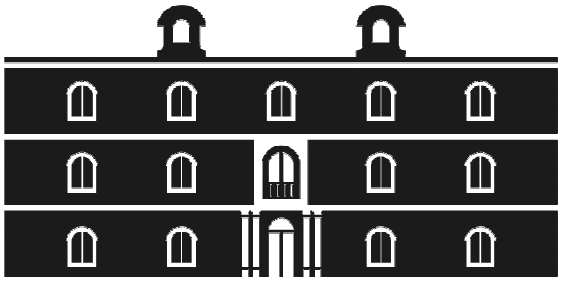
Universidad Politécnica de Cartagena



industriales

etsii UPCT

Segmentación y parametrización de transiciones del habla aplicada al reconocimiento de locutores.

Titulación: Ingeniero en Automática y Electrónica Industrial
Intensificación:
Alumno/a: Jose C. Alcaraz Sánchez
Director/a/s: Antonio Guerrero González

Cartagena, 15 de Octubre de 2012

Presentación

El análisis del habla por sus características y complejidad constituye un campo de investigación, que en las últimas décadas ha concentrado un gran número de importantes aportaciones científicas y técnicas.

Por un lado, la señal del habla es una señal temporal extremadamente compleja, sus parámetros físicos varían rápidamente y la implementación para aplicaciones reales de los modelos matemáticos capaces de caracterizarlos exige una gran potencia de computación.

El caso de verificación e identificación de locutores con fines forenses es una de sus vertientes más complejas, no sólo por la necesidad de obtener grandes márgenes de confianza en las identificaciones, sino que las especiales características de la señal forense (ruido, manipulaciones, grabaciones defectuosas, etc) hacen que este campo sea especialmente dificultoso desde el punto de vista de desarrollar técnicas adecuadas.

Los algoritmos de segmentación del habla han utilizado tradicionalmente criterios basados en parámetros de la señal temporal, tales como la amplitud de la señal, el número de cruces por cero o frecuencia fundamental (pitch), y caracterizaciones de la señal mediante el modelo Todo-Polo (o modelo LPC), formulado en el dominio del tiempo o bien de la frecuencia.

Teniendo en cuenta las peculiaridades que se dan en la generación del habla natural (acoplamientos, coarticulación, silencios,...), buscaremos una alternativa a estas técnicas clásicas en métodos basados en la dinámica no lineal. Esto nos llevará a que, necesariamente, primero centremos el estudio en la naturaleza caótica de las señales del habla natural. En este sentido se estudiarán parámetros de la dinámica no lineal entre los que podemos destacar la Dimensión de Correlación, que nos permitirán concluir sobre el citado carácter caótico.

Se desarrollarán diversos algoritmos basados en técnicas no lineales (Entropía Aproximada, K2-Entropía y Dinámica Simbólica) para la caracterización de transiciones entre fonemas de distinta naturaleza, en concreto aquellas que presentan coarticulación nasal, que suponen por su complejidad una de las situaciones que presentan mayor dificultad para los sistemas de segmentación clásicos.

La parametrización LPC, tal y como se ha podido constatar en anteriores estudios, no es adecuada para la modelización de segmentos transicionales. A tal fin, se implementará un algoritmo de parametrización ARMA basado en el método de Shanks. Teniendo en cuenta que el objetivo último de este proyecto es plantear un estudio de reconocimiento de locutores, proponemos como medida de comparación una métrica ARMA, que se calcula a partir de los coeficientes ARMA utilizando determinantes (en concreto la resultante entre dos polinomios).

Finalmente, realizaremos una comparación entre la técnica clásica de Análisis Discriminante y la nueva técnica introducida, basada en la métrica ARMA, sobre una población extraída de la Base de Datos Ahumada, desarrollada por la Dirección General de la Guardia Civil. Concretamente se planteará el problema de reconocimiento de locutores trabajando con señales del habla natural en las que está

presente el fenómeno de coarticulación nasal, admitiendo la hipótesis de que la coarticulación nasal representa una característica fuertemente dependiente del locutor.

Por otra parte, la ingeniería de software vinculada al tratamiento de señales de voz conduce a herramientas con unas especificaciones técnicas muy exigentes. Generalmente, se requiere que la aplicación sea lo suficientemente rápida para procesar gran cantidad de información en intervalos reducidos de tiempo o incluso que trabaje en tiempo real.

Todos los algoritmos serán implementados en el entorno de programación LabVIEW®, de la empresa National Instrument, creando una librería de funciones que podrán ser utilizadas para desarrollar aplicaciones de caracterización y segmentación automática de señales del habla y aplicaciones de reconocimiento de locutores.

Índice

Capítulo 1: Introducción

Generación de la señal del habla
Procesado de señales
Modelado de señal, modelo de predicción lineal
Segmentación, métodos de la dinámica no lineal
Reconocimiento de locutores
Objetivos del Proyecto

1. Introducción

1.1. Generación de la señal del habla

La señal del habla es el resultado de la acción voluntaria y coordinada de los aparatos respiratorio y digestivo. Participan los siguientes órganos y cavidades: pulmones, traquea, laringe (con cuerdas vocales y glotis), faringe, boca (con lengua y labios) y la cavidad nasal. Al conjunto de estas cavidades y órganos lo llamaremos aparato fonador.

La fuente de energía es la fuerza muscular que cierra la caja torácica y provoca la contracción lenta del diafragma. Esta acción conjunta expele un flujo de aire desde los pulmones hacia la tráquea. Las propiedades de este flujo de aire pueden cambiar a lo largo del aparato fonador. Por una parte, puede ser modulado en las cuerdas vocales, que actuarían transformando el flujo continuo de aire en un tren de impulsos cuasi-periódico. La frecuencia de vibración de las cuerdas vocales viene determinada por la presión de aire en los pulmones y las propiedades mecánicas de los ligamentos y se denomina tono o frecuencia fundamental (también se conoce como *pitch*). Este tren de impulsos al entrar en el tracto vocal daría lugar a un sonido sonoro (las vocales y algunas consonantes, como *b*, *g*, *m*, etc.). Si las cuerdas vocales no vibran, la señal que pase será semejante a un ruido blanco y se origina un sonido sordo (como *p*, *t*, etc.), (Holmes, J.N., 1988).

Las características de la señal se modifican en las cavidades del tracto vocal, que actúa como resonador, formado por laringe, faringe, boca y labios; en algunos sonidos también se produce un acoplamiento con la cavidad nasal, y ésta entraría a formar parte del tracto vocal. Este se puede asemejar a un filtro acústico que amplificará o atenuará distintas frecuencias. Al cambiar el trato vocal ya sea cambiando la longitud y forma de las cavidades o bien por medio del movimiento de estructuras móviles llamadas articuladoras como son la lengua, los labios y el velo paladar, cambiaríamos las características del filtro y con ello sus frecuencias de resonancia o formantes (Sundberg, J., 1977).

Veamos a continuación dos ejemplos de señales, sonora y sorda, y sus respectivos espectros:

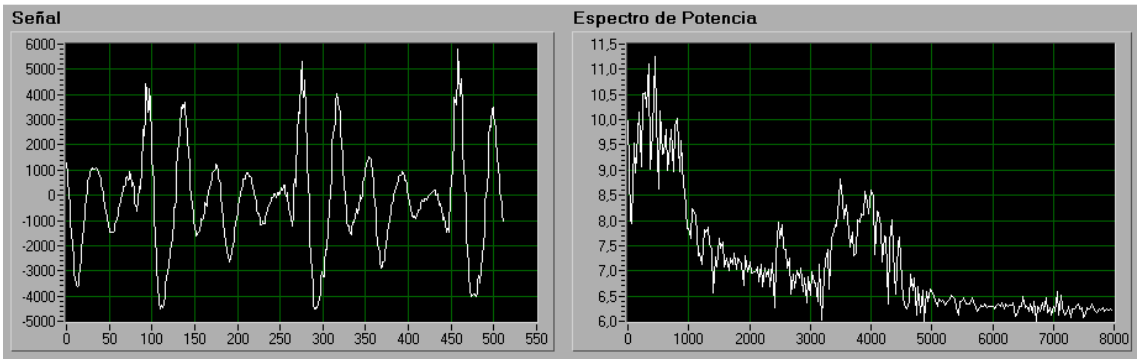


Figura 1.1: *Señal y espectro de potencia correspondientes al fonema /o/*

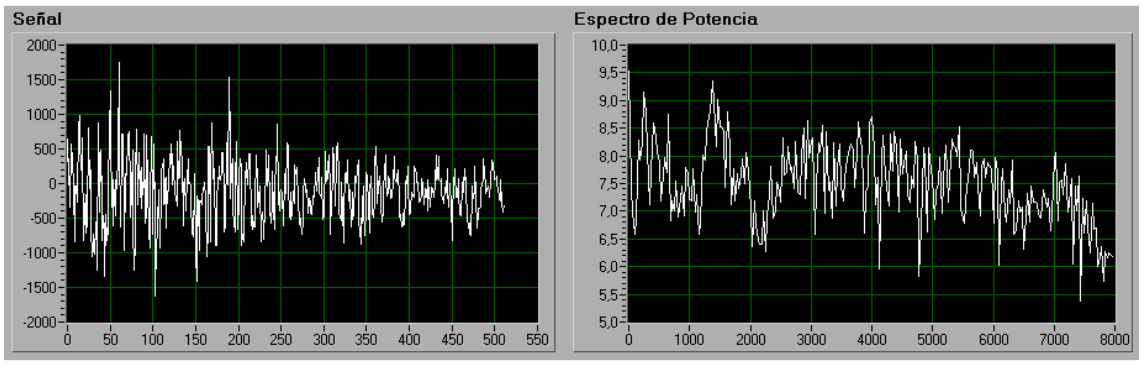


Figura 1.2: *Señal y espectro de potencia correspondientes al fonema /s/*

En la figura1.1, correspondiente a un sonido sonoro, observamos como aparecen los formantes (picos) donde la señal es amplificada. Las frecuencias que se sitúen próximos a las frecuencias de resonancia serán intensificados, mientras que aquellos con frecuencias intermedias entre dos formantes serán amortiguados modificando por tanto la señal que llega al tracto vocal. En el sonido sordo, por el contrario, nos encontramos con que todas las frecuencias tienen una amplificación similar. Este espectro es semejante al de un ruido. En cualquier caso se observa que ambas señales tendrán un contenido rico en armónicos.

Determinadas formas del tracto vocal influyen decisivamente en cada uno de los formantes. Así, puede decirse que el primer formante está relacionado con la apertura de la mandíbula, el segundo formante con la posición del cuerpo de la lengua y el tercer formante, con la posición de la punta de la lengua, siendo el resto de las frecuencias de resonancia más característico de cada aparato fonador.

Los procesos de coarticulación, entonación, variación de la intensidad, etc., que son consustanciales al habla, complican más el estudio, cuando se trabaja con habla conectada, en un contexto de habla natural. Se desprende por todo ello que la señal acústica del habla es no estacionaria, produciéndose fuertes modificaciones en la forma de la onda de los distintos fonemas (Atal, B. S., 1976).

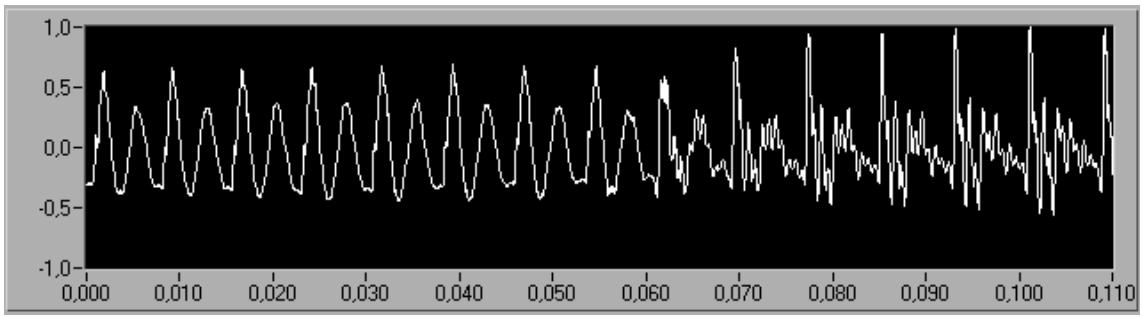


Figura 1.3*: Señal correspondiente a la transición con coarticulación nasal correspondiente al fonema /m/a/*

Sin embargo, en muchas aplicaciones prácticas se puede considerar la estacionalidad local para algunos fonemas, en intervalos muy cortos de tiempo de 20 a 40 ms (Teodorescu, H. y otros, 1997).

1.2. Procesado de señales

Una señal se define como una *cantidad física que varía con el tiempo, el espacio o cualquier otra variable o variables independientes* (Proakis, J.G., Manolakis, D.G., 1998). Las señales del habla son señales naturales cuya variable independiente es el tiempo. En su forma original, el dominio temporal en el que se produce la señal es el tiempo continuo, pero generalmente se discretizará para su estudio en ordenadores o DSP, de modo que consideramos el análisis en tiempo continuo y en tiempo discreto.

1.2.1. El proceso de muestreo

El sonido es una vibración que se propaga a través del aire u otro medio físico, debido a que las moléculas del medio transmiten la vibración.

Un micrófono conectado a una tarjeta de sonido incorporada a su vez en una computadora realiza una función semejante a la del oído humano. Ambos transforman pequeñas variaciones en la presión del aire en señal eléctrica que puede ser comprendida, almacenada y analizada por la CPU o el cerebro humano respectivamente.

A la información obtenida al transformar la onda de moléculas que vibran en el aire en una forma eléctrica que puede analizarse, manipularse, guardarse y reproducirse, se le denomina señal, y al proceso de transformación se le llama muestreo de dicha señal.

Cuando las ondas de sonido llegan al micrófono, el movimiento mecánico se traduce en una señal eléctrica, esta señal se denomina analógica porque es una señal continua, análoga al sonido original.

El elemento más delicado de la toma de datos mediante un equipo de audio para el análisis del habla con un sistema informático, es la transformación de la señal analógica en una señal digital que la CPU del ordenador pueda manipular.

Una incorrecta discretización de la señal provocará que el análisis posterior de la misma no sea fiable, ya que el proceso puede introducir informaciones falsas. Debe observarse, por tanto, un cierto cuidado en el proceso de muestreo o discretización de la señal analógica.

1.2.2. El proceso de Conversión Analógica-Digital (Conversión A/D).

El proceso de conversión Analógica-Digital convierte la señal analógica continua (tanto en amplitud como en el tiempo), en series de valores digitales discretos tomando medidas instantáneas de la amplitud de la señal a una velocidad de muestreo generalmente constante.

Existen diversas cuestiones que deben tenerse en cuenta a la hora de convertir una señal analógica en una digital, ya que, si esta conversión no es adecuada, se perderá información existente en la señal original y no existe ningún método para recobrar la información una vez realizado el proceso.

