[3] a*	
margin	*80000000*	*rvad[8]*	*Todos a 0*

Tabla 4.4 Constantes y Valores iniciales para la adaptación de umbral.

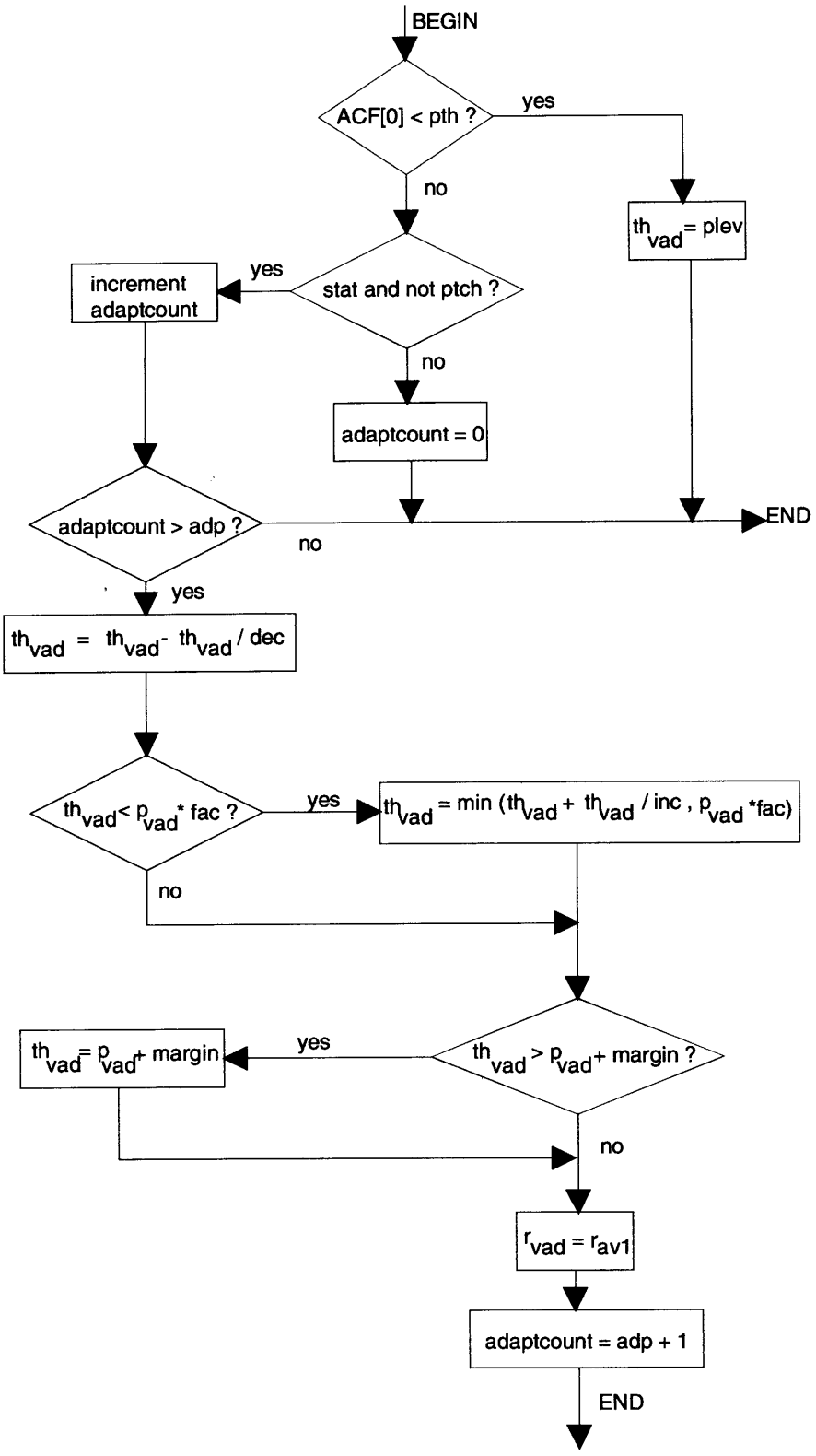


Fig 4.2. Diagrama de flujo de la adaptación de umbral.

4.2.7 DECISIÓN DEL VAD.

La condición para que la trama procesada sea considerara como voz es que la variable booleana *vvad* sea verdadera, en otro caso, se considera una trama de ruido. La comparación para obtenerla es la siguiente:

vvad = pvad > thvad

4.2.8 ADICIÓN DE HANGOVER.

El hangover es sólo añadido cuando el tamaño de la trama supera la variable burstcount. La variable *vad* indica la decisión del VAD con hangover incluido. Las constantes y valores iniciales se dan en la tabla 4.5. El algoritmo de adición de hangover es el que se muestra a continuación:

```
if vvad then increment (burstcount) else burstcount = 0

if burstcount >= hangcount then
begin
        hangcount = hangconst
        burstcount = burstconst
end

vad = vvad or (hangcount >=0)

if hangcount >0 then decrement (hangcount)
```

Constante	*Valor*	*Variable*	*Valor Inicial*
burstconst	*3*	*burstcount*	*0*
hangconst	*5*	*hangcount*	*-1*

Tabla 4.5 Constantes y Valores iniciales para la adición de hangover.

4.3 MONTAJE DEL VAD.

La figura 4.3 muestra el montaje del Detector de Actividad Vocal funcionando conjuntamente con el codificador RPE-LTP. La salida del codificador son 260 bits de la voz codificada y la salida del VAD es un bit que indica la decisión: "0" si no hay presencia vocal y "1" si la hay.

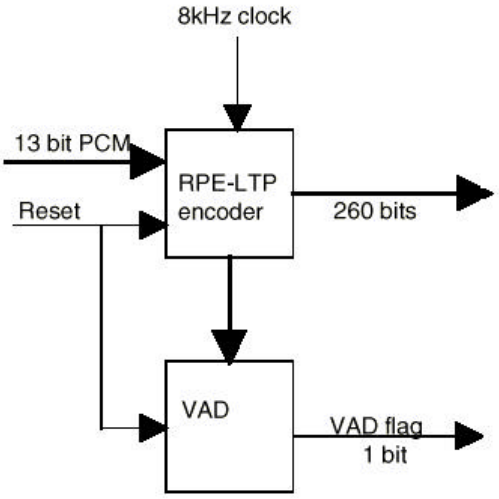


Fig 4.3 Esquema de montaje del codificador con el VAD.

4.4 TRAMAS SID.

La salida del codificador RPE-LTP son 260 bits, los cuales son modificados de la siguiente forma. Si la variable *vad* está a cero durante cuatro tramas consecutivas, se procede a construir una trama de silencio, llamada comúnmente trama SID (Silence Descriptor). Para ello, los parámetros de salida del codificador son sustituidos por otros tal como se describe a continuación[4]:

Cada *LARc(i)* se sustituyen por el valor medio de los *LARc(i)* de las cuatro tramas marcadas con *vad* a cero. (36 bits).

Los cuatro parámetros correspondientes a *xmaxc* se sustituyen por el valor medio de los *xmaxc(i)* de las cuatro tramas marcadas con *vad* a cero. (24 bits).

Los parámetros *Nc, bc, Mc y xmc* son puestos a cero. (200 bits).

Esto supone un ahorro de 200 bits en la transmisión, que quedan libres y cuyo uso tiene que ver con el protocolo. Éstos 260 bits de salida de la trama SID son tratados posteriormente igual que los 260 bits de salida del codificador.

4.5 RUIDO CONFORMADOR.

Al decodificador llegan los 260 bits de la voz codificada y la variable *vad* que indica si hay presencia vocal. Si la variable *vad* recibida está a cero durante cuatro tramas consecutivas, indica que hay una trama de silencio, en cuyo caso se actúa de la siguiente forma[4]:

Los pulsos RPE *(xmc)* son sustituidos por una secuencia de números enteros, uniformemente distribuidos entre 1 y 6.

Las posiciones RPE (*Mc*) son sustituidas por números enteros aleatorios uniformemente distribuidos entre 0 y 3.

Las ganancias LTP (*bc*) son puestas a cero.

Los retrasos LTP (*Nc*) son puestos a 40, 120, 40 y 120 respectivamente.

Los LARc y los valores máximos (*xmaxc*) son los que se reciben en las tramas SID.

Capítulo 5 .
CODIFICACIÓN DE CANAL

5.1 INTRODUCCIÓN.

En el sistema GSM se usa una estructura jerárquica de tramas[3]. Los intervalos de tiempo son multiplexados en tramas TDMA, y éstas a su vez en multitramas, supertramas e hipertramas. Veremos cómo los bits de la voz codificada son mapeados en intervalos de tiempo de tramas TDMA. Centraremos nuestra atención en el mecanismo de corrección de errores hacia delante (FEC) y el esquema de entrelazado usado para las transmisiones de voz.

La salida del codificador RPE-LTP produce 260 bits cada 20 ms a una tasa, por tanto, de 13 Kbps. El detector de actividad vocal produce 1 bit cada 20 ms, indicando si ha detectado voz o no. Si no ha detectado voz durante cuatro tramas consecutivas, se construye una trama SID, de 260 bits también, tal como hemos visto. En definitiva, tenemos 260 bits y le aplicaremos una codificación de canal para adaptarlos al formato de una trama para poder transmitirla a la interfaz radio. El receptor conoce a través del protocolo si es o no una trama de silencio, y por tanto, el valor de la variable *vad*. Veremos los distintos procesos aplicados en la codificación de canal, que son:

- Reordenación (5.2)
- Código Cíclico (5.3)
- Entrelazado (5.4)
- Código Convolucional (5.5)
- Particionamiento (5.6)
- Entrelazado de bloques (5.7)
- Entrelazado de trama (5.8)

5.2 REORDENACIÓN.

En primer lugar, se procede a reordenar los 260 bits[1] según su importancia, debido a que, por ejemplo, los primeros LAR son más importantes para una buena decodificación, por ello se le aplica una mejor protección frente a otros bits. En la tabla 5.1 se ilustra esta reordenación y agrupación de los bits según su importancia, y en la figura 5.1 observamos tal como queda la trama resultante:

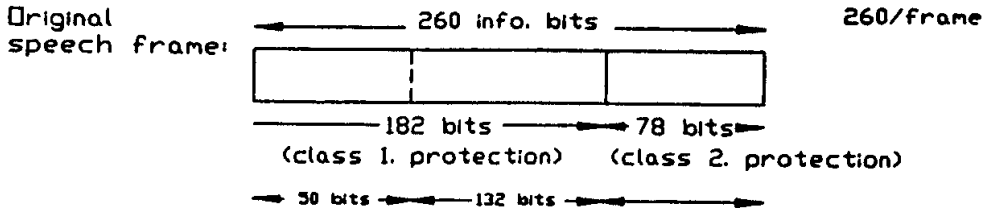


Fig 5.1 Trama de voz original.

Importancia de los bits	*Nombre del parámetro*	*Parámetro Nº*	*Número del bit.*
Clase Ia			
1	*LAR 1*	*1*	*b6*
2,3,4,5	*Valor max*	*12,29,46,63*	*b53,b109,b165,b221*
6	*LAR 1*	*1*	*b5*
7	*LAR 2*	*2*	*b12*
8	*LAR 3*	*3*	*b17*
...	*LAR 1*	*1*	*b4*
...	*LAR 2*	*2*	*b11*
...	*LAR 3*	*3*	*b16*
...	*LAR 4*	*4*	*b22*
...	*Retraso LTP*	*9,26,43,60*	*b43,b99,b155,b211*
...	*Valor max*	*12,29,46,63*	*b52,108,b164,b220*
...	*LAR 2,5,6*	*2,5,6*	*b10,b26,b30*
...	*Retraso LTP*	*9,26,43,60*	*b42,b98,b154,b210*
...	*Retraso LTP*	*9,26,43,60*	*b41,b97,b153,b209*
...	*Retraso LTP*	*9,26,43,60*	*b40,96,b152,b208*
...	*Retraso LTP*	*9,26,43,60*	*b39,b95,b151,b207*
...	*Valor max*	*12,29,46,63*	*b51,b107,b163,b219*
...	*LAR 1*	*1*	*b3*
...	*LAR 4*	*4*	*b21*
...	*LAR 7*	*7*	*b33*
... 49,50	*Retraso LTP*	*9,26,43,60*	*b38,b94,b150,b206*

	Clase Ib		
51,52	*LAR 5,6*	*5,6*	*b25,b29*
...	*Ganancia LTP*	*10,27,44,61*	*b45,b101,b157,b213*
...	*Retraso LTP*	*9,26,43,60*	*b37,b93,b149,b205*
...	*Posición RPE*	*11,28,45,62*	*b47,b103,b159,b215*
...	*LAR1*	*1*	*b2*
...	*LAR 2,3,8,4*	*2,3,8,4*	*b9,b15,b36,b20*
...	*LAR 5,7*	*5,7*	*b24,b32*
...	*Ganancia LTP*	*10,27,44,61*	*b44,b100,b156,b212*
...	*Valor max*	*12,29,46,63*	*b50,b106,b162,b218*
...	*Pulso RPE*	*13 ... 25*	*b56,b59, ... ,b92*
...	*Pulso RPE*	*30 ... 42*	*b112,b115, ... ,b148*
...	*Pulso RPE*	*47 ... 59*	*b168,b171, ... ,b204*
...	*Pulso RPE*	*64 ... 76*	*b224,b227, ... ,b260*
...	*Posición RPE*	*11,28,45,62*	*b46,b102,b158,b214*
...	*Valor max*	*12,29,46,63*	*b49,b105,b161,b217*
...	*Pulso RPE*	*13 ... 25*	*b55,b58, ... ,b91*
...	*Pulso RPE*	*30 ... 42*	*b111,b114, ... ,b147*
...	*Pulso RPE*	*47 ... 59*	*b167,b170, ... ,b203*
... 182	*Pulso RPE*	*64 ... 67*	*b223,b226,b229,b232*
	Clase II		
183 ...	*Pulso RPE*	*68 ... 76*	*b235,b238, ... ,b259*
...	*LAR 1*	*1*	*b1*
...	*LAR 2,3,6*	*2,3,6*	*b8,b14,b28*
...	*LAR 7*	*7*	*b31*
...	*LAR 8*	*8*	*b35*
...	*LAR 8,3*	*8,3*	*b34,b13*
...	*LAR 4*	*4*	*b19*
...	*LAR 4,5*	*4,5*	*b18,b23*
...	*Valor max*	*12,29,46,63*	*b48,b104,b160,b216*
...	*Pulso RPE*	*13 ... 25*	*b54,b57, ... ,b90*
...	*Pulso RPE*	*30 ... 42*	*b110,b113, ... ,b146*
...	*Pulso RPE*	*47 ... 59*	*b166,b169, ... ,b202*
...	*Pulso RPE*	*64 ... 76*	*b222,b225, ... ,258*
259,260	*LAR 2,6*	*2,6*	*b7,b27*

Tabla 5.1 Subdivisión de los bits según su protección

5.3 CÓDIGO CÍCLICO.

Ahora ya tenemos los 260 bits de la trama original ordenados por su importancia en tres clases: C1a, C1b y C2. A los 50 bits de la clase C1a se les aplica un código detector de errores en bloque[6]. El código implementado es un código cíclico (53,50) con el polinomio generador $G4(D) = D^3 + D + 1$, el cual toma 50 bits de entrada y produce a su salida 53 bits, los 50 bits originales más tres bits de paridad capaces de detectar si ha habido errores en el bloque de 50 bits. El correspondiente diagrama del codificador se ilustra en la figura 5.2.

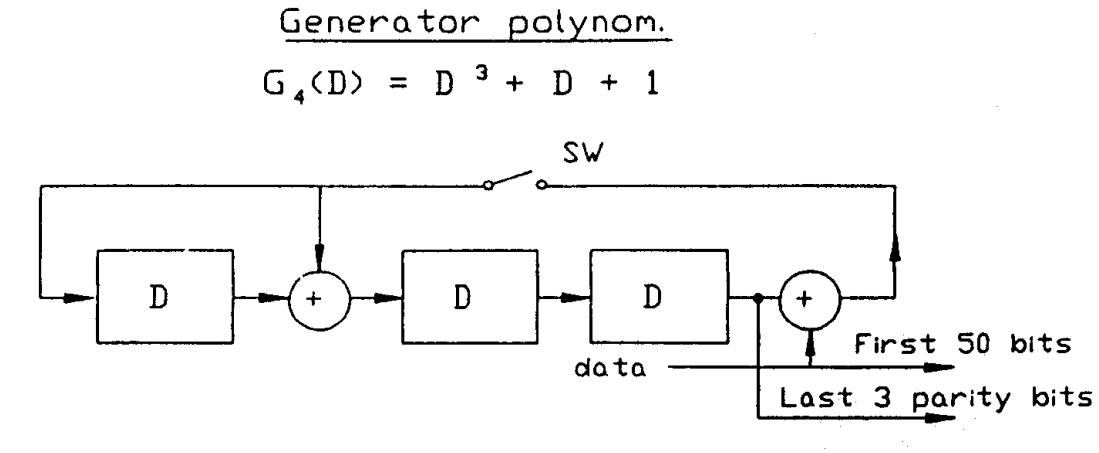


Fig 5.2. Diagrama del codificador (53,50)

El reloj SW permanece cerrado durante los primeros cincuenta pulsos de reloj, y los bits de información van entrando en el codificador. Después de estos cincuenta primeros pulsos, el reloj SW se abre y los bits de paridad P0, P1 y P2 salen del codificador. Estos 3 bits son añadidos tras los 50 bits originales, quedando la trama resultante tal como muestra la figura 5.3.

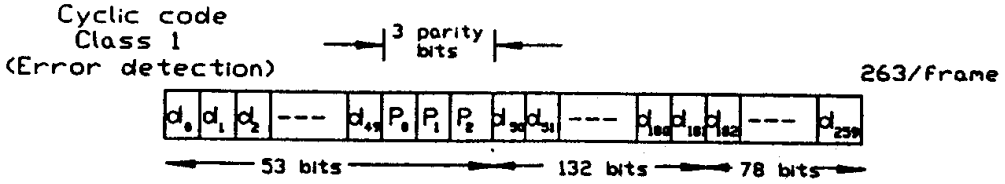


Fig 5.3 Trama resultante tras el CC(53,50)

5.3.1 FUNDAMENTO TEÓRICO.

De aquí en adelante, usaremos las operaciones + y * que servirán para hacer las operaciones módulo2 y que se definen como[7]:

+	**0**	**1**
0	0	1
1	1	0

*	**0**	**1**
0	0	0
1	0	1

A cada palabra mensaje de 50 bits de entrada al codificador, se le asocia un polinomio de grado a lo sumo 49.

$$\underline{x} = [\ x49\ \ x48\ \ ...\ \ x1\ \ x0] \ \Rightarrow\ x(p) = x49*p^{49} + x48*p^{48} + ... + x1*p + x0$$

Posteriormente, el polinomio asociado a cada palabra mensaje lo multiplicamos por p^3, y el resultado lo dividimos por el polinomio generador $g(p)= p^3 + p + 1$, resultando un resto de grado a lo sumo 2, asociado a los tres bits de paridad.

$$r(p) = r(2)*p^2 + r(1)*p + r(0) \ \Rightarrow\ \underline{r} = [\ r2\ r1\ r0]$$

La palabra código de 53 bits que se transmitirá será $\underline{c} = [\ x \mid r\ ]$

Para no tener que hacer todas estas operaciones cada vez que se quiera formar una palabra código, se construye la matriz generadora de códigos de manera sistemática, que tiene 50 filas y 53 columnas y se forma cogiendo una matriz unitaria de orden 50 y hallando, para cada palabra mensaje correspondiente a cada fila, su palabra código correspondiente, por lo que tendremos 50 filas y 53 columnas. Cuando tengamos una palabra mensaje, $\underline{x}$, la multiplicamos por la matriz generadora de códigos $\underline{H}$ y obtendremos la palabra código: $\underline{c} = \underline{x} * \underline{H}$

A la hora de decodificar sólo tendremos que construir la matriz inversa, que tendrá 53 filas y 3 columnas y se forma cogiendo las últimas 3 columnas de la matriz $\underline{H}$, y añadiéndole abajo una matriz unitaria de orden 3. Es decir si:

$$\underline{H} = \{\underline{I} \quad \underline{P}\} \Rightarrow Ht = \begin{Bmatrix} \underline{P} \\ \underline{I} \end{Bmatrix}$$

Una vez construida la matriz $\underline{Ht}$, las palabra código recibidas se multiplican por ella, y obtendremos un vector de 3 elementos, llamado síndrome: $\underline{s} = \underline{c} * \underline{Ht}$.

Si todos los elementos del síndrome son cero, nos indica que la palabra recibida es una palabra de código, aunque pueda haber un error en ella, sigue siendo una palabra de código y no tiene error. En caso contrario, es decir, que alguno de los tres elementos del síndrome sea uno, entonces se notifica de que hay un error detectado. Este código es capaz de corregir dicho error si es sólo en un bit, pero en este caso, no se hace, sólo se notifica de que se ha detectado un error.

5.4 ENTRELAZADO.

Posteriormente, estos 263 bits se someten a un entrelazado resultando 267 bits de salida[6]. Primeramente, los bits pertenecientes a las clases C1a y C1b con índices pares, desde d0 hasta d180, son colocados en la primera parte de la trama, seguidos de los tres bits de paridad calculados anteriormente. Después se colocan los bits pertenecientes a las clases C1a y C1b con índices impares, y en orden descendente, desde d181 hasta d1, seguido de cuatro bits a cero necesarios para resetear el codificador convolucional.Y por último se colocan los bits pertenecientes a la clase C2. La estructura de la trama resultante podemos observarla en la figura 5.4.

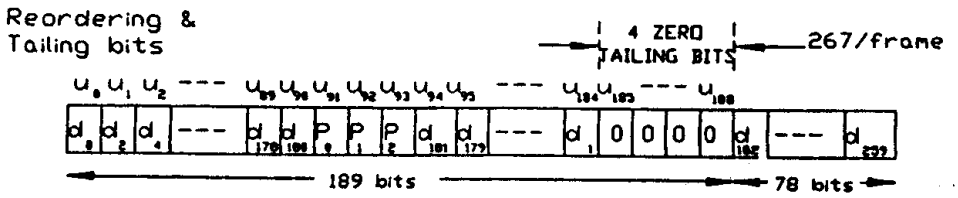


Fig 5.4 Trama resultante tras el entrelazado.

5.5 CÓDIGO CONVOLUCIONAL.

El siguiente paso es aplicar un código convolucional[6] de tasa ½ y profundidad k=5 designado por CC(2,1,5) para proteger los 189 bits de la clase C1 (u0 ... u188). El código convolucional usa los polinomios generadores $G0 = 1 + D^3 + D^4$ y $G1 = 1 + D + D^3 + D^4$. El correspondiente diagrama del codificador se ilustra en la figura 5.5.

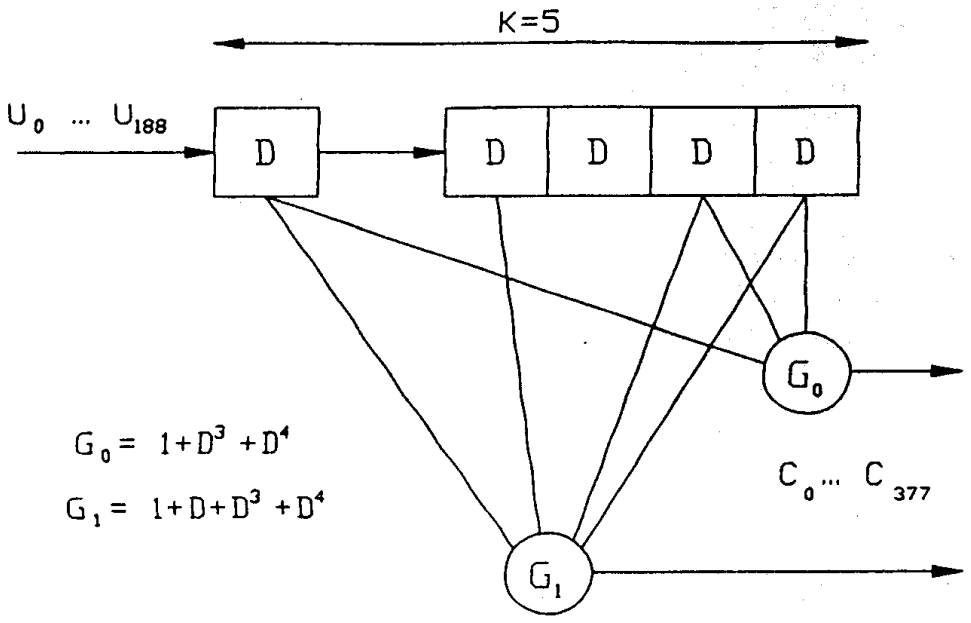


Fig 5.5 Diagrama del codificador (2,1,5)

La longitud total de la trama es, por tanto, de 2*189 + 78 = 456 bits.

La trama resultante se ilustra en la figura 5.6.

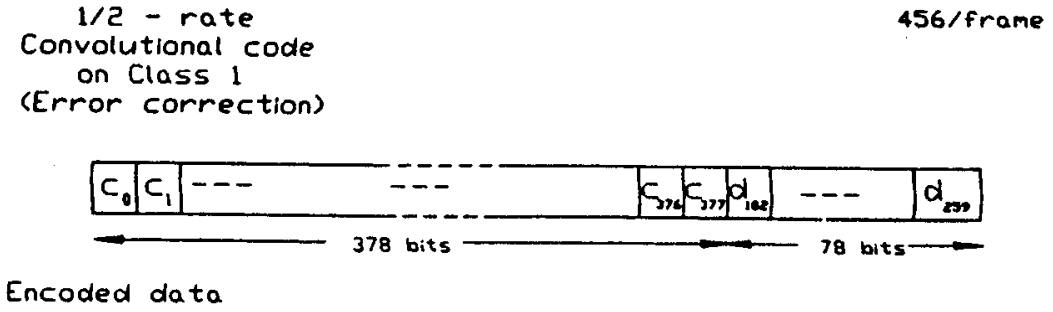


Fig 5.6 Trama resultante tras el CC(2,1,5)

5.5.1 FUNDAMENTO TEÓRICO.

La forma de generar una palabra de código de 378 bits a partir de una palabra mensaje de 179 bits es la que se ilustra en la figura 5.5. Los primeros cuatro bits son todos ceros para resetar el codificador. Vamos a centrarnos en la decodificación, que es capaz de corregir errores y que se basa en un decodificador Viterbi[5].

Un decodificador Viterbi se basa en un árbol con los posibles caminos y pesos asociado a cada camino. La profundidad del árbol puede ser cualquiera, a mayor profundidad tendremos una decodificación más exacta, pero también ralentizará y será más compleja la decodificación. En nuestro caso, y tal como marca la recomendación, la profundidad será de 3 bits. El proceso es el siguiente:

Los primeros cuatro bits recibidos son siempre cero, ya que los primeros cuatro bits transmitidos eran cero. Se ponen esos cuatro primeros bits recibidos en los cuatro registros iniciales del codificador. Se cogen todas las ocho posibles palabras mensaje de tres bits y se calculan para ellas las ocho posibles palabras código de seis bits, originadas por el codificador, con los cuatro registros iniciales tal como se explicó anteriormente. Ahora se toman los primeros seis bits recibidos y se calcula la diferencia con cada una de las ocho palabras código de seis bits generadas anteriormente. La que tenga menor diferencia de las ocho será la elegida, y se toma como bit recibido el primero de los tres bits de las ocho alternativas. Se actualiza los cuatro anteriores bits recibidos y se continúa el proceso recursivamente con los siguientes bits recibidos. Una descripción un poco más exhaustiva en forma de algoritmo se expone a continuación:

```
bit_elegido (1:4) = [ 0 0 0 0]
mensajes_posibles = [ 000 ; 001 ; 010 ; 011 ; 100 ; 101 ; 110 ; 111 ]
registro_inicial = [ 0 0 0 0]
for i = 1:185
        recibidos = bits_recibidos (2i –1:2i+4)
        for j = 1:8
                codigo (j) = codifica ( mensajes_posibles (j), inicial)
                pesos (j) = calculapesos ( recibidos, codigo)
        end for
        mensaje_elegido = elige el mensaje de menor peso ( pesos (j) )
        bit_elegido (i+4)  = primer bit de mensaje_elegido
        registro_inicial = [ bit_elegido | registro_inicial (1:3) ]
end for
```

5.6 PARTICIONAMIENTO.

Los 456 bits resultantes[6] son particionados en 8 bloques de 57 bits (B0 ... B7), tal como muestra la figura 5.7.

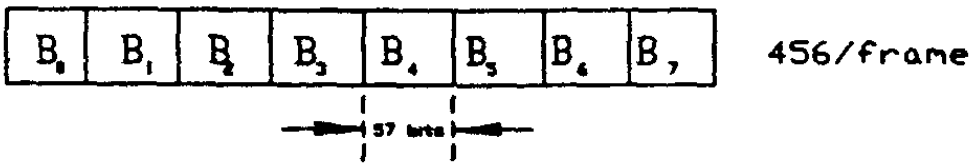


Fig 5.7 Particionado de los 456 bits.

5.7 ENTRELAZADO DIAGONAL DE BLOQUES.

Los 57 bits pertenecientes a un bloque de una trama[6] se unen con 57 bits de otro bloque de la trama anterior, formando así un bloque de 114 bits, a los cuales, posteriormente se les aplicará otro entrelazado de trama. Tal como muestra la figura 5.8, los 57 bits del bloque “0” de la trama actual se unen con los 57 bits del bloque “4” de la trama anterior. Y lo mismo el bloque “1” con el “5”, el “2” con el “6” y el “3” con el “7”. Inicialmente, los bloques “4” al “7” de la trama anterior son todos cero.

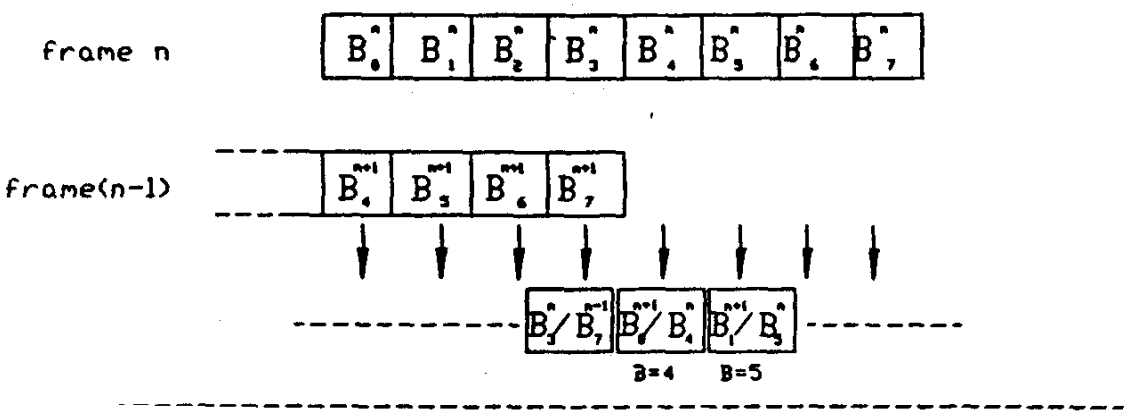


Fig 5.8 Entrelazado diagonal de bloques.

