

**UNIVERSIDAD NACIONAL DE INGENIERÍA**

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA



**IMPLEMENTACIÓN DE UN CODIFICADOR DE VOZ CELP MEJORADO PARA  
CANALES DE BANDA ANGOSTA**

**TESIS**

PARA OPTAR POR EL GRADO DE MAESTRO EN CIENCIAS

**MENCION: TELEMATICA**

PRESENTADO POR:

**FELIPE DANIEL ARGANDOÑA MARTINEZ**

**LIMA – PERÚ  
2008**

## SUMARIO

El presente trabajo de tesis describe el estudio e implementación en hardware (DSP) de un codificador de voz basado en el estandar CELP FS1016 para canales de banda angosta. En primer lugar, se realiza un breve estudio del marco teórico en el que se basan los codificadores de voz, luego se describen las técnicas de procesamiento de voz más importantes que conforman el presente trabajo, en especial las referidas al estandar FS1016, se describen también los detalles de la implementación en el DSP, y finalmente se presentan las pruebas de calidad respectivas.

En base al estudio y correcto conocimiento de todos los bloques constituyentes del esquema CELP FS1016, se proponen mejoras y optimizaciones tanto en el ámbito algorítmico, como en el ámbito del procesador DSP. Con ello se pretende obtener una mayor calidad de voz decodificada (sintética), sin comprometer seriamente la tasa de compresión del sistema ni el tiempo de ejecución del algoritmo, de tal forma que pueda ser empleado en sistemas de transmisión digital de datos de banda angosta.

En el presente trabajo se usaron varias técnicas que pertenecen al esquema de codificación de voz basado en la Predicción Lineal con Excitación de Código (Code Excited Linear Prediction) como son: Cuantización Escalar de los Coeficientes LPC usando la técnica LSP (Line Spectral Frequency), Interpolación de los LSP, Análisis por Síntesis basado en la minimización del error ponderado perceptualmente (Perceptual Weighting Error), Excitación basada en un Codebook Estocástico (Excitación Vectorial) mas un Codebook Adaptivo (Pitch), Post-Filtro con control automático de ganancia, etc.

La implementación en hardware constituye un aporte importante del presente trabajo. El codificador es implementado sobre la tarjeta de desarrollo DSK TMS320C6711 de Texas Instruments, lo cual permitirá su fácil incorporación a sistemas de comunicaciones banda angosta, a través de interfaces como RS232, RS485, etc.

Finalmente se evalúa la calidad de la voz sintética del codificador tanto sin inserción como con inserción de errores (BER variable) en la señal codificada.

## **ABSTRACT**

This thesis document describes the study and hardware implementation (DSP) of a CELP-based voice coder intended for narrow-band channels. Firstly, a brief study of the theoretical framework for voice coders is presented; then follows a description of the relevant voice processing techniques used in this work, mainly those that comprise the FS1016 standard; also, details of the implementation on DSP are described; and finally, the respective quality tests are shown.

Based on the study and correct understanding of all the blocks that constitute the CELP FS1016 model, many improvements and optimizations regarding both algorithmic and processor-related issues are proposed. It is expected to obtain a better decoded voice quality without seriously compromise neither the compression system rate nor the algorithm execution time so that it can be used in narrow band digital transmission systems.

This work used many techniques that come from the CELP (Code Excited Linear Prediction) approach, such as: Scalar quantization of linear prediction coefficients using LSP, Interpolation of LSP, analysis-by-synthesis based on the minimization of the perceptual weighting error, Stochastic Codebook excitation, Adaptive Codebook (Pitch), post-filter with automatic gain control and others.

Hardware implementation constitutes an important contribution in this work. The coder was implemented on the Texas Instruments DSK TMS320C6711 board. The coder implementation on hardware will enable it to be easily incorporated into a narrow band communication system through data interfaces such as RS232, RS485, etc.

Finally, the synthetic voice quality is tested through subjective tests, first without error insertion and then with a variable BER insertion.

## INDICE

|   |    |
|---|----|
| <b>INTRODUCCION</b> .....   | 1  |
| <b>CAPITULO I ANTECEDENTES Y PLANTEAMIENTO DEL PROBLEMA</b> .....             | 3  |
| 1.1 Antecedentes y Justificación .....  | 3  |
| 1.2 Soluciones comerciales en codificación de voz a bajas tasas de bits ..... | 5  |
| 1.3 Posibles aplicaciones prácticas del codificador .....                     | 6  |
| 1.4 Selección del codificador de voz.....                                     | 7  |
| 1.5 Objetivos de la tesis .....   | 9  |
| <b>CAPITULO II FUNDAMENTOS DE LA CODIFICACION DE LA VOZ</b> .....             | 10 |
| 2.1 Visión General de la Codificación de la Voz .....                         | 10 |
| 2.1.1 Características Optimas de un Codificador de Voz .....                  | 11 |
| 2.1.2 Clasificación de los Codificadores de Voz .....                         | 12 |
| 2.2 Modelado y Producción de la Voz .....                                     | 12 |
| 2.2.1 Estructura General de un Codificador de Voz .....                       | 14 |
| 2.3 Propiedades del sistema de audición humano .....                          | 15 |
| 2.3.1 Umbral Absoluto .....   | 15 |
| 2.3.2 Enmascaramiento .....   | 17 |
| 2.4 Estándares Actuales .....   | 17 |
| <b>CAPITULO III TECNICAS DE PROCESAMIENTO DE SEÑALES DE VOZ</b> .....         | 18 |
| 3.1 Estimación de la Frecuencia Fundamental o Pitch.....                      | 18 |
| 3.1.1 Método de Autocorrelación .....   | 18 |
| 3.1.2 Función Magnitud de la Diferencia .....                                 | 19 |
| 3.1.3 Pitch Fraccional .....  | 19 |
| 3.2 Determinación de Segmentos “sonoros” y “sordos” .....                     | 20 |
| 3.3 Densidad Espectral de Potencia .....                                      | 20 |

|   |   |           |
|---|---|-----------|
| 3.3.1   | Periodograma .....  | 21        |
| 3.4   | Modelo Autoregresivo .....  | 22        |
| 3.4.1   | Ecuación Normal o de Yule-Walker .....                                | 24        |
| 3.4.2   | Estimación de la Autocorrelación .....                                | 24        |
| 3.5   | Predicción lineal y técnicas de predicción lineal .....               | 25        |
| 3.5.1   | Predicción lineal.....  | 25        |
| 3.5.2   | Minimización del Error .....  | 27        |
| 3.5.3   | Ecuación Normal.....  | 27        |
| 3.5.4   | Mínimo Error Predictivo Cuadrático Medio.....                         | 28        |
| 3.6   | Esquemas de análisis predictivo.....                                  | 28        |
| 3.7   | Ganancia de predicción.....   | 29        |
| 3.8   | Coefficientes de Reflexión .....                                      | 30        |
| 3.9   | Algoritmo de Levinson-Durbin.....                                     | 31        |
| 3.9.1   | Conversión de los Coeficientes de Reflexión a Coeficientes LPC's..... | 33        |
| 3.9.2   | Conversión de los coeficientes LPC a coeficientes RC.....             | 33        |
| 3.10  | Predicción Lineal Long-Term.....                                      | 33        |
| 3.11  | Filtros de Síntesis .....   | 35        |
| 3.12  | Cuantización escalar de los coeficientes predictivos.....             | 37        |
| <b>CAPITULO IV PREDICCIÓN LINEAL CON EXCITACIÓN DE CÓDIGO .....</b>     |   | <b>38</b> |
| 4.1   | El modelo de Producción del esquema CELP .....                        | 38        |
| 4.2   | El principio de Análisis por Síntesis .....                           | 39        |
| 4.3   | Codificación y Decodificación .....                                   | 41        |
| 4.4   | Ponderado Perceptual (Perceptual Weighting) .....                     | 41        |
| 4.5   | Búsqueda del Codebook de Excitación Optimo .....                      | 44        |
| <b>CAPITULO V CORRECCION DE ERRORES (FEC) .....</b>                     |   | <b>45</b> |
| 5.1   | Control de errores hacia delante .....                                | 45        |
| 5.2   | Código Hamming de un solo bit .....                                   | 46        |
| 5.3   | Esquema Hamming usado por el estandar FS1016 .....                    | 46        |
| 5.4   | Esquema Hamming usado en el presente trabajo de tesis .....           | 46        |
| <b>CAPITULO VI CODIFICADOR IMPLEMENTADO: DETALLES Y RESULTADOS.....</b> |   | <b>48</b> |
| 6.1   | Señal de Entrada al Codificador .....                                 | 53        |

|          |   |     |
|----------|---|-----|
| 6.2      | Ventana de Hamming .....  | 54  |
| 6.3      | Filtro de entrada .....   | 55  |
| 6.4      | Posicionamiento de los 2 segmentos principales de análisis .....                  | 56  |
| 6.5      | Calculo de los Coeficientes RC, LPC, y Comprobación de Estabilidad .....          | 57  |
| 6.6      | Expansión del ancho de Banda a través de la modificación de los LPC's .....       | 59  |
| 6.7      | Conversión de los Coeficientes LPC a Coeficientes LSP .....                       | 61  |
| 6.8      | Cuantización de los Coeficientes LSP .....  | 70  |
| 6.9      | Interpolación de los Coeficientes LSP Cuantizados .....                           | 73  |
| 6.10     | Análisis de los 4 subsegmentos: Búsquedas Adaptiva y Estocástica .....            | 75  |
| 6.10.1   | Conversión de los coeficientes LSP a LCP .....                                    | 75  |
| 6.10.2   | Fundamentos Teóricos de la Búsqueda Adaptiva y Estocástica .....                  | 79  |
| 6.10.3   | Filtro de Síntesis del Pitch Optimizado para la minimización del error perceptual | 85  |
| 6.10.4   | PRIMERA ETAPA (Deducción del Codebook Adaptivo) .....                             | 85  |
| 6.10.4.1 | Algoritmo de Búsqueda del Codebook Adaptivo .....                                 | 90  |
| 6.10.4.2 | Delta Pitch Delay .....   | 91  |
| 6.10.4.3 | Criterio MSPE modificado .....  | 94  |
| 6.10.4.4 | Pitch fraccionario T en la búsqueda del Codebook Adaptivo .....                   | 97  |
| 6.10.4.5 | Cuantización de la ganancia b .....   | 99  |
| 6.10.4.6 | Codificación del periodo pitch T .....  | 99  |
| 6.10.5   | SEGUNDA ETAPA (Búsqueda en el Codebook Estocástico) .....                         | 101 |
| 6.10.5.1 | Estructura del Codebook Estocástico .....   | 101 |
| 6.10.5.2 | Algoritmo de Búsqueda del Codebook Estocástico .....                              | 102 |
| 6.10.5.3 | Modificación de la ganancia estocástica .....                                     | 105 |
| 6.10.5.4 | Cuantización de la ganancia estocástica modificada .....                          | 106 |
| 6.10.6   | Actualización del Codebook Adaptivo .....   | 110 |
| 6.10.7   | Empaquetamiento .....   | 111 |
| 6.11     | Resultados de las etapas de búsquedas adaptiva y estocástica .....                | 112 |
| 6.12     | Descripción del Decodificador .....   | 140 |
| 6.12.1   | Desempaquetamiento de las tramas de bits entrantes .....                          | 140 |
| 6.12.2   | Interpolación de los coeficientes LSP.....  | 140 |
| 6.12.3   | Codebook Adaptivo .....   | 140 |
| 6.12.4   | Codebook Estocástico .....  | 140 |
| 6.12.5   | Excitación Total y Filtro de Síntesis .....                                       | 140 |
| 6.12.6   | POST FILTRO .....   | 142 |
| 6.12.6.1 | Compensación Espectral Adaptiva .....   | 144 |
| 6.12.6.2 | Control Automático de la Ganancia .....   | 144 |

|        |   |     |
|--------|---|-----|
| 6.13   | RESUMEN DE LAS MEJORAS IMPLEMENTADAS .....  | 146 |
| 6.13.1 | Bloque de conversión de coeficientes LPC a LSP .....  | 146 |
| 6.13.2 | Bloques de cuantización de coeficientes LSP e interpolación de LSP .....                                | 148 |
| 6.13.3 | Optimización de la convolución recursiva para el bloque de<br>búsqueda en el Codebook Adaptivo .....    | 148 |
| 6.13.4 | Optimización de la convolución recursiva para el bloque de<br>búsqueda en el Codebook Estocástico ..... | 148 |
| 6.13.5 | Cuantización de la ganancia estocástica .....   | 150 |

## **CAPITULO VII DESCRIPCION DEL DSP TMS320C6711, DETALLES DE LA IMPLEMENTACION Y OPTIMIZACION .....**

|       |   |     |
|-------|---|-----|
| 7.1   | DSP TMS320C6711 Overview .....  | 151 |
| 7.1.1 | Tarjeta DSK C6711 .....   | 151 |
| 7.1.2 | Mapa de Memoria .....   | 153 |
| 7.1.3 | Arquitectura de la familia TMS320C67X .....   | 153 |
| 7.1.4 | Code Composer Studio .....  | 154 |
| 7.2   | Desarrollo del software en el Code Composer .....   | 156 |
| 7.2.1 | Creación del proyecto .....   | 157 |
| 7.2.2 | Añadiendo archivos básicos de soporte y librerías .....   | 159 |
| 7.2.3 | Compilando el proyecto .....  | 164 |
| 7.2.4 | Cargando y Ejecutando el programa en el DSP .....   | 164 |
| 7.3   | Mejoras y Optimización del código para óptimo desempeño en el DSP.....  | 165 |
| 7.3.1 | Estado Inicial (sin ningún tipo de optimización) .....  | 165 |
| 7.3.2 | Reacomodo de funciones críticas a la IRAM .....   | 166 |
| 7.3.3 | 2da Reubicación de funciones críticas a la IRAM .....   | 166 |
| 7.3.4 | Reubicación de variables críticas a la IRAM .....   | 167 |
| 7.3.5 | Modificación para optimización de rutinas ganancia estocástica y<br>ganancia_adaptiva. Optimización de la convolución recursiva ..... | 167 |
| 7.3.6 | Opciones del compilador para optimización .....   | 168 |

## **CAPITULO VIII PRUEBA DE CALIDAD DEL CODIFICADOR IMPLEMENTADO .....**

|     |   |     |
|-----|---|-----|
| 8.1 | Test MOS .....  | 172 |
| 8.2 | Resultados del Test MOS .....                         | 176 |
| 8.3 | Resultados del Test MOS con variaciones del BER ..... | 181 |

**CONCLUSIONES Y RECOMENDACIONES.....183**

**BIBLIOGRAFIA .....186**



## INTRODUCCION

El objetivo del presente trabajo de tesis apunta en primer lugar a estudiar, desarrollar, e implementar un codificador de voz basado en el esquema de Predicción Lineal por Excitación de Código, de calidad aceptable en función de los requerimientos de un sistema de comunicaciones de banda angosta. Conjuntamente, se proponen técnicas de mejoras y optimización apuntando a obtener una buena calidad de la voz decodificada de acuerdo al modelo. Cabe mencionar que la teoría expuesta y que fue usada en la realización del presente trabajo, fue obtenida del libro de Wai Chu, "Speech Coding Algorithms, Foundation and Evolution of Standardized Coders".

La implementación del codificador propuesto en una plataforma hardware destino constituida por un procesador digital de señales (DSP), en versión de punto flotante, constituye la parte principal del presente trabajo de tesis. Lo anterior implica diseñar modularmente los algoritmos de codificación y de decodificación de voz, apuntando a realizar mejoras futuras y/o aplicaciones de valor agregado (Ej. criptografía). Con esto se busca demostrar la operatividad del codificador de voz en hardware, lo cual constituye una base fundamental para la implementación de sistemas de comunicaciones reales. Finalmente la evaluación del codificador con los estándares subjetivos de medición establecidos (Test MOS) sirve para medir la calidad del codificador implementado.

El capítulo 1 presenta el esquema general, el contexto, la justificación y los objetivos del presente trabajo.

El capítulo 2 describe los fundamentos de la codificación de voz, los codificadores de voz, y varios conceptos básicos claves que son aplicados en muchos esquemas de codificación de voz. Así mismo, el capítulo 2 también nos ofrece información teórica acerca de la producción de la voz, su modelado y las propiedades del sistema de audición humano. Dichos conceptos juegan un papel crucial en el diseño de los codificadores modernos de voz.

El capítulo 3 describe las técnicas básicas que se emplean en el procesamiento de las señales de voz. Se describen también los algoritmos para el cálculo de los parámetros más importantes en el modelado de las señales de voz como son: el pitch, determinación de segmentos “sonoros” o “sordos”, densidad espectral de potencia, etc. también se presenta el estudio de la predicción lineal. Los algoritmos y técnicas presentadas en este capítulo son la base para algoritmos más complejos.

El capítulo 4 presenta los conceptos básicos del esquema de predicción lineal por excitación de código (CELP), el modelo, principios, el filtro de ponderación perceptual y una descripción general del proceso de búsqueda de un codebook.

El capítulo 5 nos describe el esquema FEC de corrección de errores así como el comparativo entre los esquemas FEC usados por el estándar FS1016 y la presente implementación.

El capítulo 6 presenta la descripción detallada del codificador implementado. Se hace una descripción general de los algoritmos empleados y se procede a describir paso a paso cada uno de los bloques por donde pasa la señal desde la entrada en PCM-16 bits / 8 KHz hasta la señal decodificada a la salida del post-filtro. Cabe destacar que cada uno de los bloques viene descrito con los fundamentos teóricos necesarios, fórmulas y diagramas de flujo de los algoritmos correspondientes y a su vez se presentan resultados (capturas del Code Composer Studio) los cuales respaldan la teoría subyacente y confirman en la práctica la correcta operación del codificador.

El capítulo 7 presenta la descripción de la tarjeta DSK TMS320C6711 usada en la presente implementación. Se describen las características técnicas de la tarjeta, sus prestaciones y la forma de programarla. Adicionalmente se presentan las mejoras que se realizaron al código para la optimización del mismo en el DSP.

El capítulo 8 presenta la evaluación subjetiva del codificador. Se presentan los resultados de una serie de pruebas del codificador tanto sin presencia de errores como con presencia de errores con una BER variable.

La última parte del documento presenta las conclusiones y recomendaciones del presente trabajo de tesis.

## CAPITULO I

### ANTECEDENTES Y PLANTEAMIENTO DEL PROBLEMA

#### 1.1 Antecedentes y Justificación

Debido a la masificación de las tecnologías para comunicación por voz, la codificación de la voz ha recibido, a través de los años, mucho interés por parte de la comunidad científica, las organizaciones de estándares y la industria, llegándose a obtener en la actualidad diversos modelos y soluciones (algunos muy sofisticados) orientados siempre a cumplir los requerimientos deseables en toda codificación de voz:

- Baja tasa de bits.
- Alta calidad de la voz.
- Buena calidad en presencia de otras señales (diferentes de las señales de voz).
- Bajo requerimiento de memoria.
- Baja complejidad computacional.
- Bajo retardo debido al tiempo de ejecución del código, etc.

Es muy difícil que un modelo o solución satisfaga todos los requerimientos arriba mencionados, sino que, dependiendo de la aplicación en particular se debe hacer una evaluación costo-beneficio entre los distintos requerimientos y escoger los que más se adecuen a nuestro caso. Una clasificación de los codificadores de voz es de acuerdo a la tasa de bits, tal como se muestra en la Tabla 1.1 (4).

Hoy en día la mayoría de los estándares son diseñados para tasas de bits menores que la Tasa de Bits media indicada en la Tabla. 1.1 y cada vez mas son las aplicaciones que requieren tasas de bits bajas con una buena calidad de voz. Tales aplicaciones incluyen: videojuegos, sistemas inalámbricos, sistemas celulares, defensa, *Digital Mobile Radio*, *VoIP*, comunicación multimedia, etc. La Tabla 1.2. muestra un resumen de los principales estándares de codificación de voz (4).

Tabla 1.1. Clasificación de los codificadores de voz de acuerdo a la tasa de bits (4).

| Categoría             | Rango de la Tasa de Bits |
|-----------------------|--------------------------|
| Tasa de Bits Alta     | > 15 kbps                |
| Tasa de Bits Media    | 5 a 15 kbps              |
| Tasa de Bits Baja     | 2 a 5 kbps               |
| Tasa de Bits Muy Baja | < 2 kbps                 |

Un grupo de aplicaciones donde es primordial codificar la voz a bajas tasas de bits es la transmisión por el medio inalámbrico. Como se puede ver en la Tabla 1.2 muchos estándares están destinados para trabajar en el sistema celular, en el cual se requiere un buen nivel de calidad de la voz y que debe de ser cumplido por los desarrolladores de codificadores de voz. Por otro lado, las técnicas de codificación de voz nos ofrecen las herramientas necesarias para hacer nuestros propios codificadores de voz “a la medida” para aplicaciones específicas donde no es necesario ceñirse a las especificaciones de los estándares (tamaños de los *frames* por ejemplo), consiguiendo de esta manera dar una solución particular a un determinado problema.

Cabe mencionar que al desarrollar codificadores a la medida, la interoperabilidad con otros sistemas no se ve comprometida debido a que la conversión de formatos se puede realizar en los elementos que conforman la red. De esta manera, solo es necesario incluir el codificador en algún componente de la red. Por ejemplo si se quiere implementar la interoperabilidad entre un codificador de voz de cualquier estandar (i.e. red celular usando el codificador UNI) y la red de telefonía fija, es necesario realizar dos conversiones: una para la voz proveniente desde las troncales TDM (conversión de voz de PCM a formato UNI) y la otra para la voz procedente de la red celular (conversión de voz de formato UNI a PCM). Dicho proceso de conversión puede estar ubicado en algún elemento de la red celular, y consiste en implementar el codificador y decodificador UNI en dicho elemento de red.

El presente trabajo de tesis hace uso de las técnicas de codificación de voz para implementar un codificador de voz para transmisión digital de la voz a través de canales de banda angosta, esto es, con bajas tasas de bit disponible.

Tabla 1.2. Estándares de codificación de voz (basado en (4)).

| Año en que se implementó | Nombre del Estándar          | Tasa de Bits (kbps)                            | Aplicaciones                                     |
|--------------------------|------------------------------|--|--|
| 1972                     | ITU - T G.711 PCM            | 64   | Propósito general                                |
| 1984                     | FS 1015 LPC                  | 2,4  | Comunicación segura                              |
| 1987                     | ETSI GSM 6,10 RPE-LTP        | 13   | Radio móvil digital                              |
| 1990                     | ITU - T G.726 ADPCM          | 16, 24, 32,40                                  | Propósito general                                |
| 1990                     | TIA IS54 VSELP               | 7,95   | Telefonía celular digital TDMA de norteamérica   |
| 1990                     | ETSI GSM 6,20 VSELP          | 5,6  | Sistema celular GSM                              |
| 1990                     | RCR STD-27B VSELP            | 6,7  | Sistema celular japonés                          |
| 1991                     | FS1016 CELP                  | 4,8  | Comunicación segura                              |
| 1992                     | ITU - T G.728 LD-CELP        | 16   | Propósito general                                |
| 1993                     | TIA IS96 VBR-CELP            | 8,5, 4, 2, 0,8                                 | Telefonía celular digital CDMA de norteamérica   |
| 1995                     | ITU - T G.723,1 MP-MLQ/ACELP | 5,3, 6,3                                       | Comunicaciones multimedia                        |
| 1995                     | ITU - T G.729 CS-ACELP       | 8  | Propósito general                                |
| 1996                     | ETSI GSM EFR ACELP           | 12,2   | Propósito general                                |
| 1996                     | TIA IS641 ACELP              | 7,4  | Telefonía celular digital TDMA de norteamérica   |
| 1997                     | FS MELP                      | 2,4  | Comunicación segura                              |
| 1999                     | ETSI AMR-ACELP               | 12,2, 10,2, 7,95, 7,40, 6,70, 5,90, 5,15, 4,75 | Propósito general                                |
| 2000                     | AMR-WB                       | 6.6 a 23.85                                    | UMTS, GSM  |
| 2004                     | VMR-WB                       | 0.8 a 13.3                                     | De tasa variable. Telefonía celular digital CDMA |

## 1.2 Soluciones comerciales en codificación de voz a bajas tasas de bits

Hoy en día son muchas las empresas que ofrecen soluciones para una amplia variedad de aplicaciones de codificación de la voz, sin embargo, una de las constantes de todas las soluciones comerciales es el elevado costo de sus productos, tal como se puede apreciar en la Tabla 1.3

Tabla 1.3. Precios de algunos codificadores de voz comerciales (41, 42).

| Compañía           | Producto              | Precio    |
|--------------------|-----------------------|-----------|
| Compandent         | Vocoder MELP 2.4 kpbs | \$1500    |
| Compandent         | Vocoder MELP 1.2 kpbs | > \$2000  |
| Compandent         | Vocoder MELP 600 kpbs | > \$2000  |
| DSP Wizard         | G.729.B Vocoder GSM   | > \$11000 |
| Vocal Technologies | Vocoder MELP          | > \$2000  |

En general, las soluciones comerciales presentan varias consideraciones que deberán tomarse en cuenta a la hora de optar por la selección de un codificador de voz. Un factor a considerar es que en las soluciones comerciales, solo se obtienen buenos precios si se adquieren grandes cantidades de *chips* que implementan codificadores de

voz, lo cual es ventajoso para las empresas que fabrican equipos electrónicos de consumo masivo (celulares), pero no para proyectos de pequeña escala, proyectos experimentales, o proyectos de desarrollo nacional (donde no se requieren grandes cantidades de *chips*). También, se puede dar el caso de que los codificadores de voz disponibles comercialmente realicen más de lo que en realidad se necesita en la práctica para nuestro proyecto en particular, con lo cual se estaría pagando un exceso.

Por otro lado, es primordial que nuestro país se convierta poco a poco en un foco generador de tecnología, para lo cual es necesario que se desarrollen proyectos de investigación y desarrollo que nos permitan adquirir el “*know-how*” tecnológico en diversas áreas. Por tanto, se debe dar preferencia al desarrollo de tecnología nacional que a la compra de soluciones comerciales ya que de este modo, el Perú podrá eventualmente pasar de ser un país importador a ser un país generador de conocimiento y tecnología, lo cual contribuirá a su mejor posicionamiento en el escenario internacional.

Ante esto, es fundamental que se pueda contar con una solución desarrollada localmente, hecha a la medida de los requerimientos de un canal de comunicaciones de bajas tasas de bits (del orden de 5kbps) y a bajo costo, lo cual es precisamente el aporte del presente trabajo de tesis.

### **1.3 Posibles aplicaciones prácticas del codificador**

Las aplicaciones de un codificador de voz para bajas tasas de bits son muchas: desde aplicativos de Internet, juegos en red, comunicaciones inalámbricas, etc. Una aplicación interesante es la transmisión digital de la voz por canales HF el cual es un medio que, entre otros limitantes, restringe el ancho de banda disponible, pero por otro lado permite comunicaciones de larga y muy larga distancia gracias a un fenómeno conocido como propagación ionosférica, consistente en la reflexión de las señales de radiofrecuencia en las capas altas de la atmósfera, estando la más importante de ellas situada a 250 Km. de altitud (ver Fig. 1.1). La transmisión de voz analógica mediante un enlace HF, presenta una inteligibilidad pobre de la comunicación debido al desvanecimiento del canal y a la presencia del ruido. Otro de los problemas de la transmisión analógica por HF es la degradación de la señal al atravesar grandes distancias (pues al amplificar, también se incrementa el ruido), lo que ocasiona que la voz recibida posea baja calidad. Por otro lado, las ventajas de digitalizar un canal son muchas y se pueden mencionar las siguientes: mayor inmunidad frente al ruido, posibilidad de

cifrado de la información, permite la multiplexación de varias señales digitales, permite incorporar funciones robustas de corrección de errores, entre otras. En el caso de un canal HF, la digitalización del canal es ventajosa ya que permite transmitir voz (codificada) y datos, sobre un canal en el que las señales transmitidas se encuentran expuestas a efectos de absorción atmosférica, elevado ruido y un acusado *multipath* (multicamino), y además con condiciones de transmisión que dependen de muchos factores (momento del día, estación del año, actividad de las manchas solares, tormentas ionosféricas, etc.).

En los canales presentes en la banda HF (3-30MHz) y dependiendo de la modulación digital empleada, se pueden alcanzar velocidades de transmisión de datos de 2.4 kbps, 4.8kbps y hasta 9.6 kbps (si se usan técnicas sofisticadas de modulación digital). Son precisamente éstas bajas tasas de transmisión de datos las que nos condicionan a desarrollar codificadores de Baja Tasa de Bits, tal como el que se propone en el presente trabajo.

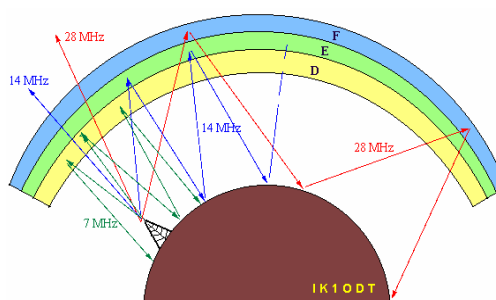


Fig. 1.1. Rebote de las ondas HF a través de la Ionosfera [43].

Por otro lado, tal como se comentó en la página anterior, se cuentan en el mercado con codificadores de voz comerciales (*vocoders*) independientes y equipos transreceptores que asimismo incluyen *vocoders* pero cuyos precios elevados los harían prohibitivos para un proyecto de desarrollo social en nuestro país. Por lo que es imperativo el desarrollo local de codificadores de voz para bajas tasas de bits, tal como se propone en el presente trabajo.

#### 1.4 Selección del codificador de voz

Para el presente trabajo de tesis, se selecciona un codificador de voz basado en el estándar FS1016 CELP (Code Excited Linear Prediction), por las siguientes razones:

- Ofrece una calidad superior a los codificadores basados en el modelo LPC-10 (como el FS1015 que fue originalmente el que se iba a considerar).
- Se consiguen las bajas tasas de bits a las que apunta el presente trabajo (tasas del orden de 5kbps).
- Los codificadores basados en el estandar FS1016 son de complejidad media-alta, lo que es favorable porque se obtendrá un “*know-how*” apreciable en las tecnologías de los codificadores de voz.
- Las variantes del esquema CELP (FS1016) constituyen la base de muchos codificadores que se usan en los sistemas de telecomunicaciones en la actualidad:
  - Codec AMR (Adaptive Multirate) usado en sistemas GSM 2.5G y UMTS. (Ver GSM 06.90 de 3GPP). El codificador AMR-WB es un codificador híbrido que usa el esquema A-CELP para las bajas tasas de bits.
  - AMR-WB usado en WCDMA (UMTS). El codificador AMR-WB esta basado en el esquema A-CELP (ver 3GPP).
  - G.729 a 8kbps usado mayormente en aplicaciones de Voz sobre IP, basado en un esquema ACELP, CELP Algebraico (ver UIT).
  - G.728 a 16Kbps el cual usa un esquema LD-CELP (Low Delay). Es usado también en voz sobre IP (ver UIT).
- Mediante el desarrollo de un codificador basado en el esquema CELP, el cual es un esquema posterior a las tecnologías de procesamiento de voz que usan técnicas LPC, se consigue alcanzar una fase intermedia para posteriormente desarrollar trabajos que usen técnicas mas avanzadas de codificación de voz (esquemas MELP, AMR-WB, AMR-WB+, etc).
- Se conseguirá implementar un sistema embebido en hardware (DSP) medianamente complejo, que implemente un codificador de voz.



### 1.5 Objetivos de la tesis

- Estudiar, desarrollar e implementar un codificador de voz basado en el estandar FS1016, de calidad aceptable en función de los requerimientos de un sistema de comunicaciones de banda angosta (del orden de 5kbps).
- Proponer técnicas de mejoras y optimización apuntando a obtener una calidad buena y aceptable de la voz decodificada con el codificador propuesto.
- Implementar el codificador propuesto en hardware (procesador de señales) pensado en integraciones futuras con sistemas de comunicaciones de bajas tasas de bits.
- Diseñar modularmente los algoritmos de codificación y de decodificación de voz, apuntando a realizar mejoras futuras y/o aplicaciones de valor agregado (Ej. criptografía).
- Evaluar el codificador con los estándares de medición establecidos (subjetivos).

## **CAPITULO II**

### **FUNDAMENTOS DE LA CODIFICACION DE LA VOZ**

#### **2.1 Visión General de la Codificación de la Voz**

La codificación de la voz persigue representar una señal de voz digitalizada usando la menor cantidad de bits como sea posible y tratando a su vez de mantener un nivel razonable de calidad de la voz de acuerdo a la aplicación en cuestión. Por ejemplo las aplicaciones de telefonía celular exigen un alto nivel de calidad de voz para satisfacer los requerimientos de usuarios finales. Por otro lado hay aplicaciones como por ejemplo los juegos en red (a través de la Internet) los cuales no son muy exigentes en cuanto a la calidad de la voz recibida o transmitida.

La codificación de la voz siempre ha sido y sigue siendo un campo de interés en la comunidad científica y cada año se siguen desarrollando teorías y técnicas para obtener codificadores de voz cada vez mas sofisticados.

Básicamente un codificador de voz es logrado mediante un algoritmo computacional. Dicho algoritmo puede ser ejecutado en un procesador específico como lo es un DSP, o también puede ser implementado puramente en hardware, como sería el caso de un FPGA. Ambas plataformas presentan ventajas y desventajas y depende del diseñador escoger la plataforma final de implementación.

Hay muchos esquemas, técnicas y estándares bien definidos y ampliamente usados dentro del campo de la codificación de la voz, los cuales pueden ser aplicados a una situación en particular sujeta a determinadas restricciones. La ubicación de un codificador de voz dentro de un sistema completo de comunicaciones es tal como se muestra en la Fig. 2.1 (ver bloques de color verde).

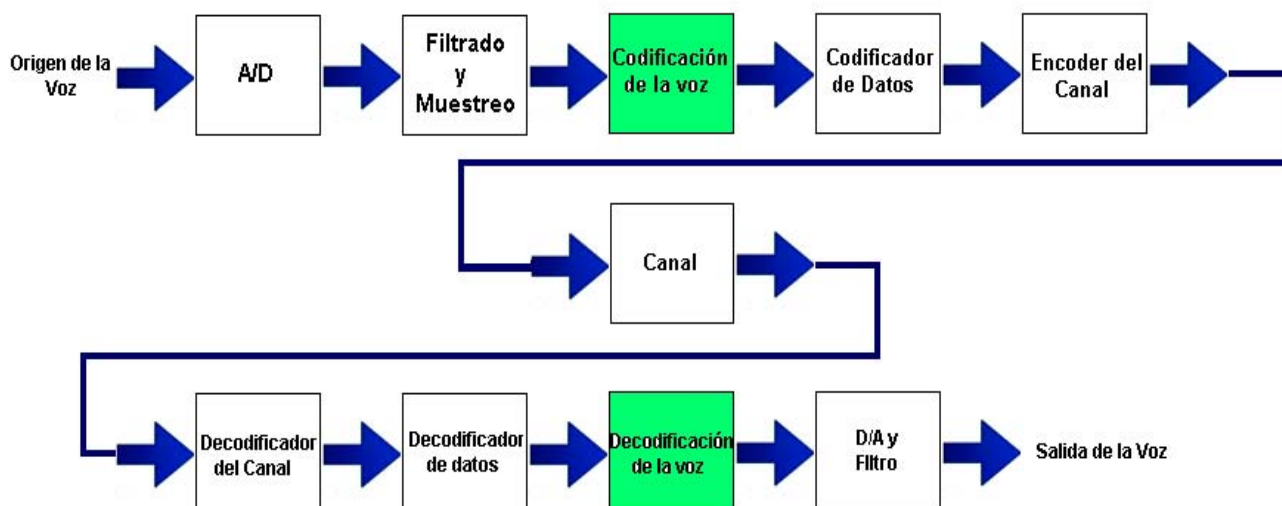


Fig. 2.1. Ubicación de un Codificador de Voz en un Sistema de Comunicación (4).

### 2.1.1 Características Óptimas de un Codificador de Voz

Son las siguientes:

- *Baja Tasa de Bits.* Mientras mas baja sea la tasa de bits, menos ancho de banda es necesario para su transmisión, conduciendo a un sistema más eficiente. Este requerimiento esta en conflicto constante con otras propiedades del sistema, principalmente la calidad de la voz. En la práctica siempre se busca el equilibrio entre estos dos factores.
- *Alta Calidad de la Voz.* La voz decodificada debería de tener una calidad acorde a la aplicación específica. Hay varios parámetros en la percepción de la calidad de la voz: inteligibilidad, naturalidad, reconocimiento del hablante, etc.
- *Robustez frente a diferentes hablantes e idiomas.* La técnica empleada en el codificador de voz debe ser lo suficientemente robusta como para modelar eficientemente diferentes hablantes (hombres y mujeres adultos, y niños) y diferentes idiomas.
- *Robustez frente a errores en la transmisión.* Es crucial para comunicaciones digitales donde los errores en la transmisión tendrán un impacto negativo en la calidad de la voz.
- *Buena respuesta frente a señales que no son de voz.* Por ejemplo, en un sistema típico de telefonía, es típico encontrar tonos DTMF. Si bien es cierto, los codificadores no están diseñados para lidiar con este tipo de señales, deben responder de tal forma que no ocasionen ruidos molestos en el receptor.

- *Bajos requerimientos de memoria.* Con el fin de que sean realizables en la práctica sobre todo en procesadores de bajo precio, es crucial que el requerimiento de memoria sea mínimo.
- *Bajo retardo de codificación.* El codificador no debe introducir demasiado retardo ya que esto podría influir en la performance de todo el sistema.

### 2.1.2 Clasificación de los Codificadores de Voz

**Clasificación de acuerdo a la tasa de bits.** Esta clasificación se resume en la Tabla 1.1.

**Clasificación por Técnica.** Tenemos las siguientes:

- *Codificadores de forma de onda.* Estos codificadores hacen un intento para preservar la forma original de la forma de onda de la señal de entrada. Aquí, el codificador puede ser aplicado a cualquier señal de entrada y son apropiados para la codificación de la voz a altas tasas de bits. Como ejemplo de estos codificadores tenemos el estandar G.726 (ADPCM) el cual ofrece 40, 32 y 24kbit/s. En la práctica, estos codificadores trabajan mejor a tasas de bits de 32 Kbps.
- *Codificadores Paramétricos.* Dentro del marco de trabajo de estos tipos de codificadores, se asume que la señal de voz es generada a partir de un modelo el cual es controlado por algunos parámetros. Durante la codificación, los parámetros del modelo son estimados de la señal de entrada, seguidamente estos parámetros son transmitidos como una secuencia de bits.
- *Codificadores Híbridos.* Este tipo de codificador combina la fuerza de un codificador de forma de onda con el de un codificador paramétrico.

## 2.2 Modelado y Producción de la Voz

La voz es una onda sonora de presión originada por movimientos controlados de estructuras anatómicas que conforman el sistema humano de producción de la voz tal como se puede apreciar en la Fig. 2.2.

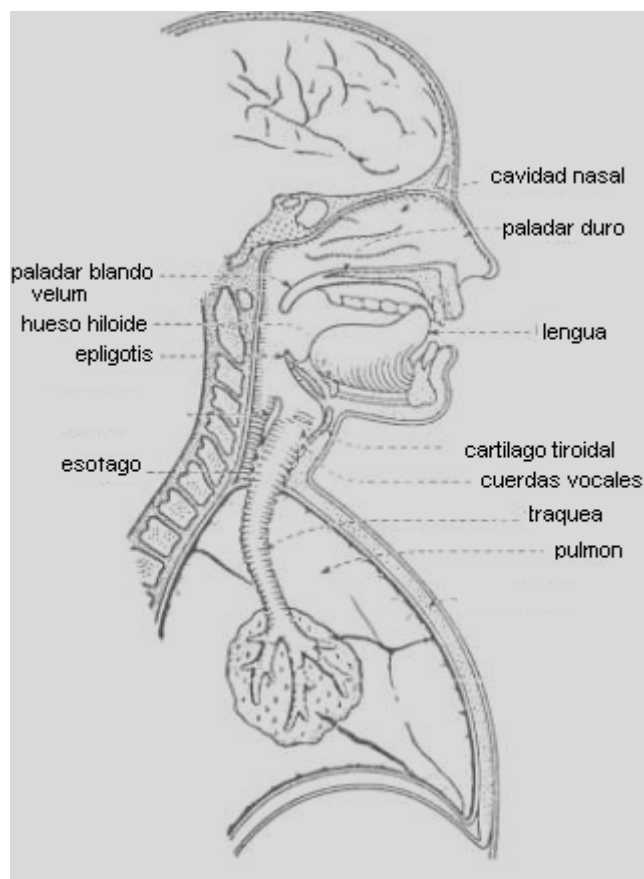


Fig. 2.2. Diagrama del sistema humano de producción de voz (44)

La voz es generada como una onda acústica que es radiada desde las fosas nasales y la boca cuando el aire es expulsado de los pulmones con el resultante flujo de aire perturbado por las constricciones (acción y efecto de apretar y cerrar) dentro del cuerpo. Es muy útil interpretar la producción de la voz en términos de filtrado acústico. Las tres principales cavidades del sistema de producción de la voz son la nasal, oral y faríngea los cuales a su vez forman el filtro acústico principal. El filtro es excitado por el aire de los pulmones y es cargado en su salida principal por una impedancia de radiación asociada con los labios.

En la faringe encontramos uno de los componentes más importante del sistema de producción de la voz: las cuerdas vocales. La localización de las cuerdas vocales esta a la altura de la manzana de Adán. Las cuerdas vocales son un par de bandas de músculos y membranas mucosas que se abren y cierran rápidamente durante la producción de la voz. La velocidad a la cual las cuerdas se abren y cierran es única para cada persona y define la característica y personalidad de una voz en particular.

Una señal de voz se puede clasificar como “sonora” o “sorda”. Las señales “sonoras” son generadas cuando las cuerdas vocales vibran de tal forma que el flujo de aire desde los pulmones es interrumpido periódicamente, creando una secuencia de pulsos que excitan el tracto vocal. Cuando las cuerdas vocales están estacionarias, la turbulencia creada por el flujo de aire pasando a través de una contracción del tracto vocal genera sonidos “sordos”. En el dominio del tiempo los sonidos “sonoros” se caracterizan por una fuerte periodicidad presente en la señal, con la frecuencia fundamental conocida como la frecuencia *pitch*. Para los hombres, el pitch se encuentra en el rango de 50 a 250 Hz mientras que para mujeres el pitch abarca las frecuencias entre 120 y 500 Hz. Los sonidos “sordos”, por el otro lado, no presentan ningún tipo de periodicidad y son esencialmente de naturaleza aleatoria (4). La Fig. 2.13 muestra segmentos de señales de voz “sonoros” y “sordos”, con sus respectivos espectros.

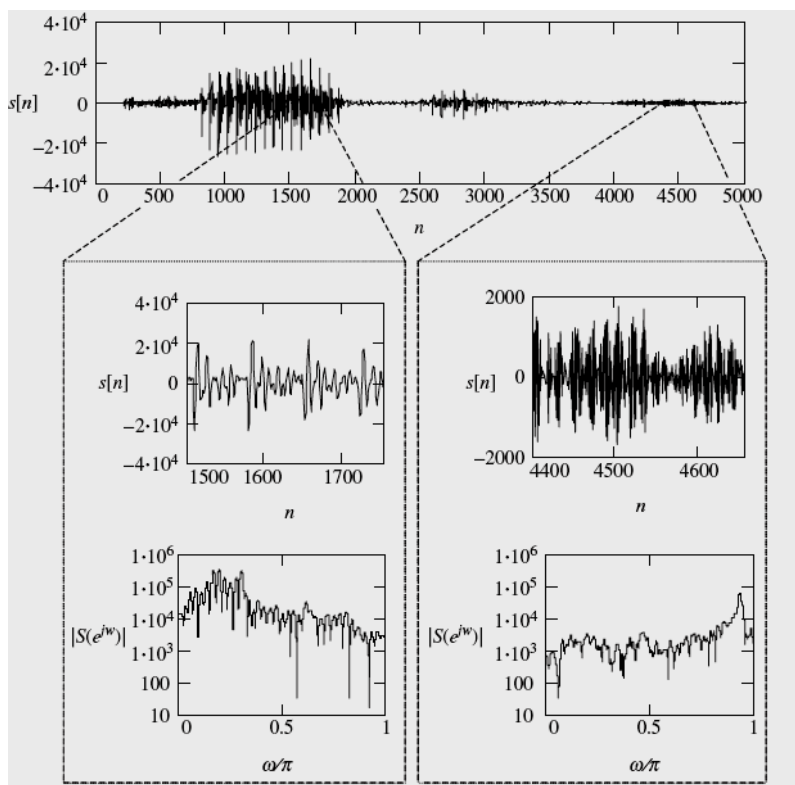


Fig. 2.3 Muestras de segmento de señales “con voz” y “sin voz” y sus espectros (4)

### 2.2.1 Estructura General de un Codificador de Voz

La Fig. 2.4 muestra un diagrama de bloques genérico de un codificador de voz paramétrico de propósito general.

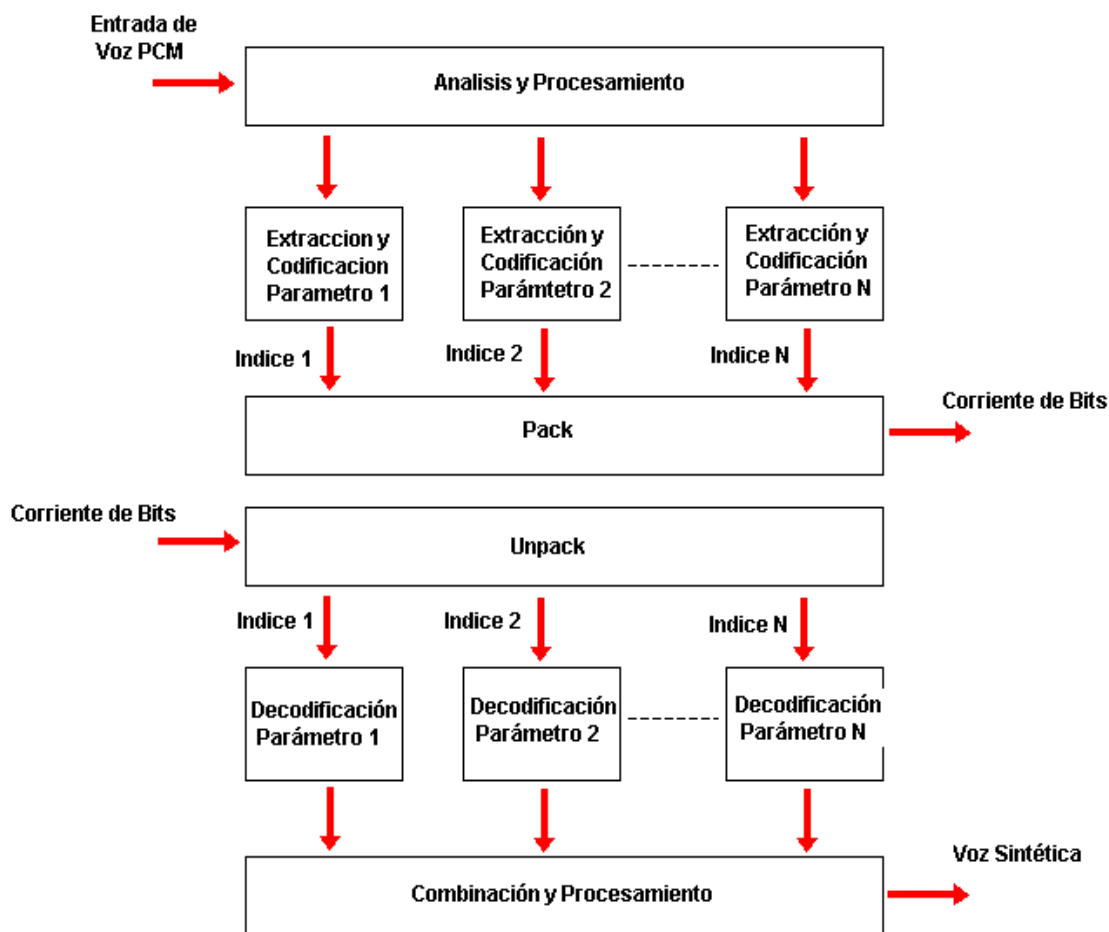


Fig. 2.4 Estructura General de un Codificador de Voz (basado en 4).

## 2.3 Propiedades del sistema de audición humano

Para desarrollar un modelo eficiente de codificación de voz, es muy importante tener en cuenta las propiedades del sistema de audición humano para de esta forma aprovechar sus características en el modelo a desarrollar.

La Fig. 2.5. muestra el diagrama del sistema de audición humano.

### 2.3.1 Umbral Absoluto

El umbral absoluto de un sonido es el nivel mínimo detectable de ese sonido en la ausencia de otros sonidos externos. De esta forma, el umbral absoluto caracteriza la cantidad de energía necesaria en un tono puro de tal forma que pueda ser detectado por un oyente en un ambiente sin ruido. La Fig. 2.6 muestra una curva típica de umbral absoluto.

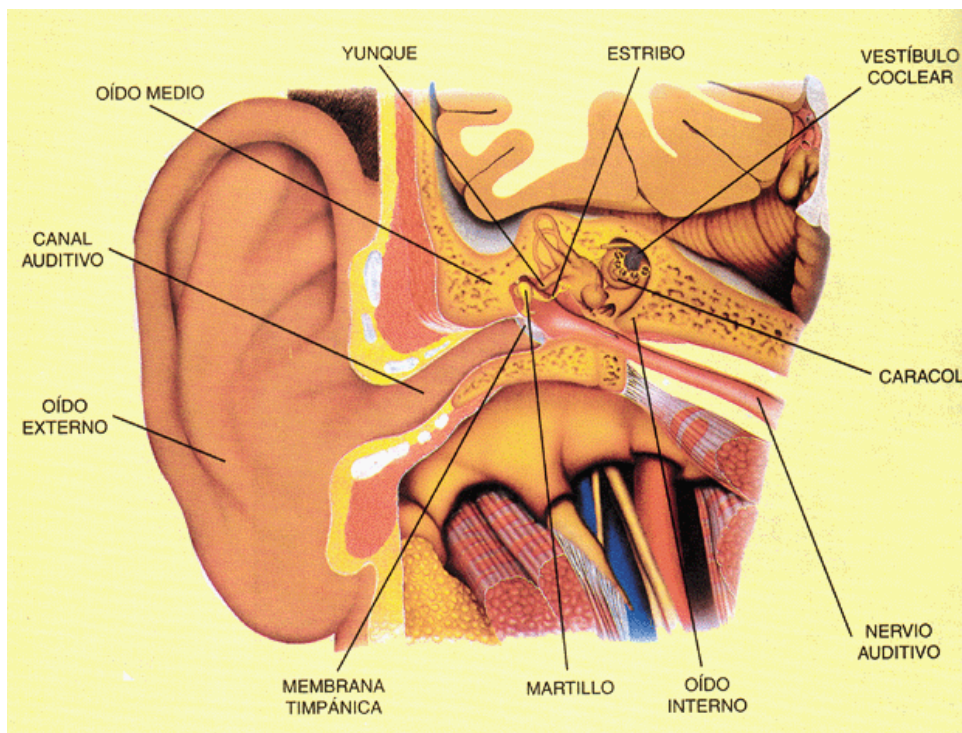


Fig. 2.5 Estructura del Sistema de Audición Humano (45)

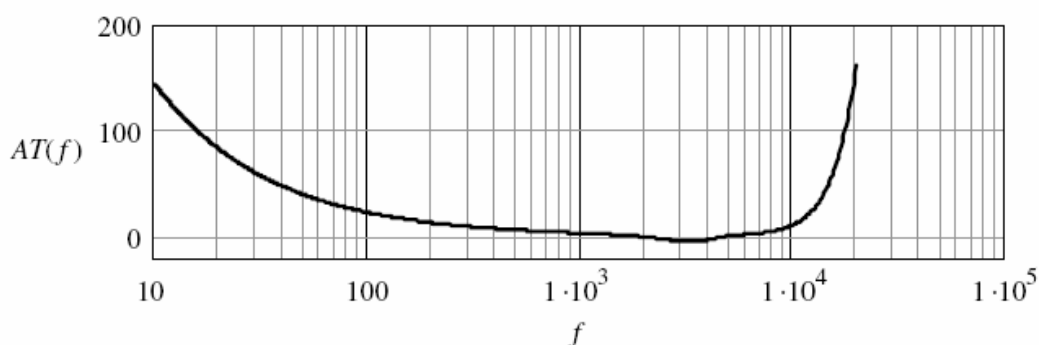


Fig. 2.6 Curva típica de umbral absoluto (4)

Algunas características del umbral absoluto se aplican en el desarrollo de codificadores. Se pueden mencionar las siguientes (4).

- Cualquier señal con una intensidad por debajo del umbral absoluto (ver Fig. 2.6) no necesita ser tomada en cuenta ya que no tiene impacto en la calidad del codificador de voz.
- La mayor parte de los recursos deben ser empleados para la representación de las señales dentro del rango de frecuencias más sensible para el oído humano, el cual en términos generales se encuentra en el intervalo 1 – 4KHz.



### 2.3.2 Enmascaramiento

El enmascaramiento se refiere al hecho de que un determinado sonido puede ser inaudible en presencia de otros sonidos. La presencia de un único tono, por ejemplo, puede enmascarar las señales vecinas, siendo la capacidad de enmascaramiento inversamente proporcional a la diferencia absoluta en frecuencia. La Fig. 2.7 muestra un ejemplo donde un único tono genera una curva de enmascaramiento que ocasiona que cualquier otra señal con intensidad menor a esta sea imperceptible.

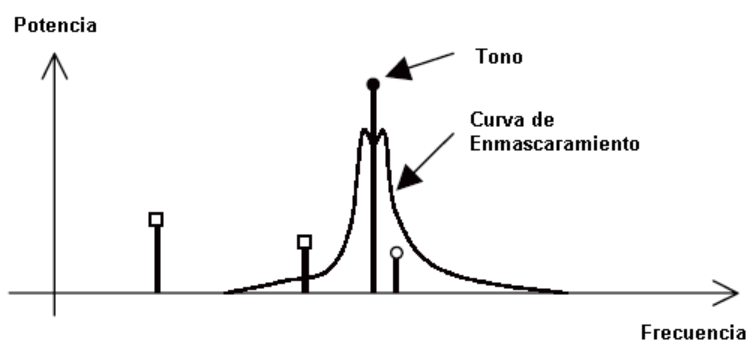


Fig. 2.7 Enmascaramiento debido a un tono (4)

### 2.4 Estándares Actuales

A lo largo de los años ha habido muchos estándares en el área de codificación de la voz, los cuales se han aplicado en la industria en productos de consumo masivo principalmente en el área de las telecomunicaciones. Ver la Tabla 1.2 y la Fig. 2.8.

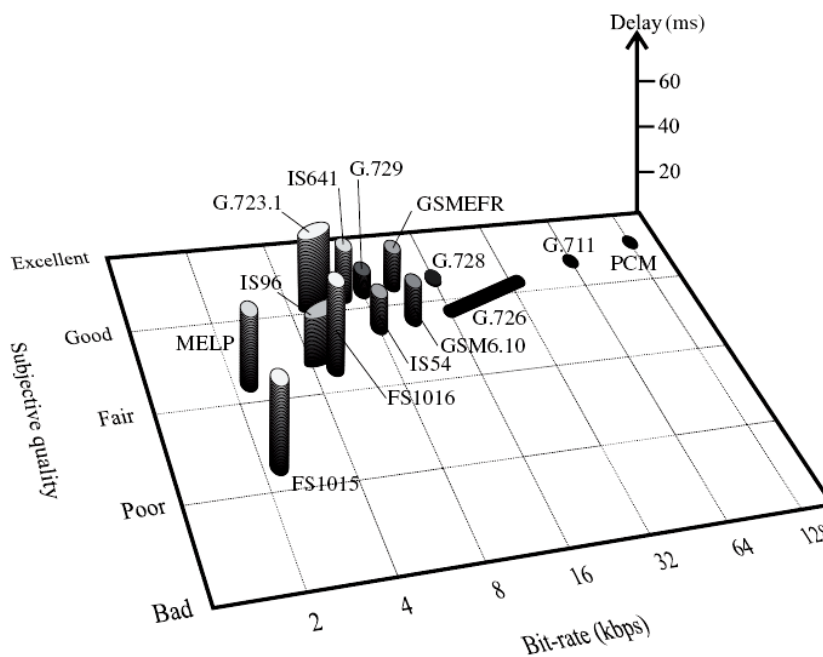


Fig. 2.8 Comparación de varios estándares por Bit-rate, Delay y Calidad [4]

## CAPITULO III

### TECNICAS DE PROCESAMIENTO DE SEÑALES DE VOZ

Se describen a continuación varios conceptos fundamentales y técnicas básicas y/o generales que se usarán a lo largo del presente trabajo de tesis. Cabe mencionar que existen muchas técnicas para procesar las señales de voz con el fin de realizar un codificador paramétrico y aquí solo se tomarán en cuenta las importantes y que serán de utilidad en el presente trabajo. Los parámetros a obtener en el modelo CELP se describen a lo largo de los capítulos III, IV, V y VI.

#### 3.1 Estimación de la Frecuencia Fundamental o Pitch

El pitch es un parámetro que identifica al locutor o hablante de tal forma que constituye uno de los parámetros más importantes en el análisis, síntesis y codificación de la voz. El pitch es una característica única de cada persona. Los segmentos “sonoros” son generados cuando el flujo de aire proveniente de los pulmones es periódicamente interrumpido por los movimientos de las cuerdas vocales. El tiempo entre aperturas sucesivas de las cuerdas vocales es llamado periodo fundamental. Para los hombres el rango del pitch ó frecuencia fundamental se encuentra en el intervalo de 50 a 250 Hz mientras que para las mujeres el rango cae dentro del intervalo de 120 a 500 Hz.

##### 3.1.1 Método de Autocorrelación

Es un método básico y directo para el cálculo del pitch, generalmente es usado para darnos una primera aproximación al valor real porque tiene un margen de error considerable. El valor de la autocorrelacion se calcula de acuerdo a la ecuación (3.1).

$$R[l, m] = \sum_{n=m-N+1}^m s[n]s[n-l] \quad (3.1)$$

La autocorrelacion refleja la similaridad entre el segmento  $s[n]$ ,  $n = m-N+1$  a  $m$ , con respecto a la versión desplazada en el tiempo  $s[n-l]$ , donde  $l$  es un entero positivo

que representa un lapso de tiempo. El rango de  $l$  es seleccionado de tal forma que cubra un amplio rango de valores del periodo del pitch. Por ejemplo  $l = 20$  a  $147$  (2.5 a 18.3 ms) y de esta forma las posibles frecuencias de pitch van de 54.5 Hz a 400 Hz (considerando un muestreo de 8KHz). Este rango de  $l$  es aplicable para la mayoría de hablantes y puede ser codificado con 7 bits. Finalmente se escoge el valor de  $l$  que corresponde al primer pico.

### 3.1.2 Función Magnitud de la Diferencia

También es usado para obtener una primera aproximación del valor real del pitch. La ventaja es que la función magnitud de la diferencia evita las multiplicaciones que se realizan en la ecuación (3.1) y es dada en la ecuación (3.2).

$$MDF[l, m] = \sum_{n=m-N+1}^m |s[n] - s[n-l]| \quad (3.2)$$

En este el valor de  $l$  que minimiza la función MDF es la correspondiente a la estimación del pitch. En el presente trabajo se usa el método de autocorrelación dado por (3.1).

### 3.1.3 Pitch Fraccional

En aplicaciones reales y eficientes es necesario obtener el valor del pitch con la mayor precisión posible. Para lograr mas precisión en el calculo del pitch se han propuestos muchos métodos siendo los mas eficientes los basados en sistemas multitasa e interpolación. Uno de los métodos para hallar valores fraccionales del pitch es el Medan-Yair-Chazan. El esquema CELP (Code Excited Linear Prediction) usa un esquema basado en barrido de valores y es el que se emplea en el presente trabajo de tesis.

## 3.2 Determinación de Segmentos “sonoros” y “sordos”

En el caso de los codificadores basados en el estandar FS1016 CELP, no existe esta clasificación, lo que los hace más óptimos ya que no existe la restricción de clasificar un segmento como “sonoros” o “sordos”. Sin embargo, en varios esquemas de codificación, sobretodo los basados en el estandar FS1015 y sus derivados, la parte del

algoritmo que determina si un determinado segmento es “sonoro” o “sordo” es crucial ya que es en base a esta determinación que el resto de estos algoritmos trabajan. Para determinar si un segmento es “sonoro” o “sordo”, existen varios métodos. Uno de ellos es el algoritmo SIFT el cual trabaja con un algoritmo de interpolación y un esquema de decisión basado en un valor entregado por la etapa de interpolación (para mayores detalles revisar el documento: Jhon Markel, "The Sift algorithm for Fundamental Frequency Estimation", IEEE Transactions on Audio and Electroacoustics, Vol 20, N° 5.

### 3.3 Densidad Espectral de Potencia

La Densidad Espectral de Potencia (PSD en inglés) describe las características estadísticas de un proceso estocástico en el dominio de la frecuencia.

Para una señal determinística  $x[n]$ , la potencia promedio esta dada por la siguiente ecuación:

$$P = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-\infty}^{\infty} |x_N[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \lim_{N \rightarrow \infty} \left( \frac{|X_N(e^{jw})|^2}{2N+1} \right) dw \quad (3.3)$$

Para un proceso estocástico  $x[n]$ , la potencia promedio esta dada por la siguiente formula:

$$P = \frac{1}{2\pi} \int_{-\pi}^{\pi} \lim_{N \rightarrow \infty} \left( \frac{E\{|X_N(e^{jw})|^2\}}{2N+1} \right) dw \quad (3.4)$$

La Densidad Espectral de Potencia esta dada por:

$$S(e^{jw}) = \lim_{N \rightarrow \infty} \left( \frac{E\{|X_N(e^{jw})|^2\}}{2N+1} \right) \quad (3.5)$$

Para un proceso estocástico estacionario en el sentido amplio (WSS), se cumple la siguiente ecuación para la densidad espectral de potencia. (Nota: las señales de voz se modelan como de este tipo, es decir como WSS):

$$S(e^{j\omega}) = \sum_{l=-\infty}^{\infty} R[l]e^{-j\omega l} \quad \text{y} \quad R[l] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(e^{j\omega})e^{j\omega l} d\omega \quad (3.6)$$

$$\text{Para el ruido blanco: } S(e^{j\omega}) = \sigma^2 \quad R[l] = \sigma^2 \delta[l] \quad (3.7)$$

Varias relaciones útiles:

$$\begin{aligned} R_{YX}[l] &= h[l] * R_X[l] & R_{XY}[l] &= h[-l] * R_X[l] \\ R_Y[l] &= h[l] * R_{XY}[l] & R_X[l] &= h[l] * h[l] * R_X[l] \end{aligned} \quad (3.8)$$

Donde  $R_{YX}[l]$ ,  $R_{XY}[l]$ ,  $R_Y[l]$  son las correlaciones correspondientes entre la entrada  $x[n]$  y la salida  $y[n]$  de un sistema lineal representado por  $h[n]$ .

### 3.3.1 Periodograma

Considerar una secuencia  $x[n]$ ,  $n = 0, 1, \dots, N-1$ . (Recordar que  $x[n]$  representa un proceso estocástico estacionario en el sentido amplio). El periodograma  $I_N(e^{j\omega})$  está definido por:

$$I_N(e^{j\omega}) = \frac{1}{N} |X_N(e^{j\omega})|^2 \quad (3.9)$$

Con

$$X_N(e^{j\omega}) = \sum_{n=0}^{N-1} w[n]x[n]e^{-j\omega n} \quad (3.10)$$

La ventana  $w[n]$  es escogida adecuadamente para minimizar el derrame espectral (*spectral leaking*) debido al truncamiento de secuencias finitas. Una ventana popular es la de Hamming.

La relación entre el periodograma y la autocorrelación de una señal está dada por las siguientes ecuaciones:

$$I_N(e^{j\omega}) = \sum_{l=-(N-1)}^{N-1} R[l]e^{-j\omega l} \quad \text{y} \quad R[l] = \frac{1}{N} \sum_{m=0}^{N-1} w[m+l]w[m]x[m+l]x[m] \quad (3.11)$$

Tener en cuenta que el periodograma es una estimación de la PSD usando un número finito de muestras de la señal de entrada, siendo el estimado una función

aproximada de la función real. En la práctica debido a que solo se cuentan con secuencias finitas, el periodograma ha llegado a ser una herramienta de análisis muy útil e importante. El periodograma se usa para representar la densidad espectral de potencia PSD de señales reales, de las cuales solo se dispone un determinado número de muestras. Por ejemplo, en la Fig. 3.1 podemos observar (en la parte superior) una señal aleatoria. Es sabido que, matemáticamente la PSD de una señal aleatoria es una línea constante, tal como se aprecia en la parte inferior (línea sólida), sin embargo en la realidad la aproximación dada por el periodograma (parte inferior línea punteada) nos da una aproximación a la PSD teórica. Mientras se tengan mas muestras de la señal la aproximación del periodograma a la PSD será mayor.

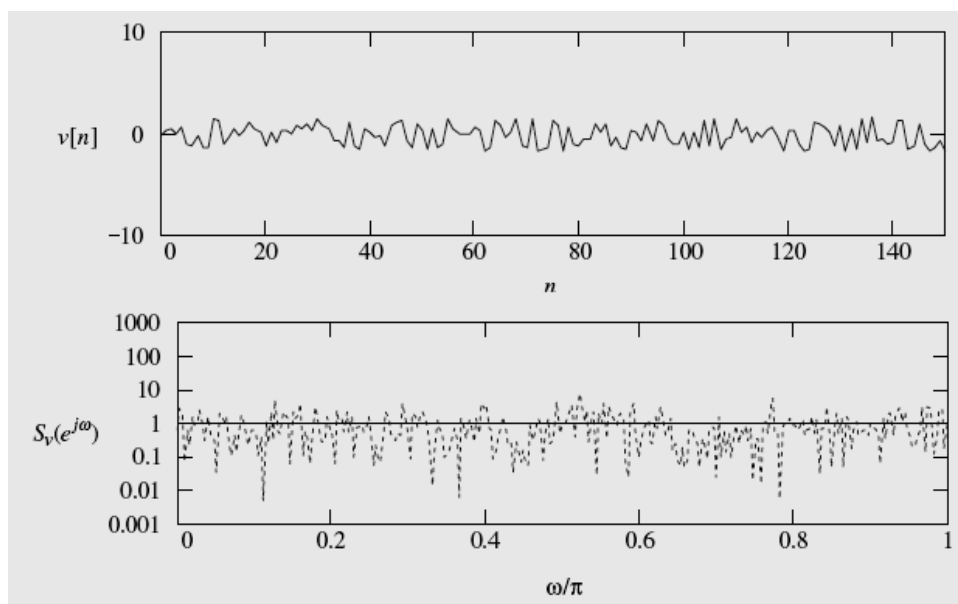


Fig. 3.1 Señal de entrada (arriba) y sus correspondientes (abajo) PSD (sólida) y Periodograma (punteada) (4)

### 3.4 Modelo Autoregresivo

El modelo autoregresivo proporciona los fundamentos de la predicción lineal (que se verá mas adelante). Y a su vez, la predicción lineal constituye una parte fundamental de los codificadores basados en el esquema FS1016 CELP, el cual será usado en el presente trabajo, tal como se mostrará en el capítulo VI. Una ecuación importante para entender las bases del modelo autoregresivo, la cual se deriva de las ecuaciones (3.8), es la siguiente:

$$S_Y(e^{j\omega}) = |H(e^{j\omega})|^2 S_X(e^{j\omega}) \quad (3.12)$$

Como se ha explicado,  $y[n]$  representa la salida de un sistema lineal representado por  $h[n]$  y con entrada igual a  $x[n]$ . Si hacemos  $x[n]$  igual a una secuencia de ruido blanco aleatorio, la densidad espectral de potencia de la salida del sistema lineal esta dada por el espectro del ruido blanco (constante) multiplicado por la magnitud al cuadrado de la respuesta del filtro, entonces se pueden producir señales aleatorias con una deseada característica espectral mediante la adecuada selección de la función de transferencia del sistema lineal  $h[n]$ .

Los valores de la secuencia (proceso estocástico estacionario en el sentido amplio)  $x[n]$ ,  $x[n-1]$ , ...,  $x[n-M]$  representan la realización de un proceso autoregresivo (AR) de orden M si este satisface la ecuación:

$$x[n] = -a_1x[n-1] - a_2x[n-2] - \dots - a_Mx[n-M] + v[n] \quad (3.13)$$

El valor presente del proceso  $x[n]$ , es igual a la combinación lineal de valores pasados del proceso mas una señal de error  $v[n]$ , la cual se muestra en la ecuación (3.13). Los coeficientes  $a_i$  conforman la función de transferencia del sistema lineal  $H_A(z)$ .

La función de transferencia de un analizador de un proceso AR es:

$$H_A(z) = \frac{V(z)}{X(z)} = \sum_{i=0}^M a_i z^{-i} \quad (3.14)$$

Donde  $V(z)$  es la transformada Z de la secuencia de error  $v[n]$ .

La función de transferencia de un sintetizador de un proceso AR es:

$$H_S(z) = \frac{X(z)}{V(z)} = \frac{1}{H_A(z)} = \frac{1}{\sum_{i=0}^M a_i z^{-i}} \quad (3.15)$$

En base al modelo autoregresivo, el objetivo ahora es conseguir los coeficientes  $a_i$  de tal manera que la secuencia de error  $v[n]$  se convierta en una secuencia de ruido blanco aleatorio.

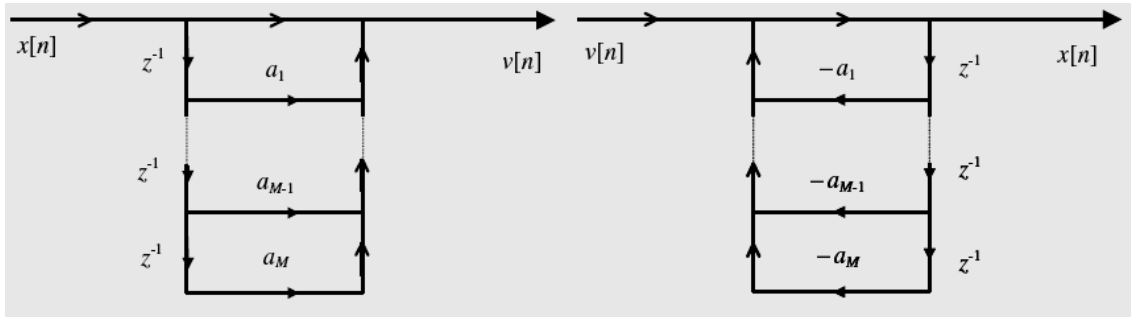


Fig. 3.2 Realizaciones de los filtros AR (analizador y sintetizador) (4)

### 3.4.1 Ecuación Normal o de Yule-Walker

Cuando se consigue que el error  $v[n]$  represente ruido blanco, entonces éste no estará correlacionado con  $x[n-l]$  para  $l \geq 1$ , es decir:

$$E\{v[n]x[n-l]\} = 0 \quad (3.16)$$

Multiplicando ambos lados de (3.16) por  $v[n]$  y tomando esperanzas.

$$E\{v[n]x[n]\} = \sigma_v^2 \quad (3.17)$$

De la última ecuación se tiene que la correlación cruzada entre  $x[n]$  y  $v[n]$  está dada por la varianza de  $v[n]$ . Multiplicando otra vez ambos lados de (3.15) por  $x[n-l]$ ,  $l=0, 1, \dots, M$  y tomando la esperanza se obtiene la ecuación de Yule Walker (3.29).

### 3.4.2 Estimación de la Autocorrelación

Como se sabe, el periodograma es un estimado de la PSD. También se sabe que la función de autocorrelación y la PSD forman un par de transformada de Fourier (ver ecuaciones (3.6)). Basado en este hecho, la autocorrelación puede ser estimada primeramente de la señal, entonces se calcula la transformada de Fourier para la estimación del espectro. Como se verá luego, la función de autocorrelación juega un papel importante en el análisis de predicción lineal, el cual es un procedimiento para calcular los coeficientes de predicción lineal del modelo predictivo (ver sección 3.5).

Por ello, es importante escoger un método adecuado que nos permita estimar de forma óptima el valor de la autocorrelación, el cual debe de ser estimado para cada



segmento de voz el cual ahora será representado por la notación  $s[n]$ . En adición a lo presentado en la sección 3.1.1, los métodos para la estimación de la autocorrelación pueden dividirse en recursivos y no recursivos, dependiendo del tipo de ventana que se use para la extracción de las muestras finitas.

Para una ventana de tamaño  $N$ , tenemos que la autocorrelación puede ser estimada de la siguiente ecuación:

$$R[l, m] = \frac{1}{N} \sum_{n=m-N+1+|l|}^m x[n]w[m-n]x[n-|l|]w[m-n+|l|] \quad (3.18)$$

En el presente trabajo se usa la ecuación (3.18) haciendo uso de una ventana de Hamming:

$$w[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \leq n \leq N-1 \quad (3.19)$$

### 3.5 Predicción lineal y técnicas de predicción lineal

La predicción lineal (LP) constituye la base de casi todos los modernos codificadores de voz de hoy en día. Básicamente consiste en que las propiedades estadísticas de un segmento actual pueden ser modelados basados en las propiedades de un segmento anterior. Se basa en el modelo autoregresivo (AR), de hecho el análisis de predicción lineal es un procedimiento de estimación para encontrar los parámetros del modelo AR, dadas las muestras de la señal.

#### 3.5.1 Predicción Lineal

Se ilustra en la Fig. 3.3. La señal de ruido blanco  $x[n]$  es filtrada por el proceso sintetizador AR para obtener la señal  $s[n]$  (la señal AR) con los parámetros AR denotados por  $\hat{a}_i$ .

Un predictor lineal es usado para predecir  $s[n]$  basado en las  $M$  muestras pasadas mediante la siguiente ecuación:

$$\hat{s}[n] = -\sum_{i=1}^M a_i s[n-i] \quad (3.20)$$

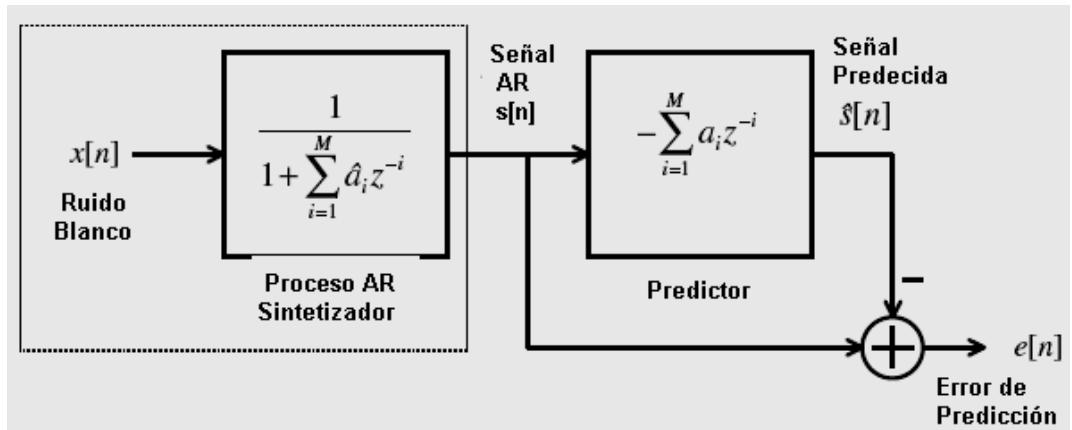


Fig. 3.3 Esquema de análisis de la Predicción Lineal (4)

Donde los  $\hat{a}_i$  son los estimados de los parámetros AR y son referidos como los coeficientes LPC.  $M$  representa el orden del predictor.

El error de predicción es:

$$e[n] = s[n] - \hat{s}[n] \quad (3.21)$$

El objetivo del esquema de predicción lineal es minimizar el error  $e[n]$  entre la señal predicha  $\hat{s}[n]$  y la señal real  $s[n]$ .

La Fig. 3.4 muestra la implementación y es conocido como el filtro de predicción. Este filtro toma la señal AR como entrada y produce la señal de error de predicción y el error como sus salidas.

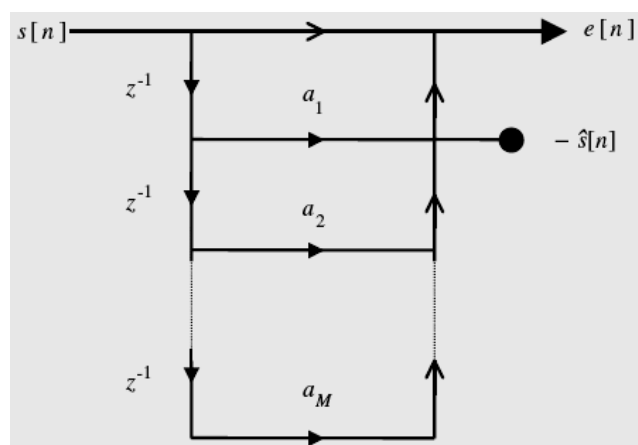


Fig. 3.4 Implementación del filtro de predicción.

### 3.5.2 Minimización del Error

Consiste en la estimación de los parámetros AR  $\hat{a}_i$  de  $s[n]$ , siendo estos los coeficientes LPC. Para ejecutar la estimación, la literatura usa el criterio de minimización del error de predicción cuadrático medio dado por la ecuación (3.22).

$$J = E\{e^2[n]\} = E\left\{\left(s[n] + \sum_{i=1}^M a_i s[n-i]\right)^2\right\} \quad (3.22)$$

Para minimizar la función J se deriva con respecto a los coeficientes predictivos.

$$\frac{\partial J}{\partial a_k} = 2E\left\{\left(s[n] + \sum_{i=1}^M a_i s[n-i]\right)s[n-k]\right\} = 0 \quad (3.23)$$

La solución de la ecuación (3.23) se da cuando los coeficientes predictivos resultantes son iguales a los correspondientes coeficientes del proceso AR  $\hat{a}_i$  (ver Fig. 3.4).