La información se puede perder en frecuencia (debido a una incorrecta frecuencia de muestreo) o bien en amplitud (debido a una inadecuada cuantización de la intensidad de la señal)

1.2.3. El muestreo mediante un tren de impulsos.

Consideremos (Oppenheim, A. y Willsky, A., 1983) el caso general del muestreo representado en la figura 1.4, en el que, mediante un tren de impulsos *p(t)* se muestrea una señal continua en el tiempo *x(t)*.

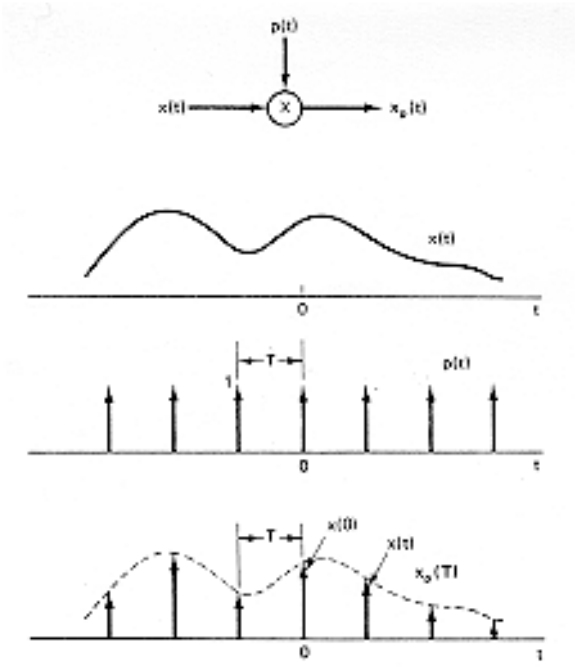


Figura 1.4a *Muestreo de la señal x(t) mediante el tren de impulsos p(t)*

Figura 1.4b *La señal x(t)*

Figura 1.4c *El tren de impulsos p(t)*

Figura 1.4d *La señal muestreada $x_p(t)$*

Figura 1.4: *El proceso de muestreo mediante tren de impulsos en el tiempo(Oppenheim, A., Willsky, A., 1994)*

El tren de impulsos *p(t)* o función de muestreo tendrá una ecuación de la forma siguiente:

$$p(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nT) \qquad [1.1]$$

En el dominio del tiempo tendremos una función $x_p(t) = x(t) \cdot p(t)$, que será un tren de impulsos cuyas amplitudes serán iguales a las muestras de *x(t)* en intervalos de amplitud *T*, esto es:

$$x_p(t) = \sum_{n=-\infty}^{+\infty} x(nT)\ \delta(t - nT) \qquad [1.2]$$

Sabiendo que, según la propiedad de la modulación, la multiplicación en el dominio del tiempo corresponde a la convolución en el dominio de la frecuencia, podemos escribir:

$$X_p(\omega) = \frac{1}{2\pi}\left[X(\omega) * P(\omega)\right] \qquad [1.3]$$

La transformada de Fourier del tren de impulsos, *p(t)*, es decir, la función de muestreo en el dominio de la frecuencia, será:

$$P(\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{+\infty} \delta(\omega - k\omega_s) \qquad [1.4]$$

Observamos que, debido a la propiedad de dualidad, esta función que en el tiempo es un tren de impulsos ponderados e igualmente espaciados, tendrá una Transformada de Fourier que es periódica en frecuencia.

De manera que

$$X_p(\omega) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} X(\omega - k\omega_s) \qquad [1.5]$$

Lo que significa que $X_p(\omega)$ es una función periódica en el dominio de la frecuencia que consiste en una suma de réplicas de $X(\omega)$ desplazadas y escaladas por *1/T*, como se representa en la figura 1.5.

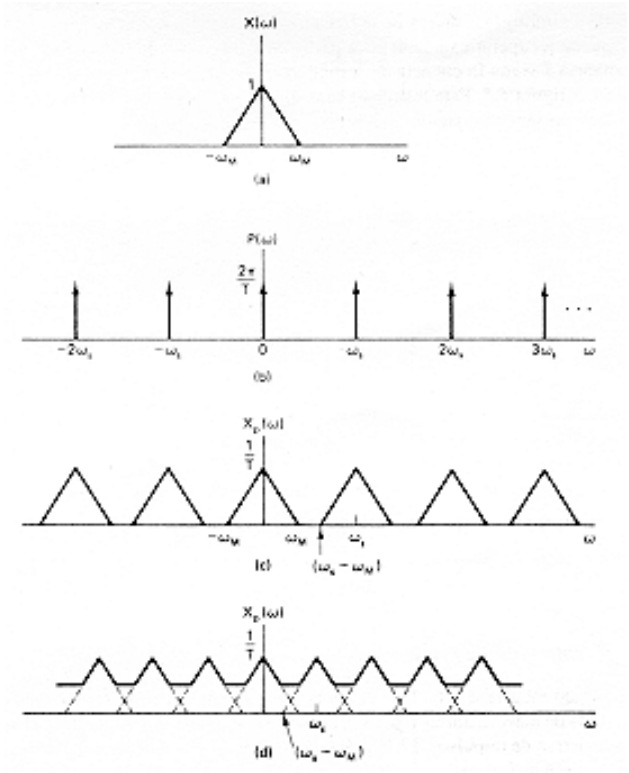


Fig. 1.5a *El espectro en frecuencia X(ω) de una señal cualquiera x(t).*

Fig. 1.5b *El espectro en frecuencia P(ω) de un tren de impulsos p(t).*

Fig. 1.5c *El espectro en frecuencia de la multiplicación de x(t) y p(t), o convolución de X(ω) con P(ω), si* $\omega_S > 2\omega_M$.

Fig. 1.5d *El espectro en frecuencia de la multiplicación de x(t) y p(t), o convolución de X(ω) con P(ω), si* $\omega_S < 2\omega_M$.

Figura 1.5: *El proceso de muestreo en el dominio de la frecuencia(Oppenheim, A., Willsky, A., 1994)*

1.2.4. El Teorema del Muestreo.

En ausencia de cualquier condición o información adicional, no es de esperar que una señal continua en el tiempo pueda ser representada o reconstruida completamente a partir de una señal discreta en el tiempo. Es decir, en principio, la discretización de una señal continua comporta una pérdida de la información que transporta, ya que, en general, hay una cantidad infinita de señales continuas que pueden generar un conjunto dado de muestras (Oppenheim, A y Willsky, A., 1983).

Por ejemplo, para las tres diferentes señales continuas en el tiempo de la figura 1.6, los valores en múltiplos enteros de T son idénticos, es decir $x_1(t)=x_2(t)=x_3(t)$.

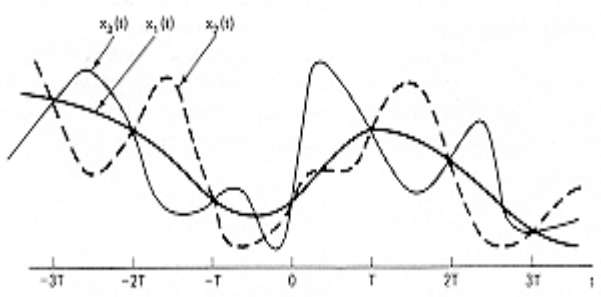


Figura 1.6: *Los valores de las tres señales son idénticos para los múltiplos de T(Oppenheim, A., Willsky, A., 1994)*

Sin embargo, si las medidas se toman a una velocidad de muestreo adecuada, en relación a la frecuencia más alta presente en la señal, de modo que la variación de la amplitud no sea excesiva, el resultado representará unívocamente a la señal y podremos reconstruirla perfectamente.

La frecuencia de muestreo necesaria para asegurar la correcta discretización de una señal continua en el tiempo viene dada por el Teorema del Muestreo, y se denomina frecuencia de Nyquist.

(Shannon, C.E., 1949) demostró que cualquier señal continua en el tiempo *x(t)*, cuyo espectro de frecuencias esté limitado a un ancho de banda, puede representarse sin pérdida de información, como una serie de muestras *x[n]* ó *x[nT]*, de la señal original, es decir, como una señal discreta en el tiempo si $\omega_S > 2\omega_M$, siendo ω_M la frecuencia más alta presente en la señal continua, con lo que $\omega = 2\omega_M$ es la frecuencia de Nyquist.

La conclusión del Teorema del Muestreo o de Shannon se pone de manifiesto en las figuras 1.4 y 1.5.

Si consideramos una señal real cualquiera *x(t)* en el dominio del tiempo, su espectro en el dominio de la frecuencia será $X(\omega)$, y consideramos un tren de impulsos *p(t)*, cuyo espectro en frecuencia será $P(\omega)$, y cuya frecuencia es ω_S, el resultado de su multiplicación en el tiempo, $x_p(t)$, o bien su convolución en la frecuencia, $X_p(\omega)$, en suma, de su modulación, poseerá una frecuencia más alta (y más baja, ya que será simétrico), a la que denominamos ω_M.

También observamos en la figura 1.5 que $X_p(\omega)$ es una función periódica en el dominio de la frecuencia que consiste en la suma de réplicas de $X(\omega)$ desplazadas y escaladas por $\omega_S = 1/T$.

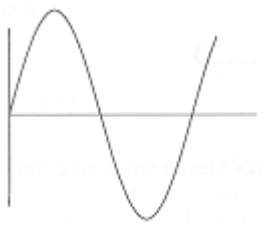


Figura 1.7: *Onda sinusoidal*

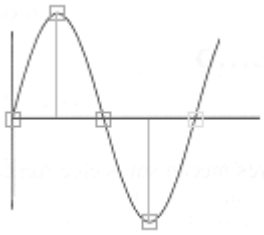


Figura 1.8: *Sobremuestreo*

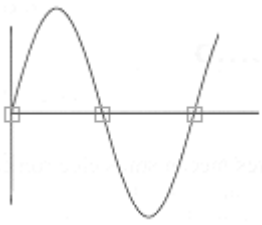


Figura 1.9 *Submuestreo*

En la figura 1.8 se puede apreciar que, en el caso de que $\omega_S > 2\omega_M$, (figura 1.5c) las réplicas de $X(\omega)$ estarán perfectamente separadas unas de otras, llamándose sobremuestreo al muestreo en estas condiciones, sin embargo, si $\omega_S < 2\omega_M$, (figura 1.5d) las réplicas se solapan con sus adyacentes, falseando la información almacenada en el dominio de la frecuencia y dándose el efecto de submuestreo o solapamiento.

Si el solapamiento es pequeño, es decir, si la diferencia entre ω_S y $2\omega_M$ no es grande, el efecto impedirá, en general, la correcta reconstrucción de la señal temporal a partir de su espectro de frecuencias (es posible esta operación bajo ciertas

condiciones), pero si el solapamiento es grande, será incluso audible la pérdida de información (en el caso de una señal acústica).

El análisis de una onda, por muy compleja que ésta sea, mediante la transformada de Fourier, la descompondrá en ondas sinusoidales, que, sumadas, darán como resultado la onda compleja original. La frecuencia más alta de la onda, ω_M, será la del armónico de mayor frecuencia, supongamos que, para una onda compleja, éste sea el de la figura 1.7.

Si muestreamos esta onda con un tren de impulsos de frecuencia mayor que la frecuencia de la onda a muestrear, por ejemplo $\omega_S = 4\omega_M > 2\omega_M$, como se representa en la figura 1.8, conseguiremos una onda bastante aproximada a la continua en el tiempo. Efectivamente, la propiedad fundamental de la onda, es decir, su periodicidad, se puede estudiar en la onda muestreada.

Sin embargo, si muestreamos la onda sinusoidal con una función de muestreo de frecuencia $\omega_S = 2\omega_M$, como en la figura 1.9, que sería el límite en el que no se cumple el Teorema de Muestreo, (es decir, la frecuencia de muestreo más alta para la que aún no se cumple el Teorema), el resultado no sólo no se parece a la onda original sino que (en el ejemplo) sería una señal continua y nula, con lo cual no podríamos estudiar ninguna propiedad de la onda original.

1.3. Modelado de señal. Modelo de predicción lineal.

Un modelo clásico del sistema de producción de voz sería el siguiente (Atal, B. S., 1976):

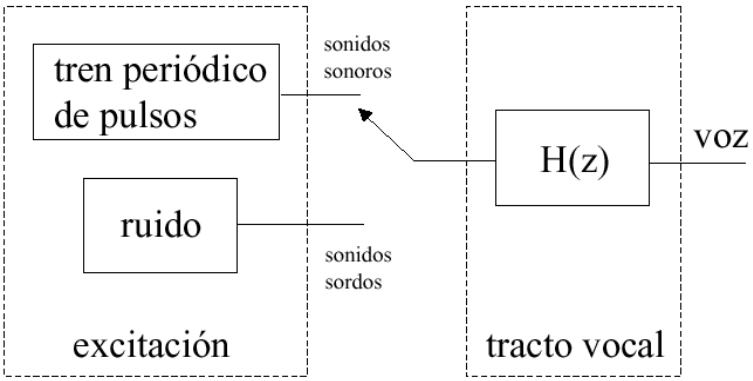


Figura 1.10: *Modelo de producción de voz*

La obtención de un modelo de un proceso a estudiar nos permite describir el comportamiento del proceso mediante expresiones sencillas, prescindiendo de información no útil para el estudio. Esto nos permitirá trabajar más fácilmente con el.

Los modelos de predicción lineal son modelos paramétricos que consideran que la señal de salida del proceso a estudiar es la señal de salida s_n de un filtro de las características anteriores con una entrada desconocida u_n, que generalmente se considera un impulso unidad o un ruido blanco.

La salida del filtro lineal en un instante n, y dada una entrada u_n, es:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G\sum_{l=0}^{q} b_l u_{n-l} \quad \text{con } b_0=1 \qquad [1.6]$$

Donde a_k ($1 \le k \le p$), b_l ($1 \le l \le q$) y G (factor de ganancia) son los parámetros del modelo.

Es decir, la salida en un instante determinado, s_n, es una combinación lineal de p salidas anteriores, q entradas anteriores y la entrada en ese mismo instante n. Esto es, la salida actual es predecible mediante una combinación lineal de salidas y entradas anteriores y la entrada actual.

Se puede expresar el modelo en el dominio de la frecuencia tomando la transformada Z a ambos miembros de la ecuación [1.6], se obtiene la función de transferencia del sistema:

$$H(z) = \frac{S(z)}{U(z)} = G\frac{1+\sum_{l=1}^{q} b_l z^{-l}}{1+\sum_{k=1}^{p} a_k z^{-k}} \qquad [1.7]$$

Siendo $S(z)$, $U(z)$ las transformadas z de la salida y la entrada, respectivamente, es decir:

$$S(z) = \sum_{n=-\infty}^{\infty} s(n) z^{-n} \quad \text{y} \quad U(z) = \sum_{n=-\infty}^{\infty} u(n) z^{-n} \qquad [1.8]$$

Las raíces del polinomio numerador de la función de transferencia 1.7 se denominan ceros del sistema y las raíces del polinomio denominador se denominan polos del sistema.