5.8 ENTRELAZADO DE TRAMA.

Los 114 bits resultantes del entrelazado anterior son sometidos de nuevo a otro entrelazado[6], esta vez dentro del mismo bloque de 114 bits. Para ello se coge el índice que tenía cada bit en la trama de 456 bits y se calcula un nuevo índice, de 0 a 113, tal como muestran las siguientes ecuaciones:

k = 0 ... 455
*j = 2 * [49*k mod 57]* *if (k mod 8) <= 3*
*j = 2 * [49*k mod 57] + 1* *if (k mod 8) > 3*

La trama resultante, de 114 bits es la que finalmente viaja por la interfaz radio y se muestra en la figura 5.9. A estos 114 bits bits se le añaden dos banderas, *hu* y *hl* indicando si los 57 bits a los que indica cada una pertenecen a voz o se roban para una ráfaga FACCH. También se le añaden 26 bits para la secuencia de entrenamiento y otros 3+3+8.25=14.25 bits de banderas de inicio y fin, resultando en total 156.25 bits totales, que forman una ráfaga y que se transmiten en un tiempo de 577 μs, resultando por tanto, una tasa de 271 KHz en la interfaz radio.

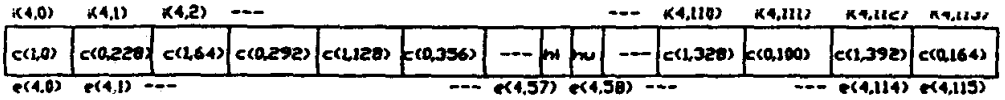


Fig 5.9 Trama final resultante.

Capítulo 6 .
EL DECODIFICADOR RPE-LTP.

Al inverso que en el caso del codificador, en el decodificador RPE-LTP[1] la señal de entrada serán 260 bits cada 20 ms correspondientes a la codificación de los distintos parámetros de la señal de voz. La salida serán 2080 bits cada 20 ms correspondiente a una señal de 8 KHz y 13 bits por muestra.

La mayor parte de los bloques están también incluidos en el codificador. Sólo el filtro de síntesis STP y el filtro de deénfasis son nuevos bloques que han sido añadidos aquí y que no están en el codificador. El decodificador comprende las cuatro siguientes secciones.

- Sección de decodificación RPE. (6.1)
- Sección LTP. (6.2)
- Sección STP (6.3)
- Postprocesado (6.4)

El diagrama de bloques completo del decodificador se ilustra en la figura 6.1. Las variables y parámetros del decodificador están marcados con el subíndice *r* para distinguir los valores recibidos de los valores del codificador.

6.1 SECCIÓN DE DECODIFICACIÓN RPE.

En esta fase reconstruimos la excitación residual, a partir de la cual, se irá construyendo la señal deseada, añadiéndole su pitch y su envolvente del espectro. Esta excitación residual se construye a partir de la decodificación y denormalización de las muestras RPE colocándolas en la posición apropiada indicado por la variable recibida *Mcr*. Al mismo tiempo, se inserta un numero apropiado de ceros para rellenar la secuencia que fue diezmada en el codificador por un factor de tres.

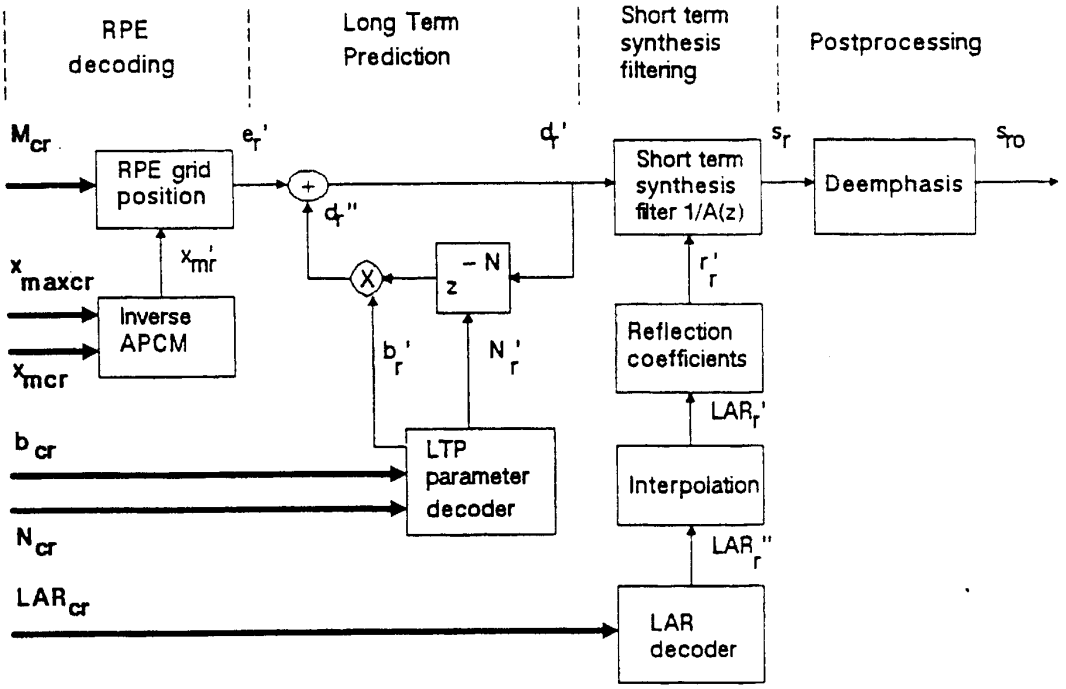


Fig 6.1 Decodificador RPE-LTP.

6.1.1 CUANTIZACIÓN INVERSA.

Los valores codificados *xmcr (i)* son decodificados obteniendo *xmr'(i)*, tal como se observa en la tabla 3.7, al igual que el valor *xmaxcr*, obteniendo *x'maxr*, tal como se observa en la tabla 3.6, y denormalizando obtenemos así la secuencia RPE decodificada *x'mr(i).*

$$x'mr(i) = xmr'(i) * x'maxr$$

6.1.2 RECONSTRUCCIÓN DE LA SECUENCIA RPE.

La secuencia resultante es rellenada con ceros con un factor de tres y de acuerdo con las posiciones que indica el valor *Mc* decodificado, obteniendo así la excitación residual decodificada.

$$er'(kj+Mc+3*i) = x'mr(i) \qquad i = 0 \ldots 12$$

$$er'(kj+Mc+3*i) = 0 \qquad \text{Para el resto.}$$

6.2 SECCIÓN LTP.

La excitación residual reconstruida *er'(kj +k)* se aplica al filtro LTP para producir la correspondiente señal reconstruida *dr'(kj +k).* Con ello se añade a la señal su pitch. Para este proceso, se utilizan los parámetros LTP que son decodificados y utilizados en el filtro, tal como se describe en las siguientes secciones.

6.2.1 DECODIFICACIÓN DE LOS RETRASOS LTP.

Se decodifican los siete bits y se recupera el valor del retraso LTP.

Nr' = Ncr

Además se chequea que Ncr esté entra 40 y 120. Para ello se efectúa un pequeño algoritmo:

Nr = Ncr;
IF (Ncr < 40) THEN Nr = nrp;
IF (Ncr>120) THEN Nr = nrp;
nrp = Nr;

6.2.2 DECODIFICACIÓN DE LAS GANANCIAS LTP.

Las ganancias LTP son decodificadas de acuerdo con la siguiente ecuación:

br' = QLB (*bcr*)

donde QLB (i) representa el nivel de cuantización (véase tabla 3.4) y *br'* la ganancia decodificada

6.2.3 FILTRADO DE SÍNTESIS LTP.

La excitación residual proveniente de la sección RPE *er'(kj + k)* era la resultante de eliminar de la señal original su envolvente del espectro y su pitch. Con ella construimos la señal reconstruida y, a la vez, actualizamos las 120 muestras anteriores reconstruidas necesarias para la decodificación LTP de la siguiente subtrama recibida. La ecuación que sigue es la siguiente:

$$dr''(kj + k) = br' * dr'(kj + k - Nr') \qquad j = 0 \ldots 3 \quad k = 0 \ldots 39 \quad kj = ko + j * 40$$

$$dr'(kj + k) = er'(kj + k) + dr''(kj + k) \quad j = 0 \ldots 3 \quad k = 0 \ldots 39 \quad kj = ko + j * 40$$

6.3 SECCIÓN STP.

En esta etapa se añade a la señal su envolvente del espectro calculada a partir de los *LAR* recibidos, que son decodificados y procesados tal como se describe a continuación.

6.3.1 DECODIFICACIÓN DE LOS LAR.

En este bloque, los *LARcr* son decodificados según la siguiente ecuación:

$$LARr''(i) = [\, LARcr(i) - B(i) / A(i) \,]$$

Donde los valores de A(i) y B(i) están en la tabla 3.2.

6.3.2 INTERPOLACIÓN DE LOS LAR.

Para evitar transitorios desagradables que pueden ocurrir cuando los coeficientes del filtro son modificados abruptamente, se interpola los LAR decodificados de la trama actual con los de la trama anterior. Esta interpolación es idéntica a la que se hace en el codificador. (véase sección 3.3.2).

6.3.3 TRANSFORMACIÓN DE LOS LAR EN COEFICIENTES DE REFLEXIÓN.

Los coeficientes de reflexión son finalmente determinados usando la ecuación de transformación de los LAR (véase 3.3.3).

6.3.4 FILTRO QUE RESTAURA LA ENVOLVENTE.

La envolvente del espectro la tenemos codificada con los *LARcr*. Con este filtro pretendemos añadirla a la señal de entrada para obtener una señal de salida con envolvente del espectro. El esquema de dicho filtro lo podemos observar en la figura 6.2. Las ecuaciones del filtro son las siguientes:

$$sr(0)(k) = dr'(k)$$
$$sr(i)(k) = sr(i-1)(k) - rr'(9-i) * v(8-i)(k-1) \quad i = 1 \ldots 8$$
$$v(9-i)(k) = v(8-i)(k) + rr'(9-i) * sr(i)(k) \quad i = 1 \ldots 8$$
$$sr'(k) = sr(8)(k)$$
$$v(0)(k) = sr(8)(k)$$

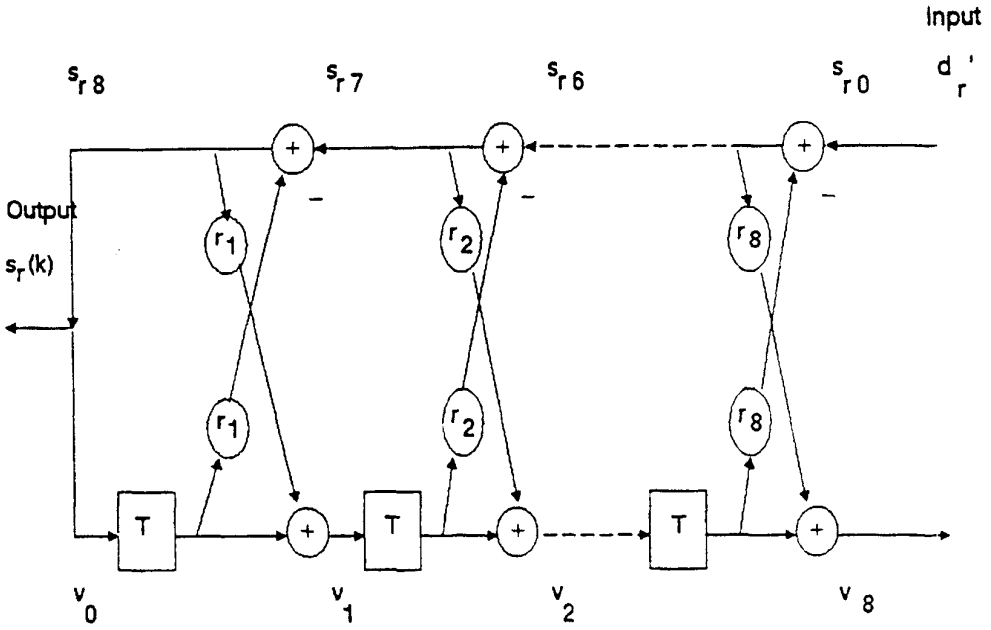


Fig 6.2 Esquema del filtro STP de reconstrucción.

6.4 POSTPROCESADO

La salida del filtro STP, *sr'(k)* es aplicada a un filtro IIR de deénfasis, obteniendo así la señal de salida final, *sro(k)* siguiendo las siguientes ecuaciones:

$$sro(k) = sr'(k) + beta * sro(k-1); \qquad beta = 28180 * 2E^{-15}$$

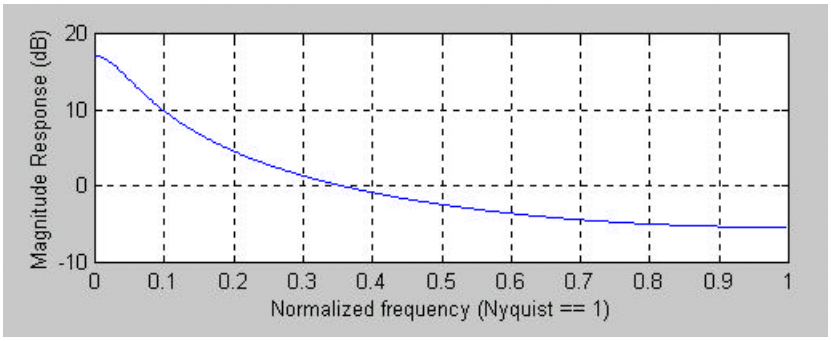


Fig 6.3 Filtro de deénfasis.

Capítulo 7 .

RESULTADOS.

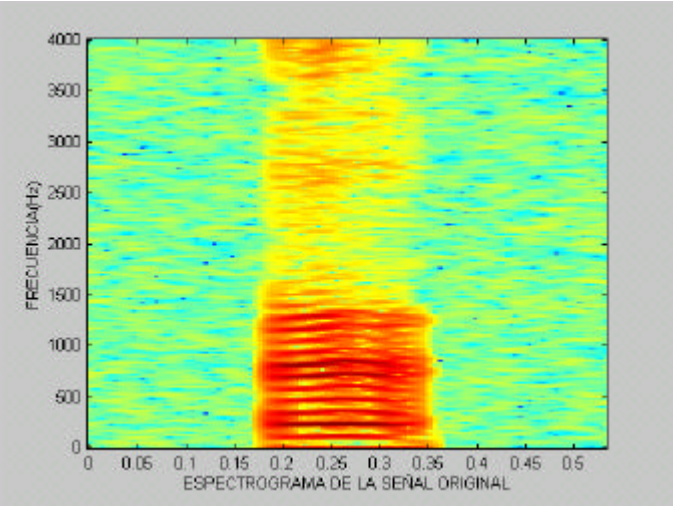
7.1 INTRODUCCIÓN.

En este capítulo realizaremos una serie de simulaciones sobre distintas tramas de voz con el fin de analizar el codificador en sus distintas partes. Veremos representaciones tanto en tiempo como en frecuencia de las señales de voz, así como de las componentes extraídas de éstas. Dividiremos cada simulación en tres apartados:

- La primera, dedicada exclusivamente al análisis y funcionamiento tanto del codificador como del decodificador funcionando conjuntamente.
- Una segunda parte, en la que activaremos el Detector de Actividad Vocal, para así analizar el funcionamiento del mismo.
- Una última en la que adicionalmente, activaremos la Codificación de Canal. Analizaremos principalmente cómo afecta la tasa de error del canal, la BER, a la trama reconstruida.

7.2 ANÁLISIS DE UNA VOCAL.

En esta primera simulación, la trama de voz a analizar contiene una vocal, en concreto la vocal *'a'*, que es la de mayor energía. En la siguiente gráfica podemos observar el espectrograma de esta secuencia de voz:



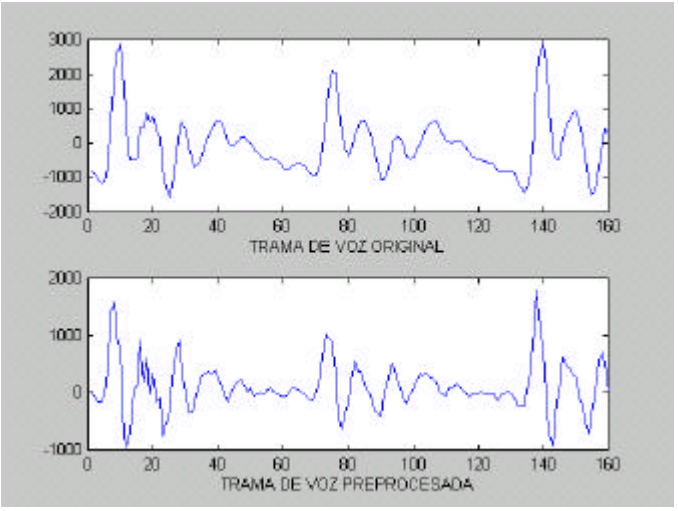
El eje vertical es la frecuencia en Hz, y el eje horizontal es el tiempo en segundos. Vemos que la parte de voz de esta señal está comprendida entre aproximadamente 0.18 seg y 0.35 seg, y que la misma está prácticamente contenida entre 0 y 1400 Hz.

7.2.1 CODIFICADOR Y DECODIFICADOR.

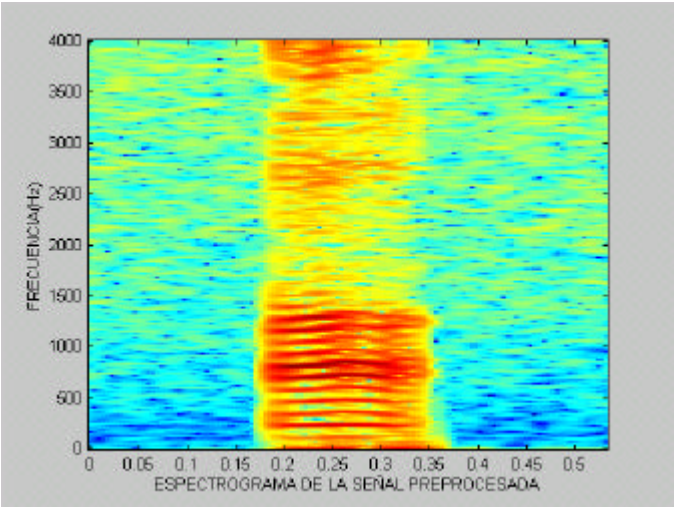
En este primer apartado, analizaremos el funcionamiento del codificador y del decodificador GSM actuando conjuntamente. Para ello, insertaremos en el codificador la trama de voz original representada anteriormente y veremos representaciones de la señal en cada una de las secciones de las que consta el codificador.

Sección de Preprocesado.

En esta primera sección, es eliminada la componente de continua de la señal de entrada, y a continuación se le aplica un filtro de preénfasis. Este filtro enfatiza un poco las altas frecuencias de la señal con el fin de conseguir mayor resolución en ellas. En el decodificador, existe el filtro inverso, que es el filtro de deénfasis, que anula el efecto provocado por el de preénfasis. A continuación veamos una trama de voz original a la entrada del codificador y la misma trama a la salida de la sección de preprocesado.



Observamos cómo se aumenta un poco la ganancia en alta frecuencia, se observa muy bien sobre todo en los picos de la señal, que son más agudos. Veamos ahora la representación en frecuencia de esta señal preprocesada:

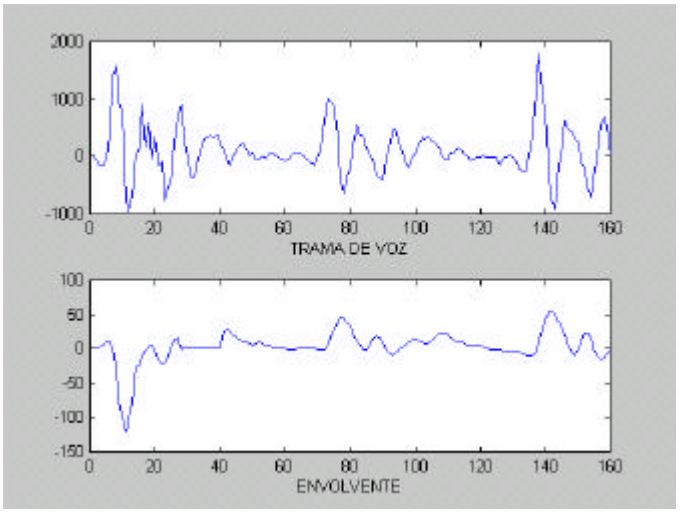


Vemos aquí cómo ha disminuido un poco la ganancia en baja frecuencia y aumentado en alta para conseguir esa resolución.

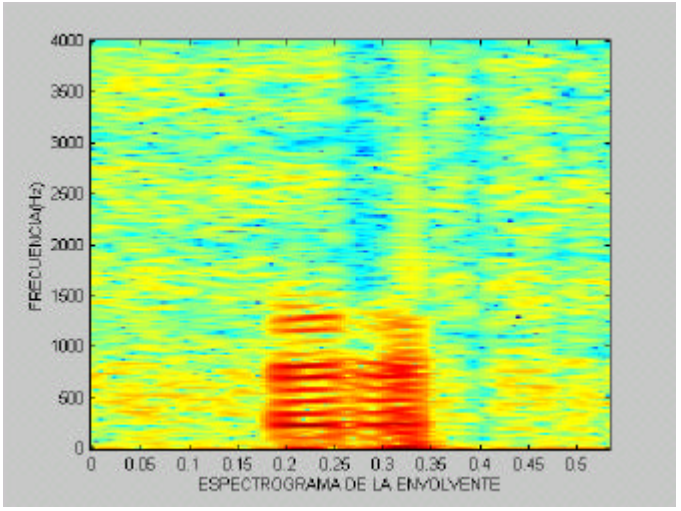
Sección STP.

El objetivo de esta sección es eliminar la envolvente del espectro de la señal original, parametrizarla, codificarla y enviarla al decodificador en los LAR. Para ello, es preciso estimar la envolvente del espectro, lo cual se hace mediante la obtención de los coeficientes de reflexión a partir de los términos de autocorrelación (Recursión de Schur), y una vez obtenidos, un filtro que trabaja con la señal y con los coeficientes de reflexión, la elimina.

En la siguiente gráfica podemos observar una trama de voz original junto con su envolvente del espectro estimada.

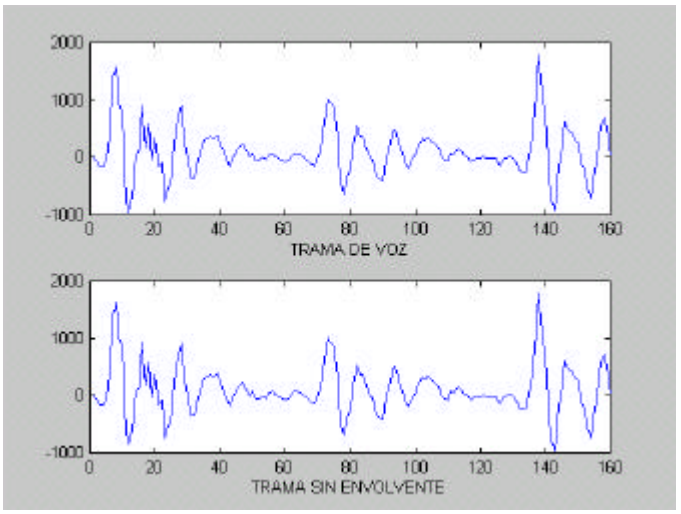


A continuación veamos el espectrograma de dicha envolvente del espectro:

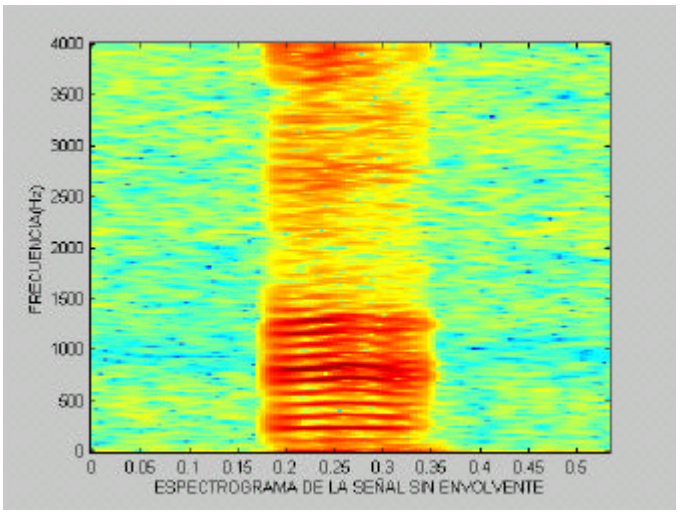


Observamos pues que este algoritmo estima bastante bien la envolvente del espectro. Internamente, en el codificador, los coeficientes de reflexión usados serán transformados a LAR, que será lo que finalmente se envíe al decodificador, en el cual se reconstruirá dicha envolvente del espectro y se le sumará a la señal para reconstruirla.

Veamos a continuación, cómo queda la señal tras eliminarse la envolvente del espectro, señal que llegará a posteriores secciones donde se les extraerá además, el pitch y la excitación residual.



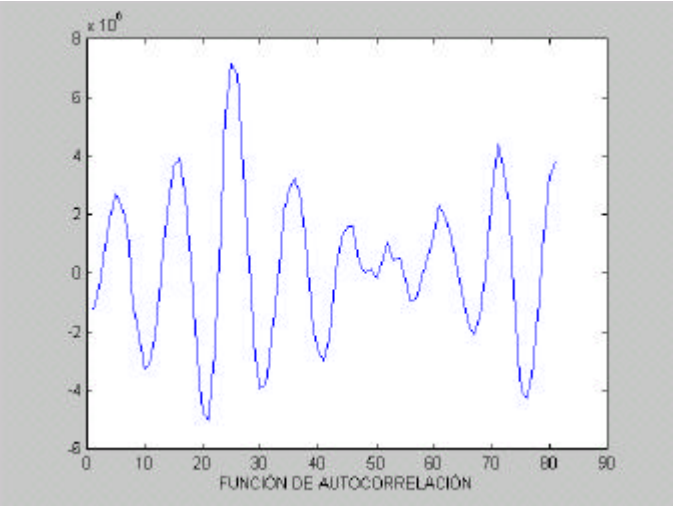
Vemos aquí que tras eliminarse la envolvente del espectro, prácticamente la señal permanece invariable. Esto es debido a que la envolvente del espectro es de muy pequeña amplitud. Para observarlo mejor, veamos el espectrograma de la señal sin la envolvente del espectro:



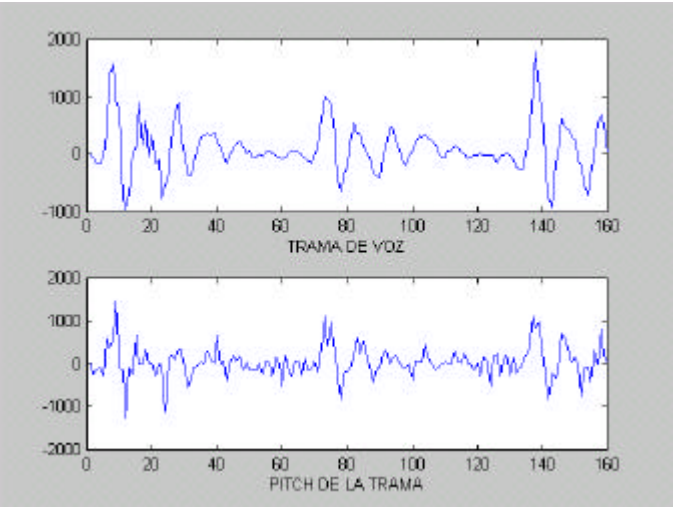
Vemos que la señal parece la misma que la original. Sin embargo, esta diferencia es notablemente superior cuando escuchamos la envolvente del espectro y la señal sin envolvente del espectro. La envolvente del espectro es la parte de la señal que contiene mayor información. Por ello, si la escuchamos es perfectamente inteligible. Sin embargo, al escuchar la señal sin envolvente del espectro suena como una señal "hueca", sin energía. Por ello, aunque gráficamente no sea posible apreciar bien la diferencia, sonoramente esta diferencia es muy apreciable.

Sección LTP.

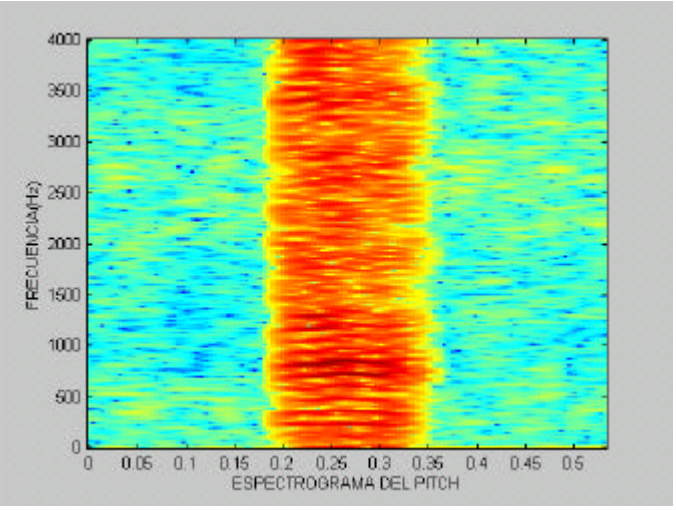
Esta sección es la encargada de estimar el pitch de la señal de entrada. Para ello, en primer lugar se calcula la función de autocorrelación de la señal de entrada, que es la señal preprocesada sin envolvente del espectro. Veamos la función de autocorrelación para nuestra trama de entrada.