### 3.5.3 Ecuación Normal

De la ecuación (3.23) se deriva la siguiente ecuación (3.24):

$$\sum_{i=1}^M a_i R_s[i-k] = -R_s[k] \quad (3.24)$$

En forma matricial:  $\mathbf{R}_s \mathbf{a} = -\mathbf{r}_s$  (3.25)

Donde

$$\mathbf{R}_s = \begin{pmatrix} R_s[0] & R_s[1] & \cdots & R_s[M-1] \\ R_s[1] & R_s[0] & \cdots & R_s[M-2] \\ \vdots & \vdots & \ddots & \vdots \\ R_s[M-1] & R_s[M-2] & \cdots & R_s[0] \end{pmatrix}$$

$$\mathbf{a} = [a_1 \quad a_2 \quad \cdots \quad a_M]^T,$$

$$\mathbf{r}_s = [R_s[1] \quad R_s[2] \quad \cdots \quad R_s[M]]^T.$$

La ganancia predictiva de un predictor esta dada por la siguiente ecuación:

$$PG = 10 \log_{10} \left( \frac{\sigma_s^2}{\sigma_e^2} \right) = 10 \log_{10} \left( \frac{E\{s^2[n]\}}{E\{e^2[n]\}} \right) \quad (3.26)$$

Un predictor óptimo es capaz de reducir la ganancia predictiva dada por (3.26).

### 3.5.4 Mínimo Error Predictivo Cuadrático Medio

En la situación óptima donde el error de predicción es igual a una señal de ruido blanco aleatorio, se cumple:

$$J \min = E\{e^2[n]\} = E\{x^2[n]\} = \sigma_x^2 \quad (3.27)$$

El mínimo error predictivo basado en los coeficientes de autocorrelación resulta entonces:

$$J \min = \sigma_x^2 = R_s[0] + \sum_{i=1}^M a_i R_s[i] \quad (3.28)$$

Combinando la ecuación (3.27) con la (3.28) obtenemos la ecuación de Yule Walker:

$$\begin{bmatrix} R_s[0] & r_s^T \\ r_s & R_s \end{bmatrix} \begin{bmatrix} 1 \\ a \end{bmatrix} = \begin{bmatrix} J \min \\ 0 \end{bmatrix} \quad (3.29)$$

### 3.6 Esquemas de análisis predictivo

El análisis predictivo puede ser implementado de distinta manera dependiendo de los requerimientos del sistema en cuestión. Existen dos técnicas principales para el análisis predictivo de las señales de voz: predicción interna y predicción externa. Estos esquemas son ilustrados en la Fig. 3.5.

La razón por la cual la predicción externa puede ser usada es debido a que las propiedades estadísticas de la señal de voz cambian lentamente con el tiempo. Si el segmento no es excesivamente largo, sus propiedades pueden ser derivadas de segmentos anteriores.

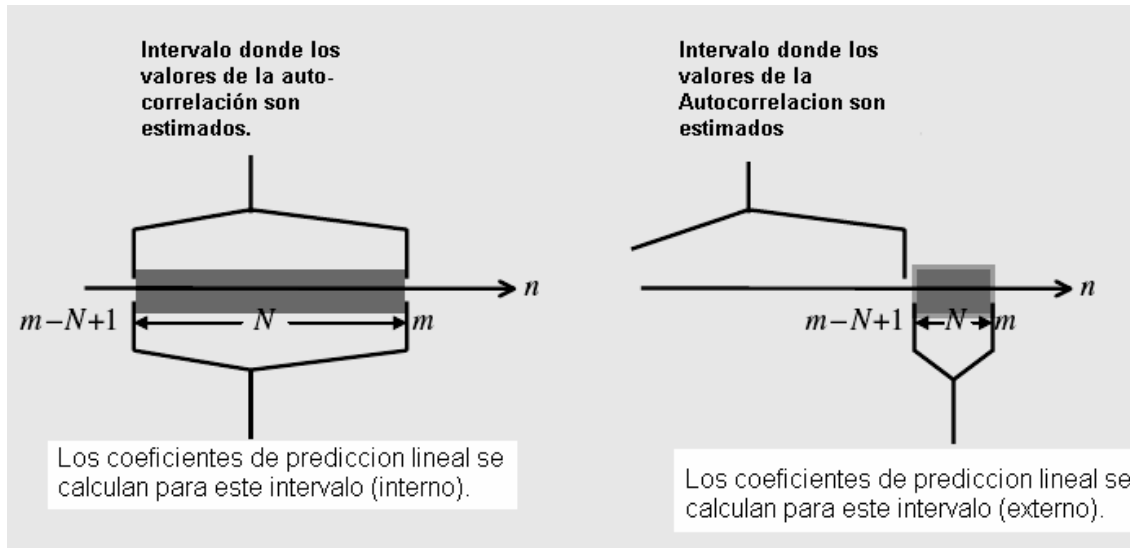


Fig. 3.5 Ilustración de la predicción interna (izquierda) y predicción externa (derecha) (4)

Varios algoritmos de codificación de voz usan predicción interna, donde los coeficientes de predicción lineal LPC (ver sección 3.5.2) de un segmento en particular son derivados de la data perteneciente a ese segmento. De esta forma, los coeficientes LPC resultantes capturan las propiedades estadísticas de ese segmento. Las longitudes típicas de los segmentos de análisis varían de 160 a 240 muestras.

La predicción externa es usada principalmente en algoritmos donde el retardo del codificador es el principal problema. En este caso se puede usar una trama más corta (del orden de 20 muestras, por ejemplo en el estándar LD-CELP). En estos casos se usan técnicas recursivas para el calculo de la estimación de la autocorrelación de tal forma que los coeficientes LPC son derivados de las muestras comprendidas antes del instante  $n=m-N+1$ . En el presente trabajo se usa el esquema de predicción interna.

### 3.7 Ganancia de Predicción

La ganancia de predicción está dada por la siguiente expresión,:

$$PG[m] = 10 \log_{10} \left( \frac{\sum_{n=m-N+1}^m s^2[n]}{\sum_{n=m-N+1}^m e^2[n]} \right) \quad (3.30)$$

Donde:

$$e[n] = s[n] - \hat{s}[n] = s[n] + \sum_{i=1}^M a_i[m] s[n-i], \quad n=m-M+1, \dots, m$$

La ecuación (3.30) es básicamente la misma que la (3.26), solo que ahora se considera la naturaleza finita del segmento de voz de análisis, en este caso es de  $N$  muestras.

Del esquema de predicción lineal se derivan observaciones muy importantes las cuales se resumen a continuación [4]:

- *Para un orden de predicción dado, la ganancia predictiva promedio dado por (3.30) es mayor para segmentos “sonoros” que para segmentos “sordos”. Esto es debido que los segmentos sonoros tienen periodicidad, la cual no tiene correlación con la señal de ruido blanco aleatorio.*
- *Para un segmento sonoro, la ganancia de predicción asociada con un predictor que tiene un orden lo suficientemente grande como para cubrir un periodo de pitch es substancialmente más grande que la ganancia de predicción asociada a un predictor con un orden más pequeño que un periodo de pitch. Esto quiere decir que para representar correctamente un segmento sonoro con alta periodicidad, se necesitan un orden del filtro predictor  $M$  (ver Fig. 3.3) mas grande que el periodo pitch.*
- *Para remover la correlación entre muestras usando un predictor lineal, se requiere un orden de predicción mayor para segmentos “sonoros” que para segmentos “sordos”. Para “blanquear” eficientemente un segmento “sonoro” se requiere que el orden del predictor sea mayor o igual al periodo pitch de dicho segmento de voz. Para mayor referencia al respecto, revisar (4).*

### 3.8 Coeficientes de Reflexión

Los coeficientes de reflexión (RC)  $K_i$  son los coeficientes del filtro predictivo mostrado en la Fig. 3.4 (el cual esta en una estructura directa), cuando se transforma en una estructura de filtro Lattice, tal como se muestra en la Fig. 3.6. Los coeficientes RC pueden ser usados para evaluar la estabilidad del sistema (basados en la propiedad de fase mínima). Tal como se verá en las siguientes secciones, la transformación entre coeficientes RC y LPC se realiza de manera directa mediante algoritmos recursivos.

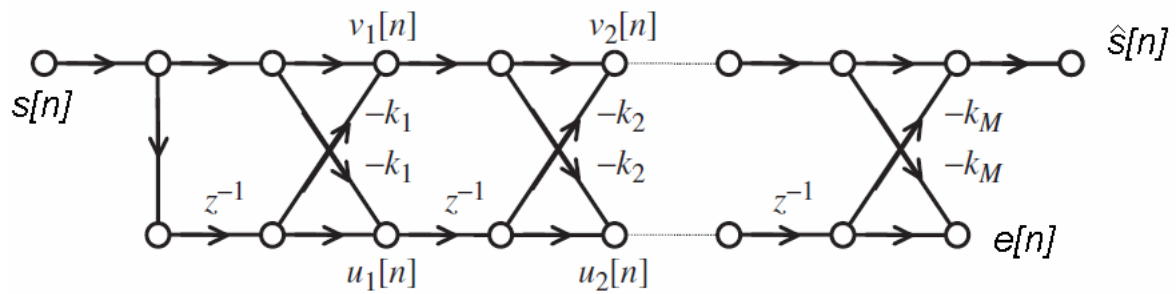


Fig. 3.6 Estructura Lattice para un filtro predictivo.

### 3.9 Algoritmo de Levinson-Durbin

La ecuación de Yule Walker dada en (3.29) puede ser resuelta encontrando la inversa de la matriz  $\mathbf{R}_s$ . En general, la inversión de una matriz es computacionalmente exigente. Afortunadamente, la ecuación matricial puede ser resuelta por algoritmos eficientes pasando por alto la inversión de dicha matriz. Dichos algoritmos aprovechan la estructura especial de la matriz de correlación.

El algoritmo de Levinson-Durbin se vale de dos propiedades claves de la matriz de autocorrelación:

- La matriz de correlación de un tamaño determinado contiene como subbloques todas las matrices de órdenes más bajos.
- La matriz de correlación es invariante a intercambios entre sus columnas o filas.

Las propiedades mencionadas corresponden a las matrices Toeplitz.

Entonces, basados en la explotación de la matriz Toeplitz, el algoritmo de Levinson – Durbin nos da como salidas los coeficientes LPC y RC..

- Inicialización:  $l = 0$ ,  
 $J_0 = R[0]$ , donde  $J_0$  es el primer error de predicción cuadrático medio
- Recursión para  $l = 1, 2, 3, \dots, M$

Paso 1. Calcular el  $l$ -ésimo  $K_l$  (coeficiente de reflexión)

$$k_l = \frac{1}{J_{l-1}} \left( R[l] + \sum_{i=1}^{l-1} a_i^{(l-1)} R[l-i] \right) \quad (3.31)$$

Paso 2. Calcular los LPC's para el predictor de orden  $l$ .

$$a_l^{(l)} = -k_l$$

$$a_i^{(l)} = a_i^{(l-1)} - k_l a_{l-i}^{(l-1)}, \quad i=1,2,\dots,l-1 \quad (3.32)$$

Detener si  $l = M$ .

Paso 3. Calcular el **mínimo** error de predicción medio cuadrático asociado con el predictor de orden  $l$ .

$$J_l = J_{l-1} (1 - k_l^2) = \sigma_X^2 \quad (3.33)$$

$l = l + 1$  y regresar al paso 1.

- El conjunto final de coeficientes LPC esta dado por

$$a_i = a_i^{(M)}, \quad i=1,2,\dots,M \quad (3.34)$$

Observar que en el proceso de solución, el conjunto de coeficientes de reflexión RC también es encontrado. Observar también que de (3.33) se obtiene la energía de la señal de ruido.

Características del Algoritmo de Levinson – Durbin:

- Es computacionalmente eficiente.
- Los coeficientes RC pueden ser usados para evaluar la estabilidad del sistema (basados en la propiedad de fase mínima).

El filtro de predicción de error con función de transferencia



$$A(z) = 1 + \sum_{i=1}^M a_i z^{-i} \quad (3.35)$$

Es un sistema de fase mínima si se cumple:  $|k_i| < 1, i=1,2,\dots,M$

### 3.9.1 Conversión de los Coeficientes de Reflexión a Coeficientes LPC

Como se mencionó anteriormente, los coeficientes RC representan una forma alternativa de los coeficientes LPC. De hecho hay una correspondencia uno-a-uno entre ellos. Los coeficientes RC poseen varias características deseables, haciéndolos preferibles en varias situaciones prácticas. Las ecuaciones para convertir los coeficientes RC a coeficientes LPC son:

$$a_i^{(l)} = -k_l$$

$$a_i^{(l)} = a_i^{(l-1)} - k_l a_{l-i}^{(l-1)}, \quad i = 1, 2, \dots, l-1 \quad (3.36)$$

### 3.9.2 Conversión de los coeficientes LPC a coeficientes RC

Dado el conjunto de coeficientes LPC  $a_i, i = 1, 2, \dots, M$ , se desea encontrar los correspondientes coeficientes RC  $k_i$ . Las ecuaciones que definen la conversión se da en (3.37).

$$k_l = -a_l^{(l)} \quad (3.37)$$

$$a_i^{(l-1)} = \frac{a_i^{(l)} + k_l a_{l-i}^{(l)}}{1 - k_l^2}, \quad i = 1, 2, \dots, l-1 \quad (3.38)$$

### 3.10 Predicción Lineal de Largo Plazo (*Long-Term*)

Cuando se resuelve la ecuación para el modelo autoregresivo se encuentra que para modelar señales "sonoras" se necesita un orden alto para el filtro predictivo, lo cual es desventajoso. El problema radica en que ordenes altos de predictores resultan en excesivas tasas de bits de la voz codificada, lo cual no es deseable. Para superar el problema mencionado la literatura plantea el esquema de predicción de largo plazo.

Dicho esquema se muestra en la Fig. 3.7. El filtro de predicción de error de largo plazo con entrada  $e_s[n]$  y salida  $e[n]$  tiene la siguiente función de transferencia:

$$H(z) = 1 + bz^{-T} \quad (3.39)$$

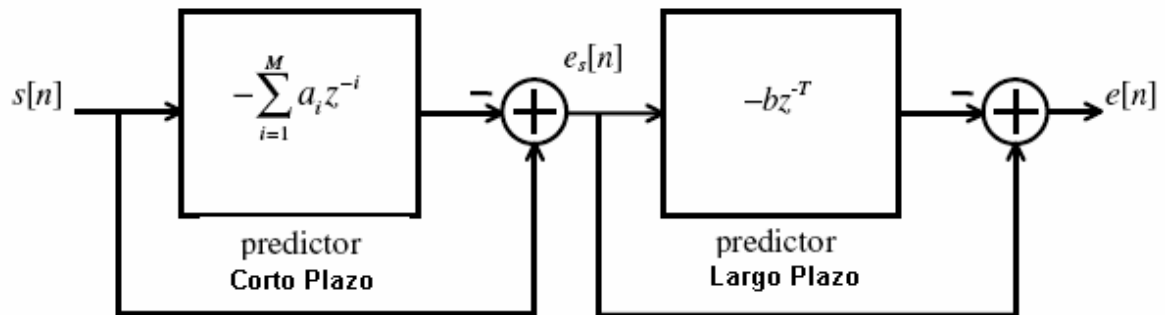


Fig. 3.7 Esquema de la predicción de Largo Plazo (*Long – Term*) (4)

EL predictor short-term también es conocido como filtro de frecuencias formantes de la voz, mientras que el predictor *Long-Term* también es conocido como filtro de frecuencia fundamental o *pitch*.

El predictor de largo plazo requiere el cálculo de dos nuevos coeficientes: el periodo del pitch T y la ganancia long-term b.

$$\hat{e}_s[n] = -be_s[n-T] \quad (3.40)$$

Para encontrar b y T se recurre a la minimización del error cuadrático:

$$J = \sum_n \left( e_s[n] - \hat{e}_s[n] \right)^2 = \sum_n \left( e_s[n] + be_s[n-T] \right)^2 \quad (3.41)$$

Diferenciando la ecuación (3.41) con respecto a b resulta en:

$$b = -\frac{\sum_n e_s[n]e_s[n-T]}{\sum_n e_s^2[n-T]} \quad (3.42)$$

Reemplazando (3.42) en (3.41), obtenemos el valor de J a ser minimizado:

$$J = \sum_n e_s^2[n] - \frac{\left( \sum_n e_s[n]e_s[n-T] \right)^2}{\sum_n e_s^2[n-T]} \quad (3.43)$$

Se hace un barrido de los valores de T en el rango de 20 a 147 (el estándar FS1016 establece que el rango de 20 a 147 es suficiente para barrer todos los pitches tanto masculinos como femeninos) y se escoge aquel que minimiza (3.43).

La efectividad del predictor long-term mejora cuando se usa un esquema de análisis basado en sub-segmentos. Dicho esquema es usado por varios codificadores entre ellos los basados en el estándar FS1016 (CELP). La gráfica se muestra en la Fig. 3.8.

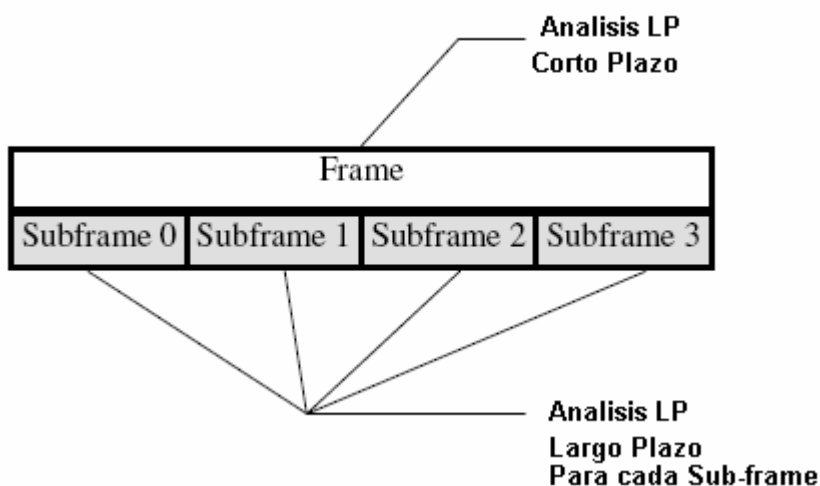


Fig. 3.8 Esquema de análisis LP basado en Sub-segmentos (4)

Como se puede apreciar en la Fig. 3.8, el análisis predictivo convencional (de corto plazo) se realiza para el segmento mayor, mientras que para cada sub-segmento se realiza una análisis predictivo de largo plazo. Esto se explica con más detalle en el capítulo VI.

### 3.11 Filtros de Síntesis

Si el proceso de predicción es eficiente en conseguir que la señal de error  $e[n]$  tenga un alto grado de “blanqueamiento” (que sea lo más cercano posible al ruido blanco aleatorio), entonces para recuperar la señal en el sintetizador solamente hace falta filtrar una señal generada por un generador de ruido blanco aleatorio a través de un filtro con función de transferencia (ver Fig. 3.9):

$$H(z) = \frac{1}{1 + \sum_{i=1}^M a_i z^{-i}} \quad (3.44)$$

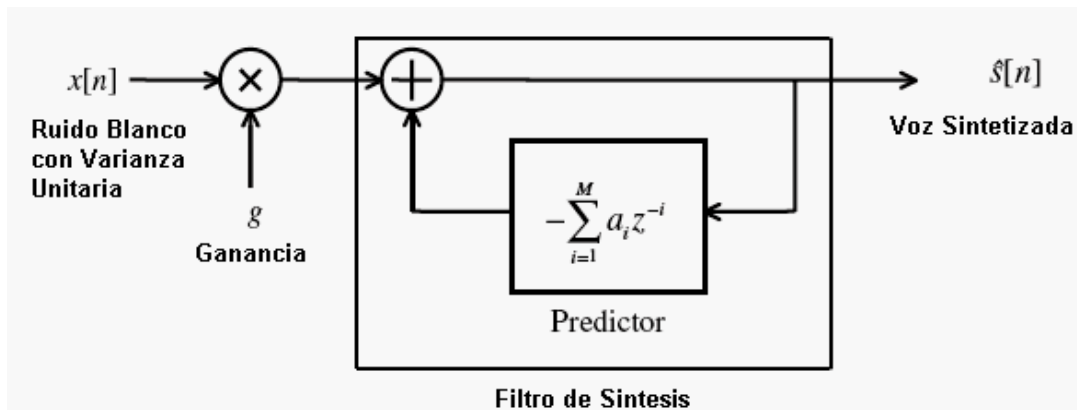


Fig. 3.9 Filtro de síntesis (4)

La ganancia se calcula mediante la siguiente ecuación, la cual se deriva de la (3.33) junto con la (3.28).

$$g = \gamma \sqrt{R_s[0] + \sum_{i=1}^M a_i R_s[i]} \quad (3.45)$$

Donde:  $\gamma$  es una constante en el rango de 1 a 2.

En el caso de predicción lineal de largo plazo, el respectivo filtro de síntesis se muestra en la Fig. 3.10 (usado en el esquema CELP).

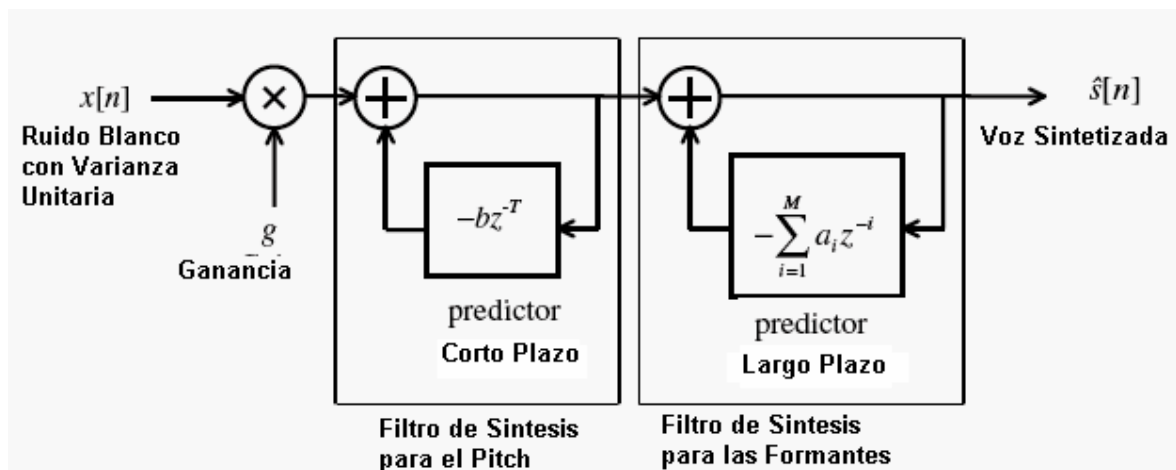


Fig. 3.10 Filtro de Síntesis para el caso de Predicción de Largo Plazo (4)

En la Fig. 3.10, recordar que el predictor *Long-term* también es conocido como filtro de frecuencia fundamental, debido a que este filtro produce la periodicidad de la voz a una frecuencia  $T$ . Por otro lado, el predictor *short-term* es el filtro de las frecuencias formantes de la voz. Ambos filtros producen la voz sintetizada con características muy similares a la voz original.

### **3.12 Cuantización escalar de los coeficientes predictivos**

Antes de transmitir los parámetros que representan la voz decodificada, estos deben de ser cuantizados de manera apropiada para garantizar que el decodificador pueda recuperar los valores con máxima exactitud y cumplir los requerimientos sobretodo de estabilidad de los filtros de síntesis. Existen varios esquemas de cuantización. El que será usado en el presente trabajo de tesis es el esquema basado en los coeficientes LSP (Linea Spectra) y se describe en el capítulo VI.

## CAPITULO IV

### PREDICCIÓN LINEAL CON EXCITACIÓN DE CODIGO

La idea de la predicción lineal con excitación de código (Code Excited Linear Prediction - CELP) nació como otro intento para mejorar los codificadores basados en el esquema LPC-10 (FS 1015 y derivados). Entre las ideas claves de este nuevo esquema está el desarrollar un mecanismo de excitación adaptiva (el cual se basa en la predicción lineal de largo plazo) para el cálculo de la frecuencia fundamental y de esta forma evitar la estricta clasificación de los segmentos como “sonoros” y “sordos”; y la incorporación de un *codebook* de excitación de los filtros (como se describirá mas adelante), el cual es buscado durante la etapa de codificación para localizar la secuencia de excitación que logre capturar mejor los parámetros de la voz. (4)

El esquema CELP es uno de los más exitosos en el campo de la codificación de la voz ya que muchos codificadores estandarizados que se usan en la actualidad se basan en dicho esquema.

#### 4.1 El modelo de producción del esquema CELP

El modelo básico se muestra en la Fig. 4.1, donde se observa que una secuencia de excitación es extraída de un *codebook* a través de un índice. Un *codebook* es simplemente un arreglo (o vector) de un determinado tamaño, que almacena ciertos valores los cuales pueden ser constantes o producidos de forma dinámica. La excitación extraída es escalada a través de una ganancia apropiada y luego es filtrada por una conexión en cascada de un filtro de síntesis para el pitch y otro filtro de síntesis para las formantes (ver Capítulos 3.10 y 3.11). El filtro de síntesis para el pitch crea la periodicidad necesaria en la señal a la correspondiente frecuencia fundamental, y filtro de síntesis de las formantes genera la envolvente espectral.

¿Qué tipo de señal está contenida en el *codebook* de excitación? El *codebook* puede ser fijo o adaptivo y puede contener señales determinísticas o ruido aleatorio. Así,

el índice de excitación selecciona una secuencia de ruido blanco del *codebook* como entrada para la conexión en cascada de los filtros. (4)

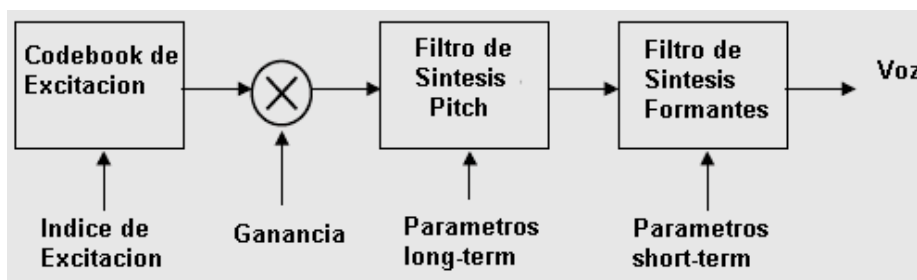


Fig. 4.1 Diagrama básico del modelo de producción del esquema CELP (4), (26)

Se pueden nombrar algunas ventajas del modelo CELP:

- La clasificación estricta de segmentos como “sonoros” y “sordos” es eliminada.
- Se preserva información parcial de la fase de la señal.

El modelo LPC no retiene ninguna información de la fase de la señal original. El modelo CELP captura información de la fase a través de un análisis de lazo cerrado (descrito luego). Al hacer esto, la señal de voz sintética no solo se asemeja en espectro sino también en el dominio del tiempo. Dicho esquema añade más naturalidad a la voz sintética, lo que conlleva a una mejora de la calidad. (4)

## 4.2 El principio de Análisis por Síntesis

En un codificador de voz, la señal de voz es representada por una combinación de parámetros: ganancia, coeficientes del filtro, etc. En un esquema de lazo abierto, los parámetros son extraídos de la señal de entrada los cuales son cuantizados y luego usados por el sintetizador. Un método más eficiente es usar los parámetros para sintetizar la señal durante la codificación de la misma y afinarla de tal forma que podamos generar la reconstrucción mas precisa de la señal original. Conceptualmente esto es llamado optimización por lazo cerrado, donde la meta es escoger los mejores parámetros de tal forma de acercar la voz sintética lo más que se pueda a la voz original. La Fig. 5.2 muestra el diagrama de bloques de un codificador con el esquema de lazo cerrado.

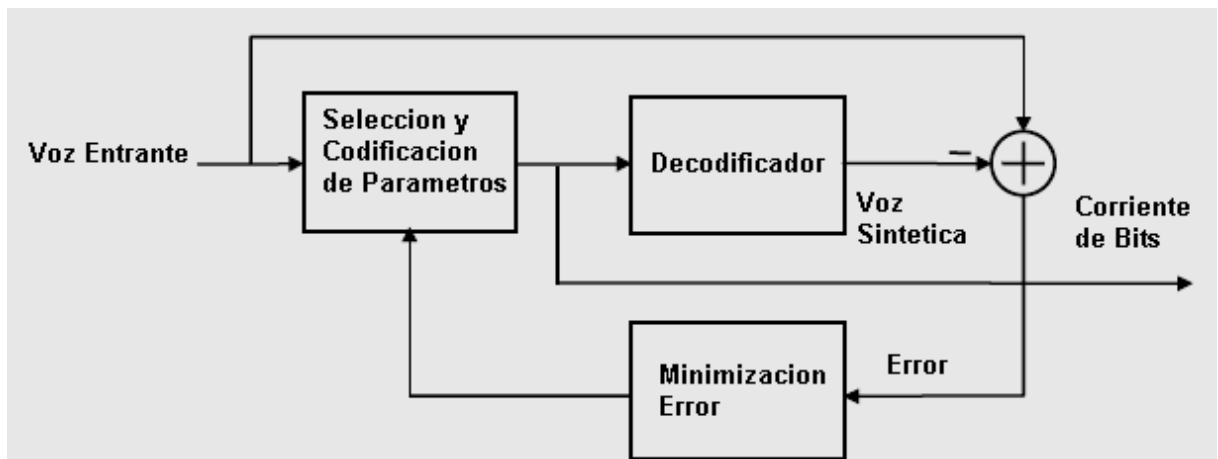


Fig. 4.2 Codificador basado en el esquema de Análisis por Síntesis (4, 26)

Teóricamente, todos los parámetros (los cuales se describen en el capítulo 6) del codificador pueden ser optimizados conjuntamente con el fin de conducir a los mejores resultados. Sin embargo los requerimientos computacionales son muy altos. En la práctica, solo un subconjunto de parámetros es seleccionado para una optimización de lazo cerrado, mientras el resto de parámetros son determinados en un análisis de lazo abierto. La Fig. 4.3 ilustra un bloque simplificado de un codificador CELP. Para la selección de la secuencia final de excitación se usa el criterio de la mínima suma del error cuadrático. En este esquema se rescatan características de la forma de onda (dominio del tiempo) de la señal de voz original, lo que resulta además en una conservación parcial de la fase.

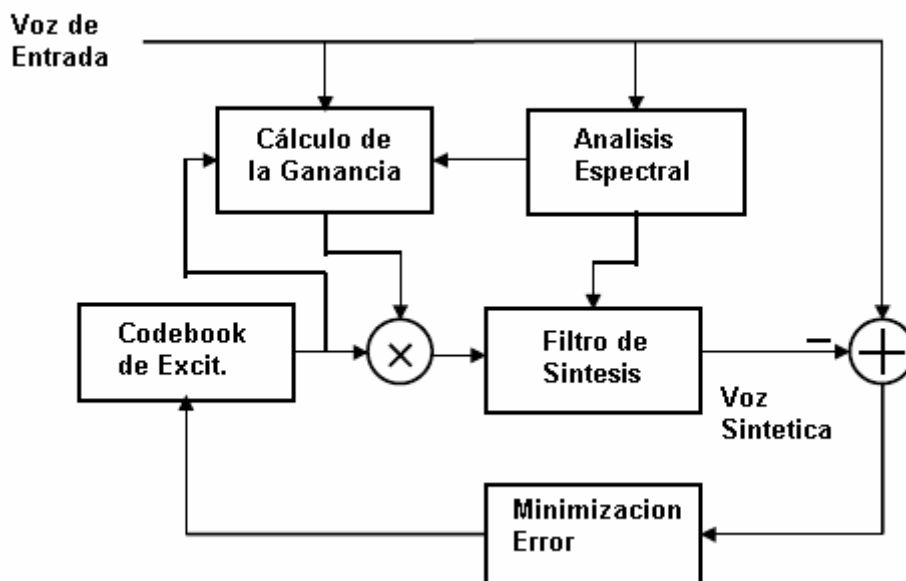


Fig. 4.3 Esquema simplificado de un Codificador CELP. (4, 26)



### 4.3 Codificación y Decodificación

El esquema CELP es un método de análisis por síntesis. En la parte codificadora, la secuencia de excitación es seleccionada mediante un procedimiento de búsqueda de lazo cerrado el cual es luego aplicado a un filtro de síntesis. El índice de la mejor secuencia de excitación es después transmitido al decodificador, en el cual se recupera la secuencia de excitación por medio de ese índice. El esquema de análisis por síntesis de lazo cerrado en el cual se basan los codificadores CELP se muestra en la Fig. 4.4.

### 4.4 Ponderado Perceptual (Perceptual Weighting)

En el diagrama de la Fig. 4.4, tenemos que la función de transferencia del filtro de las formantes  $H_f(z)$  es (ver secciones 3.10 y 3.11):

$$H_f(z) = \frac{1}{A(z)} = \frac{1}{1 + \sum_{i=1}^M a_i z^{-i}} \quad (4.1)$$

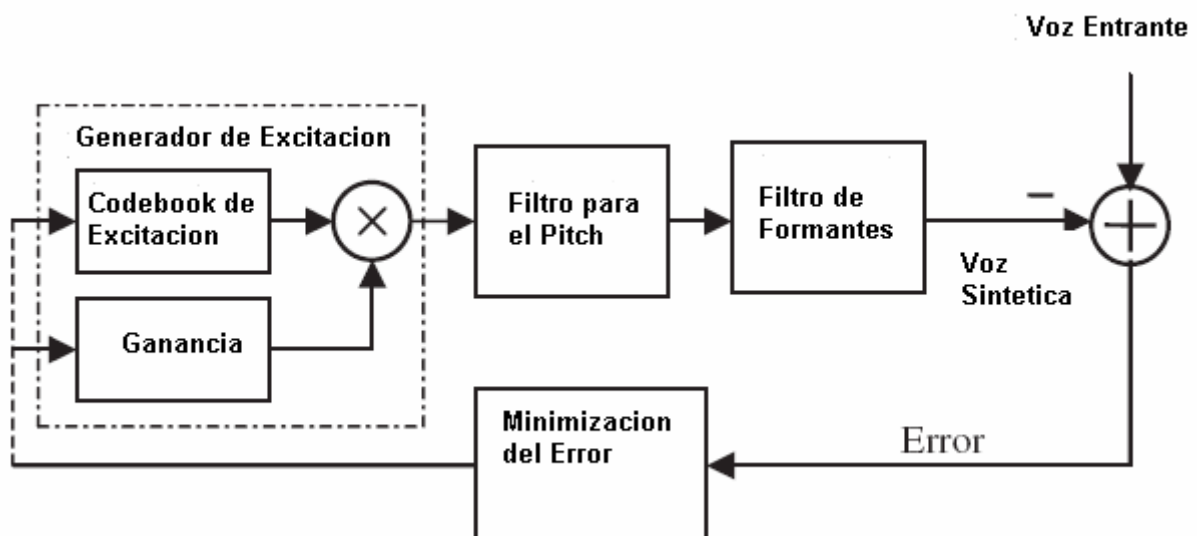


Fig. 4.4 Esquema de análisis por síntesis para un Codificador CELP (4, 26)

Se asume que el tamaño del *codebook* de excitación es  $L$ , es decir, contiene un total de  $L$  vectores de excitación. El codificador entonces, pasará a través del lazo  $L$  veces para cada segmento corto de voz de voz entrante; en cada paso por el lazo se calcula el error cuadrático medio. El índice de la secuencia de excitación que presenta el menor error es seleccionado.

El fenómeno de enmascaramiento en el sistema de audición humano (ver Cap. 2) puede ser explotado para conducir a una medida más precisa del error. Es bien conocido

que nosotros podemos tolerar mucho mas ruido en regiones de frecuencia donde la voz tiene una cantidad de energía significativa, además, los componentes del ruido en estas regiones pueden tener energía más alta relativa a los componentes del ruido en las regiones de baja energía de la voz sin incrementar la distorsión perceptual. En otras palabras, las regiones de frecuencia donde la voz tiene energía más alta pueden enmascarar proporcionalmente una cantidad más alta de ruido. Además, en un espectro típico, la cantidad de ruido en los picos puede ser mas alto que la cantidad de ruido en los valles. Una manera simple de controlar el espectro del ruido es mediante el filtrado del mismo a través de un filtro de ponderación (weighting filter o filtro W) antes de la minimización. Dicho filtro W tiene la siguiente función de transferencia[4]:

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 + \sum_{i=1}^M a_i z^{-i}}{1 + \sum_{i=1}^M a_i \gamma^i z^{-i}} \quad (4.2)$$

Donde el factor gama es una constante perteneciente al rango entre 0 y 1. La posición de este filtro se muestra en la Fig. 4.5.

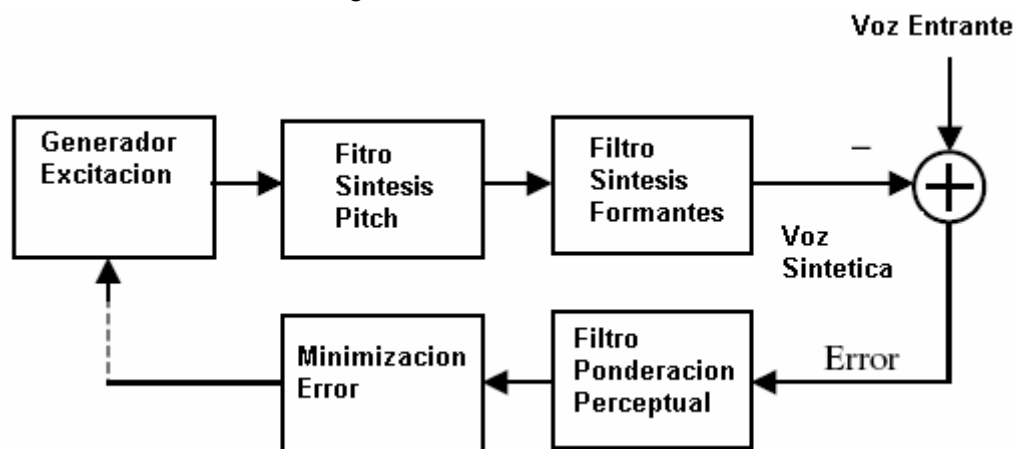


Fig. 4.5 Esquema de análisis por síntesis para un Codificador CELP con ponderación perceptual. (4, 26)

El filtro de ponderación perceptual (W) amplifica la señal de error en regiones no-formantes del espectro de la voz y a la vez atenúa el espectro del error en las regiones formantes de la voz. Aquí, una señal de error cuya energía espectral está concentrada en las regiones formantes del espectro entrante es considerada mejor que una cuya energía espectral no está localizada bajo las formantes. El efecto del filtro W es el de un efecto de

expansión. Para el siguiente conjunto de coeficientes LPC se muestran los espectros de magnitud Fig. 4.6 y el filtro W asociado para diferentes valores de gama (Fig. 4.7).

$$\begin{array}{lllll} a_1=-1.286, & a_2=1.138, & a_3=-1.047, & a_4=0.691, & a_5=-0.304 \\ a_6=0.373, & a_7=-0.071, & a_8=0.012, & a_9=0.048, & a_{10}=0.064 \end{array}$$

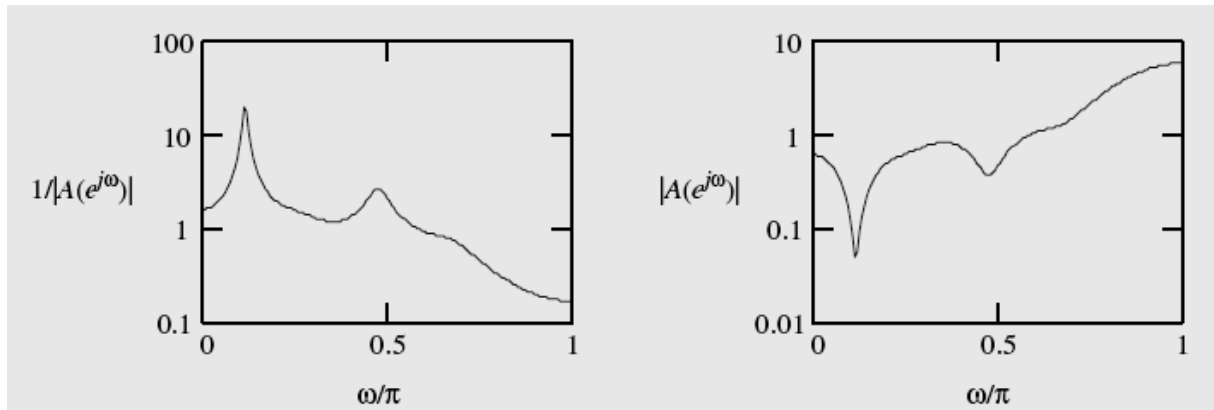


Fig. 4.6 Filtro de síntesis (izquierda) y filtro de análisis (derecha) de las formantes (4)

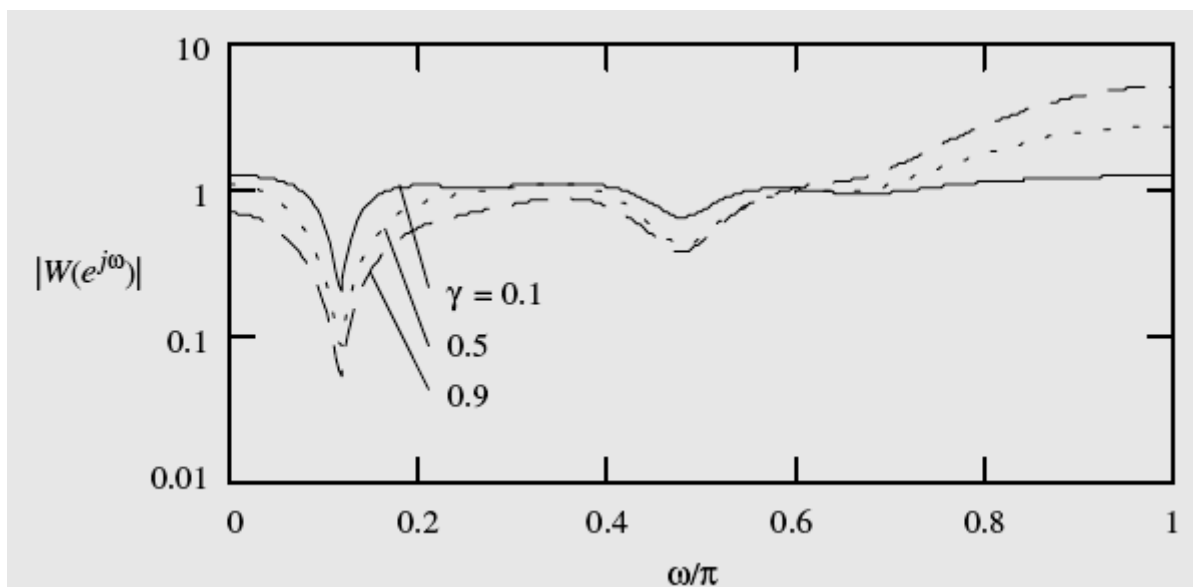


Fig. 4.7 Filtro de ponderación perceptual para diferentes valores del factor gama. (4)

#### 4.5 Búsqueda del Codebook de Excitación Optimo

La parte más exigente computacionalmente es la búsqueda de la mejor secuencia de excitación y a través de los años ha sido materia de estudio e investigación, en grandes rasgos el proceso de búsqueda de la mejor secuencia de excitación se describe a continuación:

- **Primero.** Se filtra el subsegmento de voz con el filtro de ponderación ( $W$ ).
- **Segundo.** Para cada vector de excitación del codebook:
  - Calcular la ganancia óptima y escalar el vector de excitación con dicho valor calculado.
  - Filtrar el vector de excitación escalado con el filtro de síntesis del pitch.
  - Filtrar la salida del filtro de síntesis del pitch con el filtro de síntesis de las formantes y luego con el filtro de ponderación.
  - Restar el subsegmento de voz filtrado con el filtro  $W$  de la salida del filtro  $W$  (paso anterior). El resultado es la secuencia de error.
  - Calcular la energía de la secuencia de error.
- **Tercero.** El índice del vector de excitación asociado con la energía de error mas bajo es retenido como información para dicho sub-segmento de voz.

El desarrollo detallado del codificador y de los algoritmos involucrados se dará en el capítulo 6 donde se verá el detalle del código implementado así como también la teoría subyacente correspondiente.

## **CAPITULO V**

### **CORRECCION DE ERRORES (FEC)**

Para ser usado en un sistema de comunicaciones real, el codificador de voz debe incluir alguna clase de método de corrección y/o detección de errores que puedan ocurrir durante su transmisión.

El estandar FS 1016 hace uso de la corrección de error hacia delante FEC (Forward error correction) mediante el código Hamming (15,11) de un solo bit, el cual puede detectar y corregir solamente un bit de un grupo de 11 bits protegidos (26). Esto es, el estandar selecciona un grupo de 11 bits considerados cruciales (pitch delay, ganancias adaptivas y estocásticas, etc) y luego los introduce en el esquema Hamming (15,11). Para el presente trabajo se mejoró el esquema de protección Hamming mediante el uso de la configuración (63, 57). Si bien es cierto el código Hamming de un solo bit permite detectar y corregir errores de un bit, también permite detectar errores de dos bits, sin embargo no los puede corregir (ver Comunicación de Datos, Redes de Computadoras y Sistemas Abiertos, Apéndice A, de Fred Halsall). Con el esquema Hamming (63, 57) se consigue proteger un rango más amplio de bits (57 bits) y con un muy leve incremento de la tasa de bits entregado por el codificador. Los detalles se discuten a continuación.

#### **5.1 Control de Errores hacia delante (FEC)**

El esquema FEC consiste en agregar una cantidad suficiente de dígitos de verificación a una secuencia de datos, para que el receptor no solamente pueda detectar la presencia de errores sino también determinar su posición. Y como los mensajes se dan en forma binaria basta invertir el bit detectado para corregir el error.

En la práctica, el número de dígitos de verificación adicionales que se requiere para corregir errores es mucho más grande que el necesario para únicamente detectar errores.

## 5.2 Código Hamming de un solo bit

Es un esquema básico pero que aún se usa en varias aplicaciones. En la teoría de codificación, el término con que se describe la unidad de mensaje combinada, es decir bits de datos útiles mas bits de verificación, se conoce como palabra de código. El número mínimo de posiciones de bit en que difieren dos palabras de código válidas es la distancia de Hamming del código. En general en los códigos de bloque de  $k$  dígitos fuentes se codifica a modo de producir un bloque de  $n$  dígitos ( $n$  mayor que  $k$ ) de salida. Se dice que el codificador produce un código  $(n,k)$ . La razón  $n/k$  es la tasa del código o eficiencia del código.

## 5.3 Esquema Hamming usado por el estandar FS1016

El esquema empleado por el estandar FS1016 es el (15,11). Los 11 bits para ser protegidos seleccionados por el estandar son:

40, 41, 42, 92, 93, 94 los cuales corresponden a los primeros 3 MSB (Most Significant Bits) de los pitches delay correspondientes al primer y tercer sub-segmento.

47, 72, 99, 124 los cuales corresponden al primer MSB de cada ganancia del codebook adaptivo.

139 protección del bit de uso futuro. (26)

## 5.4 Esquema Hamming usado en el presente trabajo de tesis.

En el presente trabajo de tesis se decidió mejorar el esquema de protección mediante el cambio de esquema de codificación de (15, 11) el cual es usado por el estandar FS1016, al esquema (63, 57) por varias ventajosas razones:

- Se protege un mayor rango de bits.
- Se hace uso del bit reservado por el estandar (el de uso futuro).
- Se logran detectar errores de dos bits (pero no corregirlos).
- No hay mayor alteración de la tasa de bits.

Los bits protegidos se muestran en la tabla 5.1. Se puede apreciar que al usar el esquema (63, 57) se logran proteger los MSB de todos los parámetros involucrados. Como consecuencia, se logra proteger un rango más amplio de los parámetros enviados por el codificador

| <b>Parámetro a Proteger</b>     | <b>Bits protegidos</b> |
|---------------------------------|------------------------|
| lsp1                            | primer MSB             |
| lsp2                            | 2 primeros MSB         |
| lsp3                            | 2 primeros MSB         |
| lsp4                            | 2 primeros MSB         |
| lsp5                            | 2 primeros MSB         |
| lsp6                            | primer MSB             |
| lsp7                            | primer MSB             |
| lsp8                            | primer MSB             |
| lsp9                            | primer MSB             |
| lsp10                           | primer MSB             |
| Pitch T subsegmento 1           | 2 primeros MSB         |
| ganancia adaptiva sub-seg 1     | 2 primeros MSB         |
| Indice estocastico sub-seg 1    | 4 primeros MSB         |
| ganancia estocastica sub-seg 1  | 3 primeros MSB         |
| Pitch-delay T subsegmento 2     | 2 primeros MSB         |
| ganancia adaptiva sub-seg 2     | 2 primeros MSB         |
| Indice estocastico sub-seg 2    | 4 primeros MSB         |
| ganancia estocastica sub-seg 2  | 3 primeros MSB         |
| Pitch T subsegmento 3           | 2 primeros MSB         |
| ganancia adaptiva sub-seg 3     | 2 primeros MSB         |
| Indice estocastico sub-seg 3    | 4 primeros MSB         |
| ganancia estocastica sub-seg 3  | 3 primeros MSB         |
| Pitch-delay T subsegmento 4     | 2 primeros MSB         |
| ganancia adaptiva sub-seg 4     | 2 primeros MSB         |
| Indice estocastico sub-seg 4    | 3 primeros MSB         |
| ganancia estocastica sub-seg 4  | 3 primeros MSB         |
| <b>Total de bits protegidos</b> | <b>57</b>              |

Tabla 5.1 Bits protegidos con la codificación Hamming (63, 57)

Los bits totales empleados por la etapa FEC son 7 (6 mas un bit de paridad).

## CAPITULO VI

### CODIFICADOR IMPLEMENTADO: DETALLES Y RESULTADOS

En el presente capítulo se desarrollan y explican los detalles algorítmicos y de implementación de los distintos bloques que conforman el codificador implementado. Gran parte del codificador fue implementado de acuerdo al estándar FS1016<sup>1</sup>.

Hay que resaltar que el codificador implementado no requiere ceñirse ciento por ciento al estándar, se ha tratado de tomar ventaja de ello para implementar el codificador de la forma más óptima posible, tal como será descrito más adelante.

Conjuntamente con cada bloque se presenta la teoría correspondiente asociada a cada bloque así como también se hace referencia a la teoría y ecuaciones expuestas en capítulos anteriores y/o en la literatura correspondiente.

También, con cada bloque que constituye el codificador implementado, se presentan resultados obtenidos de la ejecución de dicho bloque algorítmico en el DSP TMS320C6711 de Texas Instruments que es donde se implementó el codificador. Las gráficas y segmentos de memoria (variables) que se presentan como resultados son tomadas directamente de las pantallas del Code Composer Studio (con una captura de la pantalla).

En cada bloque se resaltan los algoritmos implementados de acuerdo al estándar FS1016. Las mejoras implementadas en cada uno de los bloques/algoritmos (donde corresponda) se detallan al final del capítulo. Dichas mejoras tienen que ver mayormente con:

- Optimización o cambios de segmentos de código de bloques en el codificador.
- Optimización de bloques para ahorro de memoria.
- Optimización de los bloques para mejora de la velocidad de ejecución.

---

<sup>1</sup> Joseph P. Campbell, Jr., Thomas E. Tremain and Vanoy C. Welch, "The Federal Standard 1016 4800 bps CELP Voice Coder", "Digital Signal Processing, A Review Journal," Volume 1, Number 3, Academic Press.



Dichas mejoras se presentan a lo largo del presente capítulo, y están además resumidas en la sección 6.13.

Las mejoras para la optimización y acondicionamiento óptimo en el DSP se dejan para el capítulo 7.

La Fig. 6.1 muestra el esquema general del codificador implementado. Se pueden apreciar los diversos bloques constituyentes y la interacción entre ellos. A la entrada se tiene la voz digitalizada en formato de 16 bits a 8 KHz. La trama de voz ingresa a dos ramas del algoritmo, una empieza con la segmentación en 4 subsegmentos y la otra es la que empieza con el filtro de entrada. Los bloques que siguen al filtro de entrada calculan los coeficientes LPC del segmento de entrada de la voz, realiza la conversión de dichos coeficientes a otro conjunto de coeficientes en formato LSP, luego se procede a cuantizar y luego a interpolar dichos coeficientes LSP en cuatro subconjuntos de 10 coeficientes LSP cada uno. Cada subconjunto de coeficientes LSP es luego transformado a los correspondientes coeficientes LPC, los cuales se emplean para los filtro de síntesis de formantes, aplicables a cada subsegmento (producto del bloque de segmentación en 4 subsegmentos). Para cada subsegmento de análisis se ejecutan los bloques correspondientes a las búsquedas adaptivas y estocásticas, los cuales están conformados por los bloques de “cálculo de respuesta impulsional”, “calculo del error a la salida del filtro H producto de una entrada igual a cero”, “ajustes de ganancias”, “búsqueda del *codebook* estocástico”. Cada uno de estos bloques se explicará con detenimiento en las siguientes secciones. Ahora sólo se pretende mostrar de una forma general y de manera gráfica los bloques que comprenden el algoritmo de la parte del codificador.

Se observa además que el segmento de voz de entrada es de duración de 30 ms y que cada subsegmento donde se realizan las búsquedas adaptivas y estocásticas, tienen una duración de 7.5 ms. El esquema mostrado nos da además una idea de la complejidad del algoritmo del codificador.

**Esquema General del Codificador Implementado**

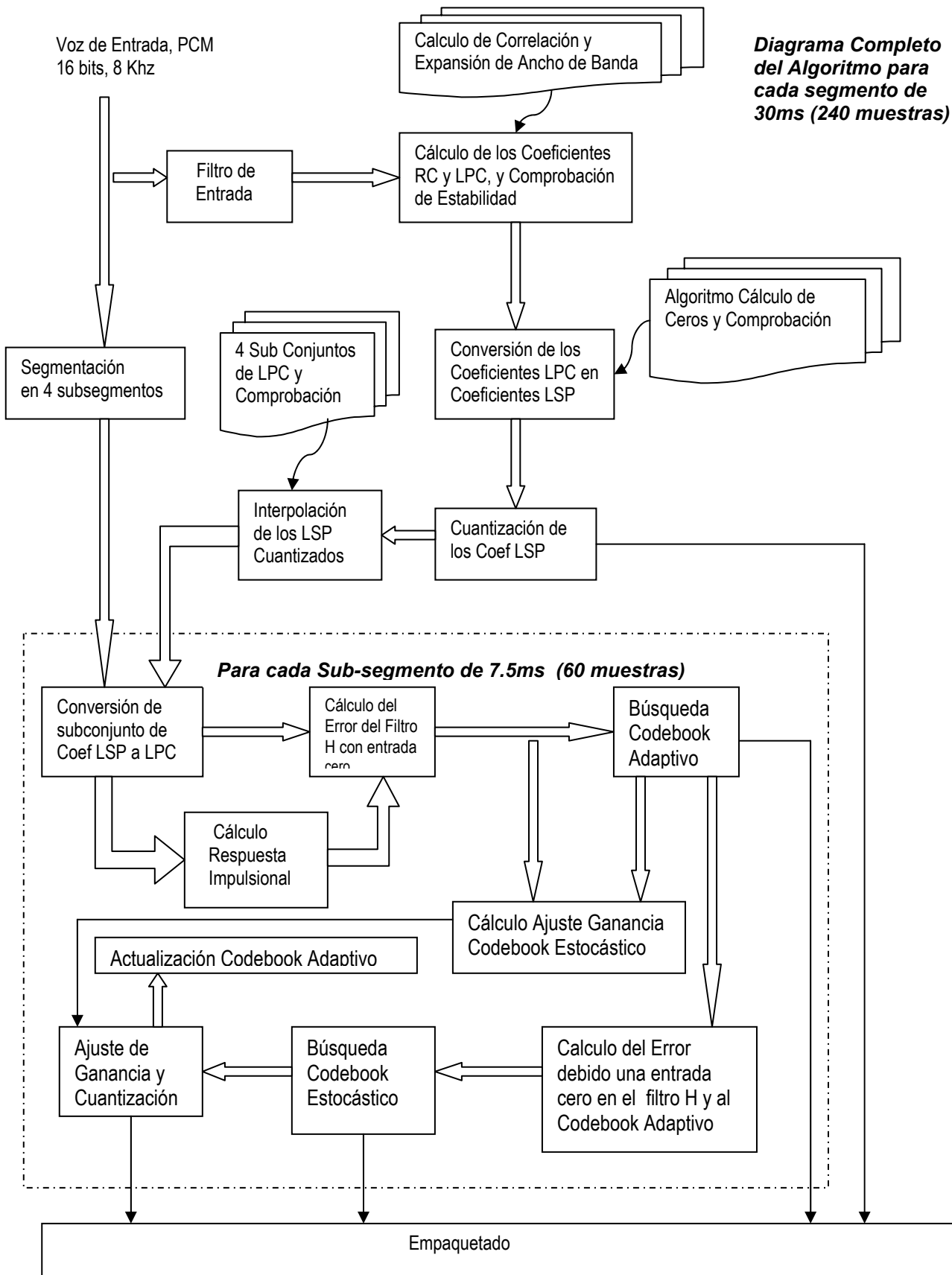


Fig. 6.1 Esquema General del Codificador Implementado, basado en (26).

### Esquema General del Decodificador Implementado

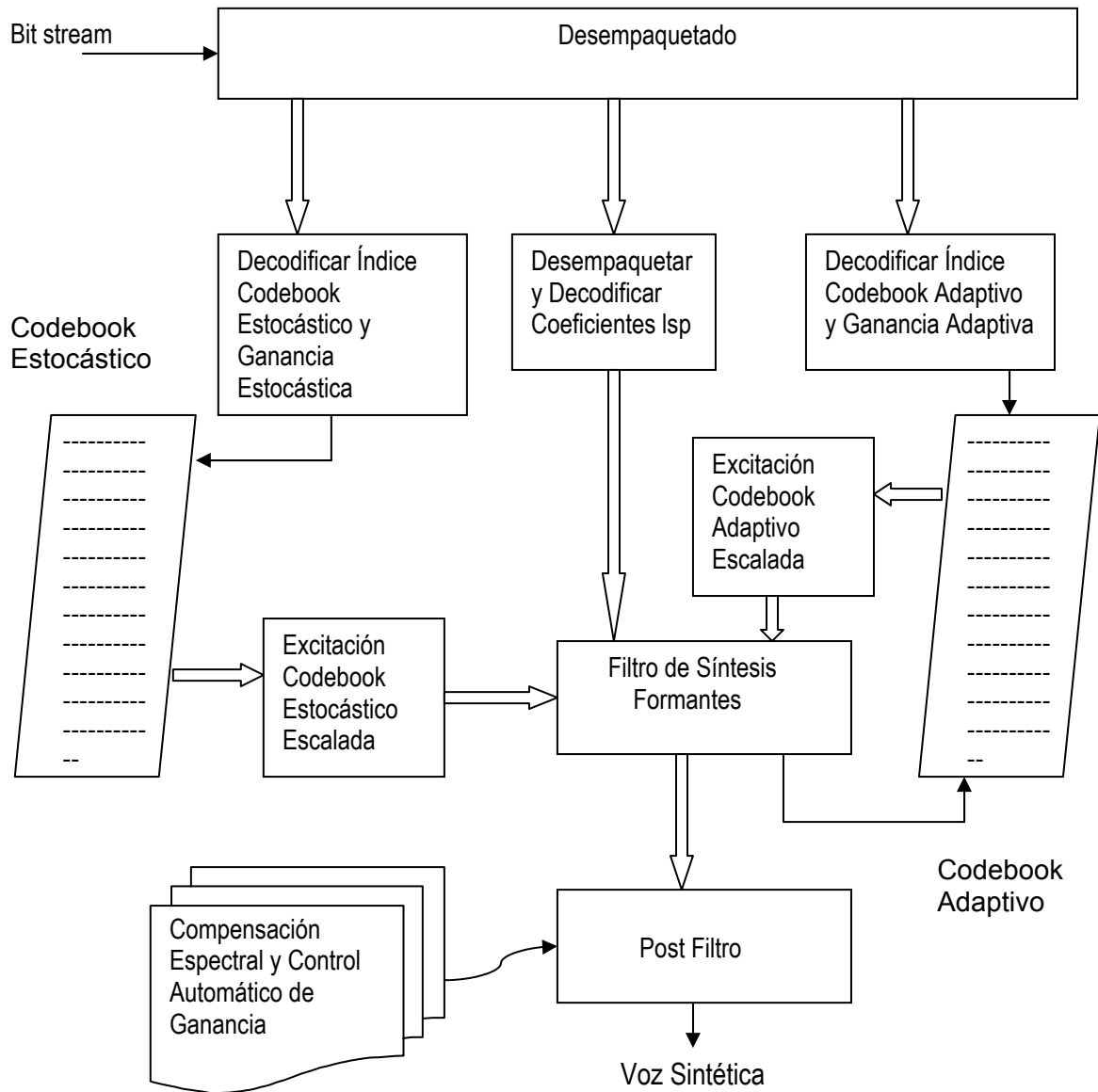


Fig. 6.2 Esquema General del Decodificador Implementado, basado en (26)

Como se puede apreciar en la Fig. 6.2, el decodificador realiza la función inversa al codificador mostrado en la Fig. 6.1. Se puede observar además que la complejidad del decodificador es menor que la del codificador, esto significa que la mayor carga computacional se ubica en la parte del codificador. El codificador primero desempaqueta cada uno de los parámetros recibidos en la corriente de bits (*bit stream*) y los recupera. De acuerdo a los índices recibidos (se explicará más adelante), se seleccionan las secuencias en cada uno de los *codebook* que entrarán a excitar de manera conjunta el filtro de síntesis construido en base a los coeficientes LSP recibidos. A la salida del filtro de síntesis se ubica un bloque denominado Post-Filtro, el cual ayuda a mejorar la calidad de la voz decodificada.

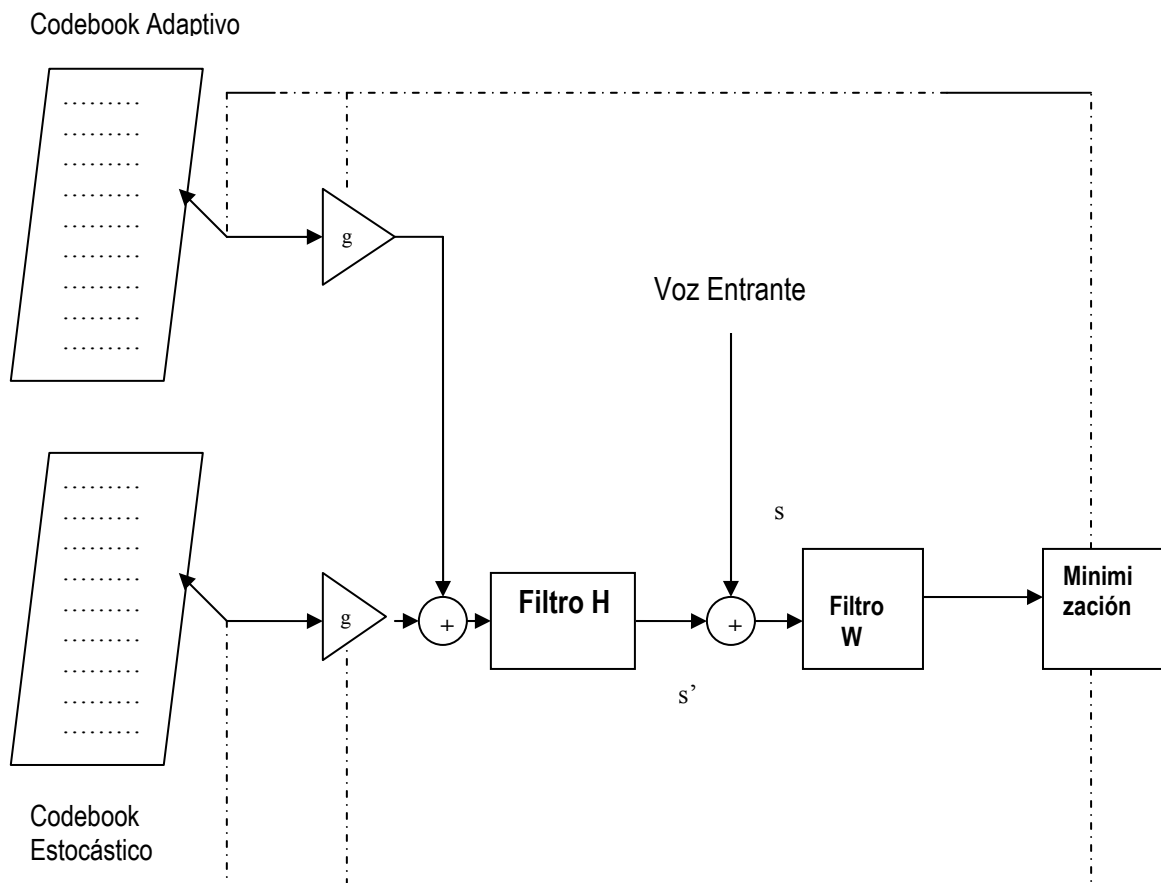


Fig. 6.3 Ubicación de los principales Filtros Involucrados en el Codificador (26)

La Fig. 6.3 muestra de manera general los principales filtros involucrados en el codificador, el filtro H es el filtro que representa la generación de la voz sintética  $s'$ , la cual se compara con la voz entrante  $s$ , el error se pasa a través del filtro W, el cual es el filtro de ponderación perceptual, cuya salida se minimiza y luego se retroalimenta para las siguientes búsquedas de excitaciones en los *codebook* adaptivo y estocástico

## 6.1 Señal de Entrada al Codificador

La señal de voz de entrada al codificador es una señal estándar PCM muestreada a 8 KHz de 16 bits, la cual es la resolución entregada por el conversor A/D del DSP TMS320C6711 (el algoritmo también puede trabajar con una señal de 8 bits). La referencia para la tasa de compresión total del codificador se calcula teniendo como base la tasa entrante de la señal PCM la cual es:

$$16 \text{ bits/muestra} * 8000 \text{ muestras /segundo} = 128\,000 \text{ bits / segundo}$$

Por consiguiente, la tasa de bits de la señal entrante es 128 Kbps.

A continuación se muestra la señal entrante que ha sido cargada a la memoria del DSP, su configuración es:

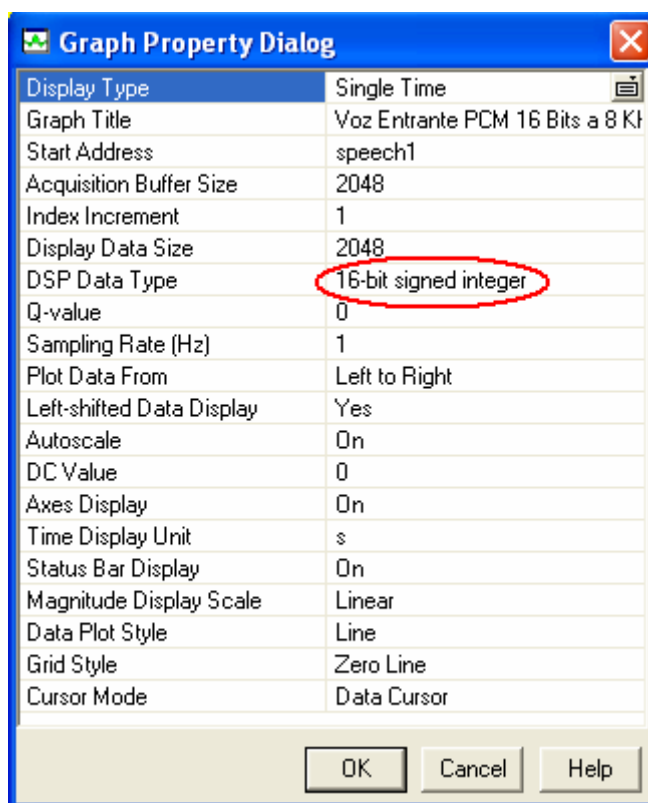


Fig. 6.4 Configuración para visualización de la señal entrante

Notar en la Fig. 6.4 que el tipo de datos de entrada es de 16 bits, el cual corresponde a la señal de la voz entrante en formato PCM de 16 bits.

La Fig. 6.5 nos muestra un segmento de aproximadamente un segundo de voz entrante (8160 muestras), como el límite de visualización de gráficos del Code Composer es de 2048, se ha tenido que partir en cuatro ventanas,

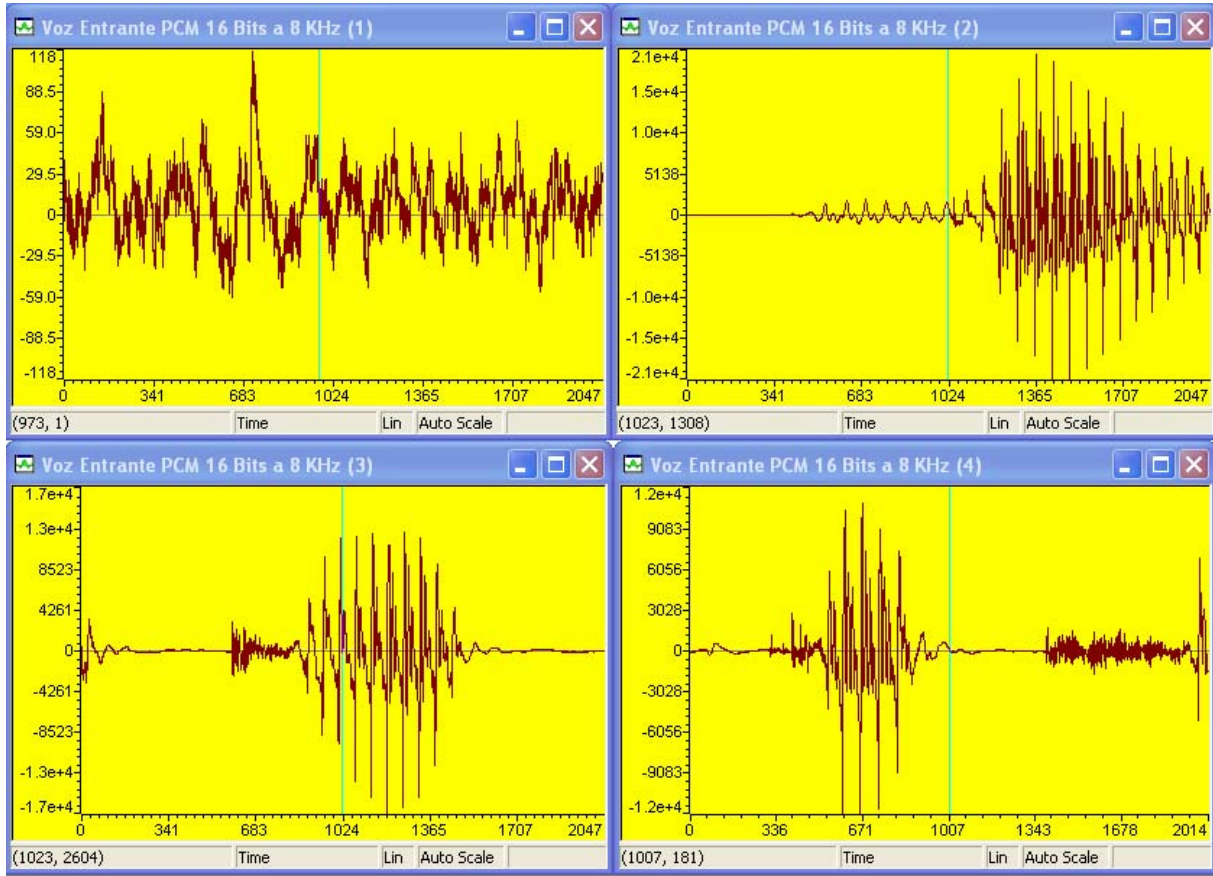


Fig. 6.5 Señal de voz entrante a 8 KHz con resolución de 16 bits (aprox. 1 seg de voz)

## 6.2 Ventana de Hamming

La Fig. 6.6 muestra una ventana de Hamming de 240 puntos (30 ms de voz).

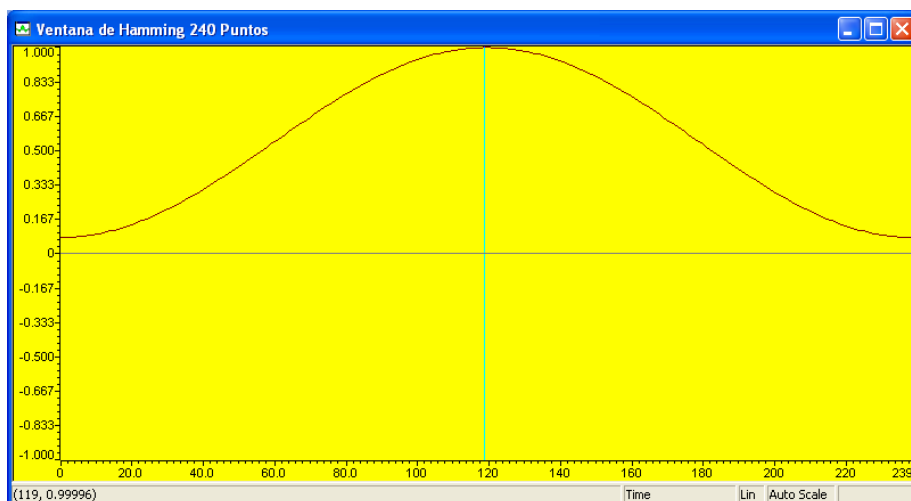


Fig. 6.6 Ventana de Hamming de 240 puntos.

La ventana de Hamming es un componente principal que se usa en varios bloques del codificador implementado, en la fase inicial del algoritmo se usa principalmente para la estimación de la autocorrelación (Ver capítulo 3).

### 6.3 Filtro de entrada

Su ubicación se muestra en la Fig. 6.1. Es un filtro digital pasa-altos que se usa para filtrar la voz entrante, eliminando frecuencias menores a 100 Hz, las cuales pueden contener componentes no deseadas como la frecuencia de 60 Hz (debido a la red eléctrica), su función de transferencia es la siguiente (26):

$$f(z) = \frac{0.946 - 1.892z^{-1} + 0.946z^{-2}}{1 - 1.889033z^{-1} + 0.8948743z^{-2}} \quad (6.1)$$

La respuesta en frecuencia y fase del filtro se muestra en la Fig. 6.7.

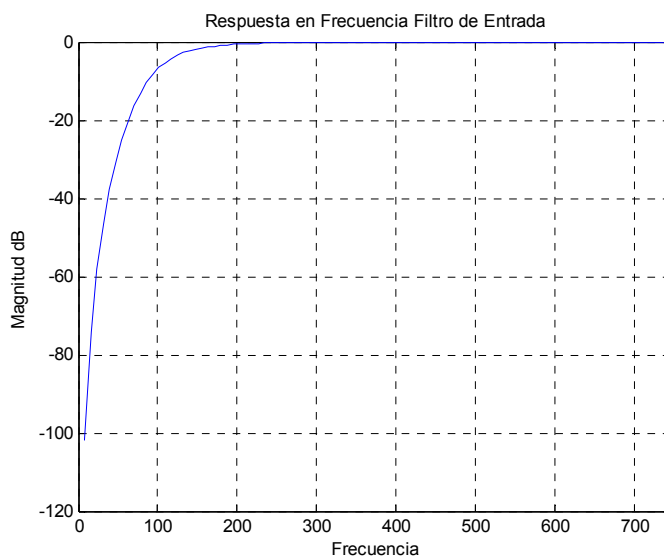


Fig. 6.7 Resposta en Frecuencia Filtro de Entrada

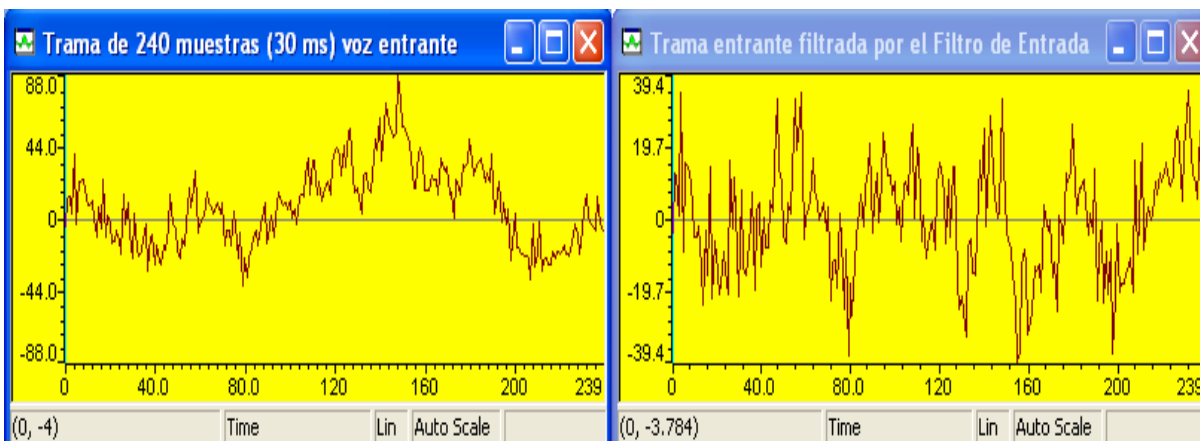


Fig. 6.8 Voz entrante (izquierda) y voz filtrada por el filtro de entrada (derecha)

#### 6.4 Posicionamiento de los 2 segmentos principales de análisis

El algoritmo segmenta las tramas en dos segmentos muy bien diferenciados, cada segmento se usa para un análisis en particular. La segmentación se ilustra en la Fig. 6.9.

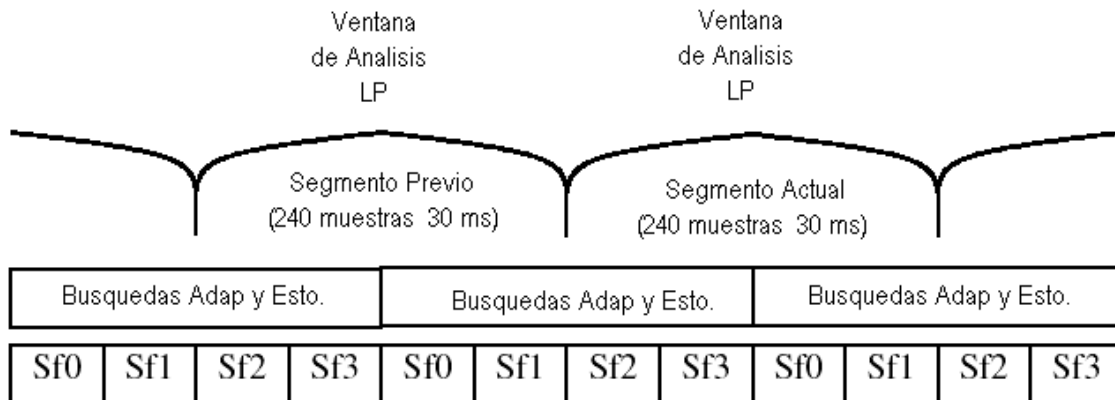


Fig. 6.9 Posiciones de los segmentos de análisis (4, 26)

Los dos segmentos principales son:

- **Segmento para análisis LP**, el cual es el segmento actual filtrado a la entrada.
- **Segmento para realizar las búsquedas adaptivas y estocásticas.** Este segmento está formado por la segunda mitad del segmento previo y por la primera mitad del segmento actual tal como se muestra en la Fig. 6.9.