Existen dos casos particulares interesantes de la expresión 1.7 (Makhoul, 1976):

1- Cuando $a_k=0$ ($1 \le k \le p$). Entonces el modelo se llama Modelo Todo-Cero o Media Móvil (en la literatura puede encontrarse abreviado como modelo MA).
2- Cuando $b_l=0$ ($1 \le l \le q$). Entonces el modelo se llama Modelo Todo-Polo o Autorregresivo (en la literatura puede encontrarse abreviado como modelo LPC o también como modelo AR).

Habitualmente el tracto vocal ha sido modelizado para sonidos vocálicos no nasales (señales cuasi-estacionarias) mediante el modelo LPC (Linear Prediction Coefficient). El modelo LPC modeliza el tracto vocal como un tubo acústico dividido en distintas secciones excitado. Este modelo resulta poco adecuado en muy diversas

situaciones en las que se requiere representar las contribuciones del flujo glotal, el tracto vocal y las radiaciones. Así mismo existen sonidos como los fricativos, en los que la excitación no se produce al inicio del tracto, y sobre todo en las transiciones entre consonantes nasales y vocales, en las que en la cavidad nasal se producen una serie de resonancias que no pueden ser caracterizadas adecuadamente con los formantes (polos de la función de transferencia). Concretamente en las transiciones consonante nasal/vocal, cierra la cavidad bucal y la cavidad nasal actúa como resonador. La aparición de este efecto resonador es lo que provoca que el modelo todo-polo (LPC) para la función de transferencia no sea satisfactorio en estas situaciones y sea necesario utilizar un modelo mixto polo-cero o más formalmente ARMA(p.q), donde el parámetro p representa el número de polos (o también llamados formantes) y q el número de ceros del modelo (en ocasiones suelen recibir el nombre de antiformantes) (Atal, B. S., 1971).

Por tanto para obtener una buena parametrización de la señal acústica en todo tipo de sonidos procedentes del habla natural, como punto de partida para el desarrollo de un sistema de reconocimiento de locutores automático, es necesario utilizar un modelado ARMA. Como inconveniente debemos indicar que la utilización de este tipo de análisis conlleva un mayor coste computacional, ya que la resolución de los sistemas LPC nos lleva a un sistema de ecuaciones lineales, fácilmente resoluble, mientras que los sistemas ARMA dan lugar a sistemas no lineales cuya resolución es más compleja.

1.4. Segmentación. Métodos de la dinámica no lineal

La segmentación representa una fase previa prácticamente imprescindible en los sistemas de reconocimiento. Entenderemos por segmentación: *La división de las señales en partes discretas, que pueden ser asociadas con un evento específico de los que componen las señales.* Un sistema de segmentación de señales del habla interpretado como procedimiento para segmentar la señal en partes sostenidas donde las características del sonido presentan una regularidad y en partes de transición donde las características varían con el tiempo; ha de incluir requisitos tales como capacidad de distinguir cambios suficientemente pequeños de las características de los sonidos, las cuales deben de obtenerse fácilmente a partir de la señal original.

Tal y como vimos, en las señales del habla por la propia naturaleza de las mismas el problema de la segmentación presenta grandes dificultades ya que en la producción del habla tiene lugar la aparición de procesos claramente no lineales.

En estudios anteriores se proponen sistemas automáticos empleados en la segmentación de señales del habla en discurso continuo que, en un primer nivel, llevan a cabo la catalogación de tramos temporales de corta duración mediante características de estas señales, fácilmente extraíbles a partir de ellas, como pueden ser: el nivel de amplitud, número de cruces por cero o el pitch (periodo fundamental). La catalogación definitiva de los segmentos se llevaba a cabo a través de realizar comparaciones entre segmentos adyacentes, tomando como medida de similitud entre ellos la distorsión LPC o bien la distorsión ARMA según usemos un tipo de modelos u otros. Estos sistemas dan buenos resultados en tramos temporales correspondientes a sonidos sostenidos, sin embargo su eficiencia se veía disminuida en fragmentos próximos a las transiciones entre distintos fonemas. Debido a esto, se plantea la necesidad de emplear técnicas basadas en la teoría de la dinámica no lineal.

En este proyecto se presentan diversos métodos de dinámica no lineal, los cuales aplicados a series temporales en general y señales del habla en particular, permiten extraer información que cuantifica la regularidad y los cambios de complejidad de la dinámica del sistema analizado. Estos cambios en la complejidad del sistema nos servirán para caracterizar transiciones entre fonemas. En los tramos de señal que correspondan a un mismo fonema la complejidad y aleatoriedad de la señal se mantendrá estable. Una vez comience la transición esta variará (aumentará o disminuirá), hasta llegar a un nuevo tramo de fonema sostenido, en el que la complejidad y aleatoriedad de la señal volverá a ser estable.

El primero de estos métodos no lineales que presentamos es el cálculo de la entropía de la señal. La entropía es un concepto que especifica la aleatoriedad y predictibilidad de los sistemas. De los diversos algoritmos para el cálculo de la entropía hemos implementado dos: algoritmo de la K_2-Entropía o entropía de Renyi y el algoritmo de la entropía aproximada (además se implemento un algoritmo para el cálculo rápido de la entropía aproximada). El segundo de los métodos será el cálculo de la dinámica simbólica de una serie temporal. Con este algoritmo transformamos una serie temporal en una serie de símbolos. De la secuencia de símbolos resultante se puede extraer características no lineales de la serie temporal original. Este algoritmo tiene la ventaja de requerir un coste computacional inferior al necesario para el cálculo de la entropía por cualquiera de los métodos presentados.

1.5. Reconocimiento de locutores

Todos somos conscientes de que las voces de distintos individuos no suenan de igual forma. Esta propiedad del habla (la dependencia del locutor) queda de manifiesto si observamos la figura 1.11 que representan la señal que generan tres locutores distintos para un mismo vocablo /oma/. Como vemos las señales son distintas en cada uno de los casos.

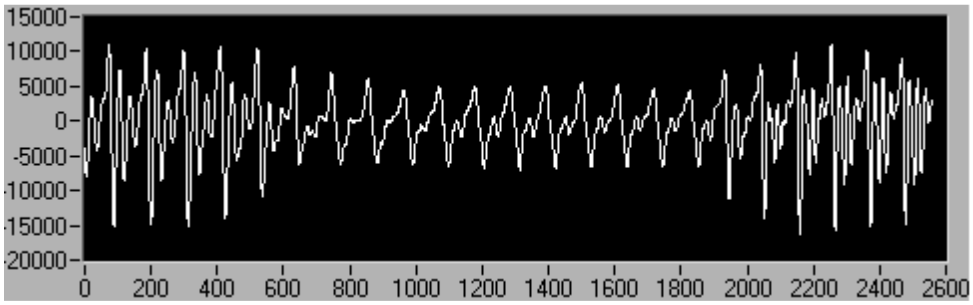


Figura 1.11 a: *Señal del fonema /oma/ locutor 1*

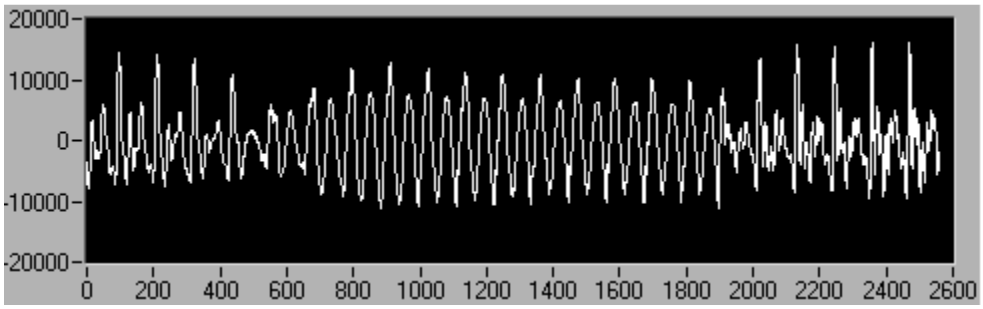


Figura 1.11b: *Señal del fonema /oma/ locutor 2*

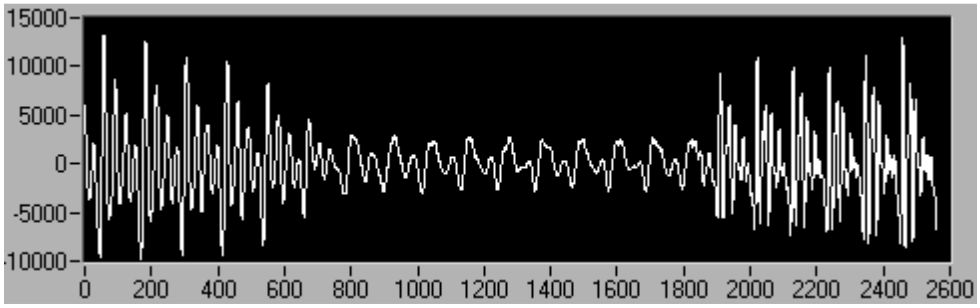


Figura 1.11c: *Señal del fonema /oma/ locutor 3*

La habilidad de reconocer a una persona solamente por su voz es a lo que llamamos reconocimiento de locutores (Klevans, R. y Rodman, R., 1997). Todo sistema de reconocimiento de locutores puede ser dividido en dos etapas: medición y clasificación. En la primera etapa, se realizan una serie de mediciones sobre una señal del habla de tal forma que nos permita llegar a obtener una serie de parámetros que representen adecuadamente la información dependiente del locutor. En la segunda etapa, a partir de una serie de reglas de decisión asignaremos o verificaremos una identidad a la señal de entrada.

1.5.1. Propiedades deseables de los parámetros usados en el reconocimiento de locutores.

En el reconocimiento de locutores tratamos de obtener la identidad de un locutor a partir de una serie de parámetros obtenidos de su señal del habla. Para ello hemos de buscar aquellos parámetros que presenten propiedades que disminuyan las diferencias intralocutor (que se mantengan inalterables para un mismo locutor) y además exista diferencia interlocutor (propiedades de cada locutor y que lo diferencia de los demás). Las propiedades que se le pueden exigir a estos parámetros son seis (Atal, B. S., 1976):

- Que representen eficientemente la información dependiente del locutor.
- Que sean fáciles de medir.
- Que sean estables con el paso del tiempo
- Que ocurran de manera habitual y natural en el habla.
- Que varíen poco de un entorno del habla a otro.
- Que no sean fácilmente imitables

1.5.2. Verificación de locutores frente a identificación de locutores

Hay dos subareas distintas dentro del reconocimiento de locutores: verificación e identificación (Klevans, R. y Rodman, R., 1997)..

En los sistemas de verificación de locutores una identidad es reclamada por el usuario del sistema, y a este se le requiere tomar una decisión estrictamente binaria, aceptar o rechazar la identidad reclamada.

El problema de la identificación de locutores difiere bastante del de la verificación. En este caso al sistema se le pide que clasifique la señal de entrada como perteneciente a alguno de los posibles locutores.

1.5.3. Modelo de sistema de reconocimiento de locutores

El esquema general de un sistema de reconocimiento de locutores lo tenemos en la figura 1.12 (Rabiner, L. y Schafer R., 1978). Tal y como vemos en una primera etapa la señal es procesada. Nos quedamos con aquellas características de la señal que nos interesa de cada locutor y que lo hace diferente de los demás y realizamos un modelo de él. En la siguiente etapa este modelo es comparado con uno o varios de los modelos de referencia de los locutores que tenemos almacenados en memoria y se calcula la distancia que hay entre la señal de entrada y cada uno de ellos. En la última etapa mediante una regla de decisión, se asigna una identidad a la señal de entrada.

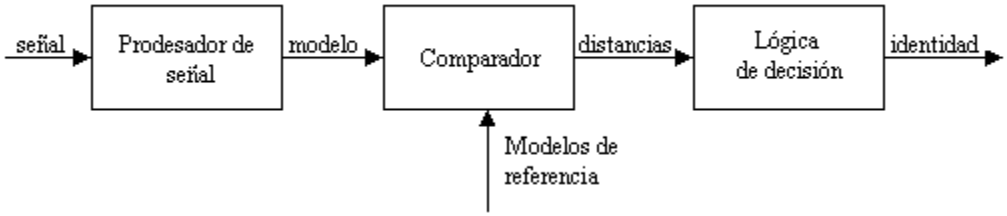


Figura 1.12: *Modelo de sistema de reconocimiento de locutores*

A partir de este modelo es fácil llegar a la conclusión de que existen tanto un gran número de similitudes, como de diferencias, entre los sistemas de verificación e identificación de locutores.

Como vemos la primera etapa es común tanto para la verificación como para la identificación de locutores. En ella las técnicas de procesado de señal empleadas para la extracción de parámetros y la creación del modelo son las mismas. Así mismo, las técnicas de comparación empleadas para establecer la similitud entre modelos también son las mismas.

Las diferencias las encontramos en el número de comparaciones que tendremos que realizar y la decisión lógica que debe tomar según se trate de un sistema de verificación o identificación de locutores.

En los sistemas de verificación de locutores una identidad es reclamada por el usuario del sistema, y a este se le requiere tomar una decisión estrictamente binaria, aceptar o rechazar la identidad reclamada. Para tomar esta decisión, sólo es necesaria una comparación entre el modelo del registro de entrada y el patrón de referencia del locutor. El problema de la identificación de locutores difiere bastante del de la verificación. En este caso al sistema se le pide que clasifique la señal de entrada como perteneciente a alguno de los posibles *N* locutores (Rabiner, L. y Schafer R., 1978).

En este caso se requerirán *N* comparaciones entre el modelo de la señal de entrada y los *N* modelos de referencia de los locutores almacenados en memoria. La

señal de entrada se clasificará como perteneciente a aquel locutor cuyo modelo de referencia sea más similar que el de la señal de entrada (Rabiner, L. y Schafer R., 1978).

1.5.4. Áreas de aplicación del reconocimiento de locutores

La habilidad para identificar a las personas únicamente por su voz puede ser interesante en varias áreas de aplicación, pero sin duda su principal aplicación la encontramos en el área de la seguridad. El acceso a zonas de seguridad actualmente puede estar restringido a teclear un código, tarjetas magnéticas o por medio de combinaciones. Sin embargo, estos dispositivos pueden fallar por diversos motivos: pueden ser manipulados, robados, olvidados…. El reconocimiento de voz puede proporcionar un sistema más seguro para permitir la entrada a áreas de seguridad restringidas. Otra aplicación la encontramos al usar el reconocimiento como prueba pericial en juicios.

1.6. Objetivos del proyecto

Los algoritmos de segmentación del habla han utilizado tradicionalmente criterios basados en parámetros de la señal temporal, tales como la amplitud de la señal, el número de cruces por cero o frecuencia fundamental (*pitch*), y caracterizaciones de la señal mediante el modelo Todo-Polo (o modelo LPC), formulado en el dominio del tiempo o bien de la frecuencia.