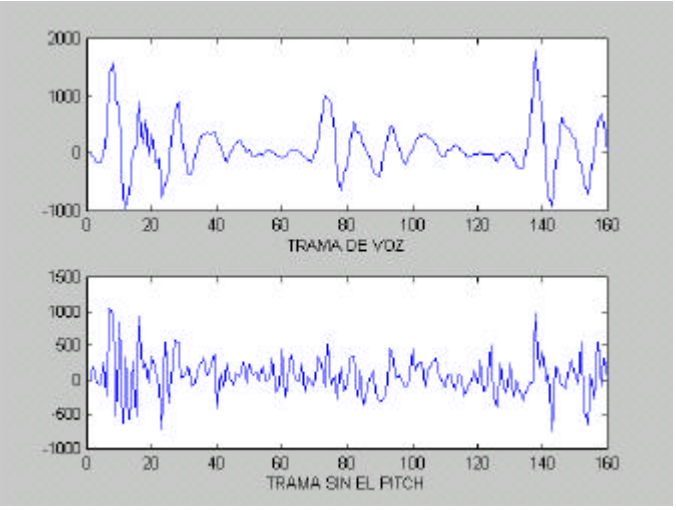
La función de autocorrelación es útil para calcular la frecuencia fundamental o frecuencia de pitch de la señal. Para ello medimos la distancia entre dos máximos (que no sean locales, sino globales). La distancia es de 47 muestras. Como 160 muestras corresponden a 20 ms, 47 muestras corresponden a 5.875 ms. La frecuencia de pitch, por tanto será de 170 Hz. Para la misma trama de voz correspondiente a una mujer, esta frecuencia de pitch sería más alta. El pitch se estima calculando la función de autocorrelación y calculando el retraso para el cual se produce el máximo. Después se calcula la ganancia correspondiente a este retraso de la función de correlación cruzada entre la señal de entrada y las 120 muestras anteriores reconstruidas. El pitch no será más que estas 120 muestras anteriores reconstruidas retrasadas por el valor anterior y multiplicadas por la ganancia anteriormente calculada. Veamos el pitch de la señal:



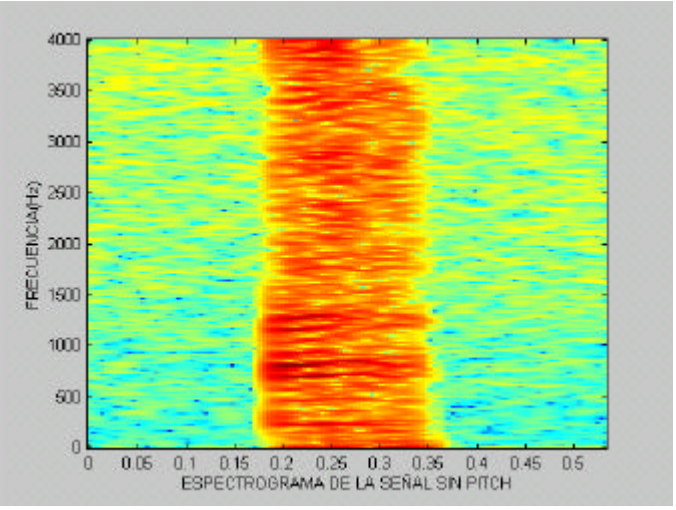
Veámosla ahora en frecuencia:



A continuación veremos la señal que queda tras eliminar la componente de pitch de la señal de entrada:

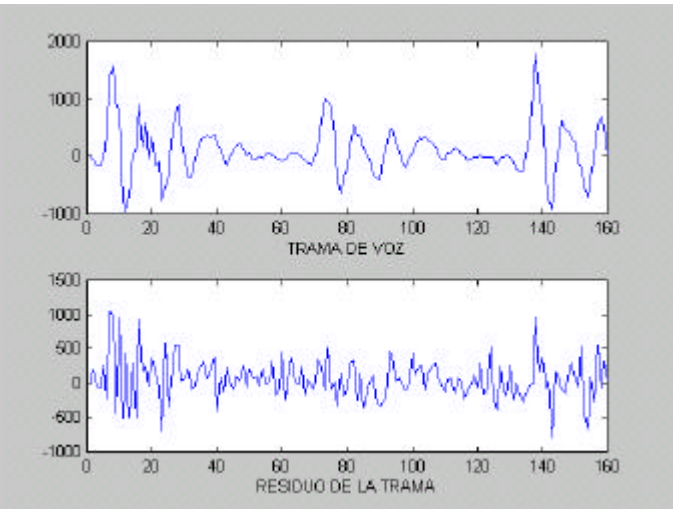


Vemos que queda una señal muy ruidosa. Esto es debido a que al eliminar el pitch de la señal original, nos queda la envolvente del espectro y la excitación residual, y la señal ruidosa que vemos ahí superpuesta a la envolvente del espectro sería la excitación residual que más tarde veremos. Veamos a continuación la señal sin pitch en frecuencia:

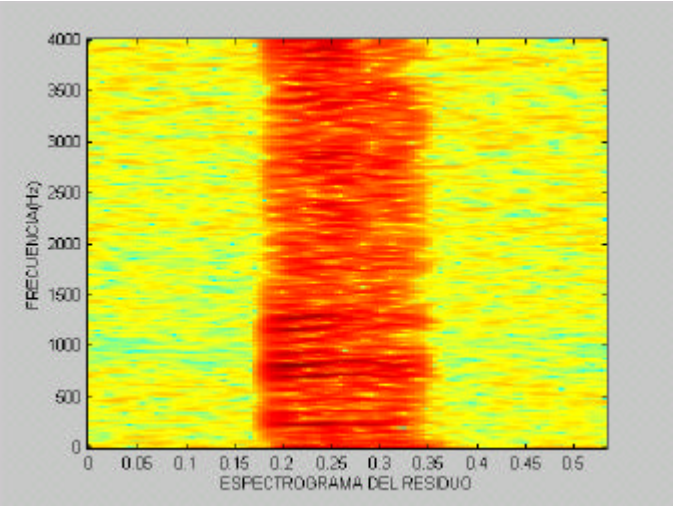


Sección RPE.

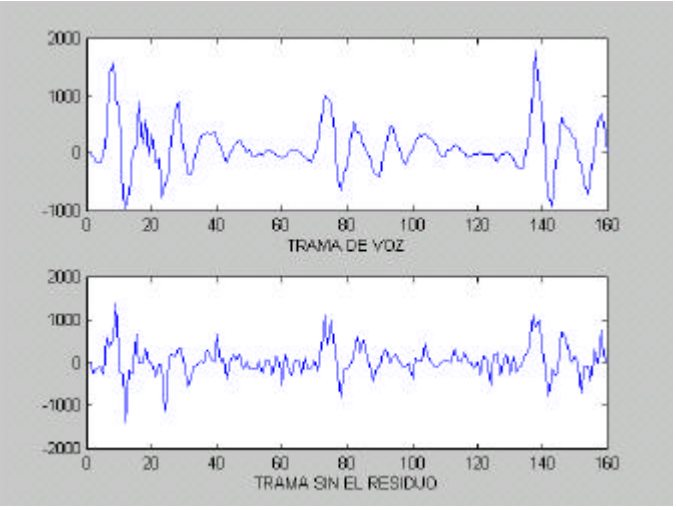
En esta última sección, trataremos de estimar y eliminar la excitación residual. Para ello primero pasamos la señal por un filtro llamado filtro de Weight, submuestreamos en tres subsecuencias y nos quedamos con la de mayor energía. Esta subsecuencia seleccionada se codifica APCM y se envía al decodificador. Veamos la excitación residual de la señal de entrada:



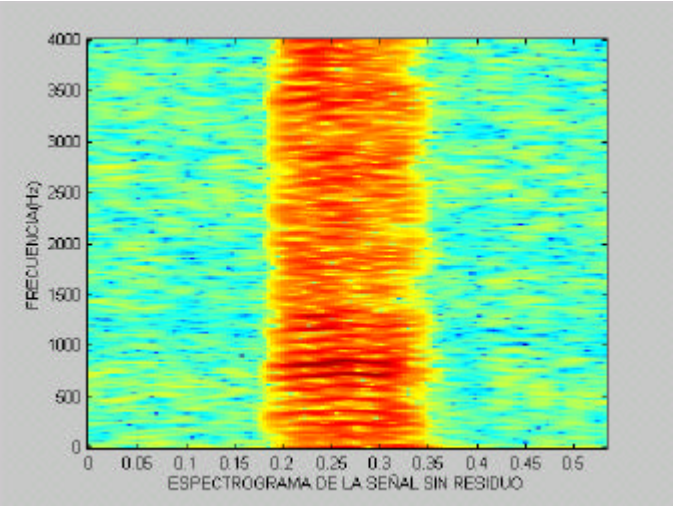
Vemos que es una señal muy ruidosa tal como cabría esperar de la anterior sección. Esta excitación residual es la base a partir de la cual se forma la señal, añadiendo posteriormente el pitch y la envolvente del espectro. Veámosla ahora en el dominio de la frecuencia.



Veamos a continuación la señal sin el residuo:

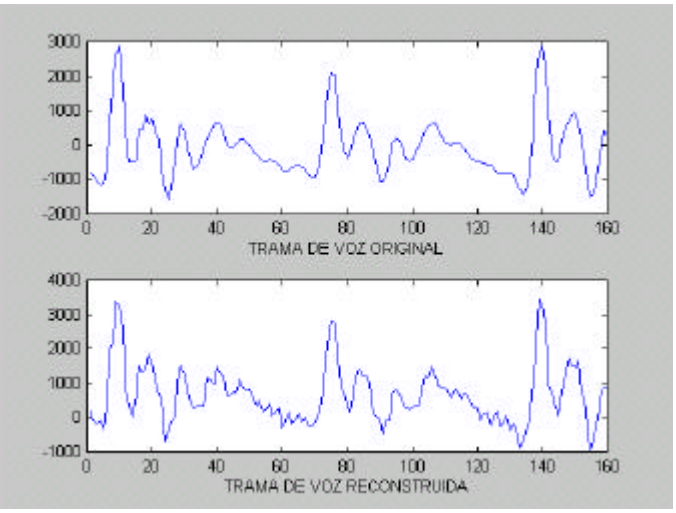


Si observamos en la anterior sección, vemos que la gráfica es aproximadamente la misma que la del pitch. Esto es porque al eliminar el residuo de la señal original, sólo nos queda el pitch más la pequeña envolvente del espectro, que es la que la diferencia. Veamos por último el espectrograma de la señal sin el residuo:

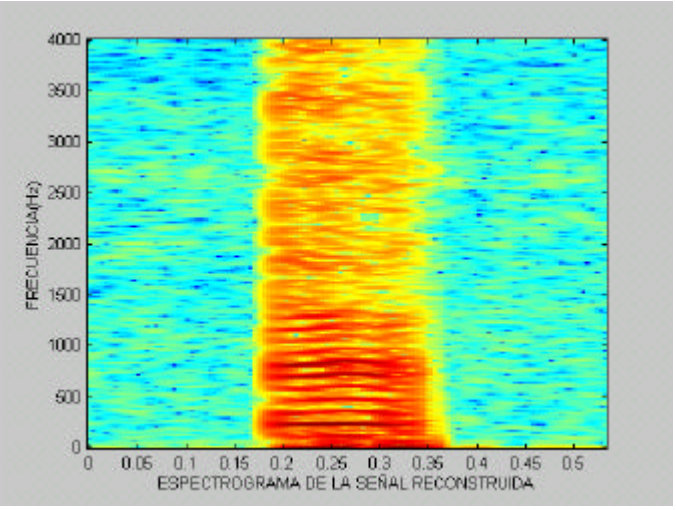


Decodificador.

El decodificador GSM tiene un funcionamiento muy fácil. Recibe los distintos parámetros codificados, tanto de la envolvente del espectro, como del pitch como del residuo. Los decodifica y reconstruye. Primero reconstruye la excitación residual y posteriormente le añade el pitch y la envolvente del espectro. Por último le añade un deénfasis para compensar el filtro de preénfasis aplicado en la sección de preprocesado. Así, lo que queda es la señal reconstruida. Veamos a continuación una comparación entre la señal original y la señal reconstruida en el decodificador:



Vemos que es prácticamente idéntica a la señal original. Tan sólo queda un pequeño ruido superpuesto que será eliminados en posteriores tratamientos antes de su salida por el altavoz. Veamos dicha señal en frecuencia:

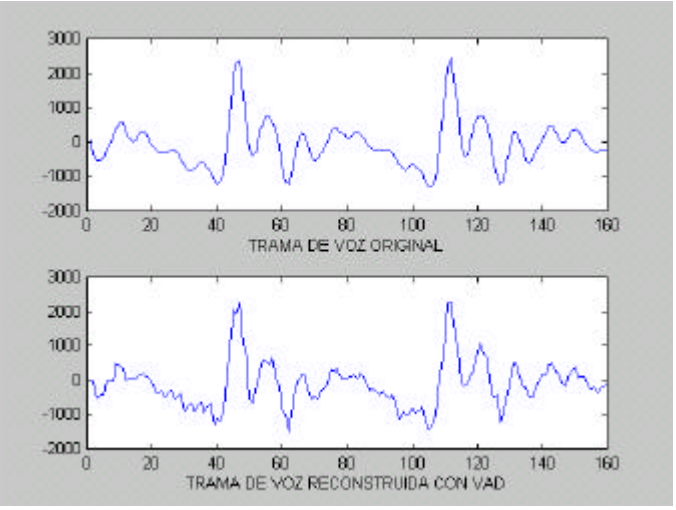


7.2.2 DETECTOR DE ACTIVIDAD VOCAL.

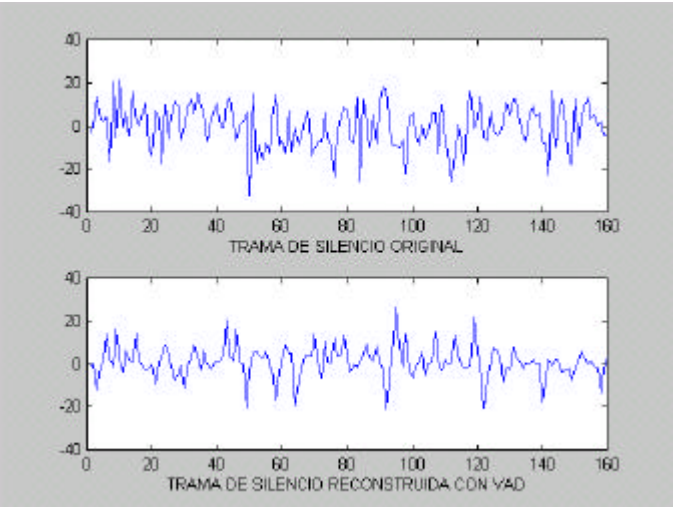
En este apartado, activaremos en la simulación el Detector de Actividad Vocal, con el fin de analizarlo. Además también hay que activar el constructor de tramas de silencio. El VAD funciona de la siguiente forma: coge una trama completa y calcula un parámetro relacionado con la energía, el cual compara con un umbral dinámico que almacena y cambia de una trama a la siguiente. Si es mayor que el umbral considera que dicha trama es de voz, considerándola trama de silencio en caso contrario. Además también trabaja con el pitch de la señal, si detecta pitch en la señal aunque su energía no supere el umbral no es considerada trama de silencio. Esto se debe a que las colas de algunas consonantes ruidosas de pequeña amplitud, como por ejemplo la 's', pueden ser confundidas con periodos de ruido. Además el VAD aplica una cierta histéresis, de manera que una trama no es considerada trama de silencio hasta que no haya cuatro valores de decisión consecutivos a cero. El VAD guarda algunos parámetros de las tramas anteriores para su algoritmo de adaptación.

Si una trama es considerada trama de silencio o trama SID, entonces se procede a sustituir los parámetros *LARc* y *xmaxc* por una media de los de tramas anteriores, el resto de los parámetros se ahorrarían en la transmisión. Ya en el decodificador, el resto de los parámetros *Nc*, *bc*, *Mc* y *xmc* se sustituyen por valores fijos o aleatorios.

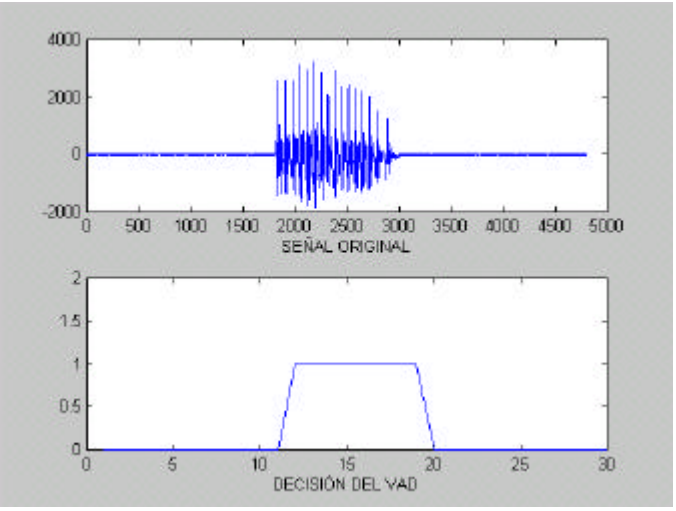
Veamos en primer lugar una trama de voz reconstruida con el VAD funcionando. Esto no es más que una trama de voz reconstruida normalmente, ya que en este caso, la decisión del VAD es uno, y no actuaría en este caso el constructor de tramas de silencio.



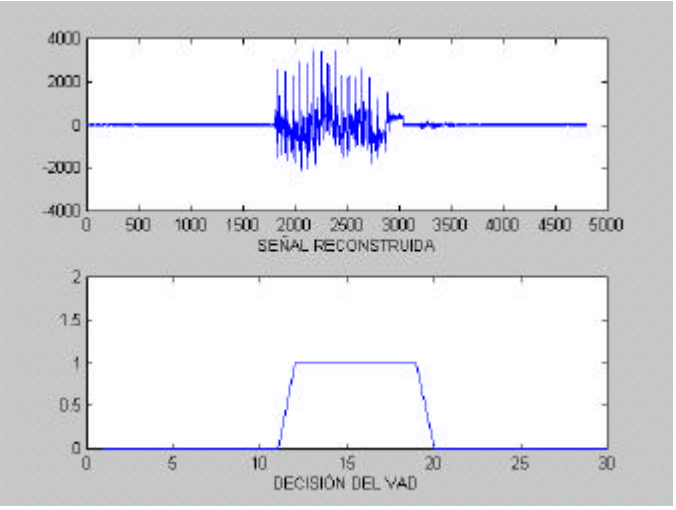
Y a continuación veamos una trama de silencio reconstruida.



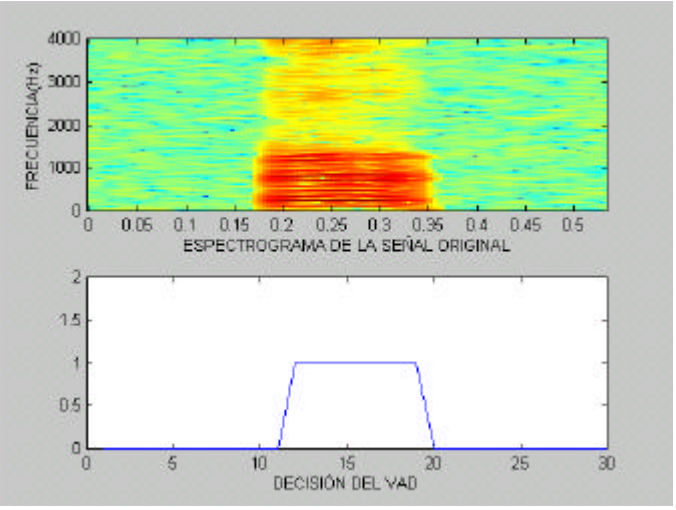
En este caso, la decisión del VAD es cero, considerándola trama de silencio. Los parámetros de la trama son sustituidos por otros, reconstruyéndose en el decodificador la trama que podemos observar. A continuación vamos a representar la señal original en su totalidad junto con los valores de decisión del VAD:



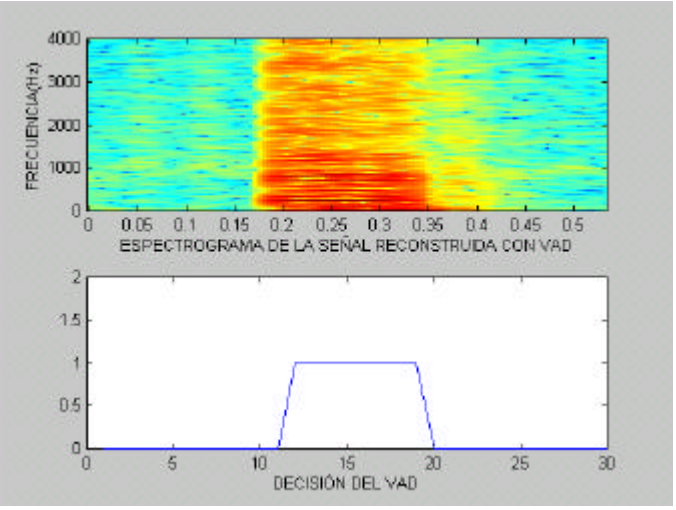
Y a continuación veamos la señal reconstruida en su totalidad junto con la decisión del VAD.



Veamos a continuación el espectrograma de la señal original junto con los valores de decisión del VAD.



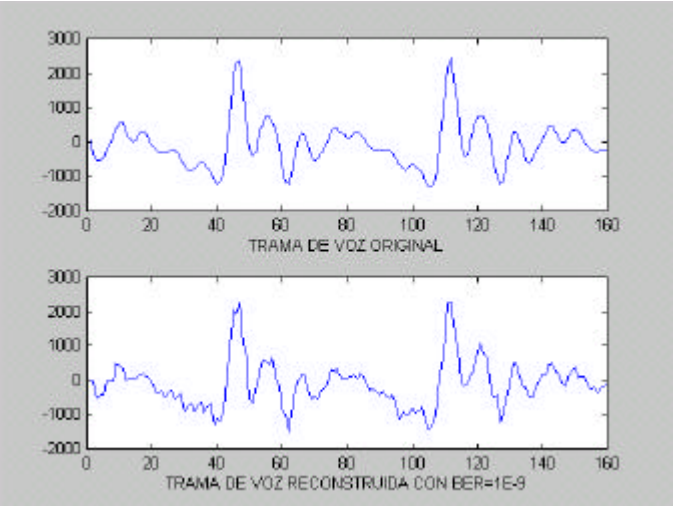
Y, por último, el espectrograma de la señal reconstruida junto con los valores de decisión del VAD.



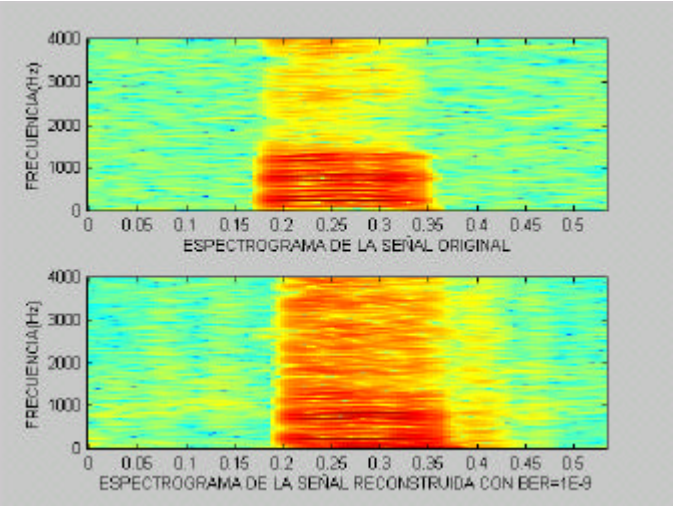
7.2.3 CODIFICACIÓN DE CANAL.

Este último apartado lo dedicaremos a analizar cómo afecta la tasa de error del canal a la señal reconstruida. El codificador GSM dispone de una serie de métodos para contrarrestar este efecto. Para ello añade información redundante a los bits transmitidos además de asignar un peso a éstos, según el cual se protegerán más ciertos bits más importantes frente a otros que lo son menos.

Empezaremos a ver cómo afecta una BER de 10^{-9} y veamos la señal reconstruida en el decodificador junto con la original.

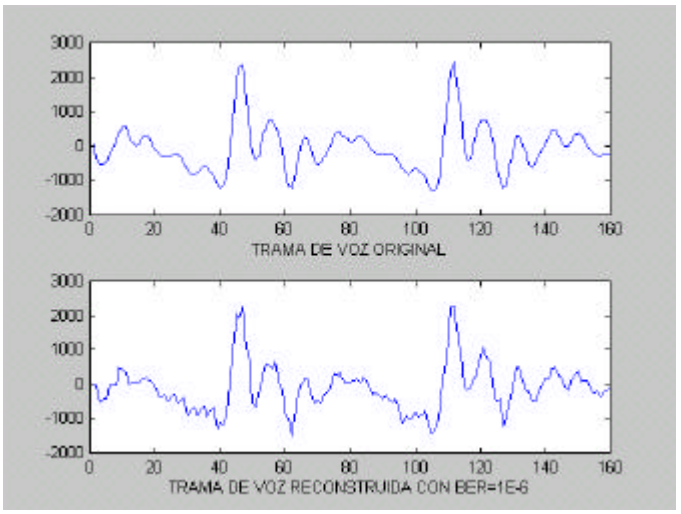


Vemos que la señal es prácticamente la misma que la original, por lo que podemos considerar que esta tasa de error es admisible. Veamos a continuación la señal en frecuencia para poder observar mejor este efecto.

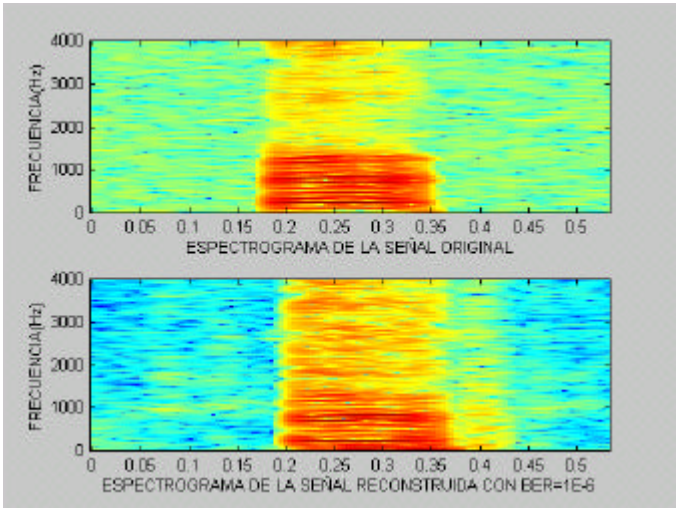


Aparecen unas componentes en alta frecuencia, pero ya aparecían sin error alguno. Se debe sobre todo al funcionamiento del codificador y decodificador y comentábamos que eran eliminadas con un filtrado posterior.

Analicemos a continuación una BER de 10^{-6}.

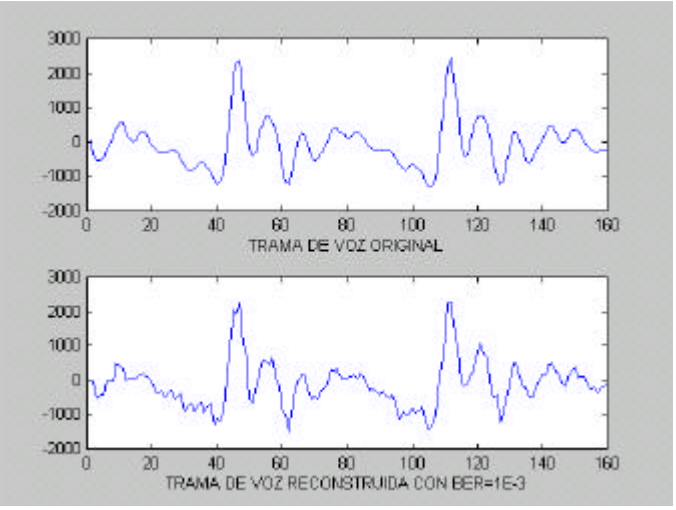


Y veámosla ahora en el dominio de la frecuencia.

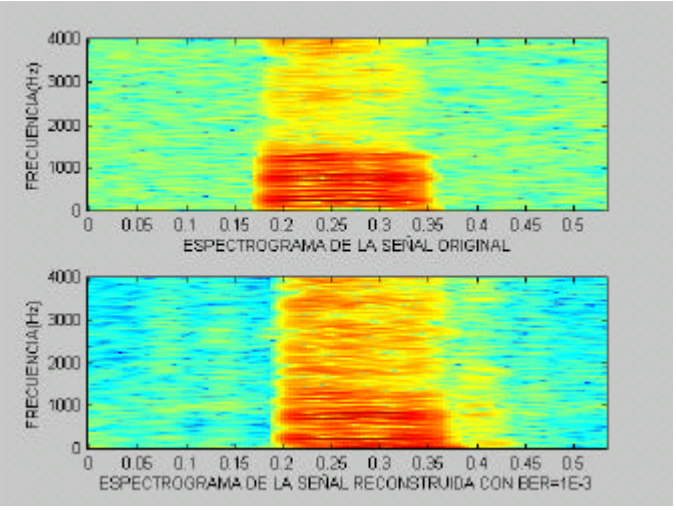


Observamos que sigue siendo una tasa de error admisible puesto que la señal reconstruida conserva sus propiedades frente a la original. Este efecto es mucho más apreciable si escuchamos la señal, ya que estas comparaciones se suelen realizar así, mostrando las señales a distintas personas y asignándole cada una de ellas una nota media de opinión o MOS.

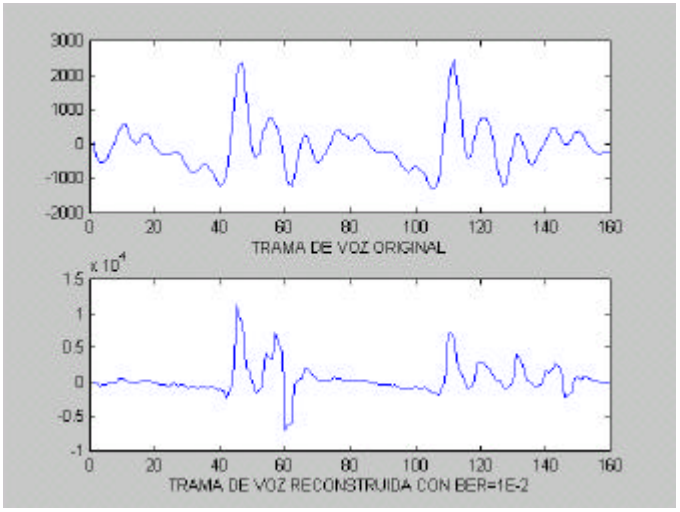
Vamos ahora a incrementar la tasa de error hasta 10^{-3}, que ya empieza a ser una tasa de error importante. Veamos la señal en el dominio del tiempo.