Cuando se inicia el algoritmo obviamente el “segmento previo” es un segmento vacío o con todos sus elementos de valor cero, por lo tanto el primer segmento para realizar las búsquedas estocásticas y adaptivas tiene la forma mostrada en la Fig. 6.10.

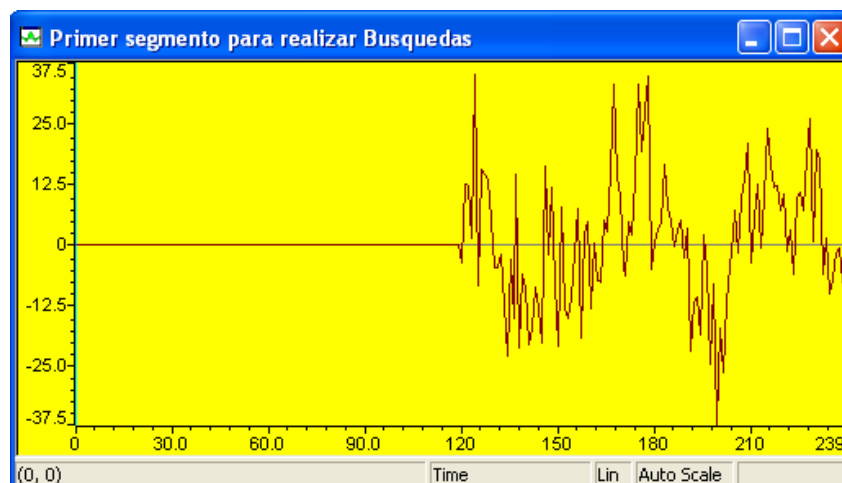


Fig. 6.10 Primer segmento para búsquedas adaptiva y estocástica.



## 6.5 Cálculo de los Coeficientes RC, LPC, y Comprobación de Estabilidad

Se puede ver la ubicación de esta parte del algoritmo en la Fig. 6.1. Recordar que esta parte del código trabaja en los segmentos o ventanas para análisis LP de la Fig. 6.9.

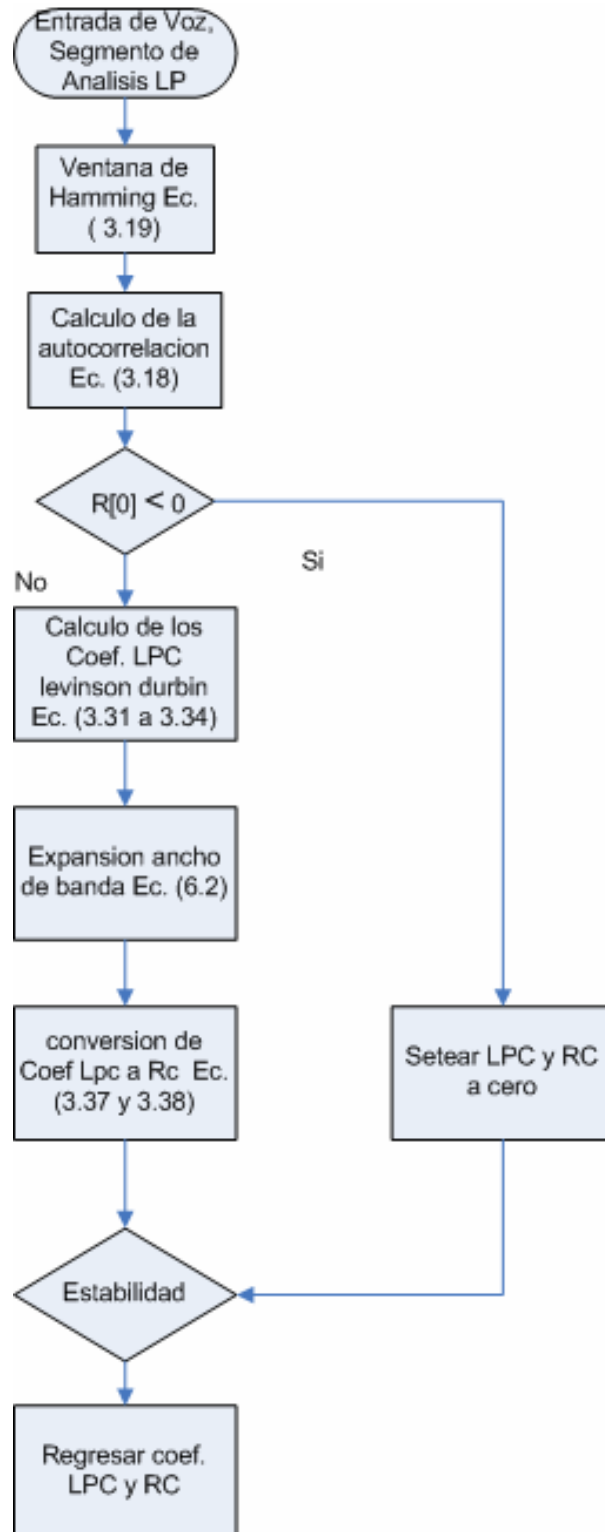


Fig. 6.11 Diagrama de flujo del algoritmo para el cálculo de los coeficientes LPC y RC

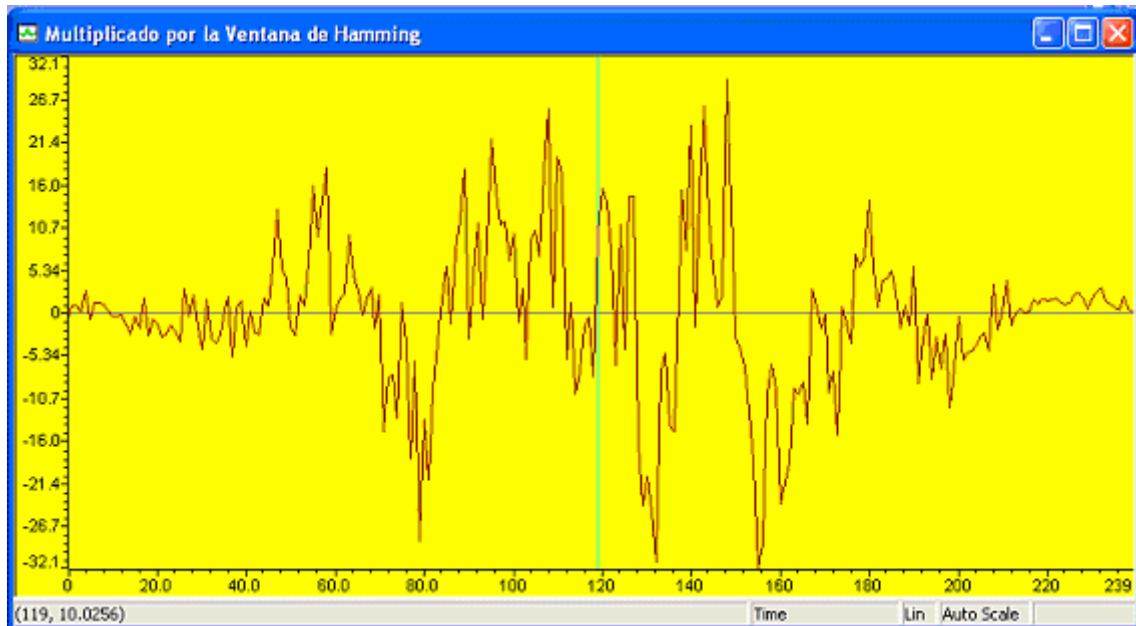


Fig. 6.12 Multiplicación de la secuencia de entrada por la Ventana de Hamming

| Name | Value      | Type      | Radix |
|------|------------|-----------|-------|
| nseg | 0          | int       | dec   |
| c0   | 21731.36   | float     | float |
| c    | 0x8001D06C | float[11] | hex   |
| [0]  | 13483.73   | float     | float |
| [1]  | 10551.2    | float     | float |
| [2]  | 8972.016   | float     | float |
| [3]  | 5985.273   | float     | float |
| [4]  | 5581.788   | float     | float |
| [5]  | 3474.93    | float     | float |
| [6]  | 909.1135   | float     | float |
| [7]  | -378.355   | float     | float |
| [8]  | 59.1725    | float     | float |
| [9]  | -530.8433  | float     | float |

Watch Locals Watch 1

Fig. 6.13 Los 11 valores de autocorrelación para el cálculo de los coeficientes LPC.

Los coeficientes LPC y RC se obtienen por medio del algoritmo de Levinson-Durbin el cual fue descrito en el capítulo 3. Las ecuaciones (3.31) a (3.34) implementan el algoritmo de Levinson mientras que las ecuaciones (3.37) y (3.38) implementan la conversión de los coeficientes LPC a RC. La autocorrelación se calcula de acuerdo a (3.18).

| Name  | Value      | Type     | Radix |
|-------|------------|----------|-------|
| nseg  | 0          | int      | dec   |
| atemp | 0x8001D094 | float[1] | hex   |
| [0]   | 1.0        | float    | float |
| [1]   | -0.5154237 | float    | float |
| [2]   | -0.1195119 | float    | float |
| [3]   | -0.1355224 | float    | float |
| [4]   | 0.1190418  | float    | float |
| [5]   | -0.1582956 | float    | float |
| [6]   | 0.02915981 | float    | float |
| [7]   | 0.09482969 | float    | float |
| [8]   | 0.1126803  | float    | float |
| [9]   | -0.1113003 | float    | float |
| [10]  | 0.01820998 | float    | float |

Fig. 6.14 Los coeficientes LPC sin expansión de ancho de banda

## 6.6 Expansión de ancho de Banda

Cuando se calculan los coeficientes LPC, el filtro de síntesis resultante puede llegar a ser marginalmente estable debido a que los polos pueden quedar ubicados demasiado cerca al círculo unidad. El problema se agrava cuando se trata de una implementación en punto fijo. Este tipo de problema crea ocasionales “chirps” u oscilaciones en la voz sintetizada. Se denomina expansión de ancho de banda al proceso mejora de la estabilidad mediante la modificación de los coeficientes LPC de acuerdo a:

$$a_{new} = \gamma^i a_i \quad , i=1,2,\dots,M \quad (6.2)$$

El factor  $\gamma$  es siempre menor que 1. En la presente implementación se trabajó con un valor de  $\gamma = 0.994127$ .

El resultado de la expansión de ancho de banda se puede apreciar en la Fig. 6.15. Se tiene que antes de la expansión el espectro del filtro de síntesis tiene picos más pronunciados lo que se traduce en una respuesta impulsional más oscilante. Por otro lado, luego de la expansión, el espectro del filtro de síntesis presenta picos menos pronunciados y la respuesta impulsional presenta muy poca oscilación.

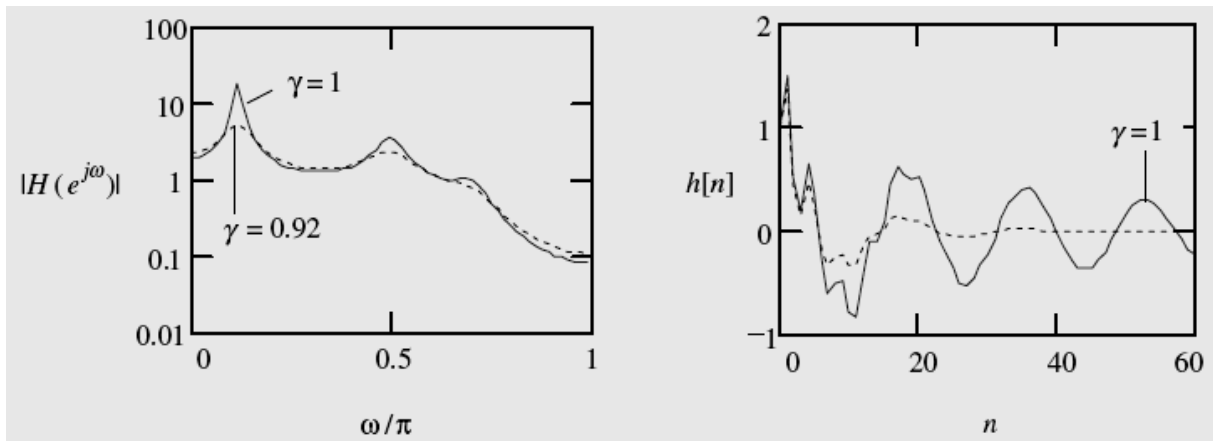


Fig. 6.15 Espectro (izquierda) y respuesta impulsional (derecha) para la expansión de ancho de banda. Sin expandir (línea sólida) y expandido (línea punteada) (4)

Los correspondientes coeficientes LPC de la Fig. 6.14 modificados por la expansión de ancho de banda con el factor  $\gamma$  de 0.994127 se muestran en la Fig. 6.16.

| Name | Value      | Type      | Radix |
|------|------------|-----------|-------|
| fcf  | 0x80014D54 | float[11] | hex   |
| [0]  | 1.0        | float     | float |
| [1]  | -0.5123966 | float     | float |
| [2]  | -0.1181122 | float     | float |
| [3]  | -0.1331486 | float     | float |
| [4]  | 0.1162698  | float     | float |
| [5]  | -0.1537015 | float     | float |
| [6]  | 0.02814724 | float     | float |
| [7]  | 0.09099916 | float     | float |
| [8]  | 0.1074937  | float     | float |
| [9]  | -0.1055536 | float     | float |
| [10] | 0.01716833 | float     | float |

Fig. 6.16 Coeficientes LPC modificados por la expansión de ancho de banda.

La Fig. 6.17 muestra los coeficientes LPC modificados por la expansión de ancho de banda, convertidos a coeficientes de reflexión RC.

| Name | Value       | Type      | Radix |
|------|-------------|-----------|-------|
| nseg | 0           | int       | dec   |
| rcn  | 0x8001E02C  | float[10] | hex   |
| [0]  | 0.6156581   | float     | float |
| [1]  | 0.1619593   | float     | float |
| [2]  | 0.09814361  | float     | float |
| [3]  | -0.07918084 | float     | float |
| [4]  | 0.08158539  | float     | float |
| [5]  | -0.07923353 | float     | float |
| [6]  | -0.1135302  | float     | float |
| [7]  | -0.0606907  | float     | float |
| [8]  | 0.09678513  | float     | float |
| [9]  | -0.01716833 | float     | float |

Fig. 6.17 Coeficientes RC derivados de los coeficientes LPC modificados

Una vez que se calculan los coeficientes RC, la comprobación de la estabilidad del filtro de síntesis es directa, en la Fig. 6.17 podemos observar que todos los coeficientes son menores que 1 lo que asegura la estabilidad.

## 6.7 Conversión de los Coeficientes LPC a Coeficientes LSP

La representación de los coeficientes LPC en coeficientes LSP (Linear Spectral Pair) presenta muchas propiedades deseables las cuales lo hacen muy ventajoso en comparación a la representación LPC y RC, es por eso que ha tenido amplia aceptación en aplicaciones de codificación de voz. (4, 13, 26)

Sea el filtro predictivo con función de transferencia:

$$A(z) = 1 + a_1 z^{-1} + \dots + a_M z^{-M} = \prod_{i=1}^M (1 - z_i z^{-1}) \quad (6.3)$$

Ya que los coeficientes LPC son reales, los  $z_i$  son ya sea reales o pares de complejos conjugados. Como (6.3) es un polinomio en  $z^{-1}$  de orden M, se pueden formar dos polinomios:

$$P(z) = A(z) \left( 1 + z^{-(M+1)} \frac{A(z^{-1})}{A(z)} \right) = A(z)(1 + G(z)) \quad (6.4)$$

$$Q(z) = A(z) \left( 1 - z^{-(M+1)} \frac{A(z^{-1})}{A(z)} \right) = A(z)(1 - G(z)) \quad (6.5)$$

Con 
$$G(z) = z^{-(M+1)} \frac{A(z^{-1})}{A(z)} \quad (6.6)$$

Entonces 
$$A(z) = (P(z) + Q(z)) / 2 \quad (6.7)$$

Los coeficientes LSP se definen como aquellos valores de frecuencia  $\omega$  que satisfacen la siguiente propiedad:

$$\{\omega \mid P(e^{j\omega}) = 0\} \text{ ó } \{\omega \mid Q(e^{j\omega}) = 0\} \text{ para } 0 < \omega < \pi \quad (6.8)$$

Desarrollando  $P(z)$  y  $Q(z)$  en función de los coeficientes LPC:

$$P(z) = 1 + (a_1 + a_M)z^{-1} + (a_2 + a_{M-1})z^{-2} + \dots + (a_1 + a_M)z^{-M} + z^{-(M+1)} \quad (6.9)$$

$$Q(z) = 1 + (a_1 - a_M)z^{-1} + (a_2 - a_{M-1})z^{-2} + \dots - (a_1 - a_M)z^{-M} - z^{-(M+1)} \quad (6.10)$$

Para  $M$  par (que es nuestro caso ya que  $M = 10$ ),  $P(z)$  y  $Q(z)$  tienen ceros en  $-1$  y  $1$  respectivamente por lo que se pueden factorizar para obtener:

$$P'(z) = \frac{P(z)}{1 + z^{-1}} \quad \text{y} \quad Q'(z) = \frac{Q(z)}{1 - z^{-1}} \quad (6.11)$$

Para el caso de  $P(z)$  y  $Q(z)$ ,

$$\begin{aligned} p_0 &= p_{M+1} = 1 \\ p_i &= p_{M-i+1} = a_i + a_{M-i+1} \\ q_0 &= -q_{M+1} = 1 \\ q_i &= -q_{M-i+1} = a_i - a_{M-i+1} \end{aligned} \quad (6.12)$$

Para el caso de  $P'(z)$  y  $Q'(z)$  y  $M$  par (nuestro caso):

$$\begin{aligned}
 p'_0 &= 1 \\
 p'_i &= p_i - p'_{i-1} \\
 q'_0 &= 1 \\
 q'_i &= q_i + q'_{i-1}
 \end{aligned} \tag{6.13}$$

Evaluando los polinomios en  $z = e^{j\omega}$  tenemos:

$$\begin{aligned}
 P'(e^{j\omega}) &= e^{-j\omega M_1} \left[ p'_0 e^{j\omega M_1} + p'_1 e^{j\omega(M_1-1)} + p'_2 e^{j\omega(M_1-2)} + \dots + p'_{2M_1} e^{-j\omega M_1} \right] \\
 Q'(e^{j\omega}) &= e^{-j\omega M_2} \left[ q'_0 e^{j\omega M_2} + q'_1 e^{j\omega(M_2-1)} + q'_2 e^{j\omega(M_2-2)} + \dots + q'_{2M_2} e^{-j\omega M_2} \right]
 \end{aligned} \tag{6.14}$$

Simplificando las expresiones dentro de los corchetes, nos conduce al siguiente par de expresiones:

$$\begin{aligned}
 P_0(\omega) &= 2 \cos(M_1\omega) + 2p'_1 \cos((M_1 - 1)\omega) + \dots + 2p'_{M_1-1} \cos(\omega) + p'_{M_1} \\
 Q_0(\omega) &= 2 \cos(M_2\omega) + 2q'_1 \cos((M_2 - 1)\omega) + \dots + 2q'_{M_2-1} \cos(\omega) + q'_{M_2}
 \end{aligned} \tag{6.15}$$

Para nuestro caso  $M_1=M_2=M/2 = 5$

Los 10 coeficientes LSP se obtienen calculando las raíces de las expresiones en (6.15).  $P_0(\omega)$  contribuye con 5 y  $Q_0(\omega)$  contribuye con 5, resultando en un total de 10 coeficientes LSP.

La Fig. 6.18 muestra el diagrama de flujo para el cálculo de los coeficientes LSP.

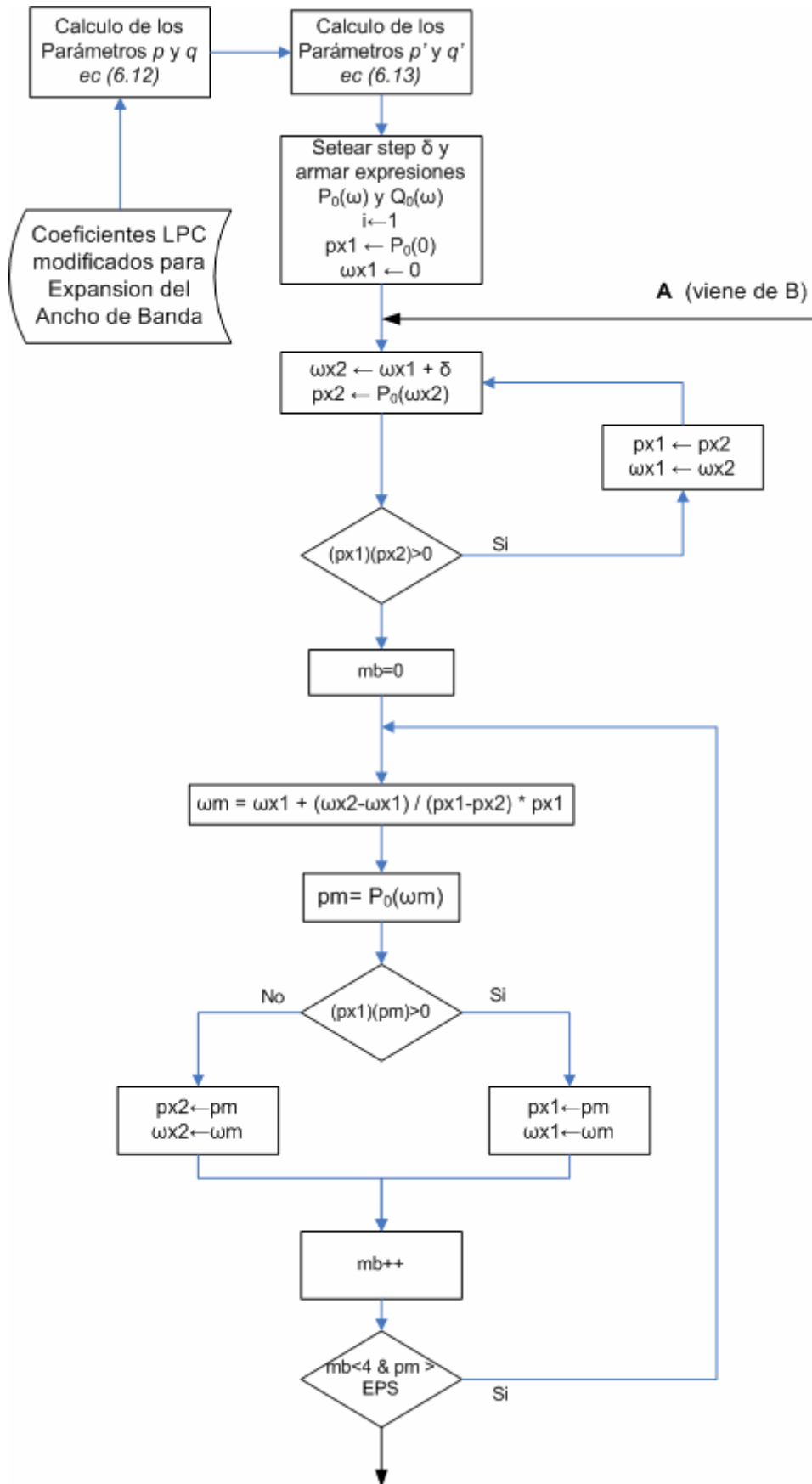


Fig. 6.18 Algoritmo para el Calculo de los Coeficientes LSP (1era parte)



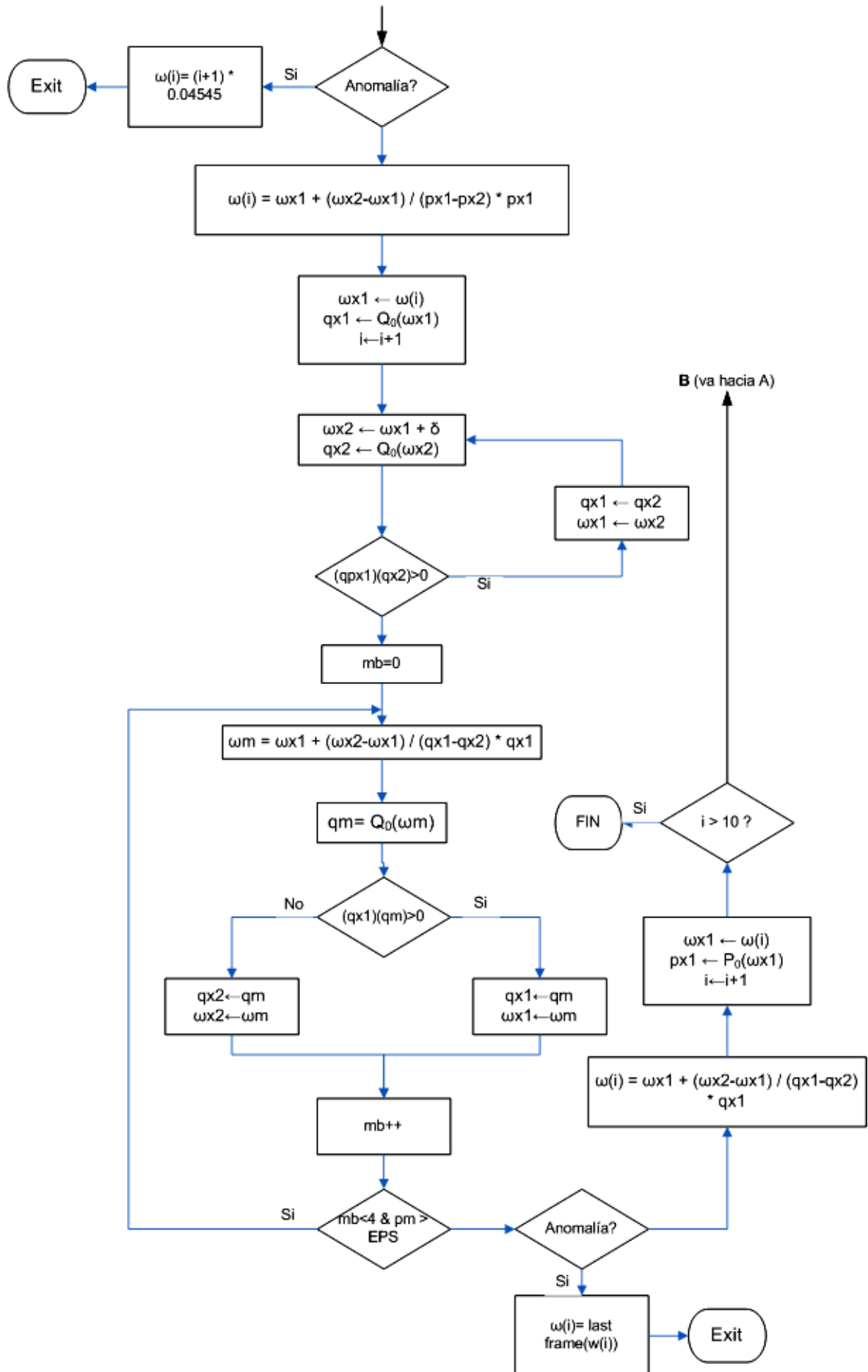


Fig. 6.19 Algoritmo para el Calculo de los Coeficientes LSP (2da parte)

Para el conjunto de coeficientes LPC de la Fig. 6.16 los parámetros  $p$  y  $q$  se muestran en la Fig. 6.20:

| Name | Value       | Type      |
|------|-------------|-----------|
| p1   | 0x8001D36C  | float[24] |
| [0]  | -0.4952283  | float     |
| [1]  | -0.2236658  | float     |
| [2]  | -0.02565493 | float     |
| [3]  | 0.207269    | float     |
| [4]  | -0.1255543  | float     |

| Name | Value       | Type      |
|------|-------------|-----------|
| q1   | 0x8001D3CC  | float[24] |
| [0]  | -0.529565   | float     |
| [1]  | -0.01255859 | float     |
| [2]  | -0.2406422  | float     |
| [3]  | 0.02527068  | float     |
| [4]  | -0.1818488  | float     |

Fig. 6.20 Coeficientes  $p$  y  $q$  de la (ec 6.12).

El conjunto de coeficientes  $p'$  y  $q'$  que conforman las expresiones en (6.15) se muestran en la Fig. 6.21.

| Name | Value      | T..  | F.▲ |
|------|------------|------|-----|
| p    | 0x8001D2AC | fla: | f   |
| [0]  | -1.495228  | fla: | f   |
| [1]  | 1.271563   | fla: | f   |
| [2]  | -1.297217  | fla: | f   |
| [3]  | 1.504486   | fla: | f   |
| [4]  | -0.8150204 | fla: | f   |

| Name | Value      | Type  |
|------|------------|-------|
| q    | 0x8001D30C | float |
| [0]  | 0.470435   | float |
| [1]  | 0.4578764  | float |
| [2]  | 0.2172342  | float |
| [3]  | 0.2425049  | float |
| [4]  | 0.03032804 | float |

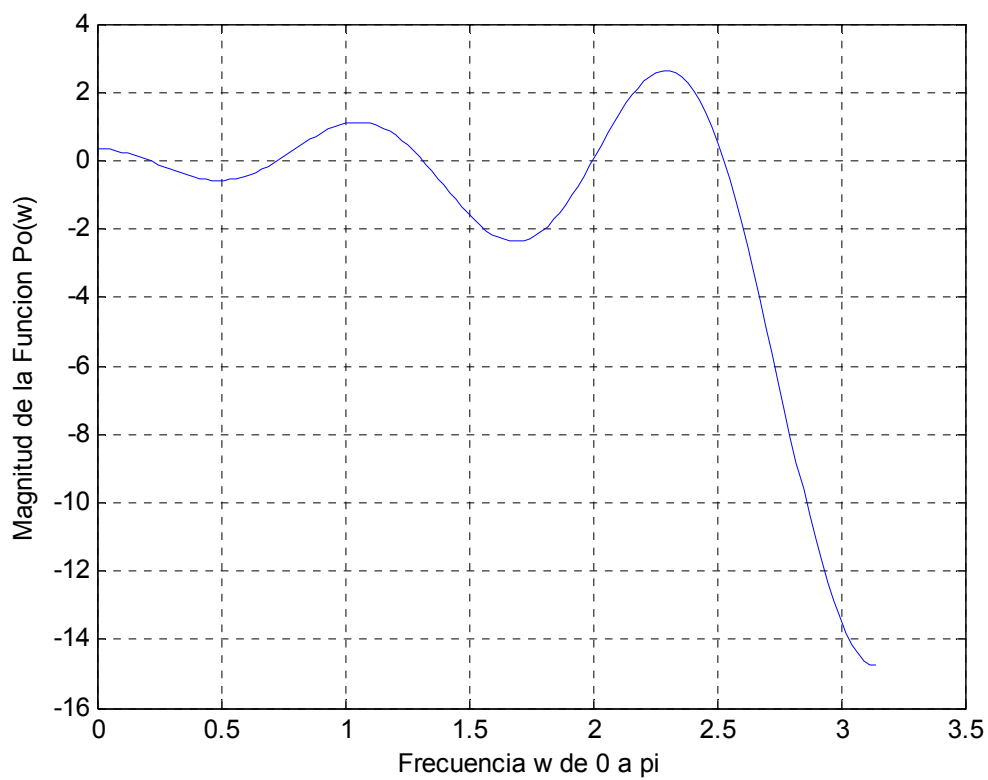
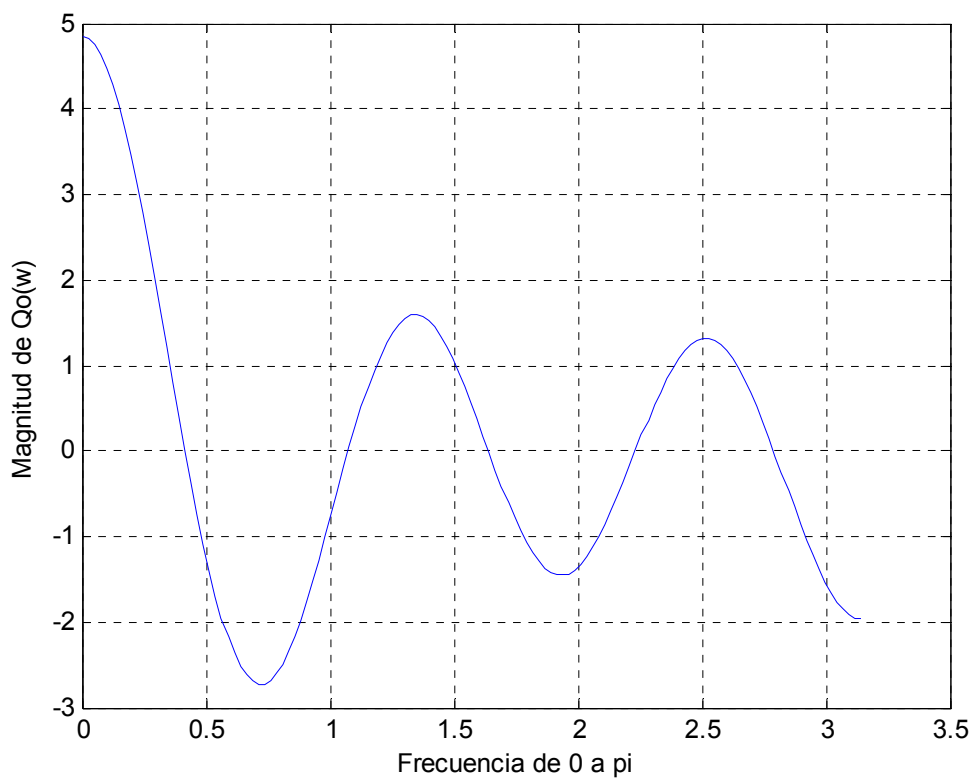
Fig. 6.21 Coeficientes  $p'$  y  $q'$  de la (ec 6.13).

De acuerdo a la Fig. 6.21 y a la ecuación (6.15) las expresiones para  $P_0(\omega)$  y  $Q_0(\omega)$  son las siguientes:

$$P_0(\omega) = 2 \cos(5\omega) - 2.990456 \cos(4\omega) + 2.543126 \cos(3\omega) - 2.594434 \cos(2\omega) + 3.008972 \cos(\omega) - 1.6300408$$

$$Q_0(\omega) = 2 \cos(5\omega) + 0.94087 \cos(4\omega) + 0.9157528 \cos(3\omega) + 0.4344684 \cos(2\omega) + 0.4850098 \cos(\omega) + 0.06065608$$

Las gráficas de  $P_0(\omega)$  y  $Q_0(\omega)$  se muestran en las figuras Fig. 6.22 y Fig. 6.23 respectivamente, la gráfica conjunta de  $P_0(\omega)$  y  $Q_0(\omega)$  se muestra en la Fig. 6.24. Nótese la posición de los ceros de ambas curvas, dicha propiedad de los ceros se llama propiedad de entrelazamiento de los coeficientes LSP.

Fig. 6.22 Gráfica de  $P_0(\omega)$ Fig. 6.23 Gráfica de  $Q_0(\omega)$

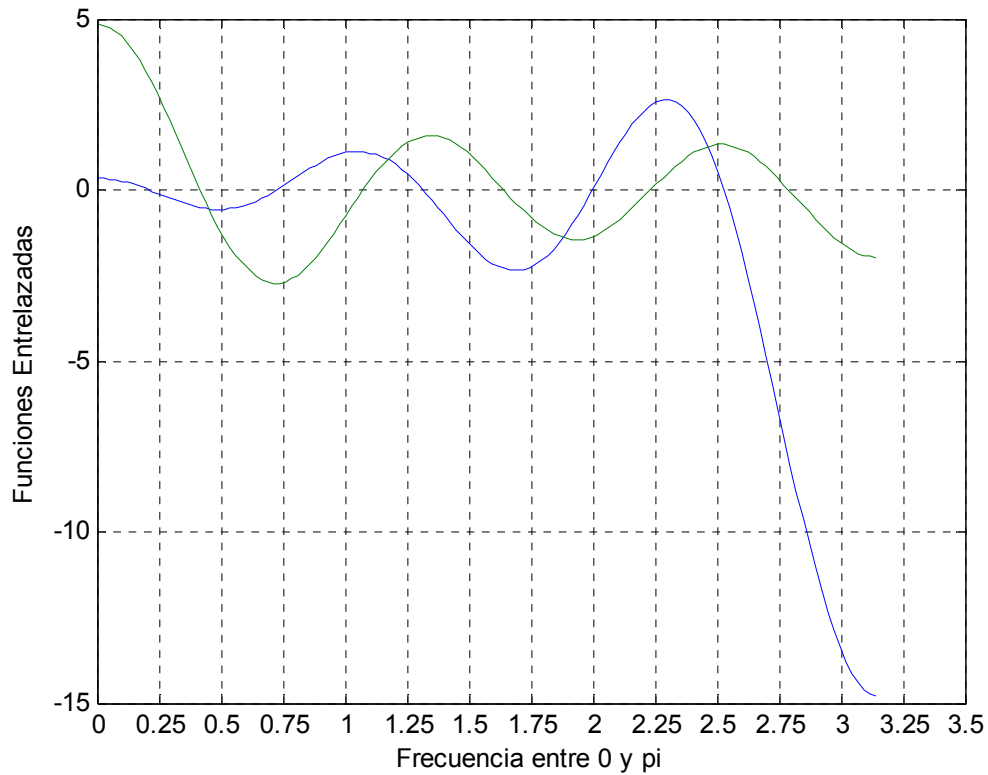


Fig. 6.24 Gráfica de  $Q_0(\omega)$  y  $P_0(\omega)$ .

El algoritmo de las Fig. 6.18 y 6.19, basados en los coeficientes de la Fig. 6.16 arrojan como resultado los 10 coeficientes LSP, los cuales se muestran en la Fig. 6.25.

| Name    | Value      | Type      | Radix |
|---------|------------|-----------|-------|
| newfreq | 0x80011640 | float[10] | hex   |
| [0]     | 0.2092482  | float     | float |
| [1]     | 0.4175615  | float     | float |
| [2]     | 0.7255591  | float     | float |
| [3]     | 1.075551   | float     | float |
| [4]     | 1.313293   | float     | float |
| [5]     | 1.641517   | float     | float |
| [6]     | 1.999025   | float     | float |
| [7]     | 2.232663   | float     | float |
| [8]     | 2.528911   | float     | float |
| [9]     | 2.79136    | float     | float |

Fig. 6.25 Coeficientes LSP calculados por el algoritmo.

Como se puede apreciar de los resultados de las figuras Fig. 6.24 y 6.25, la representación en LSP presenta varias propiedades que la hacen mas atractiva que las representaciones convencionales LPC, RC e incluso LAR (Log Area Ratio) la cual se describe en (4). Dichas propiedades son las siguientes:

- **Propiedad 1.** Si  $A(z)$  es de fase mínima entonces todos los ceros de  $P(z)$  y  $Q(z)$  están dentro del círculo unitario. Esta propiedad garantiza la existencia de los LSP cuando  $A(z)$  es de fase mínima.

Para comprobar que  $A(z)$  es de fase mínima basta con comprobar que todos los coeficientes de reflexión RC se encuentran dentro del círculo unitario. Esto es directo de la Fig. 6.17. Para afianzar aun más lo anterior, se calculan las raíces de  $A(z)$  (los coeficientes se dan en la Fig. 6.16) con el MATLAB:

-0.7031 + 0.3190i, -0.7031 - 0.3190i, -0.4189 + 0.6967i, -0.4189 - 0.6967i,  
0.2699 + 0.7565i, 0.2699 - 0.7565i, 0.7171 + 0.1732i, 0.7171 - 0.1732i,  
0.5612, 0.2212.

Los módulos correspondientes son:

0.7721 0.7721 0.8129 0.8129 0.8032 0.8032 0.7377 0.7377  
0.5612 0.2212

los cuales claramente se encuentran dentro del círculo unitario.

- **Propiedad 2.** Si  $A(z)$  es de fase mínima entonces los ceros de  $P(z)$  y  $Q(z)$  están entrelazados unos con los otros. Esto se observa en la Fig. 6.24. Esta propiedad también funciona de manera inversa: Si los ceros de  $P(z)$  y  $Q(z)$  están entrelazados unos con los otros entonces  $A(z)$  es de fase mínima y entonces la estabilidad del filtro de síntesis está garantizada. Esto es muy útil cuando se van a cuantizar los coeficientes ya que el proceso de cuantización puede mover la posición de los ceros causando inestabilidad en el filtro de síntesis, pero siempre y cuando se comprueben que los parámetros cuantizados respetan la propiedad de entrelazamiento, entonces la estabilidad del filtro de síntesis está garantizada.

## 6.8 Cuantización de los Coeficientes LSP

Se puede observar la ubicación de este bloque en la Fig. 6.1. De entre las ventajas que presenta la representación LSP mencionadas hay dos más que son muy importantes cuando se van a cuantizar dichos coeficientes (4, 25, 26):

- **Propiedad 3.** Los coeficientes controlan directamente el espectro de la señal y los cambios en algunos de los coeficientes LSP se reflejan solo en cambios locales en el espectro.
- **Propiedad 4.** Los coeficientes LSP están delimitados al rango  $[0, \pi]$ , esto es muy ventajoso para el caso de cuantización tal como se verá en el presente capítulo. Caso contrario, los coeficientes LPC no están delimitados.

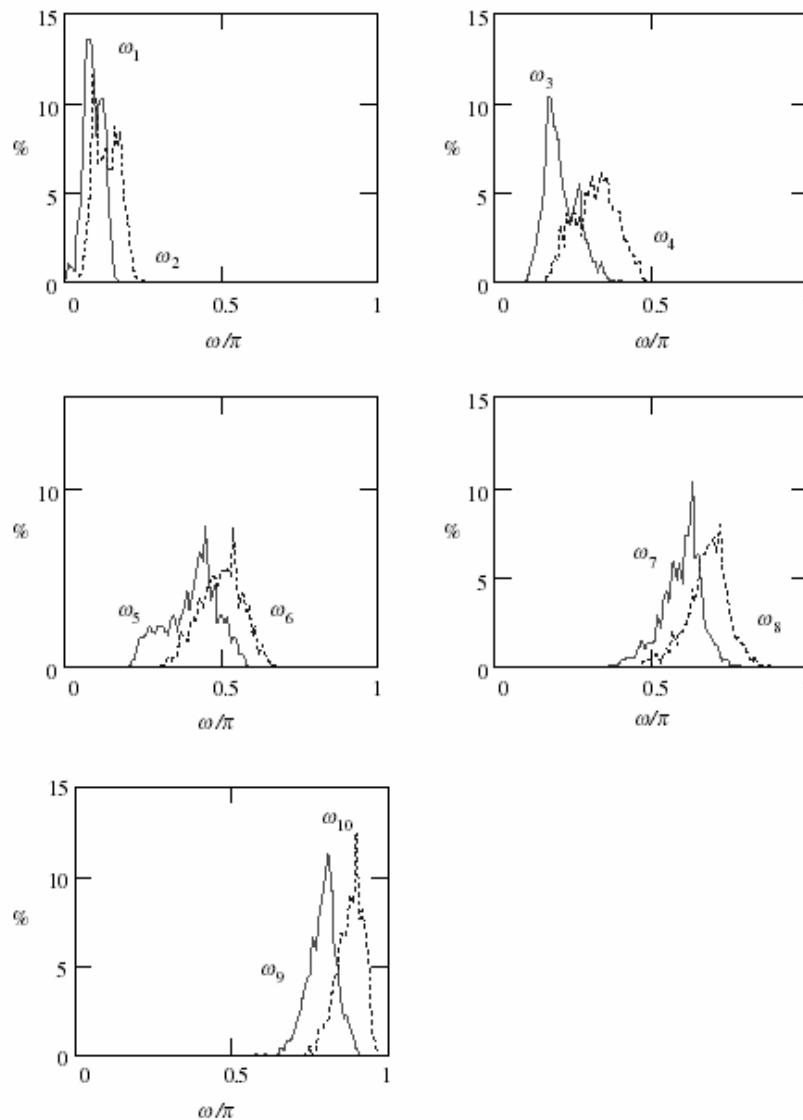


Fig. 6.26 Histogramas de los coeficientes LSP (4, 25)

El diseño del cuantizador para los coeficientes LSP se basa en las propiedades estadísticas y de acotación de los mismos. Los histogramas de algunos coeficientes LSP obtenidos de 1300 segmentos de voz se muestran en la Fig. 6.26.

Como se puede apreciar en la Fig. 6.26, los rangos entre los cuales varían los coeficientes LSP son distintos, además para el diseño del cuantizador, se tiene en cuenta la sensibilidad del oído humano a los rangos de frecuencia encontrándose que el oído humano es más sensible al rango de frecuencias abarcado por los coeficientes  $\omega_2$  a  $\omega_5$  por lo que el cuantizador usa más bits para cuantizar dichos coeficientes. Los niveles de cuantización para cada coeficiente  $\omega_i$  se ilustran en la Fig. 6.27.

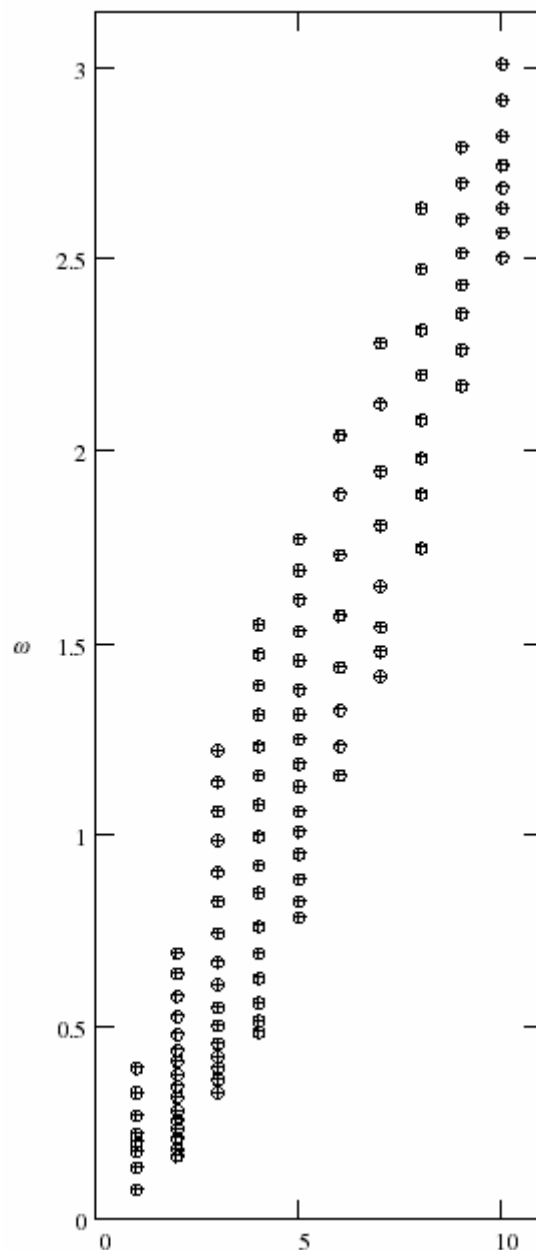


Fig. 6.27 Cuantizador para cada uno de los coeficientes LSP  $\omega_i$ . (25, 26)

En la Fig. 6.27 podemos observar que los bits usados para cuantizar cada uno de los coeficientes LSP, del  $\omega_1$  al  $\omega_{10}$  son: 3, 4, 4, 4, 4, 3, 3, 3, 3, 3 lo que resulta en un total de 34 bits para un segmento de 240 muestras (30ms).

El algoritmo que realiza la cuantización se ilustra en la Fig. 6.28.

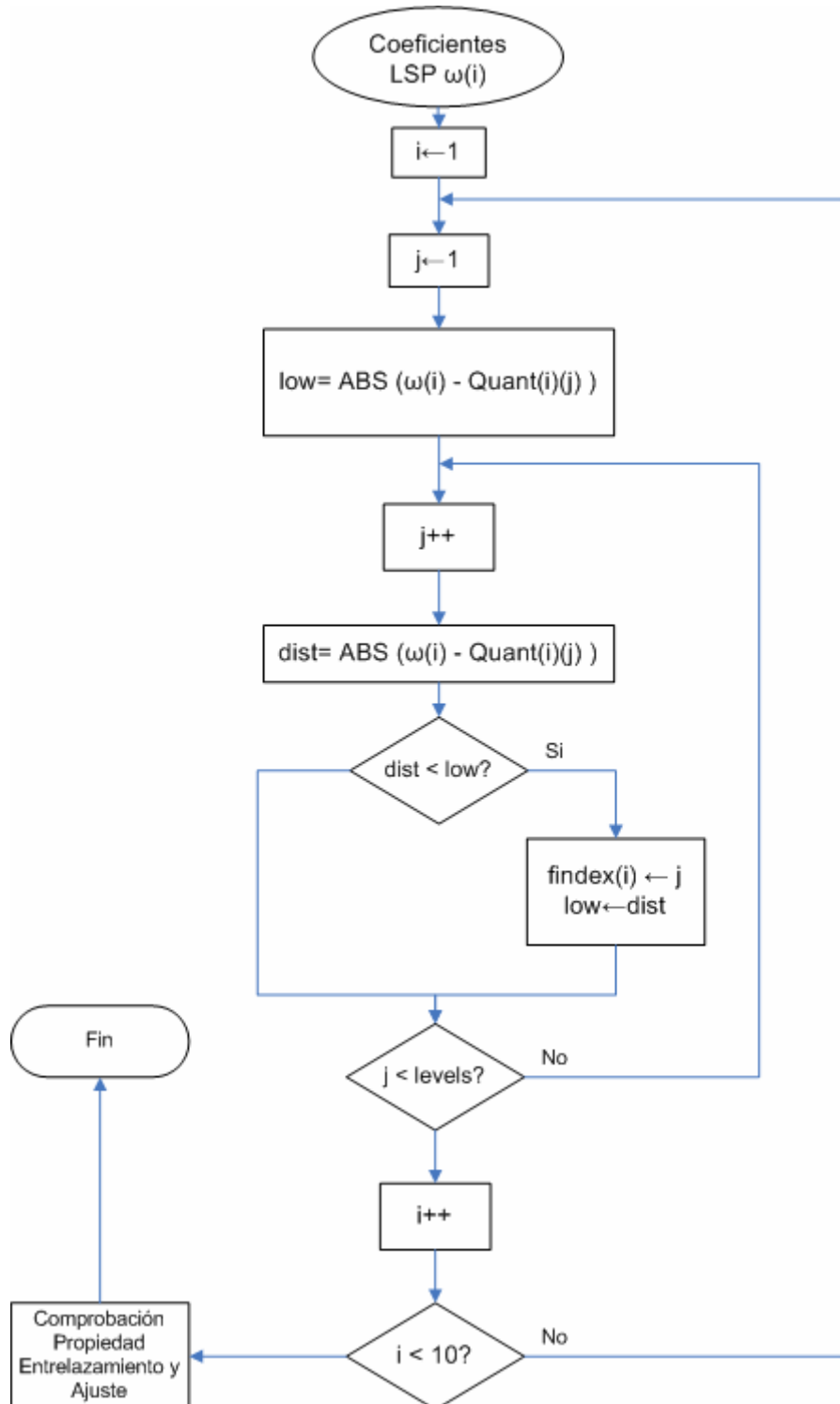


Fig. 6.28 Algoritmo para la cuantización de los LSP.



Los coeficientes LSP de la Fig. 6.25 luego de la cuantización se ven en la Fig. 6.29 así como también los respectivos índices. Observar que respetan la propiedad de entrelazamiento (propiedad 2 en la página 69), garantizando de esta forma la estabilidad del filtro de síntesis. La Fig. 6.29 muestra tanto los índices (de acuerdo al cuantizador de la Fig. 6.27) así como los valores cuantizados de los coeficientes LSP.

| Name   | Value      | Type    | R... |
|--------|------------|---------|------|
| findex | 0x80013D48 | int[10] | hex  |
| [0]    | 4          | int     | dec  |
| [1]    | 9          | int     | dec  |
| [2]    | 9          | int     | dec  |
| [3]    | 9          | int     | dec  |
| [4]    | 9          | int     | dec  |
| [5]    | 4          | int     | dec  |
| [6]    | 5          | int     | dec  |
| [7]    | 4          | int     | dec  |
| [8]    | 4          | int     | dec  |
| [9]    | 5          | int     | dec  |

| Name    | Value      | Type      |
|---------|------------|-----------|
| newfreq | 0x80011640 | float[10] |
| [0]     | 0.2199115  | float     |
| [1]     | 0.408407   | float     |
| [2]     | 0.7461283  | float     |
| [3]     | 1.075995   | float     |
| [4]     | 1.311615   | float     |
| [5]     | 1.570796   | float     |
| [6]     | 1.947787   | float     |
| [7]     | 2.199115   | float     |
| [8]     | 2.513274   | float     |
| [9]     | 2.819579   | float     |

Fig. 6.29 Índices (izquierda) y valores LSP cuantizados (derecha).

## 6.9 Interpolación de los Coeficientes LSP Cuantizados

La ubicación de este bloque se aprecia en la Fig. 6.1. Recordar que de acuerdo al posicionamiento de los segmentos de análisis (ver Fig. 6.30) los cálculos de los coeficientes LSP cuantizados de la Fig. 6.29 corresponden al segmento o ventana de análisis LP (ver Fig. 6.30). Para realizar las búsquedas adaptivas y estocásticas es necesario realizar la interpolación de los coeficientes LSP para obtener 4 subconjuntos de coeficientes LSP que corresponden a cada uno de los 4 sub-segmentos que conforman el “segmento para búsqueda adaptiva y estocástica” (ver Fig. 6.30).

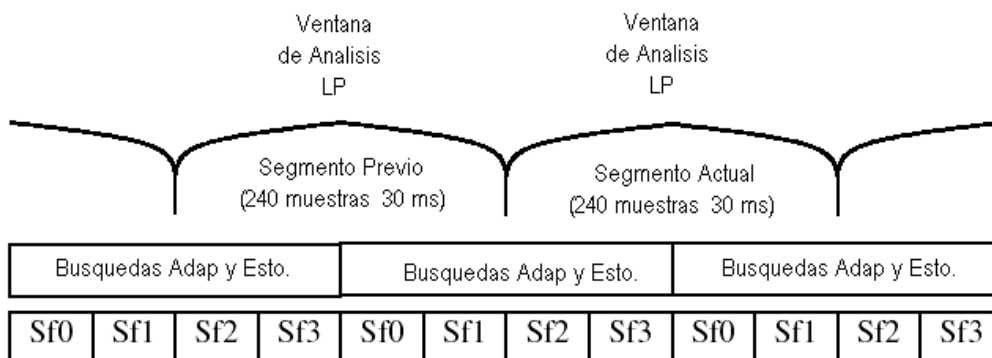


Fig. 6.30 Posicionamiento de los sub-segmentos de análisis. (4, 25, 26)

Para la interpolación de los coeficientes LSP usamos dos conjuntos de 10 coeficientes LSP cada uno. Un conjunto es el obtenido del análisis LP del segmento actual y el otro conjunto es obtenido del análisis LP del segmento previo. Los pesos respectivos para realizar la interpolación se muestran en la Fig. 6.31.



Fig. 6.31 Pesos empleados para el cálculo de la interpolación (4, 25, 26)

La matriz de pesos empleada en el algoritmo es la siguiente:

$$w[2][4] = \{ 0.875, 0.625, 0.375, 0.125, \\ 0.125, 0.375, 0.625, 0.875 \};$$

Para el primer sub-segmento de coeficientes LPS se usan los valores de la  $w[0][1]$  y  $w[1][1]$  de la matriz y se usan de la siguiente manera:

$$LSP_{\text{primer subsegmento}} = 0.875 * LSP_{\text{subsegmento previo}} + 0.125 * LSP_{\text{subsegmento previo}}$$

Para el resto de subsegmentos:

$$LSP_{\text{segundo subsegmento}} = 0.625 * LSP_{\text{subsegmento previo}} + 0.375 * LSP_{\text{subsegmento previo}}$$

$$LSP_{\text{tercer subsegmento}} = 0.375 * LSP_{\text{subsegmento previo}} + 0.625 * LSP_{\text{subsegmento previo}}$$

$$LSP_{\text{cuarto subsegmento}} = 0.125 * LSP_{\text{subsegmento previo}} + 0.875 * LSP_{\text{subsegmento previo}}$$

Nota: La primera vez que el codificador usa el interpolador no hay coeficientes LSP previos, entonces para este caso el algoritmo emplea los siguientes valores:

$$lspold[] = \{0.1885, 0.3142, 0.5655, 0.8168, 1.1938, 1.4451, 1.8221, 2.0735, 2.4504, 2.7646\};$$

La Fig. 6.32 muestra los cuatro subconjuntos de coeficientes LSP calculados por el mecanismo de interpolación descrito.

| Name | Value      | Type      | Radix |
|------|------------|-----------|-------|
| [0]  | 0x80011668 | float[10] | hex   |
| [0]  | 0.1924264  | float     | float |
| [1]  | 0.3259759  | float     | float |
| [2]  | 0.5880786  | float     | float |
| [3]  | 0.8491994  | float     | float |
| [4]  | 1.208527   | float     | float |
| [5]  | 1.460812   | float     | float |
| [6]  | 1.837811   | float     | float |
| [7]  | 2.089202   | float     | float |
| [8]  | 2.45826    | float     | float |
| [9]  | 2.771472   | float     | float |
| [1]  | 0x80011690 | float[10] | hex   |
| [0]  | 0.2002793  | float     | float |
| [1]  | 0.3495277  | float     | float |
| [2]  | 0.6332356  | float     | float |
| [3]  | 0.9139983  | float     | float |
| [4]  | 1.237981   | float     | float |
| [5]  | 1.492236   | float     | float |
| [6]  | 1.869233   | float     | float |
| [7]  | 2.120605   | float     | float |
| [8]  | 2.473978   | float     | float |
| [9]  | 2.785217   | float     | float |
| [2]  | 0x800116B8 | float[10] | hex   |
| [2]  | 0x800116B8 | float[10] | hex   |
| [0]  | 0.2081322  | float     | float |
| [1]  | 0.3730794  | float     | float |
| [2]  | 0.6783926  | float     | float |
| [3]  | 0.9787971  | float     | float |
| [4]  | 1.267434   | float     | float |
| [5]  | 1.52366    | float     | float |
| [6]  | 1.900655   | float     | float |
| [7]  | 2.152009   | float     | float |
| [8]  | 2.489697   | float     | float |
| [9]  | 2.798962   | float     | float |
| [3]  | 0x800116E0 | float[10] | hex   |
| [0]  | 0.2159851  | float     | float |
| [1]  | 0.3966312  | float     | float |
| [2]  | 0.7235497  | float     | float |
| [3]  | 1.043596   | float     | float |
| [4]  | 1.296888   | float     | float |
| [5]  | 1.555084   | float     | float |
| [6]  | 1.932076   | float     | float |
| [7]  | 2.183413   | float     | float |
| [8]  | 2.505415   | float     | float |
| [9]  | 2.812707   | float     | float |

Fig. 6.32 Los 4 sub-conjuntos de coeficientes LSP interpolados.

## 6.10 Análisis de cada uno de los 4 subsegmentos: Búsquedas Adaptiva y Estocástica

De acuerdo a la Fig. 6.3 cada uno de los 4 sub-segmentos de análisis para las búsquedas adaptivas y estocásticas requiere un filtro de síntesis de formantes (Filtro H en la Fig. 6.3) y también un filtro de ponderación perceptual (Filtro W en la Fig. 6.3). Para ello se han calculado previamente los 4 sub-conjuntos de coeficientes LSP para cada sub-segmento respectivamente, el siguiente paso es calcular los correspondientes coeficientes LPC para obtener los filtros H y W. Las ecuaciones (4.1) y (4.2) nos dan las correspondientes expresiones para H y W. Como se ve, es necesario realizar una conversión de los coeficientes LSP a LPC para cada sub-segmento.

### 6.10.1 Conversión de los coeficientes LSP a LCP

Recuérdese de las ecuaciones (6.3) y (6.7) que:

$$A(z) = 1 + a_1 z^{-1} + \dots + a_M z^{-M} = \prod_{i=1}^M (1 - z_i z^{-1})$$

$$A(z) = (P(z) + Q(z))/2$$

El algoritmo implementado se basa en el siguiente análisis: Para obtener los coeficientes LPC ( $a_i$ ) a partir de los coeficientes LSP, se calcula la respuesta impulsional de  $A(z)$ , ya que la respuesta impulsional de un sistema FIR viene dada por los coeficientes  $a_i$  (ver Proakis y Manolakis). Por otro lado,  $A(z)$  se obtiene a partir de  $P(z)$  y  $Q(z)$ .

Recuérdese de (6.11) que:  $P'(z) = \frac{P(z)}{1+z^{-1}}$  y  $Q'(z) = \frac{Q(z)}{1-z^{-1}}$

Y

$$P'(z) = \sum_{i=0}^{2M_1} p'_i z^{-i} \quad \text{y} \quad Q'(z) = \sum_{i=0}^{2M_1} q'_i z^{-i}$$

Recuérdese que tanto  $P'(z)$  como  $Q'(z)$  tienen 5 pares de raíces complejas conjugadas. De los 10 coeficientes LSP para el presente sub-segmento podemos calcular las raíces conjugadas de  $P'(z)$  y  $Q'(z)$ :

Los coeficientes LSP son:  $\{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}\}$  y como están entrelazadas tenemos:

Para  $P'(z) = \{\omega_1, \omega_3, \omega_5, \omega_7, \omega_9\}$

Para  $Q'(z) = \{\omega_2, \omega_4, \omega_6, \omega_8, \omega_{10}\}$

Sabemos que:

$$z = e^{j\omega}$$

Raíces de  $P'(z)$ :  $z_1 = e^{j\omega_1}$ ,  $z^*_1 = e^{-j\omega_1}$ ,  $z_2 = e^{j\omega_3}$ ,  $z^*_2 = e^{-j\omega_3}$ ,  $z_3 = e^{j\omega_5}$ ,  $z^*_3 = e^{-j\omega_5}$ ,  
 $z_4 = e^{j\omega_7}$ ,  $z^*_4 = e^{-j\omega_7}$ ,  $z_5 = e^{j\omega_9}$ ,  $z^*_5 = e^{-j\omega_9}$

Raíces de  $Q'(z)$ :  $z_1 = e^{j\omega_2}$ ,  $z^*_1 = e^{-j\omega_2}$ ,  $z_2 = e^{j\omega_4}$ ,  $z^*_2 = e^{-j\omega_4}$ ,  $z_3 = e^{j\omega_6}$ ,  $z^*_3 = e^{-j\omega_6}$ ,  
 $z_4 = e^{j\omega_8}$ ,  $z^*_4 = e^{-j\omega_8}$ ,  $z_5 = e^{j\omega_{10}}$ ,  $z^*_5 = e^{-j\omega_{10}}$

Entonces:

$$P'(z) = (1 - z_1 z^{-1}) (1 - z_1^* z^{-1}) (1 - z_2 z^{-1}) (1 - z_2^* z^{-1}) (1 - z_3 z^{-1}) (1 - z_3^* z^{-1}) (1 - z_4 z^{-1}) \\ (1 - z_4^* z^{-1}) (1 - z_5 z^{-1}) (1 - z_5^* z^{-1})$$

$$Q'(z) = (1 - z_1 z^{-1}) (1 - z_1^* z^{-1}) (1 - z_2 z^{-1}) (1 - z_2^* z^{-1}) (1 - z_3 z^{-1}) (1 - z_3^* z^{-1}) (1 - z_4 z^{-1}) \\ (1 - z_4^* z^{-1}) (1 - z_5 z^{-1}) (1 - z_5^* z^{-1})$$

Tomando los operandos con pares conjugados:

$$P'(z) = (1 - 2\cos\omega_1 z^{-1} + z^{-2}) (1 - 2\cos\omega_3 z^{-1} + z^{-2}) (1 - 2\cos\omega_5 z^{-1} + z^{-2}) \\ (1 - 2\cos\omega_7 z^{-1} + z^{-2}) (1 - 2\cos\omega_9 z^{-1} + z^{-2}) \quad (6.16)$$

$$Q'(z) = (1 - 2\cos\omega_2 z^{-1} + z^{-2}) (1 - 2\cos\omega_4 z^{-1} + z^{-2}) (1 - 2\cos\omega_6 z^{-1} + z^{-2}) \\ (1 - 2\cos\omega_8 z^{-1} + z^{-2}) (1 - 2\cos\omega_{10} z^{-1} + z^{-2}) \quad (6.17)$$

$$\text{Y de (6.11):} \quad P(z) = P'(z)(1 + z^{-1}) \quad \text{y} \quad Q(z) = Q'(z)(1 - z^{-1})$$

Por lo tanto,  $P(z)$  y  $Q(z)$  son filtros de 6 etapas cada uno, dichas estructuras se muestran en la Fig. 6.33. A dicha estructura se ingresa un impulso unitario para calcular su respuesta impulsional la cual, de acuerdo a (6.3) y (6.7), está definida por el conjunto de los coeficientes LPC ( $a_i$ ) requeridos.

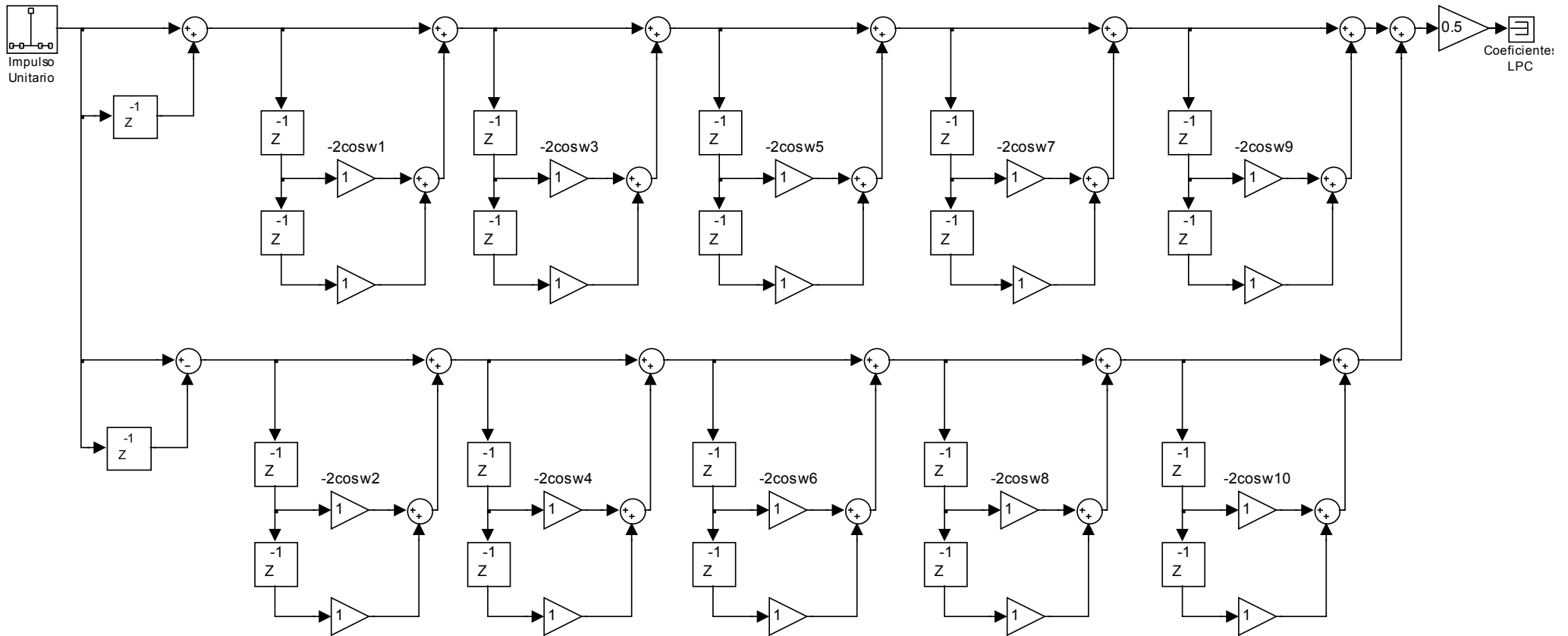


Fig. 6.33 Estructura para el cálculo de los coeficientes LPC a partir de los coeficientes LSP de acuerdo a (6.11) (6.16) y (6.17)

Los coeficientes LPC calculados conforme a la estructura de la Fig. 6.33 teniendo como entrada el primer conjunto de coeficientes LSP de la Fig. 6.32 se muestran en la Fig. 6.34.

| Name | Value        | Type      |
|------|--------------|-----------|
| fci  | 0x80014D80   | float[11] |
| [0]  | 1.0          | float     |
| [1]  | -1.418529    | float     |
| [2]  | 0.606341     | float     |
| [3]  | -0.2077384   | float     |
| [4]  | 0.1447344    | float     |
| [5]  | -0.03344035  | float     |
| [6]  | 0.02920651   | float     |
| [7]  | -0.009532928 | float     |
| [8]  | 0.0919013    | float     |
| [9]  | -0.2204832   | float     |
| [10] | 0.1612832    | float     |

Fig. 6.34 Coeficientes LPC calculados a partir de los coeficientes LSP interpolados

Este primer conjunto de coeficientes LPC se usa en el primer sub-segmento para las búsquedas adaptivas y estocásticas.

### 6.10.2 Fundamentos Teóricos de la Búsqueda Adaptiva y Estocástica

El diagrama de bloques fundamental se muestra en la Fig. 6.35. El objetivo es encontrar la mejor secuencia de excitación de entrada. (4, 25, 26)

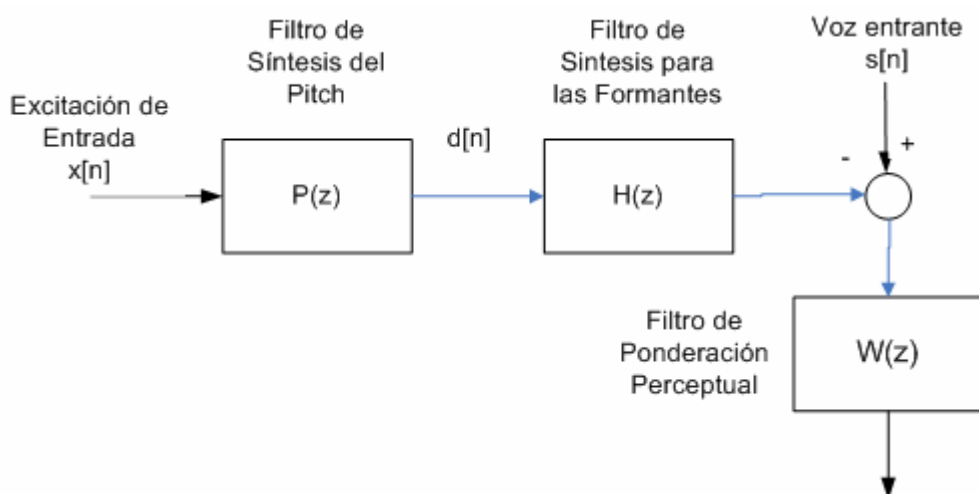


Fig. 6.35 Diagrama de Bloques Básico del modelo CELP (4, 25, 26)

Cambiando la posición del filtro W:

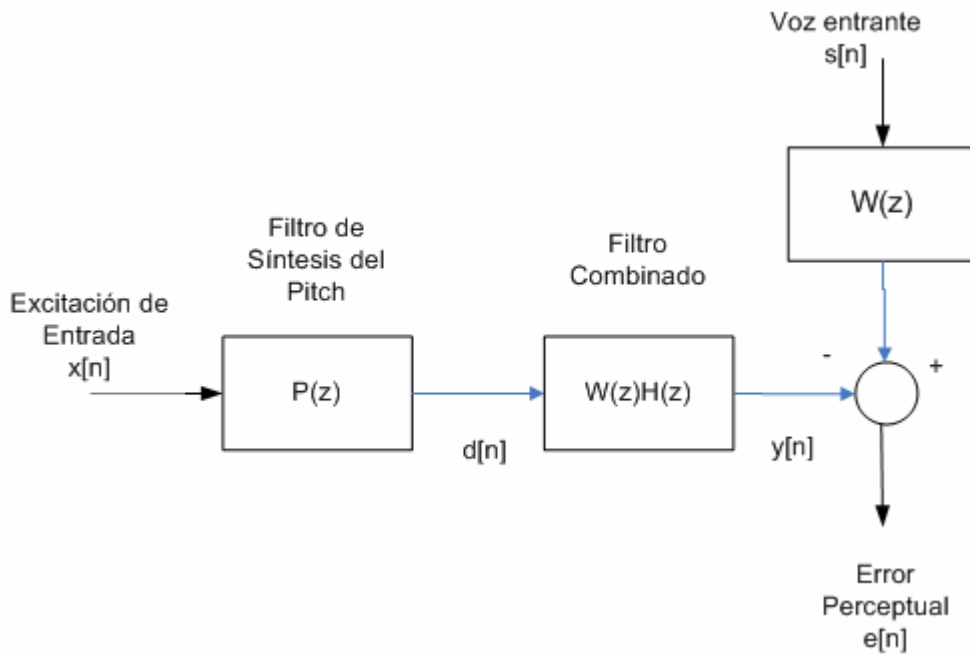


Fig. 6.36 Diagrama de Bloques Básico Modificador del modelo CELP (4, 25, 26)

De las ecuaciones (3.50), (4.1) y (4.2) tenemos que:

$$y[n] = d[n] - \sum_{i=1}^M a_i \gamma^i y[n-i] \quad \text{y} \quad d[n] = x[n] - bd[n-T] \quad (6.18)$$

Donde:

$M$  = orden del predictor short-term

$a_i$  = coeficientes LPC short-term (Fig. 6.34)

$b$  = ganancia long-term

$T$  = Periodo pitch

Recuérdese que:

$$W(z)H(z) = \frac{1}{A(z/\gamma)} = \frac{1}{1 + \sum_{i=1}^M a_i \gamma^i z^{-i}}$$

(6.19)

Aplicando el Método Entrada-Cero / Estado-Cero (*Zero-Input / Zero-State*) para desdoblarse las ecuaciones en (6.18) en  $d_1 r[n]$  y  $d_2 r[n]$ , se obtiene la gráfica de la Fig. 6.37 y el siguiente conjunto de ecuaciones:

La respuesta en estado cero (*Zero-State*) para el filtro de síntesis del pitch:

$$d_1 r[n] = 0 \quad -T \leq n \leq -1; \quad (6.20)$$



$$d1_r[n] = x_r[n] - bd1_r[n-T] \quad 0 \leq n \leq N-1; \quad (6.21)$$

De (6.20) se puede observar que las primeras T muestras de (6.21) pueden ser calculadas sin necesidad de la multiplicación por lo que (6.21) queda como:

$$d1_r[n] = x_r[n] \quad 0 \leq n \leq T-1; \quad (6.22)$$

$$d1_r[n] = x_r[n] - bd1_r[n-T] \quad T \leq n \leq N-1; \quad (6.23)$$

La respuesta Zero-Input para el filtro de síntesis del Pitch es:

$$d2_r[n] = d_{r-1}[n+N] \quad -T \leq n \leq -1; \quad (6.24)$$

$$d2_r[n] = -bd2_r[n-T] \quad 0 \leq n \leq N-1; \quad (6.25)$$

La respuesta total del filtro de síntesis del pitch es:

$$d_r[n] = d1_r[n] + d2_r[n] \quad 0 \leq n \leq N-1; \quad (6.26)$$

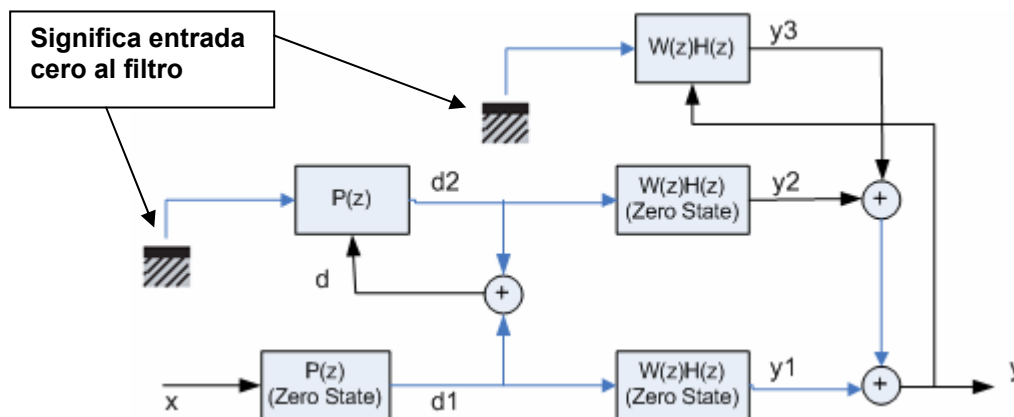


Fig. 6.37 Ilustración de los métodos *zero-input* y *zero-state* para el esquema CELP.

Para el filtro  $W(z)H(z)$  se tienen las siguientes ecuaciones:

$$y1_r[n] = 0 \quad -M \leq n \leq -1; \quad (6.27)$$

$$y1_r[n] = d1_r[n] - \sum_{i=1}^M a_i \gamma^i y1_r[n-i] \quad 0 \leq n \leq N-1; \quad (6.28)$$

$$y_{2,r}[n] = 0 \quad -M \leq n \leq -1; \quad (6.29)$$

$$y_{2,r}[n] = d_{2,r}[n] - \sum_{i=1}^M a_i \gamma^i y_{2,r}[n-i] \quad 0 \leq n \leq N-1; \quad (6.30)$$

$$y_{3,r}[n] = y_{r-1}[n+N] \quad -M \leq n \leq -1; \quad (6.31)$$

$$y_{3,r}[n] = -\sum_{i=1}^M a_i \gamma^i y_{3,r}[n-i] \quad 0 \leq n \leq N-1; \quad (6.32)$$

La respuesta total del filtro combinado  $W(z)H(z)$  es:

$$y_r[n] = y_{1,r}[n] + y_{2,r}[n] + y_{3,r}[n] \quad 0 \leq n \leq N-1; \quad (6.33)$$

La incorporación del codebook de excitación en el esquema de la Fig. 6.37 se muestra en la Fig. 6.38.