Teniendo en cuenta las peculiaridades que se dan en la generación del habla natural (acoplamientos, coarticulación, silencios,...), buscaremos una alternativa a estas técnicas clásicas en métodos basados en la dinámica no lineal. Esto nos llevará a que, necesariamente, primero centremos el estudio en la naturaleza caótica de las señales del habla natural. En este sentido se estudiarán parámetros de la dinámica no lineal entre los que podemos destacar la Dimensión de Correlación, que nos permitirán concluir sobre el citado carácter caótico.

Se desarrollarán diversos algoritmos basados en técnicas no lineales (Entropía Aproximada, K_2-Entropía y Dinámica Simbólica) para la caracterización de transiciones entre fonemas de distinta naturaleza, en concreto aquellas que presentan coarticulación nasal, que suponen por su complejidad una de las situaciones que presentan mayor dificultad para los sistemas de segmentación clásicos.

La parametrización LPC, tal y como se ha podido constatar en anteriores estudios, no es adecuada para la modelización de segmentos transicionales. A tal fin, se implementará un algoritmo de parametrización ARMA basado en el método de Shanks. Teniendo en cuenta que el objetivo último de este proyecto es plantear un estudio de reconocimiento de locutores, proponemos como medida de comparación una métrica ARMA, que se calcula a partir de los coeficientes ARMA utilizando determinantes (en concreto la resultante entre dos polinomios).

Finalmente, realizaremos una comparación entre la técnica clásica de Análisis Discriminante y la nueva técnica introducida, basada en la métrica ARMA, sobre una

población extraída de la Base de Datos Ahumada, desarrollada por la Dirección General de la Guardia Civil. Concretamente se planteará el problema de reconocimiento de locutores trabajando con señales del habla natural en las que está presente el fenómeno de coarticulación nasal, admitiendo la hipótesis de que la coarticulación nasal representa una característica fuertemente dependiente del locutor.

Todos los algoritmos serán implementados en el entorno de programación LabVIEW®, de la empresa *National Instrument*, creando una librería de funciones que podrán ser utilizadas para desarrollar aplicaciones de caracterización y segmentación automática de señales del habla y aplicaciones de reconocimiento de locutores.

Capítulo 2: Fundamentos Matemáticos

Dinámica no lineal
Modelo ARMA
Cepstrum

2. Fundamentos matemáticos.

2.1. Dinámica no lineal

La descripción de un sistema dinámico consiste en dos partes, el estado y la dinámica del sistema. El estado nos da el valor del sistema en un instante de tiempo determinado, mientras que la dinámica es el conjunto de reglas bajo las cuales el estado evoluciona a lo largo del tiempo. El estado de un sistema es formalmente representado por el *vector de estado*. Este vector tiene un tamaño m, donde m es el número de variables necesarias para caracterizar completamente el sistema. Por tanto, el vector de estado representa un punto en un espacio de estados m-dimensional. La dinámica o evolución del sistema con el tiempo es por tanto representada en el espacio estado. Cuando la evolución del sistema converge a una misma zona del espacio estado se dice que existe un atractor. Los sistemas que son continuos, estacionarios y disipativos siempre tienen atractores. Si además al aumentar el tiempo las trayectorias del atractor se separan de las condiciones iniciales y nunca se repiten, se dice que el atractor es extraño, y el sistema analizado es caótico. Los atractores extraños se caracterizan por la dimensión fractal ν, que es una cota inferior de los grados de libertad del sistemas dinámicos (Pritchard, W. y Duke, W., 1995).

En la producción del habla, tal y como vimos, aparecen procesos no lineales: flujo turbulento del aire en el tracto vocal, acoplamiento no lineal entre distintas partes del tracto debido a ordenes neuromusculares, … Para el análisis de sistemas no lineales como este, recurriremos a su representación en el espacio estado (Teodorescu, H. y otros, 1997).

La serie temporal correspondiente a una señal del habla natural nos da una representación unidimensional del sistema. La base teórica que nos permite llevar la serie temporal a una representación de espacio estado de dimensión mayor que nos permita descubrir si existe un atractor, es el *Takens´ embedding theorem* (Takens, F., 1983). En éste se establece, por un lado, la construcción de un atractor topológicamente equivalente al del sistema usando, a partir de la observación de una única componente, y por otro, la independencia de la reconstrucción respecto de la componente utilizada. La aplicación fundamental del teorema de Takens es aproximar estados próximos de la señal llevando a cabo la *reconstrucción dinámica*.

Dada una serie temporal, construimos una sucesión de vectores de R^{d_E} dada por:

$$x(i) = \left[u(i), u(i+\tau), \ldots, u(i+(d_E-1)\tau)\right] \quad \text{para } 1 \le i \le n-(d_E-1)\tau .$$

A continuación aplicamos el teorema de Takens a distintas señales del habla continua. Estas tendrán un tamaño de 512 puntos muestreados a 16000 Hz, es decir, corresponden a 32 ms. Usando este tamaño podemos asumir la estacionalidad de la señal. Al aplicar el teorema de Takens llevaremos cada serie temporal a un espacio

estado bidimensional, es decir, $d_E=2$. El resultado será que representamos $x(i)$ frente a $x(i+\tau)$.

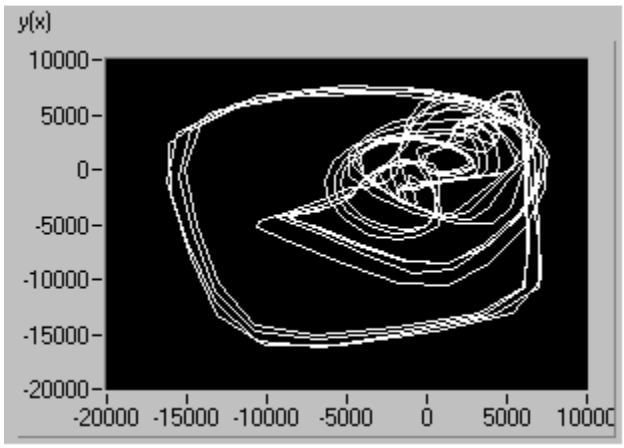


Figura 2.1: *Atractor correspondiente al fonema /a/*

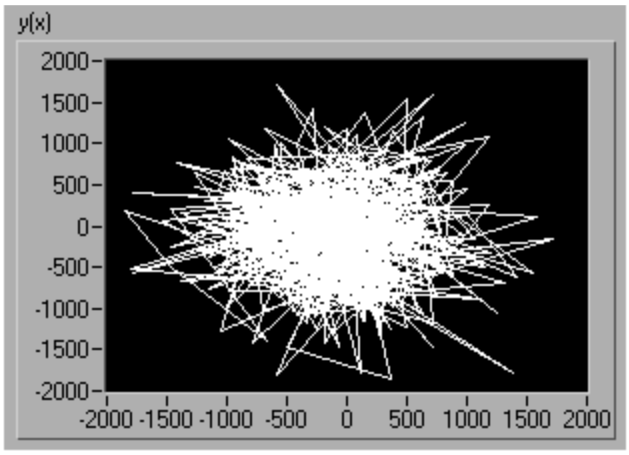


Figura2.2: *Atractor correspondiente al fonema /z/*

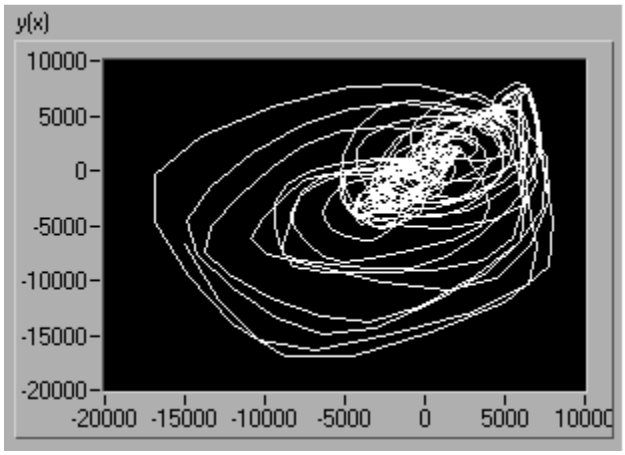


Figura 2.3: *Atractor correspondiente al fonema /m/a/*

Como vemos todas ellas presentan un atractor extraño. Esto es previsible ya que como dijimos anteriormente corresponden a un sistema continuo (señal del habla continua), estacionario (32ms de un mismo evento: sonido sostenido o transición) y además disipativo (conforme avanzamos en el tiempo la energía inicial, el aire procedente de los pulmones, se pierde). Podemos asegurar por tanto que las señales procedentes del habla son caóticas. Además del atractor correspondiente a cada sonido podemos extraer distintas conclusiones:

- El atractor correspondiente al sonido vocálico, presenta unas trayectorias exteriores que inicialmente mantienen un radio similar. Esto nos da una idea de periodicidad en la señal. Por otro lado, en la zona central esta característica no se da. Este cambio podríamos interpretarlo como la distorsión que introduce los procesos no lineales del tracto vocal a la señal periódica procedente de las cuerdas vocales.
- El atractor correspondiente al sonido sordo vemos que no presenta ningún tipo de periodicidad.
- En el atractor correspondiente a la transición entre sonidos sonoros, vemos que existen las órbitas circulares típicas de estos sonidos, pero estas no mantienen su radio en ningún momento (avanzamos en la transición).

2.1.1. Dimensión de correlación

La dimensión de correlación es una estimación de la dimensión fractal del atractor, más concretamente $D_c \le \upsilon$, por lo que D_c es una cota inferior de los grados de libertad de la serie temporal y, en este sentido, puede interpretarse como una medida de la complejidad del sistema dinámico analizado (Grassberger, P. y Procaccia, I., 1983). A su vez nos da una idea del valor de d_E, ya que el teorema de Takens aporta una condición suficiente para d_E, que es: $d_E \ge 2\upsilon + 1$. Teniendo en cuenta que la dimensión de correlación es una cota inferior de la dimensión del atractor, resulta $d_E \ge 2D_c + 1$.

Para $\varepsilon \in R^+$ se define la *dimensión de correlación*, (Grassberger, P. y Procaccia, I., 1983). como:

$$D_c = \lim_{\varepsilon \to 0} \frac{\ln C_{d_E}(\varepsilon)}{\ln \varepsilon} \qquad [2.1]$$

donde:

$$C_{d_E}(\varepsilon) = \lim_{n \to \infty} \frac{\{N^o \quad pares \quad (i,j) \quad / \quad \|x(i) - x(j)\| \le \varepsilon\}}{n^2} \qquad [2.2]$$

recibe el nombre de *correlación acumulada o integral*.

Esta definición es la base para el desarrollo del algoritmo para la computación de la dimensión de correlación en una serie temporal. Para ello se define la correlación acumulada como:

$$C_{d_E}(\varepsilon) = \lim_{n\to\infty} \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1, i\neq j}^{n} H\left(\varepsilon - \left\|x(i) - x(j)\right\|\right) \qquad [2.3]$$

donde H es la función de Heaviside, dada por

$$H(x) = \begin{cases} 0, & x \leq 0 \\ 1, & en \quad otro \quad caso \end{cases}$$

y la norma $\|\cdot\|$ utilizada es la norma del supremo.

En la siguiente figura aparece la evolución de la dimensión de correlación a lo largo de un registro que corresponde a una transición con coarticulación nasal, concretamente a la transición m~a. En ella podemos apreciar como en la parte inicial de la señal el valor de la dimensión de correlación es menor que en la parte final. Este comportamiento es coherente con lo expuesto anteriormente, ya que si bien en toda la señal se puede apreciar periodicidad, el tramo inicial que corresponde al sonido /m/ es bastante más regular que el final, con lo que es de suponer que su dimensión fractal sea inferior.

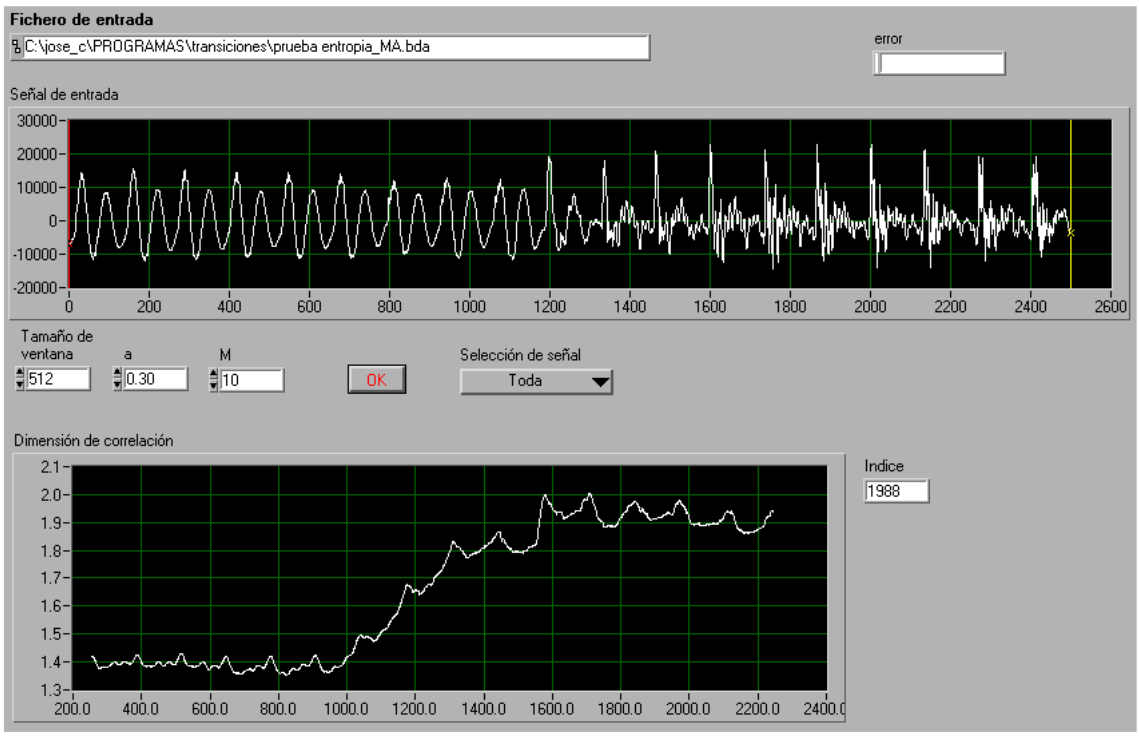


Figura 2.4: *Evolución de la dimensión de correlación en el registro /ma/*

2.1.2. Entropía

Un método para determinar si una señal es caótica (en lugar de multiperiódica o aleatoria) y poder caracterizarla es el cálculo de su entropía. La entropía es una medida que establece la posibilidad de predicción de los valores de amplitud subsiguientes de una serie temporal, a partir del conocimiento de valores previos de la misma. Así, en una serie temporal perfectamente regular, el conocimiento de valores previos, hace posible la predicción de los valores subsiguientes, siendo en tal caso el valor de la entropía cero. Conforme aumenta la irregularidad en la serie temporal, aún conociendo valores previos, la predicción del valor subsiguiente podría ser errónea, aumentando el valor de la entropía. En una serie temporal enteramente irregular, el conocimiento de valores previos no permite la predicción del valor subsiguiente, dependiendo el valor de la entropía del número de muestras y del número de valores previos utilizados para la predicción del valor (Jörgen, M., 2000).