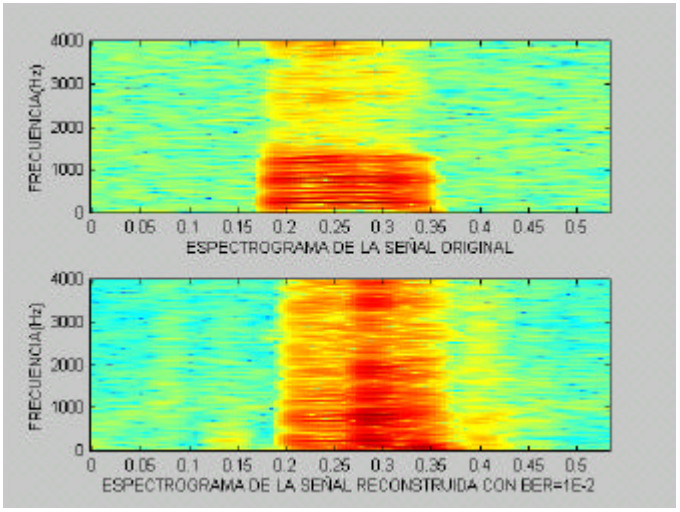
Y veámosla ahora en el dominio de la frecuencia.



Vemos que incluso estas tasa de error no es lo suficiente como para poder afirmar de forma rotunda que la señal ha sufrido una degradación tal que se considera inadecuada. Y esta afirmación es más rotunda si escucháramos la señal. Además, según la Recomendación, esta tasa de error es admisible, pues estamos en la máxima tasa de error permitida. Tasa de errores mayores ya no lo serían. Analicemos a continuación una tasa de error de 10^{-2}.

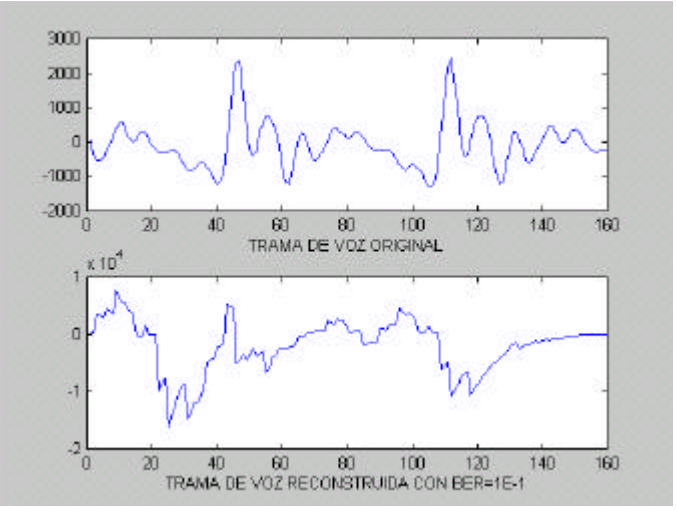


Y veámosla también en frecuencia.

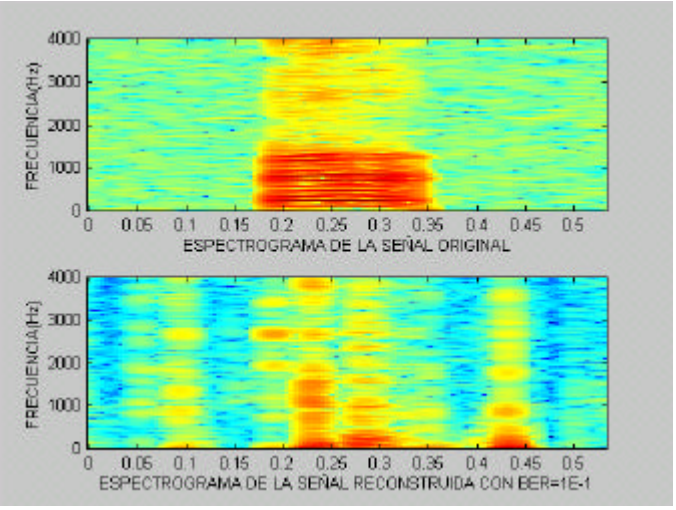


Vemos ya aquí que aunque el espectro de energía sigue concentrado en el mismo dominio temporal, la forma de onda de la señal resulta ya errónea. Aunque sigue levemente los picos de la señal, no sigue correctamente el resto de la forma de onda.

Veamos para finalizar cómo afectaría una tasa de error de 10^{-1}.



Y también en frecuencia.



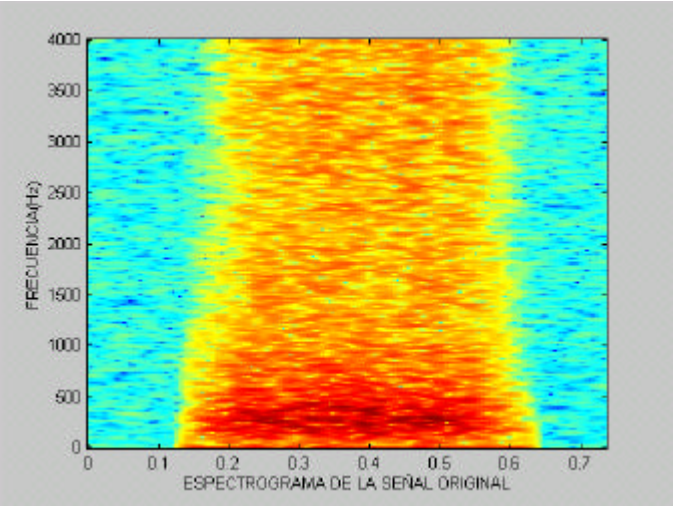
Vemos aquí que ni incluso en el dominio de la frecuencia la señal se parece lo más mínimo a la original.

Por tanto, podemos concluir que la máxima tasa de error admisible para una comunicación GSM es de 10^{-3}.

7.3 ANÁLISIS DE UNA CONSONANTE.

En este apartado vamos a realizar la simulación de una consonante, en concreto de la consonante *'s'*. Ello es debido a que las vocales son fonemas de gran energía, muy impulsivas y limpias, pero algunas consonantes tiene largas colas que se pueden confundir con ruido, sobre todo ésta en concreto. Principalmente veremos el funcionamiento del Detector de Actividad Vocal en esta situación, aunque veremos todos los apartados. En esta segunda simulación omitiremos algunas gráficas, por haberse visto ya, centrándonos en las más importantes.

En primer lugar, veremos el espectrograma de la señal original.

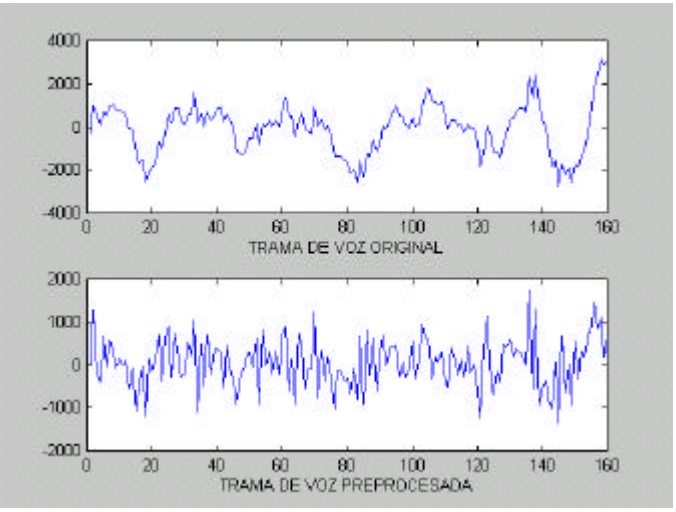


Vemos que esta señal es más ruidosas, tiene más componentes en alta frecuencia que las vocales. La señal está localizada temporalmente entre 0.15 seg y 0.65 seg, y frecuencialmente entre 0 y 1000 Hz aunque como hemos dicho, las componentes superiores a 1000 Hz no son despreciables.

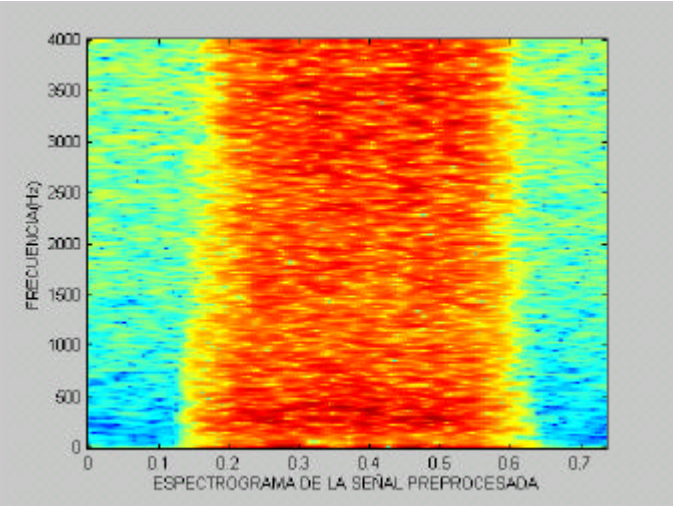
7.3.1 CODIFICADOR Y DECODIFICADOR.

Sección de Preprocesado.

Veamos una trama de voz original junto con la misma trama a la salida de la sección de preprocesado.

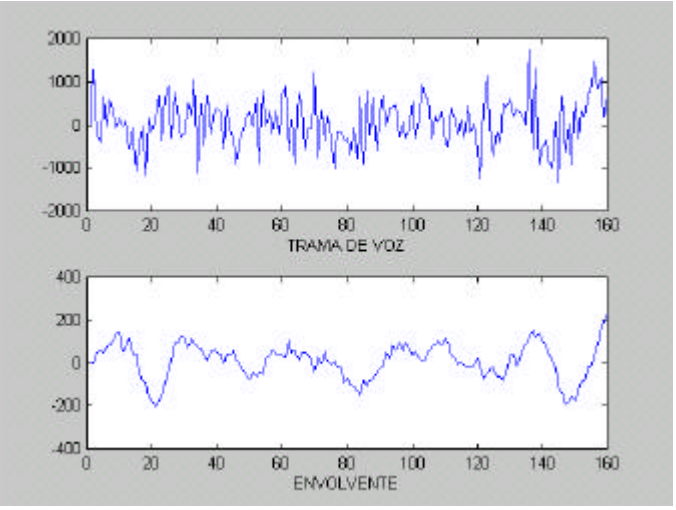


Como podemos observar, el filtro de preénfasis que actúa sobre la alta frecuencia, hace que la señal parezca muy ruidosa, puesto que como hemos dicho, esta tipo de señal tiene más componentes en estas frecuencias. Veamos esta apreciación en el dominio de la frecuencia, donde se puede ver más claro.

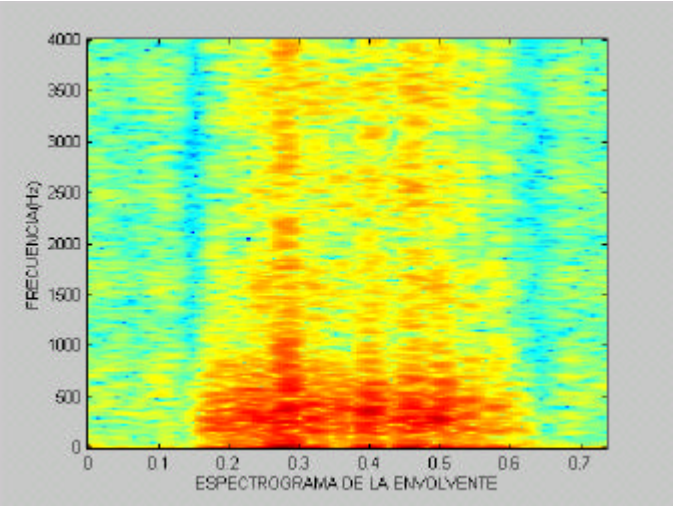


Sección STP.

A continuación veamos la envolvente del espectro estimada a partir de los coeficientes de reflexión.



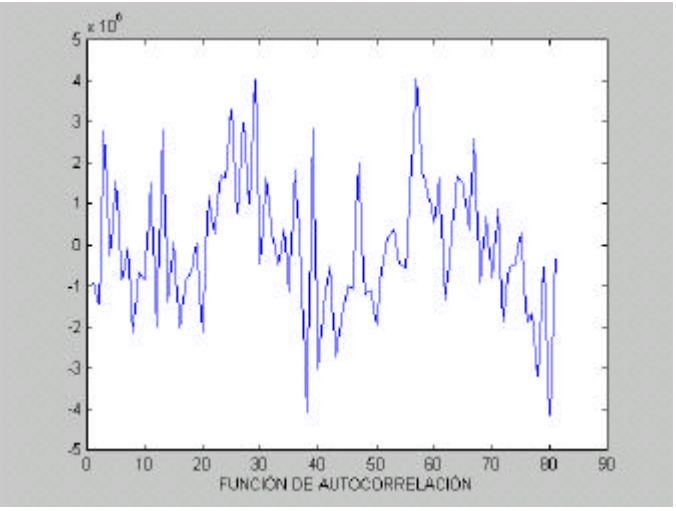
Vemos aquí que la estima bastante bien. Veámosla ahora en el dominio de la frecuencia.



Observamos, por tanto, que la envolvente del espectro es estimada bastante bien como cabía de esperar.

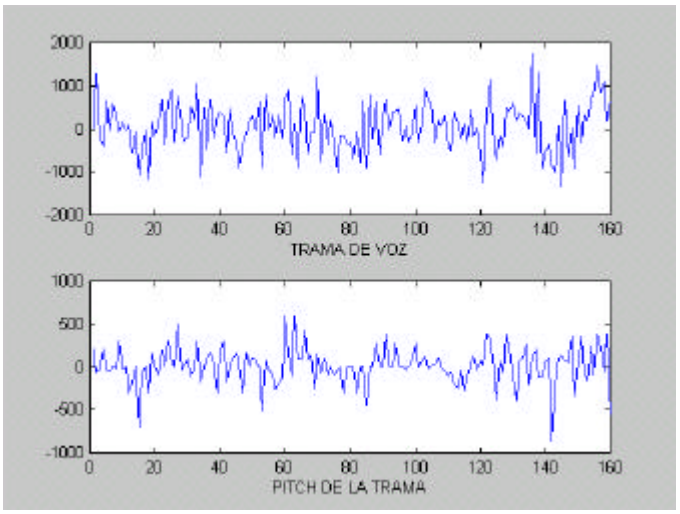
Sección LTP.

Veamos en primer lugar la función de autocorrelación de la señal, calculada en esta sección para estimar el pitch.

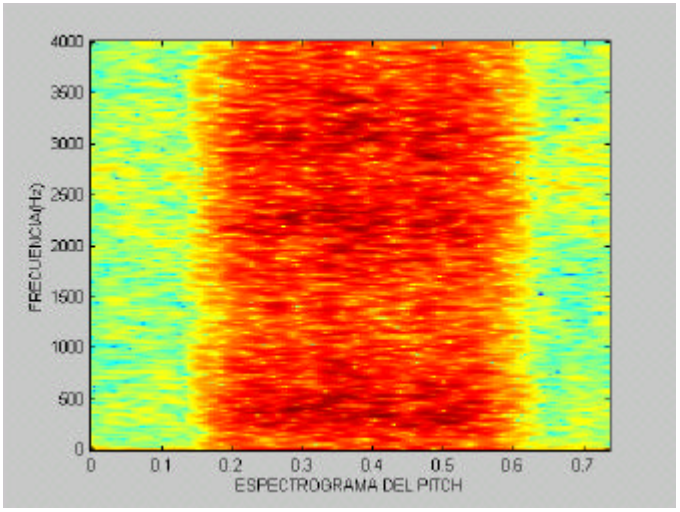


Vemos que la función de autocorrelación es muy ruidosa, por lo que resulta bastante más difícil que en el caso de la vocal afirmar donde están localizados los máximos.

Veamos el pitch estimado para esta señal.



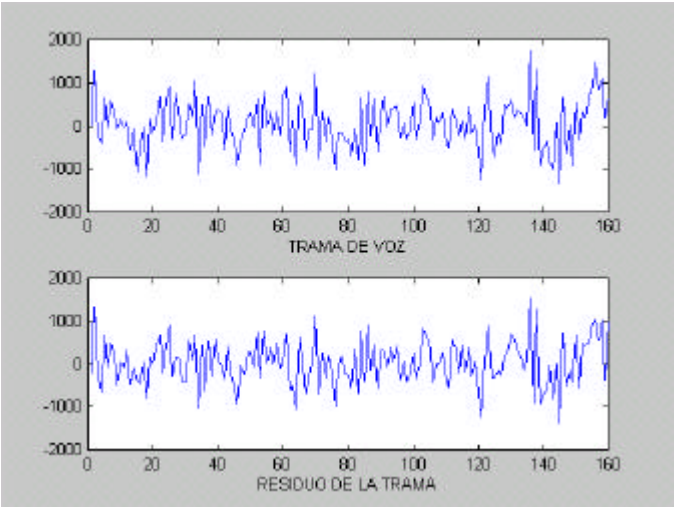
Y veámoslo también en frecuencia.



Sección RPE.

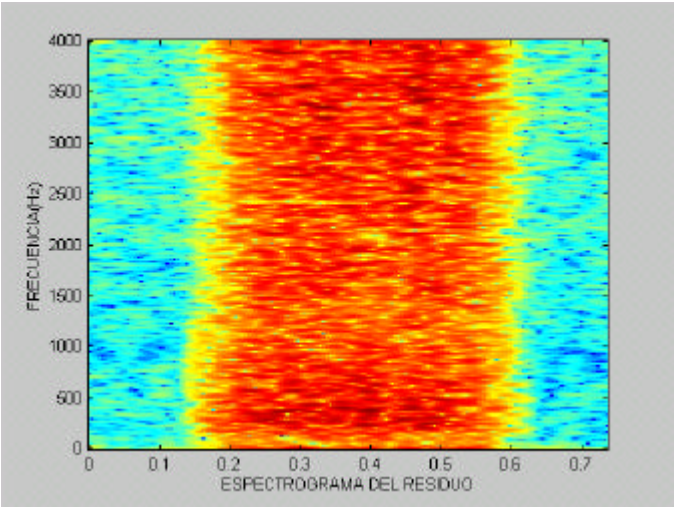
Para finalizar con el análisis del codificador y decodificador, veamos por último el residuo de este tipo de señal, que prácticamente es la base de la misma, a partir del cual se construye.

Veamos una gráfica de una trama de la señal original junto con su residuo correspondiente.



Si observamos en detalle, podemos ver que el residuo de la trama es prácticamente igual que la trama de voz preprocesada, a diferencia del caso anterior de la vocal, en el cual no se parecían en nada. Esto es debido a que, como hemos comentado anteriormente, el residuo es la parte más importante de esta señal.

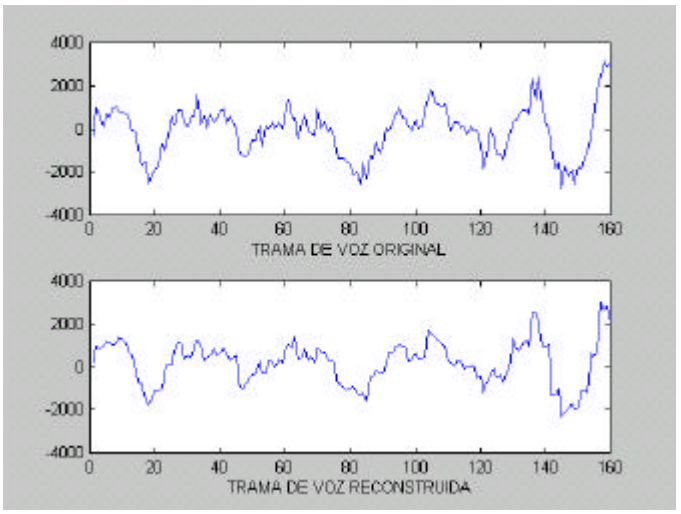
Veámoslo también en frecuencia.



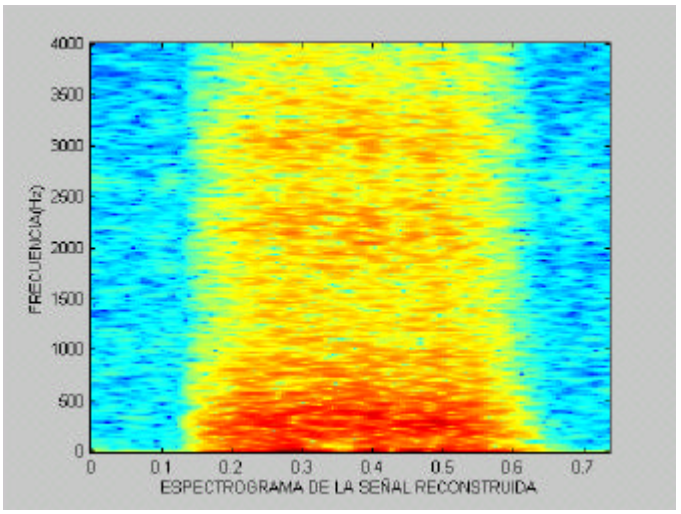
Si observamos el espectrograma de la señal preprocesada, podemos ver que se parecen muchísimo.

Decodificador.

Por último, veamos la señal reconstruida a la salida del decodificador, sumados ya envolvente del espectro pitch y residuo y aplicado el filtro de deénfasis.



Como podemos observar, la señal está reconstruida muy bien, pareciéndose mucho a la original, incluso si observamos los picos de la señal, parece que la señal reconstruida tiene menos componentes de alta frecuencia que la señal original. Veamos esta comparación en el dominio de la frecuencia.



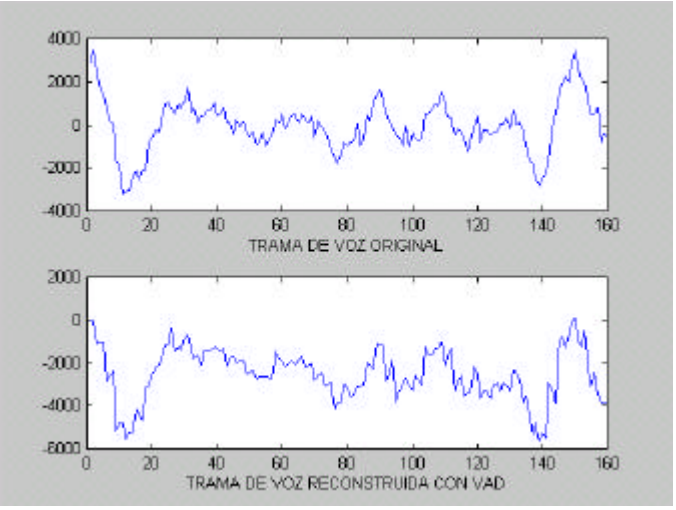
Como hemos comentado, las componentes de alta frecuencia han desaparecido en la señal reconstruida, siendo ésta un poco menos ruidosa. Ocurre todo lo contrario que en el caso de la vocal, en el que eran añadidas algunas componentes de alta frecuencia, siendo la señal resultante un poco más ruidosa.

Vamos a entrar ahora en el apartado más importante de esta señal, que es el Detector de Actividad Vocal.

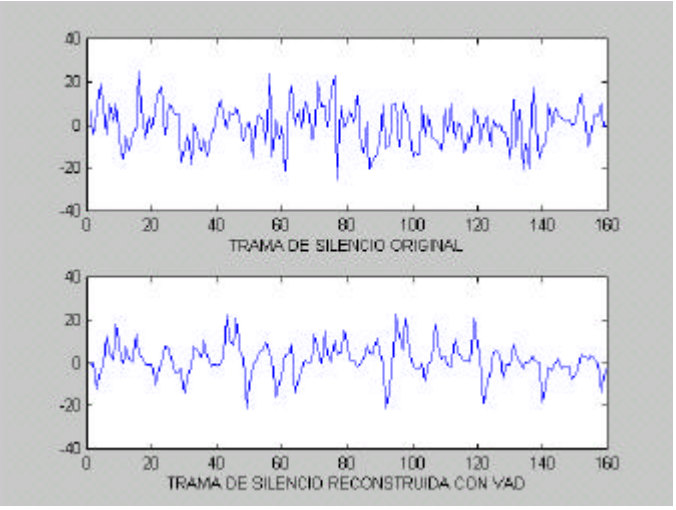
7.3.2 DETECTOR DE ACTIVIDAD VOCAL.

Para este tipo de señal, la consonante ‘s’ ,el detector de actividad vocal puede causar un fallo al final de la señal, debido a que la cola de la ‘s’ puede ser confundido con ruido, debido a que son señales muy ruidosas y, debido a que está situada en la cola, donde ya la señal se va atenuando, es de baja amplitud. Esta situación la contrarresta el VAD extrayendo el pitch de la señal, y viendo así que si en la señal hay presente componente de pitch, entonces aunque su potencia no supere el umbral es considerara como trama de voz y no como trama de silencio.

Veamos en primer lugar una trama de voz original y reconstruida.

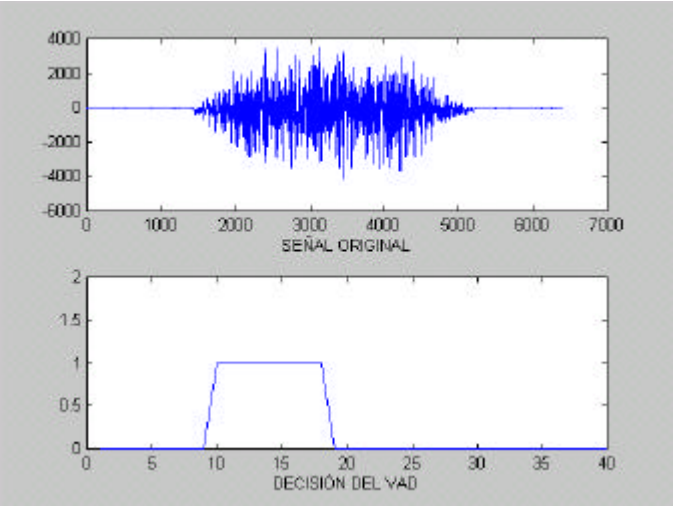


Aquí la decisión del VAD es un uno, por lo que no se modifican los parámetros enviados y se decodifica normalmente. Veamos ahora una trama de silencio original junto con la reconstruida.



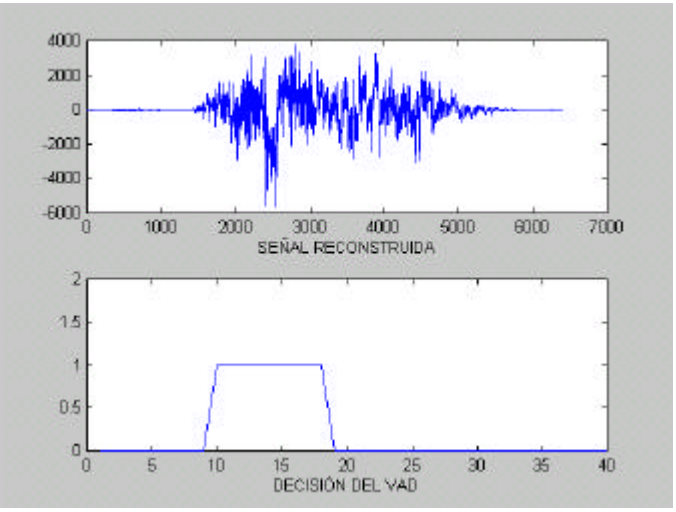
Aquí la decisión del VAD es cero, por lo que se ahorran bits transmitidos, se sustituyen en el decodificador los parámetros y se reconstruye dicha trama de silencio.

Veamos ahora la trama de voz original junto con los valores de decisión del Detector de Actividad Vocal.

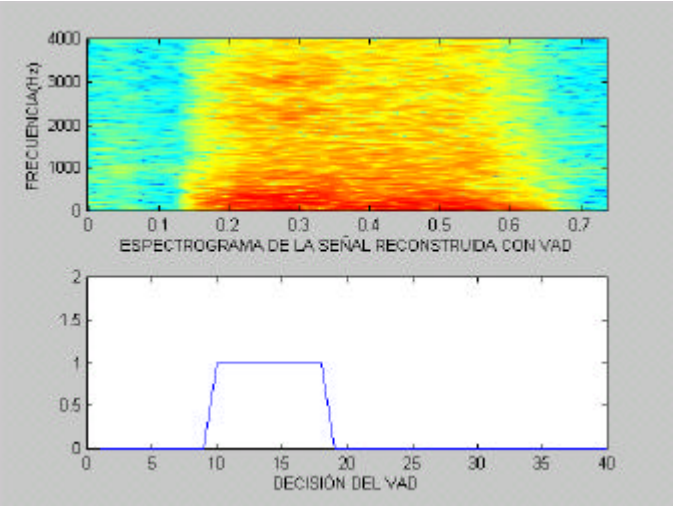


Vemos aquí que en la cola de la señal, aunque tenía suficiente amplitud, el Detector de Actividad Vocal ha estimado que es una trama de silencio. Esto se puede deber en primer lugar, al cambio adaptativo del umbral de potencia, ya que el umbral con el que compara no es fijo, sino que cambia de una trama a otra, subiendo si la señal sube y bajando si la señal lo hace. Por otra parte, también el VAD habrá estimado que al final de la señal ya no hay componente de pitch, debido a que el final de esta señal es un ruido parecido a un 'soplo de aire' y carece por tanto de pitch.

Veamos ahora la trama de voz reconstruida junto con la decisión del VAD.



Aquí vemos que aún cuando el final de la señal ha sido considerado como trama de silencio, la señal tiene bastante amplitud. Esto se debe a que los parámetros más importantes de la señal, como son los *LARc*, que contiene la amplitud de la envolvente del espectro, fueron modificados por una media de los *LARc* de las cuatro trama anteriores. Por tanto la señal sigue conservando su amplitud. Veamos el espectrograma de la señal reconstruida junto con la decisión del VAD para poder así ver mejor este efecto.