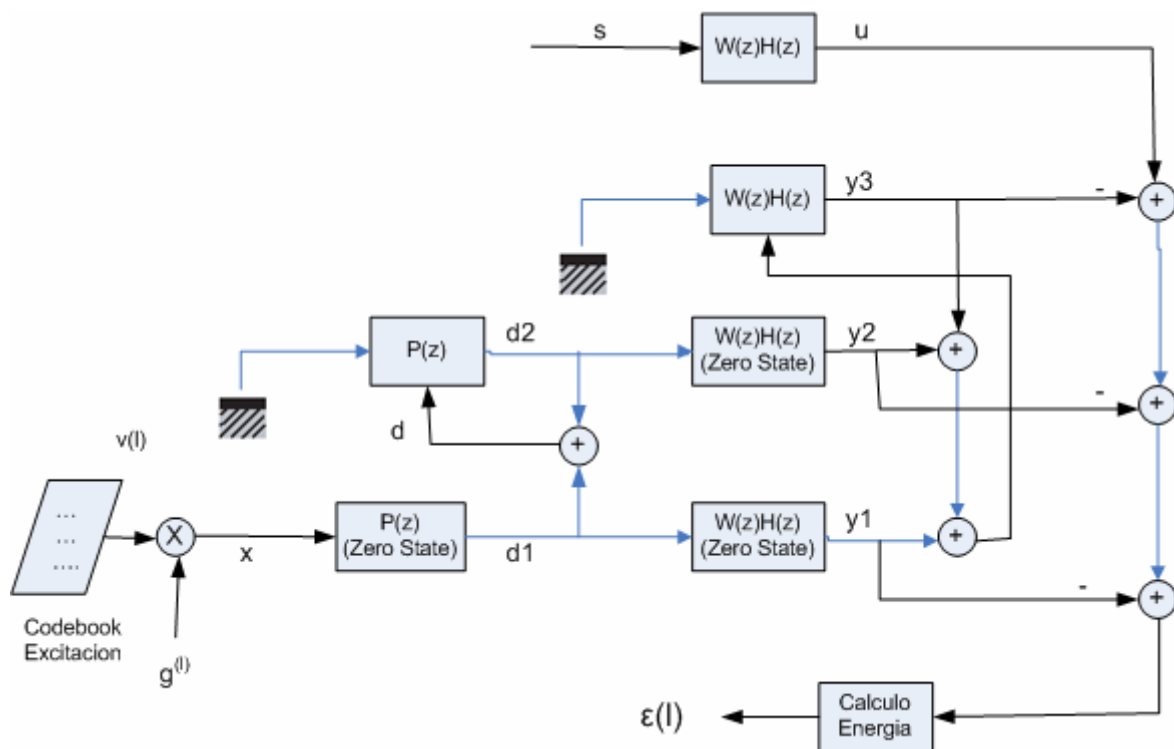


Fig. 6.38 Incorporación del Codebook al esquema CELP.

El codebook de excitación consiste de  $L$  vectores de longitud  $N$  cada uno denotados por:

$$v^{(l)}[n]; \quad l=0, \dots, L-1; \quad n=0, \dots, N-1$$

Cada uno de los vectores es escalado por una ganancia  $g^{(l)}$  y luego pasa a través de los filtros para obtener al final la secuencia de error. Se calcula la energía del error y finalmente se escoge aquella secuencia de error con energía mínima.

De la Fig. 6.38 se puede deducir la expresión para la energía de la secuencia de error:

$$\varepsilon^{(l)} = \sum_{n=0}^{N-1} (u[n] - y1^{(l)}[n] - y2[n] - y3[n])^2 \quad (6.34)$$

Sean:

$$y1_0^{(l)}[n] = \frac{y1^{(l)}[n]}{g^{(l)}}; \quad n = 0, \dots, N-1 \quad (6.35)$$

$$u_0[n] = u[n] - y2[n] - y3[n] \quad (6.36)$$

Reemplazando (6.35) y (6.36) en (6.34) se tiene:

$$\varepsilon^{(l)} = \sum_{n=0}^{N-1} (u_0[n] - g^{(l)} y1_0^{(l)}[n])^2 \quad (6.37)$$

Como se puede observar en la ec (6.37) se tienen dos variables que dependen de la secuencia de excitación: la ganancia  $g^{(l)}$  y la energía del error  $\varepsilon^{(l)}$ , y además uno depende del otro. El camino es calcular  $g^{(l)}$  con el fin de minimizar  $\varepsilon^{(l)}$ . Esto se logra diferenciando  $\varepsilon^{(l)}$  con respecto a  $g^{(l)}$ , con lo cual se tiene que:

$$g^{(l)} = \frac{\sum_{n=0}^{N-1} u_0[n] y1_0^{(l)}[n]}{\sum_{n=0}^{N-1} (y1_0^{(l)}[n])^2} \quad (6.38)$$

Reemplazando (6.38) en (6.37) se tiene:

$$\varepsilon^{(l)} = \left( \sum_{n=0}^{N-1} (u_0[n])^2 \right) - P^{(l)} \quad (6.39)$$

Donde:

$$P^{(l)} = \frac{\left( \sum_{n=0}^{N-1} u_0[n] y1_0^{(l)}[n] \right)^2}{\sum_{n=0}^{N-1} (y1_0^{(l)}[n])^2} \quad (6.40)$$

El problema se reduce a maximizar la expresión (6.40) para todas las secuencias de excitación y se escoge el valor de  $l$  para el cual  $P^{(l)}$  es máximo.

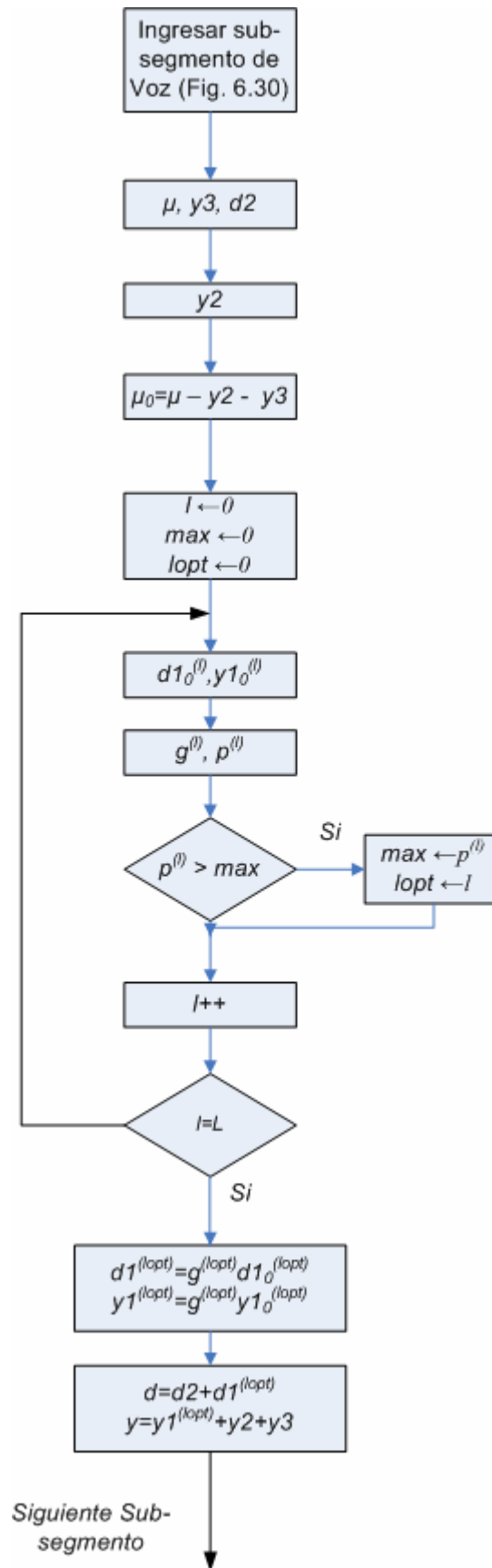


Fig. 6.39 Algoritmo para el cálculo de la mejor secuencia de excitación. Ec (6.40)

### 6.10.3 Filtro de Síntesis del Pitch Optimizado para la minimización del error perceptual

En las figuras Fig. 6.35 y 6.36 los parámetros del filtro de síntesis del Pitch  $P(z)$ ,  $b$  y  $T$ , son calculados para cada sub-segmento de análisis de acuerdo a (4.23), (4.24) y (4.25), y se puede observar que tanto  $b$  como  $T$  se calculan *para minimizar el error cuadrático medio*. En el caso del esquema CELP el objetivo es *minimizar el error **perceptual** cuadrático medio* como se ve en la Fig. 6.35. Entonces se tienen 4 variables a optimizar (4, 25, 26, 40):

- La mejor secuencia de excitación.
- La ganancia asociada a dicha secuencia de excitación.
- El periodo pitch  $T$  del filtro  $P(z)$ .
- La ganancia  $b$  del filtro  $P(z)$ .

El procedimiento es realizar una búsqueda completa para cada uno de dichos parámetros de tal forma que se minimice el error perceptual cuadrático medio. Dicho procedimiento es *muy exigente* computacionalmente ya que son mas lo parámetros que hay que optimizar. El estándar FS1016 usa un esquema simplificado para calcular los 4 parámetros de interés y consta de dos etapas:

- **Primera etapa.** Se asume que la secuencia de excitación es cero y se calculan los parámetros del filtro  $P(z)$ . Esto deriva en un *codebook* adaptivo.
- **Segunda etapa.** Se mantienen constantes los parámetros del filtro  $P(z)$  encontrados en la primera etapa y se procede a calcular la mejor secuencia de excitación. Aquí el *codebook* es estocástico.

### 6.10.4 PRIMERA ETAPA (Deducción del Codebook Adaptivo)

Con la secuencia de excitación igual a cero, (6.27) queda como (se ha cambiado  $\varepsilon$  por  $J_r$ ):

$$J_r = \sum_{n=0}^{N-1} (u_r[n] - y_{2_r}[n] - y_{3_r}[n])^2 \quad (6.41)$$

El objetivo es minimizar  $J_r$ :  $\frac{\partial J_r}{\partial b} = 0$  (6.42)

Tenemos que:

$$d2_r[n] = d_{r-1}[n + N] \quad -T \leq n \leq -1 \quad (6.43)$$

$$d2_r[n] = -bd2_r[n - T] \quad 0 \leq n \leq N - 1 \quad (6.44)$$

$$y2_r[n] = 0 \quad -M \leq n \leq -1 \quad (6.45)$$

$$y2_r[n] = d2_r[n] - \sum_{i=1}^M a_i \gamma^i y2_r[n - i] \quad 0 \leq n \leq N - 1 \quad (6.46)$$

La ecuación (6.46) puede ser expresada conociendo la respuesta impulsional del filtro IIR  $W(z)H(z)$ :

$$y2_r[n] = \sum_{k=0}^n h[k] d2_r[n - k] \quad 0 \leq n \leq N - 1 \quad (6.47)$$

Donde  $h[n]$  puede ser encontrado directamente de los coeficientes LPC (Fig. 6.34).

Entonces (6.42) queda como:

$$\frac{\partial J_r}{\partial b} = (-2) \sum_{n=0}^{N-1} (u_r[n] - y2_r[n] - y3_r[n]) \frac{\partial y2_r[n]}{\partial b} \quad (6.48)$$

$$\text{De (6.47):} \quad \frac{\partial y2_r[n]}{\partial b} = \sum_{k=0}^n h[k] \frac{\partial d2_r[n - k]}{\partial b}, \quad 0 \leq n \leq N - 1 \quad (6.49)$$

El valor de la derivada de  $d2_r$  depende del valor de  $T$ .

**Caso 1:**  $T \geq N$ ,  $d2_r$  depende enteramente de las muestras pasadas, entonces:

$$\frac{\partial d2_r[n]}{\partial b} = -d2_r[n - T] \quad 0 \leq n \leq N - 1 \quad (6.50)$$

Y substituyendo (6.50) en (6.49) tenemos que:

$$\frac{\partial y2_r[n]}{\partial b} = -\sum_{k=0}^n h[k] d2_r[n - k - T], \quad 0 \leq n \leq N - 1 \quad (6.51)$$

Reemplazando (6.51) en (6.48) resulta en:

$$\sum_{n=0}^{N-1} \left( u_r[n] - y_{3_r}[n] - \sum_{k=0}^n h[k] d_{2_r}[n-k] \right) \left( \sum_{k=0}^n h[k] d_{2_r}[n-k-T] \right) = 0 \quad (6.52)$$

Sustituyendo (6.44) en (6.52) resulta en:

$$b = - \frac{\sum_{n=0}^{N-1} (u_r[n] - y_{3_r}[n]) y_{4_r}[n]}{\sum_{n=0}^{N-1} (y_{4_r}[n])^2} \quad (6.53)$$

Donde:

$$y_{4_r}[n] = \sum_{k=0}^n h[k] d_{2_r}[n-k-T] \quad (6.54)$$

Y finalmente sustituyendo (6.53) en (6.41):

$$J(T) = \sum_{n=0}^{N-1} (u_r[n] - y_{3_r}[n])^2 - \frac{\left( \sum_{n=0}^{N-1} (u_r[n] - y_{3_r}[n]) y_{4_r}[n] \right)^2}{\sum_{n=0}^{N-1} (y_{4_r}[n])^2} \quad (6.55)$$

Como se ve en la ecuación (6.53) la expresión para el calculo de b es manejable, pero ahora veamos que pasa cuando  $T \leq N$ .

**Caso 2:**  $N/2 \leq T < N$ . Entonces  $d_{2r}$  (6.44) queda como:

$$d_{2_r}[n] = \begin{cases} -b d_{2_r}[n-T], \dots, 0 \leq n \leq T-1 \\ b^2 d_{2_r}[n-2T], \dots, T \leq n \leq N-1 \end{cases} \quad (6.56)$$

$$y \quad \frac{\partial}{\partial b} d_{2_r}[n] = \begin{cases} -d_{2_r}[n-T], \dots, 0 \leq n \leq T-1 \\ 2b d_{2_r}[n-2T], \dots, T \leq n \leq N-1 \end{cases} \quad (6.57)$$

Se procede como en el caso 1 pero esta vez se obtienen expresiones más complejas, luego de realizar las operaciones, la ecuación para  $b$  es:

$$2b^3 \sum_{n=T}^{N-1} (y_{5_r}[n])^2 + b \left( \sum_{n=0}^{T-1} (y_{4_r}[n])^2 - 2 \sum_{n=T}^{N-1} (u_r[n] - y_{3_r}[n]) y_{5_r}[n] \right) + \sum_{n=0}^{T-1} (u_r[n] - y_{3_r}[n]) y_{4_r}[n] = 0 \quad (6.58)$$

Como se puede apreciar en la expresión (6.58) esta vez el cálculo de  $b$  no es tan directo como en el caso de la ecuación (6.53) ya que se trata de una ecuación cúbica. Para  $T < N/2$  se obtiene expresiones más complejas aún.

Entonces ordenando las expresiones para  $d_{2r}[n]$ :

$$\text{Para } T \geq N: \quad d_{2r}[n] = -bd_{2r}[n-T] \quad 0 \leq n \leq N-1$$

$$\text{Para } N/2 \leq T < N: \quad d_{2r}[n] = \begin{cases} -bd_{2r}[n-T], \dots & 0 \leq n \leq T-1 \\ b^2d_{2r}[n-2T], \dots & T \leq n \leq N-1 \end{cases}$$

$$\text{Para } N/3 \leq T \leq N/2 \quad d_{2r}[n] = \begin{cases} -bd_{2r}[n-T], \dots & 0 \leq n \leq T-1 \\ b^2d_{2r}[n-2T], \dots & T \leq n \leq 2T-1 \\ -b^3d_{2r}[n-3T], \dots & 2T \leq n \leq N-1 \end{cases} \quad (6.59)$$

Como se puede ver de la ecuación (6.59) mientras más pequeño es el valor de  $T$  las expresiones para  $d_{2r}$  involucran valores de  $b$  elevados a distintas potencias. El codebook adaptivo surge de reemplazar todas las potencias por 1 y de esta forma evitar ecuaciones de altos grados, lo cual, según la literatura y el estándar, no afecta el esquema del decodificador ni la calidad del mismo. La expresión para el codebook adaptivo es la siguiente:

$$d_{2r}[n] = -b \begin{cases} d_{2r}[n-T], \dots & 0 \leq n < T \\ d_{2r}[n-2T], \dots & T \leq n < 2T \\ d_{2r}[n-3T], \dots & 2T \leq n < 3T \\ \dots & \dots \end{cases} \quad (6.60)$$

La funcionalidad del Codebook se ilustra en las figuras Fig. 6.40 y Fig. 6.41

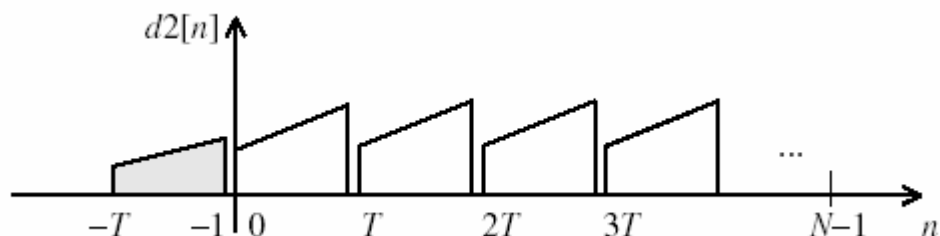


Fig. 6.40 Ilustración del Codebook Adaptivo para valores de  $T < N$  (4)



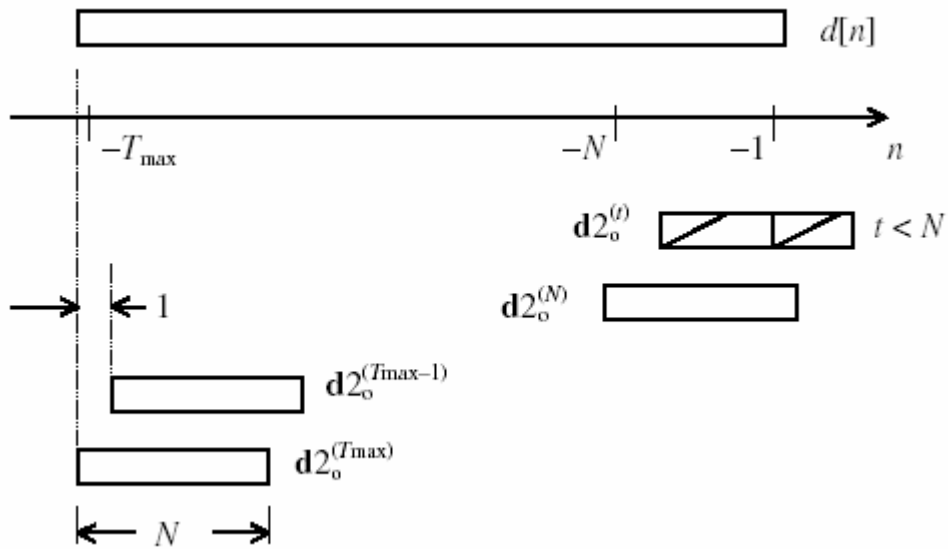


Fig. 6.41 ilustración del Codebook Adaptivo para valores de  $T$  entre  $T_{min}$  y  $T_{max}$  [4]

Una vez localizada la mejor secuencia de excitación, el codebook se actualiza de acuerdo a:

$$d_r[n] = d1_r^{(l)}[n] + d2_r^{(Topr)}[n] \quad n = 0 \text{ a } N - 1 \quad (6.61)$$

$$d_r[n] \leftarrow d_r[n + N] \quad n = -T_{max}, -T_{max} + 1, \dots, -1 \quad (6.62)$$

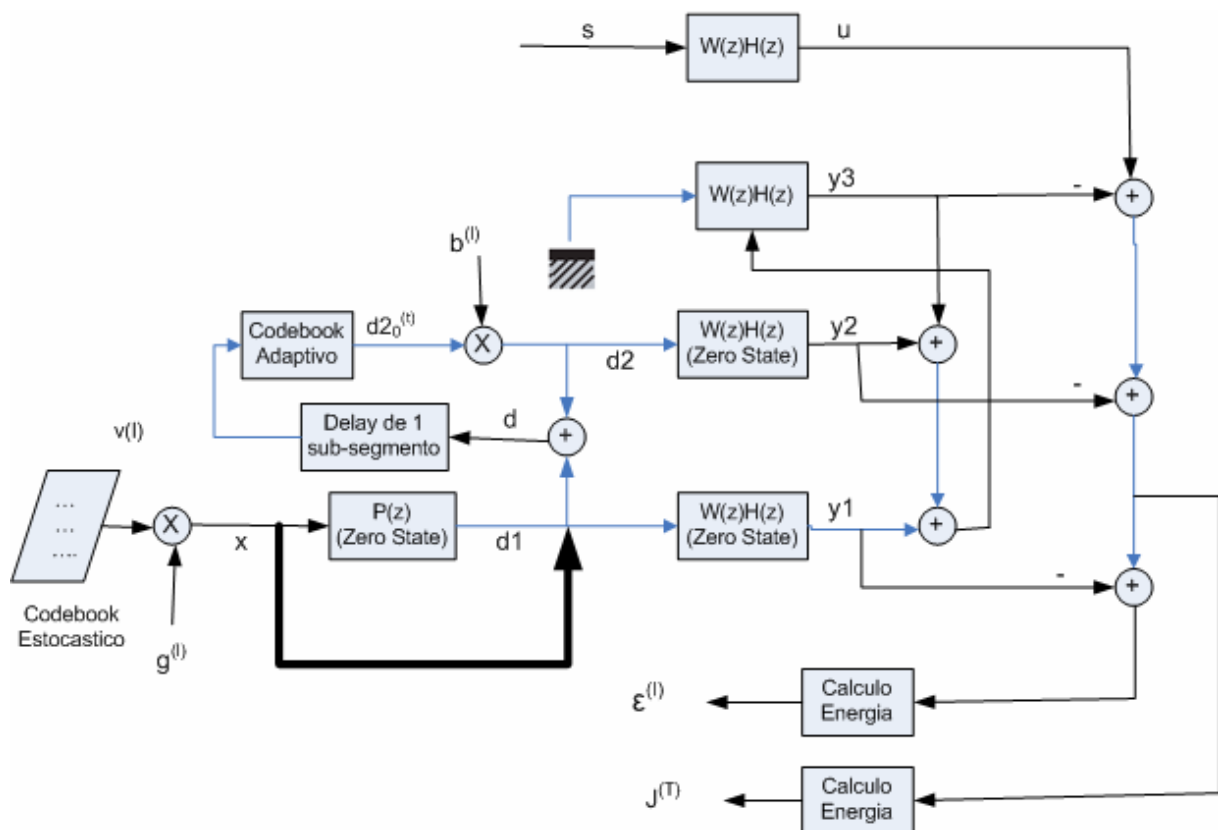


Fig. 6.42 Incorporación del Codebook adaptivo de acuerdo al estándar FS 1016 (4, 25, 26)

Cuando T es mayor que N se puede obviar el filtro P(z) zero-state debido a que la salida es igual a la entrada. El estándar FS1016 prescinde de este filtro incluso cuando T es menor N, lo cual genera una distorsión despreciable en la práctica (de acuerdo al estándar FS1016). El “by-pass” del filtro P(z) zero-state se puede ver en la Fig. 6.42. Por consiguiente la actualización del codebook estocástico se da con la siguiente expresión:

$$d[n] = v^{(lopt)}[n]g^{(lopt)} + d2_0^{(Topt)}[n]b^{(Topt)} \quad n = 0 \text{ a } N - 1 \quad (6.63)$$

$$d[n] \leftarrow d[n + N] \quad n = -T_{max}, -T_{max} + 1, \dots, -1 \quad (6.64)$$

#### 6.10.4.1 Algoritmo de Búsqueda del Codebook Adaptivo

El análisis se realiza con expresiones matriciales:

$$\text{Sea} \quad y2^{(t)} = (WH)d2_0^{(t)}b^{(t)} \quad (6.65)$$

$$\text{Con:} \quad WH = \begin{bmatrix} h0 & 0 & 0 & 0 \\ h1 & h0 & 0 & 0 \\ \cdot & \cdot & \cdot & 0 \\ h_{L-1} & \dots & h1 & h0 \end{bmatrix} \quad (6.66)$$

$$\text{Entonces} \quad J^{(t)} = \|u - y2^{(t)} - y3\|^2 = \|u_0 - b^{(t)}WHd2_0^{(t)}\|^2 \quad (6.67)$$

$$\text{Donde:} \quad u_0 = u - y3$$

Derivando (6.67):

$$\frac{\partial J^{(t)}}{\partial b^{(t)}} = \frac{\partial}{\partial b^{(t)}} \left[ \left( u_0 - b^{(t)}WHd2_0^{(t)} \right)^T \left( u_0 - b^{(t)}WHd2_0^{(t)} \right) \right] = -2u_0^T WHd2_0^{(t)} + 2b^{(t)} \|WHd2_0^{(t)}\|^2 = 0$$

Conduce a:

$$b^{(t)} = \frac{u_0^T WHd2_0^{(t)}}{\|WHd2_0^{(t)}\|^2} \quad (6.68)$$

Expandiendo (6.67):

$$J^{(t)} = \|u_0\|^2 \left[ 1 - \frac{b^{(t)}}{\|u_0\|^2} \left( 2u_0^T WHd2_0^{(t)} - b^{(t)} \|WHd2_0^{(t)}\|^2 \right) \right] \quad (6.69)$$

Sustituyendo (6.68) en (6.69) resulta en:

$$J^{(t)} = \|u_0\|^2 \left[ 1 - \frac{(u_0^T WHd2_0^{(t)})^2}{\|u_0\|^2 \|WHd2_0^{(t)}\|^2} \right] = \|u_0\|^2 - p^{(t)} \quad (6.70)$$

donde: 
$$p^{(t)} = \frac{(u_0^T WHd2_0^{(t)})^2}{\|WHd2_0^{(t)}\|^2} \quad (6.71)$$

El valor de  $t = T_{opt}$  (óptimo) es aquel que maximiza la expresión (6.71). Es decir, se tiene que hacer un barrido para todos los  $t$ ,  $T_{min} \leq t \leq T_{max}$  para maximizar  $p^{(t)}$ , recordar que  $d2_0^{(t)}$  se obtiene del codebook adaptivo:

$$d2_0^{(t)}[n] = \begin{cases} d[n-t], \dots 0 \leq n < t \\ d[n-2t], \dots T \leq n < 2t \\ d[n-3t], \dots 2T \leq n < 3t \\ \dots \dots \dots \end{cases} \quad (6.72)$$

El primer esquema del algoritmo de búsqueda del codebook adaptivo se muestra en la Fig. 6.43.

#### 6.10.4.2 Delta Pitch Delay

Como se observa del algoritmo de búsqueda en la Fig. 6.43 y en la ecuación (6.71), se tiene que realizar un barrido para todos los valores de  $t$  en  $T_{min} \leq t \leq T_{max}$ . Donde  $T_{min} = 20$  y  $T_{max} = 147$ . El estándar especifica que hay 256 valores de  $t$  los cuales dependen de las resoluciones aplicadas a distintos intervalos de valores de pitch:

- Para el intervalo  $\left[ 20, 25\frac{2}{3} \right]$ , la resolución es de  $1/3$ .
- Para el intervalo  $\left[ 26, 33\frac{3}{4} \right]$ , la resolución es de  $1/4$  la cual es la más fina (mujeres).
- Para el intervalo  $\left[ 34, 79\frac{2}{3} \right]$ , la resolución es de  $1/3$ .
- Para el intervalo  $[80, 147]$ , la resolución es de 1.

Los 256 valores de pitch son predeterminados por el estándar y se muestran en la Fig. 6.44

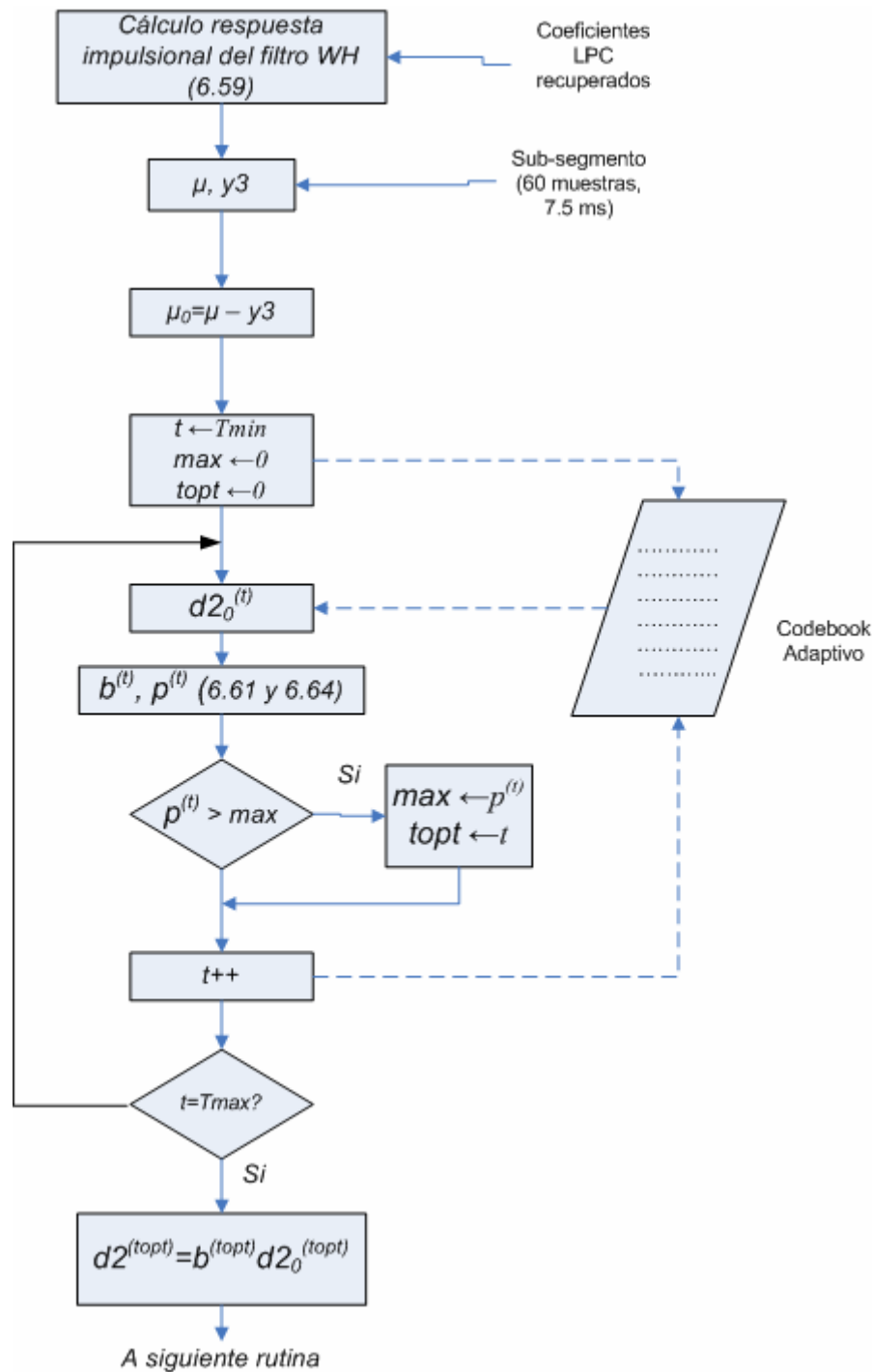


Fig. 6.43 Primer esquema del algoritmo de búsqueda del pitch para el codebook adaptivo.

Hay que mencionar que los valores del periodo pitch  $T$  se encuentran en el rango de 20 a 147 y representan números de muestras. Si se quiere hacer la equivalencia en Hz, el rango correspondiente va de 54 Hz (147 muestras) a 400 Hz (20 muestras).

|           |           |           |           |            |
|-----------|-----------|-----------|-----------|------------|
| 20.00000, | 34.00000, | 50.66667, | 67.33334, | 92.00000,  |
| 20.33334, | 34.33334, | 51.00000, | 67.66667, | 93.00000,  |
| 20.66667, | 34.66667, | 51.33334, | 68.00000, | 94.00000,  |
| 21.00000, | 35.00000, | 51.66667, | 68.33334, | 95.00000,  |
| 21.33334, | 35.33334, | 52.00000, | 68.66667, | 96.00000,  |
| 21.66667, | 35.66667, | 52.33334, | 69.00000, | 97.00000,  |
| 22.00000, | 36.00000, | 52.66667, | 69.33334, | 98.00000,  |
| 22.33334, | 36.33334, | 53.00000, | 69.66667, | 99.00000,  |
| 22.66667, | 36.66667, | 53.33334, | 70.00000, | 100.00000, |
| 23.00000, | 37.00000, | 53.66667, | 70.33334, | 101.00000, |
| 23.33334, | 37.33334, | 54.00000, | 70.66667, | 102.00000, |
| 23.66667, | 37.66667, | 54.33334, | 71.00000, | 103.00000, |
| 24.00000, | 38.00000, | 54.66667, | 71.33334, | 104.00000, |
| 24.33334, | 38.33334, | 55.00000, | 71.66667, | 105.00000, |
| 24.66667, | 38.66667, | 55.33334, | 72.00000, | 106.00000, |
| 25.00000, | 39.00000, | 55.66667, | 72.33334, | 107.00000, |
| 25.33334, | 39.33334, | 56.00000, | 72.66667, | 108.00000, |
| 25.66667, | 39.66667, | 56.33334, | 73.00000, | 109.00000, |
| 26.00000, | 40.00000, | 56.66667, | 73.33334, | 110.00000, |
| 26.25000, | 40.33334, | 57.00000, | 73.66667, | 111.00000, |
| 26.50000, | 40.66667, | 57.33334, | 74.00000, | 112.00000, |
| 26.75000, | 41.00000, | 57.66667, | 74.33334, | 113.00000, |
| 27.00000, | 41.33334, | 58.00000, | 74.66667, | 114.00000, |
| 27.25000, | 41.66667, | 58.33334, | 75.00000, | 115.00000, |
| 27.50000, | 42.00000, | 58.66667, | 75.33334, | 116.00000, |
| 27.75000, | 42.33334, | 59.00000, | 75.66667, | 117.00000, |
| 28.00000, | 42.66667, | 59.33334, | 76.00000, | 118.00000, |
| 28.25000, | 43.00000, | 59.66667, | 76.33334, | 119.00000, |
| 28.50000, | 43.33334, | 60.00000, | 76.66667, | 120.00000, |
| 28.75000, | 43.66667, | 60.33334, | 77.00000, | 121.00000, |
| 29.00000, | 44.00000, | 60.66667, | 77.33334, | 122.00000, |
| 29.25000, | 44.33334, | 61.00000, | 77.66667, | 123.00000, |
| 29.50000, | 44.66667, | 61.33334, | 78.00000, | 124.00000, |
| 29.75000, | 45.00000, | 61.66667, | 78.33334, | 125.00000, |
| 30.00000, | 45.33334, | 62.00000, | 78.66667, | 126.00000, |
| 30.25000, | 45.66667, | 62.33334, | 79.00000, | 127.00000, |
| 30.50000, | 46.00000, | 62.66667, | 79.33334, | 128.00000, |
| 30.75000, | 46.33334, | 63.00000, | 79.66667, | 129.00000, |
| 31.00000, | 46.66667, | 63.33334, | 80.00000, | 130.00000, |
| 31.25000, | 47.00000, | 63.66667, | 81.00000, | 131.00000, |
| 31.50000, | 47.33334, | 64.00000, | 82.00000, | 132.00000, |
| 31.75000, | 47.66667, | 64.33334, | 83.00000, | 133.00000, |
| 32.00000, | 48.00000, | 64.66667, | 84.00000, | 134.00000, |
| 32.25000, | 48.33334, | 65.00000, | 85.00000, | 135.00000, |
| 32.50000, | 48.66667, | 65.33334, | 86.00000, | 136.00000, |
| 32.75000, | 49.00000, | 65.66667, | 87.00000, | 137.00000, |
| 33.00000, | 49.33334, | 66.00000, | 88.00000, | 138.00000, |
| 33.25000, | 49.66667, | 66.33334, | 89.00000, | 139.00000, |
| 33.50000, | 50.00000, | 66.66667, | 90.00000, | 140.00000, |
| 33.75000, | 50.33334, | 67.00000, | 91.00000, | 141.00000, |
|           |           |           |           | 142.00000, |
|           |           |           |           | 143.00000, |
|           |           |           |           | 144.00000, |
|           |           |           |           | 145.00000, |
|           |           |           |           | 146.00000, |
|           |           |           |           | 147.00000  |

Fig. 6.44 Valores del pitch para la búsqueda adaptiva. (26)

Obsérvese que los valores dados en la Fig. 6.44 son valores fraccionarios de los periodos pitch  $T$ , lo que se refleja en los correspondientes valores fraccionarios de las frecuencias Pitch (más precisión).

Ya que el pitch  $T$  entre sub-segmentos adyacentes no difiere mucho se puede realizar una búsqueda basada en el valor del pitch del sub-segmento anterior. Esta búsqueda es conocida como búsqueda *Delta Pitch Delay*. El estándar FS1016 especifica que para los sub-segmentos pares la búsqueda será 32 valores de pitch hacia arriba y 32 valores de pitch hacia abajo con referencia al valor del pitch del segmento anterior. Esto permite dos cosas muy importantes:

- Reduce la complejidad computacional.
- Reduce la tasa de bits ya que para codificar los sub-segmentos pares (donde se realiza la búsqueda delta) se necesitan solo 6 bits y no 8 bits como en el caso de la búsqueda completa (en los sub-segmentos impares).

En esta fase del algoritmo la búsqueda se realiza solo para los pitches enteros de la Fig. 6.44. El tratamiento de los *pitchs* fraccionarios se verá mas adelante. El algoritmo modificado de búsqueda adaptiva de los *pitchs* enteros se muestra en la Fig. 6.46

#### 6.10.4.3 Criterio MSPE modificado

Una mejora de la búsqueda estocástica es mediante la modificación del criterio MSPE. Esto se ilustra en la Fig. 6.45. Cuando se encuentran los máximos correspondientes a un cierto valor del periodo pitch  $T$  se tienen que comprobar en los submúltiplos respectivos ya que pueden haber picos correspondientes a valores de pitch mas adecuados que el valor máximo encontrado. Esto se realiza para sub-segmentos impares.

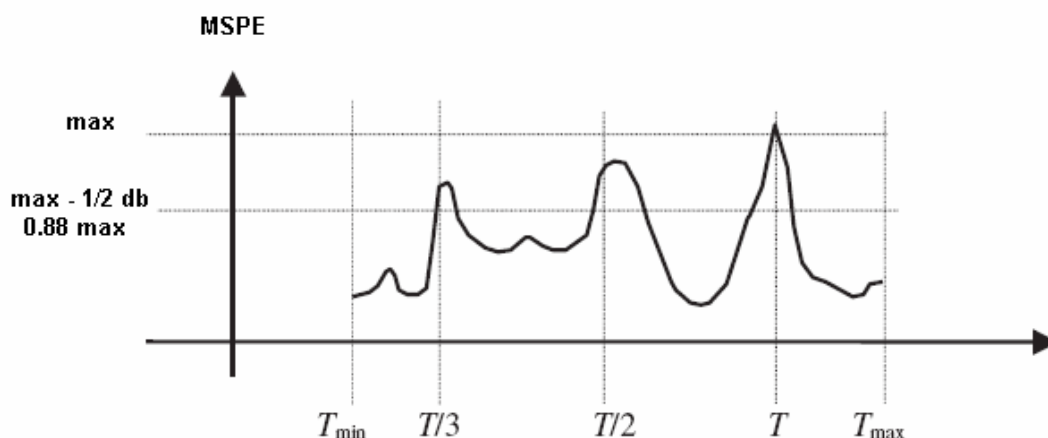


Fig. 6.45 Criterio MSPE modificado para búsqueda adaptiva del pitch  $T$  (4)

En el algoritmo, el criterio de búsqueda en los submúltiplos del pitch es de  $\frac{1}{2}$  dB (0.88). Es decir si el pico en alguno de los submúltiplos es mayor a 0.88 del valor pico máximo, entonces se selecciona dicho submúltiplo como el valor del pitch  $T$ . Ver Fig. 6.47.

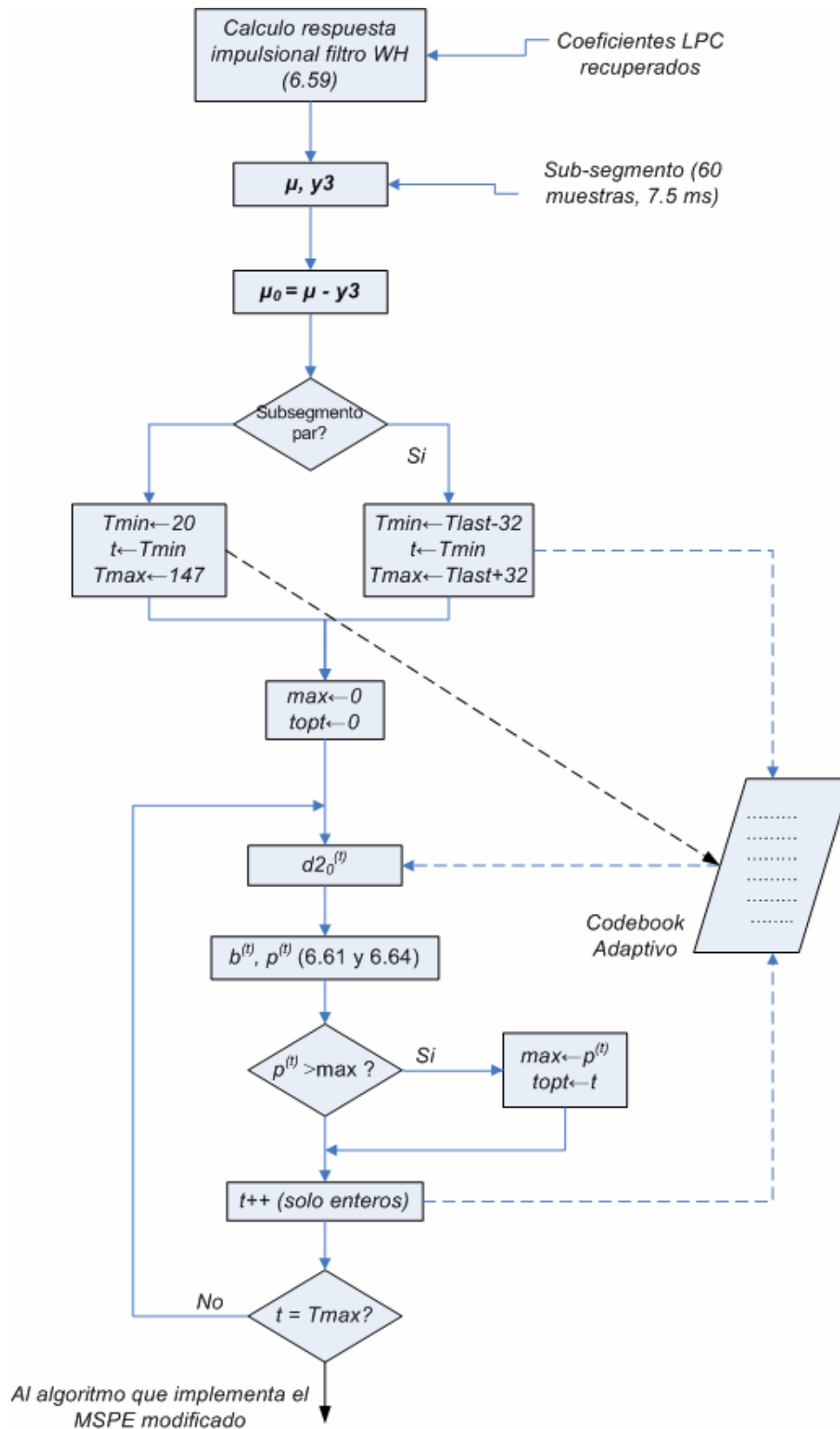


Fig. 6.46 Búsqueda del pitch entero mediante la búsqueda Delta Pitch Delay (4)

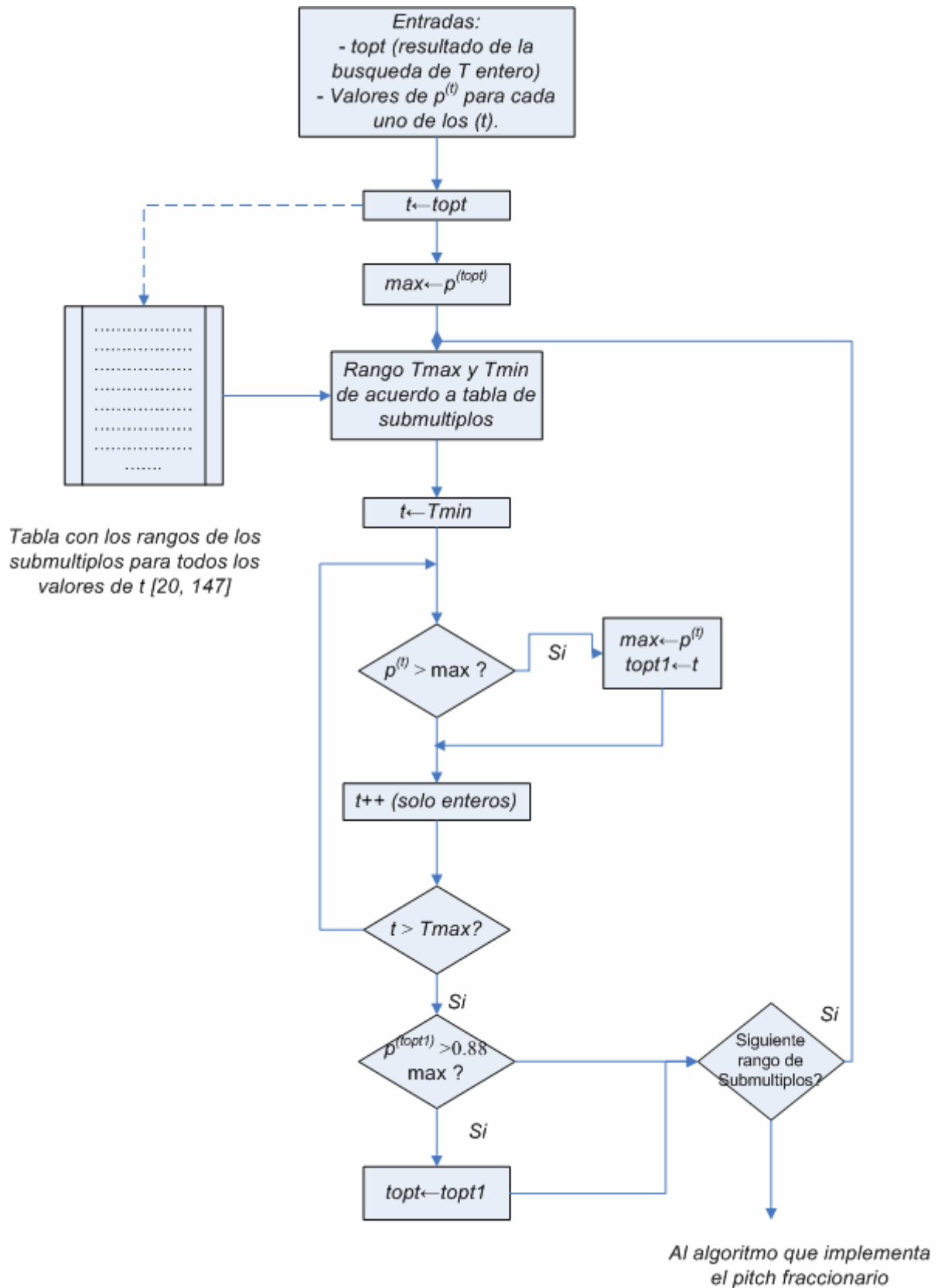


Fig. 6.47 Búsqueda del pitch entero en los submúltiplos aplicando el criterio MSPE Modificado (para sub-segmentos impares) (26)



#### 6.10.4.4 Pitch fraccionario T en la búsqueda del Codebook Adaptivo

Se sabe que una alta resolución en el valor del pitch T mediante el pitch fraccionario reduce la distorsión reverberante y la aspereza de hablantes con pitch agudo (valores pequeños del pitch T). Cuando se introducen los pitch fraccionarios el ruido del codificador es reducido porque se mejora el cálculo del pitch el cual a su vez reduce el ruido en el componente estocástico de excitación. (4, 25, 26, 40).

Remitiéndonos a las expresiones (6.71), (6.72) y a las figuras Fig. 6.46 y Fig. 6.47, se ha visto que hasta ahora la búsqueda del pitch ha sido realizada en el dominio entero y a partir del valor  $T_{opt}$  encontrado se calcula la correspondiente secuencia de excitación  $d2_0^{(T_{opt})}$ , la cual, como se ha visto, es una secuencia de 60 muestras.

De la Fig. 6.44 vemos que los valores del pitch fraccionarios donde se realizarán los cálculos son:  $f = 1/4, 1/3, 1/2, 2/3$  y  $3/4$ . La interpolación se trabajará en función a estos cinco valores. Es decir en base al valor del  $T_{opt}$  hallado se calcularán las secuencias  $d2_0^{(T_{opt} + f)}$  hacia atrás y hacia delante y en base a estas secuencias se volverá a calcular (6.71).

El estándar especifica una técnica de interpolación basada en la función sinc enventanada por una ventana de hamming de 40 puntos:

$$w_0(n) = 0.54 + 0.46 \cos\left(\frac{2\pi n}{40}\right) \quad (6.73)$$

$$w(n, f) = w_0(n + f) \frac{\sin((n + f)\pi)}{(n + f)\pi} \quad (6.74)$$

El rango de  $n$  es de  $-20, -19, \dots, 19$ .

La secuencia  $d2_0$  asociada a un pitch fraccional  $(t+f)$  se da por las siguientes expresiones:

$$dd^{(t)}[n] = \begin{cases} d[n]_p, & -147 \leq n < 0 \\ d2_0^{(t)}[n]_p, & 0 \leq n < 60 \end{cases} \quad (6.75)$$

Y:

$$d2_0^{(t+f)}[n] = \sum_{k=-20}^{19} w(k, f) dd^{(t)}[n - t + k] \quad 0 \leq n < 60 \quad (6.76)$$

La expresión (6.76) crea versiones fraccionalmente retrasadas de los vectores  $d2_0^{(t)}$ .

El algoritmo se ilustra en la figura Fig. 6.48.

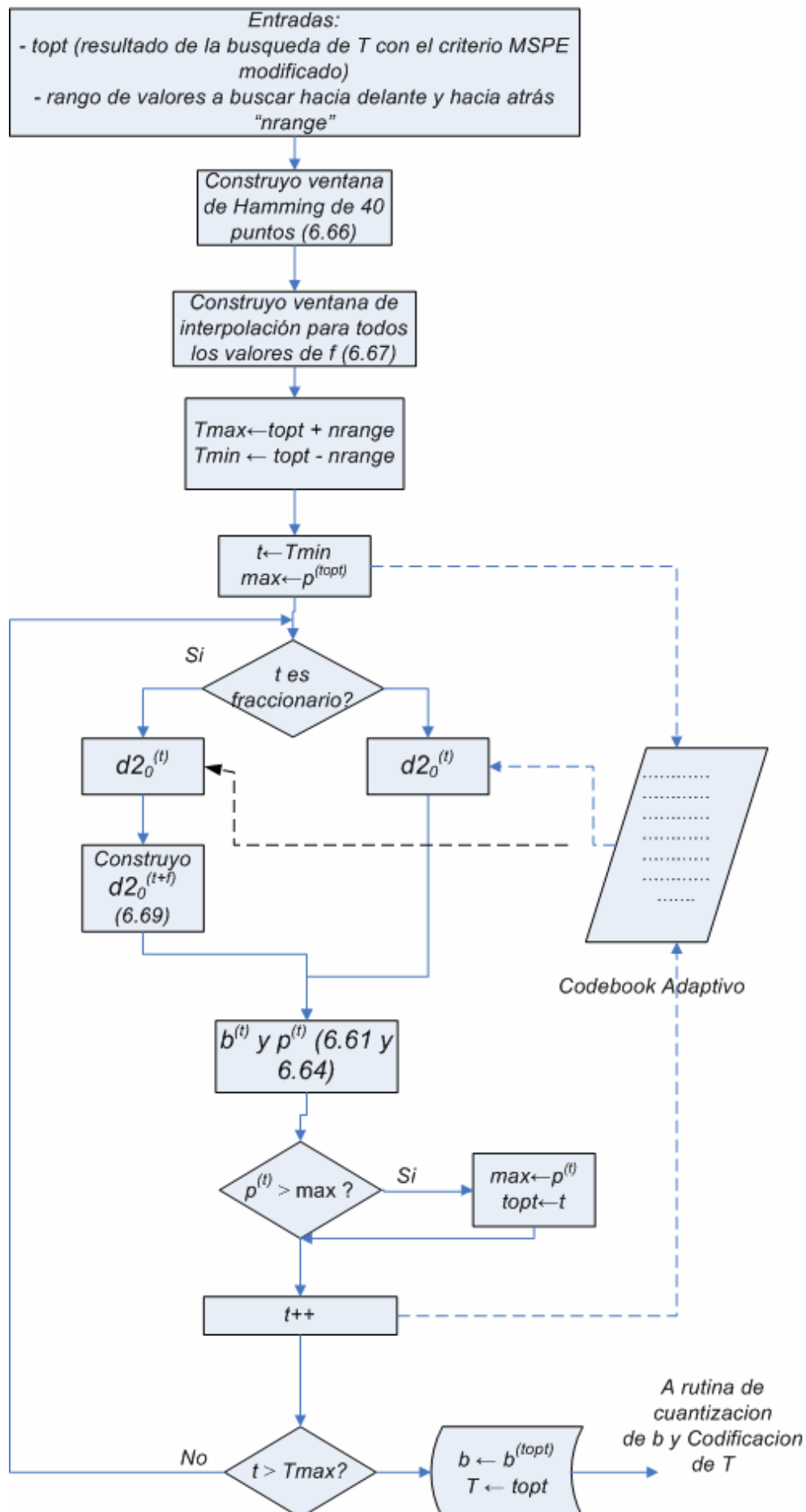


Fig. 6.48 Búsqueda del pitch fraccionario en la periferia del pitch entero derivado de la Fig. 6.47

#### 6.10.4.5 Cuantización de la ganancia b

Una vez calculada la ganancia b es necesario cuantizarla antes de empaquetarla en el bit-stream. Para diseñar el cuantizador el estándar FS 1016 se basa en el criterio de mínima distorsión. El valor de la ganancia siempre se encuentra en el intervalo -1 a 2, y además los bits destinados para su cuantización son 5 (32 niveles). El cuantizador que usa el estándar FS1016 es:

```
-0.993, -0.831, -0.693, -0.555, -0.414, -0.229, 0.0, 0.139,
0.255, 0.368, 0.457, 0.531, 0.601, 0.653, 0.702, 0.745,
0.780, 0.816, 0.850, 0.881, 0.915, 0.948, 0.983, 1.020,
1.062, 1.117, 1.193, 1.289, 1.394, 1.540, 1.765, 1.991
```

#### 6.10.4.6 Codificación del periodo pitch T

Como se ha visto, la búsqueda de T es una búsqueda delta delay. Para sub-segmentos impares se realiza una búsqueda completa de los 256 valores del pitch, mientras que para sub-segmentos pares la búsqueda es parcial y depende del valor de T hallado en el sub-segmento impar anterior.

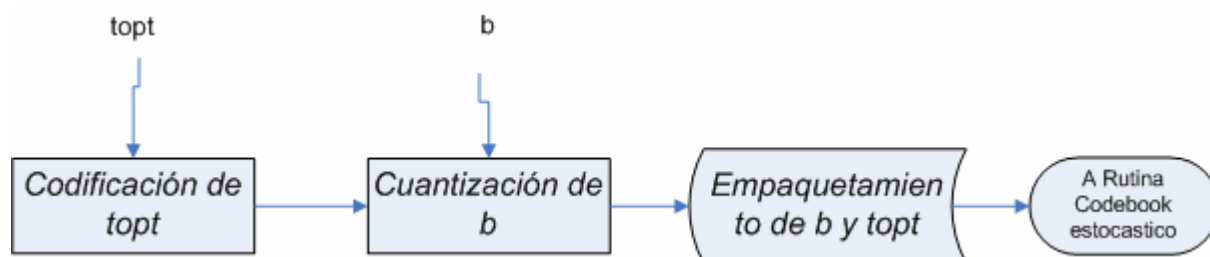
Para segmentos impares, el valor del índice de T hallado es mapeado de acuerdo a los valores mostrados en la Fig. 6.49.

Es decir si el índice resultante de T luego de los algoritmos de las figuras 6.46, 6.47 y 6.48 es 40, entonces el valor de posición 40 es buscado en la tabla de la figura Fig. 6.49 siendo el valor recuperado 0XF5. Es este último valor el que es empaquetado en el bit-stream. Por lo tanto para codificar los valores de T para sub-segmentos impares se necesitan 8 bits.

Esto último no se aplica para los sub-segmentos pares ya que en estos sub-segmentos se realiza la búsqueda delta basados en el valor de T del sub-segmento anterior. Los valores de T para los sub-segmentos pares se codifican directamente y usan solamente 6 bits.

|       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|
| 0X42, | 0XE1, | 0X5F, | 0X12, | 0X7A, | 0X3C, | 0X14, |
| 0X46, | 0XF1, | 0X5E, | 0X0,  | 0X7E, | 0X2E, | 0X1C, |
| 0X47, | 0XF5, | 0X5C, | 0X8,  | 0X6A, | 0X2C, | 0XF9, |
| 0X57, | 0X61, | 0X5D, | 0X6,  | 0X6E, | 0X26, | 0XFA, |
| 0X56, | 0X65, | 0X54, | 0XE,  | 0X6F, | 0X24, | 0XFD, |
| 0X59, | 0X75, | 0X55, | 0XF,  | 0XC4, | 0X49, | 0XE9, |
| 0X58, | 0X71, | 0X50, | 0X7,  | 0XC5, | 0X48, | 0XFE, |
| 0XAE, | 0X6D, | 0X51, | 0X17, | 0XC9, | 0X4C, | 0XE8, |
| 0XBE, | 0X7D, | 0XAA, | 0X1F, | 0XC8, | 0X4D, | 0XFC, |
| 0XBA, | 0X69, | 0XA6, | 0XD,  | 0XC7, | 0X44, | 0X43, |
| 0XB8, | 0X79, | 0XA2, | 0X5,  | 0XCB, | 0X45, | 0XF2, |
| 0XBC, | 0X7B, | 0XB6, | 0X1D, | 0XC6, | 0X40, | 0XF6, |
| 0XAC, | 0X7F, | 0XB2, | 0X15, | 0XCA, | 0X41, | 0XF8, |
| 0XA8, | 0X6B, | 0XBB, | 0XFB, | 0XD6, | 0XA7, | 0X5B, |
| 0X94, | 0XC1, | 0XB0, | 0XFF, | 0XDA, | 0XA3, | 0X5A, |
| 0X84, | 0XC0, | 0XB9, | 0XEB, | 0XDB, | 0XB7, | 0X63, |
| 0X8C, | 0XC3, | 0XB4, | 0XEF, | 0XD7, | 0XB3, | 0X62, |
| 0X9C, | 0XC2, | 0XBD, | 0XED, | 0XD9, | 0XB1, | 0X77, |
| 0X9E, | 0XD2, | 0XA4, | 0XEA, | 0XD5, | 0XB5, | 0X76, |
| 0X8E, | 0XD3, | 0XA0, | 0XEE, | 0XD8, | 0XA5, | 0X52, |
| 0X86, | 0XD1, | 0XA9, | 0XEC, | 0XD4, | 0XA1, | 0X53, |
| 0X96, | 0XD0, | 0XAD, | 0XE6, | 0X20, | 0X97, | 0X66, |
| 0XA,  | 0X30, | 0X95, | 0XE2, | 0X28, | 0X87, | 0X67, |
| 0X2,  | 0X32, | 0X85, | 0XE4, | 0X38, | 0X9F, | 0XCC, |
| 0XB,  | 0X3A, | 0X9D, | 0XE0, | 0X22, | 0X8F, | 0XCD, |
| 0X3,  | 0X31, | 0X8D, | 0XF4, | 0X2A, | 0X81, | 0XAB, |
| 0X1B, | 0X33, | 0X89, | 0XF0, | 0X39, | 0X91, | 0XCF, |
| 0X13, | 0X3B, | 0X99, | 0X60, | 0X29, | 0X9B, | 0XCE, |
| 0X9,  | 0X3F, | 0X88, | 0X64, | 0X21, | 0X8B, | 0XDE, |
| 0X1,  | 0X37, | 0X98, | 0X74, | 0X23, | 0X83, | 0XBF, |
| 0X19, | 0X3E, | 0X90, | 0X70, | 0X2B, | 0X93, | 0XDF, |
| 0X11, | 0X36, | 0X80, | 0X72, | 0X27, | 0X18, | 0XDD, |
| 0XF3, | 0X34, | 0X9A, | 0X73, | 0X2F, | 0X10, | 0XDC, |
| 0XF7, | 0X4A, | 0X8A, | 0X6C, | 0X25, | 0X4,  | 0XAF, |
| 0XE7, | 0X4B, | 0X82, | 0X7C, | 0X2D, | 0XC,  |       |
| 0XE3, | 0X4E, | 0X92, | 0X68, | 0X3D, | 0X16, |       |
| 0XE5, | 0X4F, | 0X1A, | 0X78, | 0X35, | 0X1E, |       |

Fig. 6.49 Valores para codificar el pitch T para segmentos impares.

Fig. 6.50 Codificación de *topt*, cuantización de *b* y empaquetamiento.

## 6.10.5 SEGUNDA ETAPA (Búsqueda en el Codebook Estocástico)

### 6.10.5.1 Estructura del Codebook Estocástico

La estructura básica del codebook estocástico es una matriz de  $L \times N$ , es decir  $L$  vectores de longitud  $N$ . Para el caso del estándar FS 1016, este especifica 512 vectores de tamaño 60 cada uno (sub-segmento de 7.5 ms). Un problema con este codebook es el tamaño de memoria requerido. Para superar este problema el estándar recomienda el uso de un *overlapping-codebook*, es decir vectores consecutivos que comparten muestras en común ( 4, 25, 26, 29, 31, 32, 34, 36).

La estructura de un *overlapping-codebook* se ilustra en la Fig. 6.51.

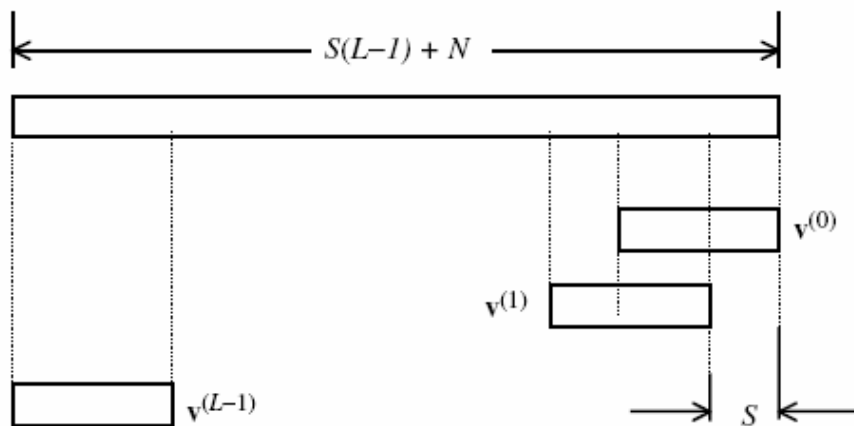


Fig. 6.51 Estructura de un Overlapping-Codebook (4)

Para el caso del Codebook usado en el estándar FS1016,  $N = 60$ ,  $S=2$  y  $L = 512$ , por lo que el tamaño del codebook se reduce a  $(512-1)*2 + 60 = 1082$ .

Para que el *overlapping-codebook* sea igual de eficiente que un *nonoverlapping-codebook*, es necesario que el *overlapping-codebook* contenga muestras de ruido blanco no correlacionados entre sí. El estándar FS1016 utiliza una forma especial de muestras aleatorias para conformar el *overlapping-codebook*, las cuales se derivan de una fuente de ruido blanco Gaussiano de varianza unitaria, centrado en 1.2 y cuantizado con los valores ternarios  $[-1, 0, 1]$ , tal como se especifica en el estándar. Dicho *codebook* se muestra en la Fig. 6.51. Este *codebook* presenta varias ventajas, entre ellas la reducción de la carga computacional ya que el 77% de las muestras son cero. También este *codebook* es compacto y causa pequeña degradación en la calidad de la voz.

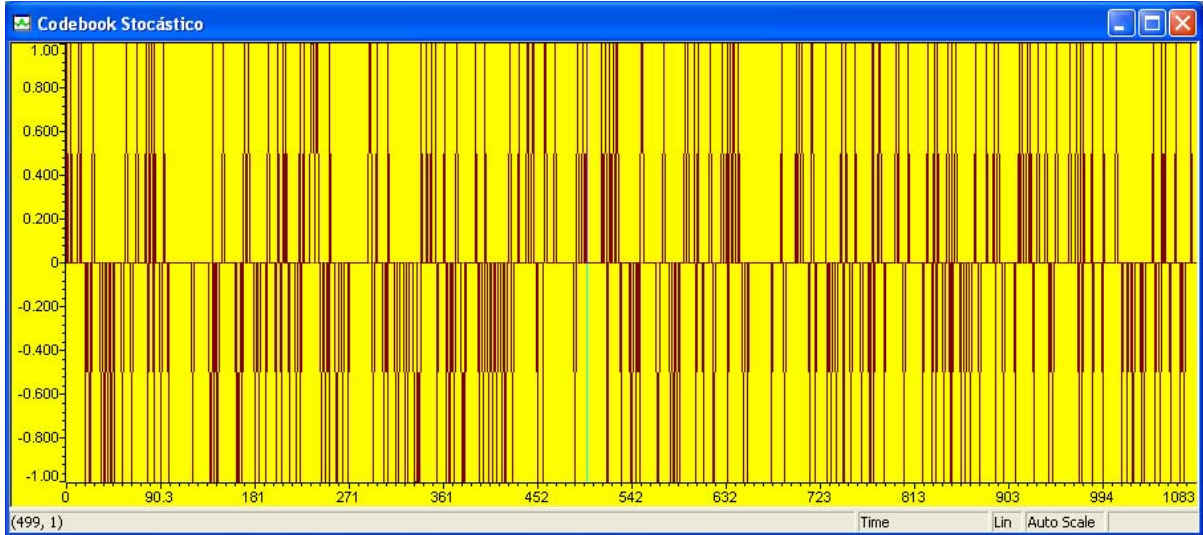


Fig. 6.52 Codebook estocástico usado por el estandar FS1016. (4, 31)

### 6.10.5.2 Algoritmo de Búsqueda del Codebook Estocástico

En la primera etapa (sección 6.10.4), se realizó la búsqueda del codebook adaptivo y se determinaron los valores del periodo  $T_{opt}$  y de la ganancia  $b^{(T_{opt})}$  correspondiente. En esa etapa se determinó que la excitación estocástica fuera igual a cero, por lo que  $y1^{(l)} = 0$  (ver Fig. 6.42). La ecuación de análisis fue la siguiente:

$$J_r = \sum_{n=0}^{N-1} (u_r[n] - y2_r[n] - y3_r[n])^2 \quad (6.77)$$

Ahora, en esta etapa, ya se han determinado  $T_{opt}$  y  $b^{(T_{opt})}$  y por lo tanto la correspondiente excitación adaptiva es:

$$d^{(T_{opt})}[n] = d2_0^{(T_{opt})}[n] b^{(T_{opt})} \quad n = 0 \text{ a } N - 1 \quad (6.78)$$

Y el valor de  $y2^{(T_{opt})}$  correspondiente es:

$$y2^{(T_{opt})}[n] = (WH) d2_0^{(T_{opt})}[n] b^{(T_{opt})} \quad (6.79)$$

Ahora en esta etapa de análisis, la incógnita es hallar la mejor excitación estocástica manteniendo constante la excitación adaptiva hallada en la etapa anterior. La ecuación (6.34) es ahora:

$$\varepsilon^{(l)} = \sum_{n=0}^{N-1} (u[n] - y1^{(l)}[n] - y2^{(T_{opt})}[n] - y3[n])^2 \quad (6.80)$$

Al igual que en el caso del codebook adaptivo, el análisis se realiza con expresiones matriciales:

$$\text{Sea} \quad y1^{(l)}[n] = (WH)v^{(l)}[n]g^{(l)} \quad (6.81)$$

$$\text{Con:} \quad WH = \begin{bmatrix} h0 & 0 & 0 & 0 \\ h1 & h0 & 0 & 0 \\ \cdot & \cdot & \cdot & 0 \\ h_{L-1} & \cdot & h1 & h0 \end{bmatrix} \quad (6.82)$$

$$\text{Entonces} \quad \varepsilon^{(l)} = \|u - y1^{(l)} - y2^{(Topl)} - y3\|^2 = \|u_0 - g^{(l)}WHv^{(l)}\|^2 \quad (6.83)$$

$$\text{Donde:} \quad u_0 = u - y2^{(Topl)} - y3$$

Derivando (6.83):

$$\frac{\partial \varepsilon^{(l)}}{\partial g^{(l)}} = \frac{\partial}{\partial g^{(l)}} \left[ (u_0 - g^{(l)}WHv^{(l)})^T (u_0 - g^{(l)}WHv^{(l)}) \right] = -2u_0^T WHv^{(l)} + 2g^{(l)} \|WHv^{(l)}\|^2 = 0$$

Conduce a:

$$g^{(l)} = \frac{u_0^T WHv^{(l)}}{\|WHv^{(l)}\|^2} \quad (6.84)$$

Expandiendo (6.75):

$$\varepsilon^{(l)} = \|u_0\|^2 \left[ 1 - \frac{g^{(l)}}{\|u_0\|^2} \left( 2u_0^T WHv^{(l)} - g^{(l)} \|WHv^{(l)}\|^2 \right) \right] \quad (6.85)$$

Sustituyendo (6.77) en (6.78) resulta en:

$$\varepsilon^{(l)} = \|u_0\|^2 \left[ 1 - \frac{(u_0^T WHv^{(l)})^2}{\|u_0\|^2 \|WHv^{(l)}\|^2} \right] = \|u_0\|^2 - p^{(l)} \quad (6.86)$$

$$\text{donde:} \quad p^{(l)} = \frac{(u_0^T WHv^{(l)})^2}{\|WHv^{(l)}\|^2} \quad (6.87)$$

El esquema del algoritmo se ilustra en la Fig. 6.53.

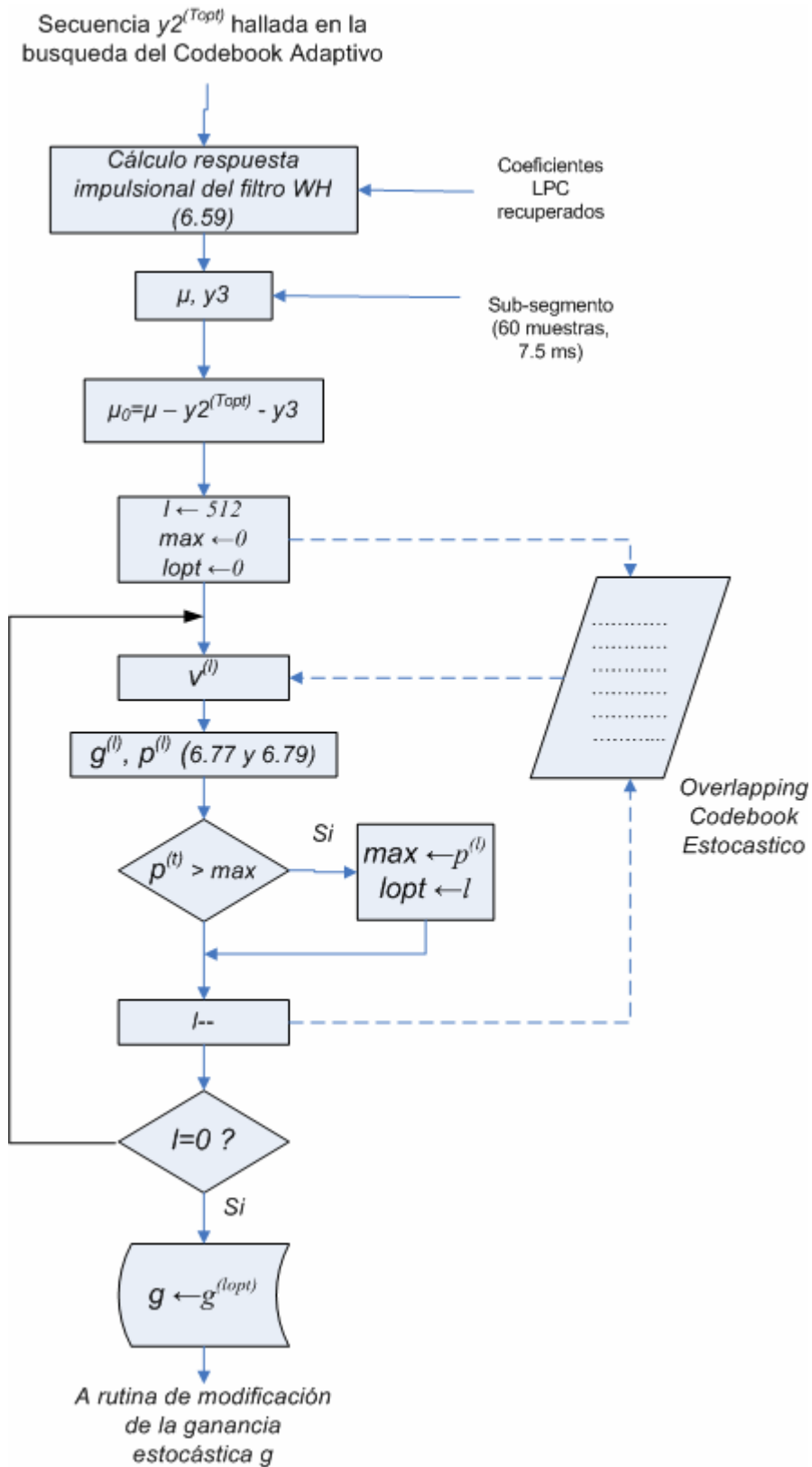


Fig. 6.53 Algoritmo de búsqueda para el codebook estocástico.



### 6.10.5.3 Modificación de la ganancia estocástica

Si bien es cierto usar un overlapping codebook de valores ternarios presenta varias ventajas, también presenta desventajas como la introducción de componentes ruidosos a la voz sintética, disminuyendo la calidad. Estos componentes de ruido no pueden ser removidos por el filtro de síntesis pero si por el postfiltro. Para superar este inconveniente, el estándar FS1016 realiza una modificación de la ganancia estocástica mediante la atenuación adaptiva de la misma en base a la eficiencia del codebook adaptivo. Este esquema mejora la calidad subjetiva de la voz porque reduce la aspereza y el ruido de cuantización en los segmentos "sonoros". La eficiencia del codebook adaptivo puede medirse por la siguiente expresión (4, 37):

$$R = \frac{(u - y_3)^T (u - y_3 - y_2^{(Opt)})}{\|u - y_3\|^2} \quad (6.88)$$

Mientras mas eficiente es el codebook adaptivo, menor es la norma del vector  $(u - y_3 - y_2^{(Opt)})$ , resultando en un valor de R mas pequeño. El factor de escala por el que será multiplicado el valor de la ganancia hallada en la búsqueda estocástica es:

$$sf(R) = \begin{cases} 0.2, & |R| < 0.04 \\ 1.4\sqrt{|R|}, & |R| > 0.81 \\ \sqrt{|R|}, & \text{otros} \end{cases} \quad (6.89)$$

Por lo tanto:

$$g' = sf(R) \cdot g^{(Opt)} \quad (6.90)$$

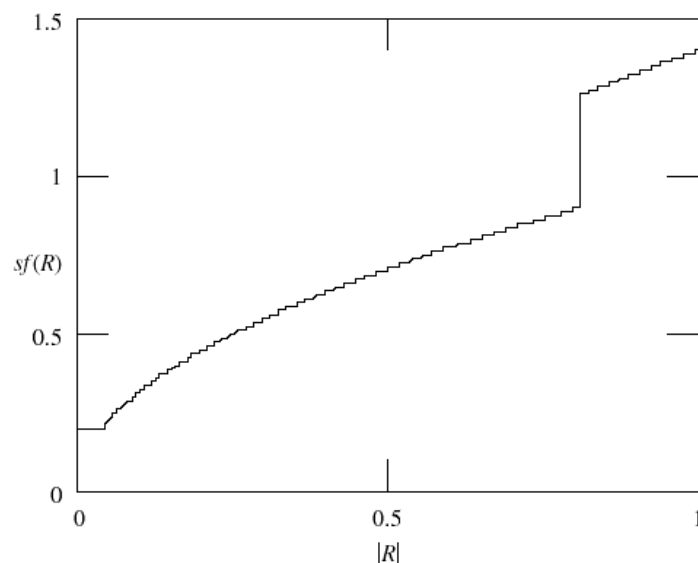


Fig. 6.54 Modificación de la ganancia estocástica vs.  $|R|$  (4)

#### 6.10.5.4 Cuantización de la ganancia estocástica modificada

Para cuantizar la ganancia estocástica, el estándar especifica un cuantizador de 32 niveles (5 bits). En el presente trabajo se ha implementado un cuantizador de 64 niveles (6 bits) para lograr mayor calidad de la voz ya que en la práctica se ha encontrado que usando más bits para cuantizar valores cercanos a cero permite mayor calidad en la voz decodificada.

Para diseñar el cuantizador nos valemos de la herramienta Matlab, la cual implementa el algoritmo de Lloyd. El conjunto de valores de entrenamiento se obtiene de secuencias de voz, de las cuales se han extraído las ganancias estocásticas correspondientes.

Se usa un grupo de 2728 ganancias como conjunto de entrenamiento para el diseño del cuantizador basado en el algoritmo de Lloyd.