De entre las posibles formulaciones de la entropía, en este proyecto nos hemos centrado en la Entropía Aproximada. El algoritmo de cálculo de la Entropía Aproximada se recoge en el capítulo 5, y los resultados obtenidos en su aplicación sobre distintas señales del habla natural se exponen en el capítulo 6.

No se ha querido dar por finalizado el presente proyecto sin llevar a cabo una primera aproximación a la formulación general del concepto de entropía de Kolmogorov, que nos aporta la Entropía de Renyi de orden q; siendo la Entropía Aproximada un caso particular de la misma. En concreto, vamos a centrar todos los desarrollos en la Entropía de Renyi de orden q=2 o K_2-entropía.

2.1.2.1. K_2-entropía o entropía de Renyi de orden q=2

Una medida no lineal utilizada en el análisis de series temporales, es la *K_2-entropía*, que se introduce como un método para estimar directamente la *entropía de Kolmogorov,* K, de una señal temporal (Grassberger, P. y Procaccia, I., 1983). La entropía de Kolmogorov es definida de la siguiente forma: Consideremos un sistema dinámico con *F* grados de libertad, supongamos que el espacio-estado de dimensión *F* es dividido en cajas de tamaño ε^F. Además existe un atractor en el espacio-estado y la trayectoria $\vec{x}(t)$ está dentro de la zona de atracción. A continuación medimos el estado del sistema a intervalos de tiempo τ. Sea $p(i_1, i_2, \ldots, i_d)$ la posibilidad conjunta de que $\vec{x}(t=\tau)$ esté en la caja i_1, $\vec{x}(t=2\tau)$ este en la caja i_2,... y $\vec{x}(t=d\tau)$ esté en la caja i_d. La entropía de Kolmogorov se puede expresar entonces como:

$$K = -\lim_{\tau\to 0}\lim_{\varepsilon\to 0}\lim_{d\to\infty}\frac{1}{d\tau}\sum_{i_1\ldots i_d} p(i_1\ldots i_d)\times\ln p(i_1\ldots i_d) \qquad [2.4]$$

En un sistema ordenado $K \cong 0$, y en un sistema aleatorio *K* es infinita. Sin embargo, *K* es una constante distinta de cero en un sistema caótico (determinístico). Es muy fácil estimar el valor de *K* a partir de las ecuaciones que describen la evolución

del sistema entre dos puntos, sin embargo estimar el valor de *K* directamente desde una señal medida en el tiempo es muy difícil.

Definimos una nueva cantidad K_2, la cual presenta las siguientes propiedades:

i. $K_2 \geq 0$
ii. $K_2 < K$
iii. K_2 es infinita para sistemas aleatorios
iv. $K_2 \neq 0$ para sistemas caóticos

Si $K_2>0$, tendremos una condición suficiente para asegurar que la señal a partir de la cual calculamos K_2 es caótica. Sin embargo, la propiedad más importante de K_2, de cualquier modo, será que pueda ser extraída fácilmente de una señal experimental.

Para ver como se llega a esta cantidad, consideremos ahora el grupo de entropías de *Renyi de orden q* definidas como sigue:

$$K_q = -\lim_{\tau\to 0}\lim_{\varepsilon\to 0}\lim_{d\to\infty}\frac{1}{d\tau}\frac{1}{q-1}\ln\sum_{(i_1,\ldots,i_d)} p^q(i_1,\ldots,i_d) \qquad [2.5]$$

Escribiendo $p^q=p$ exp[(q-1) ln p] y utilizando el desarrollo en serie del exponente, es fácil ver que $lim_{q\to 1^+} K_q = K$ y $lim_{q\to 0^+} K_q \leq h$. Además es fácil observar que $K_q > K_{q'}$ para todo $q'> q$.

De todas las cantidades K_q, se elige K_2 ya que es fácil de calcular para series temporales. A continuación, consideremos la ecuación [2.5] para un valor fijo de *d* (por ejemplo, *d=1*) y un valor fijo ε. Para *q=2* y *d=1*, necesitamos calcular $\sum_i p_i^2 = C(\varepsilon)$, donde p_i es la probabilidad que tenemos de que parte del atractor se encuentre en la caja i-esima y el sumatorio incluiría a todas las cajas en el espacio-fase que contienen una parte del atractor. Esta cantidad es fácil de calcular en una serie temporal. Consideremos la serie temporal $\{\vec{X}_i\}_{i=1}^N$ donde $\vec{X}_i = \vec{X}(t = i\tau)$. A partir de un factor independiente ε que define el tamaño de las cajas calculamos la cantidad *C(ε),*

$$C(\varepsilon) = \lim_{N\to\infty}\frac{1}{N^2}\,[n^o \text{ de pares de puntos (n,m) con } \left|\vec{X}_n - \vec{X}_m\right| < \varepsilon\,] \qquad [2.6]$$

Despreciando un factor de escalado podemos aproximar *C(ε)* como

$$C(\varepsilon) \sim \varepsilon^{D_c} \qquad [2.7]$$

donde D_c es el exponente de correlación y tiene la propiedad de ser una cota inferior de la dimensión fractal ν del atractor, es decir, $D_c \leq \nu$.

Si generalizamos lo desarrollado anteriormente para cualquier *d*, podemos considerar ahora,

$$\tilde{C}(\varepsilon) = \lim_{N\to\infty} \frac{1}{N^2} [\textit{número de pares de puntos (n,m) con } \left[\left(\vec{X}_n - \vec{X}_m\right)^2 + \left(\vec{X}_{n+1} - \vec{X}_{m+1}\right)^2 + \cdots + \left(\vec{X}_{n+d-1} - \vec{X}_{m+d-1}\right)^2\right]^{1/2} < \varepsilon] \qquad [2.8]$$

con d=*2,3,…*

$$\tilde{C}_d(\varepsilon) \cong \sum_{(i_1,\ldots,i_d)} p^2(i_1,\ldots,i_d)$$

Consecuentemente , las ecuaciones [2.5] y [2.7] llevan a

$$\tilde{C}_d(\varepsilon) \underset{d\to\infty}{\sim} \varepsilon^{\nu} \exp(-d\tau K_2) \qquad [2.9]$$

En la práctica no necesitamos seguir la evolución de todos los grados de libertad. Generalmente, la trayectoria entera puede ser reconstruida desde d mediciones $(d \geq F)$ de alguna coordenada sencilla. Tomando una coordenada sencilla y denotándola como *X*, consideramos entonces

$$C(\varepsilon) = \lim_{N\to\infty} \frac{1}{N^2} [\textit{nº de pares de puntos (n,m) con } \left(\sum_{i=1}^{d} \left|\vec{X}_{n+i} - \vec{X}_{m+i}\right|^2\right)^{1/2} < \varepsilon] \qquad [2.10]$$

y esperar que de la misma estimación que $C_d(\varepsilon) \sim \varepsilon^{\nu} \exp(-d\tau K_2)$.

Podemos definir la cantidad

$$K_{2,d}(\varepsilon) = \frac{1}{\tau} \ln \frac{C_d(\varepsilon)}{C_{d+1}(\varepsilon)} \qquad [2.11]$$

Calculando su límite obtendremos la estimación de la *K_2-entropía*

$$\lim_{\substack{d\leftarrow\infty \\ \varepsilon\to 0}} K_{2,d}(\varepsilon) \sim K_2 \qquad [2.12]$$

$$\lim_{\substack{d\to\infty\\ \varepsilon\to 0}} \frac{1}{\tau}\ln\frac{C_d(\varepsilon)}{C_{d+1}(\varepsilon)} = \lim_{\substack{d\to\infty\\ \varepsilon\to 0}} \frac{1}{\tau}\ln\frac{\varepsilon^{\nu}\cdot e^{-d\tau K_2}}{\varepsilon^{\nu}\cdot e^{-(d+1)\tau K_2}} = \lim_{\substack{d\to\infty\\ \varepsilon\to 0}} \frac{1}{\tau}\ln e^{\tau K_2} = K_2$$

Elección de parámetros

En la actualidad estamos trabajando en la elección de los valores óptimos de l0os parámetros para la aplicación de la K_2-Entropía a señales del habla natural. En este sentido, presentamos algunos resultados preliminares que hemos obtenido en relación a los parámetros *d* y ε.

Mediante el algoritmo que presentamos en el capítulo 5, calculamos la *K_2-entropía* para distintos valores de los parámetros ε, ($\varepsilon = \alpha * SD$, con $\alpha \in$ *{0.2, 0.3,0.4,0.5}*) y d_E (*d=1, 2, …, 11*), para diferentes sonidos del habla. Los resultados obtenidos los presentamos en las siguientes figuras:

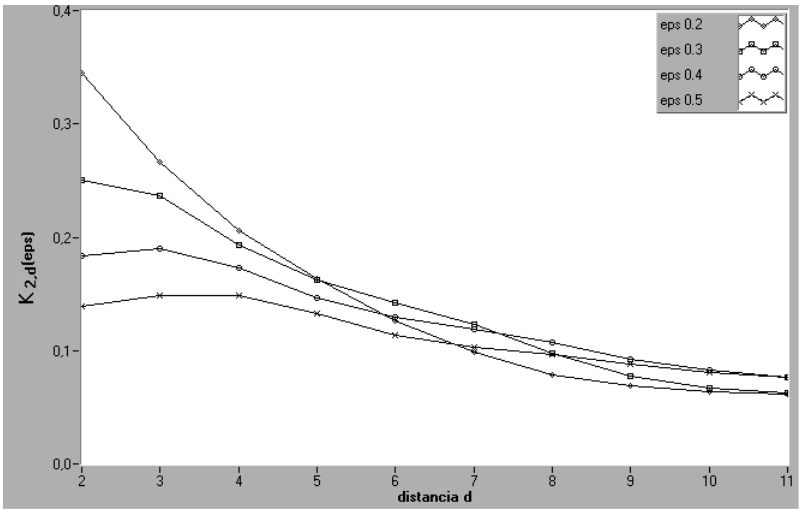


***Figura 2.5**: Evolución de le Entropía de Renyi frente a d para el sonido /m/*

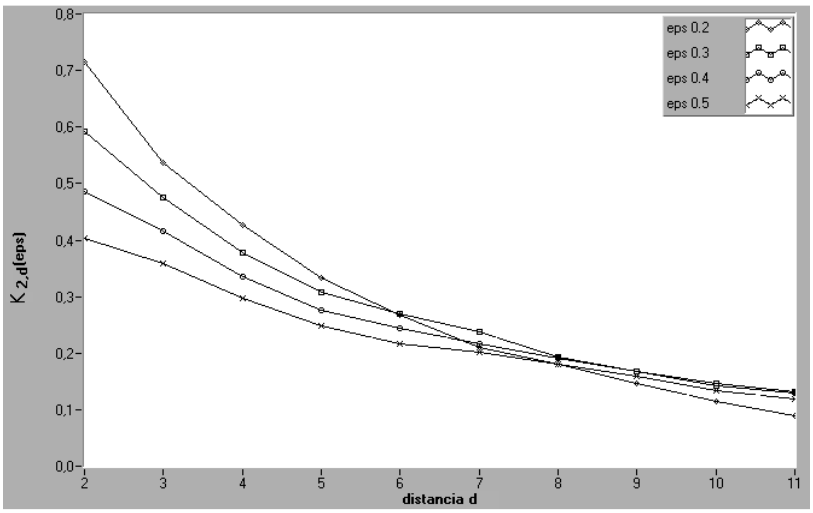


***Figura 2.6**: Evolución de le Entropía de Renyi frente a d para el sonido /a/*

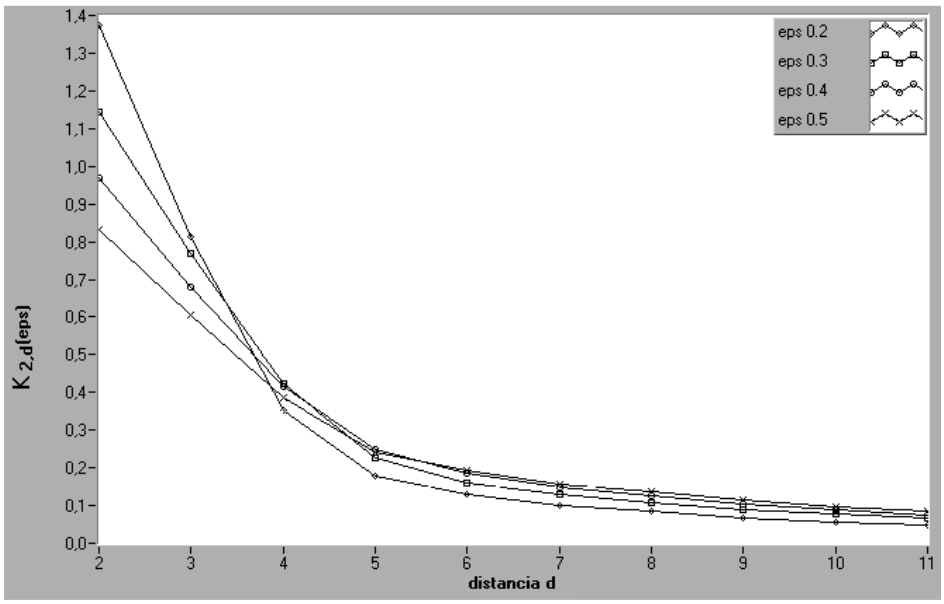


***Figura 2.7**: Evolución de le Entropía de Renyi frente a d para el sonido /s/*

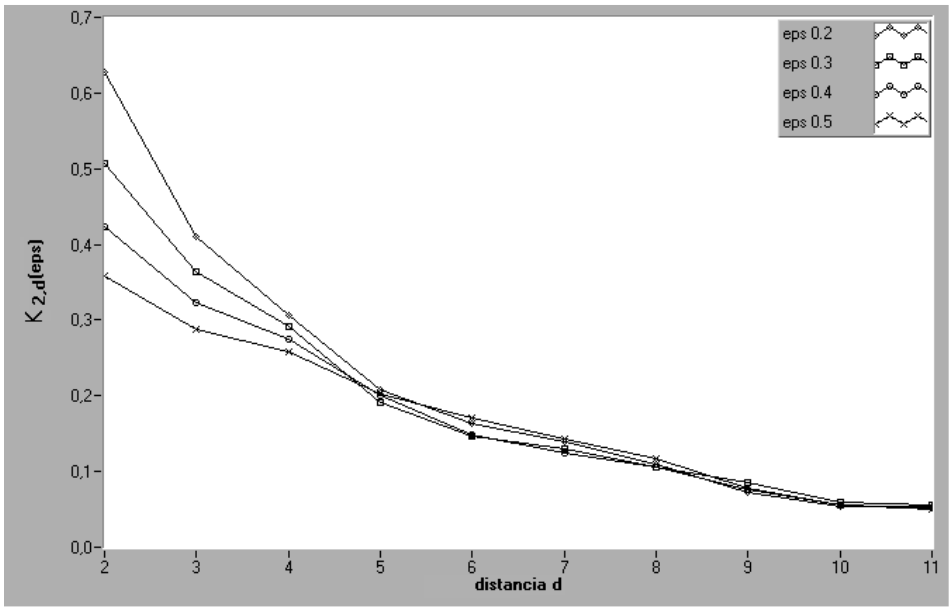


***Figura 2.8**: Evolución de le Entropía de Renyi frente a d para el sonido /i/*

A partir de los resultados presentados, podemos llegar a la conclusión de que tomando vectores de *10* componentes (*d=10*), los valores de entropía se mantienen en mayor o menor medida constantes, luego este será el valor óptimo. A su vez, observamos que para estos valores de *d*, el valor de la entropía no se ve afectado por el valor α, por el cual multiplicamos la desviación típica (*SD)*. Tomaremos como valor óptimo *0.3*.