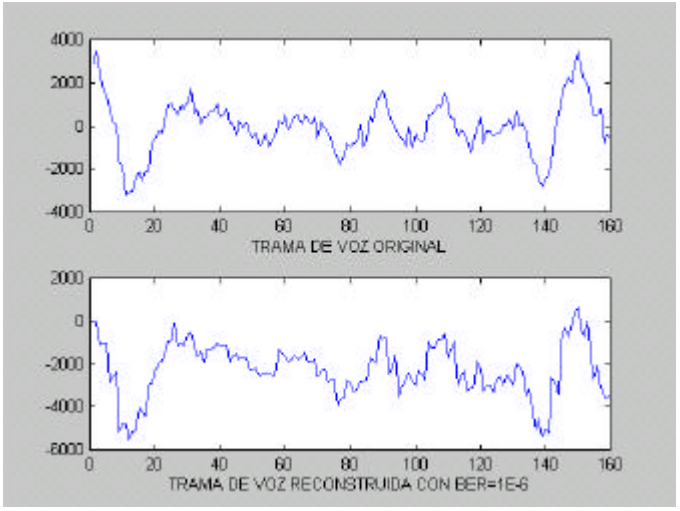


Vemos como realmente, al reconstruir la señal, sigue conservando una amplitud grande, aunque la decisión del VAD es cero.

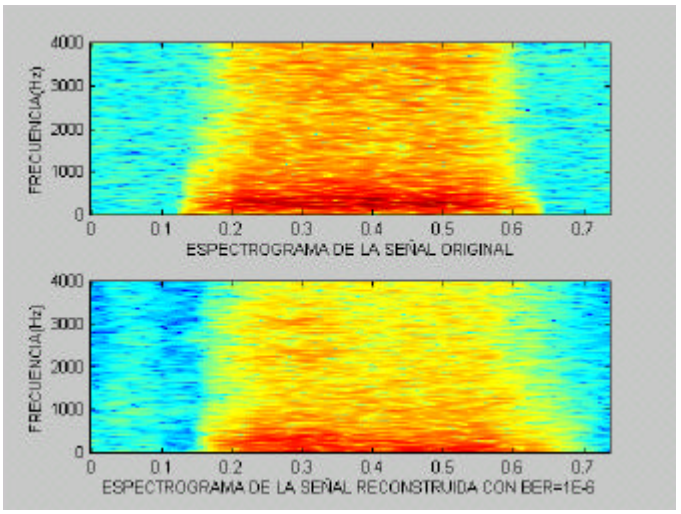
7.3.3 CODIFICACIÓN DE CANAL.

Veamos, por último, cómo afecta la tasa de error del canal a esta señal. Del análisis anterior pudimos deducir que la tasa de error máxima admisible era de 10^{-3}, por lo que omitiremos el análisis para la tasa de 10^{-9}.

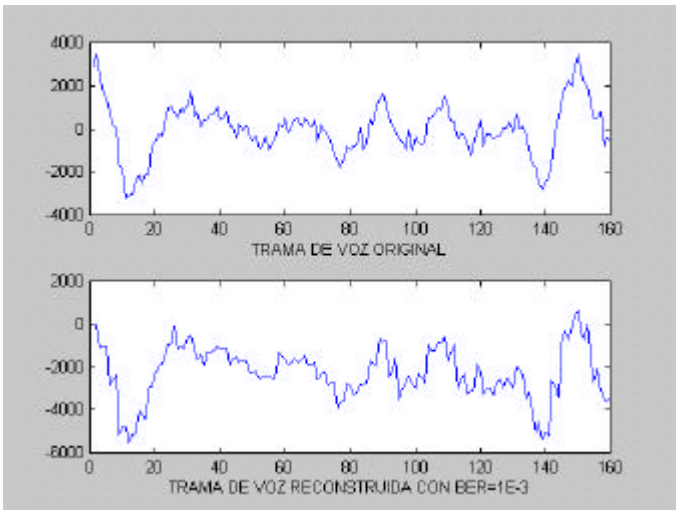
Y empezaremos viendo los resultados para una BER = 10^{-6}.



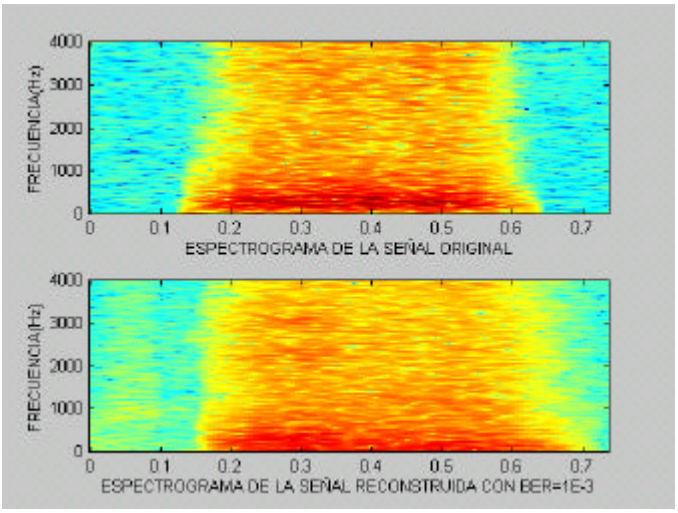
Vemos que la señal está reconstruida bastante bien. Veámosla en frecuencia.



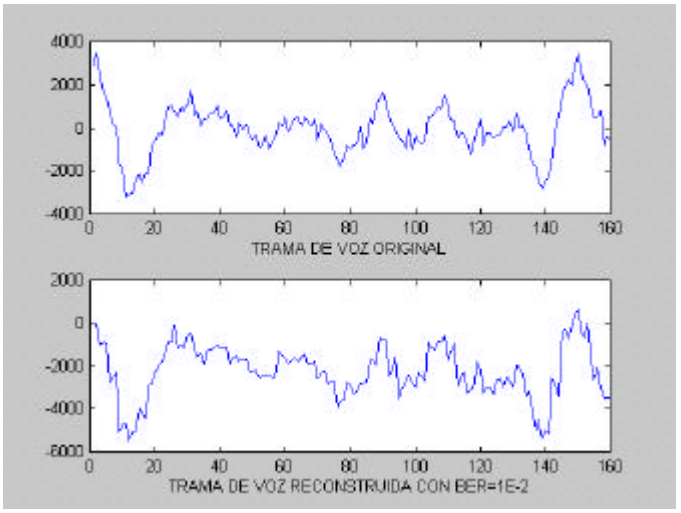
Podemos observar que esa tasa de error es admisible. Veamos ahora la tasa máxima admisible, que era de 10^{-3}.



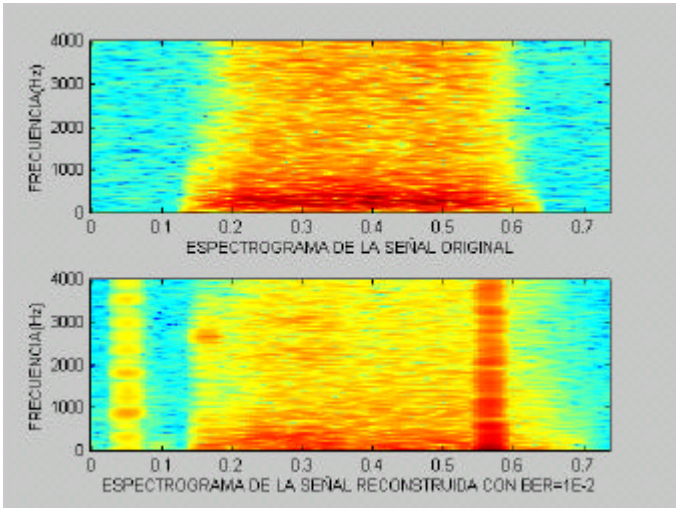
Vemos que aún así, la señal parece válida. Observémosla en frecuencia.



Aunque el espectro se ha ensanchado un poco, parece que la señal sigue siendo válida. Analicemos ahora una tasa de error de 10^{-2}.



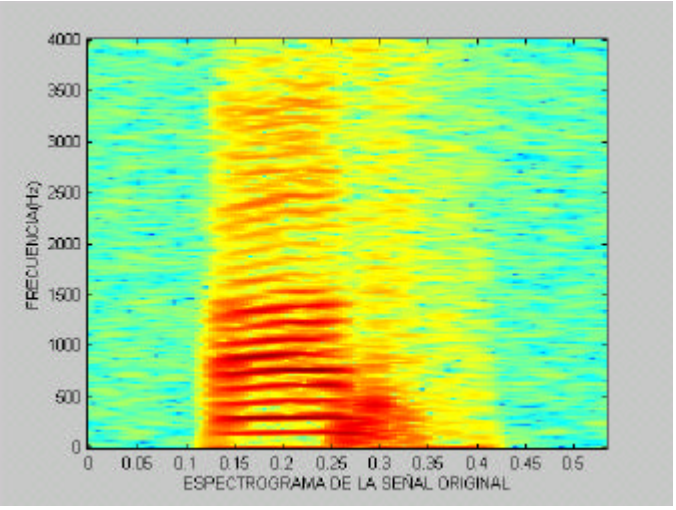
Y veámosla también en frecuencia.



Aquí ocurre todo lo contrario que en caso de vocal. Aunque parece que la señal sigue siendo válida en el dominio temporal, cuando la observamos en frecuencia vemos que el espectro se ha deformado lo bastante como para considerarla errónea.

7.4 ANÁLISIS DE UNA PALABRA.

A continuación, la simulación será sobre una palabra completa, en concreto la palabra *'as'*. Esta palabra contiene los dos fonemas analizados anteriormente, tanto una vocal como una consonante. Así podremos observar mejor las diferencias comentadas anteriormente entre ambos fonemas. Veamos en primer lugar el espectrograma de la señal original.

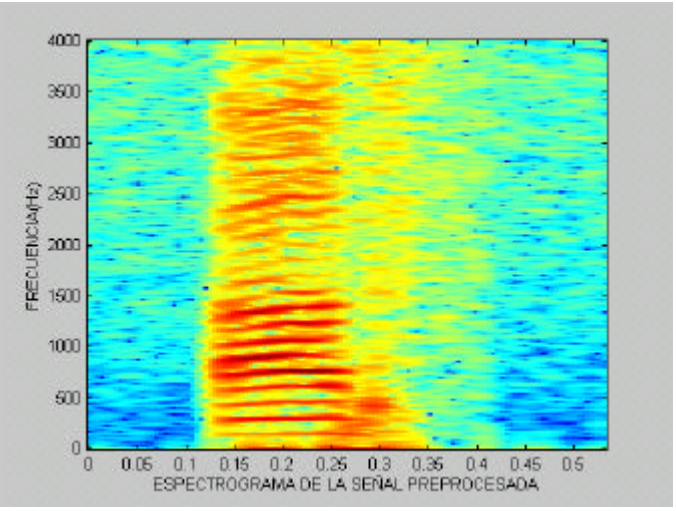


Observamos que la señal está compuesta por dos fonemas, uno inicial que dura desde 0.12 seg hasta 0.25 seg y el segundo fonema que dura desde 0.25 seg hasta 0.4 seg. Esta diferencia se puede apreciar gracias a que tiene distintos comportamientos en frecuencia.

7.4.1 CODIFICADOR Y DECODIFICADOR.

Sección de Preprocesado.

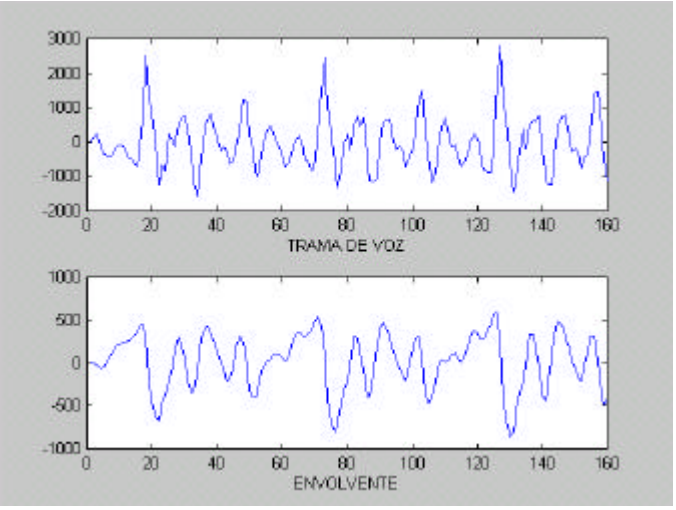
Únicamente vamos a ver en esta sección el espectrograma de la señal preprocesada.



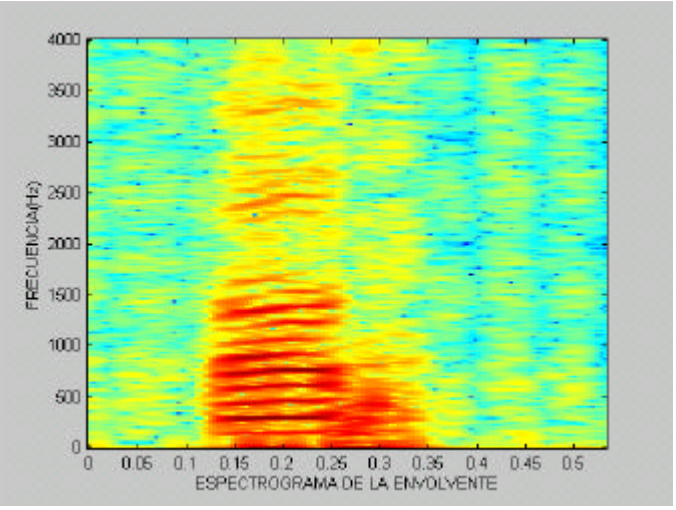
Observamos lo que mismo que comentamos en anteriores apartados, se enfatiza un poco la alta frecuencia para ganar resolución.

Sección STP.

A continuación, veamos una trama de voz original y la envolvente del espectro estimada a través de los coeficientes de reflexión.

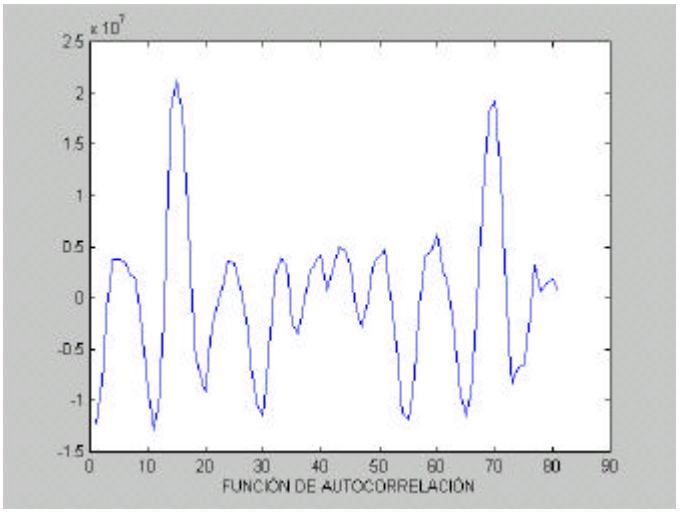


Y veamos también el comportamiento en frecuencia de la envolvente del espectro estimada.

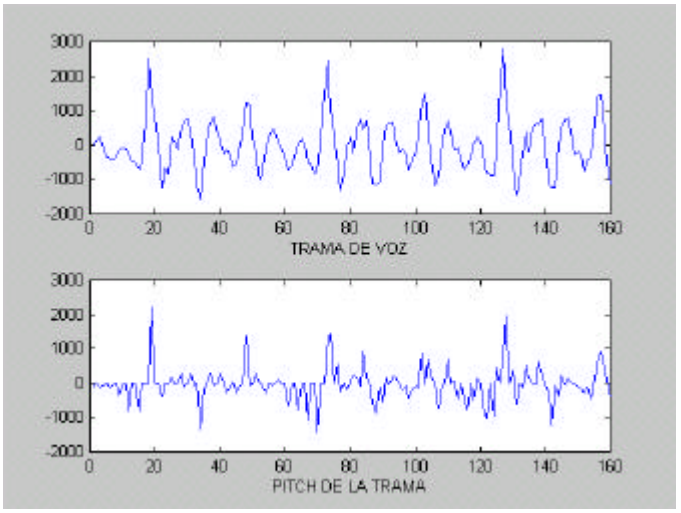


Sección LTP.

En primer lugar veremos la función de autocorrelación de una trama de voz, necesaria en esta sección para estimar el pitch de la señal.

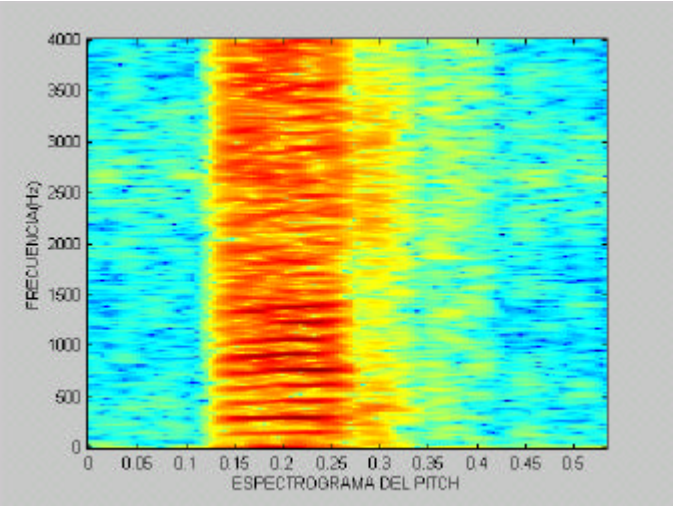


La distancia entre dos máximos es de 55 muestras, que se corresponde con una frecuencia de pitch de 145 Hz. Veamos a continuación el pitch estimado para una trama de voz.



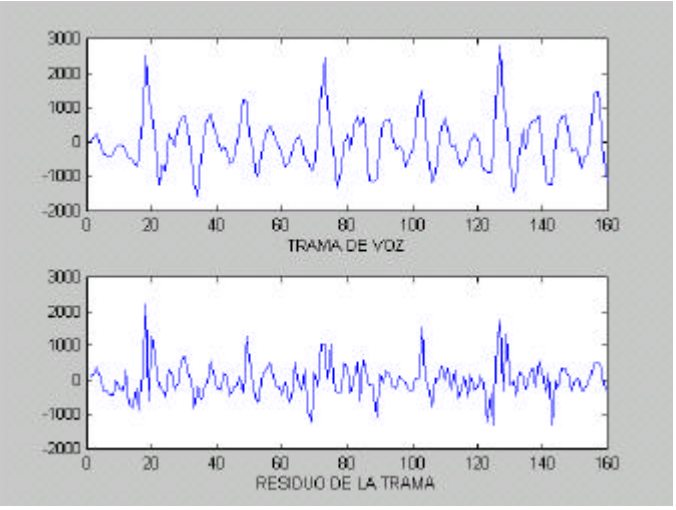
Vemos como el pitch sigue la forma de onda de la señal, pero con la frecuencia fundamental o frecuencia de pitch, que es la componente más importante de la señal.

Veamos, a continuación, el espectrograma del pitch.



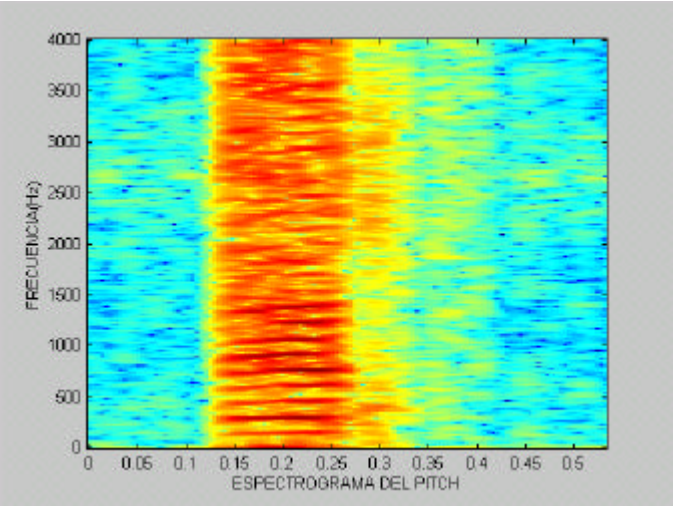
Sección RPE.

En esta sección, veremos, en primer lugar, una trama de voz original junto con el residuo estimado para dicha trama.



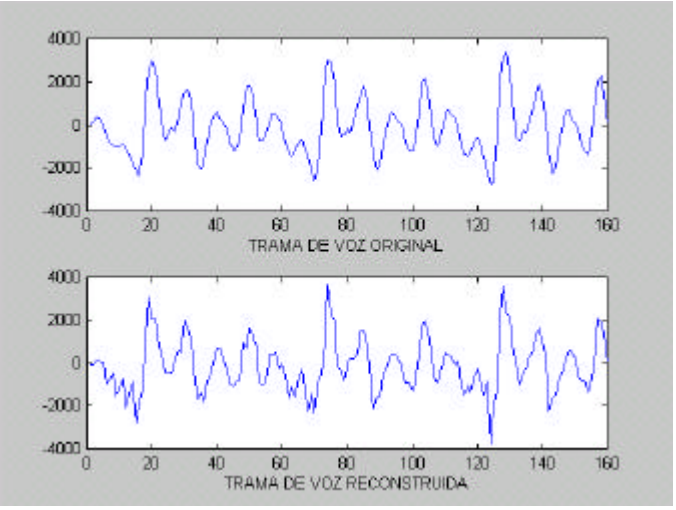
Vemos la excitación residual de dicha trama, base a partir de la cual se reconstruirá en el decodificador, agregándole el pitch y la envolvente del espectro estimados.

Veamos ahora el pitch completo de la señal en frecuencia.

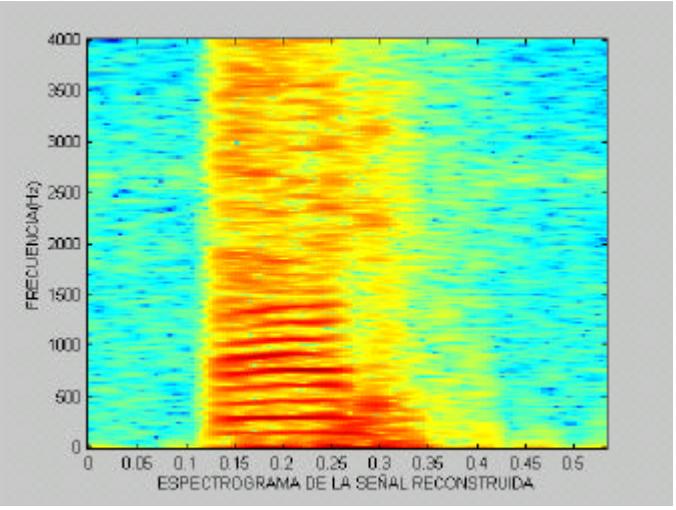


Decodificador.

Por último, veamos una trama de voz original junto con la trama de voz reconstruida en el decodificador.

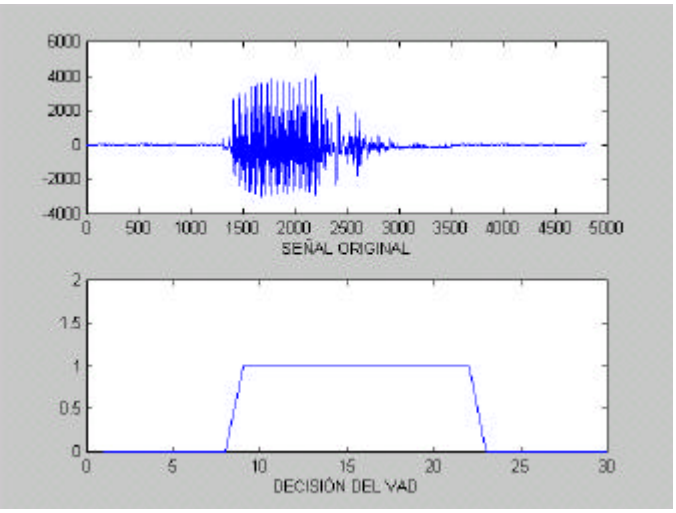


Y veamos también el espectrograma de la señal reconstruida.

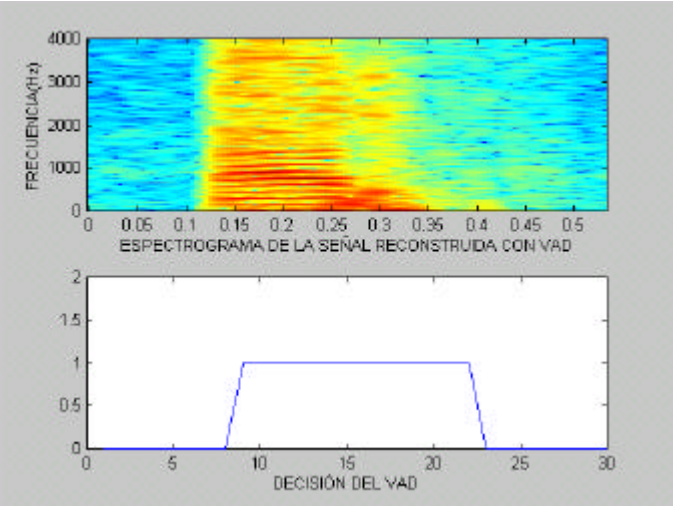


7.4.2 DETECTOR DE ACTIVIDAD VOCAL.

Veamos el apartado más importante para esta señal, que es el VAD. Y lo es porque esta señal contiene dos fonemas adyacentes. El Detector de Actividad Vocal debe ser capas de identificarlos y de no estimar ninguna trama de silencio en la transición entre ambos fonemas. Para ello, veamos la señal original completa junto con los valores de decisión.

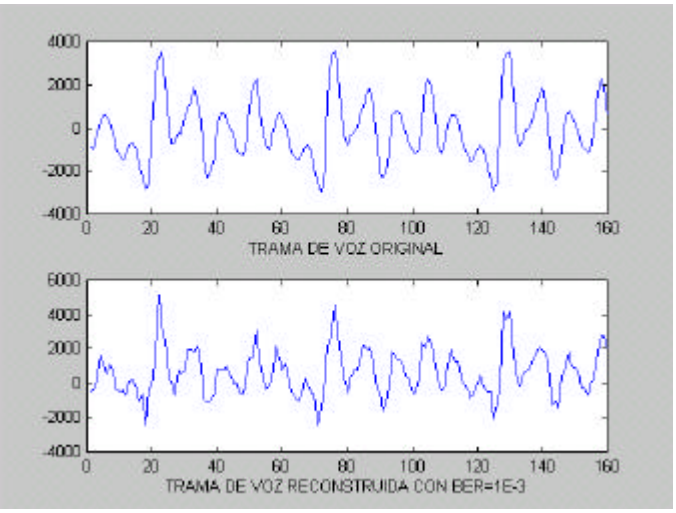


Vemos claramente los dos fonemas separados. En primer lugar, la ‘a’, señal con gran energía y de gran amplitud. Seguidamente, la ‘s’, señal un poco más ruidosa y de menor amplitud, con una larga cola que aún así el VAD actúa correctamente y la distingue como trama de voz. Veamos también en este apartado, el comportamiento en frecuencia de la señal reconstruida, junto con los valores de decisión.

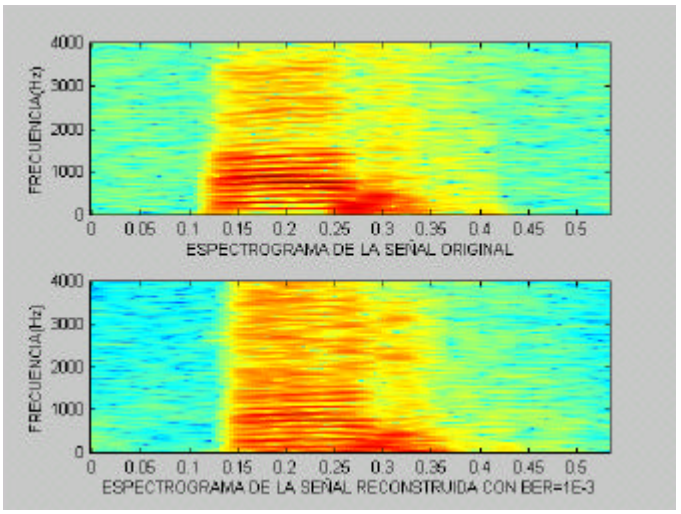


7.4.3 CODIFICACIÓN DE CANAL.

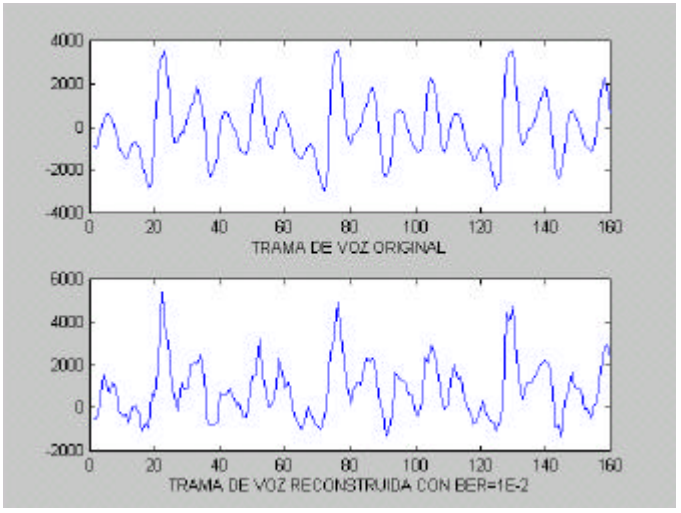
Por último, para esta señal, hagamos un análisis del comportamiento ante errores en el canal. Vamos a analizar, en primer lugar, una tasa de error del canal de 10^{-3}, que es mayor tasa de error admisible en una comunicación GSM.



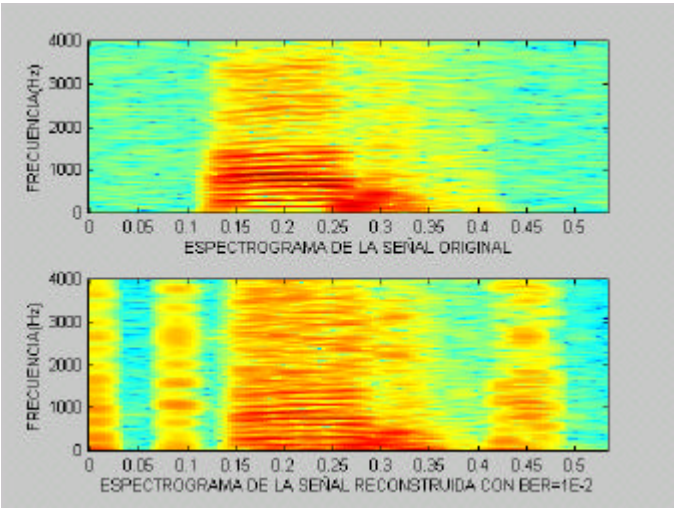
Vemos, como realmente, ya la señal empieza un poco a degradarse. Aún así, la señal parece seguir siendo completamente válida. Para reforzar esta afirmación, veamos la señal en frecuencia.