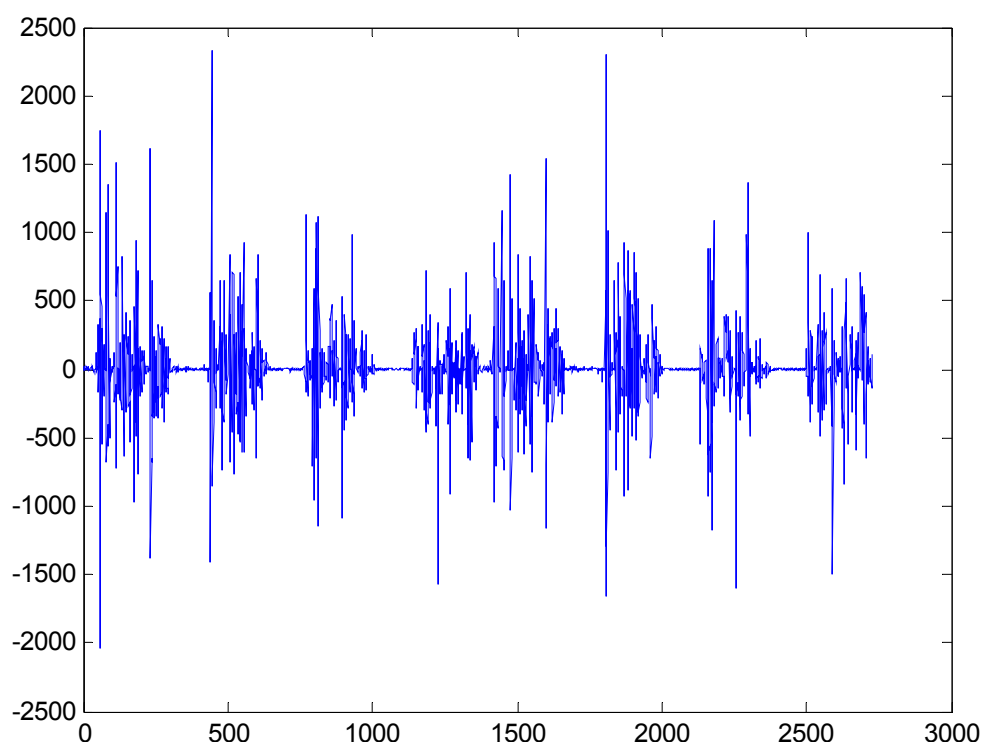


Fig. 6.55 Conjunto de ganancias para el entrenamiento del cuantizador.

El algoritmo de Lloyd es un algoritmo para el diseño de cuantizadores de mínima distorsión basado en las propiedades estadísticas resultantes del conjunto de entrenamiento de entrada. En este caso el número de niveles es 64 (6 bits). El histograma del conjunto de entrenamiento se muestra en la Fig. 6.56.

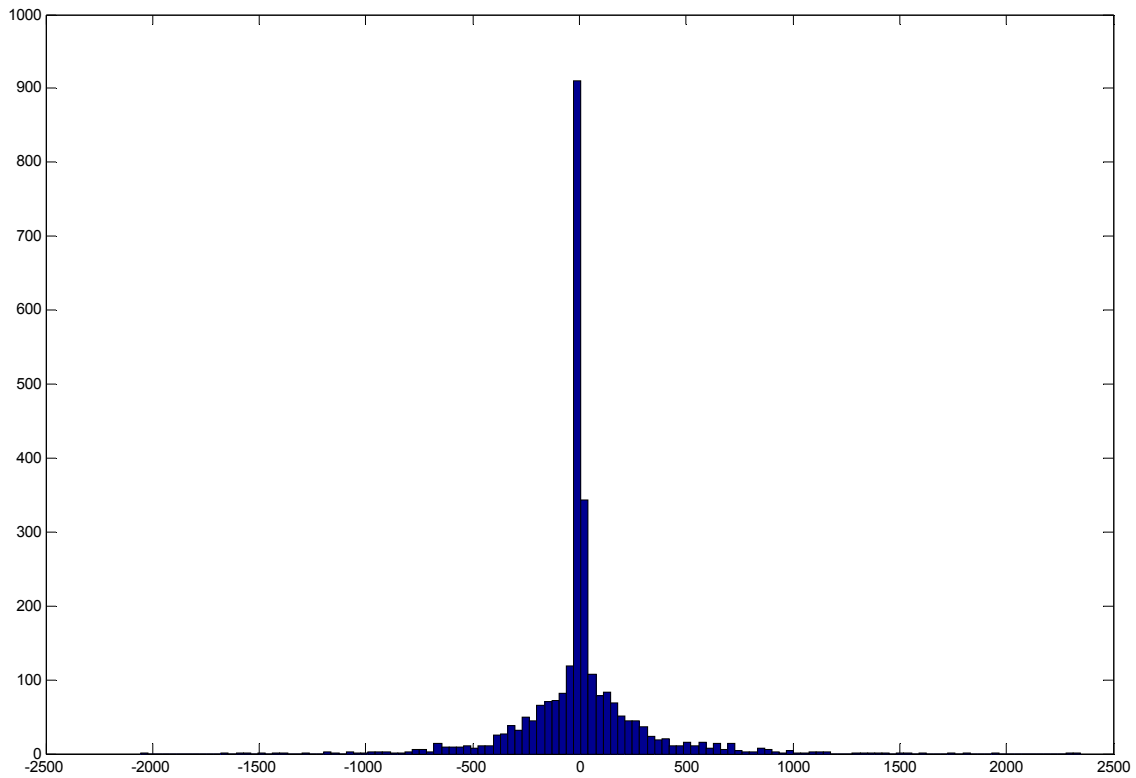


Fig. 6.56 Histograma del conjunto de valores de entrenamiento.

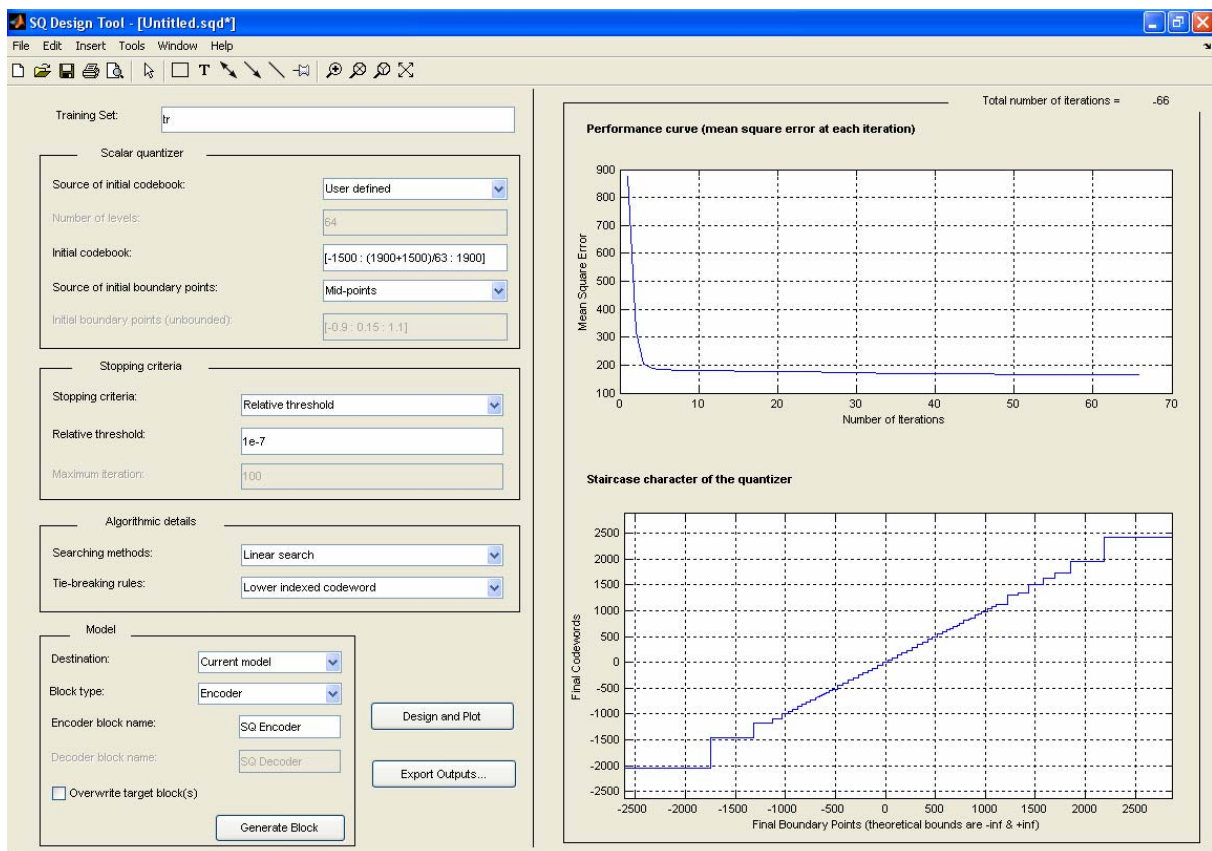


Fig. 6.57 La herramienta para el diseño de cuantizadores escalares, SQDTool. (46)

El cuantizador diseñado se muestra en la Fig 6.58.

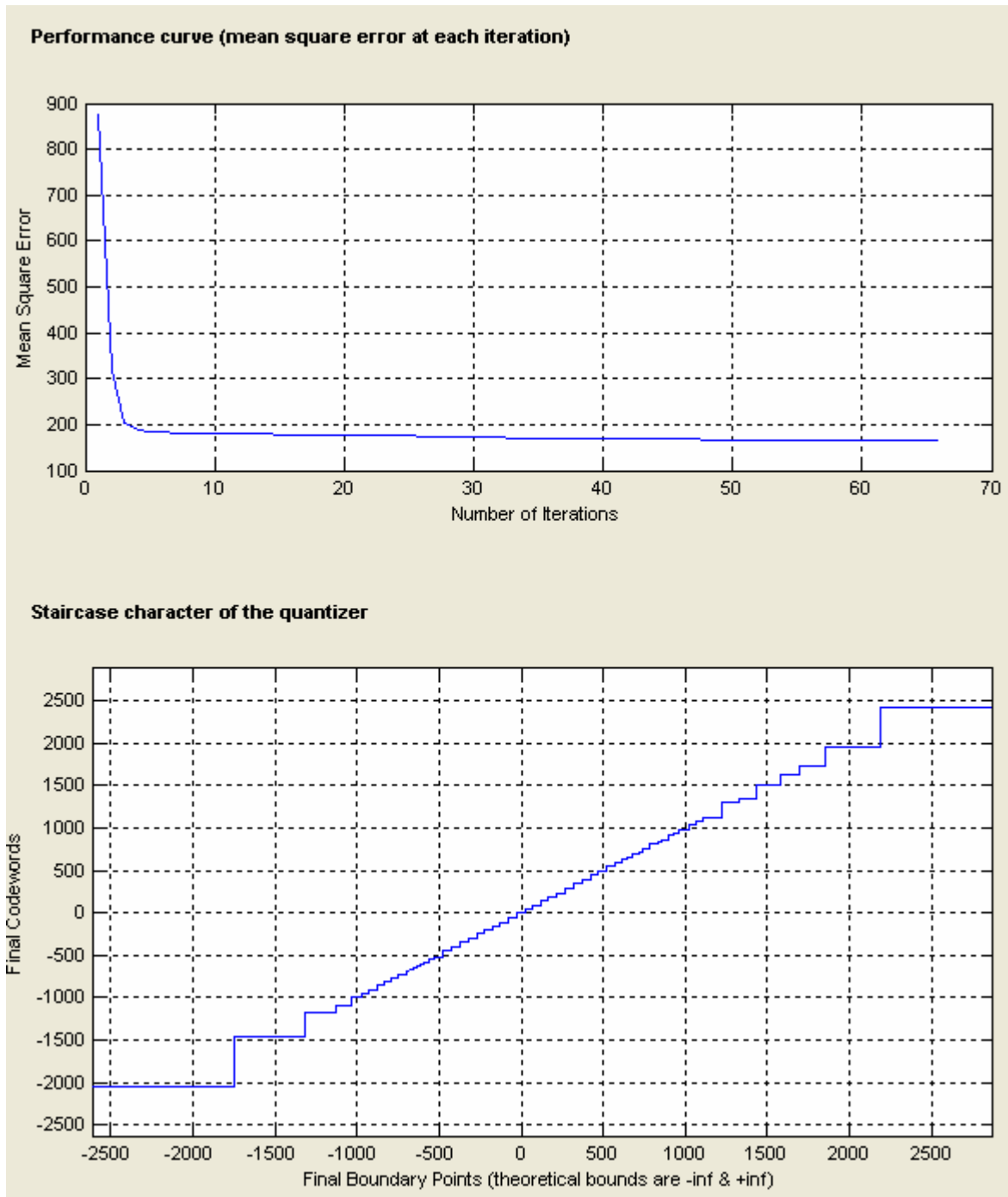


Fig 6.58 Cuantizador diseñado para cuantificar la ganancia estocástica.

Los valores de los niveles del cuantizador diseñado son los siguientes:

|                 |               |              |
|-----------------|---------------|--------------|
| -2040.577027,   | -295.5640324, | 697.3169353, |
| -1454.314616,   | -247.4046494, | 706.454361,  |
| -1172.770935,   | -197.1196924, | 724.7807617, |
| -1085.5979,     | -156.6444271, | 752.1510927, |
| -993.736755,    | -109.3834816, | 816.8738607, |
| -955.6785073,   | -53.51185921, | 841.5972166, |
| -906.8666583,   | 0.0,          | 866.3574827, |
| -857.560669,    | 48.85696963,  | 917.209717,  |
| -809.7743833,   | 95.52581763,  | 934.8488157, |
| -774.6642546,   | 143.6939185,  | 987.0843913, |
| -723.6878457,   | 188.4293087,  | 1051.535278, |
| -676.3752237,   | 237.2142683,  | 1088.538574, |
| -664.9915465    | 289.59412,    | 1132.033529, |
| , -650.4023924, | 345.2554195,  | 1308.075806, |
| -624.307434,    | 397.8680812,  | 1347.394165, |
| -608.8291423,   | 449.1075825,  | 1508.457642, |
| -582.3727538,   | 492.3514536,  | 1641.059143, |
| -550.8025777,   | 545.9058432,  | 1742.552856, |
| -514.6395902,   | 591.7544557,  | 1957.25769,  |
| -451.0640335,   | 604.523254,   | 2419.623291  |
| -399.0209931,   | 629.1141153,  |              |
| -343.3811465,   | 653.0890231,  |              |

Tras analizar el rendimiento del cuantizador diseñado, se encontró que dicho cuantizador introducía una cierta cantidad de ruido a la señal de voz sintetizada, esto es debido a que hay poca sensibilidad en los niveles del cuantizador cercanos a 0. El codebook modificado que ofrece resultados óptimos en la calidad de la voz sintetizada es el siguiente:

|                 |               |              |
|-----------------|---------------|--------------|
| -1454.314616,   | -197.1196924, | 629.1141153, |
| -1172.770935,   | -156.6444271, | 653.0890231, |
| -1085.5979,     | -109.3834816, | 697.3169353, |
| -993.736755,    | -53.51185921, | 706.454361,  |
| -906.8666583,   | -25.8         | 724.7807617, |
| -857.560669,    | -12.9         | 752.1510927, |
| -809.7743833,   | 0.0,          | 816.8738607, |
| -774.6642546,   | 12.9          | 841.5972166, |
| -723.6878457,   | 25.8          | 866.3574827, |
| -676.3752237,   | 48.85696963,  | 917.209717,  |
| -664.9915465    | 95.52581763,  | 934.8488157, |
| , -650.4023924, | 143.6939185,  | 987.0843913, |
| -624.307434,    | 188.4293087,  | 1051.535278, |
| -608.8291423,   | 237.2142683,  | 1088.538574, |
| -582.3727538,   | 289.59412,    | 1132.033529, |
| -550.8025777,   | 345.2554195,  | 1308.075806, |
| -514.6395902,   | 397.8680812,  | 1347.394165, |
| -451.0640335,   | 449.1075825,  | 1508.457642, |
| -399.0209931,   | 492.3514536,  | 1641.059143, |
| -343.3811465,   | 545.9058432,  | 1742.552856, |
| -295.5640324,   | 591.7544557,  | 1957.25769,  |
| -247.4046494,   | 604.523254,   |              |

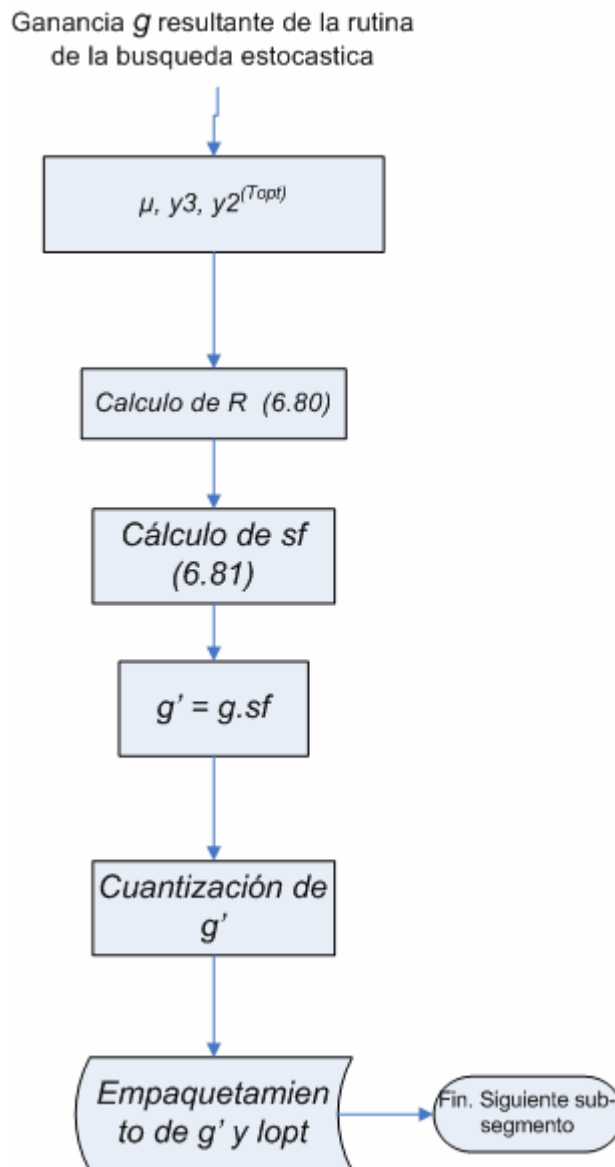


Fig 6.59 Cuantizador diseñado para cuantificar la ganancia estocástica

### 6.10.6 Actualización del Codebook Adaptivo

Una vez obtenidos el periodo pitch  $T_{opt}$ , la ganancia cuantizada  $b$ , el índice óptimo del codebook estocástico  $lopt$  y la ganancia estocástica modificada cuantizada  $g'$ , se actualiza el codebook estocástico mediante las siguientes expresiones:

$$adap\_cb[n] = v^{(lopt)}[n] g'^{(lopt)}_{(cuantizado)} + d 2_0^{(T_{opt})}[n] b^{(T_{opt})}_{(cuantizado)} \quad n = 0 \text{ a } N - 1 \quad (6.91)$$

$$adap\_cb[n] \leftarrow adap\_cb[n + N] \quad n = -T_{max}, -T_{max} + 1, \dots, -1 \quad (6.92)$$

## 6.10.7 Empaquetamiento

## Empaquetamiento de bits

|                           | Coeficientes LSP   | Codebook Adaptivo   | Codebook Estocástico   |
|---------------------------|--|---|--|
| Actualización             | Cada Frame de 30 ms (240 muestras)   | Cada Sub-segmento de 7,5 ms (60 muestras)   | Cada Sub-segmento de 7,5 ms (60 muestras)                              |
| Coeficientes Involucrados | 10 Coeficientes <b>LSP</b>   | 1 valor de <b>T</b> y un valor de ganancia adaptiva <b>b</b>                          | 1 valor de índice <b>I</b> y un valor de ganancia estocastica <b>g</b> |
| Tamaño Codebooks          | -  | 256 x 60  | 512 x 60   |
| Tipo de Análisis          | Lazo Abierto, ventana de Hamming, expansión de ancho de banda, metodo de correlacion, no pre-emfasis | Lazo Cerrado, Búsqueda Entera, modificación del criterio MSPE, búsqueda fraccionaria. | Lazo cerrado, MSPE VQ, , shift por 2, 77% de esparcidad en el codebook |
| Asignación de Bits        | 34 bits [3 4 4 4 4 3 3 3 3 3]  | index <b>T</b> [ 8 6 8 6]<br>ganancia <b>b</b> : 5 (por 4)                            | index <b>I</b> : 9 (por 4)<br>ganancia <b>g</b> : 6 (por 4)            |
| Total de Bits por Frame   | 34   | 48  | 60   |
| Tasa de Bits              | 1133.3 bps   | 1600 bps  | 2000 bps   |

Tabla 6.1 Resumen de Distribución de los bits del codificador.

Como se describió en secciones anteriores, la presente implementación usa 6 bits para la cuantización de la ganancia estocástica, y además debido al esquema FEC (63,57) emplea 6 bits adicionales mas un bit de paridad, lo que nos da un total de 149 bits empleados por trama, lo cual se traduce en un bitrate de 4.96 Kbps. Como se ha visto, este ligero incremento de la tasa de bits es aceptable ya que se gana en calidad de la voz.

|                                       |     |     |                                     |     |     |                           |                           |     |                                       |                                     |     |                        |                        |     |       |     |     |                           |     |     |     |    |    |    |
|---------------------------------------|-----|-----|-------------------------------------|-----|-----|---------------------------|---------------------------|-----|---------------------------------------|-------------------------------------|-----|------------------------|------------------------|-----|-------|-----|-----|---------------------------|-----|-----|-----|----|----|----|
| 1                                     | 2   | 3   | 4                                   | 5   | 6   | 7                         | 8                         | 9   | 10                                    | 11                                  | 12  | 13                     | 14                     | 15  | 16    | 17  | 18  | 19                        | 20  | 21  | 22  |    |    |    |
| lsp 1                                 |     |     | lsp 2                               |     |     | lsp 3                     |                           |     | lsp 4                                 |                                     |     | lsp 5                  |                        |     | lsp 6 |     |     |                           |     |     |     |    |    |    |
| 23                                    | 24  | 25  | 26                                  | 27  | 28  | 29                        | 30                        | 31  | 32                                    | 33                                  | 34  | 35                     | 36                     | 37  | 38    | 39  | 40  | 41                        | 42  | 43  | 44  | 45 | 46 | 47 |
| lsp 7                                 |     |     | lsp 8                               |     |     | lsp 9                     |                           |     | lsp 10                                |                                     |     | pitch T sub-segmento 1 |                        |     |       |     |     | ganancia b sub-segmento 1 |     |     |     |    |    |    |
| 48                                    | 49  | 50  | 51                                  | 52  | 53  | 54                        | 55                        | 56  | 57                                    | 58                                  | 59  | 60                     | 61                     | 62  | 63    | 64  | 65  | 66                        | 67  | 68  |     |    |    |    |
| indice estocastico I sub-segmento 1   |     |     |                                     |     |     |                           |                           |     | ganancia estocastica g sub-segmento 1 |                                     |     |                        | pitch T sub-segmento 2 |     |       |     |     |                           |     |     |     |    |    |    |
| 69                                    | 70  | 71  | 72                                  | 73  | 74  | 75                        | 76                        | 77  | 78                                    | 79                                  | 80  | 81                     | 82                     | 83  | 84    | 85  | 86  | 87                        | 88  |     |     |    |    |    |
| ganancia b sub-segmento 2             |     |     | indice estocastico I sub-segmento 2 |     |     |                           |                           |     | ganancia estocastica g sub-segmento 2 |                                     |     |                        |                        |     |       |     |     |                           |     |     |     |    |    |    |
| 89                                    | 90  | 91  | 92                                  | 93  | 94  | 95                        | 96                        | 97  | 98                                    | 99                                  | 100 | 101                    | 102                    | 103 | 104   | 105 | 106 | 107                       | 108 | 109 | 110 |    |    |    |
| pitch T sub-segmento 3                |     |     |                                     |     |     | ganancia b sub-segmento 3 |                           |     |                                       | indice estocastico I sub-segmento 3 |     |                        |                        |     |       |     |     |                           |     |     |     |    |    |    |
| 111                                   | 112 | 113 | 114                                 | 115 | 116 | 117                       | 118                       | 119 | 120                                   | 121                                 | 122 | 123                    | 124                    | 125 | 126   | 127 |     |                           |     |     |     |    |    |    |
| ganancia estocastica g sub-segmento 3 |     |     | pitch T sub-segmento 4              |     |     |                           | ganancia b sub-segmento 4 |     |                                       |                                     |     |                        |                        |     |       |     |     |                           |     |     |     |    |    |    |
| 128                                   | 129 | 130 | 131                                 | 132 | 133 | 134                       | 135                       | 136 | 137                                   | 138                                 | 139 | 140                    | 141                    | 142 | 143   | 144 | 145 | 146                       | 147 | 148 | 149 |    |    |    |
| indice estocastico I sub-segmento 4   |     |     |                                     |     |     |                           |                           |     | ganancia estocastica g sub-segmento 4 |                                     |     |                        | paridad                | FEC |       |     |     |                           |     |     |     |    |    |    |

Tabla 6.2 Distribución de los bits en el stream-bit de salida del codificador.

### 6.11 Resultados de las etapas de búsquedas adaptiva y estocástica

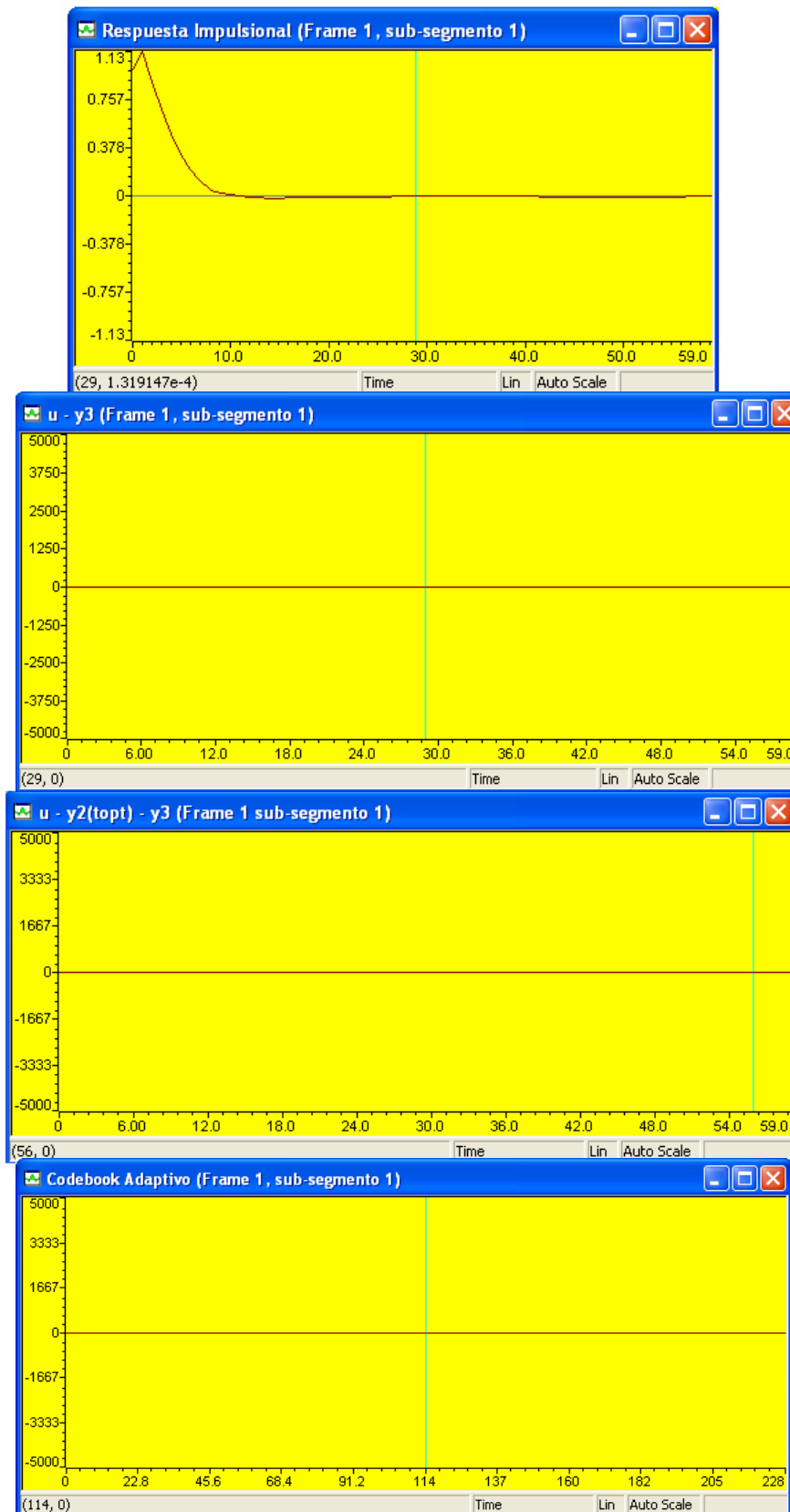


Fig. 6.60 Frame 1 Sub-segmento 1. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.



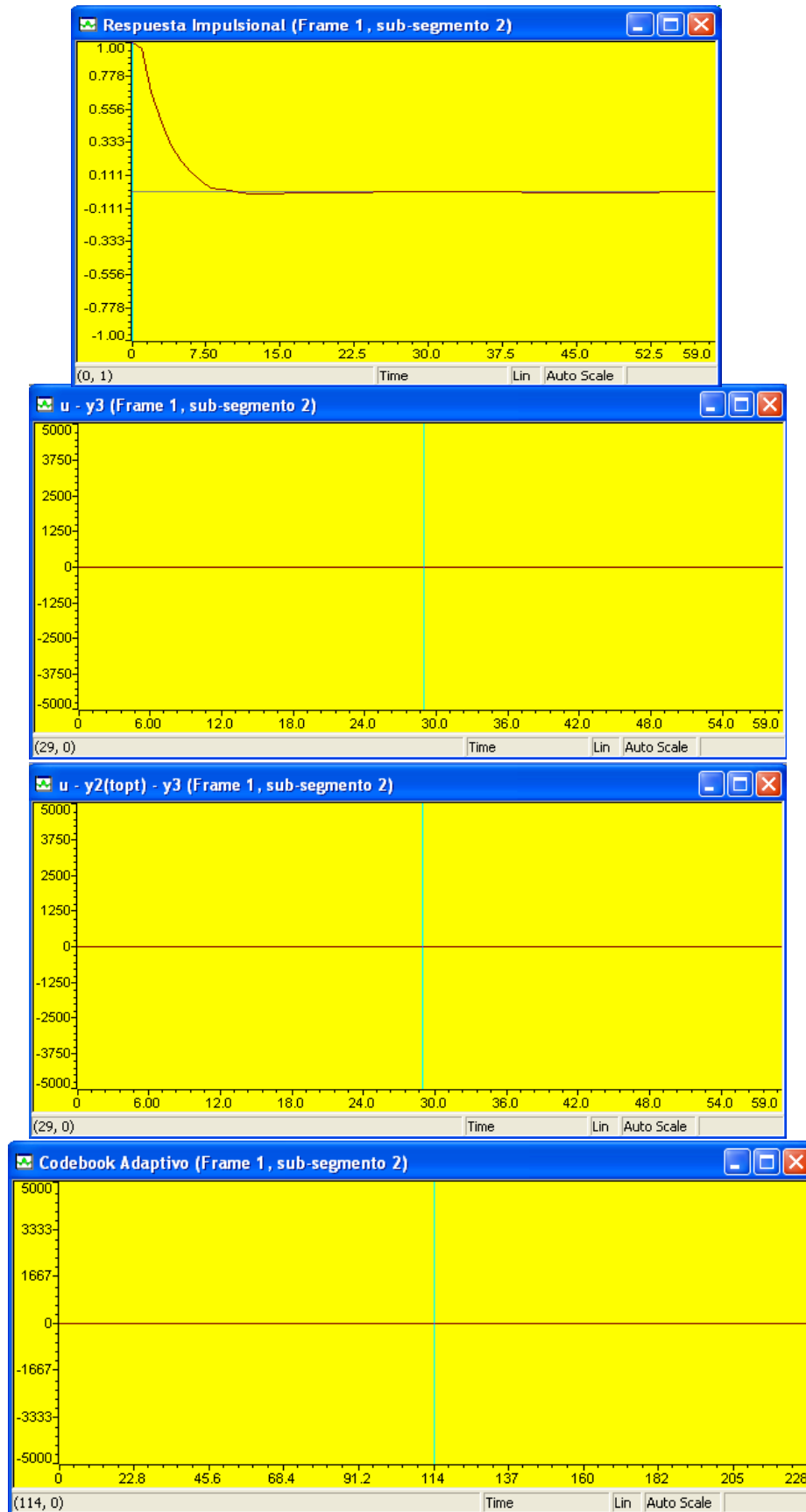


Fig. 6.61 Frame 1 sub-segmento 2. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

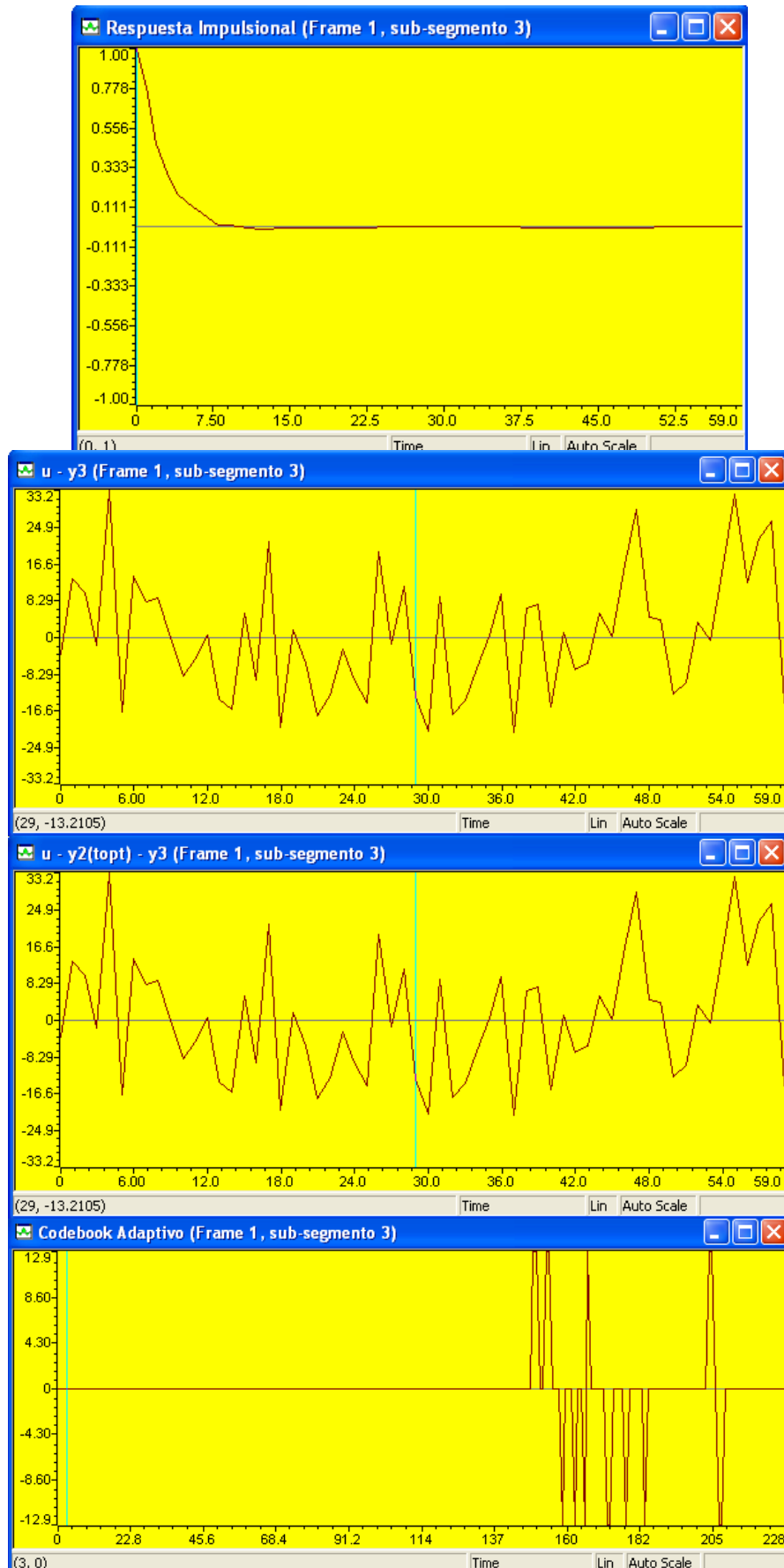


Fig. 6.62 Frame 1 sub-segmento 3. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

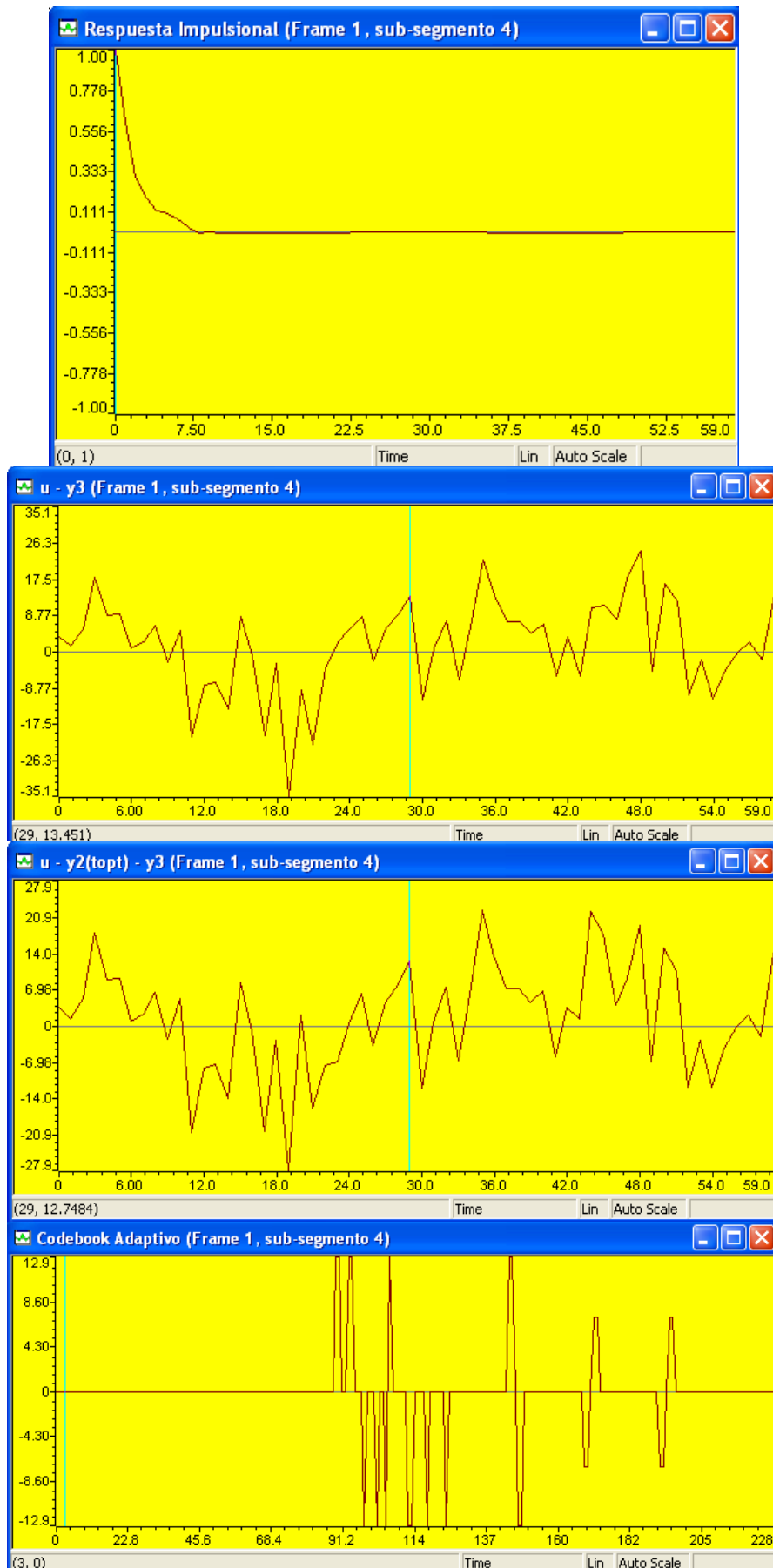


Fig. 6.63 Frame 1 sub-segmento 4. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

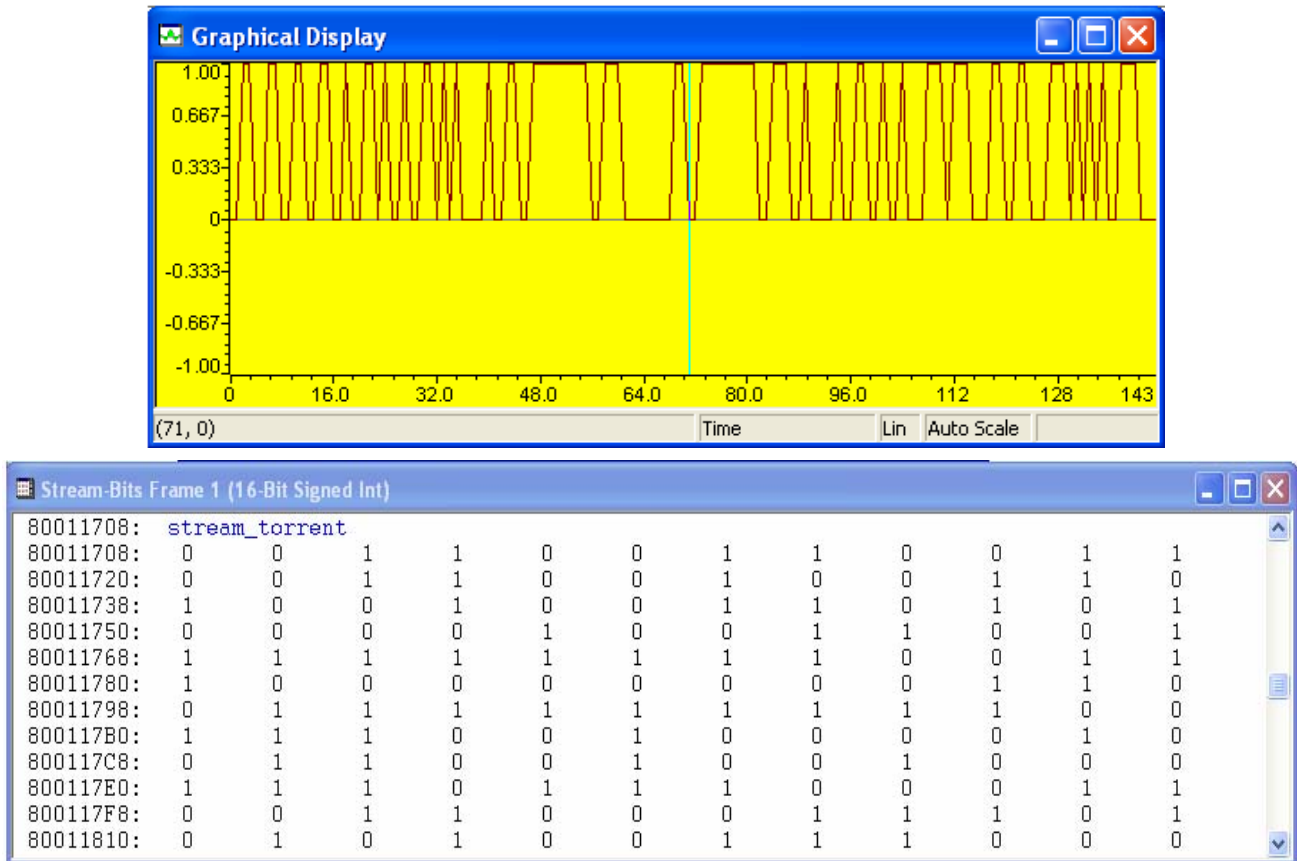


Fig. 6.64 Stream bits para el frame 1, arriba: stream-bits de salida, abajo: valores.

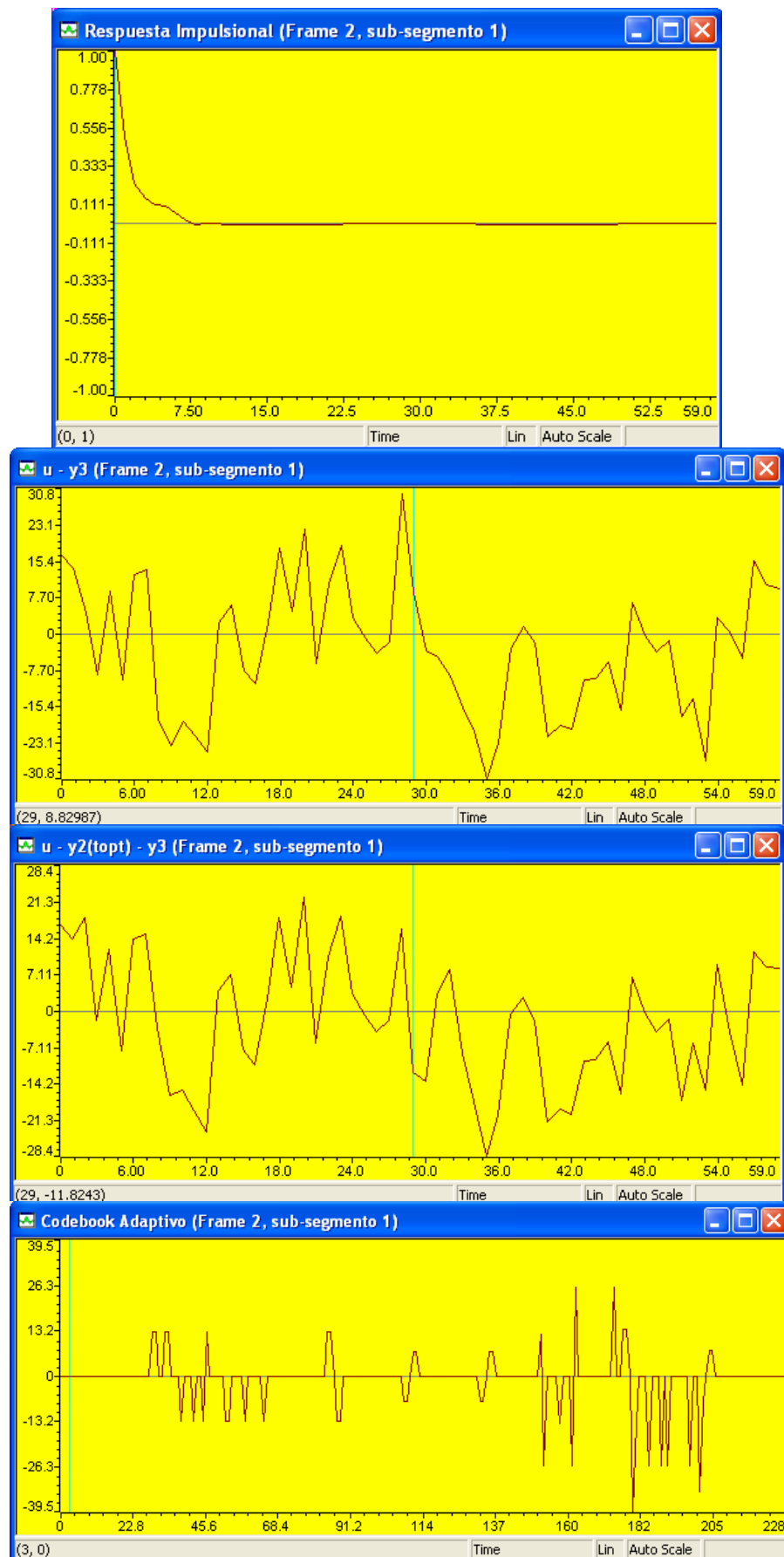


Fig. 6.65 Frame 2 sub-segmento 1. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

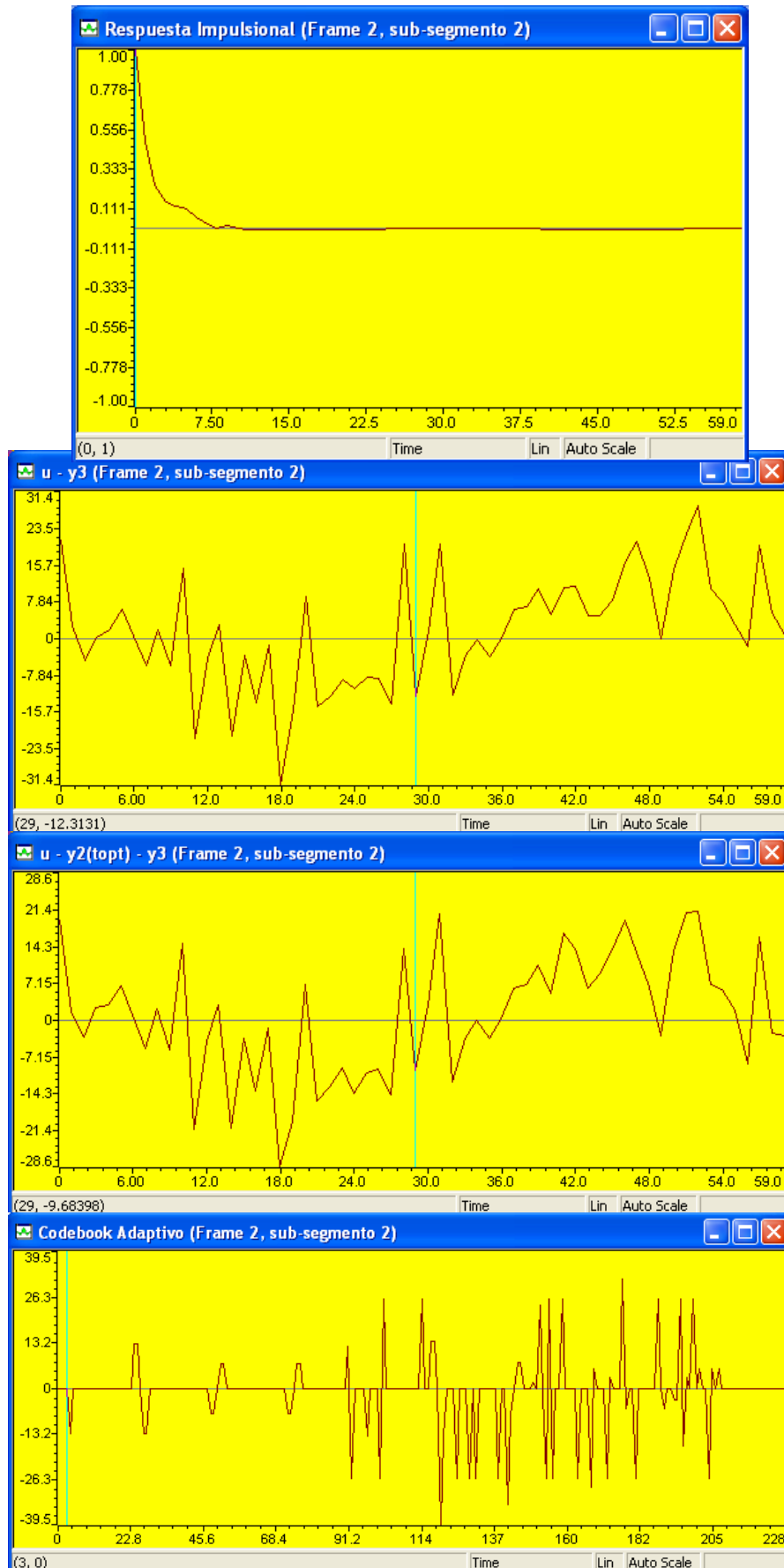


Fig. 6.66 Frame 2 sub-segmento 2. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

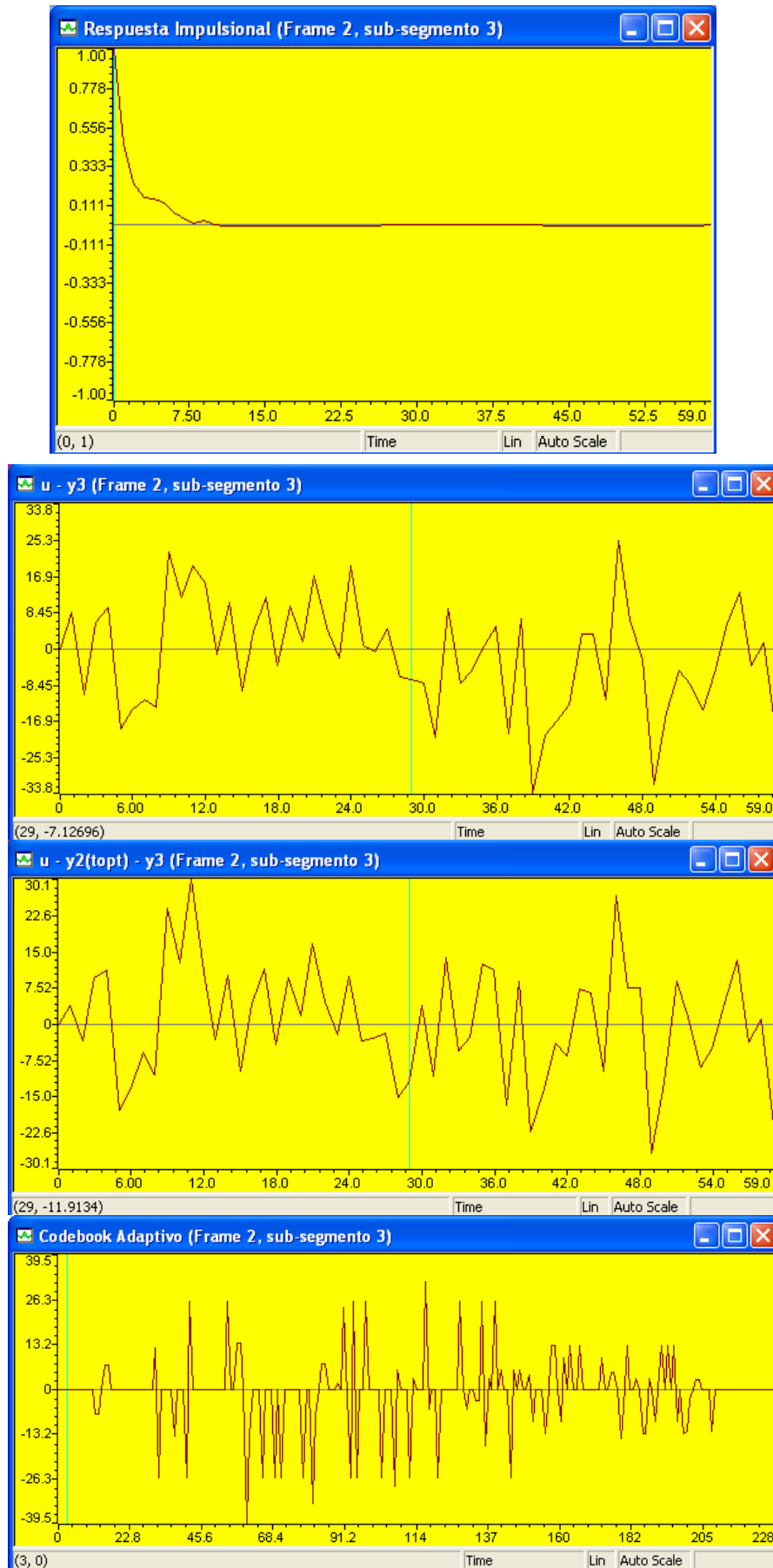


Fig. 6.67 Frame 2 sub-segmento 3. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

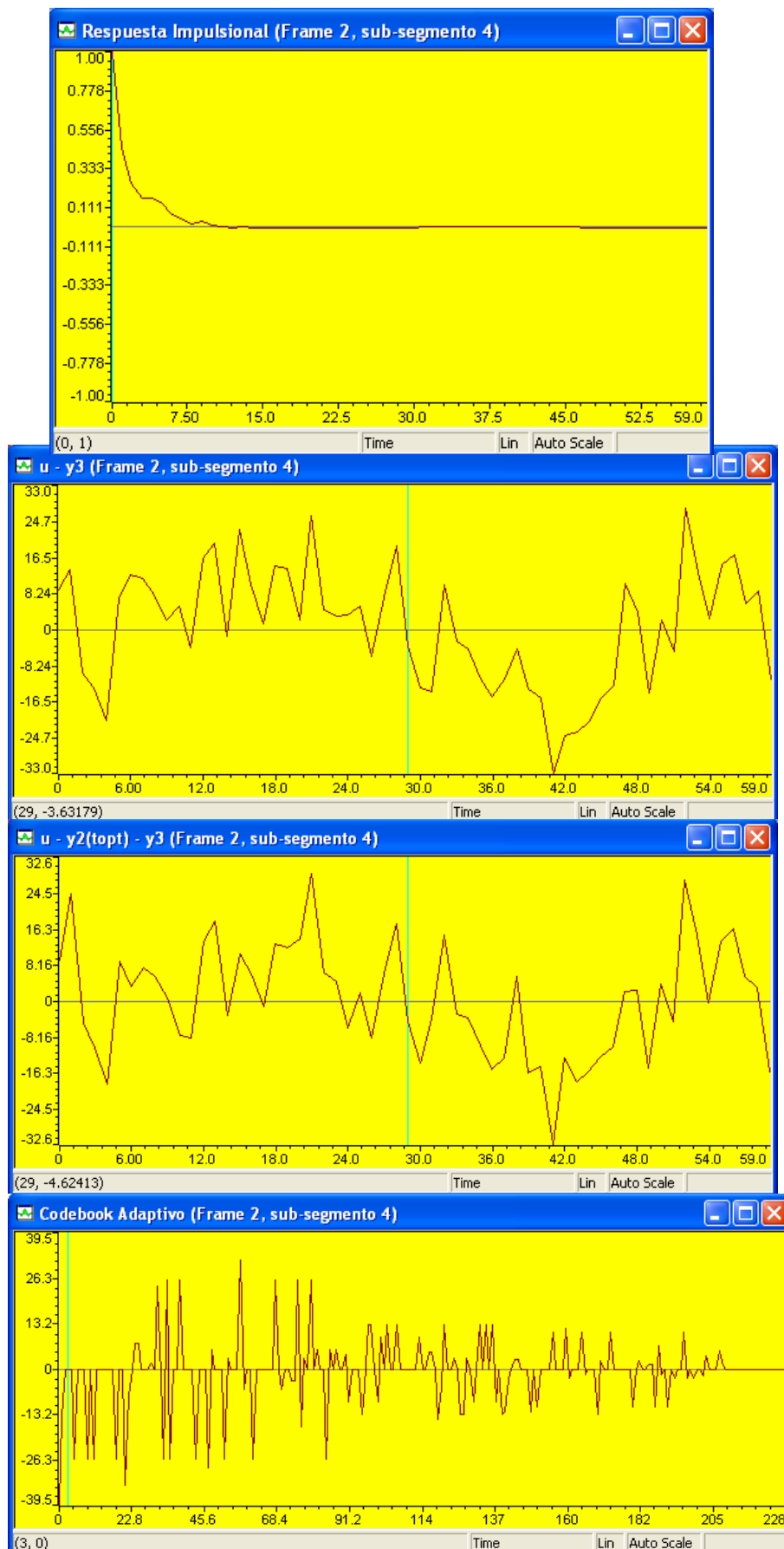


Fig. 6.68 Frame 2 sub-segmento 4. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.



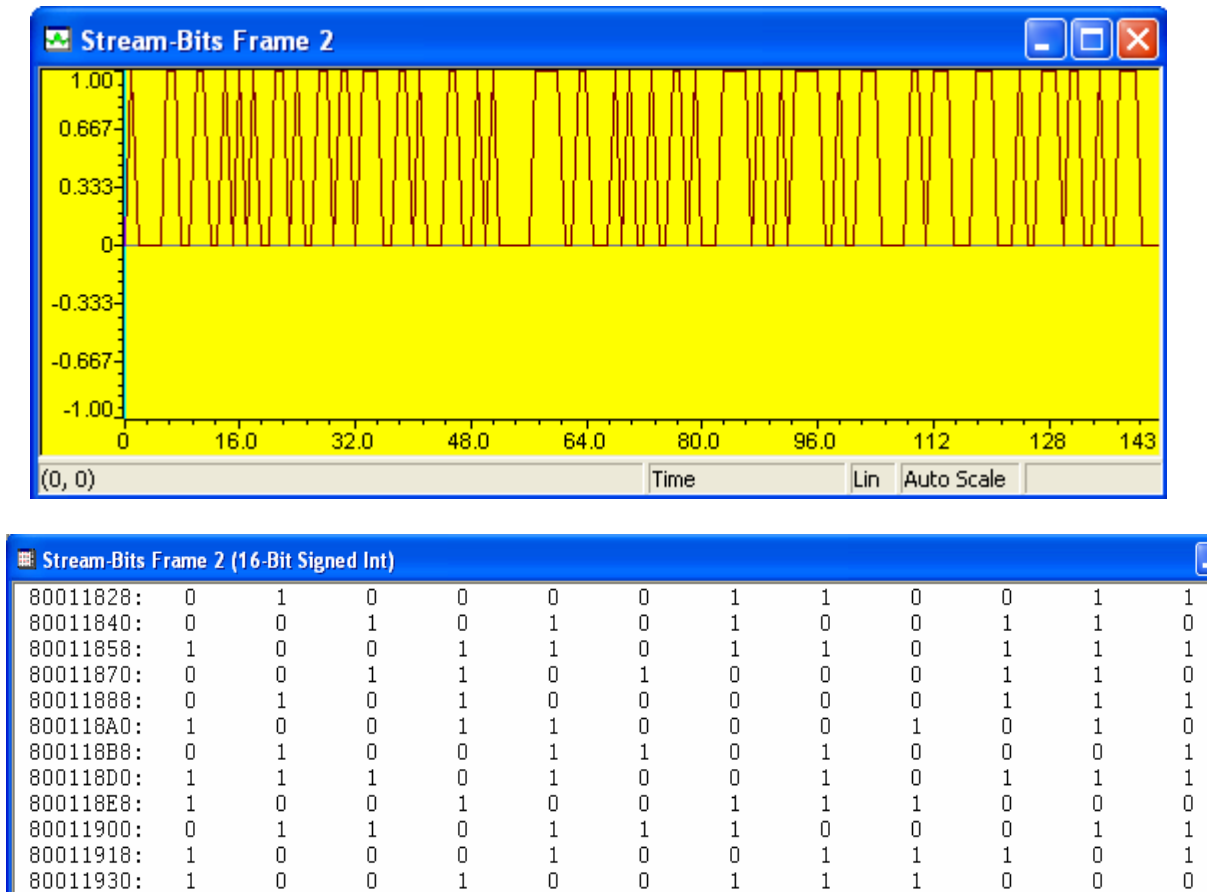


Fig. 6.69 Stream bits para el frame 2, arriba: stream-bits de salida, abajo: valores.

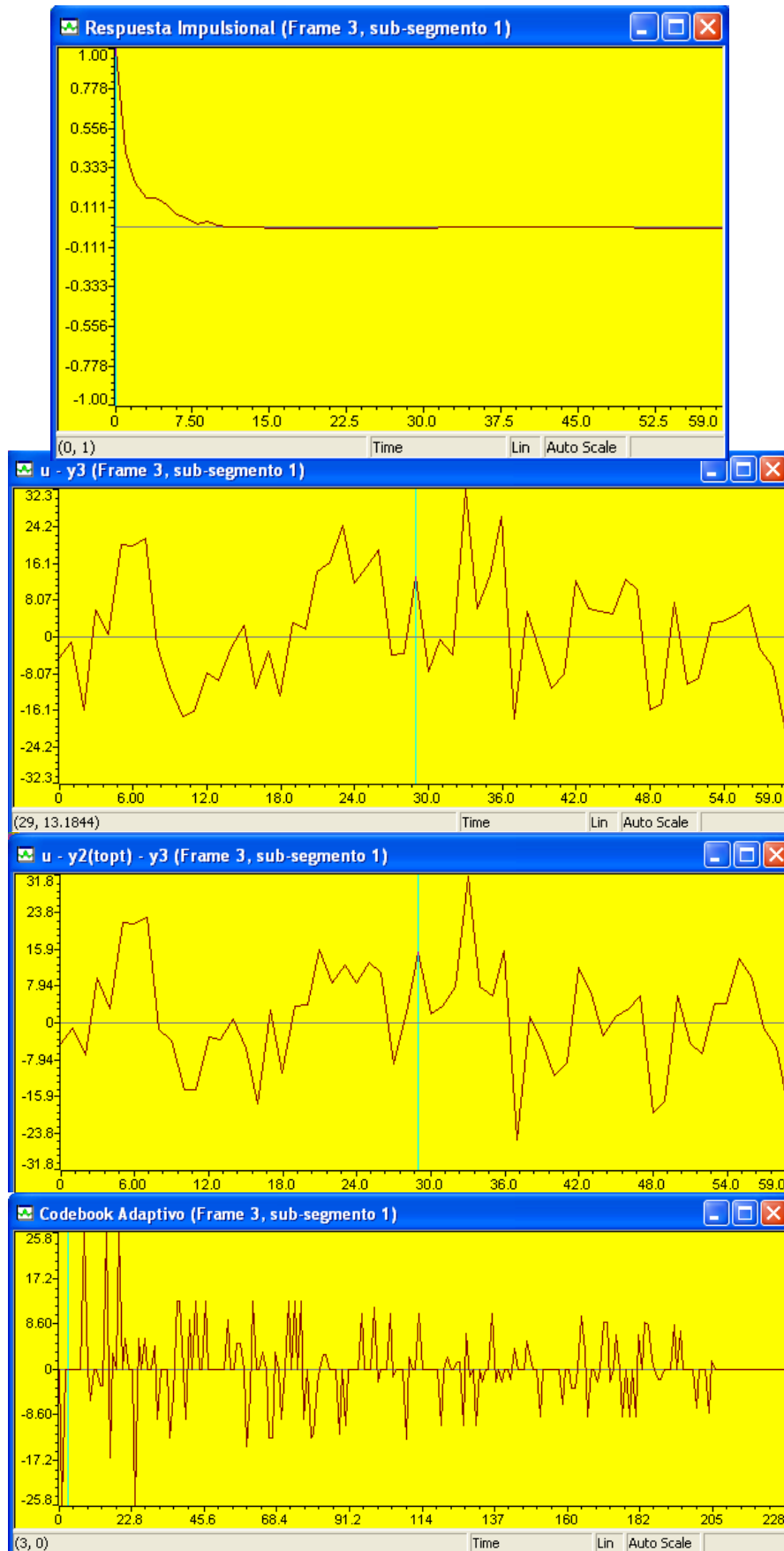


Fig. 6.70 Frame 3 sub-segmento 1. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

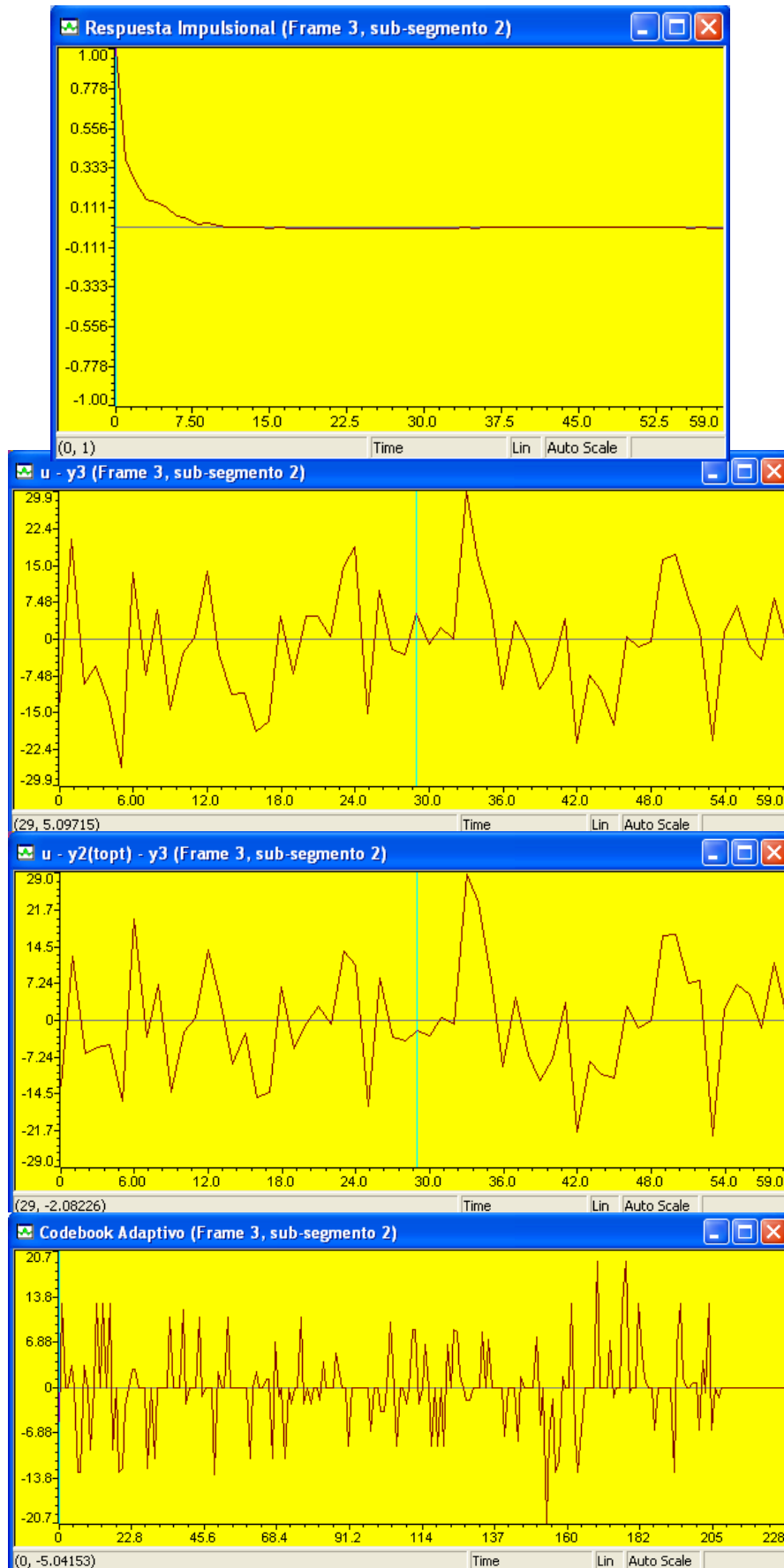


Fig. 6.71 Frame 3 sub-segmento 2. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

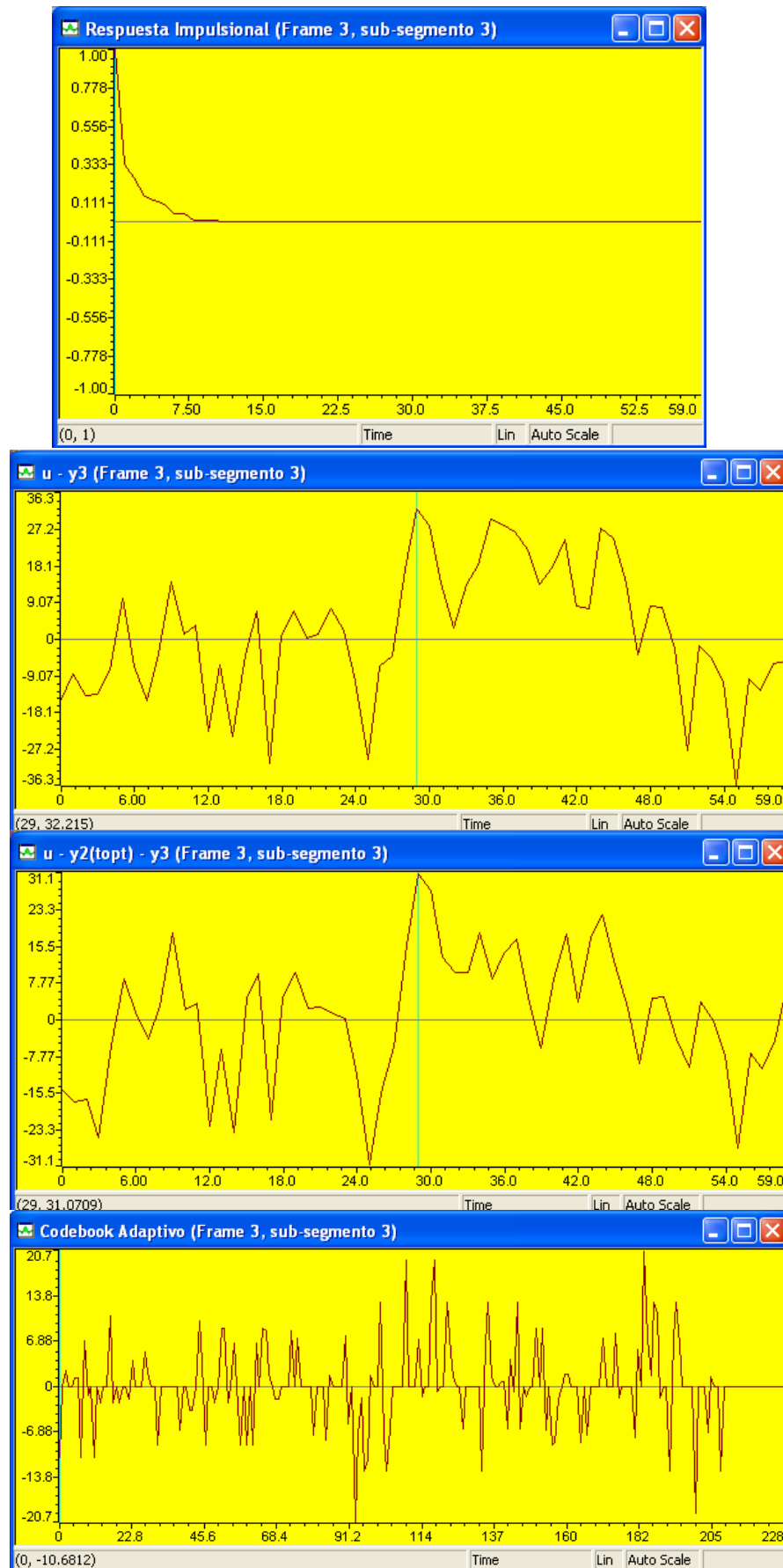


Fig. 6.72 Frame 3 sub-segmento 3. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

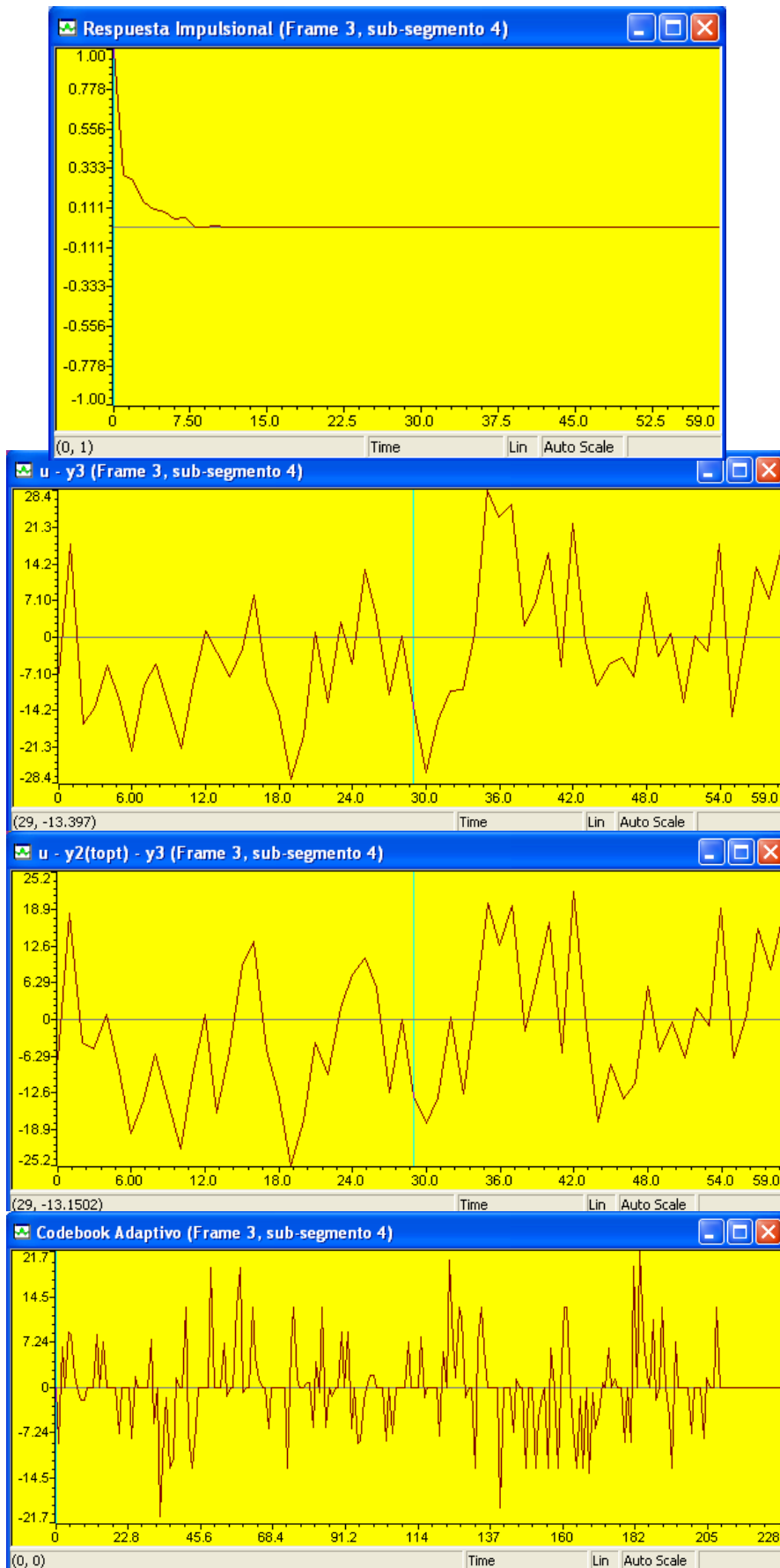
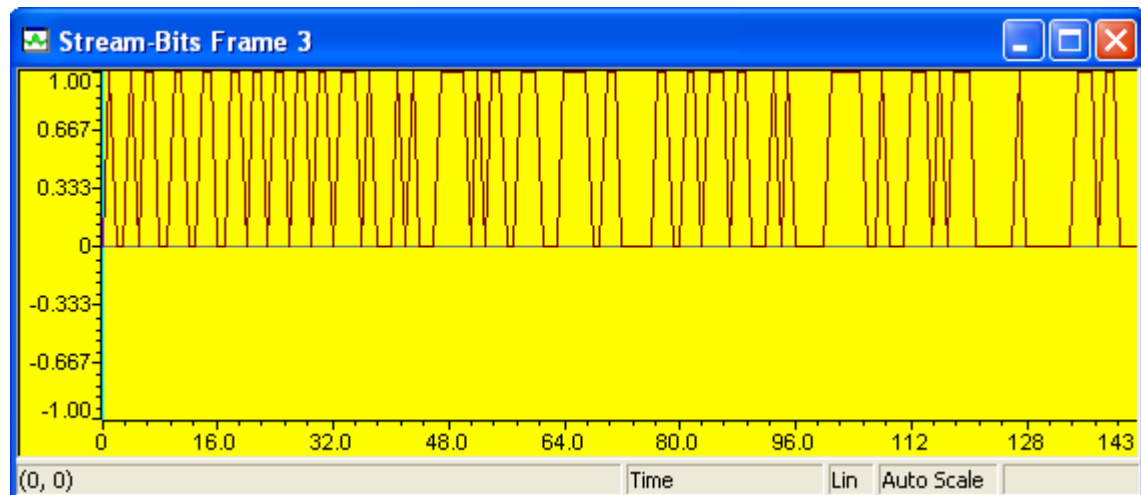


Fig. 6.73 Frame 3 sub-segmento 4. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.



| Stream-Bits Frame 3 (16-Bit Signed Int) |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 80011828:                               | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 80011840:                               | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 80011858:                               | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 80011870:                               | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 80011888:                               | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 800118A0:                               | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 800118B8:                               | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 800118D0:                               | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 800118E8:                               | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 80011900:                               | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 80011918:                               | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 80011930:                               | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Fig. 6.74 Stream bits para el frame 3, arriba: stream-bits de salida, abajo: valores.