Finalmente aplicamos el algoritmo de cálculo a una señal del habla natural, concretamente una transición con coarticulación nasal, aplicando los valores óptimos (*d=10* y α*=0.3*) obtenidos. El resultado lo tenemos en la siguiente gráfica:

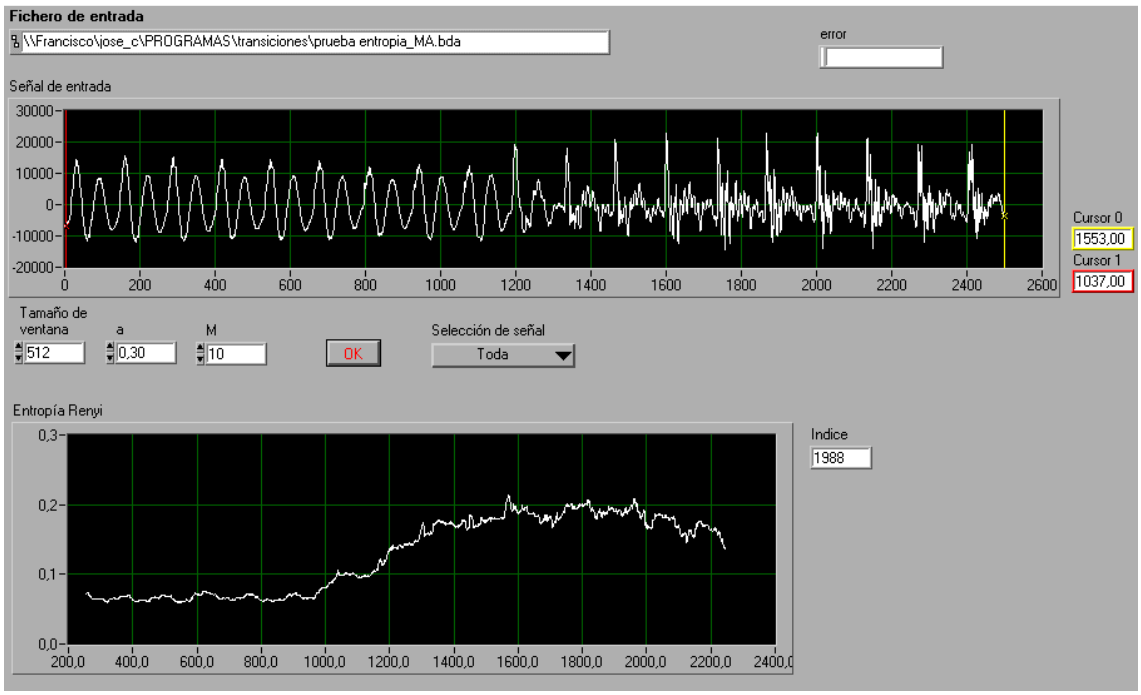


Figura 2.9: *Evolución de la entropía de Renyi en el registro /ma/*

En ella podemos apreciar como en la parte inicial de la señal el valor de la entropía es menor que en la parte final. Esto corresponde con el comportamiento que inicialmente decíamos iba atener la entropía, es decir, en una serie temporal regular (tramo inicial de la señal), el conocimiento de valores previos, hace posible la predicción de los valores subsiguientes, siendo en tal caso el valor de la entropía próximo a cero. Conforme aumenta la irregularidad en la serie temporal, aún conociendo valores previos, la predicción del valor subsiguiente podría ser errónea, aumentando el valor de la entropía (Montero, T. y otros. 2001).

2.1.2.2. Entropía Aproximada

Sea una serie temporal *u(1),u(2),…,u(N).* Fijado $m \in Z^+$ se puede construir una sucesión de vectores de R^m, dada por: $x(i) = (u(i), u(i+1), ..., u(i+m-1))$, para $1 \leq i \leq N-m+1$.

Para $r \in R^+$ y para cada $1 \leq i \leq N+m-1$ calculamos (Grassberger, P. y Procaccia, I., 1983):

$$C_i^m(r) = (\text{número de } x(j) \text{ / } d(x(i),x(j)) \leq r) \text{ / } (N-m+1) \qquad [2.13]$$

Trabajamos en (R^m, d_∞), donde d_∞ es la *métrica del máximo*, que viene dada por:

$$d_\infty(x,y) = \max\{|x_k - y_k| : 1 \le k \le m\}.$$

para cada par de vectores $x=(x_1,x_2,\ldots,x_m)$, $y=(y_1,y_2,\ldots,y_m)$ de R^m.

A continuación, se define

$$\phi^m(r) = (N-m+1)^{-1} \sum_{i=1}^{N-m+1} \log C_i^m(r) \qquad [2.14]$$

La entropía aproximada viene dada por:

$$ApEn = \phi^m(r) - \phi^{m+1}(r) \qquad [2.15]$$

El valor de la entropía aproximada es una medida relativa que depende de tres parámetros:

(i) la longitud de la ventana usada, *N.*
(ii) el número de valores previos para la predicción del valor subsiguiente, *m*
(iii) un nivel de filtro, *r*

Hemos optado por utilizar la *entropía aproximada*, en lugar de la *entropía Kolmogorov-Sinai (K-S)*, dada por:

$$\text{K-S Entropía} = \lim_{r\to 0} \lim_{m\to\infty} \lim_{N\to\infty} \left[\phi^m(r) - \phi^{m+1}(r)\right]$$

ya que no hay indicado un procedimiento explícito para calcular el límite, no estando además garantizada su existencia. Además, si se analiza detenidamente la expresión anterior, el tiempo de cómputo de la expresión del límite crece exponencialmente con *m,* lo que hace aumentar el grado de dificultad en la aproximación del límite.

En la siguiente figura observamos la evolución de la entropía aproximada a lo largo del registro /ma/. Al igual que ocurría con la K_2-entropía, podemos apreciar como en la parte inicial de la señal el valor de la entropía es menor que en la parte final, es decir, el tramo inicial de la señal es el más regular y el conocimiento de valores previos hace posible la predicción de los valores subsiguientes. Conforme aumenta la irregularidad en la serie temporal, aún conociendo valores previos, la predicción del valor subsiguiente podría ser errónea, aumentando por tanto el valor de la entropía.

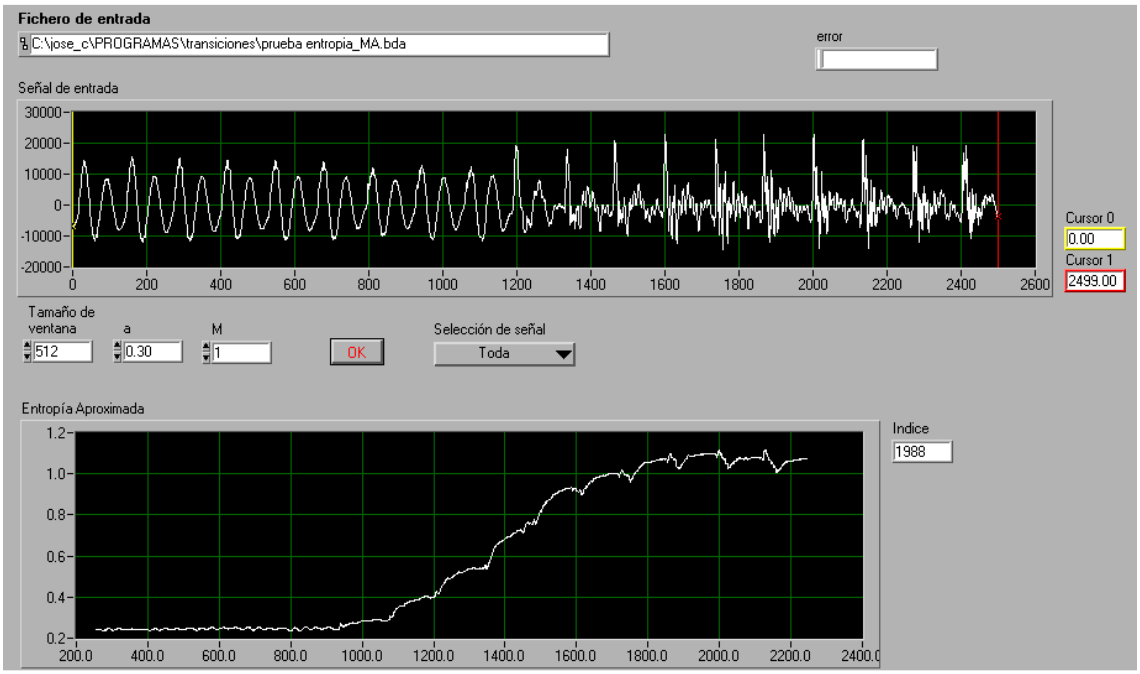


Figura 2.10: *Evolución de la entropía aproximada en el registro /ma/*

2.1.3. Dinámica Simbólica

El principio básico de dinámica simbólica es transformar una serie temporal en una secuencia de símbolos. Esto provee un modelo para las órbitas del sistema dinámico vía un espacio de sucesiones. Dado un conjunto de datos *u(1),u(2),…,u(N)* la secuencia de símbolos se obtiene cuantificando el conjunto de datos en cajas etiquetadas con un símbolo. Calculando atributos de la secuencia de símbolos puede revelar características no lineales de la serie temporal original y del sistema dinámico examinado (Guillén, P. y otros, 2000). Las secuencias de símbolos es estimarán de acuerdo al siguiente algoritmo como sigue: Una ventana de longitud *M* muestras se ajusta sobre el primer valor de la serie. Dentro de la ventana de *M* muestras se determinan las diferencias consecutivas muestra a muestra. Se cuantifican en la ventana todas las diferencias de los valores menores que la desviación estándar, *SD*. El resultado da un símbolo S_x y la ventana se desplaza al segundo valor de la serie y el procedimiento comienza de nuevo para obtener el siguiente símbolo. La fórmula matemática para este procedimiento es:

$$S_j = \sum_{i=1}^{M} \begin{cases} 0 & si \quad |u_i - u_{i+1}| \geq a \cdot SD \\ 1 & si \quad |u_i - u_{i+1}| \geq a \cdot SD \end{cases} \qquad siendo \quad j = 1,\ldots N - M + 1 \qquad [2.16]$$

donde *N* es la longitud de la serie y a es un parámetro usado para filtrar la desviación estándar de los *M* valores que constituyen la ventana.

El procedimiento descrito por la ecuación 2.16 para la obtención de los símbolos, refleja que se obtienen valores altos cuando las diferencias de las muestras consecutivas son pequeñas. Para poder efectuar comparaciones con los métodos de

cálculo de la entropía presentados en este proyecto, es necesario restar el valor *M* a los resultados de la dinámica simbólica y dividir el resultado por *M*, es decir, $S_k = (M - S_j)/M$ siendo $k=j=1,\ldots N–M+1$.

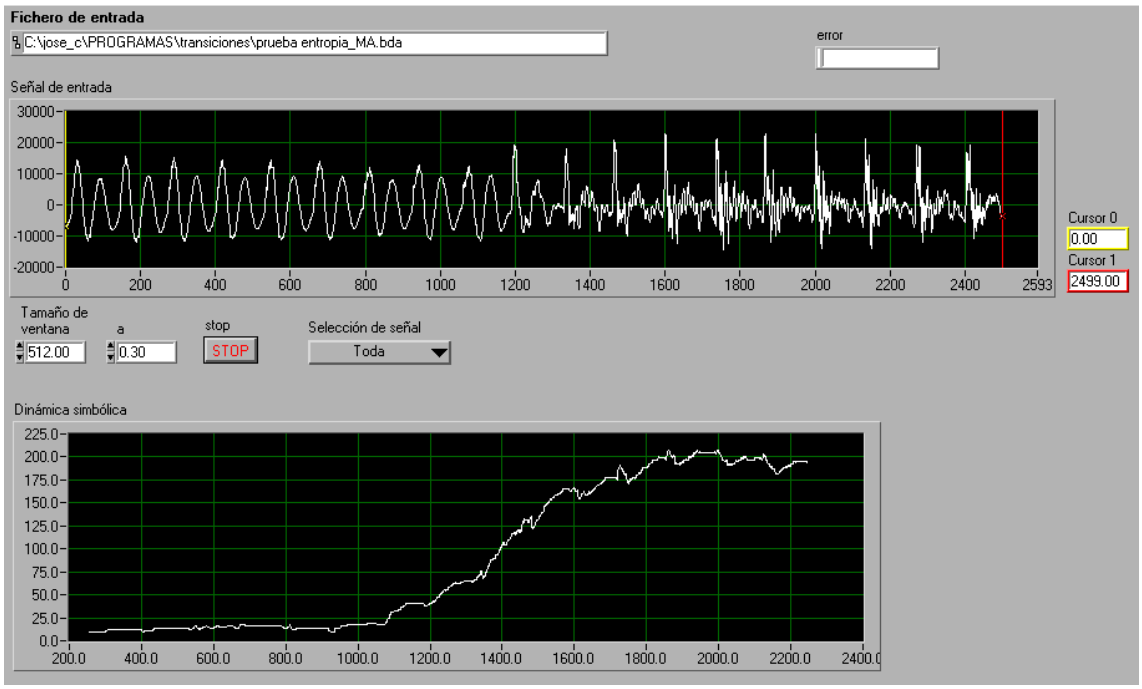


Figura 2.11: *Evolución de la dinámica simbólica en el registro /ma/*

2.2. Modelo ARMA.

Tradicionalmente, el tracto vocal ha sido modelizado para sonidos vocálicos no nasales mediante el modelo LPC, no siendo adecuada esta representación a la hora de modelizar sonidos de diferente naturaleza, por ejemplo, a la hora de estudiar transiciones entre distintos sonidos o cuando la excitación no se produce en el inicio del tracto.