Vemos como la señal reconstruida se parece bastante a la señal original, conservando todas las características frecuenciales de ésta última. Si subimos la BER hasta 10^{-2}, veamos cómo afecta a la señal.



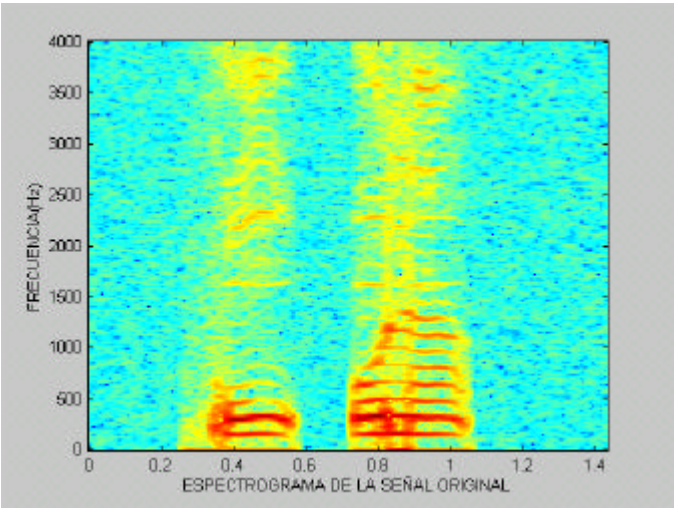
La señal reconstruida tiene una forma de onda parecida a la original, aunque tiene superpuesto un ruido debido al error que ya parece que sea excesivo, por lo que no se consideraría como una trama válida, y por tanto, sería una BER excesiva. Veamos la señal en frecuencia.



Vemos como la señal en frecuencia sufre ya una modificación de las componentes espectrales lo suficiente como para considerar esta BER como excesiva.

7.5 ANÁLISIS DE UN PAR DE PALABRAS SIMPLES.

En esta última simulación, analizaremos la frase “si, uno”. Así, este capítulo de simulaciones quedará más completo, puesto que si en el anterior apartado analizamos una palabra completa, aquí además va a haber otra palabra adicional, con un periodo de silencio intermedio entre ambas. Es por ello que el apartado más importante será el análisis del VAD. Veamos en primer lugar, el espectrograma de la señal original.

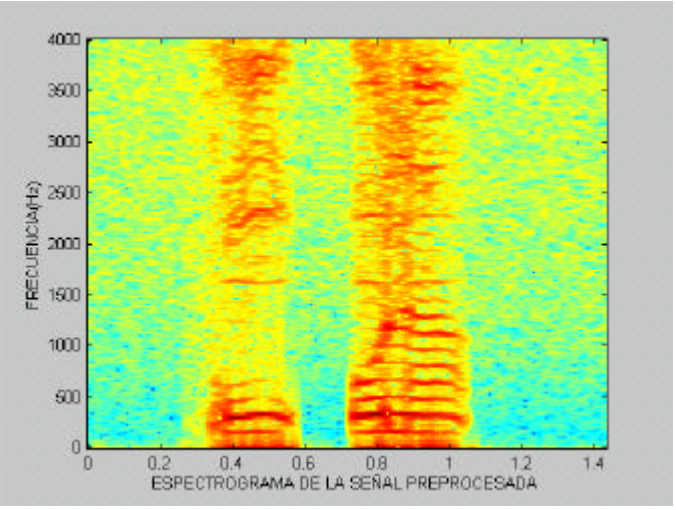


Vemos claramente diferenciadas ambas palabras, e incluso dentro de cada palabra se pueden diferenciar los fonemas.

7.5.1 CODIFICADOR Y DECODIFICADOR.

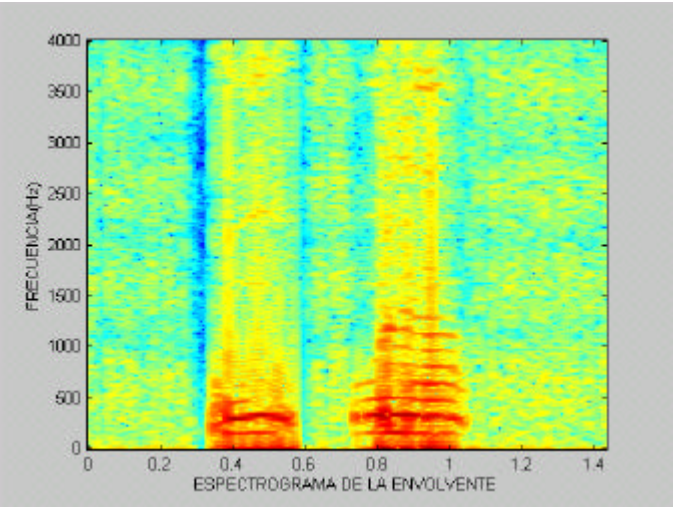
Sección de Preprocesado.

Veamos el espectrograma de la señal preprocesada.



Sección STP.

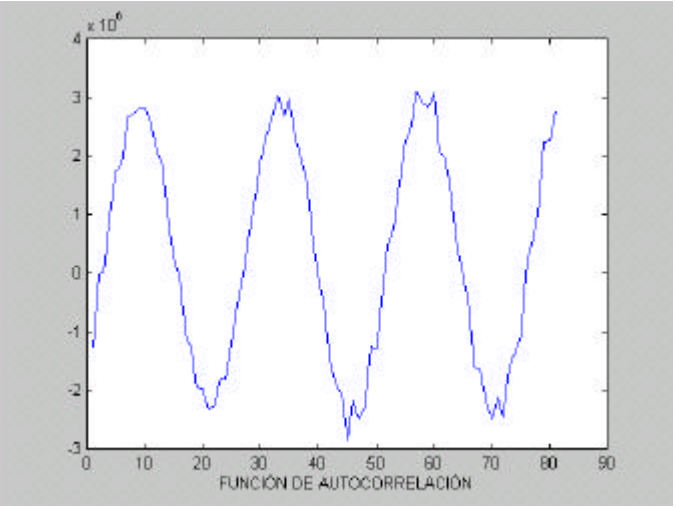
Vamos a ver únicamente la envolvente del espectro estimada en frecuencia.



Vemos de nuevo, que la envolvente del espectro es estimada, aparentemente, muy bien por el filtro de esta sección.

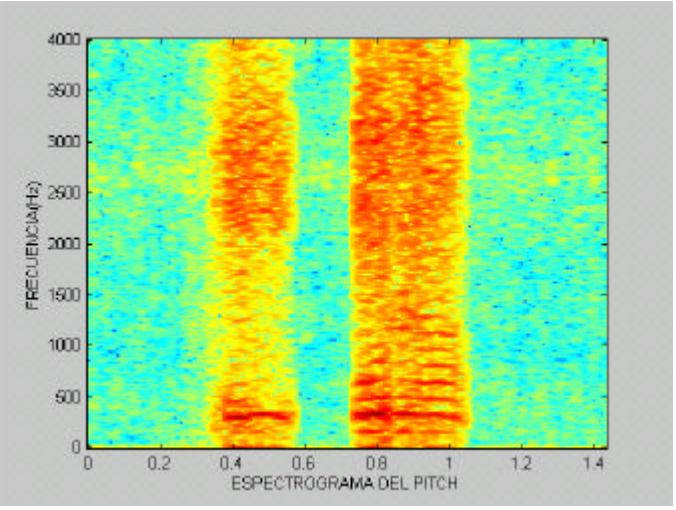
Sección LTP.

Veamos la función de autocorrelación para una trama de voz de esta señal.



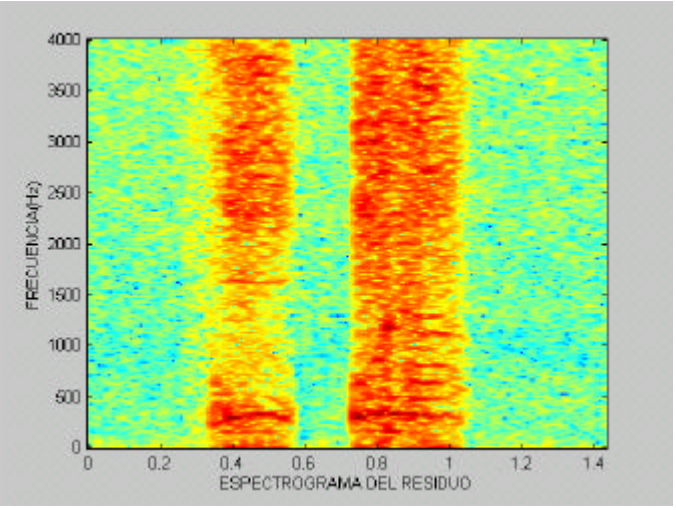
Aproximadamente, la distancia entre picos es de 30 muestras, que se corresponde con una frecuencia de pitch de 266 Hz.

Veamos también el pitch estimado en frecuencia.



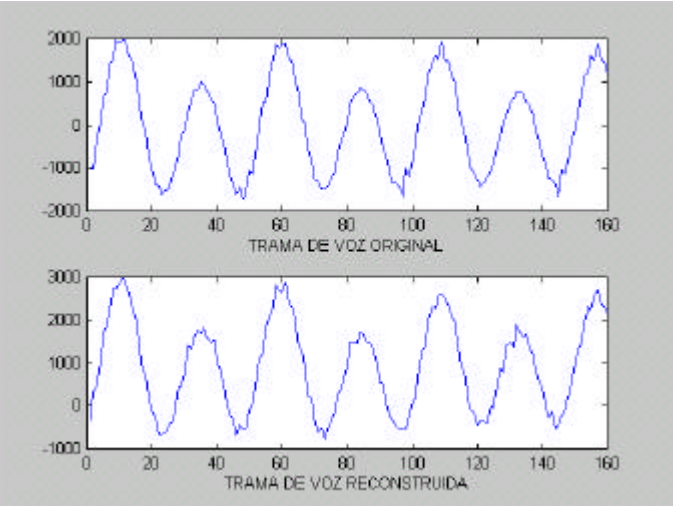
Sección RPE.

Veamos únicamente el espectrograma de la excitación residual estimada para esta señal.

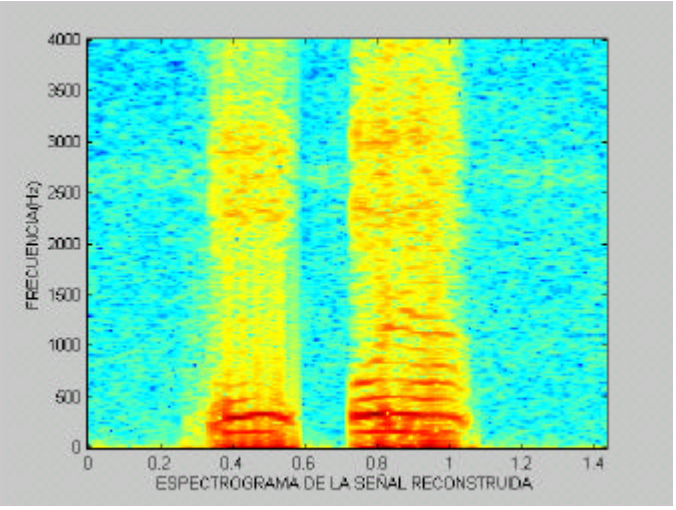


Decodificador.

Vamos a ver una trama de voz original junto con la misma trama de voz reconstruida a la salida del decodificador.

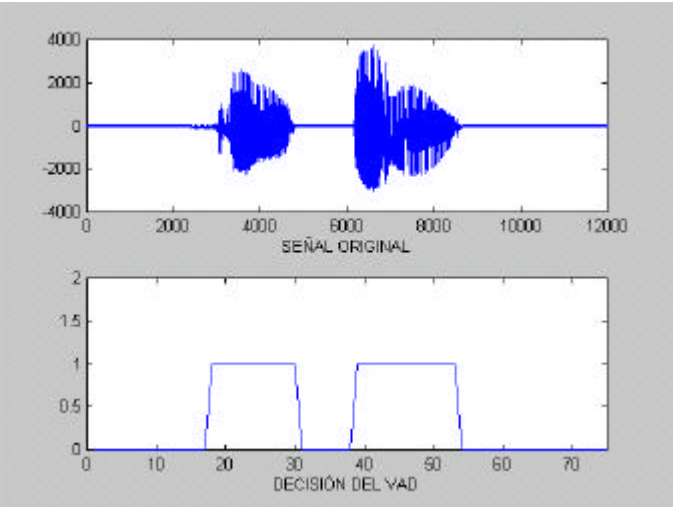


Vemos como la señal reconstruida se parece de forma casi perfecta a la señal original. Para ver mejor este parecido, veamos la señal reconstruida en frecuencia.

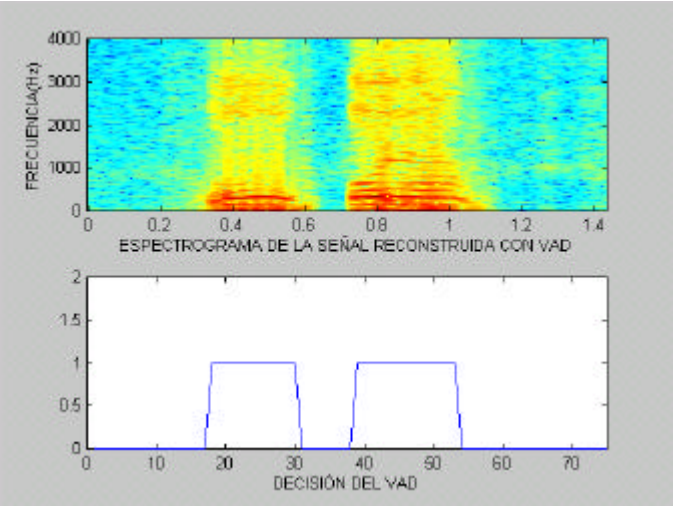


7.5.2 DETECTOR DE ACTIVIDAD VOCAL.

Entramos ahora en el apartado más importante de esta señal, que es el Detector de Actividad Vocal. El VAD debe ser capaz de distinguir ambas palabras, y además de insertar entre ambas un periodo de silencio. Veamos la señal original junto con los valores de decisión.

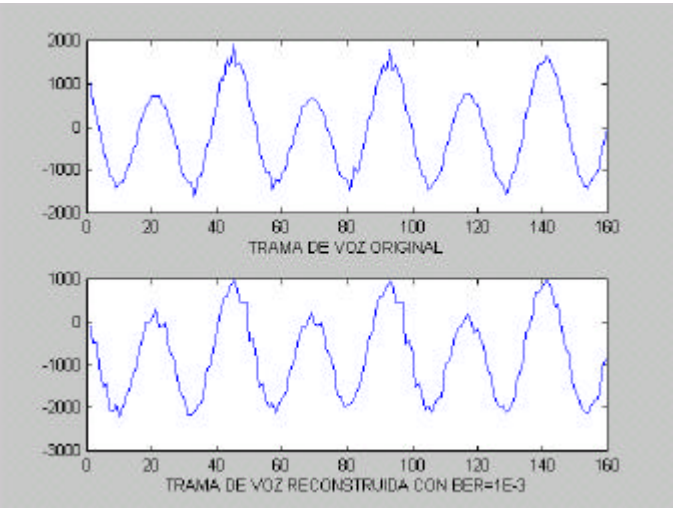


Vemos que el VAD actúa correctamente, distinguiendo claramente cuando hay señal y cuando no la hay. Con ello se consigue una mejor eficiencia en la transmisión de los bits, ahorrándose gran parte de ellos. También podemos observar como la ‘s’ inicial la estima correctamente y no la confunde con ruido. Veamos por último la señal reconstruida en frecuencia junto con los valores de decisión.

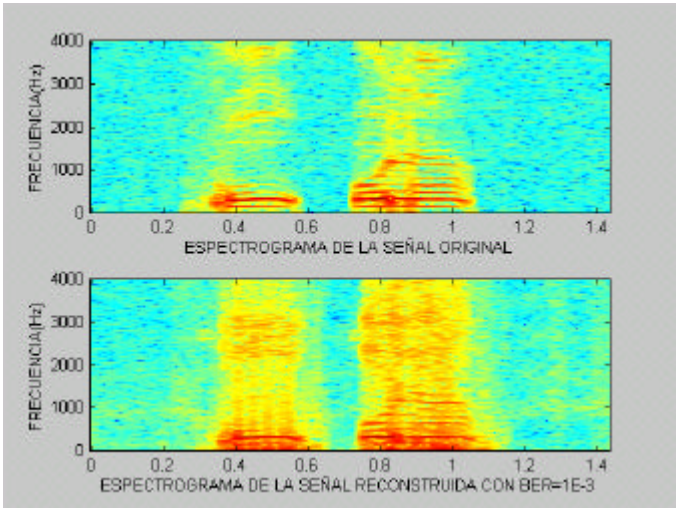


7.5.3 CODIFICACIÓN DE CANAL.

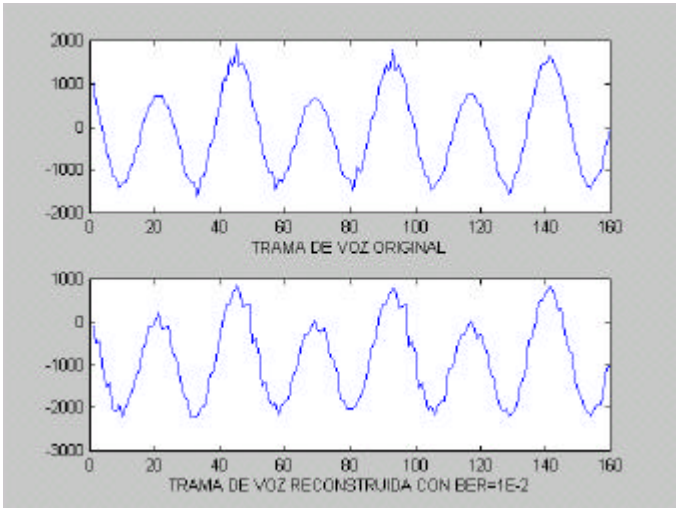
Por último veamos el efecto de una tasa de error del canal. En primer lugar para una BER de 10^{-3}.



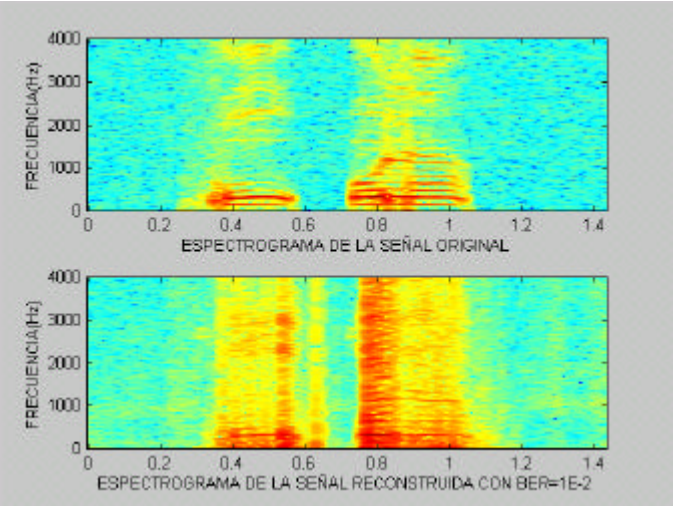
Aunque tiene un pequeño ruido superpuesto a la señal, podemos seguir considerando la trama como válida. Si la observamos en frecuencia, vemos como las principales componentes espectrales de la señal siguen conservándose, por lo que podemos concluir que esta tasa de error es admisible.



Analicemos, a continuación, una BER de 10-2.



La señal sigue conservando la forma de onda original, pero tiene un ruido superpuesto debido al error excesivo, por lo que ya no sería válida. Veamos dicha señal en frecuencia.



Observamos una modificación de las componentes espectrales excesivo, por lo que podemos concluir que la tasa de error máxima admisible para una transmisión GSM es de BER = 10^{-3}.

Capítulo 8 .
CONCLUSIONES.

8.1 INTRODUCCIÓN.

En este último capítulo comentaremos los resultados obtenidos en el capítulo de simulaciones, y llegaremos a una serie de conclusiones. No se trata de llegar a resultados cuantitativos, sino más bien cualitativos. Veremos los aspectos de funcionamiento de las distintas partes del codificador GSM, y cómo afectan a éste distintos factores externos, como pueden ser el ruido, la amplitud, o la forma de onda a la entrada.

8.2 CONCLUSIONES.

Empezaremos con las distintas tasas que maneja el codificador. Antes del codificador GSM, llega una señal PCM 64 Kbis/seg, y existe un módulo especial que eleva la resolución de 8 bits a 13 bits. Por tanto, al codificador GSM le llega una trama de 160 muestras cada 20 ms, cuantizadas y codificadas con 13 bits, por lo que resulta una tasa de entrada al codificador de 104 Kbits/seg. La tasa de salida del codificador GSM, antes de la codificación de canal es de 260 bits cada 20 ms, que provocan una tasa de 13 Kbis/seg. Posteriormente a la codificación de canal, dado que ésta le añade bits de redundancia para protegerlo contra posibles errores en el canal de transmisión, van saliendo por la interfaz aérea cuatro tramas de 114 bits cada 20 ms, que producen una tasa aérea de 22,8 Kbits/seg.

Independientemente de cada tasa, el codificador GSM, incluido el Detector de Actividad Vocal y el Codificador de Canal, deben ser capaces de procesar una trama completa en un tiempo de 20 ms. Además el VAD debe esperar a que termine el Codificador, y el Codificador de Canal a que terminen ambos. Todo esto debe realizarse en tiempo real, lo que implica la implementación de este codificador en DSP's, que son dispositivos que pueden trabajar con tasas elevadas en tiempo real.

Otro aspecto importante a destacar sería el ruido que viene sumado a la señal que llega. La señal puede provenir de un ambiente tranquilo, como del interior de una oficina, de la casa, o de ambientes tan ruidosos como una obra o una calle con mucho tráfico. El codificador GSM es lo suficientemente robusto como para poder distinguir entre señal y ruido, todo ello gracias al Detector de Actividad Vocal, que elimina largos periodos de ruido sustituyéndolos por silencio.

En relación al VAD, vimos en el capítulo anterior que podía provocar fallos en las colas de algunas consonantes, como por ejemplo, la cola de la 's', que la confunde con ruido, aún conservando una amplitud apreciable. Éste efecto era provocado por el cambio adaptativo del umbral de potencia con el que compara y a que no se detectaba componente de pitch en la señal. Sin embargo, este error no se cometía cuando la 's' iba seguida de otro fonema, es decir, cuando la 's' es inicial o intermedia. Sólo se pueden producir fallos cuando es el fonema final de una palabra.

Por último comentaremos la eficiencia de la codificación de canal. La codificación de canal GSM es bastante robusta. En primer lugar se reordenan los bits por su importancia, para a continuación codificar mejor algunos bits según el peso asignado. Además hay dos mecanismos potentes de protección, como son el código cíclico y el código convolucional. Posteriormente hay otros mecanismos como entrelazado de bloques, entrelazado de tramas, etc. Todo esto nos lleva a que una tasa de error de 10^{-3} sea una tasa error admisible, lo cual no lo sería para otros tipos de transmisiones, como radioenlaces, etc.

ANEXO A

Describimos a continuación las funciones MATLAB que hemos utilizado para la simulación. En primer lugar veamos el programa principal. Recibe como entrada la señal original, muestreada a 8 KHz y cuantificada con 13 bits, y genera a su salida la señal procesada, muestreada también a 8 KHz y cuantificada con 13 bits. Contiene a su vez otros ocho submódulos, a los cuales llama, que son, por este orden: Codificador GSM, Detector de Actividad Vocal, Codificador de Tramas de Silencio, Codificador de Canal, Canal de Transmisión, Decodificador de Canal, Decodificador de Tramas de Silencio y Decodificador GSM.

```
%CODIFICADOR Y DECODIFICADOR GSM CON VAD Y CODIFICACIÓN DE CANAL.
%function sp=gsm(so)
%Entradas:
%so:señal muestreada a 8KHz y cuantificada con 13 bits.
%Salidas:
%sp:señal muestreada a 8KHz y cuantificada con 13 bits.
function sp=gsm(so)
inicia %fichero de variables iniciales.
for i=1:ntramas
   [LARc,Nc,bc,xmaxc,xmc,ACF,LARantc,dpc]=gsmcod(so(160*i-159:160*i),
LARantc,dpc);%codificamos la señal de entrada trama a trama.
   [vad1,rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac]=vad(ACF,
Nc,rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac);%aplicamos el
vad.
   [LARc,Nc,bc,Mc,xmaxc,xmc,flag,vad2,vad3,vad4,LARsid,xmaxsid]=sid(
LARc,Nc,bc,Mc,xmaxc,xmc,flag,vad2,vad3,vad4,LARsid,xmaxsid]=sid(LARc,
Nc,bc,Mc,xmaxc,xmc,vad1,vad2,vad3,vad4,LARsid,xmaxsid);%construimos
la trama sid, si procede.
   [tr1,tr2,tr3,tr4,t5a,t6a,t7a,t8a]=codcanal(LARc,Nc,bc,Mc,xmaxc,xmc,
t5a,t6a,t7a,t8a); %añadimos la codificación de canal.
   [tr1,tr2,tr3,tr4]=canal(tr1,tr2,tr3,tr4,BER); %enviamos las tramas
a través del canal.
   [LARc,Nc,bc,Mc,xmaxc,xmc,t1a,t2a,t3a,t4a]=descodcanal(tr1,tr2,tr3,
tr4,t1a,t2a,t3a,t4a); %quitamos la codificación de canal.
   [LARc,Nc,bc,Mc,xmaxc,xmc]=dessid(LARc,Nc,bc,Mc,xmaxc,xmc,flag);
%procesamos las tramas sid.
   [sp(160*i-159:160*i),Nact,dpd,LARantd]=gsmdec(LARc,Nc,bc,Mc,xmaxc,
xmc,Nact,dpd,LARantd);%decodificamos para construir la señal de salida
end
```

Veamos a continuación, el módulo del codificador GSM. La entrada es una trama de 160 muestras, y las salidas son los parámetros de la trama. Devuelve además a la salida los coeficientes de autocorrelación calculados, debido a que los utilizará después el Detector de Actividad Vocal. Además recursivamente va guardando los LAR y las 120 muestras anteriores reconstruidas de la trama actual, necesarios para la trama siguiente. Contiene a su vez cuatro submódulos a los cuales llama, correspondientes a las cuatro secciones de las que se compone el codificador: sección de preprocesado, sección STP, sección LTP y sección RPE.

```
%CODIFICADOR GSM.
%function
[LARc,Nc,bc,Mc,xmaxc,xmc,ACF,LARantc,dpc]=gsmcod(so,LARantc,dpc)
%Entradas:
%so:trama de 160 muestras muestreadas a 8KHz y cuantificadas con 13
bits.
%LARantc:LAR de la anterior trama.
%dpc:120 muestras anteriores reconstruidas.
%Salidas:
%LARc:LAR codificados.
%Nc:término de autocorrelación codificado.
%bc:ganancia codificada.
%Mc:posición RPE codificada.
%xmaxc:valor RPE máximo codificado.
%xmc:subsecuencia RPE codificada.
%ACF:término de autocorrelación para el vad.
%LARantc:LAR anteriores para la siguiente trama.
%dpc:120 muestras anteriores para la siguiente trama.
function
[LARc,Nc,bc,Mc,xmaxc,xmc,ACF,LARantc,dpc]=gsmcod(so,LARantc,dpc)
s=preproc(so);%aplicamos un preprocesado.
[LARc,ACF,LARantc,d]=stpc(s,LARantc); %codificación STP.
for n=1:4 %efectuamos el procesado dividiendo la trama en cuatro
subtramas.
   [Nc(n),bc(n),dpp]=ltpc(d(40*n-39:40*n),dpc); %codificación LTP.
   e=d(40*n-39:40*n)-dpp; %calculamos el error residual.
   [Mc(n),xmaxc(n),xmc(13*n-12:13*n),ep]=rpec(e); %codificación RPE.
   dp=ep+dpp; %recontruimos la señal que saldría del decodificador.
   dpc=[dpc(41:120),dp]; %actualizamos las 120 muestras anteriores
reconstruidas.
end
```

Veamos cada una de las cuatro secciones. En primer lugar, la sección de preprocesado. Este módulo contiene a su vez dos llamadas, una al filtro que elimina la componente de continua, y otro al filtro de preénfasis.

```
%PREPROCESADO.
%function s=preproc(so)
%Entradas:
%so:trama de 160 muestras.
%Salidas:
%s:trama de 160 muestras sin offset y amplificada.
function s=preproc(so)
sof=offset(so);%quitamos el offset de la entrada.
s=preemp(sof);%aplicamos un filtro de preénfasis.
```

Veamos cada uno de estos dos filtros. En primer lugar el filtro que elimina la componente de continua.

```
%ELIMINACIÓN DE LA COMPONENTE DE CONTINUA.
%function sof=offset(so)
%Entradas:
%so:trama de 160 muestras muestreadas a 8KHz y codificada con 13 bits.
%Salidas:
%sof:salida sin offset.
function sof=offset(so)
alpha=32735*(2^(-15)); %constante que define así la recomendación.
sof(1)=0;
for k=2:160
   sof(k)=so(k)-so(k-1)+alpha*sof(k-1);
end
```

Y a continuación el filtro de preénfasis.

```
%PREÉNFASIS.
%function s=preemp(sof)
%Entradas:
%sof:señal sin offset.
%Salidas:
%s:señal sin offset y amplificada.
function s=preemp(sof)
beta=28180*(2^(-15)); %constante que define así la recomendación.
s(1)=sof(1);
for k=2:160
   s(k)=sof(k)-beta*sof(k-1);
end
```

A continuación, veamos el módulo correspondiente a la sección STP. Este módulo recibe como entrada la trama de entrada preprocesada y los LAR de la trama anterior. Devuelve a su salida los LAR codificados, los coeficientes de autocorrelación, los LAR para la siguiente trama y la envolvente del espectro estimada para la trama actual.

```
%CODIFICACIÓN DE LA ENVOLVENTE.
%function [LARc,ACF,LARantc,d]=stpc(s,LARantc)
%Entradas:
%so:trama de entrada de 160 muestras muestreadas a 8KHz y
cuantificadas con 13 bits preprocesada.
%LARantc:LAR de la trama anterior.
%Salidas:
%LARc:LAR codificados.
%ACF:términos de autocorrelación para el vad.
%LARantc:LAR anteriores para la siguiente trama.
%d:señal sin envolvente para posterior procesado.
function [LARc,ACF,LARantc,d]=stpc(s,LARantc)
ACF=autocorr(s);%calculamos los coeficientes de autocorrelación.
r=rschur(ACF);%calculamos los coeficientes de reflexión.
LAR=logarea(r);%calculamos los LAR.
LARc=larcod(LAR);%codificamos los LAR.
LARact=lardec(LARc);%decodificamos los LAR codificados.
LARi=interpol(LARact,LARantc);%interpolamos los LAR con los de la
trama anterior.
LARantc=LARact;%actualizamos los LAR de la trama anterior para la
siguiente trama.
rp=reflect(LARi);%calculamos los coeficientes de reflexión a partir de
los LAR interpolados.
d=invfilt(s,rp);%aplicamos el filtro que elimina la envolvente.
```