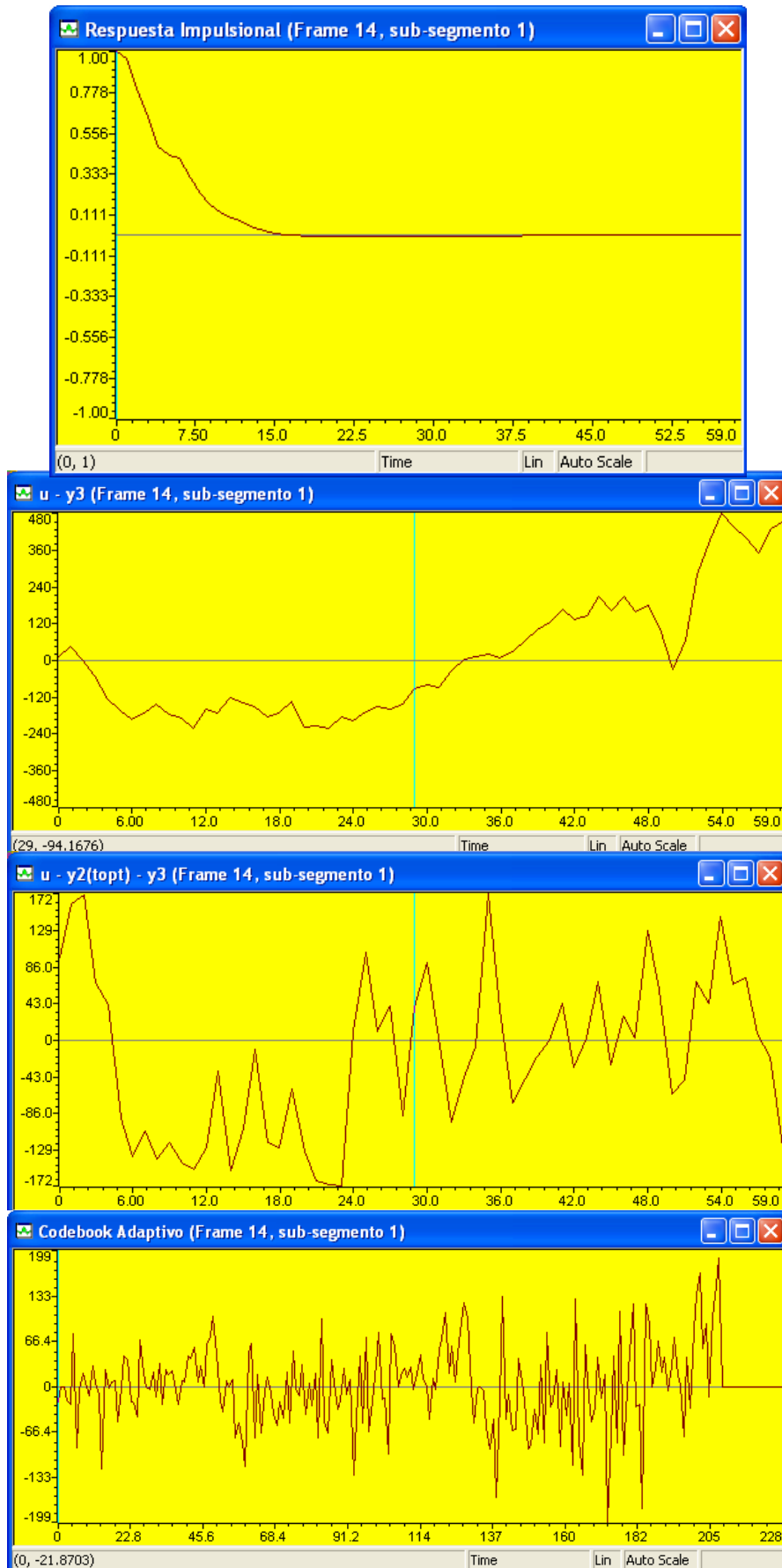


Fig. 6.75 Frame 14 sub-segmento 1. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

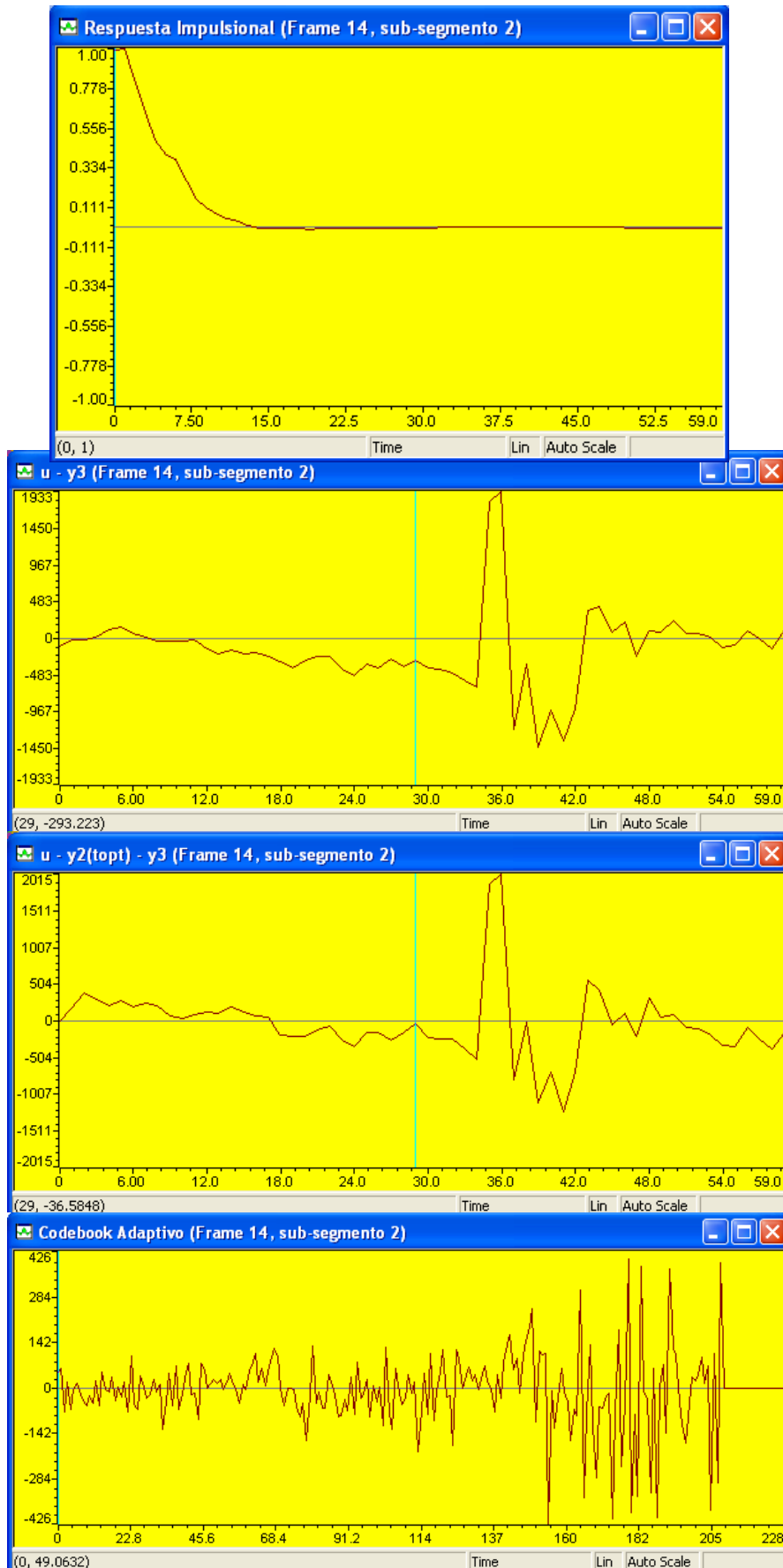


Fig. 6.76 Frame 14 sub-segmento 2. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.



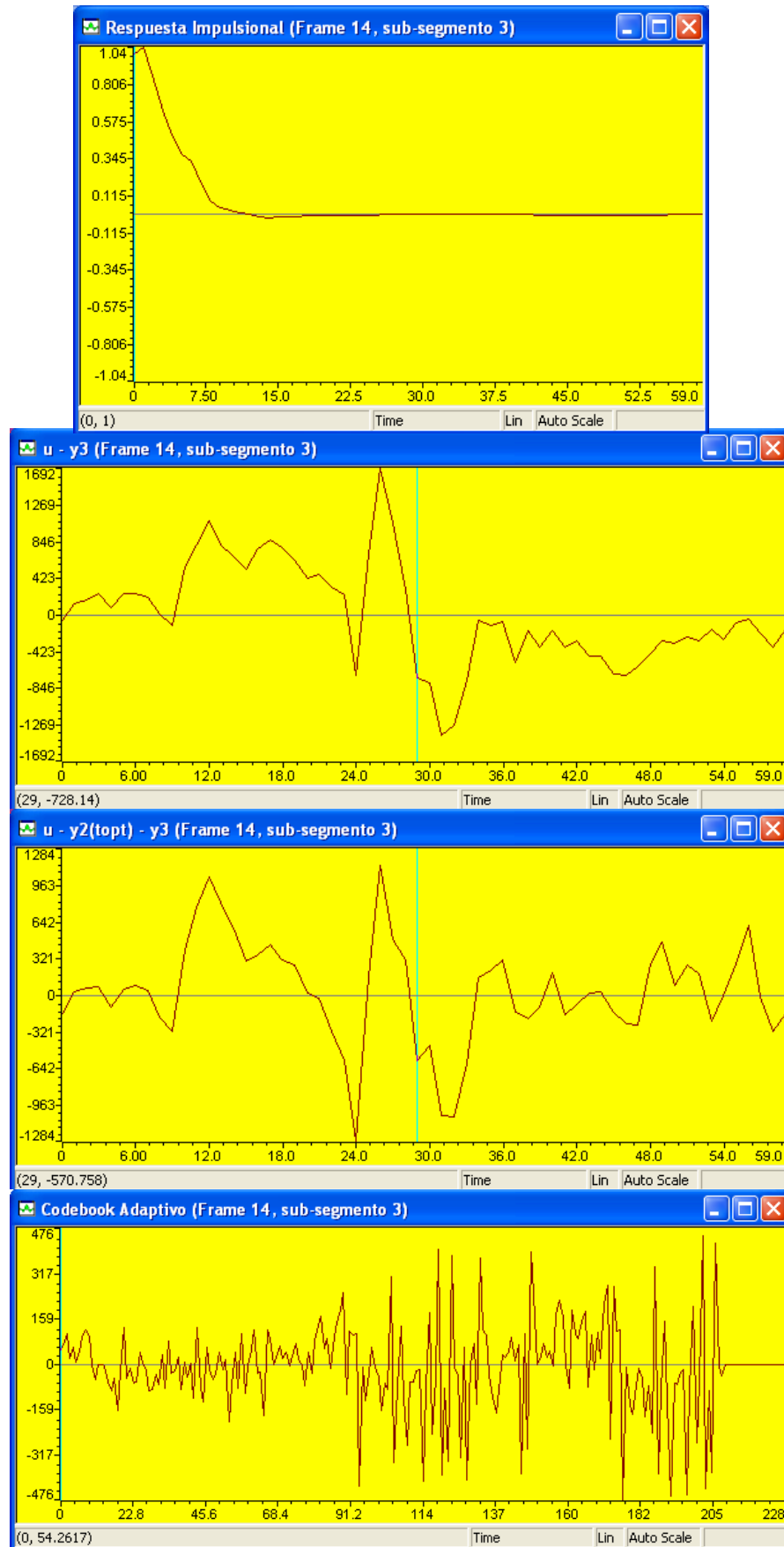


Fig. 6.77 Frame 14 sub-segmento 3. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

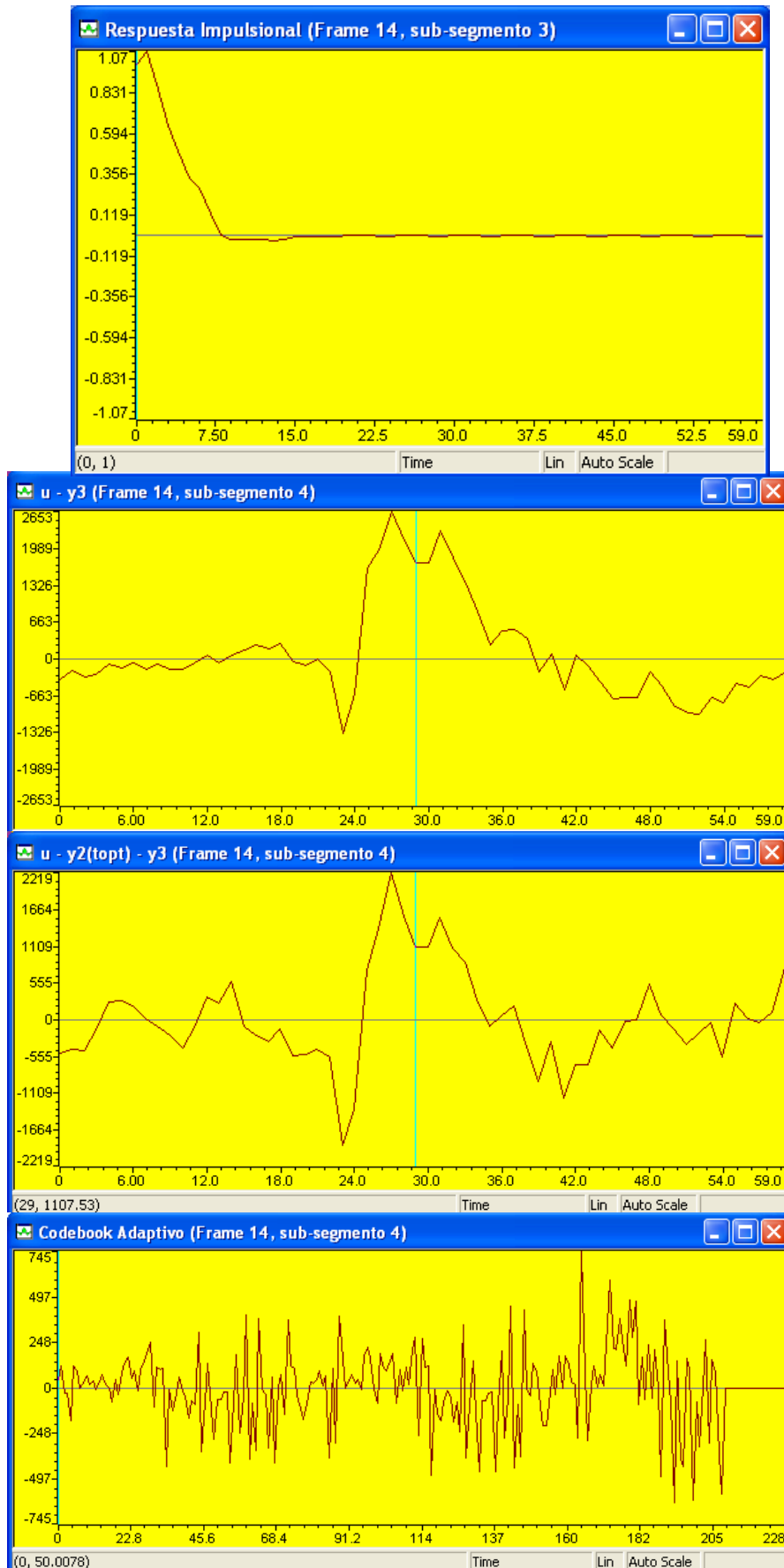
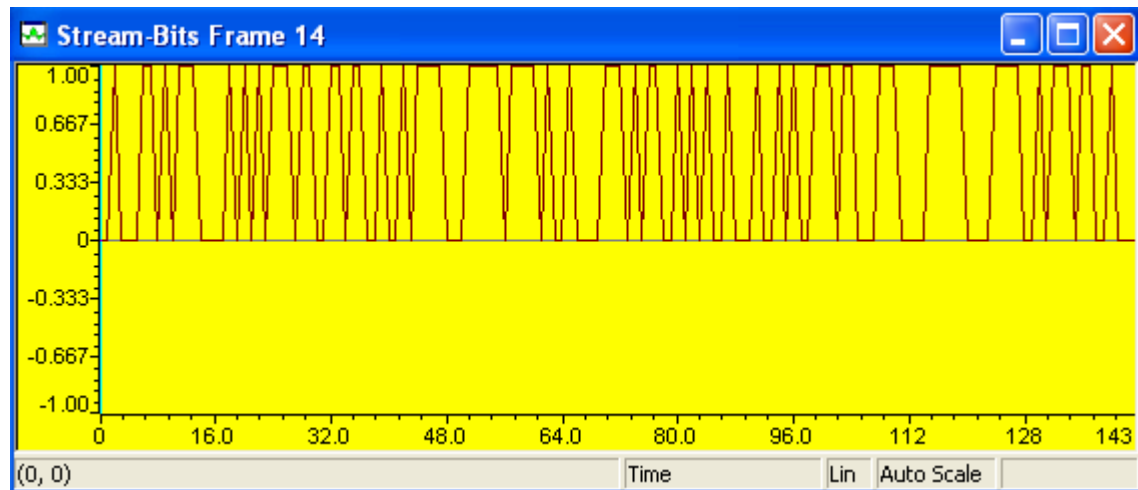


Fig. 6.78 Frame 14 sub-segmento 4. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.



| Stream-Bits Frame 14 (16-Bit Signed Int) |   |   |   |   |   |   |   |   |   |   |   |   |
|--|---|---|---|---|---|---|---|---|---|---|---|---|
| 800125A8:                                | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 800125C0:                                | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 800125D8:                                | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 800125F0:                                | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 80012608:                                | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 80012620:                                | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 80012638:                                | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 80012650:                                | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 80012668:                                | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 80012680:                                | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 80012698:                                | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 800126B0:                                | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

Fig. 6.79 Stream bits para el frame 14, arriba: stream-bits de salida, abajo: valores.

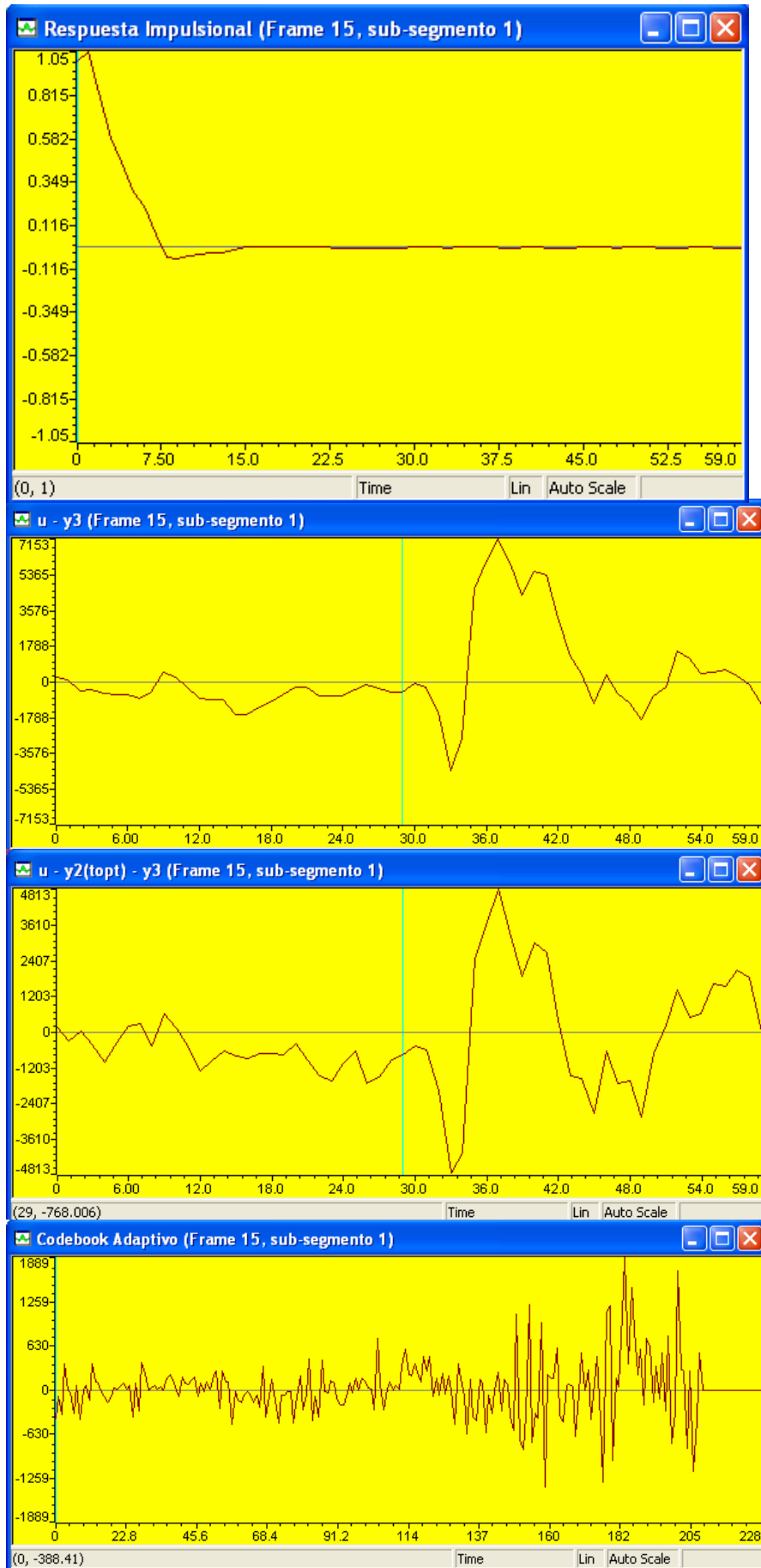


Fig. 6.80 Frame 15 sub-segmento 1. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

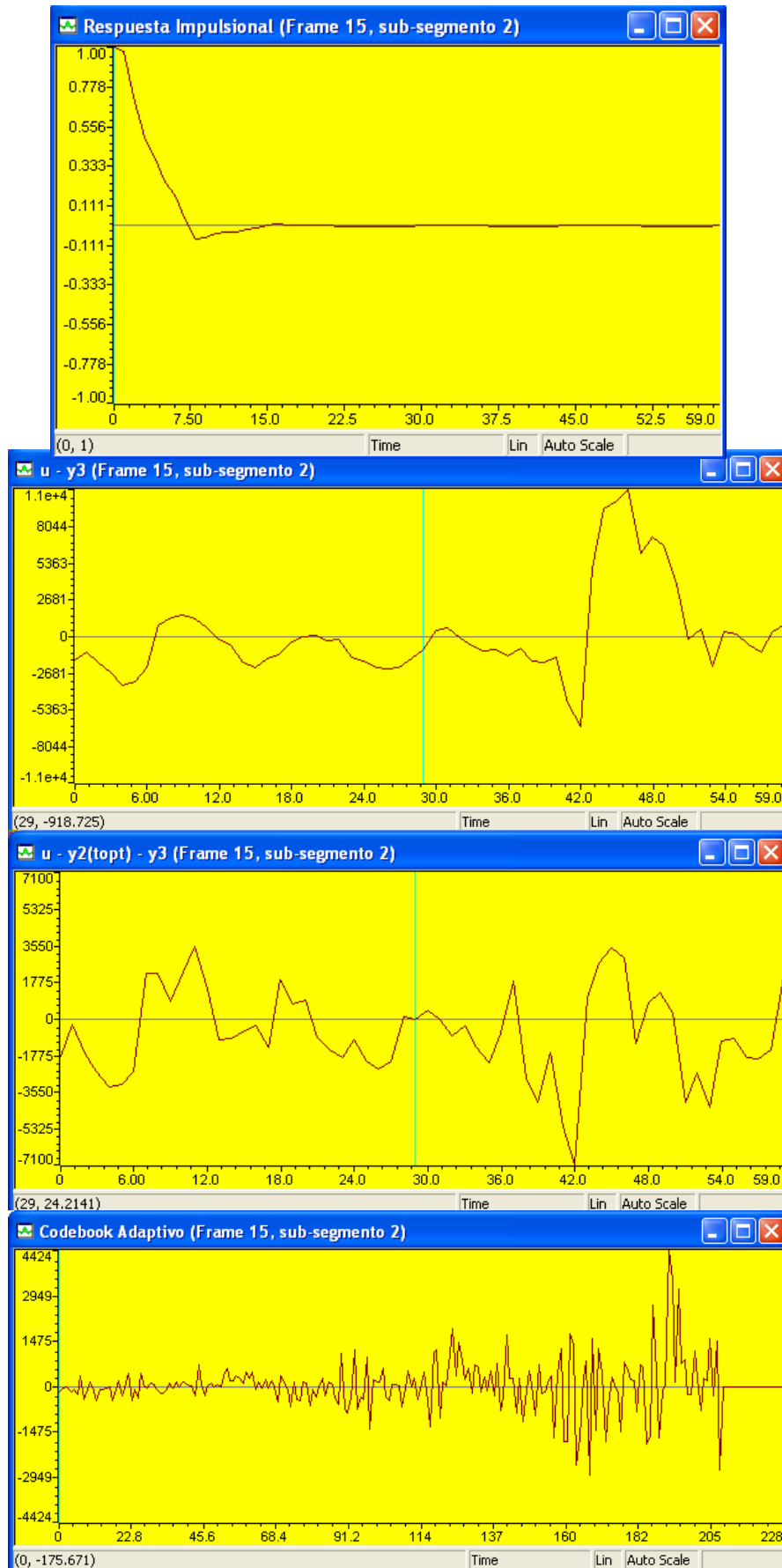


Fig. 6.81 Frame 15 sub-segmento 2. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

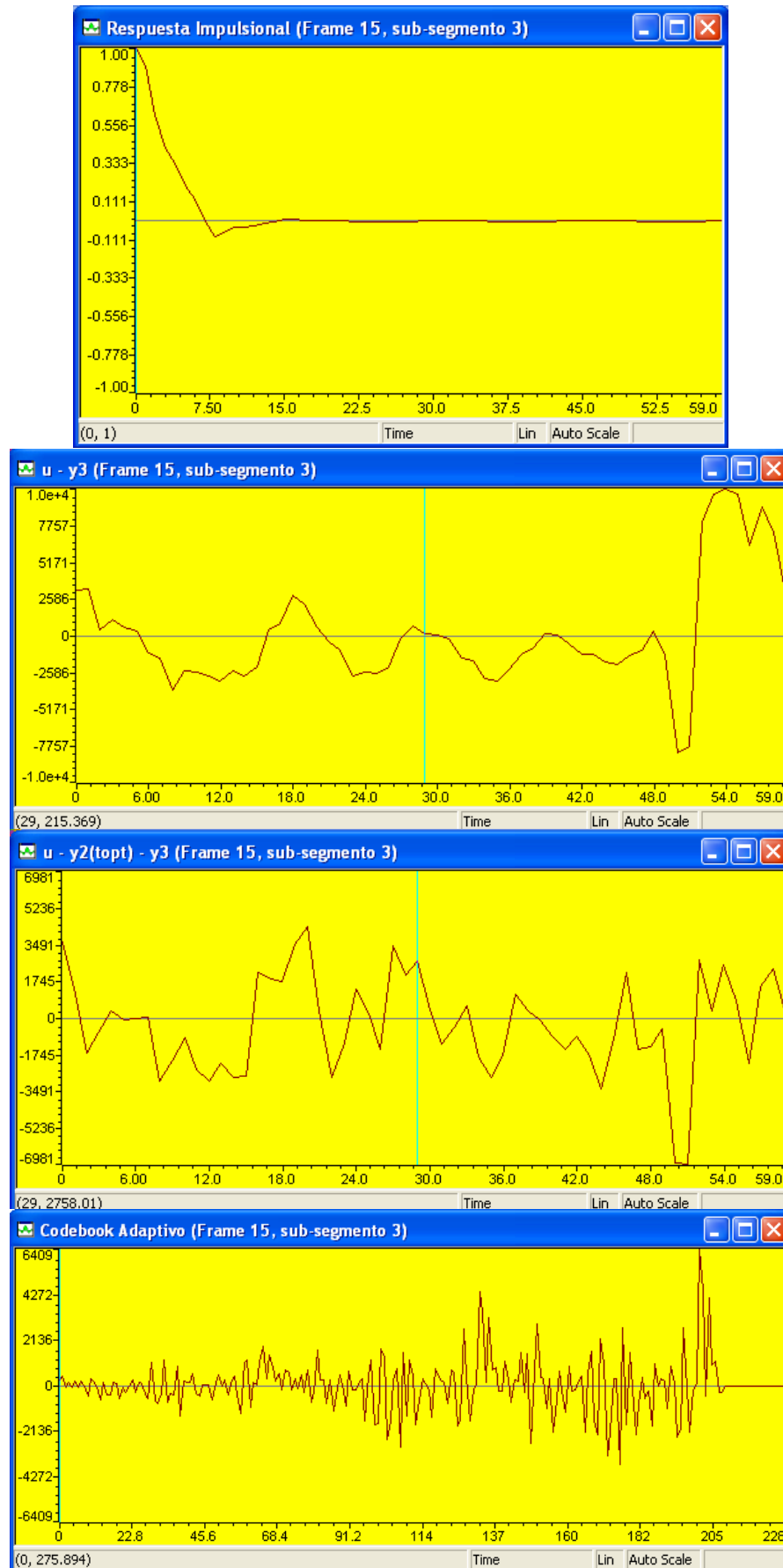


Fig. 6.82 Frame 15 sub-segmento 3. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.

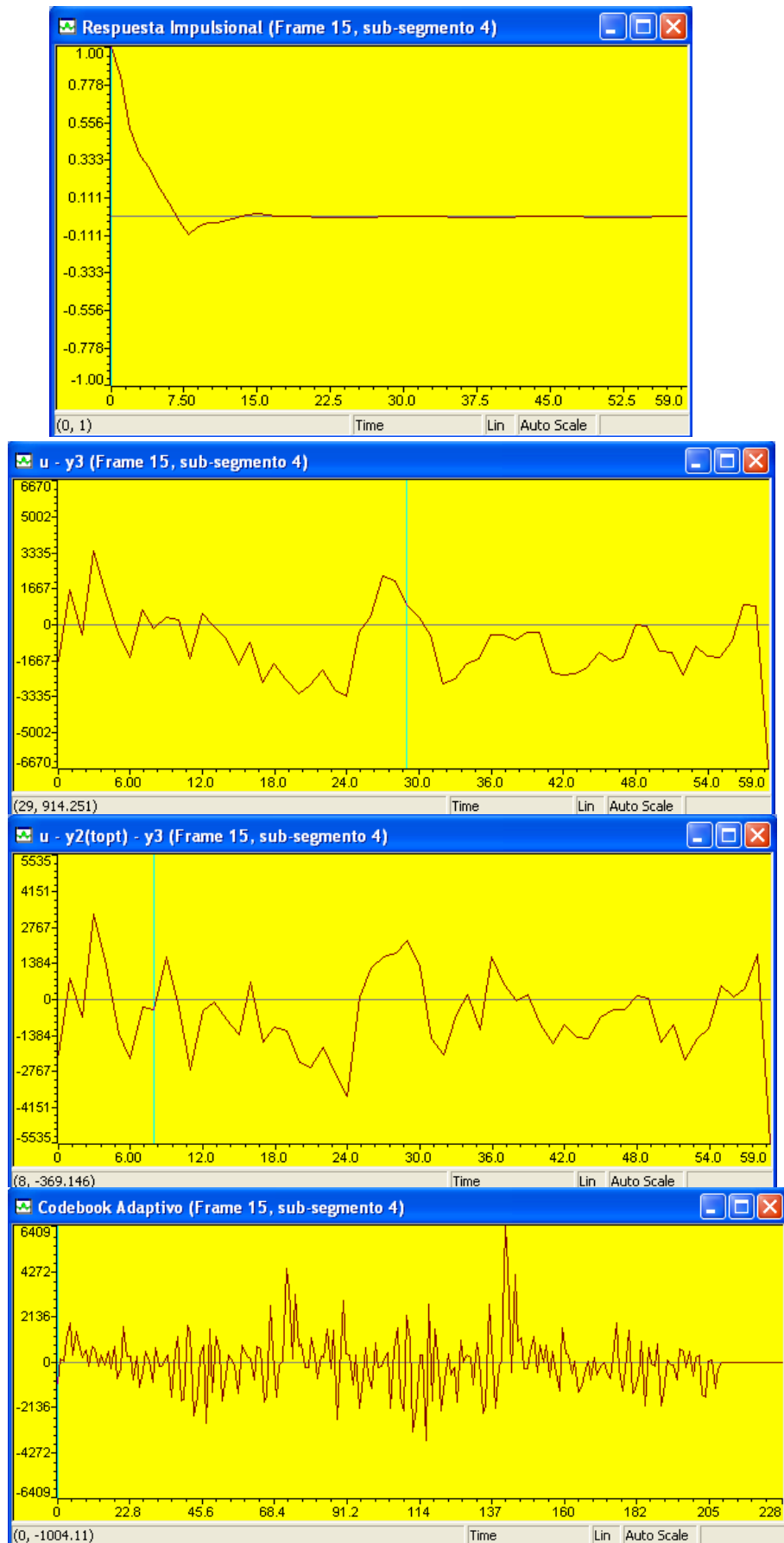
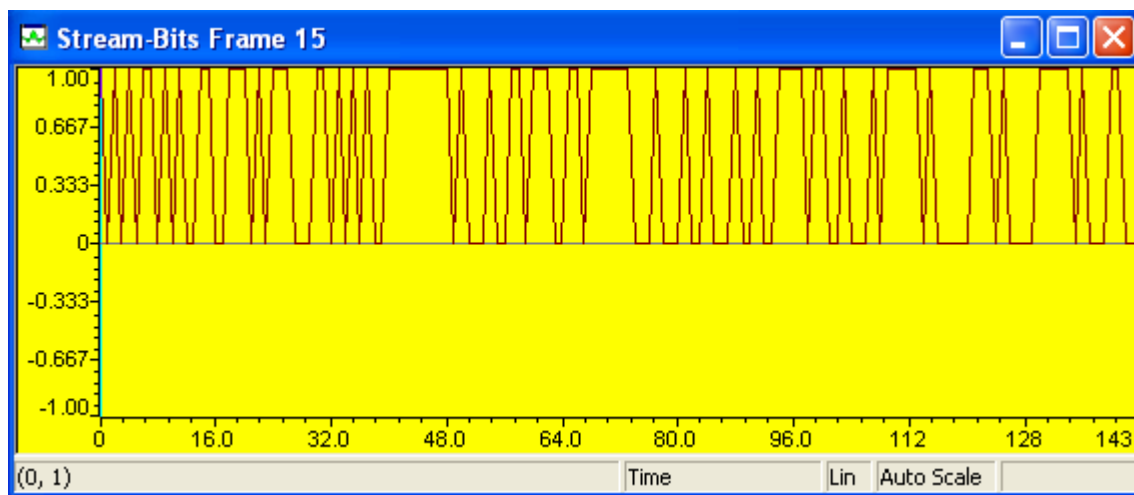


Fig. 6.83 Frame 15 sub-segmento 4. De arriba a abajo: Respuesta impulsional filtro  $W(z)H(z)$ ,  $u - y_3$ ,  $u - y_2(\text{topt}) - y_3$ , Codebook Adaptivo.



| Stream-Bits Frame 15 (16-Bit Signed Int) |   |   |   |   |   |   |   |   |   |   |   |   |
|--|---|---|---|---|---|---|---|---|---|---|---|---|
| 800126C8:                                | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 800126E0:                                | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 800126F8:                                | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 80012710:                                | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 80012728:                                | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 80012740:                                | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 80012758:                                | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 80012770:                                | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 80012788:                                | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 800127A0:                                | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 800127B8:                                | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 800127D0:                                | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

Fig. 6.84 Stream bits para el frame 15, arriba: stream-bits de salida, abajo: valores.

En la Tabla 6.3 se muestran los valores de los parámetros hallados en las búsquedas adaptiva y estocástica para el segmento de análisis mostrado al inicio del capítulo en la Fig. 6.5. El segmento de análisis consta de 8160 muestras que corresponden a 1020 ms.



| <b>Nro. Frame</b> | <b>Nro. Sub-segmento</b> | <b>Periodo Pitch <math>T_{opt}</math></b> | <b>Ganancia Codebook Adaptivo Cuantizado <math>b</math></b> | <b>Índice del Codebook Estocástico <math>l</math></b> | <b>Ganancia Estocástica Cuantizada <math>g</math></b> |
|-------------------|--------------------------|---|---|---|---|
| 1                 | 1                        | 20.000000                                 | 0.000000  | 512   | 0.000000  |
|                   | 2                        | 20.000000                                 | 0.000000  | 512   | 0.000000  |
|                   | 3                        | 20.000000                                 | 0.000000  | 394   | 12.900000   |
|                   | 4                        | 24.000000                                 | -0.555000   | 344   | 0.000000  |
| 2                 | 5                        | 93.000000                                 | 1.062000  | 21  | 25.799999   |
|                   | 6                        | 76.000000                                 | -0.229000   | 90  | 25.799999   |
|                   | 7                        | 119.000000                                | 0.368000  | 271   | 12.900000   |
|                   | 8                        | 112.000000                                | -0.414000   | 312   | 0.000000  |
| 3                 | 9                        | 105.000000                                | -0.693000   | 432   | 0.000000  |
|                   | 10                       | 134.000000                                | 0.601000  | 305   | 12.900000   |
|                   | 11                       | 91.000000                                 | -0.993000   | 160   | 0.000000  |
|                   | 12                       | 89.000000                                 | -0.993000   | 258   | -12.900000  |
| 4                 | 13                       | 113.000000                                | -0.693000   | 455   | 0.000000  |
|                   | 14                       | 139.000000                                | 1.193000  | 30  | -12.900000  |
|                   | 15                       | 52.666672                                 | 0.702000  | 201   | -12.900000  |
|                   | 16                       | 45.000000                                 | -0.414000   | 180   | 0.000000  |
| 5                 | 17                       | 74.666672                                 | 0.780000  | 63  | 0.000000  |
|                   | 18                       | 95.000000                                 | -0.693000   | 502   | 0.000000  |
|                   | 19                       | 30.000000                                 | 1.193000  | 306   | 25.799999   |
|                   | 20                       | 32.000000                                 | -0.229000   | 500   | -12.900000  |
| 6                 | 21                       | 121.000000                                | 0.255000  | 269   | 12.900000   |
|                   | 22                       | 120.000000                                | 0.653000  | 398   | -12.900000  |
|                   | 23                       | 73.333344                                 | -0.693000   | 266   | -12.900000  |
|                   | 24                       | 92.000000                                 | -0.229000   | 312   | 12.900000   |
| 7                 | 25                       | 71.666672                                 | 0.745000  | 302   | 0.000000  |
|                   | 26                       | 63.666672                                 | 0.850000  | 129   | 0.000000  |
|                   | 27                       | 79.333344                                 | 1.193000  | 86  | 0.000000  |
|                   | 28                       | 96.000000                                 | -0.693000   | 294   | 25.799999   |
| 8                 | 29                       | 73.666672                                 | 0.368000  | 391   | 0.000000  |
|                   | 30                       | 88.000000                                 | -0.555000   | 485   | 25.799999   |
|                   | 31                       | 134.000000                                | 0.915000  | 461   | 0.000000  |
|                   | 32                       | 102.000000                                | -0.555000   | 58  | 0.000000  |
| 9                 | 33                       | 107.000000                                | 0.915000  | 493   | 25.799999   |
|                   | 34                       | 116.000000                                | -0.693000   | 440   | 25.799999   |
|                   | 35                       | 106.000000                                | -0.414000   | 189   | 25.799999   |
|                   | 36                       | 105.000000                                | -0.229000   | 59  | -12.900000  |
| 10                | 37                       | 136.000000                                | 0.457000  | 304   | 0.000000  |
|                   | 38                       | 91.000000                                 | -0.693000   | 182   | -12.900000  |
|                   | 39                       | 125.000000                                | 1.394000  | 401   | 25.799999   |
|                   | 40                       | 99.000000                                 | -0.414000   | 161   | 0.000000  |
| 11                | 41                       | 64.333344                                 | 1.289000  | 33  | 12.900000   |
|                   | 42                       | 62.666672                                 | 0.457000  | 407   | 12.900000   |
|                   | 43                       | 120.000000                                | -0.414000   | 76  | 12.900000   |
|                   | 44                       | 147.000000                                | 0.601000  | 400   | 0.000000  |
| 12                | 45                       | 93.000000                                 | -0.993000   | 319   | 48.856968   |
|                   | 46                       | 99.000000                                 | -0.993000   | 226   | 95.525818   |
|                   | 47                       | 85.000000                                 | 0.702000  | 186   | -25.799999  |
|                   | 48                       | 75.000000                                 | 0.881000  | 308   | 25.799999   |
| 13                | 49                       | 79.000000                                 | 0.653000  | 163   | -25.799999  |

|    |     |            |           |     |             |
|----|-----|------------|-----------|-----|-------------|
|    | 50  | 81.000000  | 1.193000  | 48  | -25.799999  |
|    | 51  | 78.333344  | 1.289000  | 360 | -12.900000  |
|    | 52  | 79.333344  | 1.289000  | 338 | -25.799999  |
| 14 | 53  | 80.000000  | 1.540000  | 498 | 25.799999   |
|    | 54  | 72.666672  | 1.394000  | 155 | 345.255432  |
|    | 55  | 83.000000  | 1.117000  | 398 | 143.693924  |
|    | 56  | 75.666672  | 1.394000  | 234 | -343.381134 |
| 15 | 57  | 70.666672  | 1.991000  | 140 | 987.084412  |
|    | 58  | 68.666672  | 1.991000  | 274 | -906.866638 |
|    | 59  | 69.000000  | 1.289000  | 325 | 706.454346  |
|    | 60  | 69.333344  | 0.531000  | 249 | 752.151123  |
| 16 | 61  | 69.333344  | 1.289000  | 114 | -723.687866 |
|    | 62  | 68.000000  | 1.062000  | 346 | 449.107574  |
|    | 63  | 67.000000  | 0.948000  | 350 | 449.107574  |
|    | 64  | 65.666672  | 0.915000  | 67  | 816.873840  |
| 17 | 65  | 65.000000  | 0.816000  | 238 | -451.064026 |
|    | 66  | 63.666672  | 0.780000  | 40  | -723.687866 |
|    | 67  | 62.666672  | 0.850000  | 207 | -343.381134 |
|    | 68  | 61.333340  | 0.850000  | 506 | 188.429306  |
| 18 | 69  | 60.666672  | 0.780000  | 245 | 188.429306  |
|    | 70  | 60.000000  | 0.745000  | 192 | 449.107574  |
|    | 71  | 60.666672  | 0.457000  | 431 | -247.404648 |
|    | 72  | 55.000000  | 0.255000  | 498 | 95.525818   |
| 19 | 73  | 60.333340  | 0.255000  | 171 | -25.799999  |
|    | 74  | 62.333340  | 0.457000  | 451 | 25.799999   |
|    | 75  | 53.000000  | -0.414000 | 27  | 12.900000   |
|    | 76  | 62.333340  | 0.653000  | 106 | 12.900000   |
| 20 | 77  | 113.000000 | -0.555000 | 404 | 0.000000    |
|    | 78  | 132.000000 | -0.993000 | 266 | -12.900000  |
|    | 79  | 73.333344  | 1.193000  | 348 | -12.900000  |
|    | 80  | 68.666672  | 1.062000  | 496 | -12.900000  |
| 21 | 81  | 45.000000  | 1.991000  | 102 | 1308.075806 |
|    | 82  | 39.666672  | -0.414000 | 472 | 816.873840  |
|    | 83  | 75.666672  | -0.229000 | 157 | -343.381134 |
|    | 84  | 88.000000  | 0.255000  | 370 | -156.644424 |
| 22 | 85  | 45.000000  | -0.993000 | 230 | -156.644424 |
|    | 86  | 52.666672  | 1.991000  | 397 | 752.151123  |
|    | 87  | 59.666672  | 1.289000  | 391 | -676.375244 |
|    | 88  | 63.000000  | 1.062000  | 78  | -664.991516 |
| 23 | 89  | 63.666672  | 1.193000  | 489 | 545.905823  |
|    | 90  | 63.333340  | 1.117000  | 33  | -723.687866 |
|    | 91  | 62.666672  | 1.062000  | 416 | 345.255432  |
|    | 92  | 62.666672  | 1.062000  | 15  | -295.564026 |
| 24 | 93  | 63.000000  | 0.850000  | 402 | -197.119690 |
|    | 94  | 63.333340  | 0.745000  | 488 | -197.119690 |
|    | 95  | 64.333344  | 0.368000  | 363 | -109.383484 |
|    | 96  | 69.000000  | 0.139000  | 256 | -53.511860  |
| 25 | 97  | 70.000000  | 0.368000  | 499 | 12.900000   |
|    | 98  | 78.666672  | 0.368000  | 213 | 0.000000    |
|    | 99  | 84.000000  | 0.531000  | 251 | 0.000000    |
|    | 100 | 87.000000  | 0.653000  | 367 | 0.000000    |
| 26 | 101 | 20.666670  | -0.693000 | 294 | 0.000000    |
|    | 102 | 27.750000  | 0.255000  | 317 | 25.799999   |

|    |     |            |           |     |              |
|----|-----|------------|-----------|-----|--------------|
|    | 103 | 49.000000  | -0.414000 | 229 | 0.000000     |
|    | 104 | 57.666672  | 0.702000  | 382 | 25.799999    |
| 27 | 105 | 117.000000 | -0.993000 | 26  | -12.900000   |
|    | 106 | 146.000000 | 1.991000  | 420 | 143.693924   |
|    | 107 | 128.000000 | 1.991000  | 499 | 143.693924   |
|    | 108 | 90.000000  | 0.139000  | 472 | -12.900000   |
| 28 | 109 | 115.000000 | 0.139000  | 445 | 0.000000     |
|    | 110 | 101.000000 | 1.991000  | 385 | -197.119690  |
|    | 111 | 46.666672  | 0.531000  | 158 | -109.383484  |
|    | 112 | 39.333340  | -0.993000 | 468 | -774.664246  |
| 29 | 113 | 140.000000 | -0.993000 | 319 | -295.564026  |
|    | 114 | 132.000000 | 1.394000  | 406 | 1508.457642  |
|    | 115 | 57.000000  | 1.062000  | 256 | -1172.770874 |
|    | 116 | 67.666672  | 1.020000  | 91  | 816.873840   |
| 30 | 117 | 70.666672  | 0.816000  | 428 | -399.020996  |
|    | 118 | 72.666672  | 0.653000  | 35  | 653.089050   |
|    | 119 | 73.666672  | 0.368000  | 439 | 289.594116   |
|    | 120 | 71.000000  | 0.139000  | 179 | -109.383484  |
| 31 | 121 | 49.666672  | -0.555000 | 189 | -25.799999   |
|    | 122 | 47.666672  | -0.693000 | 472 | -25.799999   |
|    | 123 | 74.666672  | 0.368000  | 26  | -12.900000   |
|    | 124 | 90.000000  | 0.368000  | 64  | 0.000000     |
| 32 | 125 | 98.000000  | -0.414000 | 464 | 12.900000    |
|    | 126 | 124.000000 | -0.555000 | 13  | -12.900000   |
|    | 127 | 88.000000  | 0.368000  | 365 | 12.900000    |
|    | 128 | 89.000000  | 1.991000  | 361 | -197.119690  |
| 33 | 129 | 60.000000  | 0.780000  | 437 | -109.383484  |
|    | 130 | 50.000000  | -0.414000 | 471 | 237.214264   |
|    | 131 | 74.666672  | -0.993000 | 59  | -156.644424  |
|    | 132 | 66.000000  | -0.693000 | 274 | -295.564026  |
| 34 | 133 | 39.000000  | 0.745000  | 462 | 237.214264   |
|    | 134 | 36.000000  | 0.368000  | 205 | 345.255432   |
|    | 135 | 67.000000  | -0.555000 | 290 | -156.644424  |
|    | 136 | 62.666672  | 0.255000  | 127 | 143.693924   |
|    |     |            |           |     |              |

Tabla 6.3. Valores de los Parámetros de los Codebook Adaptivo y Estocástico para 34 segmentos de voz ( 1020 ms)

## 6.12 DESCRIPCIÓN DEL DECODIFICADOR

Ocurre el proceso inverso que el codificador, su diagrama general se ilustra en la Fig. 6.2. Los bloques que conforman el decodificador se describen a continuación.

### 6.12.1 Desempaquetamiento de las tramas de bits entrantes

Se desempaquetan todos los parámetros de acuerdo a la estructura de las tablas 6.1 y 6.2. Los parámetros obtenidos así son los 10 coeficientes LSP, 4 juegos de periodo ***T<sub>opt</sub>*** y ganancia del codebook adaptivo ***b***, índice del codebook estocástico ***l*** y ganancia estocástica ***g***.

### 6.12.2 Interpolación de los coeficientes LSP

Mediante la interpolación de los coeficientes LSP (ver Fig. 6.30 y Fig. 6.31) se obtienen 4 conjuntos de coeficientes LSP donde cada conjunto, previamente convertido en coeficientes LPC (ecuaciones (6.16), (6.17) y Fig (6.33), se usará en cada sub-segmento de síntesis (60 muestras, 7.5 ms).

### 6.12.3 Codebook Adaptivo

Con los parámetros del codebook adaptivo recuperados, se logra la reconstrucción exacta del codebook obtenido en la etapa de codificación. Esto se grafica en la Fig. 6.85. Si el pitch es fraccionario, entonces se interpola la secuencia y la nueva secuencia interpolada es la excitación de salida del codebook adaptivo.

### 6.12.4 Codebook Estocástico

Con el índice recuperado se selecciona la secuencia del codebook estocástico correspondiente y luego se escala por la correspondiente ganancia. Ver Fig. 6.85.

### 6.12.5 Excitación Total y Filtro de Síntesis

La excitación total conforma la entrada al filtro de síntesis cuya función de transferencia esta dada en la ecuación (5.1). La salida del filtro de síntesis es la voz sintética. Ver Fig. 6.85.

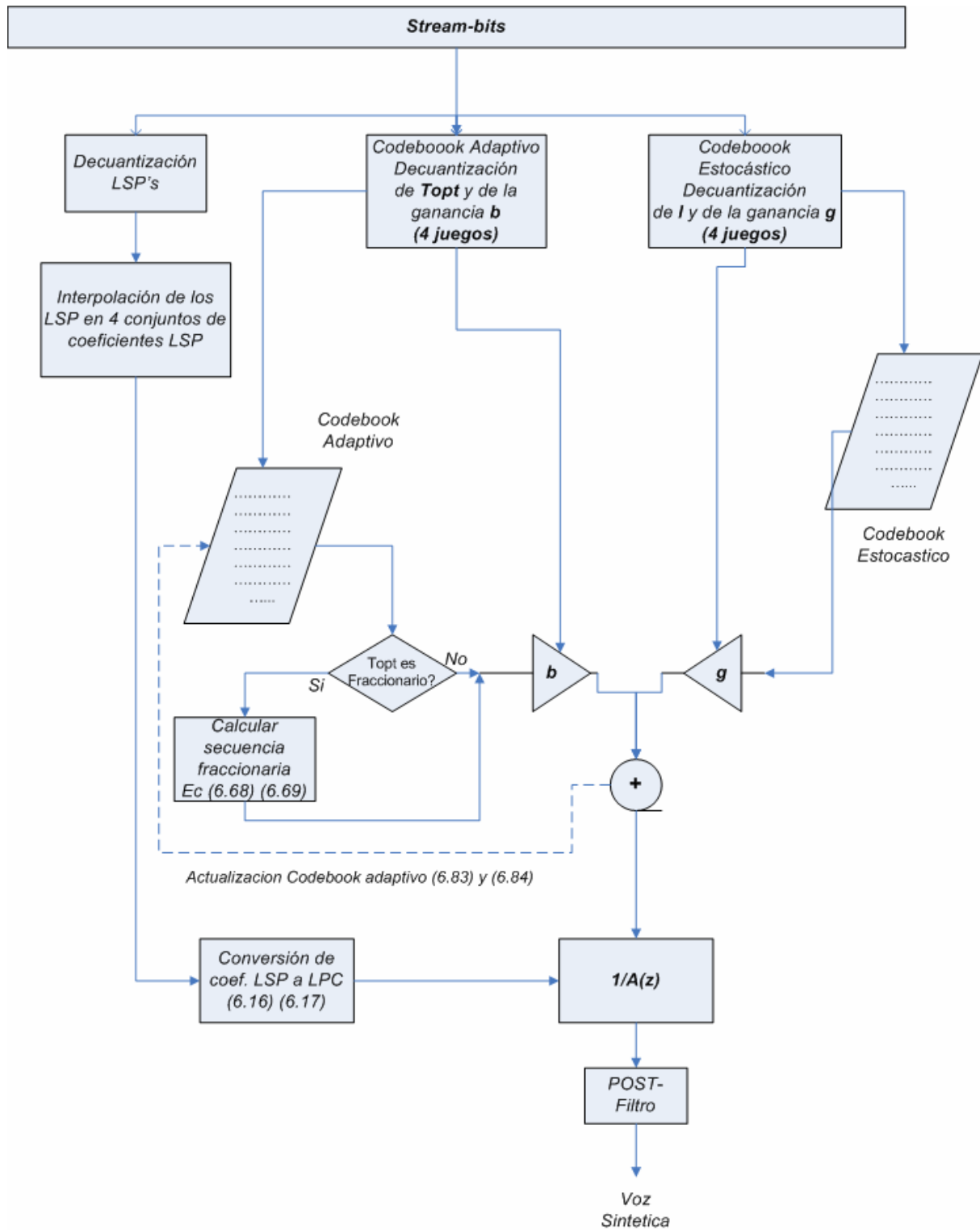


Fig. 6.85 Esquema del algoritmo que ejecuta el decodificador.

Observar que en la Fig. 6.85 se ha incluido un bloque denominado Post-filtro a la salida, dicho bloque se explica mas adelante. La Fig. 6.86 muestra la voz sintética decodificada sin hacer uso del Post Filtro.

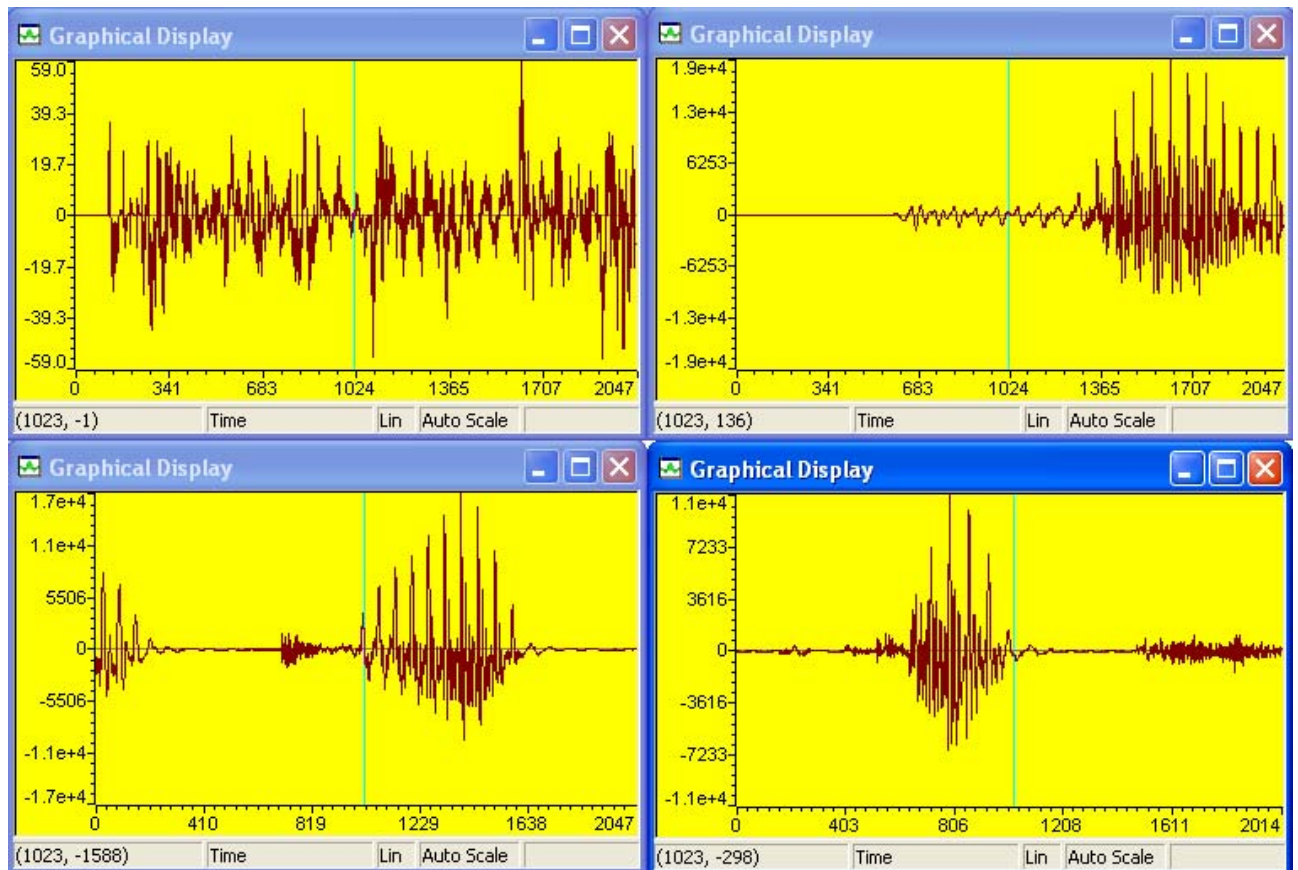


Fig. 6.86 Voz sintética decodificada sin Post-Filtro (comparar con la original en la Fig. 6.5)

### 6.12.6 POST FILTRO

Como se vio, la calidad subjetiva de un codificador basado en el esquema CELP puede ser mejorado mediante la incorporación de un filtro de ponderación perceptual en el proceso de minimización del error. Este filtro perceptual reduce los componentes de ruido en los valles espectrales pero incrementa las componentes del ruido en los picos espectrales. Para mejorar mas la calidad subjetiva de la voz sintética, un postfiltro puede ser añadido para atenuar las componentes en los valles espectrales. (4)

La respuesta en frecuencia de un post-filtro ideal debería seguir los picos y los valles de la envolvente del espectro de la voz sin ocasionar una distorsión global. En primera instancia se puede usar el siguiente post-filtro:

$$H1(z) = \frac{1}{1 + \sum_{i=1}^M a_i \alpha^i z^{-i}} \quad (6.93)$$

La respuesta magnitud se muestra en la Fig. 6.87 para distintos valores de  $0 < \alpha < 1$ .

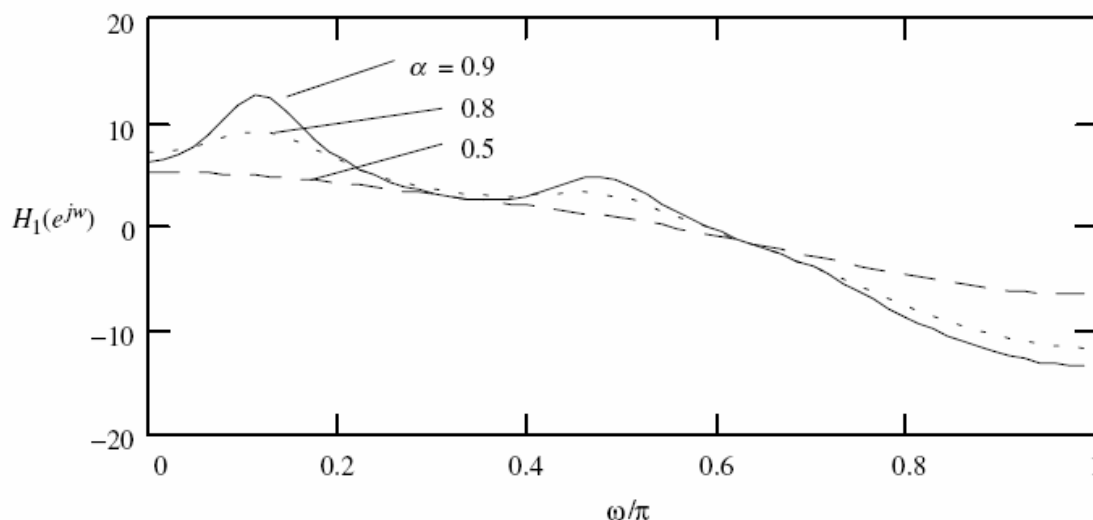


Fig. 6.87 Post-filtro H1 para distintos valores de  $\alpha$ . (4)

Como se puede observar en la Fig. 6.87 el postfiltro reduce el nivel del ruido, pero sin embargo, debido al comportamiento de filtro pasabajo este post-filtro hace que la voz se apague. Para contrarrestar el efecto de filtro pasabajo, se pueden añadir ceros:

$$H2(z) = \frac{1 + \sum_{i=1}^M a_i \beta^i z^{-i}}{1 + \sum_{i=1}^M a_i \alpha^i z^{-i}} \quad (6.94)$$

Con  $0 < \beta < \alpha < 1$ , los cuales son determinados basados en pruebas subjetivas. Si bien es cierto, un post-filtro de la forma dada en la expresión se puede afinar aun mas para mitigar la inclinación del espectro que aun persiste. El post-filtro refinado se da en la siguiente expresión:

$$H_3(z) = (1 - \mu z^{-1}) \frac{1 + \sum_{i=1}^M a_i \beta^i z^{-i}}{1 + \sum_{i=1}^M a_i \alpha^i z^{-i}} \quad (6.95)$$

La comparación entre los tres post-filtros se muestra en la Fig. 6.88.

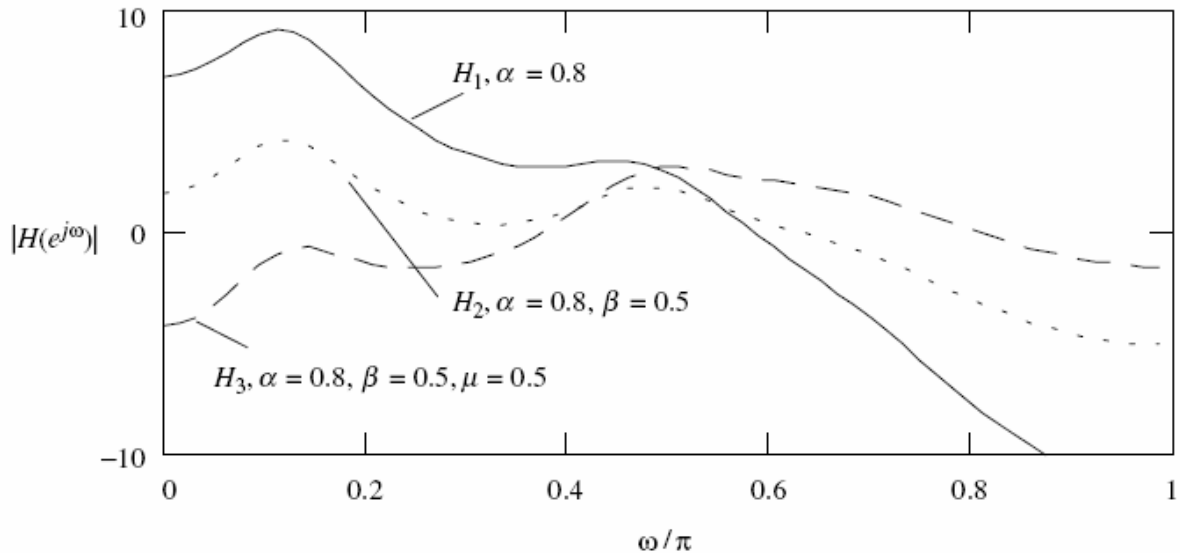


Fig. 6.88 Comparación entre los 3 post-filtros H1, H2 y H3. (4)

### 6.12.6.1 Compensación Espectral Adaptiva

De la ecuación se puede ver que se introduce el término  $(1 - \mu z^{-1})$  para compensar la inclinación en la función de transferencia de tal forma que se reduzca el efecto pasabajo. Ya que la inclinación es variable con la señal, es posible mejorar el filtro haciendo que  $\mu$  sea adaptivo. Esto se logra haciendo que  $\mu$  sea proporcional al primer coeficiente de reflexión.

$$\mu = 0.5 k_1 \quad (6.96)$$

### 6.12.6.2 Control Automático de la Ganancia

Para neutralizar los cambios de magnitud introducidos por el post-filtro, se incluye un bloque de normalización de la magnitud dentro de la arquitectura del post-filtro tal como se muestra en la Fig. 6.89.



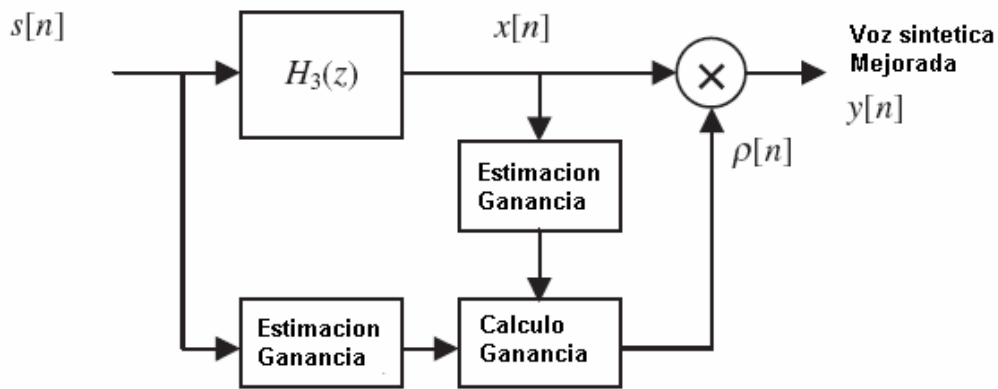


Fig. 6.89 Diagrama de bloques del control automático de la ganancia. [4]

Las ecuaciones que gobiernan esta etapa son las siguientes:

$$\sigma_s^2[n] = \zeta \sigma_s^2[n-1] + (1-\zeta)s^2[n] \quad (6.97)$$

$$\sigma_x^2[n] = \zeta \sigma_x^2[n-1] + (1-\zeta)x^2[n] \quad (6.98)$$

$$\rho[n] = \sqrt{\frac{\sigma_s^2[n]}{\sigma_x^2[n]}} \quad (6.99)$$

$$y[n] = \rho[n]x[n] \quad (6.100)$$

Luego de añadido el post-filtro con las características dadas en la expresión (6.98) y en la Fig. 6.89, y con valores de  $\alpha = 0.8$  y  $\beta = 0.5$ , la salida de la voz sintética asociada se ilustra en la Fig. 6.90. Comparar esta salida con la mostrada en la Fig. 6.86. Si bien es cierto las gráficas son muy similares en el dominio del tiempo, la mejora subjetiva se aprecia cuando se escuchan las secuencias por medio de un altavoz.

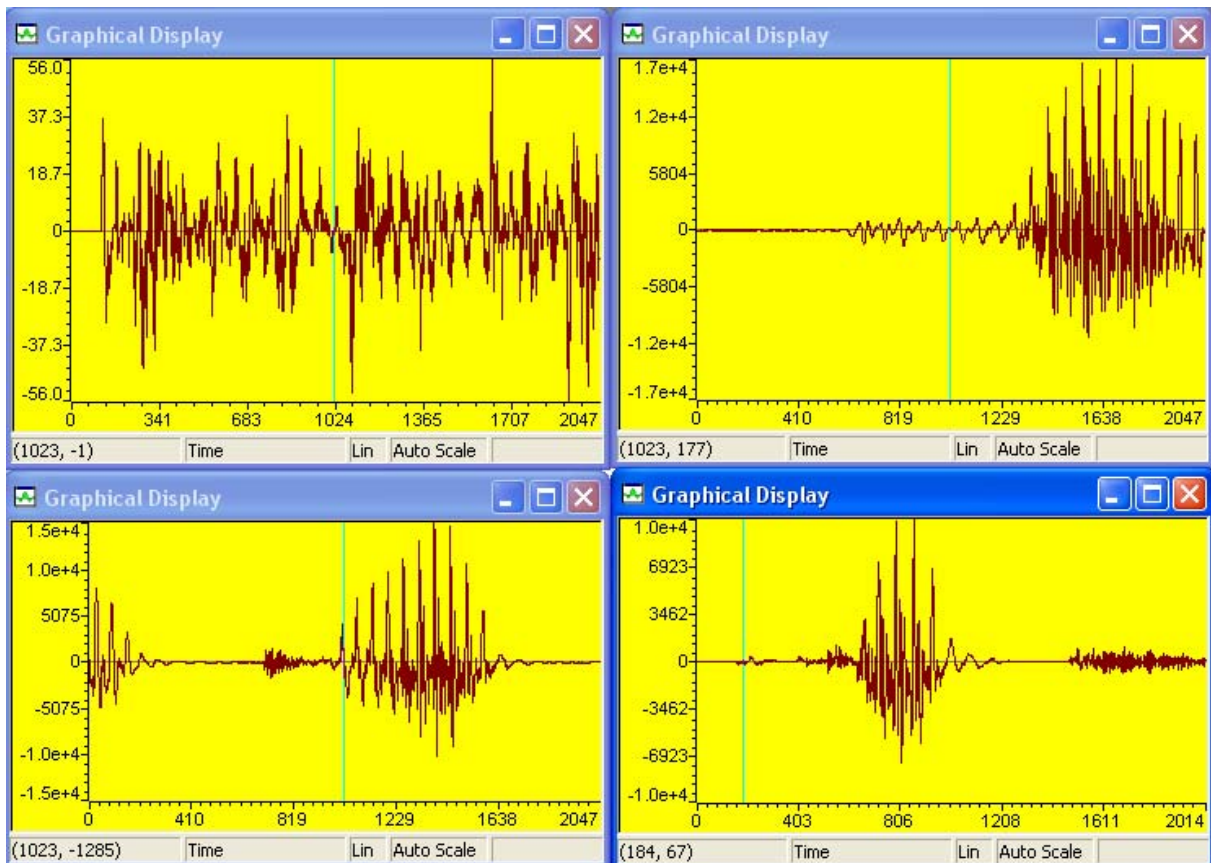


Fig. 6.90 Voz sintética a la salida del Post-filtro con control automático de ganancia.

### 6.13 Resumen de las mejoras implementadas

En esta sección se describen las mejoras incorporadas en varios de los bloques que conforman tanto el codificador como el decodificador. Hay que recalcar que en el capítulo 7 se describen las mejoras para optimizar el código en el DSP TMS320C6711.

#### 6.13.1 Bloque de conversión de coeficientes LPC a LSP

El algoritmo empleado (ver figuras Fig. 6.18 y Fig. 6.19) aprovecha la propiedad de entrelazamiento de los coeficientes LSP (ver Fig. 6.24). De esta forma se optimiza la búsqueda de los ceros para la expresión (6.15). Un algoritmo convencional hace un barrido completo del rango  $[0, \pi]$  para cada una de las funciones  $P_o(\omega)$  y  $Q_o(\omega)$  tal como se muestra en la Fig. 6.91. De esta forma el algoritmo tiene que barrer 2 veces el rango de  $[0, \pi]$ . El algoritmo implementado solo hace un solo barrido del rango  $[0, \pi]$ , y alterna la búsqueda entre raíces de  $P_o(\omega)$  y  $Q_o(\omega)$ , esta búsqueda con alternancia es permitida por la propiedad de entrelazamiento. Ver Fig. 6.92.

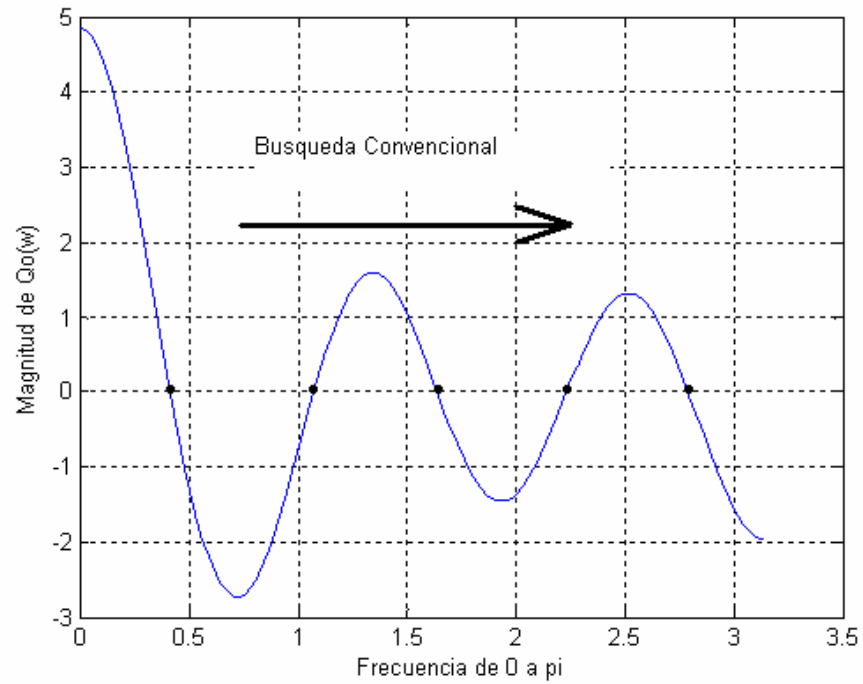


Fig. 6.91 Búsqueda convencional de raíces.

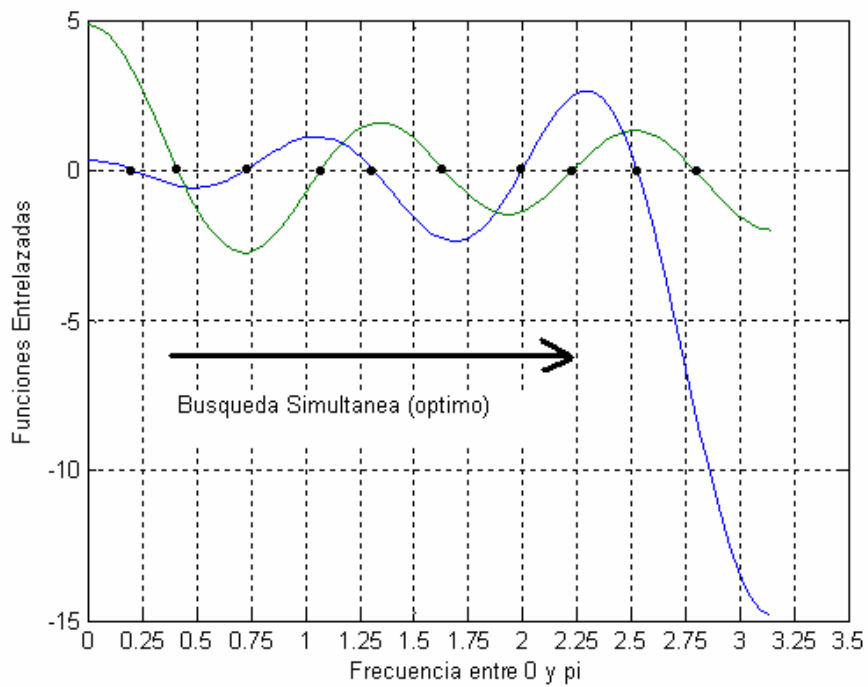


Fig. 6.92 Búsqueda óptima que aprovecha la propiedad de entrelazamiento de los coeficientes LSP (raíces).

### 6.13.2 Bloques de cuantización de coeficientes LSP e interpolación de LSP

Se encontró que no es necesario normalizar las frecuencias al rango  $[0, 1]$  para el bloque de cuantización de los coeficientes LSP (sección 6.8) ni para el bloque de interpolación de los coeficientes LSP (sección 6.9). De esta forma, el trabajo en el dominio  $[0, \pi]$  no crea inconvenientes en los resultados ni en la calidad de la voz decodificada.

### 6.13.3 Optimización de la convolución recursiva para el bloque de búsqueda en el Codebook Adaptivo

Se implementó y optimizó la convolución recursiva para el codebook adaptivo de acuerdo al *shift* correspondiente que para el caso de este codebook es de 1. La optimización del algoritmo de convolución recursiva evita loops innecesarios y mejora la velocidad de ejecución del bloque. El algoritmo de convolución recursiva optimizado para el Codebook adaptivo se muestra a continuación:

```

for (i = l-1; i > len-1; i--)
    y[i] = y[i - 1];

for (i = len-1; i > 0; i--)
    y[i] = y[i-1] + ex[0] * h[i];

y[0] = ex[0] * h[0];

```

Para lograr optimizar la convolución recursiva se analizó al detalle lo que ocurría dentro del esquema recursivo y se encontró que el número de loops que incluyen multiplicaciones podía ser reducido a la longitud de la respuesta impulsional truncada '*len*'. Como se aprecia en el algoritmo mostrado líneas arriba hay dos bucles 'for', el primero no contiene multiplicaciones solo desplazamientos (*shift*). El otro bucle 'for' contiene multiplicaciones pero estas son reducidas en número al valor de '*len*'.

