Práctica 4

# Codificación de voz y audio

## 4.1. Introducción

En esta práctica realizaremos algunos ejercicios que nos ayudarán a comprender mejor la codificación de formas de onda de voz y/o audio. Todos los ejercicios se realizarán en Matlab, algunos con la ayuda del simulador de diagramas de bloques denominado Simulink.

## 4.2. Codificación de forma de onda

### 4.2.1. Pulse Code Modulation (PCM)

#### Cuantificación uniforme

En el diagrama de bloques de Simulink denominado "diagrama1.mdl" se representa la forma de onda de una señal de voz y la salida de un cuantificador.

**Ejercicio 1** *Observa las diferencias entre la señal original y la cuantificada para un número de niveles de 8,16,32,64 y 256.*

#### Cuantificación no uniforme

En el diagrama de bloques "diagrama2" se representa el diagrama de bloques correspondiente a un cuantificador no uniforme que emplea ley $\mu$.

**Ejercicio 2** *Explica el procedimiento empleado para realizar la cuantificación no uniforme. Observa la influencia del parámetro $\mu$ en la ley de compansión y en la característica del cuantificador.*

**Ejercicio 3** *Ahora estudiaremos el comportamiento de los cuantificadores uniforme y no uniforme ante sinusoides de amplitudes por encima y por debajo del valor de sobrecarga.*

*a) En los ficheros "snrqmu.m" y "snrqa.m" se calcula la relación señal a ruido para sinusoides de distintas amplitudes. Analiza el código.*

*b) Observa el código de "compara_snr.m", ejecútalo e interpreta la gráfica resultante.*

**Cuantificación adaptativa**

Una alternativa a la cuantificación no uniforme consiste en hacer variar el tamaño del escalón de cuantificación en función de la potencia de la señal. Se llega así al codificador APCM (Adaptive PCM). En este esquema el tamaño del escalón varía de forma directamente proporcional a la potencia de la señal.

**Ejercicio 4** *Observa el comportamiento del APCM del diagrama "diagrama3.mdl" para distintos números de niveles del cuantificador.*

**Ejercicio 5** *La utilización del esquema anterior tal y como está diseñado en el diagrama de bloques en un sistema de comunicación requeriría la transmisión de la evolución del tamaño del escalón. ¿Se te ocurre alguna modificación en el diagrama de bloques de forma que no sea necesario transmitirlo?*

### 4.2.2. Sistemas de codificación diferencial

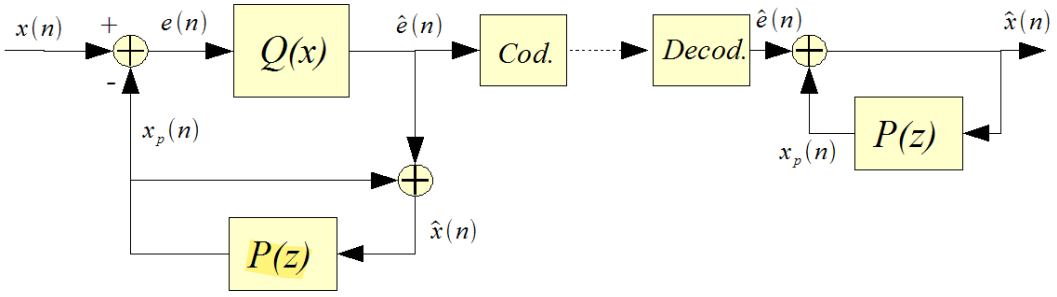


Figura 4.1: DPCM

**Codificación DPCM**

En la figura 4.1 se representa el esquema de un DPCM (PCM diferencial). La secuencia $e(n)$ se calcula como

$$e(n) = x(n) - x_p(n) \tag{4.1}$$

donde $x_p(n)$ es una estimación de $x(n)$ obtenida a partir del filtro de predicción. De la figura se deduce que el error cometido por el sistema DPCM es únicamente el introducido por el cuantificador:

$$\varepsilon(n) = x(n) - \widehat{x}(n) = x_p(n) + e(n) - x_p(n) - \widehat{e}(n) = e(n) - \widehat{e}(n) \tag{4.2}$$

Cuando los coeficientes del predictor no son fijos (varían con el tiempo) el esquema recibe el nombre de DPCM adaptativo (ADPCM), incorporando además, normalmente, un escalón de cuantificación también variable. Los coeficientes del predictor pueden reestimarse cada nuevo valor cuantificado mediante el algoritmo adaptativo

$$\underline{c}(n+1) = \underline{c}(n) + \mu \cdot \hat{e}(n) \cdot \hat{x}(n) \tag{4.3}$$

siendo $\underline{c}(n)$ el vector de coeficientes del predictor en el instante n-ésimo.