Veamos cada uno de los módulos que utiliza el codificador STP. En primer lugar el módulo que calcula los coeficientes de autocorrelación.

```
%AUTOCORRELACIÓN.
%function ACF=autocorr(s)
%Entradas:
%s:señal sin offset y amplificada.
%Salidas:
%ACF:coeficientes de autocorrelación.
function ACF=autocorr(s)
for k=1:9
   aux=zeros(1,160);
   for i=k:160
      aux(i)=s(i)*s(i-k+1);
   end
   ACF(k)=sum(aux); %calculamos la autocorrelación para cada trama.
end
```

Veamos ahora un módulo muy importante. Es el que calcula los coeficientes de reflexión a partir de los coeficientes de autocorrelación usando la recursión de Schur.

```
%RECURSIÓN DE SCHUR.
%function r=rschur(ACF)
%Entradas:
%ACF:coeficientes de autocorrelación.
%Salidas:
%r:coeficientes de reflexión.
function r=rschur(ACF)
if (ACF(1)==0)
   r=zeros(1,8);
   return
end
for i=2:8
   k(10-i)=ACF(i);
end
for j=1:9
   P(j)=ACF(j);
end
for n=1:8
   if (P(1)<abs(P(2)))
      r(n:8)=zeros(1,(9-n));
      return
   else
      r(n)=abs(P(2))/P(1);
   end
   if (P(2)>0)
      r(n)=-r(n);
   end
   if(n==8)
      return
   end
   P(1)=P(1)+P(2)*r(n);
   for m=1:(8-n)
      P(m+1)=P(m+2)+r(n)*k(9-m);
      k(9-m)=k(9-m)+r(n)*P(m+2);
   end
end
```

Veamos ahora el módulo que calcula los LAR a partir de los coeficientes de reflexión.

```
%ESTIMADOR DE LOS LAR.
%function LAR=logarea(r)
%Entradas:
%r:coeficientes de reflexión.
%Salidas:
%LAR:Log-Area-ratios.
function LAR=logarea(r)
for i=1:8
   if(abs(r(i))<0.675) %Hacemos una codificación por intervalos.
      LAR(i)=r(i);
   elseif(abs(r(i))<0.95)
      LAR(i)=sign(r(i))*(2*abs(r(i))-0.675);
   elseif (abs(r(i))<=1)
      LAR(i)=sign(r(i))*(8*abs(r(i))-6.375);
   end
end
```

Veamos ahora el módulo que a partir de los LAR estimados, los codifica finalmente para su envío.

```
%CODIFICADOR DE LOS LAR.
%function LARc=larcod(LAR)
%Entradas:
%LAR:Log-Area-Ratios.
%Salidas:
%LARc:LAR codificados.
function LARc=larcod(LAR)
A=[20 20 20 20 13.637 15 8.334 8.824];
B=[0 0 4 -5 0.184 -3.5 -0.666 2.235];
MAC=[31 31 15 15 7 7 3 3];
MIN=[-32 -32 -16 -16 -8 -8 -4 -4];
for i=1:8;
   LARc(i)=round(A(i)*LAR(i)+B(i));
end
for i=1:8
   if(LARc(i)<MIN(i))
      LARc(i)=MIN(i);
   elseif(LARc(i)>MAC(i))
      LARc(i)=MAC(i);
   end
end
```

A continuación, el módulo que decodifica los LAR para posteriormente interpolarlos.

```
%DECODIFICADOR DE LOS LAR.
%function LARd=lardec(LARc)
%Entradas:
%LARc:LAR codificados.
%Salidas:
%LARd:LAR decodificados.
function LARd=lardec(LARc)
A=[20 20 20 20 13.637 15 8.334 8.824];
B=[0 0 4 -5 0.184 -3.5 -0.666 2.235];
i=1:8;
LARd(i)=(LARc(i)-B(i))/A(i);
```

Y a continuación el módulo que interpola los LAR con los de la trama anterior.

```
%INTERPOLADOR DE LOS LAR.
%function LARi=interpol(LARact,LARant)
%Entradas:
%LARact:LAR decodificados de la trama actual.
%LARant:LAR decodificados de la trama anterior.
%Salidas:
%LARi:LAR interpolados.
function LARi=interpol(LARact,LARant)
j=1:8;
LARi(j)=0.75*LARant(j)+0.25*LARact(j);
LARi(j+8)=0.50*LARant(j)+0.50*LARact(j);
LARi(j+16)=0.25*LARant(j)+0.75*LARact(j);
LARi(j+24)=LARact(j);
```

Veamos el módulo que, a partir de los LAR interpolados, calcula los coeficientes de reflexión, necesarios posteriormente para estimar la envolvente.

```
%DECODIFICACIÓN DE LOS COEFICIENTES DE REFLEXIÓN.
%function r=reflect(LARi)
%Entradas:
%LARi:LAR interpolados.
%Salidas:
%r:coefientes de reflexión.
function r=reflect(LARi)
for j=1:32
   if(abs(LARi(j))<0.675)
      r(j)=LARi(j);
   elseif(abs(LARi(j))<1.225)
      r(j)=sign(LARi(j))*(0.5*abs(LARi(j))+0.3375);
   elseif(abs(LARi(j))<=1.625)
      r(j)=sign(LARi(j))*(0.125*abs(LARi(j))+0.796875);
   end
end
```

Por último, para finalizar con esta sección, veamos el filtro que estima y elimina la envolvente del espectro de la trama actual.

```
%FILTRO QUE QUITA LA ENVOLVENTE.
%function d=invfilt(s,rp)
%Entradas:
%s:trama de 160 muestras de entrada.
%rp:coeficientes de reflexión decodificados.
%Salidas:
%d:trama de 160 muestras de salida sin envolvente.
function d=invfilt(s,rp)
uaux=zeros(1,8);
for k=1:160
   if(k<14)
      raux=rp(1:8);
   elseif(k<28)
      raux=rp(9:16);
   elseif(k<41)
      raux=rp(17:24);
   else
      raux=rp(25:32);
   end
   daux(1)=s(k);
   u(1)=s(k);
   for j=2:9
      daux(j)=daux(j-1)+raux(j-1)*uaux(j-1);
      u(j)=uaux(j-1)+raux(j-1)*daux(j-1);
   end
   d(k)=daux(9);
   uaux=u;
end
```

A continuación, estudiaremos el módulo correspondiente a la sección LTP, encargada de estimar y codificar el pitch de la señal.

```
%CODIFICADOR DEL PITCH.
%function [Nc,bc,dpp]=ltpc(d,dpc)
%Entradas:
%d:señal sin envolvente.
%dpc:120 muestras anteriores reconstruidas.
%Salidas:
%Nc:término de correlación codificado.
%bc:ganancia codificada.
%dpp:pitch de la señal.
function [Nc,bc,dpp]=ltpc(d,dpc)
[N,b]=calcltp(d,dpc);%calculamos los términos LTP.
[Nc,bc]=ltpcod(N,b);%codificamos los términos LTP.
[Np,bp]=ltpdec(Nc,bc);%decodificamos los términos LTP codificados.
dpp=bp*dpc((121-Np):(160-Np));%calculamos el pitch de la señal.
```

Veamos cada uno de los tres módulos que la componen. El primero es el encargado de calcular los parámetros LTP de la señal. Para ello calcula la función de correlación entre la trama actual y la anterior.

```
%ESTIMADOR DE LOS TÉRMINOS LTP.
%function[N,b]=calcltp(d,dp)
%Entradas:
%d:trama de 120 muestras de entrada.
%dp:120 muestras anteriores reconstruidas.
%Salidas:
%N:término de autocorrelación.
%b:ganancia.
function[N,b]=calcltp(d,dp)
for lambda=40:120
   R(lambda)=sum(d.*dp((121-lambda):(160-lambda)));
end
[var,Naux]=max(R(40:120));
N=Naux+39;
S=sum(dp((121-N):(160-N)).^2);
if(var<=0)
   b=0;
elseif(var>S)
   b=1;
else
   b=var/S;
end
```

A continuación, el módulo que codifica dichos parámetros para su posterior envío.

```
%CODIFICADOR DE LOS TÉRMINOS LTP.
%function[Nc,bc]=ltpcod(N,b)
%Entradas:
%N:término de autocorrelación.
%b:ganancia de autocorrelación.
%Salidas:
%Nc:término de autocorrelación codificado.
%bc:ganancia de autocorrelación codificada.
function[Nc,bc]=ltpcod(N,b)
Nc=N;
DLB=[0.2 0.5 0.8];
if(b<=DLB(1))
bc=0;
elseif(b<=DLB(2))
bc=1;
elseif(b<=DLB(3))
bc=2;
else
bc=3;
end
```

Y, por último, el módulo que los decodifica para estimar el pitch y eliminarlo de la trama actual.

```
%DECODIFICADOR DE LOS TÉRMINOS LTP.
%function[Np,bp]=ltpdec(Nc,bc)
%Entradas:
%Nc:término de autocorrelación codificado.
%bc:ganancia de autocorrelación codificado.
%Salidas:
%Np:término de autocorrelación decodificado.
%bp:ganancia de autocorrelación decodificado.
function[Np,bp]=ltpdec(Nc,bc)
Np=Nc;
QLB=[0.10 0.35 0.65 1.00];
bp=QLB(bc+1);
```

El último módulo es el correspondiente a la sección RPE, encargado de estimar y codificar la excitación residual de la trama.

```
%CODIFICADOR DE LA EXCITACIÓN RESIDUAL.
%function [Mc,xmaxc,xmc,ep]=rpec(e)
%Entradas:
%e:excitación residual.
%Salidas:
%Mc:posición RPE codificada.
%xmaxc:valor máximo RPE codificado.
%xmc:subsecuencia RPE codificada.
%ep:excitación residual decodificada.
function [Mc,xmaxc,xmc,ep]=rpec(e)
x=ponder(e); %aplicamos el filtro de weight.
[Mc,xm]=rpesel(x); %calculamos la subsecuencia seleccionada.
[xmc,xmaxc]=apcmcod(xm); %codificamos los términos RPE.
xmp=apcmdec(xmc,xmaxc); %decodificamos los términos RPE codificados.
ep=rpepos(Mc,xmp); %decodificamos la excitación residual.
```

Veamos cada uno de estos cinco submódulos. El primero es el correspondiente al filtro de ponderación.

```
%FILTRO DE PONDERACIÓN.
%function x=ponder(e)
%Entradas:
%e:excitación residual.
%Salidas:
%x:salida filtrada.
function x=ponder(e)
H=[-134 -374 0 2054 5741 8192 5741 2054 0 -374 -134]/8192;
%coeficientes del filtro.
eaux=[zeros(1,5),e,zeros(1,5)];
for k=7:46
   acum=zeros(1,11);
   for i=1:11
      acum(i)=H(i)*eaux(k+5-i);
   end
   xaux(k)=sum(acum);
end
x=xaux(7:46);
```

Tras este filtrado, el módulo que calcula la subsecuencia de máxima energía es el siguiente.

```
%CODIFICADOR RPE.
%function[Mc,xm]=rpesel(x)
%Entradas:
%x:excitación residual filtrada.
%Salidas:
%Mc:posición RPE codificada.
%xm:subsecuencia de máxima energía.
function[Mc,xm]=rpesel(x)
for m=1:4
   for i=1:13
      xmp(m,i)=x(m+3*(i-1));
   end
end
for m=1:4
   E(m)=sum(xmp(m,:).^2);
end
[var,Mcaux]=max(E);
Mc=Mcaux-1;
xm=xmp(Mc+1,:);
```

Tras el cálculo de los parámetros, el siguiente módulo es el encargado de codificar dichos parámetros.

```
%CODIFICADOR APCM.
%function[xmc,xmaxc]=apcmcod(xm)
%Entradas:
%xm:subsecuencia de máxima energía.
%Salidas:
%xmc:subsecuencia seleccionada codificada.
%xmaxc:valor máximo codificado.
function[xmc,xmaxc]=apcmcod(xm)
xmax=max(abs(xm));
xmaxpc=[31 63 95 127 159 191 223 255 287 319 351 383 415 447 479 511
575 639];
xmaxpc=[xmaxpc,703 767 831 895 959 1023 1151 1279 1407 1535 1663 1791
1919];
xmaxpc=[xmaxpc,2047 2303 2559 2815 3071 3327 3583 3839 4095 4607 5119
5631];
xmaxpc=[xmaxpc,6143 6655 7167 7679 8191 9215 10239 11263 12287 13311
14335];
xmaxpc=[xmaxpc,15359 16389 18431 20479 22527 24575 26623 28671 30719
32767];
xmaxc=sum(xmax>xmaxpc);
if (xmaxc==64)
   xmaxc=63;
end
xmaxp=xmaxpc(xmaxc+1);
xp=xm*((2^15)/xmaxp);
interval=[-24577 -16385 -8193 -1 8191 16383 24575 32767];
for i=1:13;
   xmc(i)=sum(xp(i)>interval);
   if(xmc(i)==8)
      xmc(i)=7;
   end
end
```

Y el siguiente es el encargado de decodificarlos, para reconstruir la excitación residual y eliminarla de la trama actual.

```
%DECODIFICADOR APCM.
%function xmp=apcmdec(xmc,xmaxc)
%Entradas:
%xmc:subsecuencia seleccionada codificada.
%xmaxc:valor máximo codificado.
%Salidas:
%xmp:subsecuencia de máxima energía decodificada.
function xmp=apcmdec(xmc,xmaxc)
interval=[-28672 -20480 -12288 -4096 4096 12288 20480 28672];
for i=1:13
   xm(i)=interval(xmc(i)+1);
end
xmaxpc=[31 63 95 127 159 191 223 255 287 319 351 383 415 447 479 511
575 639];
xmaxpc=[xmaxpc,703 767 831 895 959 1023 1151 1279 1407 1535 1663 1791
1919];
xmaxpc=[xmaxpc,2047 2303 2559 2815 3071 3327 3583 3839 4095 4607 5119
5631];
xmaxpc=[xmaxpc,6143 6655 7167 7679 8191 9215 10239 11263 12287 13311
14335];
xmaxpc=[xmaxpc,15359 16389 18431 20479 22527 24575 26623 28671 30719
32767];
xmax=xmaxpc(xmaxc+1);
```

```
xmp=xm*(xmax/2^15);
```

El último módulo se encarga de, a partir de los parámetros RPE, reconstruir la excitación residual de la trama.

```
%DECODIFICADOR RPE.
%function ep=rpepos(Mc,xmp)
%Entradas:
%Mc:posición RPE codificada.
%xmp:subsecuencia de máxima energía decodificada.
%Salidas:
%ep:excitación residual decodificada.
function ep=rpepos(Mc,xmp)
ep=zeros(1,40);
for i=1:13
   ep(Mc+1+3*(i-1))=xmp(i);
end
```

A continuación, pasamos a ver otro gran módulo, que es el Detector de Actividad Vocal. Tiene un gran número de entradas, de las cuales las más importantes son los coeficientes de autocorrelación, que utiliza para el cálculo de la energía, y los coeficientes Nc del pitch, necesarios para detectar si hay pitch presente en la señal o no. También tiene un gran número de salidas, entre las cuales la más importante es la decisión del VAD. El resto son variables que el algoritmo guarda de una trama a la siguiente.

```
%DETECTOR DE ACTIVIDAD VOCAL.
%function[vadc,rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac]=
vad(ACF,Nc,rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac)
%Entradas:
%ACF:coeficiente de autocorrelación.
%Nc:término de autocorrelación.
%rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac:diversos
parámetros que han de ser guardados de una trama para otra.
%Salidas:
%vadc:variable de decisión del VAD.
%rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac:diversos
parámetros que han de ser guardados de una trama para otra.
function[vadc,rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac]=vad
(ACF,Nc,rvad,ACFvad,av0ant,dm,Nj,oldl,voldl,adp,thv,buc,hac)
pvad=adapfil(ACF,rvad);%filtrado adaptativo.
Nj=[Nj(5),Nc];%guardamos los términos de autocorrelación de las cuatro
tramas anteriores.
ACFvad=[ACFvad(10:36),ACF];%guardamos los ACF de las cuatro tramas
anteriores.
[av0,av1]=ACFav(ACFvad,av0ant);%interpolador de los ACF.
av0ant=[av0ant(10:36),av0];%actualizamos los av0ant.
if(i<=4)
   rav1=zeros(1,9);
else
   rav1=predval(av1);%predictor de valores de autocorrelación.
end
[stat,dm]=speccom(rav1,av0,dm);%comparación espectral.
[ptch,oldl,voldl]=perdec(Nj,oldl,voldl);%detección de periodicidad.
[adp,thv,rvad]=thadap(ACF,ptch,rav1,stat,pvad,adp,thv);%adaptación de
umbral.
vvad=vaddec(pvad,thv);%calculamos la decisión del vad.
[vadc,bc,hc]=vadhang(vvad,buc,hac);%le añadimos un hangover.
```

Veamos detalladamente cada uno de estos módulos que utiliza el VAD. En primer lugar el filtrado adaptativo.

```
%FILTRADO ADAPTATIVO.
%function pvad=adapfil(ACF,rvad)
%Entradas:
%ACF:coeficientes de autocorrelación.
%rvad:coeficientes de reflexión del vad.
%Salidas:
%pvad:potencia filtrada del vad.
function pvad=adapfil(ACF,rvad)
pvad=sum([1 2 2 2 2 2 2 2].*ACF.*rvad);
```

A continuación, el interpolador de los coeficientes de autocorrelación.

```
%INTERPOLADOR DE LOS ACF.
%function [av0,av1]=ACFav(ACFvad,av0ant)
%Entradas:
%ACFvad:ACF de las cuatro tramas anteriores.
%av0ant:av0 de las cuatro tramas anteriores.
%Salidas:
%av0:ACF de las cuatro tramas anteriores interpolados.
%av1:son iguales a los av0ant,para un posterior módulo.
function [av0,av1]=ACFav(ACFvad,av0ant)
i=1:9;
av0(i)=ACFvad(i)+ACFvad(i+9)+ACFvad(i+18)+ACFvad(i+27);
av1(i)=av0ant(i);
```

A continuación, el predictor de valores de autocorrelación.

```
%PREDICTOR DE LOS VALORES DE AUTOCORRELACIÓN.
%function rav1=predval(av1)
%Entradas:
%av1:av0 de las cuatro tramas anteriores.
%Salidas:
%rav1:predicción de los valores de autocorrelación.
function rav1=predval(av1)
R=[av1(1:8);av1(2),av1(1:7);av1(3),av1(2),av1(1:6);av1(4),av1(3),
av1(2),av1(1:5)];
R=[R;av1(5),av1(4),av1(3),av1(2),av1(1:4);av1(6),av1(5),av1(4),av1(3),
av1(2),av1(1:3)];
R=[R;av1(7),av1(6),av1(5),av1(4),av1(3),av1(2),av1(1:2)];
R=[R;av1(8),av1(7),av1(6),av1(5),av1(4),av1(3),av1(2),av1(1)];
aav1=inv(R)*av1(2:9)';
aav1=[-1;aav1(1:8)];
for i=1:9
   sum=0;
   for k=1:(10-i)
      sum=sum+(aav1(k)*aav1(k+i-1));
   end
   rav1(i)=sum;
end
```

Veamos ahora el módulo de la comparación espectral.

```
%COMPARACIÓN ESPECTRAL.
%function [stat,dm]=speccom(rav1,av0,lastdm)
%Entradas:
%rav1:predicción de los valores de autocorrelación.
%av0:ACF de las cuatro tramas anteriores interpolados.
%lastdm:distorsión media anterior.
%Salidas:
%stat:valor booleano que indica un estado.
%dm:distorsión media.
function [stat,dm]=speccom(rav1,av0,lastdm)
thresh=0.05;
dm=sum([1 2 2 2 2 2 2 2].*rav1.*av0)/av0(1);
difference=abs(dm-lastdm);
if(difference<thresh)
   stat=1;
else
   stat=0;
end
```

Y a continuación, el módulo que decide si hay pitch o no en la trama.

```
%DETECCIÓN DE PERIODICIDAD.
%function
[ptch,oldlagcount,veryoldlagcount]=perdec(Nj,oldlagcount,veryoldlagcou
nt)
%Entradas:
%Nj:términos de autocorrelación de las cuatro tramas anteriores.
%oldlagcount:contador para variables Nj.
%veryoldlagcount:valor anterior del contador para variables Nj.
%Salidas:
%ptch:variable que indica si hay periodicidad o no.
%oldlagcount:contador para variables Nj.
%veryoldlagcount:valor anterior del contador para variables Nj.
function
[ptch,oldlagcount,veryoldlagcount]=perdec(Nj,oldlagcount,veryoldlagcou
nt)
lthresh=2;
nthresh=4;
if(oldlagcount+veryoldlagcount>=nthresh)
   ptch=1;
else
   ptch=0;
end
lagcount=0;
for i=1:4
   smallag=mod(max(Nj(i+1),Nj(i)),min(Nj(i+1),Nj(i)));
   if(min(smallag,min(Nj(i+1),(Nj(i)-smallag)))<lthresh)
      lagcount=lagcount+1;
   end
   veryoldlagcount=oldlagcount;
   oldlagcount=lagcount;
end
```

Veamos ahora un módulo muy importante, dentro del VAD. Es el encargado de la adaptación del umbral de potencia con el cual hay que comparar para decidir si hay una trama de voz o de silencio.

```
%ADAPTACIÓN DE UMBRAL.
%function
[adaptcount,thvad,rvad]=thadap(ACF,ptch,rav1,stat,pvad,adaptcount,thva
d)
%Entradas:
%ACF:coeficientes de autocorrelación.
%ptch:variable que indica si hay periodicidad o no.
%rav1:predicción de los valores de autocorrelación.
%stat: detector de estacionaridad.
%pvad:potencia filtrada del vad.
%adapcount:contador para la adaptación.
%thvad:potencia umbral de decisión.
%Salidas:
%adapcount:contador para la adaptación.
%thvad:potencia umbral de decisión.
%rvad:coeficientes de reflexión del vad.
function
[adaptcount,thvad,rvad]=thadap(ACF,ptch,rav1,stat,pvad,adaptcount,thva
d)
pth=300000;
plev=800000;
fac=3;
adp=8;
inc=16;
dec=32;
margin=80000000;
rvad=[6 -4 1 0 0 0 0 0 0];
if(ACF(1)<pth)
   thvad=plev;
   return
end
if(stat==1 & ptch==0)
   adaptcount=adaptcount+1;
else
   adaptcount=0;
   return
end
if(adaptcount<=adp)
   return
end
thvad=thvad-(thvad/dec);
if(thvad<(pvad*fac))
   thvad=min(thvad-(thvad/dec),pvad*fac);
end
if(thvad>pvad+margin)
   thvad=pvad+margin;
end
rvad=rav1;
adaptcount=adp+1;
```

A continuación, el módulo que calcula la decisión del VAD.

```
%DECISIÓN DEL VAD.
%function vvad=vaddec(pvad,thvad)
%Entradas:
%pvad:potencia filtrada del vad.
%thvad:potencia umbral de decisión.
%Salidas:
%vvad:decisión del vad.
function vvad=vaddec(pvad,thvad)
if(pvad>thvad)
   vvad=1;
else
   vvad=0;
end
```

Y por último el módulo de la adición de hangover.

```
%ADICIÓN DE HANGOVER.
%function
[vad,burstcount,hangcount]=vadhang(vvad,burstcount,hangcount)
%Entradas:
%vvad:decisión del vad.
%burstcount:contador de tramas.
%hangcount:contador de hangover.
%Salidas:
%vad:decisión del vad con hangover.
%burstcount:contador de tramas.
%hangcount:contador de hangover.
function [vad,burstcount,hangcount]=vadhang(vvad,burstcount,hangcount)
burstconst=3;
hangconst=5;
if (vvad==1)
   burstcount=burstcount+1;
else
   burstcount=0;
end
if(burstcount>=burstconst)
   hangcount=hangconst;
   burstcount=burstconst;
end
if(hangcount>=0);
   vad=1;
else
   vad=vvad;
end
if(hangcount>=0)
   hangcount=hangcount-1;
end
```

Pasamos ahora a ver otro gran bloque. Es el codificador de tramas de silencio. Tiene un funcionamiento bastante sencillo. Utiliza los valores de decisión de la trama actual y el de las tres tramas anteriores. Si los cuatro son cero, pone una bandera a valor alto indicando que lo que se envía es una trama de silencio, y modifica algunos parámetros de la trama. Si no son cero los cuatro valores de decisión, los actualiza y no modifica dichos parámetros.

```
%TRAMAS SID.
%function [LARc,Nc,bc,Mc,xmaxc,xmc,flag,vad2,vad3,vad4,LARsid,xmaxsid]
=sid(LARc,Nc,bc,Mc,xmaxc,xmc,vad1,vad2,vad3,vad4,LARsid,xmaxsid)
%Entradas:
%LARc:LAR codificados.
%Nc:término de autocorrelación codificado.
%bc:ganancia de autocorrelación codificado.
%Mc:posición RPE codificada.
%xmaxc:valor máximo RPE codificado.
%xmc:subsecuencia seleccionada codificada.
%vad1,vad2,vad3,vad4:decisión del vad para las cuatro tramas
anteriores.
%LARsid:LARc de las cuatro tramas anteriores.
%xmaxsid:xmaxc de las cuatro tramas anteriores.
%Salidas:
%LARc:LAR codificados aplicado el sid.
%Nc:término de autocorrelación codificado aplicado el sid.
%bc:ganancia de autocorrelación codificado aplicado el sid.
%Mc:posición RPE codificada aplicado el sid.
%xmaxc:valor máximo RPE codificado aplicado el sid.
%xmc:subsecuencia seleccionada codificada aplicado el sid.
%flag:variable que indica si es una trama sid o no.
%vad2,vad3,vad4:decisión del vad de las tres tramas anteriores.
function
[LARc,Nc,bc,Mc,xmaxc,xmc,flag,vad2,vad3,vad4,LARsid,xmaxsid]=sid(LARc,
Nc,bc,Mc,xmaxc,xmc,vad1,vad2,vad3,vad4,LARsid,xmaxsid)
LARsid=[LARsid(9:32),LARc];%actualizamos los LARsid para la siguiente
trama.
xmaxsid=[xmaxsid(5:16),xmaxc];%actualizamos los xmaxsid para la
siguiente trama.
flag=0;
if(vad1==0 & vad2==0 & vad3==0 & vad4==0)%comprobamos si hay que
aplicar el sid.
   flag=1;%ponemos esa bandera a uno indicando que es una trama sid.
   j=1:8;%sustituimos los LARc por una media ponderada.
   LARc(j)=round((LARsid(j)+LARsid(j+8)+LARsid(j+16)+LARsid(j+24))/4);
   j=1:4;%sustituimos los xmaxc por una media ponderada.
   xmaxc(j)=round(sum(xmaxsid)/16);
   Nc=zeros(1,4);%los demás parámetros los ponemos a cero.
   bc=zeros(1,4);
   Mc=zeros(1,4);
   xmc=zeros(1,52);
end
vad2=vad1;%actualizamos los valores de decisión.
vad3=vad2;
vad4=vad3;
```

Pasamos ahora al último gran bloque encargado de la codificación, antes del envío de las tramas por el canal radio. Es el codificador de canal. Recibe como entrada los parámetros de la trama, y devuelve a su salida cuatro tramas con dichos parámetros codificados. Además va guardando y enviando bits de las tramas anteriores, puesto que los parámetros de una trama, se envían en cuatro ráfagas.

```
%CODIFICACIÓN DE CANAL.
%function
[tr1,tr2,tr3,tr4,t5a,t6a,t7a,t8a]=codcanal(LARc,Nc,bc,Mc,xmaxc,xmc,t5a
,t6a,t7a,t8a)
%Entradas:
%LARc:LAR codificados.
%Nc:términos de autocorrelación codificados.
%bc:ganancias de autocorrelación codificados.
%Mc:posiciones RPE codificadas.
%xmaxc:valores máximos RPE codificados.
%xmc:subsecuencia seleccionada codificada.
%t5a,t6a,t7a,t8a:bloques del 5 al 8 de la trama anterior.
%Salidas:
%tr1,tr2,tr3,tr4:las cuatro tramas de salida.
%t5a,t6a,t7a,t8a:bloques del 5 al 8 para la siguiente trama.
function [tr1,tr2,tr3,tr4,t5a,t6a,t7a,t8a]=
codcanal(LARc,Nc,bc,Mc,xmaxc,xmc,t5a,t6a,t7a,t8a)
a=bitcod(LARc,Nc,bc,Mc,xmaxc,xmc);%codificamos en bits todos los
parámetros.
b=reord(a);%aplicamos una reordenación.
c=crc5350(b(1:50));%aplicamos un código cíclico.
d=entrelazado(b(1:182),c);%aplicamos un entrelazado.
e=crc215(d);%aplizamos un código convolucional.
f=[e,b(183:260)];%formamos los 456 bits de salida.
[t1,t2,t3,t4,t5,t6,t7,t8]=partic(f);%aplicamos un particionamiento.
tt1=entdiagb(t1,t5a);%aplicamos un entrelazado diagonal de bloques.
tt2=entdiagb(t2,t6a);
tt3=entdiagb(t3,t7a);
tt4=entdiagb(t4,t8a);
t5a=t5;%actualizamos los bloques 5 al 8 para la siguiente trama.
t6a=t6;
t7a=t7;
t8a=t8;
tr1=enttrama(tt1,1);%aplicamos un entrelazado de trama.
tr2=enttrama(tt2,2);
tr3=enttrama(tt3,3);
tr4=enttrama(tt4,4);
```