### 6.13.4 Optimización de la convolución recursiva para el bloque de búsqueda en el Codebook Estocástico

Al igual que en el caso del codebook adaptivo, en el caso del codebook estocástico podemos optimizar la convolución recursiva para reducir el número de multiplicaciones. En este caso el *shift* es de 2 y también sacamos ventaja de la naturaleza del codebook (77% de esparcidad) es decir, ya que la mayoría de términos de este codebook son cero, podemos diseñar el algoritmo para que no ejecute multiplicaciones (ya que cuando el valor es cero la

salida es cero). También sacamos ventaja del hecho de la ocurrencia de 2 valores cero seguidos. El algoritmo optimizado de convolución recursiva para el codebook estocástico se muestra a continuación:

```

for (i = 59; i > len; i--)
    y_cgain[i] = y_cgain[i - 2];

if ((ex[0] == 0) & (ex[1] == 0))
    {
        y_cgain[30]= y_cgain[28];
        for (i = 29; i > 1; i--)
            y_cgain[i] = y_cgain[i-2];
            y_cgain[1]= 0;
            y_cgain[0]= 0;
        }

    else if ((ex[0] == 0) & (ex[1] == 1))
        {y_cgain[30]= y_cgain[28]+h[29];
        for (i = 29; i > 1; i--)
            y_cgain[i] = y_cgain[i-2] + h[i-1];
        y_cgain[1]= h[0];
        y_cgain[0]= 0;
        }

    else if ((ex[0] == 0) & (ex[1] == -1))
        {y_cgain[30]= y_cgain[28]-h[29];
        for (i = 29; i > 1; i--)
            y_cgain[i] = y_cgain[i-2] -h[i-1];
        y_cgain[1]= -h[0];
        y_cgain[0]= 0;
        }

    else if ((ex[0] == 1)& (ex[1] == 0))
        {y_cgain[30]= y_cgain[28];
        for (i = 29; i > 1; i--)
            y_cgain[i] = y_cgain[i-2] + h[i];
        y_cgain[1]= h[1];
        y_cgain[0]= h[0];
        }

    else if ((ex[0] == -1) & (ex[1] == 0))
        {y_cgain[30]= y_cgain[28];
        for (i = 29; i > 1; i--)
            y_cgain[i] = y_cgain[i-2] -h[i];
        y_cgain[1]= -h[1];
        y_cgain[0]= -h[0];
        }

    else if ((ex[0] == 1) & (ex[1] == -1))
        {y_cgain[30]= y_cgain[28]-h[29];
        for (i = 29; i > 1; i--)
            y_cgain[i] = y_cgain[i-2] + h[i] -h[i-1];
        y_cgain[1]= h[1] -h[0];
        }

```

```

    y_cgain[0]= h[0];
}

else if ((ex[0] == -1) & (ex[1] == 1))
    {y_cgain[30]= y_cgain[28]+h[29];
    for (i = 29; i > 1; i--)
        y_cgain[i] = y_cgain[i-2] -h[i] + h[i-1];
    y_cgain[1]= h[0]-h[1];
    y_cgain[0]= -h[0];
}

else if ((ex[0] == 1) & (ex[1] == 1))
    {y_cgain[30]= y_cgain[28]+h[29];
    for (i = 29; i > 1; i--)
        y_cgain[i] = y_cgain[i-2] + h[i] + h[i-1];
    y_cgain[1]= h[0]+h[1];
    y_cgain[0]= h[0];
}

else /*ex[0] == -1) & (ex[1] == -1*/
    {y_cgain[30]= y_cgain[28]-h[29];
    for (i = 29; i > 1; i--)
        y_cgain[i] = y_cgain[i-2] -h[i] - h[i-1];
    y_cgain[1]= -h[0]-h[1];
    y_cgain[0]= -h[0];
}

```

Como se puede apreciar en el código mostrado, el primer bucle 'for' realiza solo desplazamientos (no incluye multiplicaciones). Luego se tiene una serie de bucles 'if' que detectan todos los pares posibles, esto es óptimo ya que de esta forma se han eliminado todas las multiplicaciones de los bucles quedando solo sumas y restas.

### 6.13.5 Cuantización de la ganancia estocástica

El estandar FS1016 usa 5 bits para la cuantización de la ganancia estocástica y además usa 4 bits para el bloque FEC. En la presente implementación se destina un bit mas para dicho bloque cuantizador, es decir 6, mientras que se destinan 7 bits para el bloque FEC, tal como se describe en el capítulo 5. La tasa de bits total se incrementa ligeramente de 4.8 Kbps a 4.9666 Kbps lo cual es aceptable ya que se obtiene un poco más de calidad en la voz decodificada, tal como se ha encontrado en la práctica.

El cuantizador, por lo tanto, es de 64 niveles (6 bits) y fue diseñado de acuerdo al algoritmo de Lloyd con la ayuda de la herramienta matemática MATLAB. Los detalles del cuantizador se describen en la sección 6.10.5.4. Destinando un bit mas al cuantizador, resulta en una calidad subjetiva mejorada con relación al esquema original del FS1016.

## CAPITULO VII

### DESCRIPCION DEL DSP TMS320C6711, DETALLES DE LA IMPLEMENTACION Y OPTIMIZACION

El presente capítulo describe, en primer lugar, las características y prestaciones del DSP utilizado. Luego se pasa a describir los algoritmos implementados que corresponden al Codificador y Decodificador en dicho DSP. Finalmente se muestran las etapas de optimización que se realizaron para conseguir una mejor performance en la ejecución del código.

#### 7.1 DSP TMS320C6711 Overview

Los Procesadores Digital de Señales tales como los de la familia TMS320 de Texas Instruments vienen siendo usados en un amplio rango de aplicaciones de control, comunicaciones, procesamiento de imágenes, de voz, etc. El TMS320C6711 es un procesador digital de señales de punto flotante que es adecuado para algoritmos numéricamente intensivos. (47)

Los DSP de la familia TMS320C6X (C6X) de Texas Instruments son DSP de propósito especial con un tipo de arquitectura especializada y un set de instrucciones apropiado para procesamiento de señales. La arquitectura de un Dsp de la familia C6X es adecuado para algoritmos que requieren flexibilidad y eficiencia. Basado en una arquitectura VLIW (Very Long Instruction Word), los DSP de la familia C6X son considerados los más potentes DSP desarrollados por Texas Instruments.

##### 7.1.1 Tarjeta DSK C6711



Fig. 7.1 Tarjeta DSK TMS320C6711 (41)

## ■ Hardware

- 150 MHz 'C6711 DSP
- TI 16-bit A/D Converter ('AD535)
- External Memory
  - 16M Bytes SDRAM
  - 128K Bytes Flash ROM
- LED's
- Daughter card expansion
- Power Supply & Parallel Port Cable

## ■ Software

- Code Generation Tools
  - (C Compiler, Assembler & Linker)**
- Code Composer Debugger
- Example Programs & S/W Utilities
  - Power-on Self Test
  - Flash Utility Program
  - Board Confidence Test
  - Host access via DLL
  - Sample Program(s)

Las partes mas importantes de la tarjeta se describen en la Fig. 7.2:

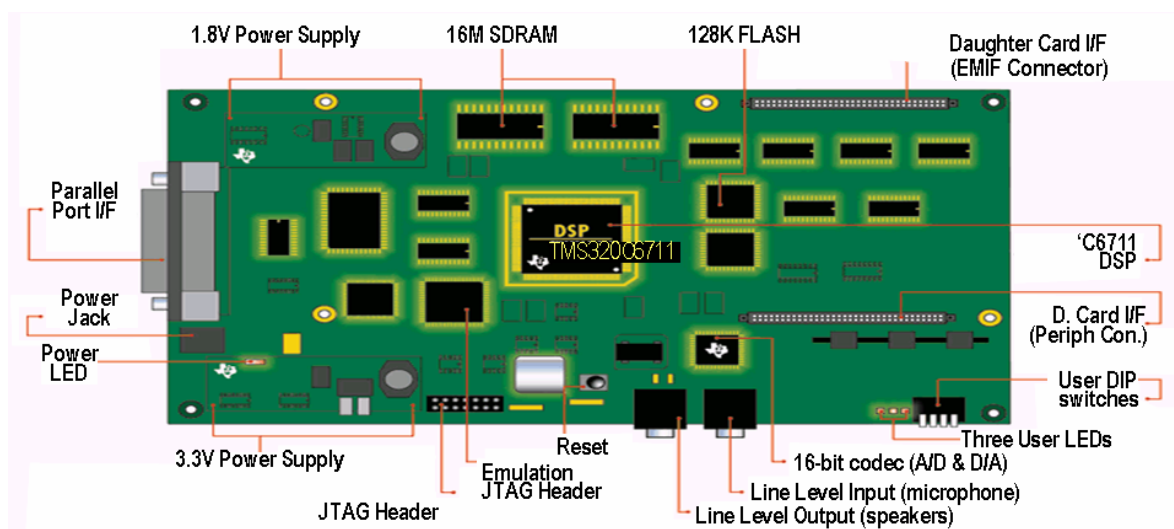


Fig. 7.2 Partes de la tarjeta DSK TMS320C6711 (41)



### 7.1.2 Mapa de Memoria

Se muestra en la Fig. 7.3.

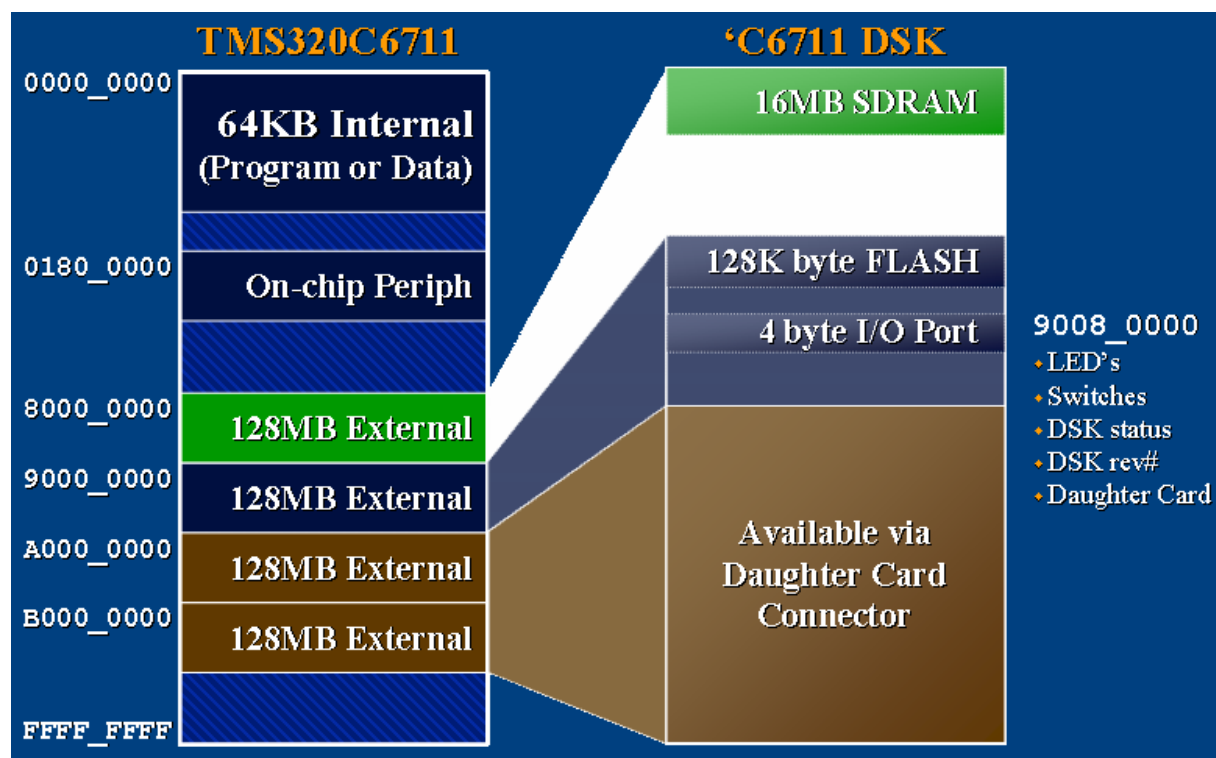


Fig. 7.3 Mapa de memoria del DSK TMS320C6711 (47)

### 7.1.3 Arquitectura de la familia TMS320C67X

El procesador C6711 está basado en una arquitectura VLIW (Very Long Instruction Word). La memoria de programa interna está estructurada de tal forma que un total de 8 instrucciones pueden ser ejecutadas en cada ciclo de reloj. Por ejemplo con un clock rate de 150MHz, el C6711 es capaz de poner en "fetch" 8 instrucciones de 32 bits cada 1 / 150 MHz (o 6.6ns). El C6711 incluye una memoria interna de 72KB (64KB+4KB+4KB), 8 unidades funcionales (6 ALU's y 2 multiplicadores), un bus de dirección de 32 bits capaz de direccionar 4 Gigabytes, y además 2 conjuntos de registros de 32 bits multipropósito.

La Fig. 7.4 ilustra el diagrama de bloques funcional del procesador C6711. Dos bancos de memoria independientes pueden ser accedidos usando dos buses independientes. Tanto la memoria de programa, la memoria de datos y la memoria DMA (Direct Access Memory), cada una tiene un bus individual, permitiendo al C6X ejecutar simultáneamente "program fetches", lectura y escritura de datos, y operaciones DMA. El espacio de memoria es direccionable por bytes. La memoria interna está organizada como

memoria separada de programa y espacio de memoria de datos, contando con 2 puertos internos de 32 bits para acceder memoria interna.

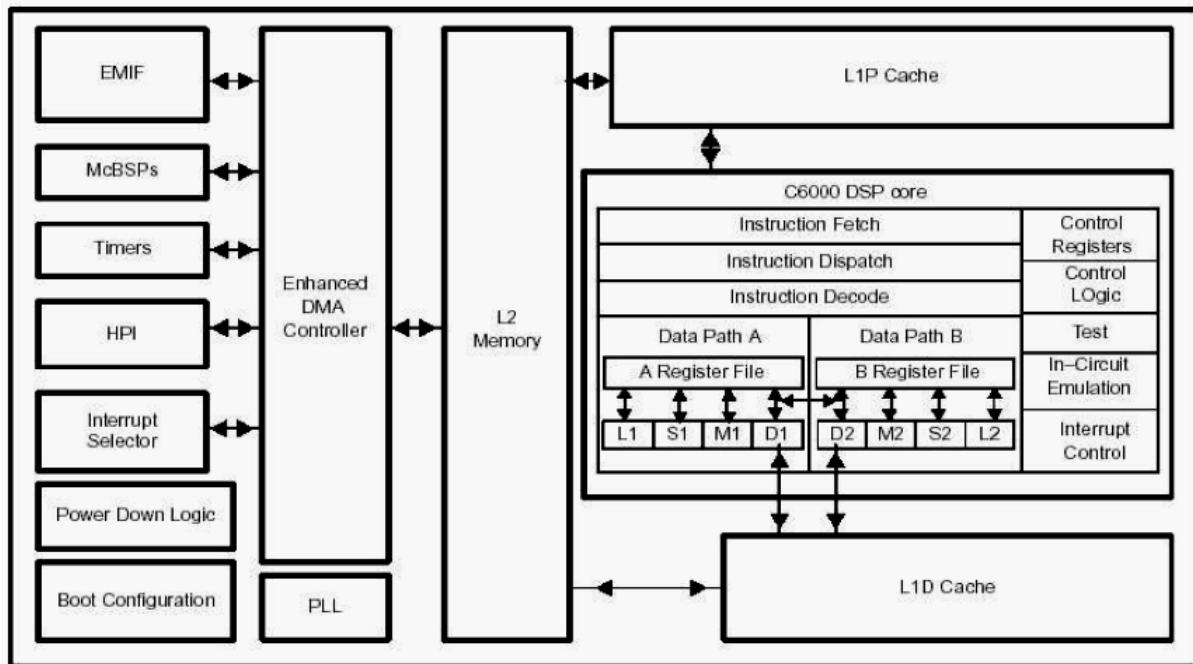


Fig. 7.4 Arquitectura interna de la familia TMS320C67X (49)

### 7.1.4 Code Composer Studio

Es un ambiente completo de desarrollo integrado (IDE) que soporta todas las plataformas de DSP de Texas: TMS320C6000, Texas TMS320C5000, Texas TMS320C2000.

Características y beneficios:

- Ambiente de desarrollo que integra todas las herramientas dentro de una aplicación amigable y fácil de usar.
- Herramientas para análisis en Tiempo Real sin interferir en el programa del DSP.
- Compilador con herramientas de optimización de código orientado a tamaño y performance.
- Kernel escalable DSP/BIOS.
- Visualización de datos con posibilidad de múltiples formatos.
- Arquitectura open plug-in que te permite integrar herramientas especializadas de terceros.
- Escaneo Real-Time Jtag.

El entorno de trabajo se muestra en la Fig. 7.5.

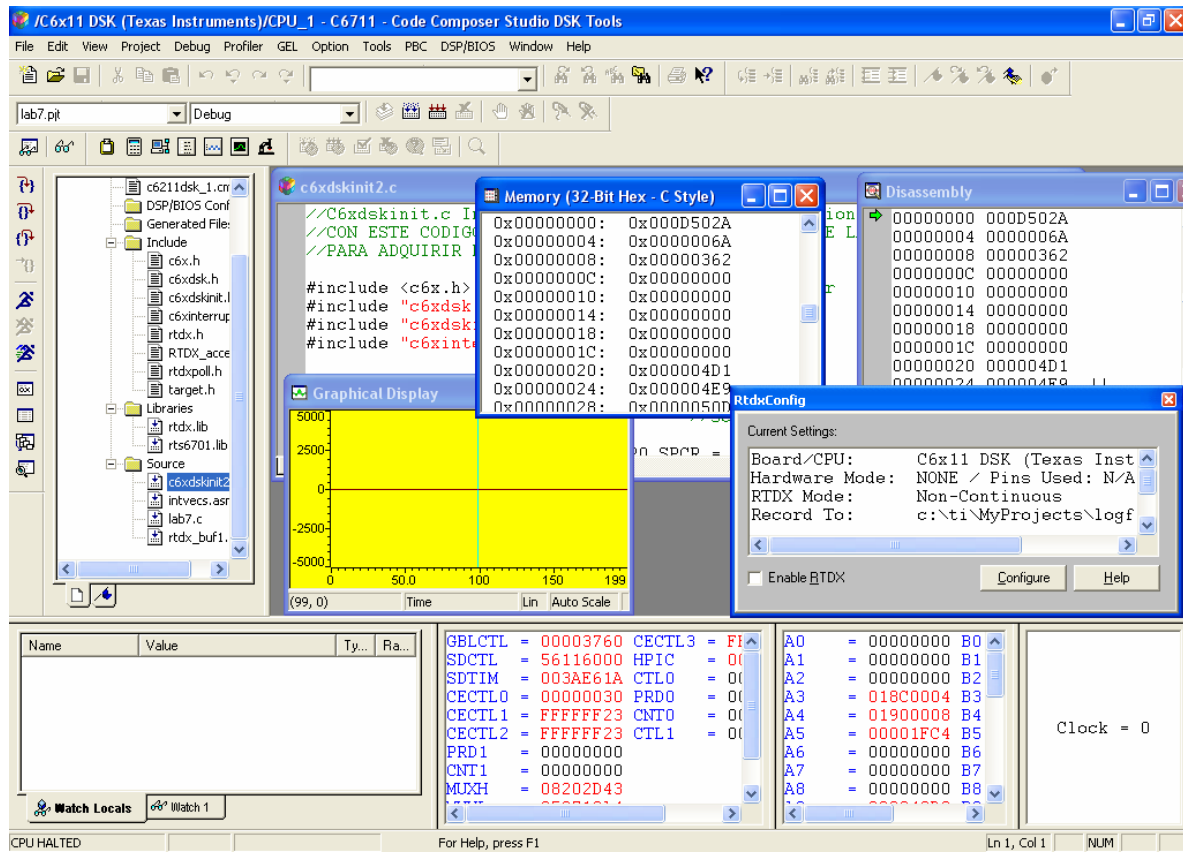


Fig. 7.5 Entorno de desarrollo Code Composer Studio.

## 7.2 Desarrollo del software en el Code Composer

De entre los componentes que nos ofrece el Code Composer para el desarrollo de software se pueden mencionar las siguientes:

- Herramientas de generación de código para el TMS320C6000.
- Code Composer Studio Integrated Development Environment (IDE).
- DSP/BIOS plug-ins and API.
- RTDX plug-in, interface para computadora y API.

Estos componentes trabajan juntos tal como se muestra en la Fig 7.6.

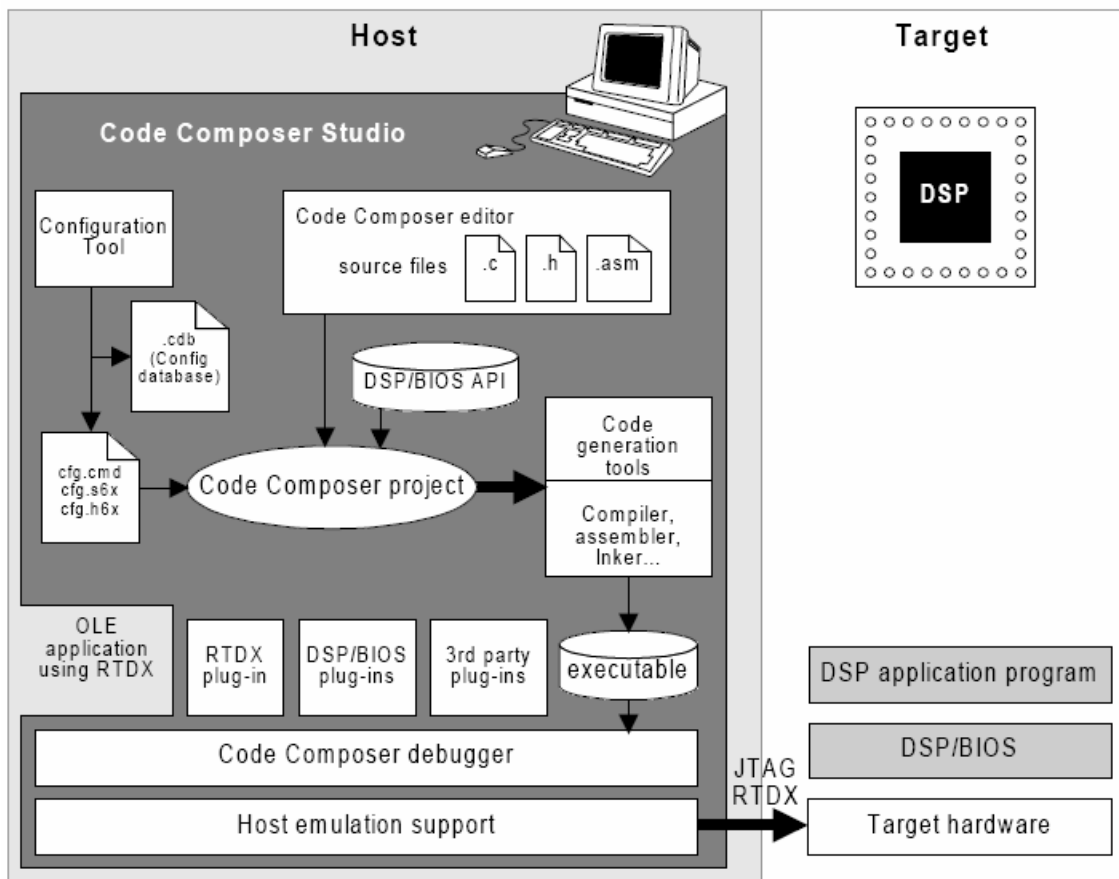


Fig. 7.6 Componentes en el desarrollo de Software en el Code Composer Studio. (48)

El diagrama de flujo del desarrollo de software en el Code Composer para los DSP de la familia C6X se muestra en la Fig. 7.7.

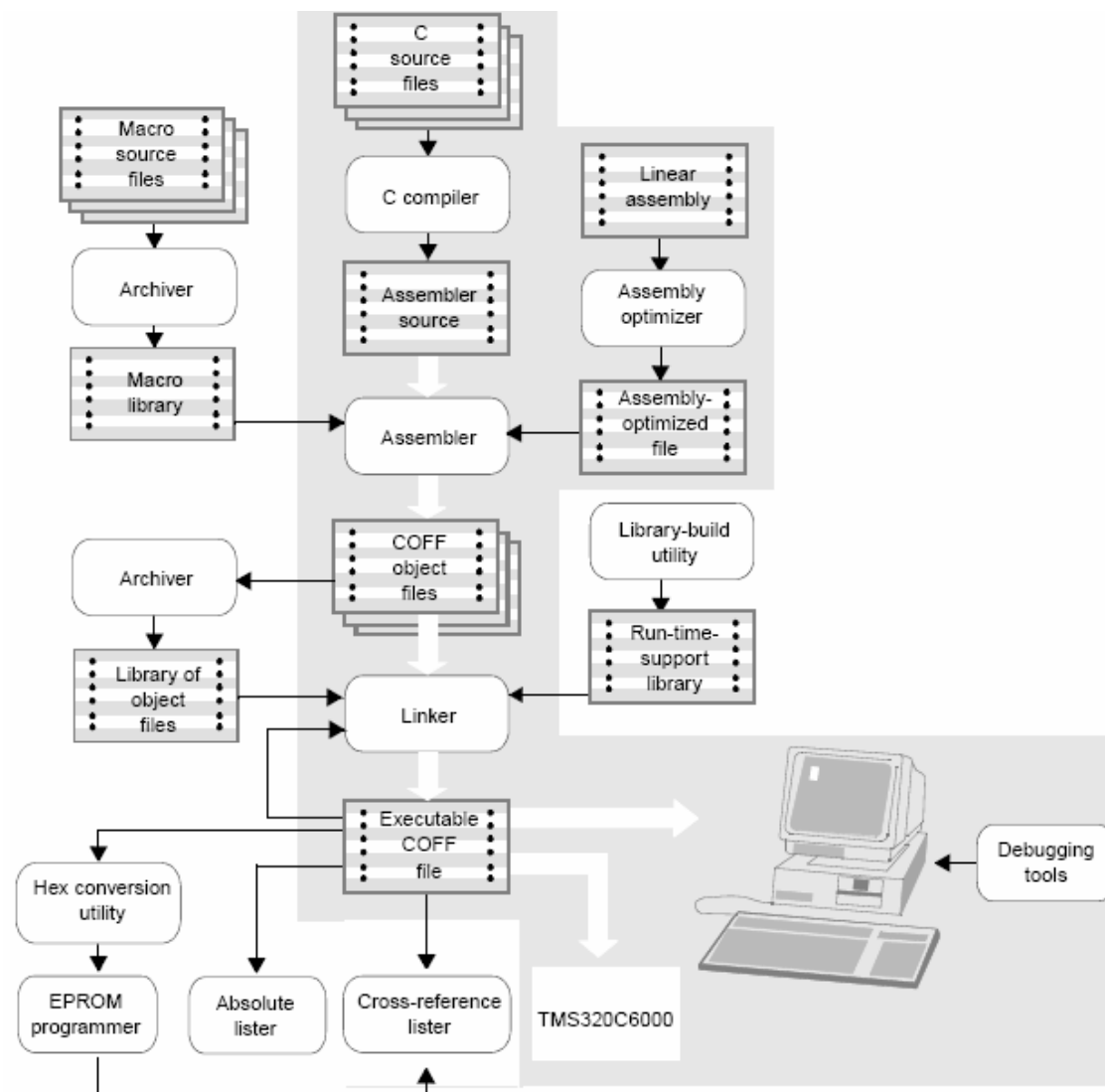


Fig. 7.7 Diagrama de Flujo para la generación de software en el Code Composer. (48)

### 7.2.1 Creación del proyecto

Abrir el Code Composer Studio, Ir al Menú Project → New. Se abrirá la ventana mostrada en la Fig. 7.8, donde se introduce el nombre y la ruta del proyecto a crear.

El tipo de proyecto es un archivo ejecutable (.out) es decir es un programa ejecutable por el Target (DSP TMS320C67XX) el cual está totalmente compilado, ensamblado y enlazado. Al hacer clic en Finalizar, automáticamente se crea un archivo de proyecto lab1.pjt el cual guarda las rutas y todos los archivos relacionados al proyecto.

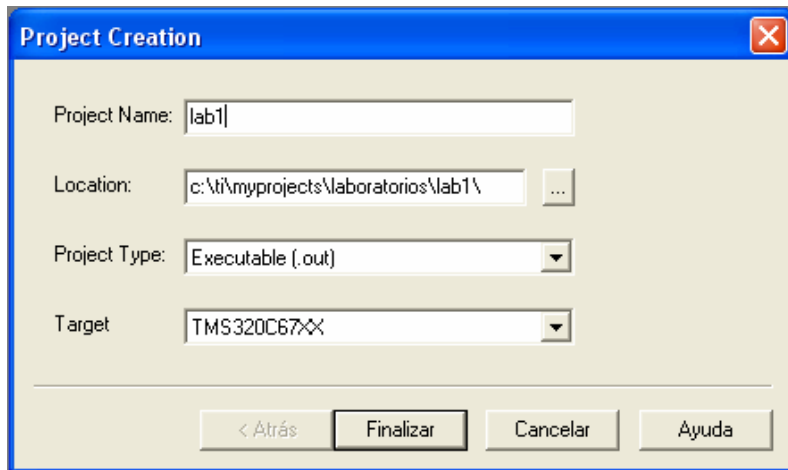


Fig. 7.8 Creación de un proyecto en el Code Composer Studio.

Nótese que en la ventana Project View (Fig. 7.9), el único archivo agregado por defecto al entorno de trabajo (en GEL files) es el DSK6211\_6711.gel : El cual es un archivo usado por el Code Composer en conjunto con el Target (DSKC6711 en el presente caso), independientemente de cualquier proyecto y el cual permite al CCS hacer las siguientes funciones: quick test , reset del CPU, inicializar el EMIF (External Memory Interface) con lo cual el CCS se entera de la geometría y tamaño de la memoria, entre otras cosas.

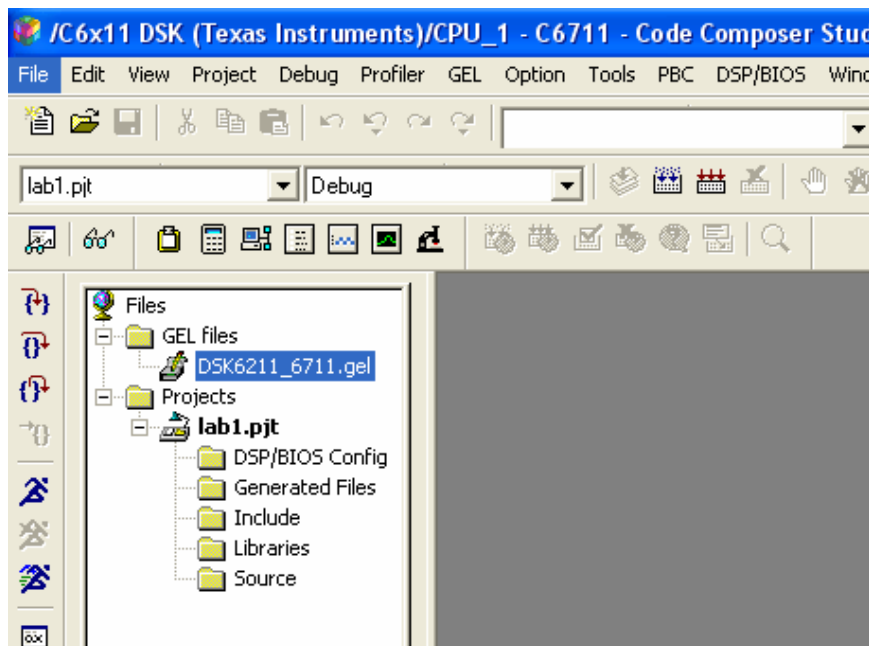


Fig. 7.9 Vista de la ventana Project.

## 7.2.2 Añadiendo archivos básicos de soporte y librerías.

Hay varios archivos, dados por Texas Instruments que deben ser añadidos a los proyectos. Dichos archivos contienen definiciones de registros, rutinas de inicialización de la tarjeta DSK y soporte de librerías pre-compiladas. La necesidad de agregar más o menos archivos de soporte y/o librerías depende del tipo de aplicación que se vaya a desarrollar. Para nuestro caso debemos agregar los siguientes archivos:

- **C6211dsk.cmd**: Este archivo es un archivo de tipo Linker Command Files, el cual es muy importante porque permite configurar la memoria del sistema mediante una distribución eficiente de las secciones (de código o data) dentro del mapa de memoria. Este archivo informa al enlazador cómo están organizados los vectores, la memoria interna, externa y la flash. Este archivo consta de éstas dos partes principales:

```
MEMORY
{
    VECS: o=00000000h   l=00000200h           /* interrupt vectors */
    PMEM: o=00000200h   l=0000FE00h       /* Internal RAM (L2) mem */
    BMEM: o=80000000h   l=01000000h       /* CE0, SDRAM, 16 MBytes */
}
```

Con ésta directiva que empieza con la palabra Memory, se delimitan las secciones de la memoria especificando exactamente el mapeo de todas las memorias físicas que contiene la tarjeta Dsk, memoria interna (L2 Cache), Memoria Externa (RAM, Flash, EEPROM), las cuales se traducen en los espacios de memoria VECS, PMEM y BMEM para éste caso.

```
SECTIONS
{
    .intvecs    >    VECS
    .text       >    PMEM
    mydata      >    PMEM
    mycode      >    BMEM
    .rt dx_text >    BMEM
    .far        >    BMEM
```

```

.stack      >  BMEM
.bss       >  BMEM
.cinit     >  BMEM
.pinit     >  PMEM
.cio       >  BMEM
.const    >  BMEM
.data     >  BMEM
.rtdx_data >  BMEM
.switch   >  BMEM
.systemem >  BMEM

}

```

Con la directiva Sections se distribuye el código escrito en las diferentes partes de la memoria. La sección .text generalmente contiene las primeras partes de un código extenso por lo que debe ir ubicada en una memoria “rápida”, la cual en este caso es la PMEM (Cache de nivel 2 L2 Cache), el resto del código y variables adicionales van en la memoria externa BMEM (Banco de memoria SDRAM de 16Mb).

Observése que se han creado dos secciones llamadas “mydata” y “mycode”, cada una de estas secciones ayuda a direccionar ya sea partes del código o variables a un determinado segmento de memoria (interna o externa).

- **rts6701.lib:** Es una actualización de la librería rts6700.lib (Run Time Support Library) que viene incluido en el Code Composer Studio, la cual contiene código de soporte para las aplicaciones y permite al Code Composer la depuración en Tiempo Real, es obligatorio incluir éste archivo en todo proyecto.
- **c6xdskinit.c:** En este archivo se incluyen las definiciones de las funciones que inicializan los periféricos de la tarjeta DSK, para tenerlo listo para la adquisición de los datos a través del Codec. Las rutinas que se incluyen en este archivo son las siguientes:

*void mcbsp0\_init():* Para inicializar el puerto serial.

*void mcbsp0\_write(int):* Rutina de escritura hacia el Codec a través del puerto serial.

*int mcbsp0\_read():* Rutina de lectura del Codec a través del puerto serial.

*void TLC320AD535\_Init():* Inicialización del Codec a través del puerto Serial.

*void c6x\_dsk\_init():* Reseteo de interrupciones y configuración del EMIF.



*void comm\_poll():* Para hacer una lectura/escritura del Codec via Polling.

*void comm\_intr():* Dispara (triggering) todas las anteriores funciones.

*int input\_sample():* Rutina que hace uso de *mcbasp0\_read()*.

*void output\_sample(int):* Rutina que hace uso de *mcbasp0\_write(int)*.

La ejecución de todas las funciones anteriores deja todo listo para trabajar con todos los periféricos disponibles de la tarjeta DSK. Es un archivo muy útil y modificable de acuerdo a cada necesidad, evitará que se escriban desde el comienzo todo el código para inicializar la tarjeta DSK.

- **intvecs1.asm:** Definición de las interrupciones en el DSP. Aquí se crea la IST (Interrupt Service Table). En este caso la única interrupción que se va a necesitar es la interrupción producida por el Codec cada vez que genera una muestra de la señal de entrada, la cual se ubica por conveniencia como Interrupción 10.

Hasta el momento, el entorno de trabajo luce tal como se muestra en la Fig. 7.10:

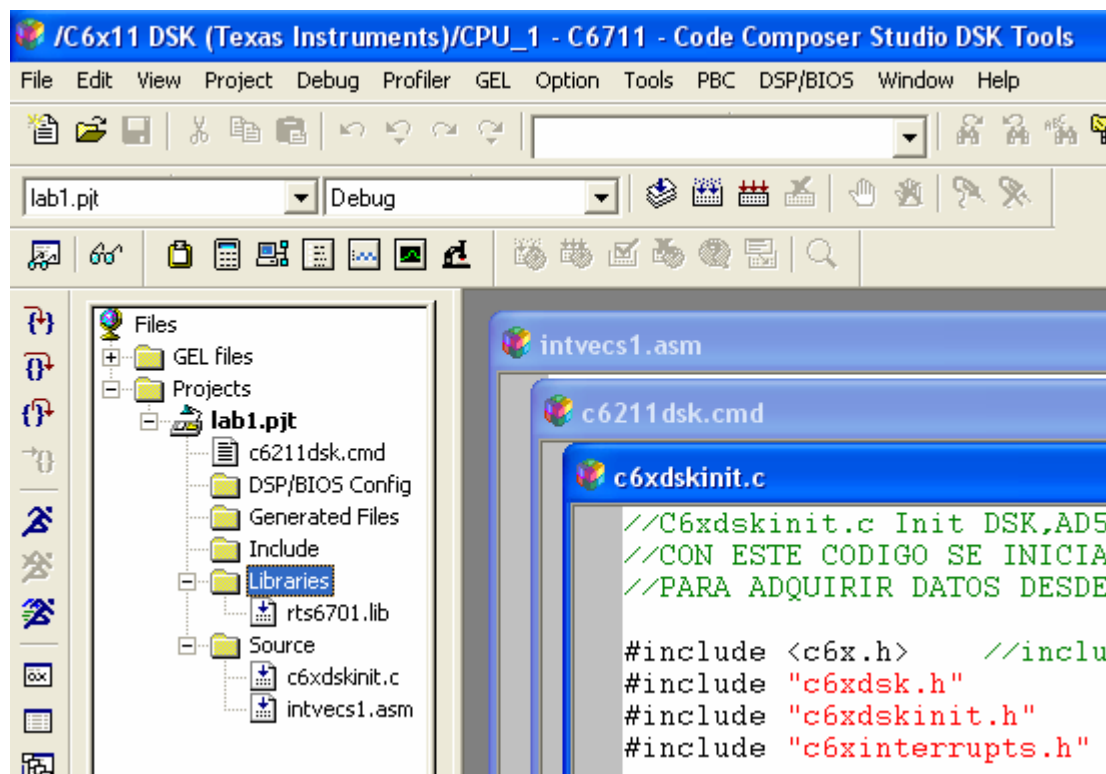


Fig. 7.10 Proyecto con los archivos de soporte añadidos.

Se pueden observar los 4 archivos que se han agregado, pero notar también que la primera parte del archivo `c6xdskinit.c` hace referencia a 4 archivos de cabecera, los cuales aparecen después de cada directiva `#include`. Para que estos archivos sean visibles en el proyecto hacer lo siguiente:

Menu Project → Scan All Dependencies.

Con lo cual ya podemos ver los 4 archivos en la ventana Project View ubicados en la carpeta Include:

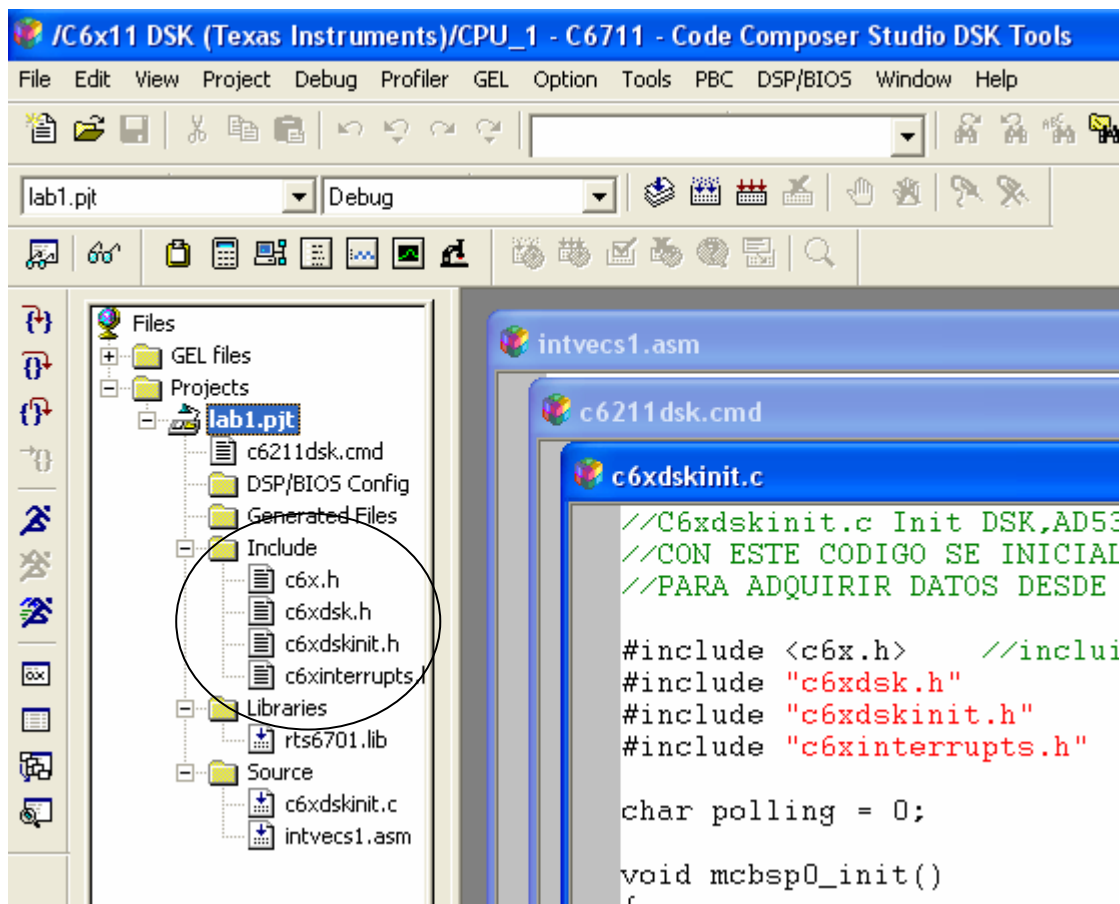


Fig. 7.11 Visualización de las dependencias en la ventana Project.

Con los archivos de soporte ya agregados, se procede a añadir los archivos de código fuente que ejecutarán las funciones del codificador. El decodificador se ha implementado como otro proyecto separado del codificador.

Los archivos de código fuente realizan cada uno una función determinada la cual puede ser optimizada sin interferir con el resto de funciones del codificador.

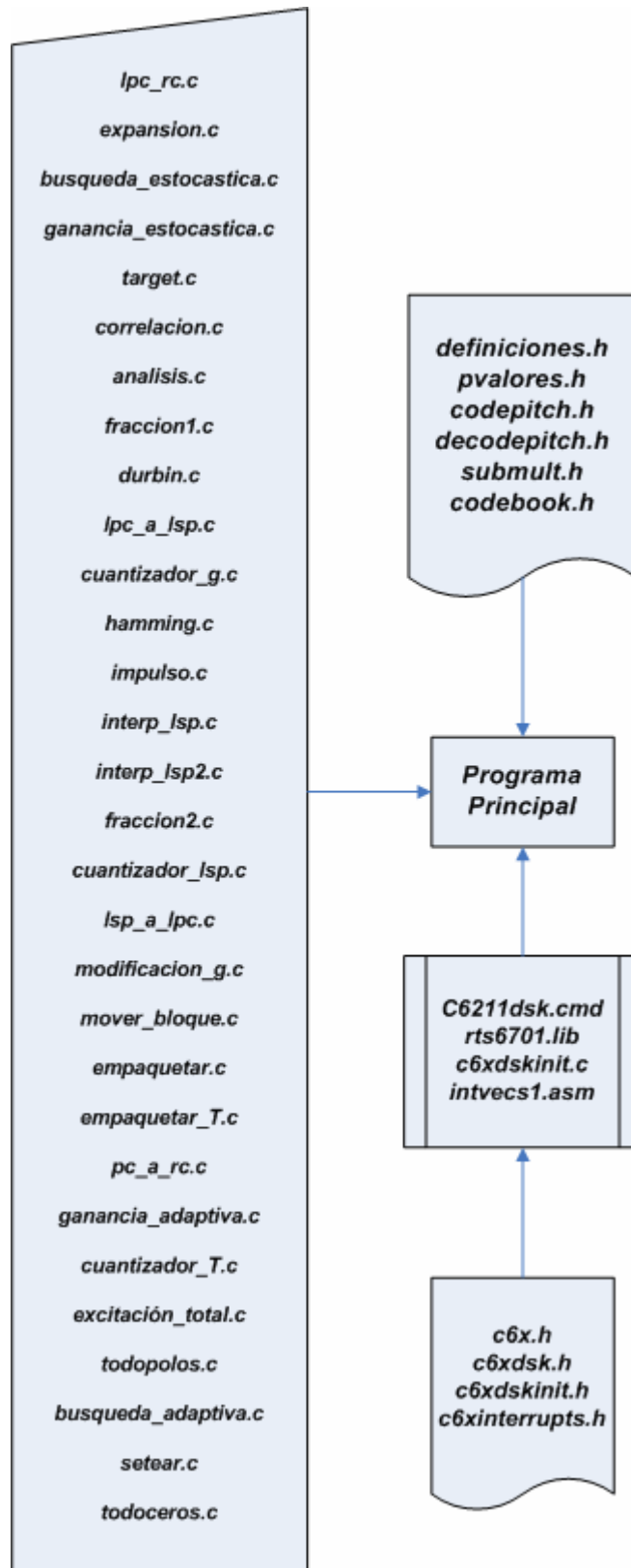
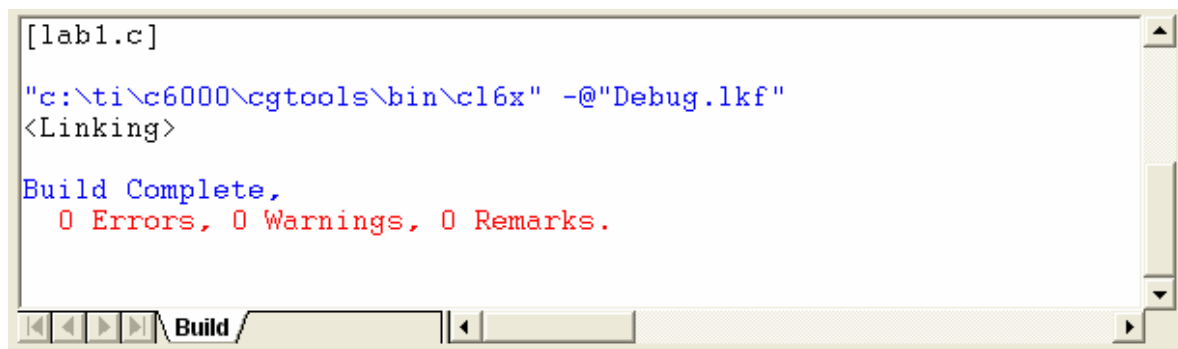


Fig. 7.12 Archivos de cabeceras y código fuente para el codificador.

### 7.2.3 Compilando el proyecto.

Lo siguiente es compilar el proyecto y obtener un archivo ejecutable que pueda ser ejecutado por el DSP. Para ello nos vamos al menú Project → Build. Si todo va bien, la ventana de salida "Output Window" aparecerá en la parte inferior lo cual indica que la compilación se realizó con éxito y sin errores, tal como se muestra en la Fig. 7.13.



```
[lab1.c]
"c:\ti\c6000\cgtools\bin\cl6x" -@"Debug.lkf"
<Linking>
Build Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

Fig. 7.13 Compilación exitosa del proyecto.

Cuando el CCS construye el archivo ejecutable, los archivos fuente y de cabecera codificados en lenguaje C son convertidos a código ensamblador a través del compilador. Luego, éste código ensamblado se convierte en un archivo con formato de archivo de objeto (COFF) que contiene las instrucciones del programa organizados en módulos. Finalmente el enlazador organiza esos módulos y la biblioteca en tiempo de ejecución (rts6701.lib) en posiciones de memoria para crear un archivo ejecutable que pueda ser descargado a la memoria del DSP. Cuando este ejecutable es cargado al DSK, las instrucciones de programa ensambladas, las variables globales y las bibliotecas en tiempo de ejecución son cargadas en las posiciones de memoria especificadas por el compilador.

### 7.2.4 Cargando y Ejecutando el programa en el DSP

Para cargar el programa a la memoria del Dsp, primero se debe resetear el CPU, nos vamos a **Debug** → **Reset CPU**, una vez reseteado el CPU (CPU HALTED) se va al menú **File** → **Load Program** y se escoge el archivo lab1.out.

Una vez cargado el proyecto hay que asegurarse que un micrófono o la salida de cualquier dispositivo de audio (walkman, discman) estén conectados en el Plug IN del DSK y el Plug Out se conecta a unos altavoces.

**NOTA:** Si se va a usar un micrófono, éste debe ser del tipo auto alimentado, ya que el Codec del DSK no se comporta como una tarjeta de sonido de una PC (la cual envía un phantom voltage que alimenta al micrófono).

Con todo lo anterior listo, se va al menú **Debug** → **Run** y observar que en la parte inferior de la ventana se cambia el estado de “**CPU HALTED**” a “**CPU RUNNING**”

### 7.3 Mejoras y Optimización del código para óptimo desempeño en el DSP

El compilador, “builder”, “linker” y otras herramientas del Code Composer, ofrecen una serie de alternativas para depurar las aplicaciones y optimizarlas de acuerdo a nuestras necesidades. En todo proceso de optimización es necesario hacer un balance entre el tamaño del código, velocidad de ejecución, nivel de paralelismo, anular unas funciones, mantener otras e ir ajustando. En este caso se ha tratado de optimizar la velocidad de ejecución del código, y en particular, la optimización de las rutinas de búsquedas adaptiva y estocástica para un segmento completo (240 muestras) que son las que mas potencia computacional requieren. En el caso del código implementado, el proceso de optimización fue el siguiente:

#### 7.3.1 Estado Inicial (sin ningún tipo de optimización)

En el estado inicial, los ciclos de reloj que requerían las rutinas de búsquedas adaptiva y estocástica para un segmento completo se muestran en la Fig. 7.14. El valor de interés es el promedio que en este caso es de 469.49 ms. Tener en cuenta que el reloj es de 150 Mhz.

| Ciclos reloj    | ms                |
|-----------------|-------------------|
| 55202031        | 368,01354         |
| 63617428        | 424,116187        |
| 75002430        | 500,0162          |
| 67570447        | 450,469647        |
| 77281191        | 515,20794         |
| 70568671        | 470,457807        |
| 77322954        | 515,48636         |
| 69855019        | 465,700127        |
| 77402074        | 516,013827        |
| <b>Promedio</b> | <b>469,497959</b> |

Fig. 7.14 Tiempo empleado para búsquedas adaptiva y estocástica (inicial).

### 7.3.2 Reacomodo de funciones críticas a la IRAM

El reacomodo de las funciones críticas a la IRAM tiene como objetivo hacer que los segmentos de código que deben ser ejecutados con más rapidez se carguen a la memoria que es más rápida, en este caso es la IRAM (ver Fig. 7.3). Dicho reacomodo se logra con la directiva:

```
#pragma CODE_SECTION(funcion,seccion)
```

Luego del reacomodo de las funciones críticas a la IRAM, el tiempo de ejecución se muestra en la Fig. 7.15. Como se puede apreciar, hay una disminución sustancial del tiempo de ejecución.

| Ciclos reloj    | ms               |
|-----------------|------------------|
| 31849749        | 212,33166        |
| 37028055        | 246,8537         |
| 42815509        | 285,436727       |
| 39440005        | 262,933367       |
| 44290384        | 295,269227       |
| 40936315        | 272,908767       |
| 44319343        | 295,462287       |
| 40339689        | 268,93126        |
| 44363788        | 295,758587       |
| <b>Promedio</b> | <b>277,94424</b> |

Fig. 7.15 Tiempo empleado para búsquedas adaptiva y estocástica (reacomodo 1).

### 7.3.3 2da Reubicación de funciones críticas a la IRAM

Inspeccionando el código se llega a un segundo reacomodo del mismo, esta vez se retiraron funciones no críticas de la IRAM y se introdujeron otras funciones críticas. El resultado se muestra en la Fig. 7.16. Como se aprecia hay una ligera reducción en el tiempo de ejecución.

| Ciclos reloj    | ms                |
|-----------------|-------------------|
| 30663703        | 204,424687        |
| 35687120        | 237,914133        |
| 41186320        | 274,575467        |
| 38020646        | 253,470973        |
| 42574349        | 283,828993        |
| 39482111        | 263,214073        |
| 42604795        | 284,031967        |
| 38876767        | 259,178447        |
| 42646022        | 284,306813        |
| 38633006        | 257,553373        |
| 42624538        | 284,163587        |
| 38125757        | 254,171713        |
| <b>Promedio</b> | <b>266,946322</b> |

Fig. 7.16 Tiempo empleado para búsquedas adaptiva y estocástica (reacomodo 2).

### 7.3.4 Reubicación de variables críticas a la IRAM

El siguiente ajuste es el direccionamiento de las variables que son usadas por las funciones críticas, a la memoria IRAM, por medio de la directiva

```
#pragma DATA_SECTION (variable, seccion);
```

Por ejemplo para la rutina del cálculo de la ganancia en la búsqueda del codebook adaptivo se emplean las siguientes directivas:

```
#pragma DATA_SECTION (y2_pgain, "mydata");
#pragma DATA_SECTION (y_pgain, "mydata");
#pragma DATA_SECTION (pgain1_pgain, "mydata");
#pragma DATA_SECTION (cor1_pgain, "mydata");
#pragma DATA_SECTION (eng_pgain, "mydata");
float cor1_pgain, eng_pgain;
float y2_pgain[MAXLP], pgain1_pgain;
float y_pgain[MAXLP];
```

Luego de la reubicación de las variables críticas a la IRAM se obtiene, para la ejecución de una búsqueda adaptiva más estocástica, un promedio de **190 ms**.

### 7.3.5 Modificación para optimización de rutinas ganancia\_estocástica y ganancia\_adaptiva. Optimización de la convolución recursiva.

El siguiente ajuste consiste en optimizar las convoluciones recursivas tal como se explicó en las secciones 6.13.3 y 6.13.4. Esto se hace para aprovechar al máximo la forma del codebook estocástico. Como sus valores son ceros, unos o menos unos, se optimiza la convolución recursiva para evitar que tenga multiplicaciones y todo se maneja al nivel de sumas y restas.

Luego de optimizar esta etapa, se obtiene el valor para la ejecución de una búsqueda adaptiva y estocástica: **169 ms**.

Cuando se comparan los valores de **169 ms** (ultimo ajuste) con el inicial (Fig, 7.14), en total se ha conseguido bajar el tiempo de ejecución de la rutinas de búsqueda adaptiva y estocástica combinadas de **469 ms** al inicio a **169 ms** al final. Se vé que la mejora global es sustancial, pero aún se puede optimizar más, tal como se muestra en la siguiente sección.

### 7.3.6 Opciones del compilador para optimización

Para todas las optimizaciones anteriores, no se emplearon las opciones de optimización del compilador. Dichas opciones se describen en la presente sección y al final se presentan los resultados que resultan luego de aplicar dichas opciones de optimización. Para mas información sobre optimización del código, referirse al documento spru 187: "TMS320C6000 Optimizing Compiler user's Guide". Las opciones de compilación se muestran en las figuras Fig. 7.17 y Fig. 7.18.

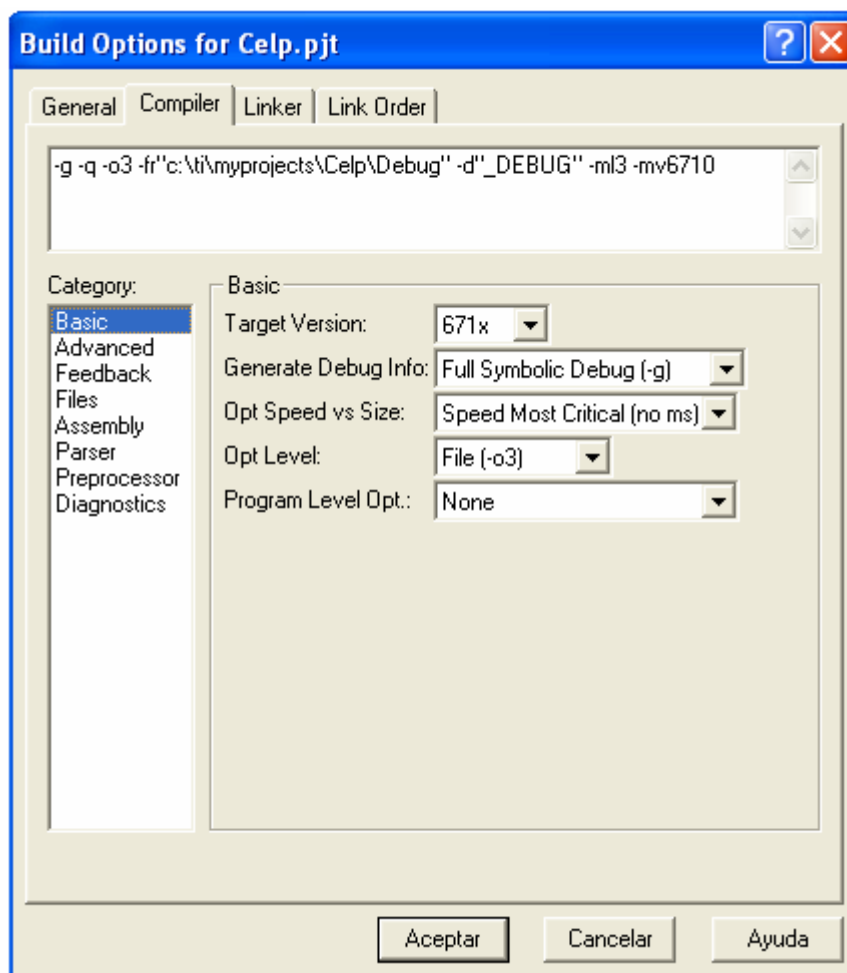


Fig. 7.17 Opciones de compilación. Básicas.



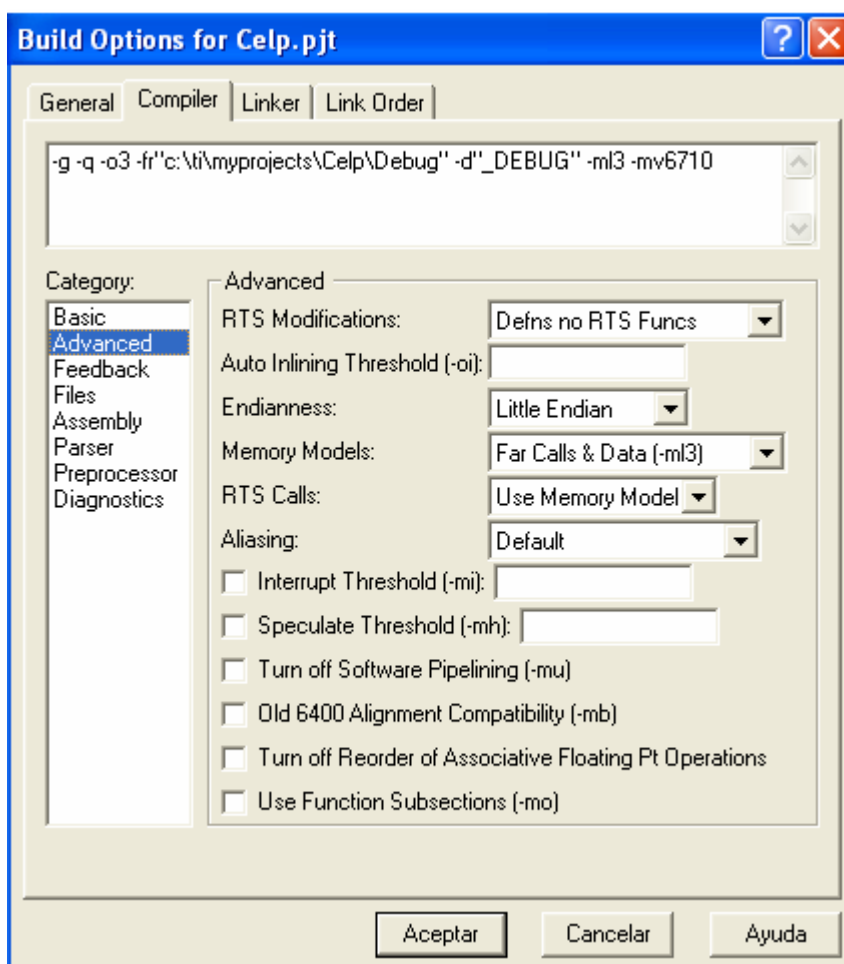


Fig. 7.18 Opciones de compilación. Avanzadas

Las opciones de compilación empleadas fueron las siguientes:

- **Opt Speed vs Size.** Se configuró a “*Speed Most Critical (no ms)*”. Con esta configuración se orientó la prioridad hacia la velocidad de ejecución sacrificando el tamaño del código. De esta forma el compilador optimiza el código en velocidad.
- **Opt Level.** Se configuró a “**-o3**”. En esta configuración el optimizador ejecuta lo siguiente:

Ejecuta simplificación control-flow-graph, Ubica variables en registros, Ejecuta rotación en los bucles, Elimina código no usado, Simplifica expresiones y expresiones, Ejecuta propagación local, Retira asignaciones no usadas, Elimina expresiones locales comunes, Ejecuta paralelismo en software, Ejecuta optimización de bucles, Elimina subexpresiones globales comunes, Ejecuta unrolling en los bucles, Retira todas las funciones que nunca son llamadas, Reordena las declaraciones de funciones de tal forma

que los atributos de las funciones a llamadas son conocidas cuando el llamante es optimizado, etc.

- **Generate Debug Info.** Configurado en “none”. Normalmente cuando se desarrolla un software se necesita depurar el mismo. Para ello el Code Composer inserta múltiples etiquetas y comandos adicionales al código de la aplicación para lograr la depuración. Para evaluar la performance máxima del código se desactiva el debugger con la opción “none”.
- **Memory Models.** Configurado en “Far Calls & Data”. Cuando se va a implementar un programa grande y son muchas las variables, arreglos y matrices que se van a emplear, es de esperar que la memoria IRAM no alcanzará para ingresar dichos códigos y variables. Para ello se tiene que recurrir al modelo de memoria de llamadas lejanas a funciones y datos. De esta forma se puede aprovechar todo el mapa de memoria. La desventaja es que se añaden cabeceras a las instrucciones los cuales consumen más ciclos de reloj.

Luego de aplicar las opciones de optimización mostradas, se llega al valor de **101.01 ms** en promedio para el tiempo de procesamiento de un segmento completo el cual, tal como se ha descrito en el capítulo 6 incluye: correlaciones, calculo de los coeficientes  $I_{pc}$ ,  $I_{sp}$ , cuantización, interpolaciones y búsquedas adaptivas y estocásticas, etc.

Como se puede apreciar, hay una mejora sustancial con relación al último valor obtenido el cual fue de **169 ms**. Con esto se ha demostrado que el proceso de optimización es un proceso de refinamiento de código. Es obvio que se puede seguir optimizando aún más para mejorar el tiempo de ejecución. Si se quiere hacer una reducción aún mayor en el tiempo de ejecución, lo siguiente es pasar a assembler las rutinas críticas.

|  | Codebook 512 ( No Optimizado) | Codebook 512 ( Sin optimizaciones del Compilador) | Codebook 512 (Todas las optimizaciones) | Codebook 64 |
|--|-------------------------------|---|---|-------------|
| Procesamiento de un segmento completo (4 búsquedas adaptiva + estocastica + otras funciones) | 469.49 ms                     | 169 ms  | 101.01 ms                               | 82.05 ms    |

Fig. 7.19 Optimización del tiempo de ejecución

La Fig. 7.19 resume el proceso de optimización del código en tiempo de ejecución. Se puede seguir optimizando aún más pero eso requiere rehacer las rutinas críticas en assembler, lo cual es muy costoso para el presente trabajo de tesis.

En el caso del decodificador, el tiempo promedio requerido para decodificar un segmento de código, es decir para recuperar los 240 valores a partir del stream de 144 bits es de: **23.80 ms**.

## CAPITULO VIII

### PRUEBA DE CALIDAD DEL CODIFICADOR IMPLEMENTADO

En el presente capítulo se presentan los resultados de las pruebas subjetivas del codificador implementado evaluando la voz sintética decodificada que se obtiene del codificado, realizado por un grupo de personas. También se presentan pruebas para las mismas secuencias de voz pero ahora variando la BER. Finalmente se exponen las conclusiones del presente trabajo de tesis.

#### 8.1 Test MOS

El test MOS consiste en una evaluación subjetiva de la calidad de síntesis de voz de un sistema. Fue normalizado por el comité Consultivo Internacional de Telefonía y Telegrafía (CCITT, hoy UIT) a principios de los años 80 y se ha utilizado principalmente para medir la calidad en sistemas de comunicación celular digital.

El test consiste en realizar una encuesta de opinión a un conjunto de individuos de prueba los cuales deben evaluar una grabación de voz según la Tabla 8.1.

| <b>NOTA</b> | <b>CALIDAD</b>   | <b>ESFUERZO DE ESCUCHA</b>                                       | <b>DEGRADACION</b>             |
|-------------|------------------|--|--------------------------------|
| <b>5</b>    | <b>Excelente</b> | <b>Posible relajación completa, no requiere ningún esfuerzo.</b> | <b>Inaudible</b>               |
| <b>4</b>    | <b>Buena</b>     | <b>Atención necesaria, no se requiere esfuerzo apreciable</b>    | <b>Audible pero no molesta</b> |
| <b>3</b>    | <b>Aceptable</b> | <b>Se necesita esfuerzo moderado</b>                             | <b>Ligeramente molesta</b>     |
| <b>2</b>    | <b>Mediocre</b>  | <b>Se necesita esfuerzo considerable</b>                         | <b>Molesta</b>                 |
| <b>1</b>    | <b>Mala</b>      | <b>Cualquier esfuerzo no permite comprender</b>                  | <b>Muy Molesta</b>             |

Tabla 8.1 Nivel de calificación en el test MOS

En el presente trabajo, se escogió un universo de 30 individuos (no especialistas en voz) a quienes se les hizo escuchar 5 conjuntos de secuencias de voz diferentes. Cada conjunto estaba compuesto por una secuencia original, una celp decodificada con un codebook de 64, una celp decodificada con un codebook de 512.

Las 5 secuencias originales son archivos wav de 16 bits a 8 KHz y son las siguientes:

**Secuencia “estamos.wav”**. Se muestra en la Fig. 8.1.

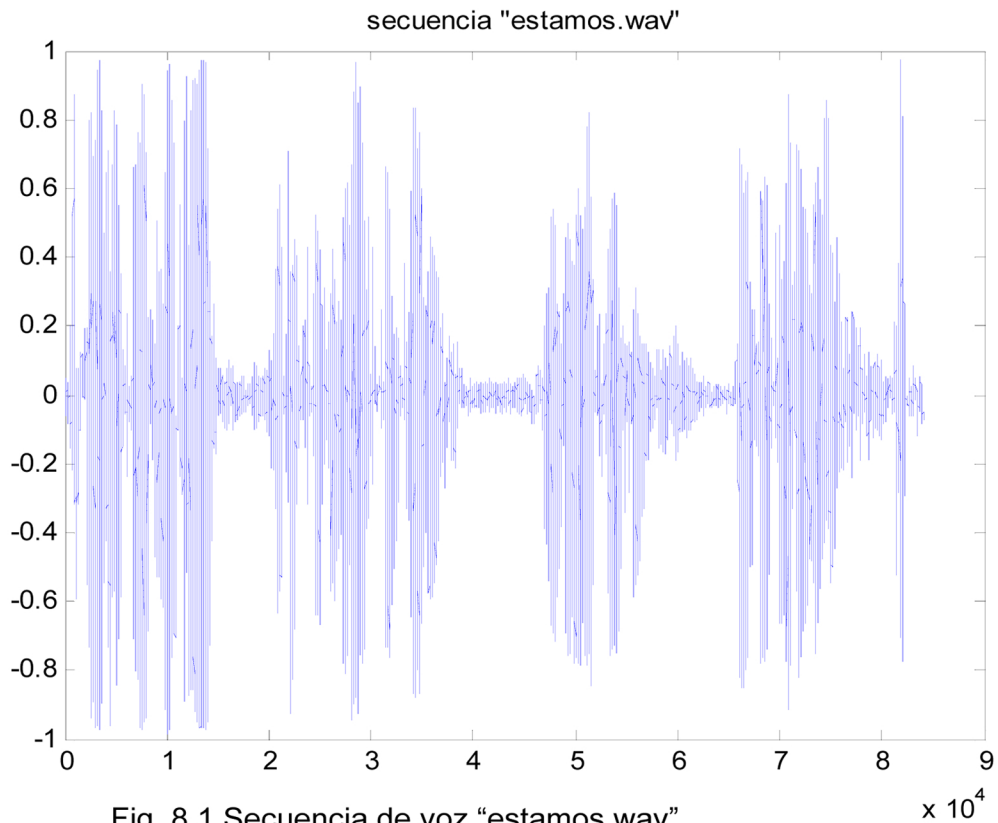


Fig. 8.1 Secuencia de voz “estamos.wav”

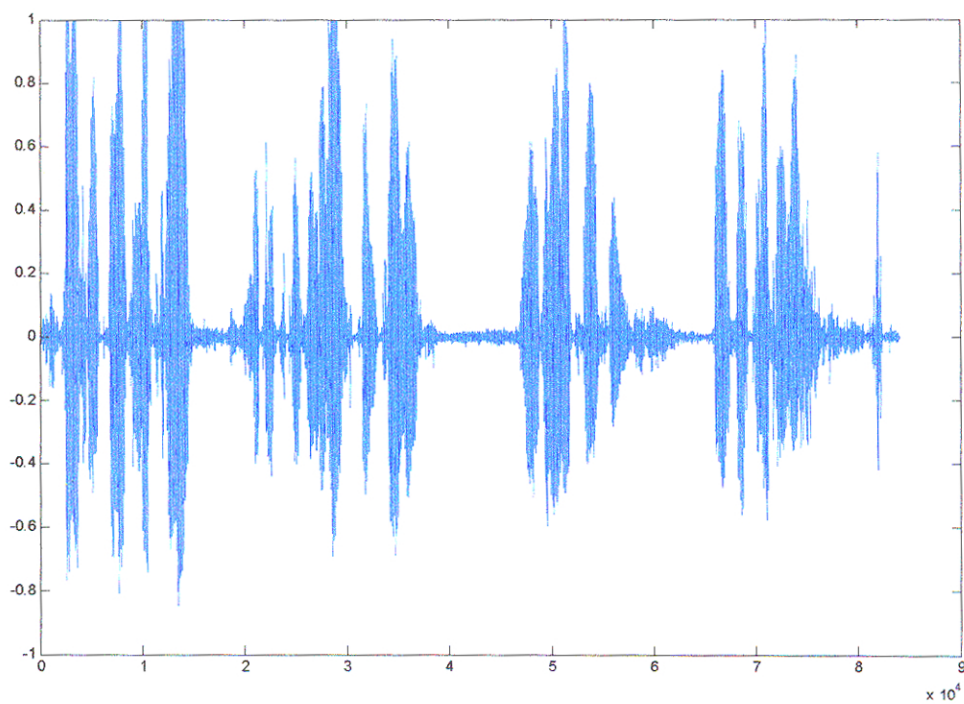


Fig. 8.2 Secuencia de voz sintética "estamos.wav", usando un *codebook* de tamaño 64

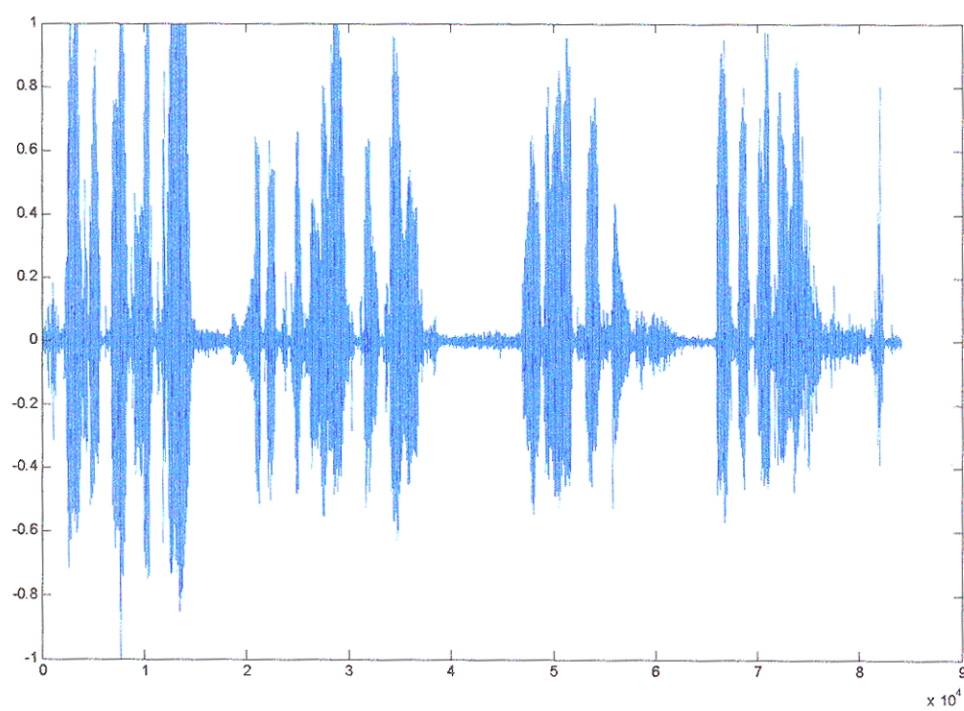


Fig. 8.3 Secuencia de voz sintética "estamos.wav", usando un *codebook* de tamaño 512

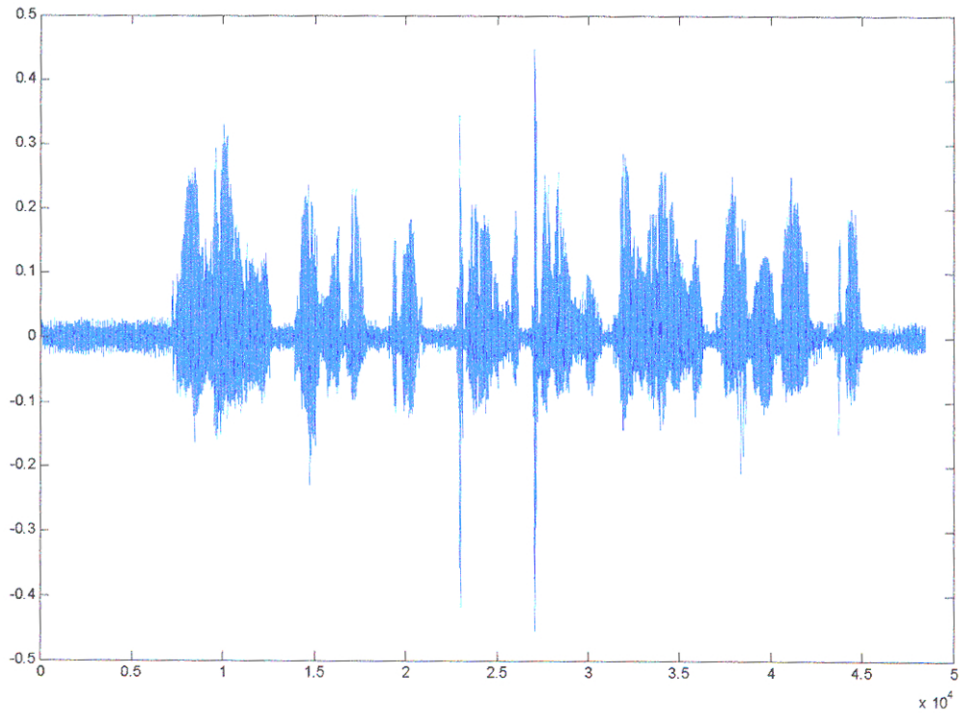


Fig. 8.6 Secuencia de voz sintética "hola\_hola.wav", usando un codebook de tamaño 512

**Secuencia "vocales.wav"**. Se muestra en la Fig. 8.7. Esta secuencia tiene picos altos (volumen alto) barriendo los rangos máximo y mínimo. Esto pone a prueba la performance ante el clipping de los codificadores.

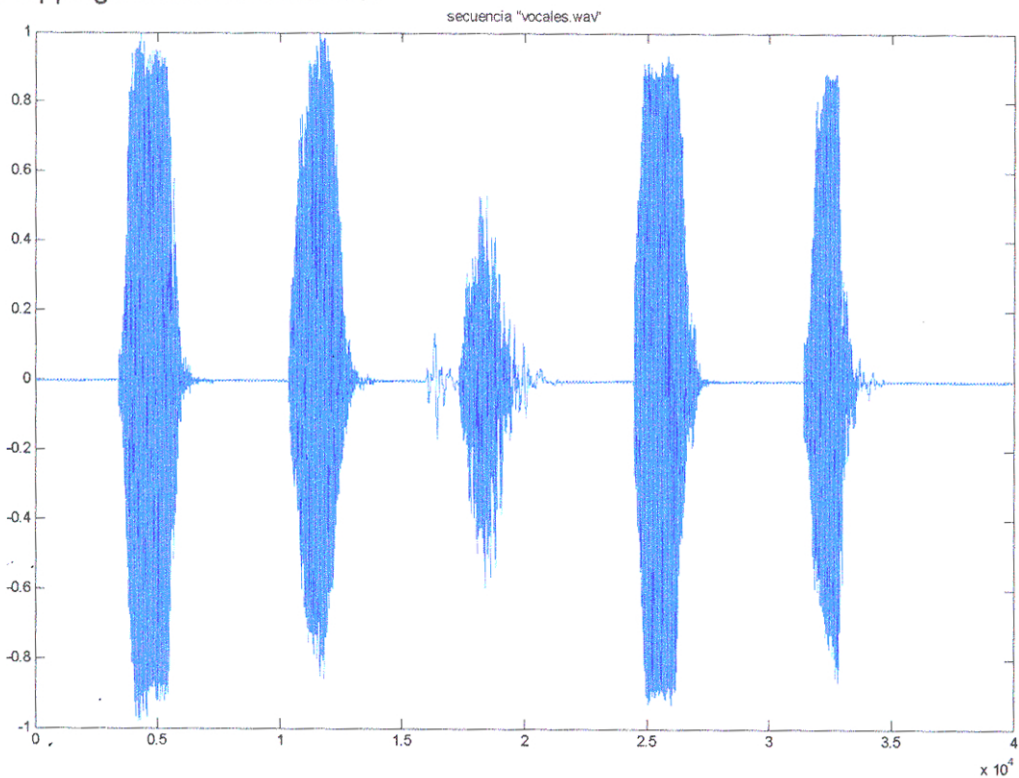


Fig. 8.7 Secuencia de voz original "vocales.wav"

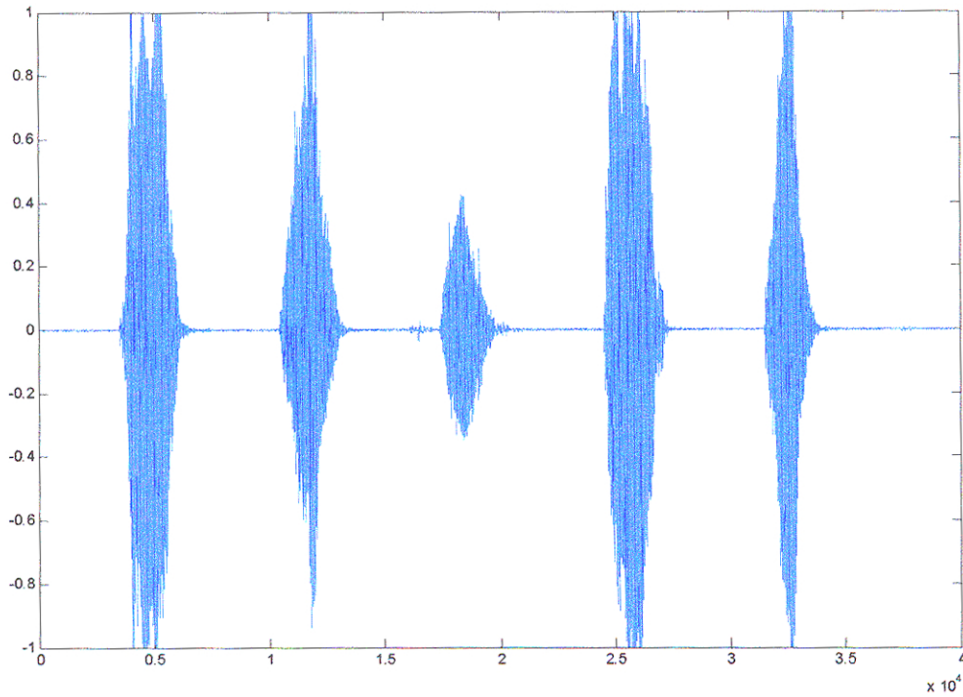


Fig. 8.8 Secuencia de voz sintética "vocales.wav", usando un *codebook* de tamaño 64

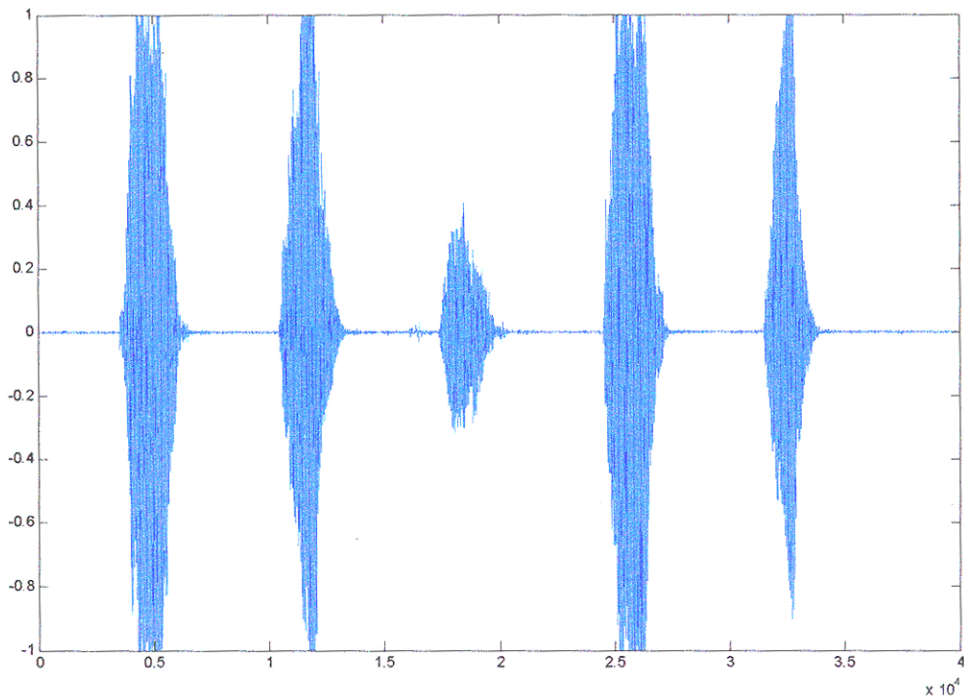


Fig. 8.9 Secuencia de voz sintética "vocales.wav", usando un *codebook* de tamaño 512



Secuencia "ingles2\_original.wav". Se muestra en la Fig. 8.4

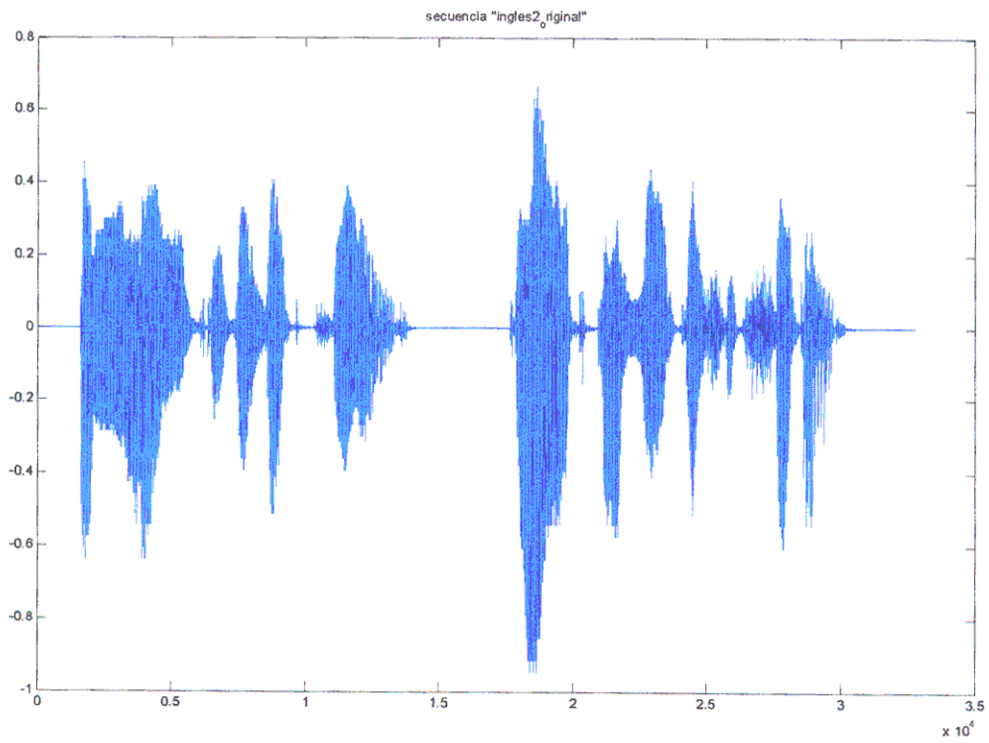


Fig. 8.10 Secuencia de voz original "ingles2\_original.wav"

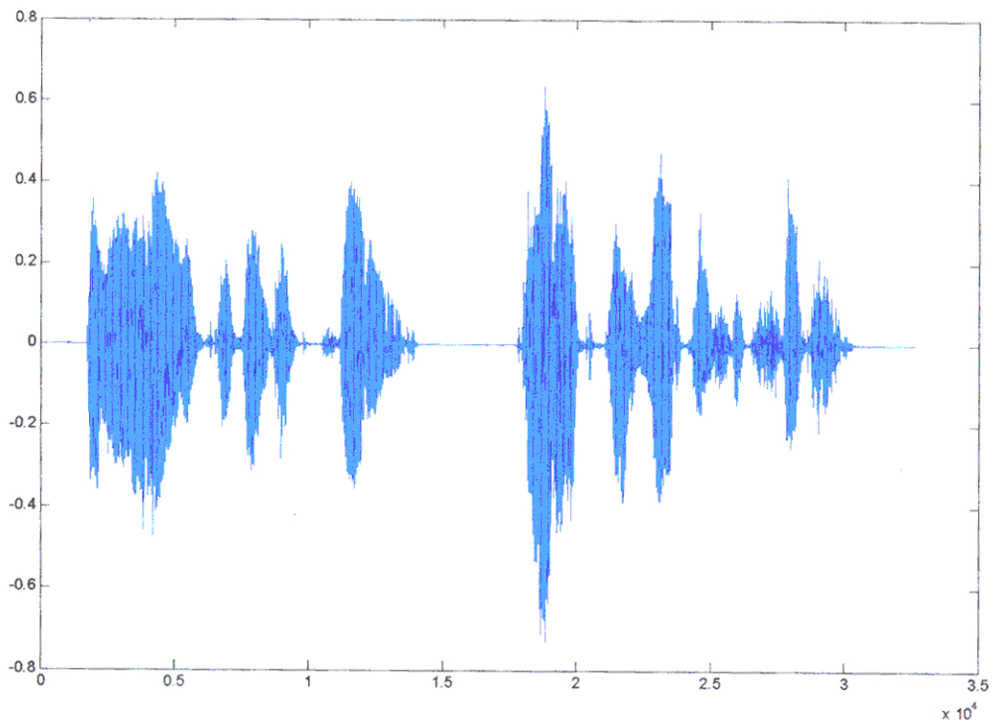


Fig. 8.11 Secuencia de voz sintética "ingles2\_original.wav", con un *codebook* de tamaño 64

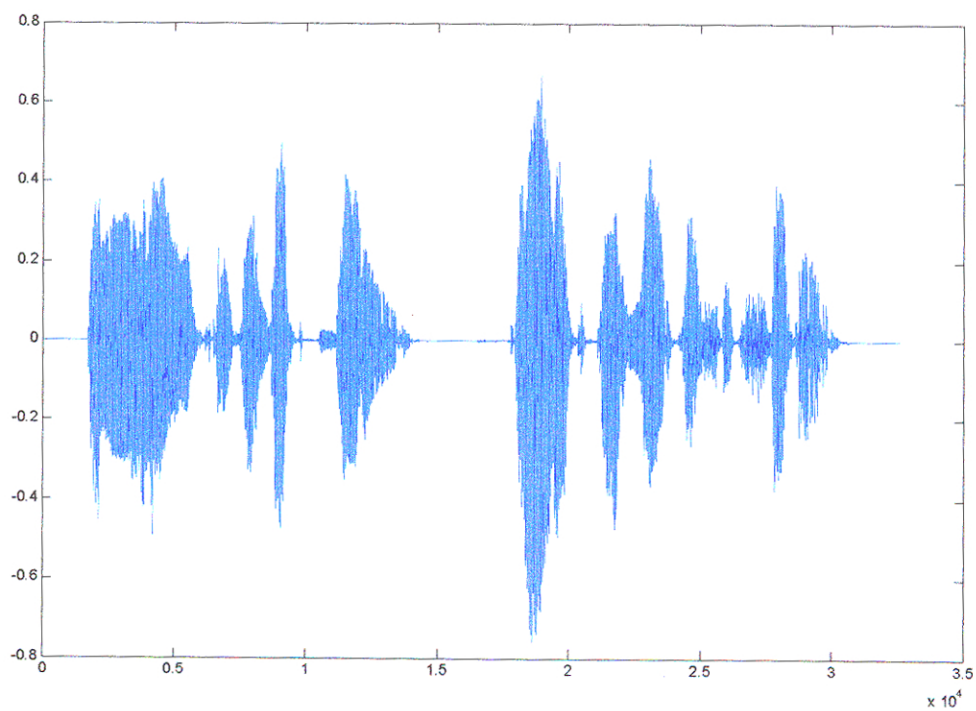


Fig. 8.12 Secuencia de voz sintética "ingles2\_original.wav", con un *codebook* de tamaño 512

Secuencia "ingles\_original.wav". Se muestra en la Fig. 8.13.

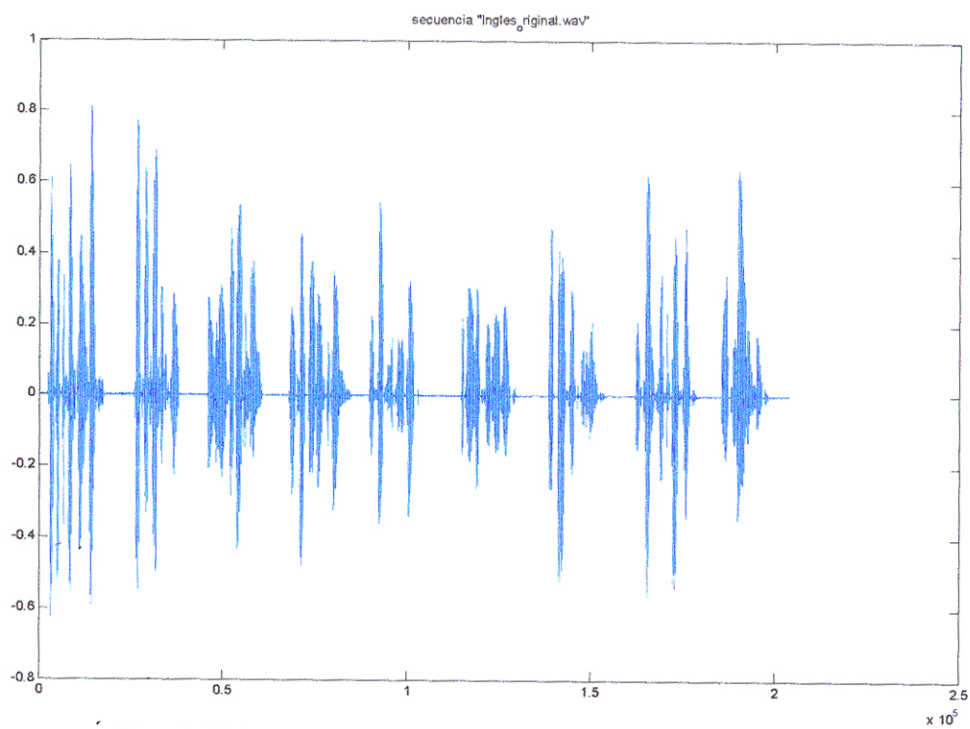


Fig. 8.13 Secuencia de voz original "ingles\_original.wav"

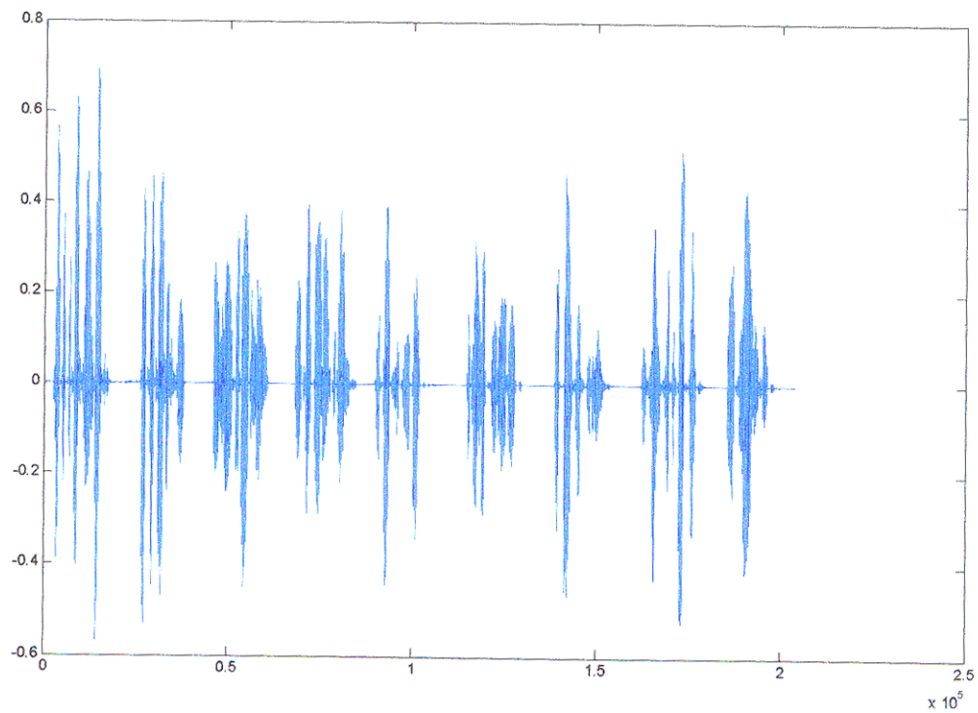


Fig. 8.14 Secuencia de voz sintética "ingles\_original.wav", con un *codebook* de tamaño 64

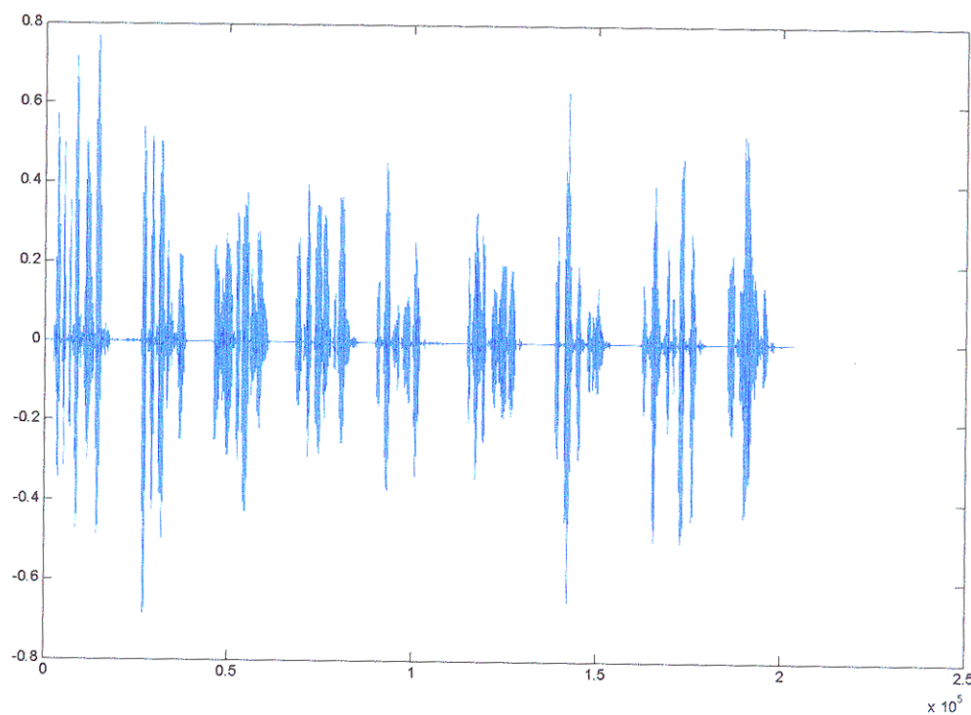


Fig. 8.15 Secuencia de voz sintética "ingles\_original.wav", con un *codebook* de tamaño 512

Las dos últimas secuencias son secuencias limpias, es decir, fueron grabadas en óptimas condiciones (esto se observa en las formas de onda). Estas fueron obtenidas del internet. En la secuencia “ingles2\_original.wav” tenemos la voz aguda de una mujer. En la secuencia “ingles\_original.wav” tenemos un conjunto de hablantes tanto hombres, como mujeres, con diferentes tonalidades.

## 8.2 Resultados del Test MOS

Los resultados de las pruebas se muestran a continuación. Cada individuo calificó 5 secuencias de voz sintéticas decodificadas (celp cb 512, celp cb 64) comparándolas con la secuencia original (nota 5).

| Secuencia     | Individuo       | Tipo de secuencia |             |
|---------------|-----------------|-------------------|-------------|
|               |                 | celp cb 512       | celp cb 64  |
| "estamos.wav" | 1               | 4,50              | 4,00        |
|               | 2               | 4,00              | 3,70        |
|               | 3               | 4,20              | 4,30        |
|               | 4               | 4,20              | 4,00        |
|               | 5               | 4,00              | 4,00        |
|               | 6               | 4,50              | 3,80        |
|               | 7               | 4,30              | 3,80        |
|               | 8               | 4,00              | 4,00        |
|               | 9               | 4,00              | 3,90        |
|               | 10              | 4,00              | 4,00        |
|               | 11              | 3,90              | 4,00        |
|               | 12              | 4,20              | 3,80        |
|               | 13              | 4,50              | 4,20        |
|               | 14              | 4,50              | 4,20        |
|               | 15              | 4,00              | 3,50        |
|               | 16              | 3,80              | 4,10        |
|               | 17              | 4,50              | 4,00        |
|               | 18              | 4,00              | 3,50        |
|               | 19              | 4,20              | 4,00        |
|               | 20              | 4,20              | 3,90        |
|               | 21              | 4,50              | 4,00        |
|               | 22              | 4,00              | 3,80        |
|               | 23              | 4,20              | 3,80        |
|               | 24              | 4,00              | 4,00        |
|               | 25              | 4,00              | 3,80        |
|               | 26              | 4,50              | 4,40        |
|               | 27              | 4,20              | 4,00        |
|               | 28              | 4,40              | 4,00        |
|               | 29              | 4,00              | 4,10        |
|               | 30              | 4,30              | 4,20        |
|               | <b>Promedio</b> | <b>4,18666667</b> | <b>3,96</b> |

Tabla 8.2. Test MOS para la secuencia “estamos.wav”

| Secuencia       | Individuo | Tipo de secuencia |             |
|-----------------|-----------|-------------------|-------------|
|                 |           | celp cb 512       | celp cb 64  |
| "hola_hola.wav" | 1         | 4,20              | 4,00        |
|                 | 2         | 4,20              | 3,00        |
|                 | 3         | 4,40              | 4,60        |
|                 | 4         | 4,50              | 4,20        |
|                 | 5         | 4,50              | 4,20        |
|                 | 6         | 4,40              | 4,00        |
|                 | 7         | 4,30              | 4,10        |
|                 | 8         | 4,00              | 4,40        |
|                 | 9         | 4,20              | 4,10        |
|                 | 10        | 4,50              | 4,20        |
|                 | 11        | 4,10              | 4,40        |
|                 | 12        | 4,20              | 3,80        |
|                 | 13        | 4,60              | 4,20        |
|                 | 14        | 4,40              | 4,00        |
|                 | 15        | 4,60              | 4,50        |
|                 | 16        | 4,50              | 4,00        |
|                 | 17        | 4,30              | 4,20        |
|                 | 18        | 4,20              | 3,80        |
|                 | 19        | 4,20              | 4,00        |
|                 | 20        | 4,40              | 4,20        |
|                 | 21        | 4,20              | 4,00        |
|                 | 22        | 4,50              | 4,30        |
|                 | 23        | 4,50              | 4,00        |
|                 | 24        | 4,00              | 4,00        |
|                 | 25        | 4,00              | 4,10        |
|                 | 26        | 4,50              | 4,00        |
|                 | 27        | 4,40              | 4,20        |
|                 | 28        | 4,20              | 3,80        |
|                 | 29        | 4,50              | 3,80        |
|                 | 30        | 4,40              | 4,00        |
| <b>Promedio</b> |           | <b>4,33</b>       | <b>4,07</b> |

Tabla 8.3 Test MOS para la secuencia "hola\_hola.wav"

| Secuencia       | Individuo | Tipo de secuencia |                   |
|-----------------|-----------|-------------------|-------------------|
|                 |           | celp cb 512       | celp cb 64        |
| "vocales.wav"   | 1         | 4,20              | 4,00              |
|                 | 2         | 5,00              | 5,00              |
|                 | 3         | 4,40              | 4,70              |
|                 | 4         | 4,20              | 4,00              |
|                 | 5         | 4,60              | 4,40              |
|                 | 6         | 4,60              | 4,50              |
|                 | 7         | 5,00              | 4,50              |
|                 | 8         | 4,20              | 4,70              |
|                 | 9         | 4,40              | 4,40              |
|                 | 10        | 4,10              | 4,50              |
|                 | 11        | 5,00              | 5,00              |
|                 | 12        | 4,50              | 4,00              |
|                 | 13        | 5,00              | 5,00              |
|                 | 14        | 4,80              | 5,00              |
|                 | 15        | 4,50              | 4,40              |
|                 | 16        | 5,00              | 4,40              |
|                 | 17        | 4,60              | 5,00              |
|                 | 18        | 4,50              | 4,40              |
|                 | 19        | 4,20              | 4,00              |
|                 | 20        | 4,70              | 4,50              |
|                 | 21        | 4,60              | 4,10              |
|                 | 22        | 4,60              | 4,60              |
|                 | 23        | 5,00              | 4,50              |
|                 | 24        | 4,70              | 4,50              |
|                 | 25        | 4,00              | 4,20              |
|                 | 26        | 4,60              | 4,30              |
|                 | 27        | 4,90              | 4,70              |
|                 | 28        | 4,40              | 4,30              |
|                 | 29        | 4,60              | 4,70              |
|                 | 30        | 4,20              | 4,20              |
| <b>Promedio</b> |           | <b>4,57</b>       | <b>4,48333333</b> |

Tabla 8.4 Test MOS para la secuencia "vocales.wav"

| Secuencia     | Individuo       | Tipo de secuencia |                   |
|---------------|-----------------|-------------------|-------------------|
|               |                 | celp cb 512       | celp cb 64        |
| "ingles2.wav" | 1               | 4,50              | 4,20              |
|               | 2               | 5,00              | 4,00              |
|               | 3               | 4,50              | 4,70              |
|               | 4               | 4,20              | 4,00              |
|               | 5               | 4,60              | 4,40              |
|               | 6               | 4,50              | 4,00              |
|               | 7               | 4,50              | 3,90              |
|               | 8               | 5,00              | 4,80              |
|               | 9               | 5,00              | 4,60              |
|               | 10              | 4,80              | 3,90              |
|               | 11              | 5,00              | 4,50              |
|               | 12              | 4,50              | 4,00              |
|               | 13              | 5,00              | 4,00              |
|               | 14              | 4,60              | 3,90              |
|               | 15              | 4,60              | 4,20              |
|               | 16              | 4,70              | 4,00              |
|               | 17              | 4,60              | 3,80              |
|               | 18              | 4,80              | 4,00              |
|               | 19              | 4,60              | 4,20              |
|               | 20              | 5,00              | 4,80              |
|               | 21              | 4,00              | 4,10              |
|               | 22              | 4,10              | 4,40              |
|               | 23              | 4,60              | 4,00              |
|               | 24              | 4,20              | 4,00              |
|               | 25              | 4,20              | 4,20              |
|               | 26              | 4,60              | 4,20              |
|               | 27              | 4,20              | 4,00              |
|               | 28              | 4,80              | 4,80              |
|               | 29              | 5,00              | 4,40              |
|               | 30              | 5,00              | 4,50              |
|               | <b>Promedio</b> | <b>4,62333333</b> | <b>4,21666667</b> |

Tabla 8 **5** Test MOS para la secuencia "ingles2.wav"

| Secuencia     | Individuo       | Tipo de secuencia |             |
|---------------|-----------------|-------------------|-------------|
|               |                 | celp cb 512       | celp cb 64  |
| "ingles1.wav" | 1               | 4,50              | 4,20        |
|               | 2               | 4,30              | 3,80        |
|               | 3               | 4,30              | 4,50        |
|               | 4               | 4,20              | 4,00        |
|               | 5               | 4,10              | 4,50        |
|               | 6               | 4,60              | 4,40        |
|               | 7               | 4,50              | 4,00        |
|               | 8               | 4,20              | 4,20        |
|               | 9               | 4,60              | 4,40        |
|               | 10              | 4,60              | 4,00        |
|               | 11              | 4,80              | 4,50        |
|               | 12              | 4,60              | 3,90        |
|               | 13              | 4,00              | 4,00        |
|               | 14              | 5,00              | 4,30        |
|               | 15              | 4,50              | 4,10        |
|               | 16              | 4,00              | 4,10        |
|               | 17              | 4,50              | 3,90        |
|               | 18              | 4,60              | 4,00        |
|               | 19              | 4,40              | 4,00        |
|               | 20              | 4,00              | 4,40        |
|               | 21              | 4,70              | 4,20        |
|               | 22              | 4,20              | 4,50        |
|               | 23              | 4,50              | 3,60        |
|               | 24              | 5,00              | 4,50        |
|               | 25              | 4,00              | 4,00        |
|               | 26              | 4,30              | 4,00        |
|               | 27              | 4,20              | 3,70        |
|               | 28              | 4,50              | 4,10        |
|               | 29              | 4,20              | 4,40        |
|               | 30              | 4,00              | 4,00        |
|               | <b>Promedio</b> | <b>4,39666667</b> | <b>4,14</b> |

Tabla 8.6. Test MOS para la secuencia "ingles1.wav"



### 8.3 Resultados del Test MOS con variaciones del BER

A continuación se presentan los resultados del Test MOS para las mismas secuencias presentadas anteriormente pero esta vez se modifica el BER desde el rango  $1 \times 10^{-3}$  al rango  $4 \times 10^{-2}$ . Esto evalúa la eficacia de la etapa FEC implementada en el codificador. En todos estos casos la voz corresponde a la voz decodificada con un CodeBook de tamaño 512. Ya que las pruebas fueron realizadas mucho después que las anteriores, solo se pudo probar con un universo de 10 individuos.

|               |           | BER (Bit Error Rate aplicada a la secuencia) |                    |                    |                    |                    |                    |                    |
|---------------|-----------|--|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Secuencia     | Individuo | $1 \times 10^{-3}$                           | $5 \times 10^{-3}$ | $8 \times 10^{-3}$ | $1 \times 10^{-2}$ | $2 \times 10^{-2}$ | $3 \times 10^{-2}$ | $4 \times 10^{-2}$ |
| "estamos.wav" | 1         | 4.20   | 4.00               | 4.00               | 3.00               | 3.00               | 2.20               | 2.00               |
|               | 2         | 4.00   | 4.20               | 4.00               | 3.20               | 2.80               | 2.10               | 1.00               |
|               | 3         | 4.00   | 4.50               | 4.20               | 3.10               | 2.50               | 2.00               | 2.20               |
|               | 4         | 4.50   | 4.00               | 4.20               | 2.80               | 2.80               | 2.00               | 2.50               |
|               | 5         | 4.50   | 4.00               | 4.00               | 3.60               | 2.50               | 2.20               | 1.50               |
|               | 6         | 4.20   | 3.80               | 4.20               | 3.00               | 3.00               | 1.80               | 2.00               |
|               | 7         | 4.50   | 3.80               | 4.30               | 3.40               | 2.80               | 2.20               | 2.00               |
|               | 8         | 5.00   | 4.00               | 4.00               | 3.00               | 3.50               | 2.40               | 2.40               |
|               | 9         | 4.50   | 3.90               | 4.10               | 3.00               | 3.20               | 2.60               | 2.10               |
|               | 10        | 4.20   | 4.00               | 4.20               | 3.20               | 3.80               | 2.00               | 2.00               |
|               |           | 4.36   | 4.02               | 4.12               | 3.13               | 2.99               | 2.15               | 1.97               |

Tabla 8.7. Test MOS con BER variable para la secuencia "estamos.wav"

|                 |           | BER (Bit Error Rate aplicada a la secuencia) |                    |                    |                    |                    |                    |                    |
|-----------------|-----------|--|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Secuencia       | Individuo | $1 \times 10^{-3}$                           | $5 \times 10^{-3}$ | $8 \times 10^{-3}$ | $1 \times 10^{-2}$ | $2 \times 10^{-2}$ | $3 \times 10^{-2}$ | $4 \times 10^{-2}$ |
| "hola_hola.wav" | 1         | 4.00   | 4.00               | 4.00               | 3.20               | 2.50               | 2.40               | 2.20               |
|                 | 2         | 4.20   | 3.90               | 4.30               | 3.40               | 2.50               | 2.00               | 1.20               |
|                 | 3         | 4.20   | 3.60               | 4.20               | 3.40               | 2.50               | 2.50               | 2.00               |
|                 | 4         | 4.50   | 3.50               | 4.00               | 3.00               | 2.80               | 2.00               | 2.50               |
|                 | 5         | 4.50   | 4.20               | 3.50               | 3.20               | 3.00               | 2.00               | 2.00               |
|                 | 6         | 4.80   | 4.00               | 4.00               | 3.00               | 3.00               | 2.00               | 2.00               |
|                 | 7         | 4.50   | 4.00               | 4.00               | 3.20               | 3.00               | 2.20               | 1.00               |
|                 | 8         | 4.40   | 4.40               | 3.80               | 3.50               | 3.20               | 2.00               | 2.00               |
|                 | 9         | 4.50   | 3.80               | 4.00               | 2.80               | 3.20               | 2.40               | 1.50               |
|                 | 10        | 4.20   | 4.50               | 3.80               | 3.60               | 3.50               | 2.20               | 1.50               |
|                 |           | 4.38   | 3.99               | 3.96               | 3.23               | 2.92               | 2.17               | 1.79               |

Tabla 8.8 Test MOS con BER variable para la secuencia "hola\_hola.wav"

|               |           | BER (Bit Error Rate aplicada a la secuencia) |                    |                    |                    |                    |                    |                    |
|---------------|-----------|--|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Secuencia     | Individuo | $1 \times 10^{-3}$                           | $5 \times 10^{-3}$ | $8 \times 10^{-3}$ | $1 \times 10^{-2}$ | $2 \times 10^{-2}$ | $3 \times 10^{-2}$ | $4 \times 10^{-2}$ |
| "vocales.wav" | 1         | 4.20   | 4.40               | 4.10               | 2.50               | 2.80               | 2.30               | 1.80               |
|               | 2         | 5.00   | 4.00               | 4.20               | 3.00               | 3.20               | 2.00               | 1.20               |
|               | 3         | 4.40   | 4.00               | 4.00               | 2.80               | 3.40               | 2.20               | 2.20               |
|               | 4         | 4.20   | 4.30               | 3.80               | 3.20               | 3.30               | 2.50               | 1.60               |
|               | 5         | 4.60   | 4.00               | 3.80               | 3.50               | 3.50               | 1.80               | 2.00               |
|               | 6         | 4.60   | 4.00               | 4.20               | 3.00               | 2.60               | 2.00               | 1.80               |
|               | 7         | 4.50   | 4.20               | 4.00               | 3.50               | 2.80               | 2.00               | 1.20               |
|               | 8         | 4.20   | 4.00               | 4.00               | 3.40               | 2.50               | 2.20               | 2.00               |
|               | 9         | 4.40   | 4.40               | 4.20               | 3.00               | 2.80               | 2.40               | 2.20               |
|               | 10        | 4.10   | 4.50               | 3.80               | 3.20               | 3.20               | 2.40               | 2.50               |
|               |           | 4.42   | 4.18               | 4.01               | 3.11               | 3.01               | 2.18               | 1.85               |

Tabla 8.9. Test MOS con BER variable para la secuencia "vocales.wav"

|               |           | BER (Bit Error Rate aplicada a la secuencia) |                    |                    |                    |                    |                    |                    |
|---------------|-----------|--|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Secuencia     | Individuo | $1 \times 10^{-3}$                           | $5 \times 10^{-3}$ | $8 \times 10^{-3}$ | $1 \times 10^{-2}$ | $2 \times 10^{-2}$ | $3 \times 10^{-2}$ | $4 \times 10^{-2}$ |
| "ingles2.wav" | 1         | 4.20   | 4.20               | 4.00               | 3.20               | 2.80               | 2.20               | 1.70               |
|               | 2         | 4.60   | 4.20               | 4.20               | 3.40               | 2.80               | 2.00               | 1.00               |
|               | 3         | 4.50   | 4.40               | 4.20               | 3.50               | 2.60               | 2.00               | 1.00               |
|               | 4         | 5.00   | 4.20               | 4.40               | 2.50               | 2.50               | 2.30               | 1.40               |
|               | 5         | 4.50   | 4.00               | 3.80               | 3.00               | 3.00               | 2.60               | 1.50               |
|               | 6         | 4.60   | 4.40               | 4.00               | 3.00               | 3.00               | 2.30               | 1.40               |
|               | 7         | 4.70   | 4.00               | 4.00               | 3.00               | 3.40               | 2.00               | 1.80               |
|               | 8         | 5.00   | 4.20               | 4.00               | 2.80               | 3.40               | 1.80               | 2.00               |
|               | 9         | 4.20   | 4.50               | 4.00               | 2.50               | 2.80               | 2.50               | 2.00               |
|               | 10        | 5.00   | 4.50               | 4.20               | 2.80               | 2.60               | 1.50               | 1.50               |
|               |           | 4.63   | 4.26               | 4.08               | 2.97               | 2.89               | 2.12               | 1.53               |

Tabla 8.10 Test MOS con BER variable para la secuencia "ingles2.wav"

|               |           | BER (Bit Error Rate aplicada a la secuencia) |                    |                    |                    |                    |                    |                    |
|---------------|-----------|--|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Secuencia     | Individuo | $1 \times 10^{-3}$                           | $5 \times 10^{-3}$ | $8 \times 10^{-3}$ | $1 \times 10^{-2}$ | $2 \times 10^{-2}$ | $3 \times 10^{-2}$ | $4 \times 10^{-2}$ |
| "ingles1.wav" | 1         | 4.60   | 4.50               | 4.20               | 3.20               | 3.00               | 2.00               | 1.40               |
|               | 2         | 5.00   | 4.20               | 4.30               | 3.00               | 2.80               | 1.80               | 1.80               |
|               | 3         | 4.60   | 4.30               | 4.00               | 3.00               | 3.00               | 1.80               | 1.20               |
|               | 4         | 4.50   | 4.50               | 4.30               | 2.60               | 2.60               | 2.00               | 1.00               |
|               | 5         | 4.40   | 4.50               | 4.00               | 2.60               | 2.50               | 1.40               | 1.20               |
|               | 6         | 4.20   | 4.30               | 3.80               | 3.20               | 2.80               | 1.20               | 1.40               |
|               | 7         | 5.00   | 4.50               | 3.80               | 3.20               | 2.50               | 2.00               | 1.20               |
|               | 8         | 4.30   | 4.40               | 4.00               | 3.20               | 3.00               | 2.00               | 1.00               |
|               | 9         | 4.50   | 4.20               | 4.00               | 2.80               | 2.50               | 2.30               | 1.20               |
|               | 10        | 4.20   | 4.40               | 4.00               | 3.50               | 3.00               | 2.20               | 1.00               |
|               |           | 4.53   | 4.38               | 4.04               | 3.03               | 2.77               | 1.87               | 1.24               |

Tabla 8.11. Test MOS con BER variable para la secuencia "ingles1.wav"

## 8.4 Mas Resultados

A continuación se muestran resultados para otro conjunto de secuencias de voces. Se presenta la secuencia de voz original, seguido de las secuencias de voz sintéticas tanto para un codificador que usa un *codebook* de tamaño 64 y 512.

### Secuencia "voz\_DArgandona.wav"

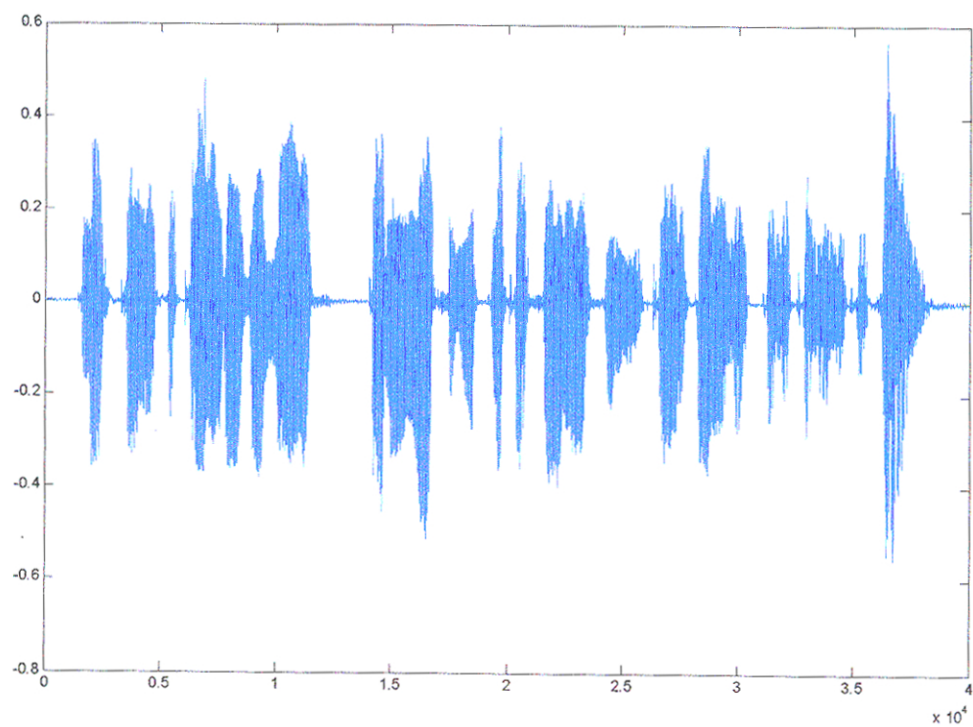


Fig. 8.16 Secuencia de voz original "voz\_DArgandona.wav"

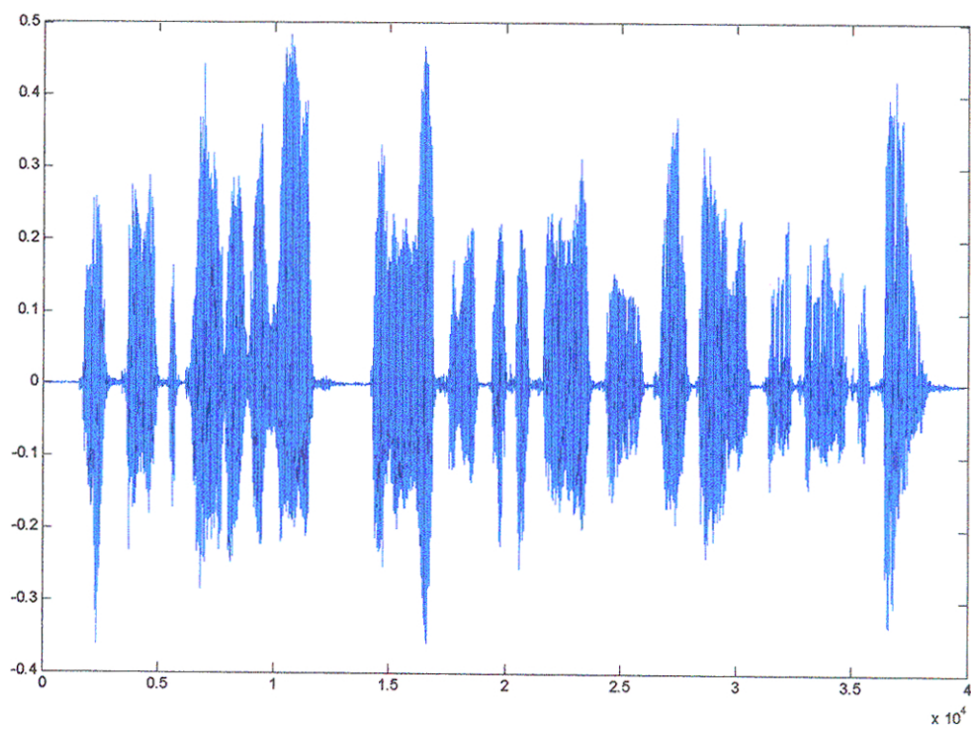


Fig. 8.17 Secuencia de voz sintética "voz\_DArgandona.wav", con *codebook* de tamaño 64

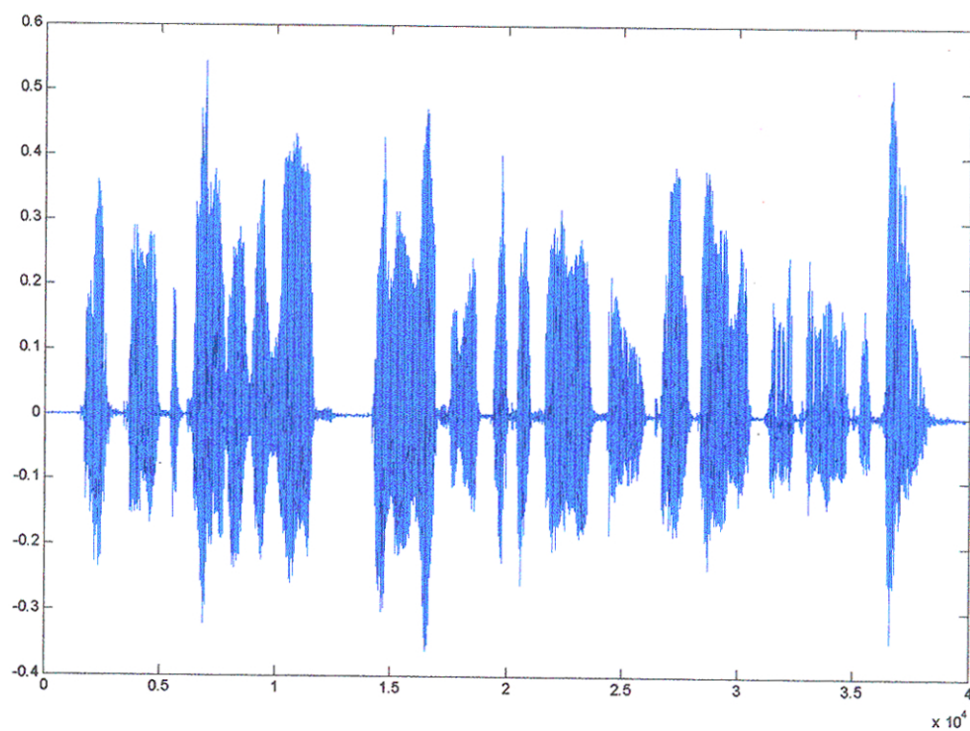


Fig. 8.18 Secuencia de voz sintética "voz\_DArgandona.wav", con *codebook* de tamaño 512

cuencia "voz\_FPujaico"

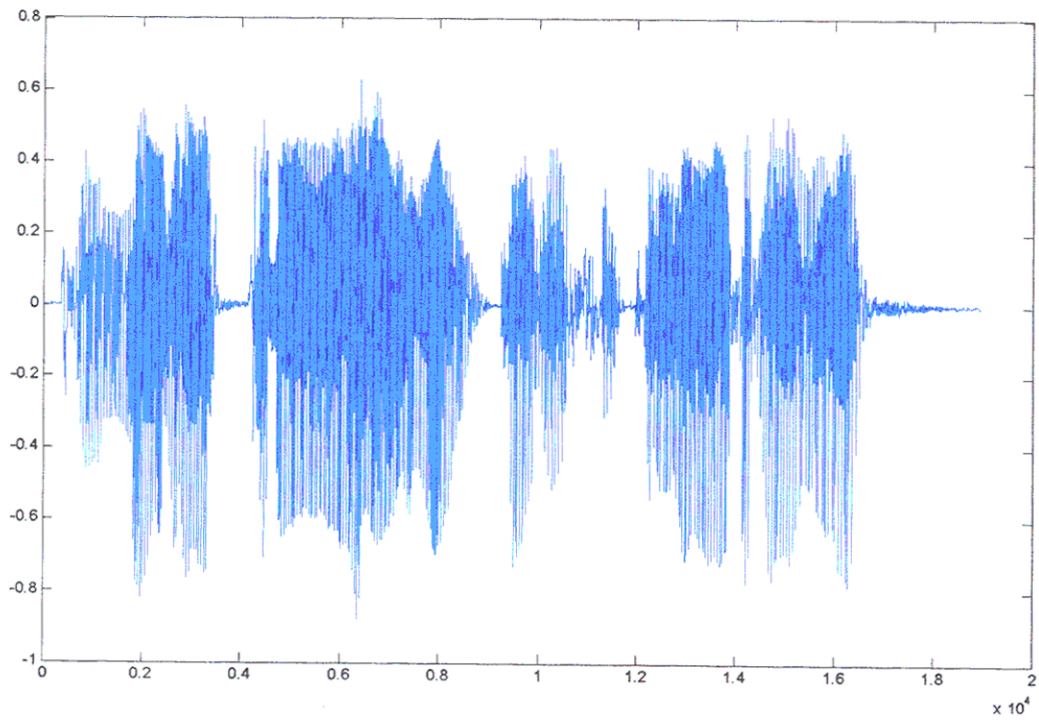


Fig. 8.19 Secuencia de voz original "voz\_FPujaico"

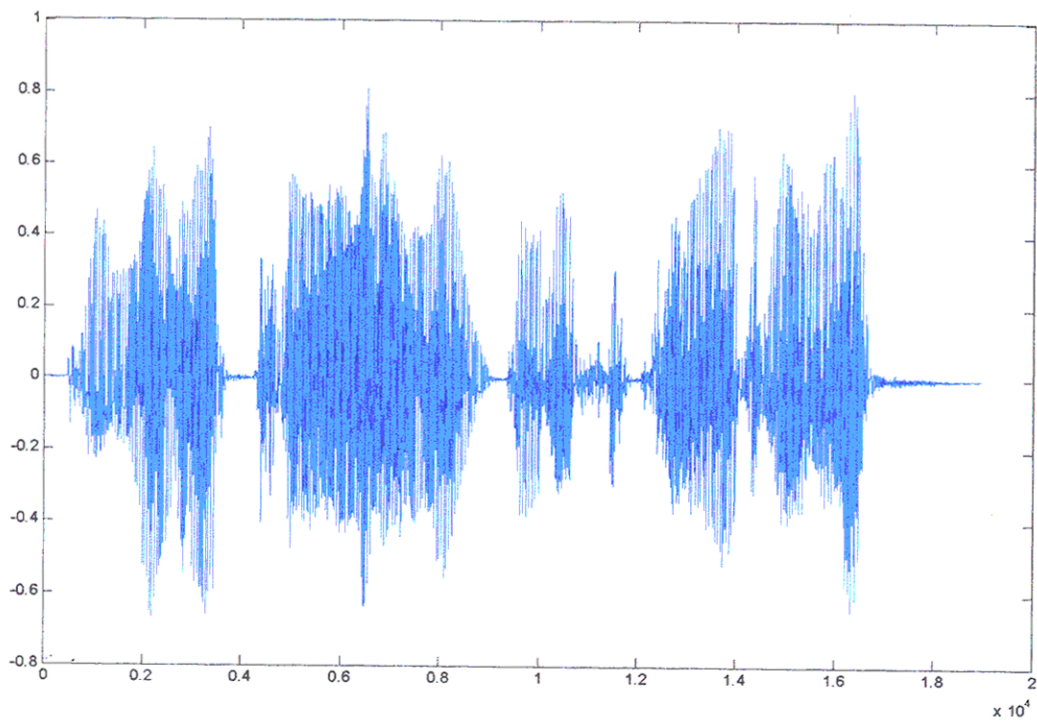


Fig. 8.20 Secuencia de voz sintética "voz\_FPujaico", con codebook de tamaño 64

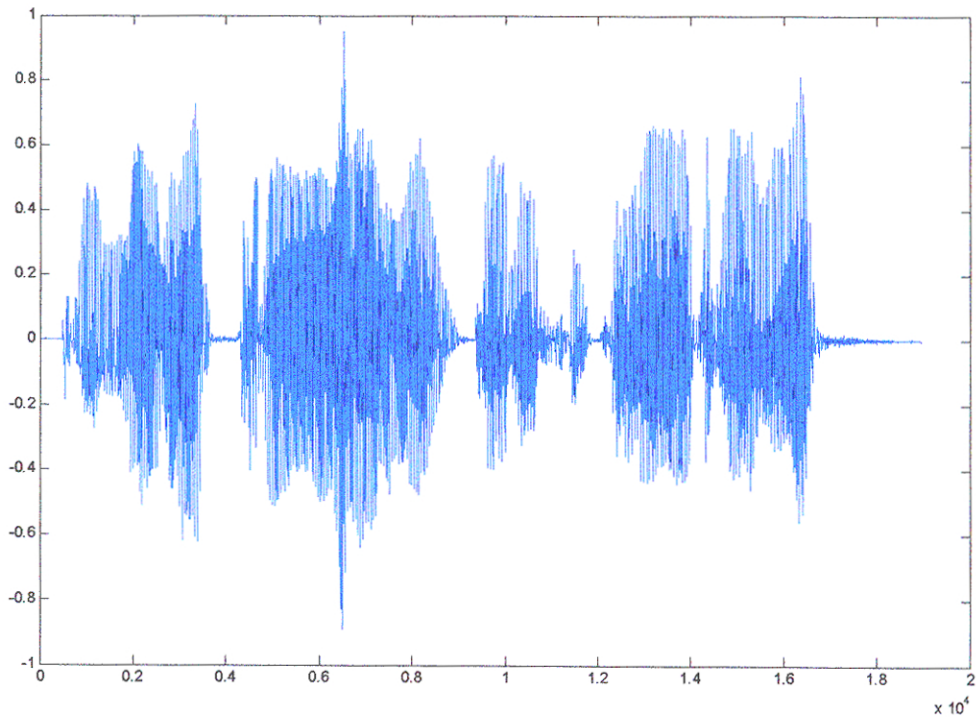


Fig. 8.21 Secuencia de voz sintética "voz\_FPujaico", con *codebook* de tamaño 512

Secuencia "voz\_JHuaman.wav"

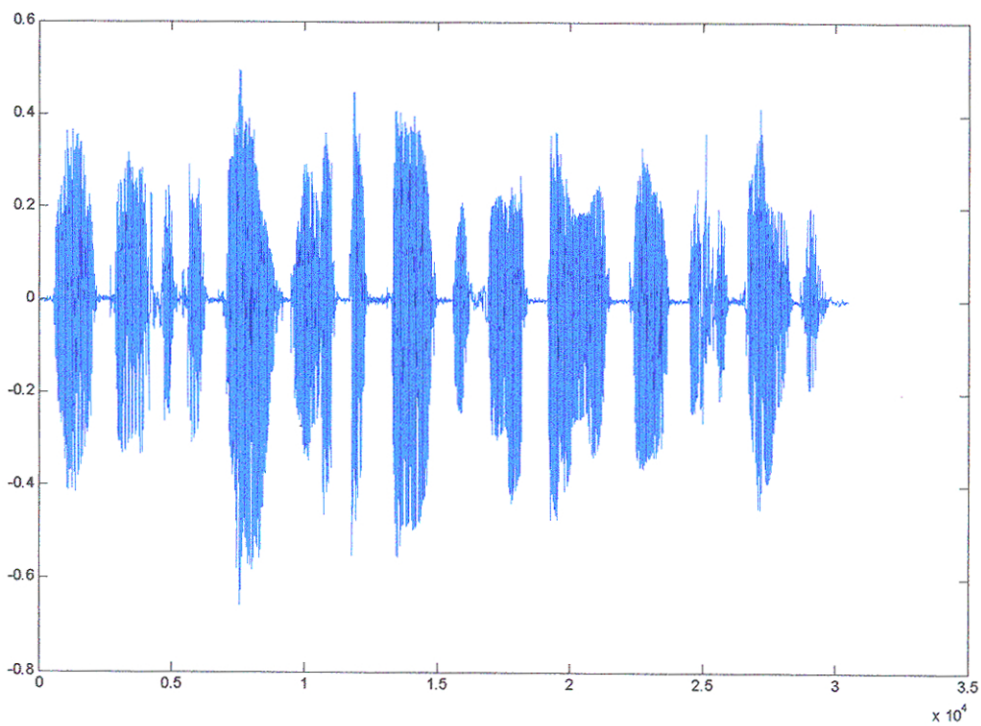


Fig. 8.22 Secuencia de voz original "voz\_JHuaman.wav"

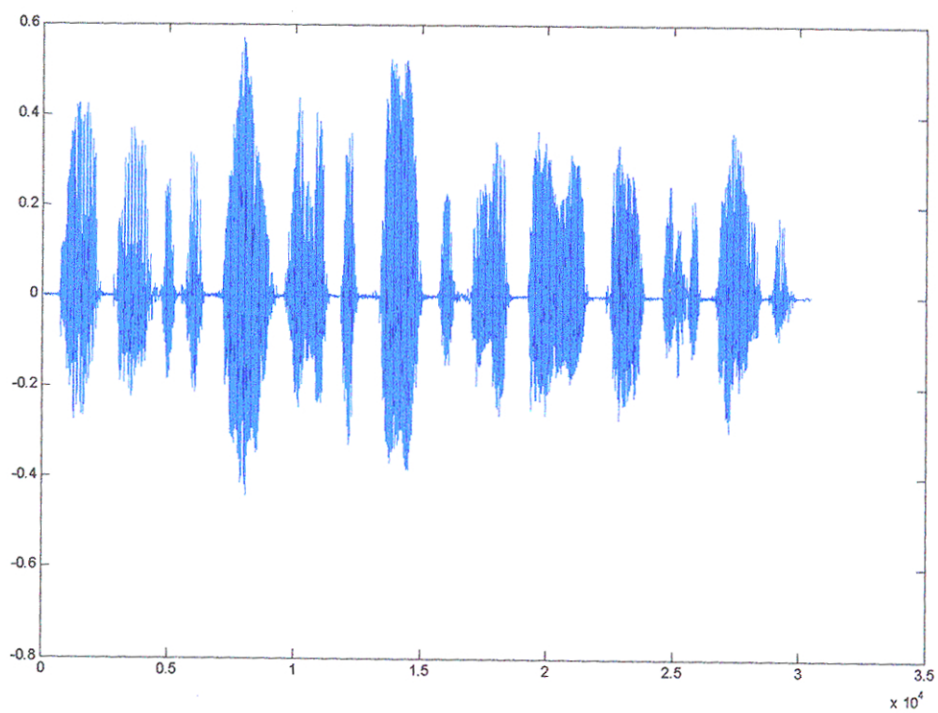


Fig. 8.23 Secuencia de voz sintética "voz\_JHuaman", con *codebook* de tamaño 64

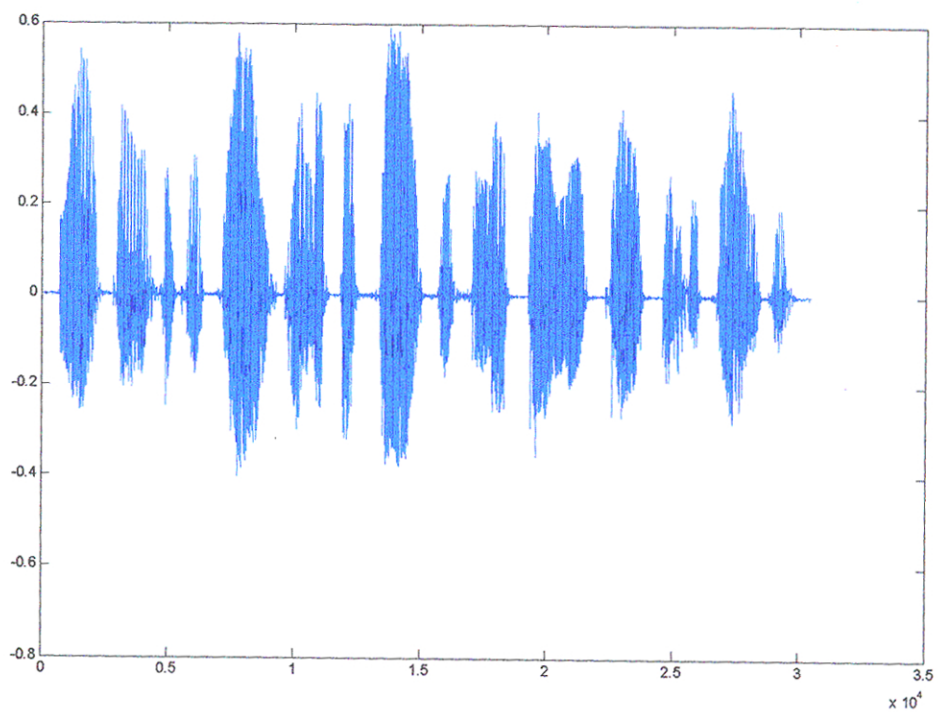


Fig. 8.24. Secuencia de voz sintética "voz\_JHuaman", con *codebook* de tamaño 512

Secuencia "voz\_JdelC.wav"

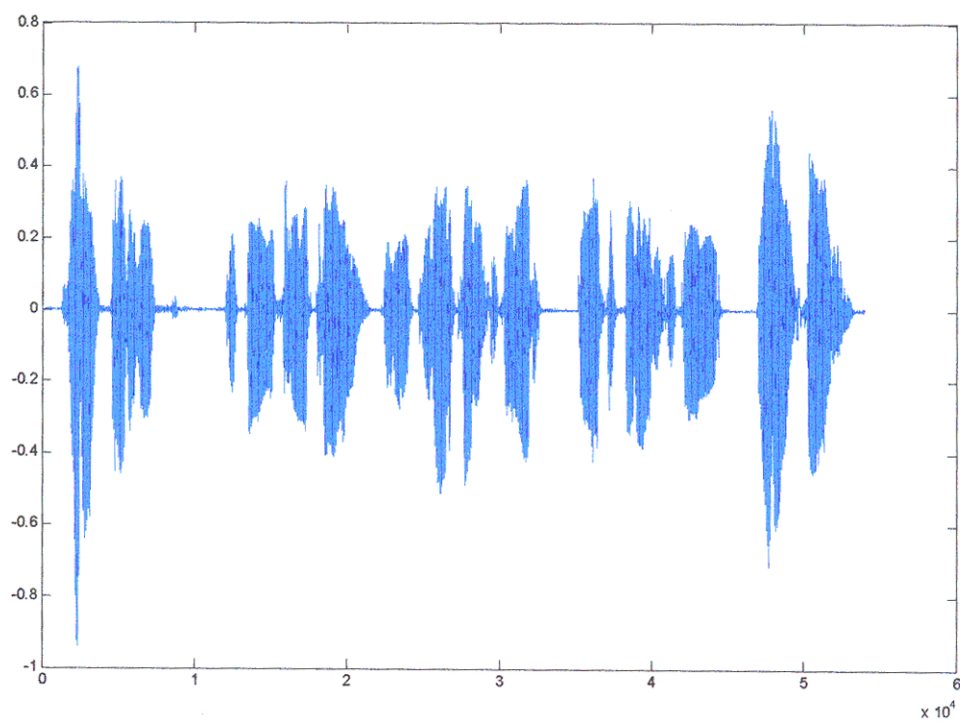


Fig. 8.25 Secuencia de voz original "voz\_JdelC.wav"

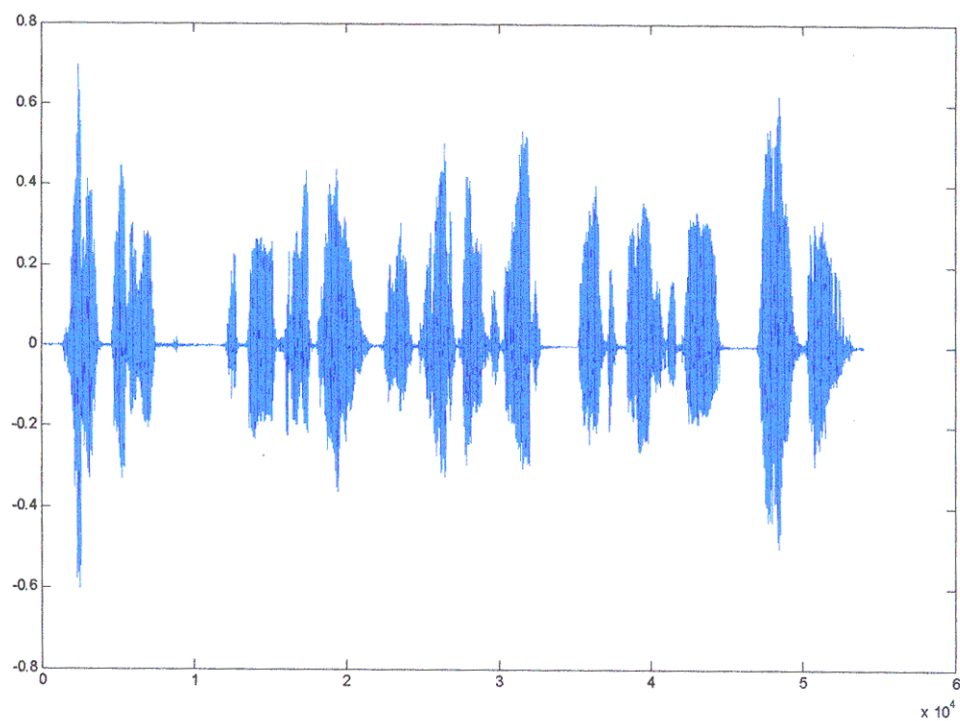


Fig. 8.26 Secuencia de voz sintética "voz\_JdelC", con *codebook* de tamaño 64



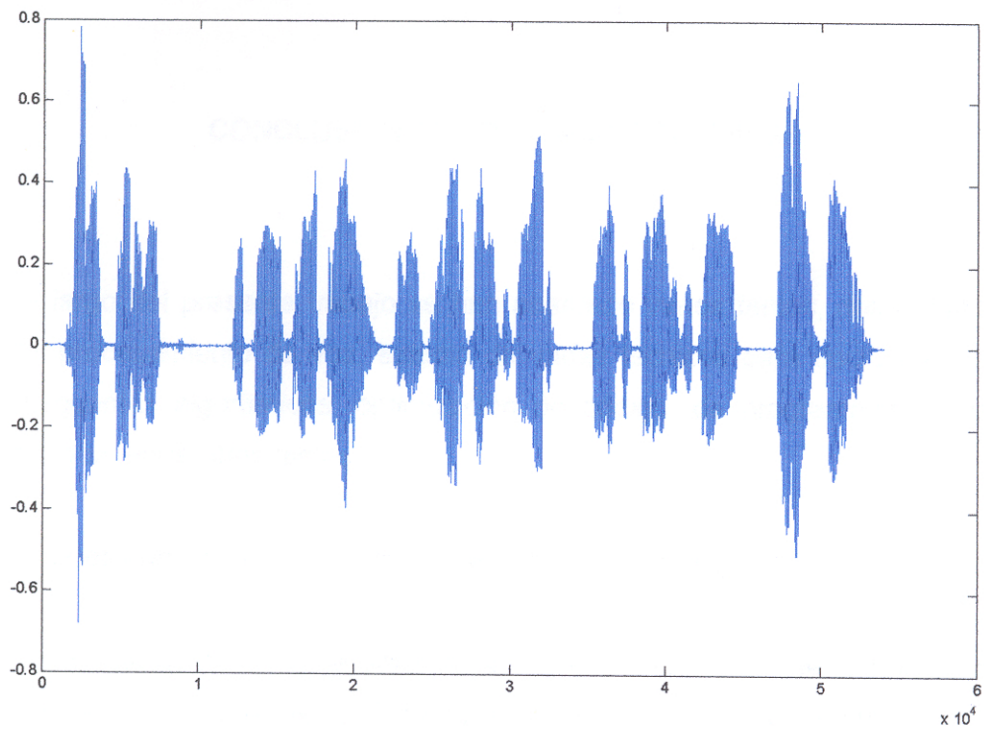


Fig. 8.27 Secuencia de voz sintética "voz\_JdelC", con *codebook* de tamaño 512

## CONCLUSIONES Y RECOMENDACIONES

A lo largo del presente trabajo se han abarcado varios temas, desde fundamentos teóricos del procesamiento de voz, estudio de varias técnicas para codificadores de voz, múltiples esquemas algorítmicos, optimización de código, de programación, pruebas de calidad de la voz, entre otros temas.

Las conclusiones que se derivan del presente trabajo son las siguientes:

- La codificación de la voz a bajas tasas de bits y con una buena calidad de la voz sintética decodificada, es posible empleando los esquemas y algoritmos apropiados.
- Existen diversos esquemas y técnicas estandarizadas que codifican la voz en un amplio rango de valores de la tasa de bits y ofrecen a la vez, diversos grados de calidad de la voz decodificada.
- Hay muchas aplicaciones que imponen una restricción fuerte en cuanto a la tasa de bits permitida para los codificadores de voz, lo que nos obliga a centrarnos en el desarrollo de codificadores para bajas tasas de bits (Capítulo 1).
- El esquema CELP se basa en un análisis por síntesis en el codificador, con el objetivo de buscar la mejor secuencia de excitación de entre 2 “*codebooks*” (uno dinámico y el otro fijo) lo cual lo hace exigente computacionalmente pero no impráctico (Capítulo 4). Con la potencia computacional actual, incluso los codificadores basados en CELP que no han sido optimizados, no presentan retardo aparente en el tiempo de ejecución.
- Los codificadores basados en el esquema CELP han sido muy exitosos a lo largo de los últimos años y actualmente muchos estándares se basan en dicho esquema ya que presenta varias ventajas con relación a otros esquemas (Capítulo 4). En el

presente trabajo se cumplió el objetivo de conocer a fondo el estandar CELP FS1016, lo cual permitió la inserción de mejoras y su posterior optimización.

- Uno de los objetivos de la presente implementación fue la de optimizar el desempeño y la calidad de la voz decodificada. En el ámbito algorítmico las mejoras logradas se resumen en las siguientes (ver Capítulo 6.13):
  - a) Modificación de la rutina de cálculo de ganancia adaptiva para minimizar las multiplicaciones.
  - b) Modificación de la rutina de cálculo de ganancia estocástica para minimizar las multiplicaciones.
  - c) Cambio en los criterios de búsqueda en el algoritmo de conversión de LPC a LSP.
  - d) Mejora del cuantizador para la ganancia estocástica (6 bits y modificación manual de los niveles del cuantizador).
  - e) Mejora del esquema FEC de (11,7) el cual es usado por el estandar original FS1016 al esquema (63,57) mas bit de paridad, usado por la presente implementación.
  
- En el ámbito de trabajo en el DSP, se requieren una serie de consideraciones especiales que dependen de la memoria física disponible, los tiempos de acceso, tipos de llamada a funciones, opciones de compilación, etc. Como se vio en el Capítulo 7, el proceso de optimización del código es un proceso escalonado y el nivel de optimización que se puede alcanzar depende del nivel al que lleguemos programando el código, si se quiere optimizar al máximo, todo el código se deberá realizar en assembler puro, lo cual muchas veces es impráctico (Capitulo 7). Se consiguió optimizar la velocidad del código mediante:
  - a) Reubicación de funciones criticas a la IRAM.
  - b) Reubicación de variables criticas a la IRAM.
  - c) Correcto uso de las opciones del compilador.
  
- La calidad de la voz sintética decodificada ha demostrado ser muy buena tal como se ha mostrado por las pruebas MOS. Incluso el desempeño del codificador frente a variaciones de BER ha sido aceptable, degradándose de forma considerable con valores de BER mayores a aproximadamente  $2 \times 10^{-2}$ , el cual corresponde al caso de mucho ruido.

- Finalmente, el código implementado se ha realizado de forma modular de modo que las futuras mejoras puedan ser realizadas solo a los bloques de código sin afectar el resto del código.

El presente trabajo de tesis puede ser empleado como referencia para otros trabajos de codificación de voz, para lo cual se recomienda lo siguiente:

- Si se opta por continuar el proceso de optimización del código, se recomienda la codificación de las secciones críticas del programa en el lenguaje assembler. También, como un paso intermedio, se podrían llevar dichas secciones a lenguaje C en punto fijo.
- Es posible usar cualquier bloque constituyente del algoritmo, como una caja negra, hacia otro sistema o codificador. Como se ha visto en el capítulo I, varios codificadores estandarizados emplean rutinas de búsqueda de *codebook* adoptivos y estocásticos.
- Para futuras implementaciones de codificadores de voz, se recomienda el uso de tarjetas de desarrollo DSP más modernas, como el TMS320C6713, TMS320C6416, etc, las cuales son muy superiores a la tarjeta usada en el presente trabajo de tesis.
- Del mismo modo, para futuros trabajos se recomienda integrar el codificador de voz a un sistema de comunicaciones digitales, y realizar pruebas de transmisión digital de la voz.

## BIBLIOGRAFÍA

- [1] Rabiner, L; Schafer, R; "Digital Processing of Speech Signals", 1978, Prentice Hall
- [2] Xuendong Huang, Alex Acero, Hon, "Spoken Language Processing – A Guide to Theory, Algorithm and System Development" 2001, Prentice Hall
- [3] Randy Goldberg, Lance Rick, "A Practical Handbook of Speech Coders" 2000, CRC
- [4] Wai Chu, "Speech Coding Algorithms, Foundation and Evolution of Standardized Coders" 2003, Jhon Wiley Sons
- [5] Wu Chou, Biing Hwang Juang, "Pattern Recognition in Speech and Language Processing", 2003, CRC Press
- [6] Tatham Mark, Morton Katherine, "Developments in Speech Synthesis", 2005, Wiley and Sons
- [7] Levinson Stephen, "Mathematical Models for Speech Technology", 2005, Wiley and Sons
- [8] Jurafsky, D; Martin, J.; "Speech and Language Processing", 2000, Prentice Hall
- [9] Divenyi, Pierre; "Speech Separation by Humans and Machines", 2005, Kluwer Academic Publishers
- [10] Héctor Kaschel C.1 Francisco Watkins1 Enrique San Juan U., "Compresión de Voz mediante Técnicas Digitales para el Procesamiento de Señales y Aplicación de Formatos de Compresión de Imágenes", Rev. Fac. Ing. - Univ.Tarapacá, vol. 13 N° 3, 2005

- [11] Javier Alejandro Bustos Jiménez, "Estudio de Sistemas de compresión de voz digital orientado a telefonía celular", Universidad de Chile, 2002
- [12] Sara Grassi, "Optimized Implementation of Speech Processing Algorithm", 1998
- [13] F. Itakura, "Line Spectrum Representation of Linear Predictive Coefficients of Speech Signals", J. Acoust. Soc. Amer., vol. 57, S35, 1975
- [14] J. Picone and G. Doddington, "A Phonetic Vocoder", Proc. IEEE Acoust., Speech Signal Processing, pp. 580-583, 1989
- [15] Faiza Anees, Mariam Shakeel, Shaista Nazir, "Implementation of Acoustic Echo Cancellation in Matlab and TI's TMS320 C6711", COMSATS INSTITUTE OF INFORMATION AND TECHNOLOGY, 2004
- [16] Islam, Tammana; "Interpolation of Linear Prediction Coefficients for Speech Coding", Dept. of Electrical Engineering, McGill University, Canada
- [17] Texas Instruments, "Implementing Vocoder y HF algorithms in DSP", 1995, Texas Instruments.
- [18] Tremain T., Kemp, David, "Evaluation of low rate speech code for HF" US Department of Defense, 1993
- [19] LeBlanc W., Bhattacharya B., "Efficient Search and Design Procedures for Robust Multi Stage VQ of LPC Parameters for 4kbps Speech coding" IEEE Transactions on Speech and Audio Processing, VOL 1, N° 4, 1993.
- [20] Tremain T., Campbell J., "A comparison of US Government Standard Voice Coders", US Department of Defense, 1989.
- [21] Crochiere R., Rabiner L. "Interpolation and Decimation of Digital Signals - A tutorial review", Proceedings of the IEEE, Vol 69, N°3, 1981
- [22] Jhon Makhoul, "Linear Prediction - A Tutorial review", Proceedings of the IEEE, Vol 63, N° 4, 1975.

- [23] P. Vaidyanathan, "Multirate Digital Filters Filter Banks Polyphase Networks and Applications: A Tutorial", Proceedings of the IEEE, Vol 78, N° 1, 1990
- [24] Juan Bello, Laurent Daudet "Onset Tutorial", IEEE Transactions on Speech and Audio Processing Vol 13, N° 5, 2005
- [25] Andreas Spanias, "Speech Coding: A Tutorial Review", Proceedings of the IEEE, Vol 82, N° 10, 1994
- [26] Campbell J, Tremain T, "The Federal Standard 1016 4800 bps CELP Voice Coder", Digital Processing 1, 1991
- [27] Jhon Markel, "The Sift algorithm for Fundamental Frequency Estimation", IEEE Transactions on Audio and Electroacoustics, Vol 20, N° 5, 1972
- [28] Campbell J, Tremain T, "Voiced / Unvoiced classification of speech with applications to the US government LPC-10E algorithm", Department of Defense, 1986
- [29] Trancoso I., Nishnu A., "Efficient Search Procedure For Selecting the Optimum Innovation in Stochastic Coders", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol 38, N° 3, 1990
- [30] McCree A., Barnwell T., "A mixed Excitation LPC Vocoder Modes for Low Bit Rate Speech", IEEE Transactions on Speech and Audio Processing Vol 3, N° 4, 1995
- [31] Tremain, Thomas E., Joseph P. Campbell, Jr and Vanoy C. Welch, "A 4.8 kbps Code Excited Linear Predictive Coder," Proceedings of the Mobile Satellite Conference, 3-5 May 1988, pp. 491-496.
- [32] Campbell, Joseph P. Jr., Vanoy C. Welch and Thomas E. Tremain, "An Expandable Error-Protected 4800 bps CELP Coder (U.S. Federal Standard 4800 bps Voice Coder)," Proceedings of ICASSP, 1989 (and Proceedings of Speech Tech, 1989.)
- [33] Thomas E. Tremain, "The Government Standard Linear Predictive Coding Algorithm: LPC-10," Speech Technology Magazine, April 1982, p. 40-49..

- [34] Xydeas, C.S., M.A. Ireton and D.K. Baghbadrani, "Theory and Real Time Implementation of a CELP Coder at 4.8 and 6.0 kbits/s Using Ternary Code Excitation," Fifth International Conference on Digital Processing of Signals in Communications, 1988, p. 167.
- [35] Kroon & Atal, "Pitch Predictors with High Temporal Resolution," ICASSP '90, S12.6
- [36] Shoham, Yair, "Constrained-Stochastic Excitation Coding of Speech at 4.8 kbps," in Advances in Speech Coding, ed. B. Atal, V.Cuperman, and A. Gersho, submitted to Kluwer Academic Publishers.
- [37] Shoham, Yair, "Constrained-Stochastic Excitation Coding of Speech," Abstracts of the IEEE Workshop on Speech Coding for Telecommunications, 1989, p. 65.
- [38] Kleijn, Krasinski and Ketchum, Improved Speech Quality and Efficient Vector Quantization in SELP, ICASSP, 1988.
- [39] Kroon, Peter and Bishnu Atal, "On Improving the Performance of Pitch Predictions in Speech Coding Systems," IEEE Speech Coding Workshop, September 1989.
- [40] Marques, J.S., et al., "Pitch Prediction with Fractional Delays in CELP Coding," European Conference on Speech Communication and Technology, September, 1989.
- [41] [www.texas.com](http://www.texas.com)
- [42] [www.compendent.com](http://www.compendent.com)
- [43] [www.radtelnetwork.com.au](http://www.radtelnetwork.com.au)
- [44] [www.unil.ch](http://www.unil.ch)
- [45] <http://escuela.med.puc.cl>
- [46] [www.matlab.com](http://www.matlab.com)