Para la simulación de los codificadores diferenciales partiremos de una secuencia discreta obtenida con una frecuencia de muestreo elevada (a la que denominaremos frecuencia de simulación) y posteriormente la diezmaremos para obtener la frecuencia de muestreo adecuada (a la que nos referiremos simplemente como frecuencia de muestreo).

**Ejercicio 6** *Observa el comportamiento del diagrama ADPCM "diagrama4". Edita los parámetros globales de la simulación y explica su significado. Observa también los parámetros del cuantificador.*

**Ejercicio 7** *¿Coincide el error cometido por el esquema anterior con el valor teórico?*

### 4.2.3. Modulación delta lineal (DM)

La modulación delta lineal es una variante del DPCM con un cuantificador de 1 bit (dos niveles) y un mero retardo como predictor. Se suele utiliza con frecuencias de muestreo elevadas (sobremuestreo) de forma que existe una alta correlación entre muestras de entrada. Ya hemos mencionado en clase de teoría que en un modulador delta lineal se pueden distinguir dos tipos de error: error de sobrecarga de pendiente y error granular.

**Ejercicio 8** *Teniendo en cuenta el diagrama de bloques DELTA. Completa la siguiente tabla y representa la potencia de ruido en función del tamaño del escalón. ¿Qué conclusiones se pueden extraer de la gráfica?*

| $\Delta$ | *Pot. ruido* |
|---|---|
| *0* | |
| *0.05* | |
| *0.1* | |
| *0.15* | |
| *0.2* | |
| *0.25* | |
| *0.3* | |
| *0.35* | |
| *0.4* | |
| *0.45* | |
| *0.5* | |

**Ejercicio 9** *Calcula el valor del parámetro "delta" de forma que no haya distorsión de sobrecarga de pendiente y el ruido granular sea lo más pequeño posible. Compruébalo en la simulación.*

**Ejercicio 10** *Para el valor de escalón óptimo, modifica el factor de sobremuestreo alterando el parámetro "diezma", y observa su efecto sobre el ruido de cuantificación. Completa la siguiente tabla, representa la potencia de ruido en función del factor de sobremuestreo y extrae conclusiones.*

200k/diezma
"

| *diezma* | $f_s$ | *Pot. ruido* |
|---|---|---|
| *40* | | 0,41024 |
| *20* | | 0,3485 |
| *15* | | 0,109215 |
| *10* | | 0,093563 |
| *4* | | 0,10224 |
| *1* | | 0,02183 |

D=0,5

D=0,25

q tiene que ser con D=0,25

## 4.3. Codificación paramétrica

En este apartado desarrollaremos la etapas de análisis y síntesis de un vocoder LPC simple, partiendo de una estructura básica definida en "vocLPC_incompleto.m". Tendrás que finalizar el código contenido en este fichero ya sea completando líneas del mismo o programando alguna función. A continuación se proporcionan más detalles:

- El codificador trabajará con una frecuencia de muestreo de 8 kHz, realizando inicialmente el análisis de la señal de voz cada 20 ms y utilizando inicialmente una ventana de análisis Hamming de 35 ms.
- Tal y como ya figura en el código el primer paso es eliminar cualquier posible componente continua en la señál de voz.
- A continuación se realizará el preénfasis con un filtro $H(z) = 1 - \alpha z^{-1}$, empleando $\alpha = 0{,}9$.
- Los elementos de memoria (retardos) del filtro LPC de síntesis deben mantenerse debidamente actualizados.
- En caso de tramas sonoras consecutivas se debe generar la señal de excitación de forma que se respete la continuidad de frecuencia fundamental entre tramas.
- La periodicidad y sonoridad se estiman a partir de la función de autocorrelación de cada trama. Deberás programar la función. Para la detección de la sonoridad utiliza simplemente el cociente entre la amplitud de la autocorrelación correspondiente al periodo fundamental y su valor en el origen, que compararás con un umbral que deberás fijar.
- También tendrás que programar la generación de la señal de excitación, consistente en un tren de deltas o ruido blanco depndiendo de la sonoridad de la trama. La ganancia de la excitación debe ajustarse de acuerdo con el procedimiento descrito en la sección 4.4.2 de la parte I de los apuntes. Así para tramos sordos consideraremos $g = \sqrt{E_p}$, mientras que para sonoros $g = \sqrt{E_p T/N}$ donde $T$ representa el periodo fundamental y $N$ la longitud de la trama de análisis.