Una modelización en la que se tenga en cuenta tanto entradas (ceros) como salidas anteriores del sistema (polos) parece más adecuada para abordar este tipo de situaciones.

En el modelo ARMA se considera que la salida en un instante dado, s_n, viene dada por una combinación lineal de salidas anteriores, entradas anteriores y la entrada en este mismo instante. En el tiempo, se puede expresar esta combinación lineal mediante:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G\sum_{l=0}^{q} b_l u_{n-l} \qquad [2.17]$$

La función de transferencia *H(z)* del filtro equivalente en el dominio de la transformada Z queda:

$$H(z)=\frac{S(z)}{U(z)}=G\frac{1+\sum_{l=1}^{q}b_l z^{-l}}{1+\sum_{k=1}^{p}a_k z^{-k}} \qquad [2.18]$$

Para caracterizar el modelo, se determinan los coeficientes a_k ($1\leq k\leq p$), los coeficientes b_l ($1\leq l\leq q$) y el factor de ganancia *G.*

.

La dificultad del modelo ARMA estriba en la no linealidad del sistema de ecuaciones que se obtiene de la aplicación del método de mínimos cuadrados. Sin embargo, si se calculan en diferentes fases los parámetros AR y los MA, el sistema de ecuaciones normales correspondiente a los primeros parámetros es lineal, aunque el sistema de ecuaciones que fija los segundos sea no lineal.

Para la obtención de los parámetros existen diversos métodos, algunos son directos y otros iterativos, a partir de una solución inicial. Durante el desarrollo de este proyecto se ha optado por el Método de Shanks (Shanks, J., 1967).

En la primera etapa del método, el cálculo de los *p* coeficientes AR, se realiza como si de un modelo LPC (todo polo) se tratara. Los coeficientes serán obtenidos mediante el método de autocorrelación, para así asegurar la estabilidad del filtro resultante. En la segunda etapa se sigue el método propuesto por Shanks para el cálculo de los *q* coeficientes MA. Finalmente obtendremos el modelo ARMA (*p,q*).

2.2.1. Cálculo de los coeficientes AR

En el modelo Todo-Polo se considera que la salida en un instante dado, s_n, viene dada por una combinación lineal de salidas anteriores y la entrada en este mismo instante, tal y como vemos en la siguiente figura:

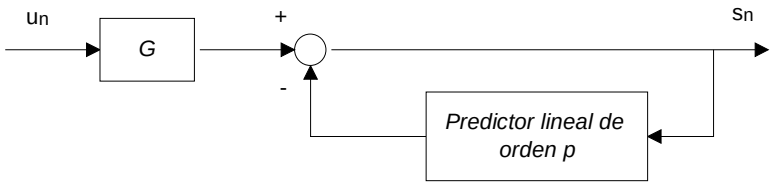


Figura 2.12: *Filtro de predicción lineal*

En el tiempo, se puede expresar esta combinación lineal mediante:

$$s_n=-\sum_{k=1}^{p}a_k s_{n-k}+G\cdot u_n \qquad [2.19]$$

La función de transferencia *H(z)* del filtro equivalente se reduce ahora a una función de transferencia Todo-Polo:

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}} \qquad [2.20]$$

Para caracterizar el modelo, se determinan los coeficientes de predicción a_k ($1 \le k \le p$) y el factor de ganancia *G*.

Para hallar los parámetros del modelo se utiliza el método de mínimos cuadrados, considerando que el proceso del cual se obtiene la señal de salida s_n es aleatorio y determinístico. El método de mínimos cuadrados se utiliza a menudo en el procesado de señales, ya que permite trabajar con los datos temporales directamente (Therrien, C.W., 1992).

2.2.1.1. Método de mínimos cuadrados para un modelo Todo-Polo

Mediante el método de mínimos cuadrados se estima un conjunto de parámetros a_k ($1 \le k \le p$) para que el comportamiento del filtro lineal que caracterizan junto al parámetro *G* se aproxime de forma óptima al sistema tracto vocal que se trata de modelar.

El desarrollo del método consiste en minimizar la suma de los errores cometidos al aproximar en cada instante la salida del sistema por la salida del filtro representado en la figura 2.12, cuya entrada u_n es desconocida, luego se aproxima la expresión [2.19] por:

$$\tilde{s}_n = -\sum_{k=1}^{p} a_k s_{n-k} \qquad [2.21]$$

Si consideramos que se calcula el error sobre una señal infinita, el error cometido en un instante *n* es:

$$e_n = s_n - \tilde{s}_n = s_n + \sum_{k=1}^{p} a_k s_{n-k} = \sum_{k=0}^{p} a_k s_{n-k} \quad \text{si se toma } a_0 = 1 \qquad [2.22]$$

El error cuadrático medio será:

$$E = \sum_n e_n^2 = \sum_n \left(s_n - \tilde{s}_n\right)^2 \qquad [2.23]$$

Para minimizar esta expresión, se toman derivadas respecto al conjunto de coeficientes a_i ($1 \le i \le p$) y se iguala a cero:

$$\frac{\delta E}{\delta a_i} = \frac{\delta\left(\sum_n e_n^2\right)}{\delta a_i} = \frac{\delta\left(\sum_n \left(s_n - \tilde{s}_n\right)^2\right)}{\delta a_i} = 0 \quad \text{con} \quad 1 \le i \le p \qquad [2.24]$$

Para cada i se obtiene una expresión:

$$\frac{\delta E}{\delta a_i} = \sum_n 2(s_n - (-a_1 s_{n-1} - \ldots - a_p s_{n-p}))s_{n-i} = 0 \quad (1 \le i \le p) \qquad [2.25]$$

$$\sum_n (s_n s_{n-i} - (-a_1 s_{n-1} s_{n-i}) - \ldots - (-a_p s_{n-p} s_{n-i})) = 0 \quad (1 \le i \le p) \qquad [2.26]$$

Si se expresa esta ecuación para cada k, se obtiene un sistema de ecuaciones:

$$\left.\begin{array}{l} \sum_n s_n s_{n-1} + a_1 \sum_n s_{n-1} s_{n-1} + \ldots + a_p \sum_n s_{n-p} s_{n-1} = 0 \\ \sum_n s_n s_{n-2} + a_1 \sum_n s_{n-1} s_{n-2} + \ldots + a_p \sum_n s_{n-p} s_{n-2} = 0 \\ \vdots \\ \sum_n s_n s_{n-p} + a_1 \sum_n s_{n-1} s_{n-p} + \ldots + a_p \sum_n s_{n-p} s_{n-p} = 0 \end{array}\right\} \qquad [2.27]$$

Este sistema se puede expresar matricialmente:

$$\begin{pmatrix} \sum_n s_n s_{n-1} & \sum_n s_{n-1} s_{n-1} & \cdots & \sum_n s_{n-p} s_{n-1} \\ \sum_n s_n s_{n-2} & \ddots & & \sum_n s_{n-p} s_{n-2} \\ \vdots & & & \\ \sum_n s_n s_{n-p} & \sum_n s_{n-1} s_{n-p} & & \sum_n s_{n-p} s_{n-p} \end{pmatrix} \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \\ 0 \end{pmatrix} \qquad [2.28]$$

Estas son las ecuaciones normales para los coeficientes del predictor lineal y se les conoce como ecuaciones de Yule-Walker. Se observa que los sumatorios no se encuentran limitados a un intervalo finito; posteriormente se limitarán al aplicar los métodos sobre señales finitas.

También se puede expresar el mínimo error total, sustituyendo las ecuaciones de [2.28] en la expresión del error [2.23]:

$$E_p = \sum_n s_n^2 + \sum_{k=1}^{p} a_k \sum_n s_n s_{n-k} \qquad [2.29]$$

La resolución del sistema de ecuaciones [2.28] puede realizarse mediante diversos métodos, entre ellos hemos seleccionado el método de autocorrelación, que nos asegura la estabilidad del filtro resultante

2.2.1.2. Método de autocorrelación

Resulta interesante hacer algunos cambios en las ecuaciones anteriores. Se puede simplificar este último sistema de ecuaciones sustituyendo la definición de la función de autocorrelación de la señal s_n:

$$R_i = \sum_{n=-\infty}^{\infty} s_n s_{n+i} \qquad [2.30]$$

Como únicamente se conocen los valores de la señal s_n en un intervalo finito en el que $0 \le n \le N_s-1$, se puede suponer que se trabaja con la función de autocorrelación aplicada sobre una señal s'_n, que no es más que el resultado de aplicar una ventana rectangular con una longitud N_s sobre la señal s_n (Makhoul, J., 1976), y así se supone en el método de autocorrelación.

$$R_i = \sum_{n=0}^{Ns-1-i} s'_n s'_{n+i} \qquad [2.31]$$

Si se sustituye esta definición de la función de autocorrelación en la ecuación [2.26], queda:

$$R_k + a_1 R_{k-1} + ... + a_p R_{k-p} = 0 \text{ para cada } 1 \le k \le p \qquad [2.32]$$

Si se expresa esta ecuación para cada k, se obtiene un sistema de ecuaciones que matricialmente se puede expresar:

$$\begin{pmatrix} R_1 & R_0 & \cdots & R_{1-p} \\ R_2 & \ddots & & R_{2-p} \\ \vdots & & & \vdots \\ R_p & R_{p-1} & \cdots & R_0 \end{pmatrix} \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \\ 0 \end{pmatrix} \qquad [2.33]$$

Se observa que la matriz de coeficientes del sistema es la matriz de autocorrelación de orden *p* de la señal s_n; además, es una matriz simétrica y cuyos elementos a lo largo de cualquier diagonal son idénticos, es una matriz de Toeplitz (Makhoul, J., 1976). Como la matriz del sistema de ecuaciones normales es de Toeplitz, los polos del filtro (las raíces del polinomio del denominador) están situados dentro del círculo unidad en el plano Z, luego el filtro resultante del conjunto de parámetros a_k será de fase mínima y esta característica nos asegura su estabilidad.

La expresión del error total queda:

$$E_p = R_0 + \sum_{k=1}^{p} a_k R_k \qquad [2.34]$$

2.2.1.3. Cálculo del factor de ganancia

En el modelo LPC se considera que la salida en un instante determinado viene dada por la expresión [2.19]. Al obtener los parámetros del modelo LPC mediante el método de mínimos cuadrados, en el cual se supone desconocida la entrada u_n, la salida vendrá dada por la expresión [2.21]. La expresión del error cometido en la aproximación se puede reescribir:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + e_n \qquad [2.39]$$

Si se comparan las expresiones [2.19] y [2.21], se observa que es posible caracterizar la señal error en mediante el término correspondiente a la entrada e_n el instante *n* ponderada por el factor de ganancia *G*, es decir $e_n=Gu_n$ (Makhoul, J., 1976).

Además, la energía de la señal de salida del filtro *H(z)* debe ser igual a la energía de la señal s_n cuando la entrada u_n esté determinada. Si suponemos como entrada una señal impulso, $u_n=\delta_{n0}$, la salida del sistema será la respuesta impulsorial:

$$h_n = -\sum_{k=1}^{p} a_k h_{n-k} + G \cdot \delta_{n0} \qquad [2.40]$$

De la expresión anterior, multiplicándola por h_{n-i} y sumando sobre todo *n*, se obtiene la expresión de la función de autocorrelación para h_n:

$$\hat{R}_i = \begin{cases} -\sum_{k=1}^{p} a_k \hat{R}_k + G^2 & \text{para } i = 0 \\ -\sum_{k=1}^{p} a_k \hat{R}_{i-k} & \text{para } 1 \le |i| < \infty \end{cases} \quad [2.41]$$

Para cualquier señal x_n, el coeficiente de autocorrelación cero es igual al total de la energía de la señal ($R_0 = \sum_n x_n^2$). Por tanto, para obtener la igualdad de la energía de las señales s_n y h_n, han de ser iguales los coeficientes:

$$\hat{R}_0 = R_0 \quad [2.42]$$

Llevando esta expresión a las ecuaciones normales del método de mínimos cuadrados, [2.27], se obtiene:

$$\hat{R}_i = R_i \quad \text{para } 0 \le i \le p \quad [2.43]$$

Y sustituyendo estas expresiones en la del error cuadrático total [2.29] se obtiene:

$$E_p = \sum_n \left(s_n + \sum_{k=1}^{p} a_k s_{n-k} \right)^2 = R_0 + k \sum_{k=1}^{p} a_k R_k = \hat{R}_0 + \sum_{k=1}^{p} a_k \hat{R}_k \quad [2.44]$$

Utilizando la expresión de la respuesta impulso, podemos relacionar el error total con el factor de ganancia:

$$E_p = \sum_n \left(h_n + \sum_{k=1}^{p} a_k h_{n-k} \right)^2 = \sum_n \left(G \cdot \delta_{n0} \right)^2 = G^2 \quad [2.45]$$

2.2.2. Cálculo de los coeficientes MA

Un modelo ARMA tiene una función del tipo $H(z) = \frac{B(z)}{A(z)}$. A partir de los coeficientes AR calculados en el paso previo, podemos definir el sistema parcial $H_A(z) = \frac{1}{A(z)}$ y escribir la siguiente aproximación:

$$X(z) \approx B(z) \cdot H_A(z).$$

Si $h_A(n)$ es la secuencia correspondiente a la respuesta del sistema $H_A(z)$ ante una entrada impulso, el error cometido en la aproximación anterior podemos representarlo como:

$$e_B(n) = x(n) - h_A(n) \cdot b(n) \qquad [2.46]$$

Si los coeficientes del filtro $B(z)$ son elegidos de tal forma que minimicen la suma de los errores al cuadrado, el problema puede ser escrito como (Shanks, J., 1967):

$$H_A \cdot b \overset{ls}{=} x \qquad [2.47]$$

Donde cada término corresponde a las siguientes expresiones:

$$H_A = \begin{bmatrix} h_A(0) & 0 & \cdots & 0 \\ h_A(1) & h_A(0) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ h_A(Q) & h_A(Q-1) & \cdots & h_A(0) \\ \vdots & \vdots & \cdots & \vdots \\ h_A(N_s-1) & h_A(N_s-2) & \cdots & h_A(N_s-Q-1) \end{bmatrix}, \qquad [2.48]$$

$$x = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(Q) \\ \vdots \\ x(N_s-1) \end{bmatrix} \qquad [2.49]$$

y

$$b = \begin{bmatrix} b(0) \\ b(1) \\ \vdots \\ b(q) \end{bmatrix} \qquad [2.50]$$

Los términos necesarios de la secuencia $h_A(n)$ pueden ser obtenidos de la siguiente ecuación:

$$h_A(n) = -\sum_{k=1}^{p} a_k \cdot h(n-k) + \delta(n) \qquad [2.51]$$

2.2.3. Reflexión de ceros

El polinomio que obtenemos al resolver el sistema de ecuaciones planteado para calcular los coeficientes MA puede presentar raíces fuera del círculo unidad. Como más tarde veremos será necesario asegurar que estas estén dentro del círculo unidad. Por tanto, tal y como vemos en la figura 2.13, será necesario reflejarlos dentro del mismo.