Veamos cada uno de estos módulos a los cuales llama. El primero es el encargado de codificar los parámetros en bits.

```
%CODIFICADOR EN BITS.
%function a=bitcod(LARc,Nc,bc,Mc,xmaxc,xmc)
%Entradas:
%LARc:LAR codificados.
%Nc:términos de autocorrelación codificados.
%bc:ganancias de autocorrelación codificados.
%Mc:posiciones RPE codificadas.
%xmaxc:valores máximos RPE codificados.
%xmc:subsecuencia seleccionada codificada.
%Salidas:
%a:260 bits de salida de los parámetros codificados.
function a=bitcod(LARc,Nc,bc,Mc,xmaxc,xmc)
bits=[6 6 5 5 4 4 3 3];%el peso de codificación de cada LAR es
distinto.
if(LARc(1)>=0)
   a=dec2bin(LARc(1),bits(1));
else
   a=dec2bin(bitcmp(abs(LARc(1)),bits(1))+1,bits(1));
end
for i=2:8;
   if(LARc(i)>=0)
      a=[a,dec2bin(LARc(i),bits(i))];
   else
      a=[a,dec2bin(bitcmp(abs(LARc(i)),bits(i))+1,bits(i))];
   end
end
for i=1:4
   a=[a,dec2bin(Nc(i),7)];
   a=[a,dec2bin(bc(i),2)];
   a=[a,dec2bin(Mc(i),2)];
   a=[a,dec2bin(xmaxc(i),6)];
   for j=1:13
      a=[a,dec2bin(xmc(13*(i-1)+j),3)];
   end
end
```

El siguiente módulo aplica una reordenación a los bits según su importancia.

```
%REORDENACIÓN.
%function b=reord(a)
%Entradas:
%a:260 bits de salida de los parámetros codificados.
%Salidas:
%b:260 bits reordenados por su importancia.
function b=reord(a)
r=[6 53 109 165 221  5 12  17 4 11 16 22 43 99
155 211  52 108 164 220 10 26 30  42  98];
r=[r,154 210 41 97  153 209 40 96 152  208 39
95 151 207  51 107 163 219 3 21 33 38 94 ];
r=[r,150 206 25 29  45 101 157 213 37 93 149
205 47  103 159  215 2 9 15 36 20 24 32 44];
r=[r,100 156 212 50  106 162 218 56 59 62 65
68 71 74 77 80 83 86 89 92 112 115 118 121];
r=[r, 124 127 130 133 136  139 142 145 148
168 171 174 177 180 183 186 189 192 195 198 ];
r=[r,201 204 224  227  230 233  236 239 242
245 248 251 254  257 260 46 102 158 214 49 ];
r=[r,105 161 217 55 58 61 64  67 70 73  76 79
82 85 88  91 111  114 117 120 123 126 129];
r=[r,132 135 138 141 144 147 167 170  173 176
179 182 185 188 191 194 197 200 203   223];
r=[r,226 229 232 235 238 241 244 247 250 253
256 259 1 8 14 28 3 35  34 13 19 18 23 48 ];
r=[r,104 160 216 54 57 60 63 66 69 72 75 78
81 84 87 90 110 113 116 119 122 125 128 131];
r=[r,134 137 140 143 146 166 169 172 175 178
181  184 187 190  193 196 199 202  222 225];
r=[r,228 231 234 237 240 243 246 249 252 255 258 7 27];
i=1:260;
b(i)=a(r(i));
```

El siguiente módulo es el encargado del código cíclico.

```
%CÓDIGO CÍCLICO.
%function p=crc5350(Ia)
%Entradas:
%Ia:50 bits de la clase Ia, la de mayor importancia.
%Salidas:
%p:tres bits de paridad.
function p=crc5350(Ia)
r=[0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 0 0;0 1 1;1  0 0;1 0 0;0 0
1;0 0  1;1 1 1;1 1 0];
r=[r;0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 1 0;0 1 1;1 0 0;0 1 0;0 0
1;1 0 1;1 1 1;1 1 0];
r=[r;0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 1 0;0 1 1;1 0 0;0 1 0;0 0
1;1 0 1;1 1 1;1 1 0];
r=[r;0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 1 0;0 1 1];
b=diag(ones(1,50),0);
for i=1:50
   for j=1:3
      b(i,j+50)=r(i,j);
   end
end
paux=Ia*b;
paux=mod(paux,2);
p=[paux(51),paux(52),paux(53)];
```

```
p=dec2bin(p,1)';
```

A continuación, veamos el bloque encargado del entrelazado.

```
%ENTRELAZADO.
%function d=entrelazado(b,c)
%Entradas:
%b:182 primeros bits de salida ordenados por su importancia.
%c:tres bits de paridad del código cíclico.
%Salidas:
%d:189 bits de salida tras el entrelazado.
function d=entrelazado(b,c)
i=1:91;
d(i)=b(2*i-1);
d(92)=c(1);
d(93)=c(2);
d(94)=c(3);
i=95:185;
d(i)=b(372-2*i);
d(186)='0';
d(187)='0';
d(188)='0';
d(189)='0';
```

Y ahora el módulo que realiza el código convolucional.

```
%CÓDIGO CONVOLUCIONAL.
%function c=crc215(u)
%Entradas:
%u:189 bits de la clase Ia y Ib entrelazados y con código cíclico.
%Salidas:
%c:378 bits de salida tras el código convolucional.
function c=crc215(u)
ant=['0' '0' '0' '0'];
for i=1:189
   reg=[u(i),ant];
   g(2*i-1)=mod((reg(1)+reg(4)+reg(5)),2);
   g(2*i)=mod((reg(1)+reg(2)+reg(4)+reg(5)),2);
   ant=reg(1:4);
end
c=dec2bin(g)';
```

Veamos el bloque encargado del particionamiento.

```
%PARTICIONAMIENTO
%function [t1,t2,t3,t4,t5,t6,t7,t8]=partic(f)
%Entradas:
%f:456 bits resultantes de la reordenación,código cíclico,entrelazado
y código convolucional.
%Salidas:
%t1,t2,t3,t4,t5,t6,t7,t8:8 bloques de 57 bits.
function [t1,t2,t3,t4,t5,t6,t7,t8]=partic(f)
t1=f(1:57);
t2=f(58:114);
t3=f(115:171);
t4=f(172:228);
t5=f(229:285);
t6=f(286:342);
t7=f(343:399);
```

```
t8=f(400:456);
```

Veamos el bloque que realiza el entrelazado diagonal de bloques.

```
%ENTRELAZADO DIAGONAL DE BLOQUES.
%function tt=entdiagb(t,ta)
%Entradas:
%t:bloque de la trama actual.
%ta:bloque de la trama anterior.
%Salidas:
%tt:114 bits de salida correspondiente a dos bloques.
function tt=entdiagb(t,ta)
tt=[t,ta];
```

Y por último el encargado del entrelazado de trama.

```
%ENTRELAZADO DE TRAMA.
%function tr=enttrama(tt,n)
%Entradas:
%tt:ráfaga de 114 bits.
%n:número de la ráfaga dentro de la trama.
%Salidas:
%tr:ráfaga entrelazada.
function tr=enttrama(tt,n)
for i=57*(n-1):57*n-1
   if(mod(i,8)<=3)
      j=2*mod(49*i,57);
      tr(j+1)=tt(i+58-57*n);
   else
      j=2*mod(49*i,57)+1;
      tr(j+1)=tt(i+58-57*n);
   end
end
for i=57*(n+3):57*(n+4)-1
   if(mod(i,8)<=3)
      j=2*mod(49*i,57);
      tr(j+1)=tt(i-113-57*n);
   else
      j=2*mod(49*i,57)+1;
      tr(j+1)=tt(i-113-57*n);
   end
end
```

Pasamos ahora al módulo que simula el canal de transmisión. Sus entradas son las cuatro ráfagas y la BER y sus salidas son las cuatro ráfagas tras pasar por el canal.

```
%CANAL DE TRANSMISIÓN.
%function [tr1,tr2,tr3,tr4]=canal(tr1,tr2,tr3,tr4,BER)
%Entradas:
%tr1,tr2,tr3,tr4:tramas antes de pasar por el canal.
%BER:tasa de error de bit en el canal.
%Salidas:
%tr1,tr2,tr3,tr4:tramas tras pasar por el canal.
function [tr1,tr2,tr3,tr4]=canal(tr1,tr2,tr3,tr4,BER)
tr1=canalt(tr1,BER);
tr2=canalt(tr2,BER);
tr3=canalt(tr3,BER);
tr4=canalt(tr4,BER);
```

Este módulo realiza cuatro llamadas al módulo siguiente, una para cada ráfaga.

```
%CANAL
%function bitsr=canalt(bitst,BER)
%Entradas:
%bitst:bits de la trama antes de pasar por el canal.
%BER:tasa de error de bit del canal.
%Salidas:
%bitsr:bits de la trama tras pasar por el canal.
function bitsr=canalt(bitst,BER)
pos=(rand(1,114)<BER);
for i=1:114
   if(pos(i)==1)
      if(bitst(i)=='1')
         bitsr(i)='0';
      else
         bitsr(i)='1';
      end
   else
      bitsr(i)=bitst(i);
   end
end
```

Una vez que tenemos las cuatro ráfagas recibidas, pasamos a la decodificación. Lo primero sería la decodificación de canal. Veamos este módulo.

```
%DECODIFICACIÓN DE CANAL.
%function
[LARc,Nc,bc,Mc,xmaxc,xmc,t1a,t2a,t3a,t4a]=descodcanal(tr1,tr2,tr3,tr4,
t1a,t2a,t3a,t4a)
%Entradas:
%tr1,tr2,tr3,tr4:las cuatro tramas de entrada.
%t1a,t2a,t3a,t4a:bloques del 1 al 4 de la anterior trama recibida.
%Salidas:
%LARc:LAR codificados.
%Nc:términos de autocorrelación codificados.
%bc:ganancias de autocorrelación codificados.
%Mc:posiciones RPE codificadas.
%xmaxc:valores máximos RPE codificados.
%xmc:subsecuencia seleccionada codificada.
%t1a,t2a,t3a,t4a:bloques del 1 al 4 para la siguiente trama.
function
[LARc,Nc,bc,Mc,xmaxc,xmc,t1a,t2a,t3a,t4a]=descodcanal(tr1,tr2,tr3,tr4,
t1a,t2a,t3a,t4a)
tt1=desenttrama(tr1,1);%efectuamos el desentrelazado de trama.
tt2=desenttrama(tr2,2);
tt3=desenttrama(tr3,3);
tt4=desenttrama(tr4,4);
[t1,t5a]=desentdiagb(tt1);%desentrelazado diagonal de bloques.
[t2,t6a]=desentdiagb(tt2);
[t3,t7a]=desentdiagb(tt3);
[t4,t8a]=desentdiagb(tt4);
f=despartic(t1a,t2a,t3a,t4a,t5a,t6a,t7a,t8a);%desparticionamiento.
t1a=t1;%actualizamos los bloques.
t2a=t2;
t3a=t3;
t4a=t4;
```

```
e=f(1:378);
d=incrc215(e);%código convolucional inverso.
[b,c]=desentrelazado(d);%desentrelazado.
b=[b,f(379:456)];
nerror=incrc5350(b(1:50),c);%comprobamos si ha habido error con el
código cíclico.
a=desreord(b);%desreordenación.
[LARc,Nc,bc,Mc,xmaxc,xmc]=bitdec(a);%decodificamos los bits recibidos.
```

Vemos que este módulo es parecido al de la codificación de canal, pero inverso. Veamos cada uno de estos submódulos que lo componen. Primeramente, el módulo encargado del desentrelazado de trama.

```
%DESENTRELAZADO DE TRAMA.
%function tt=desenttrama(tr,n)
%Entradas:
%tr:ráfaga entrelazada.
%n:número de la ráfaga dentro de la trama.
%Salidas:
%tt:114 bits de salida.
function tt=desenttrama(tr,n)
for i=57*(n-1):57*n-1
   if(mod(i,8)<=3)
      j=2*mod(49*i,57);
      tt(i+58-57*n)=tr(j+1);
   else
      j=2*mod(49*i,57)+1;
      tt(i+58-57*n)=tr(j+1);
   end
end
for i=57*(n+3):57*(n+4)-1
   if(mod(i,8)<=3)
      j=2*mod(49*i,57);
      tt(i-113-57*n)=tr(j+1);
   else
      j=2*mod(49*i,57)+1;
      tt(i-113-57*n)=tr(j+1);
   end
end
```

A continuación, el que quita el entrelazado diagonal de bloque.

```
%DESENTRELAZADO DIAGONAL DE BLOQUES.
%function [t,ta]=desentdiagb(tt)
%Entradas:
%tt:114 bits de entrada correspondientes a dos bloques.
%Salidas:
%t:bloque de la trama actual.
%ta:bloque de la trama anterior.
function [t,ta]=desentdiagb(tt)
t=tt(1:57);
ta=tt(58:114);
```

El siguiente es el que quita el particionamiento.

```
%DESPARTICINAMIENTO.
%function f=despartic(t1a,t2a,t3a,t4a,t5a,t6a,t7a,t8a)
%Entradas:
%t1a,t2a,t3a,t4a,t5a,t6a,t7a,t8a:bloques de la trama actual.
%Salidas:
%f:456 bits de salida.
function f=despartic(t1a,t2a,t3a,t4a,t5a,t6a,t7a,t8a)
f=[t1a,t2a,t3a,t4a,t5a,t6a,t7a,t8a];
```

A continuación, el módulo encargado de decodificar el código convolucional.

```
%CÓDIGO CONVOLUCIONAL INVERSO.
%function u=incrc215(c)
%Entradas:
%c:378 bits de entrada correspondientes a las clases Ia y Ib.
%Salidas:
%u:189 bits de salida tras el código convolucional inverso.
function u=incrc215(c)
matcrc=[0 3 13 14 52 55 57 58;48 51 61 62 4 7 9 10;60 63 49 50 8 11 5
6;12 15 1 2 56 59 53 54];
matcrc=[matcrc;15 12 2 1 59 56 54 53;63 60 50 49 11 8 6 5;51 48 62  61
7  4 10  9];
matcrc=[matcrc;3 0 14 13 55 52 58 57;19 16 30 29 39 36 42 41;35 32 46
45 23  20 26  25];
matcrc=[matcrc;47 44 34 33 27 24 22 21;31 28 18 17 43 40 38 37;28 31
17 18 40 43 37  38];
matcrc=[matcrc;44 47 33 34 24 27 21 22;32 35 45 46 20 23 25 26;16 19
29 30 36 39 41  42];
reg=['0' '0' '0' '0'];
for n=1:185
   aux=c(2*n-1:2*n+4);
   crc=zeros(1,8);
   for i=1:8;
      crc(i)=sum(mod(aux+dec2bin(matcrc(bin2dec(reg)+1,i),6),2));
   end
      [q,p]=min(crc);
      aux3=dec2bin((p-1),3);
      u(n)=aux3(1);
      reg=[aux3(1),reg(1:3)];
end
clear matcrc;
u(186:189)=['0' '0' '0' '0'];
```

El siguiente es el encargado del desentrelazado.

```
%DESENTRELAZADO.
%function [b,c]=desentrelazado(d)
%Entradas:
%d:189 bits de entrada.
%Salidas:
%b:182 bits de salida ordenados por su importancia.
%c:tres bits de paridad del código cíclico.
function [b,c]=desentrelazado(d)
i=1:91;
b(2*i-1)=d(i);
c(1)=d(92);
c(2)=d(93);
c(3)=d(94);
i=95:185;
b(372-2*i)=d(i);
```

A continuación, el que quita el código cíclico.

```
%CÓDIGO CÍCLICO INVERSO.
%function nerror=incrc5350(d,p)
%Entradas:
%d:50 bits de la clase Ia.
%p:tres bits de paridad.
%Salidas:
%nerror:variable que indica si hay error detectado.
function nerror=incrc5350(d,p)
aux=[d,p];
a=[0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 0 0;0 1 1;1 0 0;1 0 0;0 0 1;0
0 1;1 1  1;1 1  0];
a=[a;0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 1 0;0 1 1;1 0 0;0 1 0;0 0
1;1 0 1;1 1 1;1 1 0];
a=[a;0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 1 0;0 1 1;1 0 0;0 1 0;0 0
1;1 0 1;1 1 1;1 1 0];
a=[a;0 1 1;1 0 0;0 1 0;0 0 1;1 0 1;1 1 1;1 1 0;0 1 1];
Ht=[a;1 0 0;0 1 0;0 0 1];
nerror=aux*Ht;
nerror=mod(nerror,2);
```

Posteriormente, veamos el módulo encargado de suprimir la reordenación aplicada en el codificador.

```
%DESREORDENACIÓN.
%function a=desreord(b)
%Entradas:
%b:260 bits de entrada ordenados según su importancia.
%Salidas:
%a:260 bits de salida desreordenados.
function a=desreord(b)
r=[6 53 109 165 221  5 12  17 4 11 16 22 43
99 155 211  52 108 164 220 10 26 30  42  98];
r=[r,154 210 41 97  153 209 40 96 152  208 39
 95 151 207  51 107 163 219 3 21 33 38 94 ];
r=[r,150 206 25 29  45 101 157 213 37 93 149
 205 47  103 159  215 2 9 15 36 20 24 32 44];
```

```
r=[r,100 156 212 50  106 162 218 56 59 62 65
 68 71 74 77 80 83 86 89 92 112 115 118 121];
r=[r, 124 127 130 133 136  139 142 145 148 168
 171 174 177 180 183 186 189 192 195 198 ];
r=[r,201 204 224  227  230 233  236 239 242 245
 248 251 254  257 260 46 102 158 214 49 ];
r=[r,105 161 217 55 58 61 64  67 70 73  76 79
82 85 88  91 111  114 117 120 123 126 129];
r=[r,132 135 138 141 144 147 167 170  173 176
179 182 185 188 191 194 197 200 203   223];
r=[r,226 229 232 235 238 241 244 247 250 253
256 259 1 8 14 28 3 35  34 13 19 18 23 48 ];
r=[r,104 160 216 54 57 60 63 66 69 72 75 78
81 84 87 90 110 113 116 119 122 125 128 131];
r=[r,134 137 140 143 146 166 169 172 175 178
181  184 187 190  193 196 199 202  222 225];
r=[r,228 231 234 237 240 243 246 249 252 255 258 7 27];
i=1:260;
a(r(i))=b(i);
```

Y, por último, para finalizar, veamos el módulo encargado de decodificar los bits recibidos y reconstruir los parámetros de la trama.

```
%DECODIFICADOR EN BITS.
%function [LARc,Nc,bc,Mc,xmaxc,xmc]=bitdec(a)
%Entradas:
%a:260 bits de entrada de los parámetros codificados.
%Salidas:
%LARc:LAR codificados.
%Nc:términos de autocorrelación codificados.
%bc:ganancias de autocorrelación codificados.
%Mc:posiciones RPE codificadas.
%xmaxc:valores máximos RPE codificados.
%xmc:subsecuencia seleccionada codificada.
function [LARc,Nc,bc,Mc,xmaxc,xmc]=bitdec(a)
bits=[6 6 5 5 4 4 3 3];
pos=[1 7 13 18 23 27 31 34 37];
for i=1:8
   aux=a(pos(i):pos(i+1)-1);
   if(aux(1)=='1')
      LARc(i)=-bitcmp(bin2dec(aux)-1,bits(i));
   else
      LARc(i)=bin2dec(aux);
   end
end
posN=37;
posb=44;
posM=46;
posxm=48;
posx=[54 57 60 63 66 69 72 75 78 81 84 87 90];
for i=1:4
   aux=a(posN+56*(i-1):posN+56*(i-1)+6);
   Nc(i)=bin2dec(aux);
   aux=a(posb+56*(i-1):posb+56*(i-1)+1);
   bc(i)=bin2dec(aux);
   aux=a(posM+56*(i-1):posM+56*(i-1)+1);
   Mc(i)=bin2dec(aux);
```

```
   aux=a(posxm+56*(i-1):posxm+56*(i-1)+5);
   xmaxc(i)=bin2dec(aux);



   for j=1:13
      aux=a(posx(j)+56*(i-1):posx(j)+56*(i-1)+2);
      xmc(13*(i-1)+j)=bin2dec(aux);
   end
end
```

Veamos ahora otro módulo importante. Su finalidad es la de decodificar las tramas de silencio. Si detecta la bandera a nivel alto indicando que es una trama de silencio, entonces modifica algunos parámetros de la trama y los sustituye por una serie de valores aleatorios. En caso contrario, no actúa.

```
%DECODIFICADOR DE TRAMAS SID.
%function [LARc,Nc,bc,Mc,xmaxc,xmc]=
dessid(LARc,Nc,bc,Mc,xmaxc,xmc,flag)
%Entradas:
%LARc:LAR codificados aplicado el sid.
%Nc:término de autocorrelación codificado aplicado el sid.
%bc:ganancia de autocorrelación codificado aplicado el sid.
%Mc:posición RPE codificada aplicado el sid.
%xmaxc:valor máximo RPE codificado aplicado el sid.
%xmc:subsecuencia seleccionada codificada aplicado el sid.
%flag:variable que indica si es una trama sid o no.
%Salidas:
%LARc:LAR codificados.
%Nc:término de autocorrelación codificado.
%bc:ganancia de autocorrelación codificado.
%Mc:posición RPE codificada.
%xmaxc:valor máximo RPE codificado.
%xmc:subsecuencia seleccionada codificada.
function
[LARc,Nc,bc,Mc,xmaxc,xmc]=dessid(LARc,Nc,bc,Mc,xmaxc,xmc,flag)
if(flag==1)
   Nc=[40,120,40,120];%se sustituye por esos valores.
   bc=zeros(1,4);%se sustituye por esos valores.
   Mc=round(3*abs(rand(1,4)));%se sustituye por esos valores.
   xmc=round(5*abs(rand(1,52)))+1;%se sustituye por esos valores.
end
```

Para finalizar con este capítulo, veamos la estructura del decodificador GSM. Debido a que utiliza gran parte de los módulos que también los utiliza el codificador, alguno de ellos no los incluiremos, comentando sólo los módulos nuevos. Veamos en primer lugar la estructura del decodificador.

```
%DECODIFICADOR GSM.
%function
[sp,Nact,dpd,LARantd]=gsmdec(LARc,Nc,bc,Mc,xmaxc,xmc,Nact,dpd,LARantd)
%Entradas:
%LARc:LAR codificado.
%Nc:término de correlación codificado.
%bc:ganancia codificada.
%Mc:posición RPE codificada.
%xmaxc:valor máximo codificado.
%xmc:subsecuencia seleccionada codificada.
%Nact:valor de Nc de la trama anterior.
%dpd:120 muestras anteriores reconstruidas.
%LARantd:LAR de la trama anterior.
%Salidas:
%sp:trama de 160 muestras decodificada.
%Nact:valor de Nc de la trama actual.
%dpd:120 muestras anteriores reconstruidas.
%LARantd:LAR de la trama actual.
function
[sp,Nact,dpd,LARantd]=gsmdec(LARc,Nc,bc,Mc,xmaxc,xmc,Nact,dpd,LARantd)
for n=1:4%efectuamos el procesado en cuatro subtramas.
   e=rped(Mc(n),xmaxc(n),xmc(13*n-12:13*n));%decodificación RPE.
   if(Nc(n)<40 | Nc(n)>120)%comprobamos que Nc esté dentro del rango.
      Nc(n)=Nact;
   end
   Nact=Nc(n);
   dp=ltpd(Nc(n),bc(n),dpd);%decodificación LTP.
   d(40*n-39:40*n)=e+dp;
   dpd=[dpd(41:120),d(40*n-39:40*n)];
end
[s,LARantd]=stpd(LARc,LARantd,d);%decodificación STP.
sp=postproc(s);%aplicamos un postprocesado.
```

Veamos cada una de las cuatro secciones de decodificación. En primer lugar, la sección de decodificación RPE, que reconstruye la excitación residual de la trama, a partir de la cual se le irán sumando el pitch y la envolvente para reconstruir la trama.

```
%DECODIFICADOR RPE.
%function e=rped(Mc,xmaxc,xmc)
%Entradas:
%Mc:posición RPE codificada.
%xmaxc:valor máximo codificado.
%xmc:subsecuencia seleccionada codificada.
%Salidas:
%e:excitación residual decodificada.
function e=rped(Mc,xmaxc,xmc)
xmp=apcmdec(xmc,xmaxc);%decodificamos los términos RPE.
e=rpepos(Mc,xmp);%calculamos la excitación residual decodificada.
```

A continuación, la sección de decodificación del pitch.

```
%DECODIFICADOR LTP.
%function dp=ltpd(Nc,bc,dpd)
%Entradas:
%Nc:término de autocorrelación codificado.
%bc:ganancia codificada.
%dpd:120 muestras anteriores recontruidas.
%Salidas:
%dp:señal sin envolvente de salida.
function dp=ltpd(Nc,bc,dpd)
[N,b]=ltpdec(Nc,bc);%decodificamos los coeficientes LTP.
dp=b*dpd((121-N):(160-N));%calculamos el pitch de la señal.
```

Veamos ahora la sección de decodificación de la envolvente del espectro.

```
%DECODIFICADOR STP:
%function [s,LARantd]=stpd(LARc,LARantd,d)
%Entradas:
%LARc:LAR codificados.
%LARantd:LAR de la trama anterior.
%d:señal sin envolvente.
%Salidas:
%s:trama de 160 muestras.
%LARantd:LAR actuales para la siguiente trama.
function [s,LARantd]=stpd(LARc,LARantd,d)
LARact=lardec(LARc);%decodificamos los LAR.
LARi=interpol(LARact,LARantd);%interpolamos los LAR.
rp=reflect(LARi);%calculamos los coeficientes de reflexión.
s=synfilt(d,rp);%aplicamos el filtro que restaura la envolvente.
LARantd=LARact;%actualizamos los LAR anteriores.
```

Aquí aparece un nuevo módulo. En el codificador había un filtro que eliminaba la envolvente. Aquí, en el decodificador, aparece el filtro que restaura la envolvente. Veámoslo.

```
%FILTRO QUE RESTAURA LA ENVOLVENTE.
%function srp=synfilt(drp,rrp)
%Entradas:
%drp:señal sin envolvente.
%rrp:coeficientes de reflexión.
%Salidas:
%srp:señal con envolvente.
function srp=synfilt(drp,rrp)
vr=zeros(1,8);
for k=1:160
   if(k<14)
      rrpaux=rrp(1:8);
   elseif(k<28)
      rrpaux=rrp(9:16);
   elseif(k<41)
      rrpaux=rrp(17:24);
   else
      rrpaux=rrp(25:32);
```

```
   end
   sr(1)=drp(k);


   for j=2:9
      sr(j)=sr(j-1)-rrpaux(10-j)*vr(10-j);
      v(11-j)=vr(10-j)+rrpaux(10-j)*sr(j);
   end
   srp(k)=sr(9);
   v(1)=sr(9);
   vr=v;
end
```

Veamos la última sección, que es la sección de postprocesado.

```
%POSTPROCESADO.
%function sp=postproc(s)
%Entradas:
%s:señal decodificada.
%Salidas:
%sp:señal tras pasar por el filtro de deénfasis.
function sp=postproc(s)
sp=deemp(s);%aplicamos el filtro de deénfasis.
```

Observamos un nuevo módulo, el filtro de deénfasis, que elimina el efecto provocado por el filtro de preénfasis del codificador.

```
%DEÉNFASIS.
%function sp=deemp(s)
%Entradas:
%s:señal decodificada.
%Salidas:
%sp:señal tras pasar por el filtro de deénfasis.
function sp=deemp(s)
beta=28180*(2^(-15));%constante que define así la recomendación.
sp(1)=s(1);
for k=2:160
   sp(k)=s(k)+beta*sp(k-1);
end
```

Por último y para finalizar, veamos el fichero de variables iniciales. En él, se actualizan todas las variables iniciales utilizadas tanto en el decodificador, como VAD, codificador de canal, etc.

```
%FICHERO DE VARIABLES INICIALES.
[fil,col]=size(so);
ntramas=floor(col/160); %calculamos el número de tramas que contiene
la señal de entrada.
LARantc=zeros(1,8); %LAR anteriores.
dpc=zeros(1,120); %120 muestras anteriores reconstruidas.
rvad=[6 -4 1 0 0 0 0 0 0]; %parámetros del vad.
ACFvad=zeros(1,36);
av0ant=zeros(1,36);
dm=0;
Nj=40*ones(1,5);
oldl=0;
voldl=0;
adp=0;
thv=1000000;
buc=0;
hac=-1;
LARsid=zeros(1,32); %LAR para construir las tramas sid.
xmaxsid=zeros(1,16); %xmax para construir las tramas sid.
vad2=0; %valores anteriores del vad.
vad3=0;
vad4=0;
t5a=dec2bin(0,57); %tramas iniciales.
t6a=dec2bin(0,57);
t7a=dec2bin(0,57);
t8a=dec2bin(0,57);
BER=0; %tasa de error de bit del canal.
t1a=dec2bin(0,57); %tramas iniciales.
t2a=dec2bin(0,57);
t3a=dec2bin(0,57);
t4a=dec2bin(0,57);
Nact=40; %para que N esté en el rango [40-120].
dpd=zeros(1,120); %120 muestras anteriores reconstruidas.
LARantd=zeros(1,8); %LAR anteriores.
```

BIBLIOGRAFÍA

[1] *Recomendación GSM 06.10. GSM Full Rate Speech Transcoding. Feb 1992.*

[2] *Recomendación GSM 06.32.European Digital Cellular Communications System. (Phase 2). Voice Activity Detection (VAD). Jul 1995.*

[3] *Recomendación GSM 05.03. Digital Cellular Telecommunications System. (Phase 2 +). Channel Coding. Aug 1997.*

[4] *Recomendación GSM 06.12. European Digital Cellular Telecommunications System (Phase 2). Comfort Noise Aspects for Full Rate Speech Traffic Channels. Sep 1994.*

[5] *Digital Communications.*
Bernard Sklar. Ed. Prentice Hall.

[6] *Mobile Radio Communications.*
Raymond Steele Ed. Raymond Steele.

[7] *Apuntes de la Asignatura Transmisión de Datos. 3º Ingeniero de Telecomunicaciones. Escuela Superior de Ingenieros. Universidad de Sevilla.*

[8] *Apuntes de la Asignatura Tratamiento Digital de Señales en Comunicaciones. 5º Ingeniero de Telecomunicaciones. Escuela Superior de Ingenieros. Universidad de Sevilla.*

RECURSOS EN INTERNET

http://www.etsi.org

http://www.imagicom.co.uk/Gloss/R_Gloss/tsld034.htm

http://www.lincom-asg.com/ssadto/rpe-ltp.html

http://ceres.ugr.es/~alumnos/alonso/index.html

http://www.com.enst.fr/~rodrigez/GSM.html

http://ttt.upv.es/~aalbiol/aplictds/vozcodecs.html

http://ccl.cnu.ac.kr/~iyhwang/class/Mobile_Com/node14.html