## 4.4. Modelado sinusoidal

El fichero "codecsin.m" implementa un sistema básico de análisis/síntesis sinusoidal. La señal se modela como una suma de sinusoides cuyas frecuencias, amplitudes y fases se obtienen

de los picos espectrales de las distintas tramas. Dado que el número de componentes espectrales (sinusoides) y su posición puede variar de trama a trama, se establece un procedimiento de nacimiento y muerte de sinusoides, aunque con mucha frecuencia las componentes espectrales tienen continuidad (evolucionan lentamente) a lo largo de varias tramas. Así dos picos de tramas sucesivas se consideran modelados por una única sinusoide si su distancia en frecuencia es menor que un determinado umbral.

**Ejercicio 11** *Analiza el código de "codecsin.m" y observa su funcionamiento.*

**Ejercicio 12** *Los modelos sinusoidales armónicos asumen que durante los tramos sonoros los picos espectrales están situados en los múltiplos de la frecuencia fundamental. El modelado de los tramos sordos se puede realizar con una frecuencia fundamental ficticia de unos 100 Hz, de forma que tengamos una suficiente resolución espectral.*

*Con la ayuda del estimador de frecuencia fundamental y de sonoridad utilizado en el vocoder LPC, modifica "codecsin.m" (guardalo con otro nombre, previamente) para que realice un modelado armónico.*

## 4.5. Codificación de audio

### 4.5.1. Modified Discrete Cosine Transform (MDCT)

**Ejercicio 13** *En el fichero "fb_mdct" se ilustra la equivalencia entre la MDCT y un banco de filtros. Analiza el código y ejecútalo.*

**Ejercicio 14** *El fichero "mdct_imdct_incompleto" presenta un sistema básico de análisis/síntesis mediante la MDCT. Analiza su contenido y completa las dos líneas de código necesarias para que funcione correctamente.*

**Ejercicio 15** *Ahora cuantificaremos la MDCT en el sistema anterior de forma muy burda ya que utilizaremos el mismo cuantificador y el mismo factor de escalado para todos los coeficientes de la MDCT. Para realizar la cuantificación utilizaremos la función "quantize". Por ejemplo "w=quantize(z,5,'scale')" devuelve los valores de z cuantificados con 5 bits y ajustando el valor de sobrecarga del cuantificador al máximo en valor absoluto de la señal. Añade al fichero del ejercicio anterior el código necesario para cuantificar la MDCT con 5 bits y realizar la reconstrucción a partir de la MDCT cuantificada.*

**Ejercicio 16** *Con la MDCT cuantificada como en el apartado anterior vamos a analizar la influencia de la longitud de la ventana sobre los pre-ecos. Para ello trabajaremos con el fichero "cast.wav" y con longitudes de ventana de 24.5 ms y 6 ms. Nos centraremos en la observación de la forma de onda en torno al comienzo del sonido de las castañuelas.*

**Ejercicio 17** *Para el control de pre-ecos implementaremos un algoritmo muy simple de Temporal Noise Shaping (TNS). Para que su funcionamiento quede más patente volveremos a utilizar la ventana de 24.5 ms. El TNS se basa en utilizar predicción lineal en el dominio espectral. Así estimaremos los LPC para la MDCT de un bloque, obtendremos el correspondiente error de predicción que será lo que cuantifiquemos y posteriormente reconstruiremos los coeficientes de la MDCT a partir del error cuantificado y los LPC. A partir de los coeficientes de la MDCT así*

*obtenidos regeneraremos la señal de audio. Añade las líneas de código necesarias al fichero de los ejercicios previos y observa la forma de onda en torno al comienzo del sonido de las castañuelas. Compara los resultados con los obtenidos en caso de no utilizar TNS.*