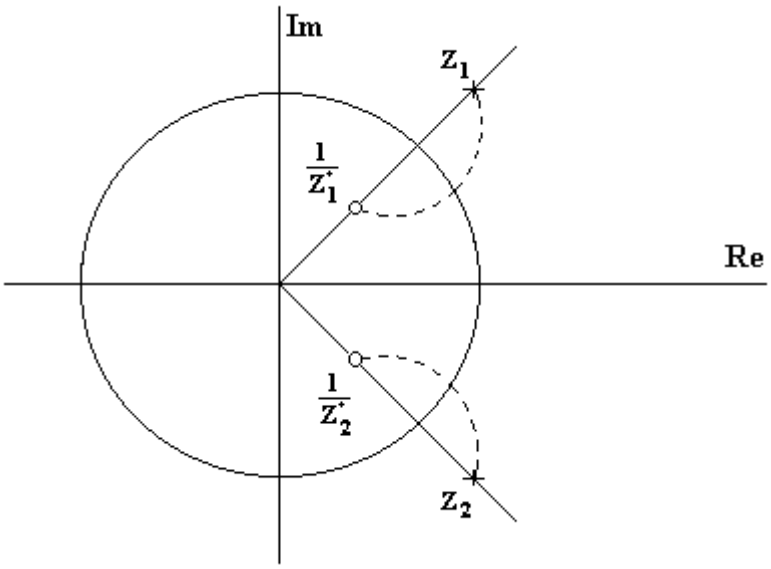


Figura 2.13: *Reflexión de ceros*

Así para un polinomio B(z) de orden 2, con ceros z_1 y z_2:

$$B(z) = 1 + b_1 z^{-1} + b_2 z^{-2} = (1 - z_1 z^{-1}) \cdot (1 - z_2 z^{-1}) \qquad [2.52]$$

Estos ceros podrán ser reales o bien complejos conjugadas, lo que da lugar a dos casos.

Caso1: Sea $B(z)=\left(1-z_1z^{-1}\right)\cdot\left(1-z_2z^{-1}\right)$ con $z_1,z_2\in R$, luego $z_1=z_1^*$, $z_2=z_1^*$. Supongamos que $|z_1|>1$. Para realizar la reflexión del cero z_1 multiplicamos *B(z)* por el término $\frac{z^{-1}-z_1^*}{1-z_1z^{-1}}$ (Therrien, 1992):

$$B(z)=\left(1-z_1z^{-1}\right)\cdot\left(1-z_2z^{-1}\right)\cdot\left(\frac{z^{-1}-z_1^*}{1-z_1z^{-1}}\right)$$

$$=\left(1-z_2z^{-1}\right)\cdot\left(z^{-1}-z_1\right)=-z_1\cdot\left(1-z_2z^{-1}\right)\cdot\left(1-\frac{z^{-1}}{z_1}\right)$$

$$B(z)=-z_1\cdot\left[1-\left(\frac{1}{z_1}+z_2\right)z^{-1}+\left(\frac{z_2}{z_1}\right)z^{-2}\right] \qquad [2.53]$$

Siendo por tanto los nuevos coeficientes:

$$b_1^{'}=-\left(\frac{1}{z_1}+z_2\right)$$
$$b_2^{'}=\frac{z_2}{z_1} \qquad [2.54]$$

El factor $-z_1$ pasaría a formar parte de la ganancia.

De igual modo operaríamos con el cero z_2 si estuviera fuera del círculo unidad, es decir, $|z_2|>1$.

Caso 2: También puede darse el caso de que los dos ceros estén fuera del círculo unidad. En este caso $z_1,z_2\in C$ y además son conjugados, luego $z_2=z_1^*$ y $\left(z_1^*\right)^*=z_1$.

$$B(z)=\left(1-z_1z^{-1}\right)\cdot\left(1-z_2z^{-1}\right)\cdot\left(\frac{z^{-1}-z_2}{1-z_1z^{-1}}\right)\cdot\left(\frac{z^{-1}-z_1}{1-z_2z^{-1}}\right)=$$

$$=\left(z^{-1}-z_2\right)\cdot\left(z^{-1}-z_1\right)=z_1z_2\cdot\left(1-\frac{z^{-1}}{z_1}\right)\cdot\left(1-\frac{z^{-1}}{z_2}\right)$$

$$B(z) = z_1 z_2 \cdot \left[1 - \left(\frac{1}{z_1} + \frac{1}{z_2} \right) z^{-1} + \left(\frac{1}{z_1 z_2} \right) z^{-2} \right] \qquad [2.55]$$

Siendo por tanto los nuevos coeficientes:

$$b_1^{'} = -\left(\frac{1}{z_1} + \frac{1}{z_2} \right)$$
$$b_2^{'} = \frac{1}{z_1 z_2} \qquad [2.56]$$

El factor $z_1 z_2$ pasaría a formar parte de la ganancia.

La introducción de este nuevo término no afecta a la respuesta en frecuencia del polinomio en lo que a la amplitud se refiere, ya que:

$$\frac{z^{-1} - z_0^*}{1 - z_0 z^{-1}} = \frac{e^{-jw} - z_0^*}{1 - z_0 e^{-jw}} = e^{-jw} \cdot \frac{1 - z_0^* e^{jw}}{1 - z_0 e^{jw}}$$

$$\left| \frac{z^{-1} - z_0^*}{1 - z_0 z^{-1}} \right| = e^{-jw} \cdot \frac{1 - z_0 e^{-jw}}{1 - z_0^* e^{jw}} \cdot \frac{1 - z_0^* e^{jw}}{1 - z_0 e^{-jw}} = e^{jw} = 1$$

Por el contrario al introducir este término la respuesta en fase si se modifica.

2.2.4. Envolvente ARMA

El análisis de la señal puede realizarse en el dominio de la frecuencia, generalmente mediante el espectro de potencia, notado $P(\omega)$. El espectro de potencia de una señal del habla proporciona información muy útil para el estudio de la misma, como el contenido espectral de los formantes, en caso de que se trate de un sonido sonoro.

La aproximación a una señal posee una estimación del espectro que debe ser lo más parecida posible al espectro de potencia original. A la aproximación al espectro se le denomina envolvente del espectro y se nota $\hat{P}(\omega)$. Se puede aproximar el espectro de potencia de la señal con la envolvente del modelo ARMA.

La función de transferencia de un modelo ARMA viene dada por la expresión [2.18]. El espectro de potencia correspondiente al modelo se puede expresar como (Makhoul, 1976):

$$\hat{P}(\omega) = \left|H(e^{j\omega})\right|^2 = G^2 \frac{\left|B(e^{j\omega})\right|^2}{\left|A(e^{j\omega})\right|^2} = G^2 \frac{N(\omega)}{D(\omega)} \qquad [2.57]$$

Donde *B(z)* y *A(z)* son los polinomios del numerador y el denominador de *H(z)*, *N(ω)* y *D(ω)* son las formas Todo-Cero y Todo-Polo del numerador y denominador de $\hat{P}(\omega)$, y vienen dadas por:

$$N(\omega) = \left|1 + \sum_{l=1}^{q} b_l \cdot e^{-jl\omega}\right|^2$$
$$D(\omega) = \left|1 + \sum_{k=1}^{p} a_k \cdot e^{-jk\omega}\right|^2 \qquad [2.58]$$

El error cometido al aproximar el espectro de potencia de la señal *P(ω)* por la envolvente $\hat{P}(\omega)$ vendrá dado por la siguiente expresión:

$$E = \frac{G^2}{2\pi} \int_{-\pi}^{\pi} \frac{P(\omega)}{\hat{P}(\omega)} d\omega \qquad [2.59]$$

sustituyendo la expresión de la envolvente ARMA [2.59] nos queda:

$$E = \frac{1}{2\pi} \int_{-\pi}^{\pi} P(\omega) \frac{D(\omega)}{N(\omega)} d\omega \qquad [2.60]$$

En las expresiones desarrolladas en los apartados anteriores se supone que el espectro que se obtiene de las señales es continuo. En nuestro caso trabajaremos con señales del habla muestreadas, luego el espectro será discreto. Tendremos que modificar las expresiones anteriores.

Los únicos cambios a realizar en las expresiones ya presentadas es el cambio de las integrales que recorren el intervalo [-π, π] por sumatorios sobre el intervalo de puntos del espectro [0, N_s-1], donde N_s es el número de muestras de la señal.

Sea el polinomio *Q(ω)* con *r* coeficientes

$$Q(\omega) = 1 + \sum_{k=1}^{r} q_r \cdot e^{-jk\omega} \ ,$$

para una señal discreta de N_s muestras, la expresión de la respuesta en frecuencia será:

$$|Q(\omega)|^2 = \left(\sum_{i=0}^{Ns-1} \sum_{k=0}^{r} q_k (\cos(k \frac{i\pi}{N_S})) \right)^2 + \left(\sum_{i=0}^{Ns-1} \sum_{k=0}^{r} q_k (\text{sen}(k \frac{i\pi}{N_s})) \right)^2 \quad [2.61]$$

A continuación tenemos dos ejemplos de envolventes para dos transiciones diferentes. Las envolventes son obtenidas a partir de sendos modelos ARMA calculados a partir del método de Shanks. La primera de ellas será una transición entre un sonido sordo (fricativa) y un sonido sonoro (vocal) y la segunda corresponderá a una transición entre dos sonidos sonoros (consonante nasal y vocal)

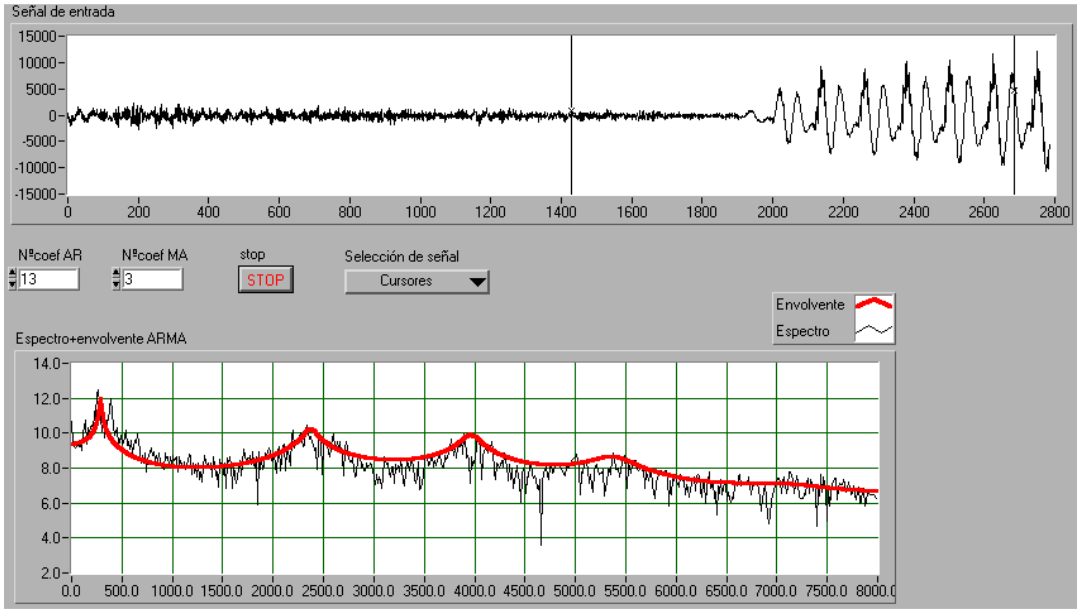


Figura 2.14: *Espectro de potencia y envolvente ARMA para el registro /si/*

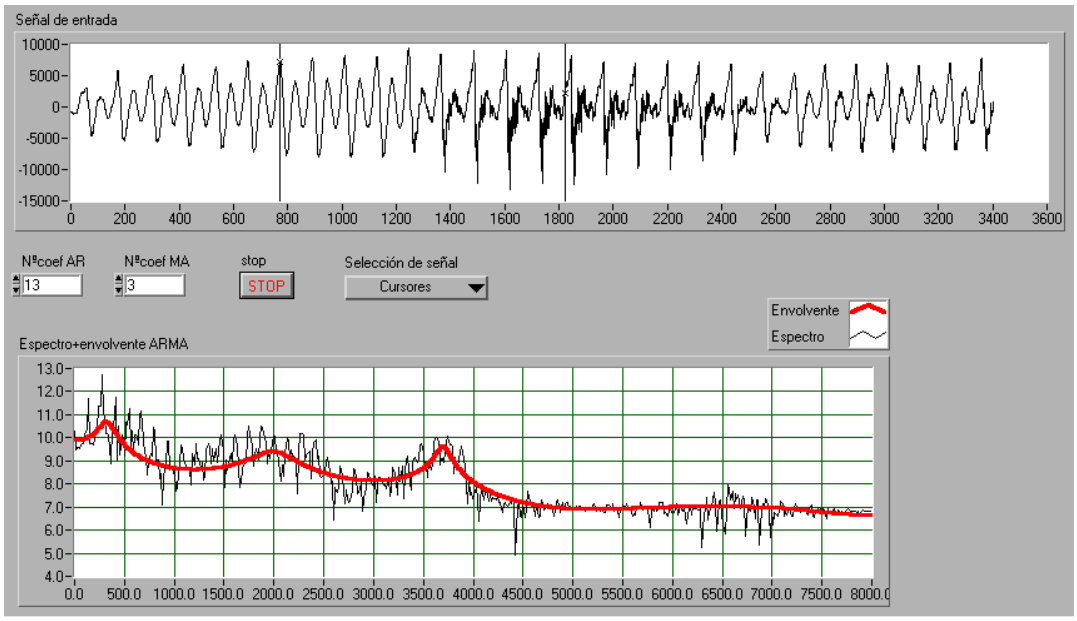


Figura 2.15: *Espectro de potencia y envolvente ARMA para el registro /ma/*

2.2.5. Número óptimo de polos y ceros

La selección del orden de un modelo ARMA presenta siempre mayor dificultad que para un modelo AR, ya que se ha de elegir un orden para el numerador y otro para el denominador. En numerosas aplicaciones como, por ejemplo, en el campo de las señales del habla, el orden del denominador es más crítico que el del numerador, al determinar el primero el número de modos o frecuencias naturales presentes en la señal. En este trabajo, para la selección del orden de un modelo ARMA, se propone una adaptación del AIC (Akaike´s information-theoric criteria) desarrollado para modelos AR. En el capítulo correspondiente a resultados recogemos la discusión de la selección del número óptimo de polos y ceros para la modelización ARMA de sonidos del habla natural (González, F., 2001).

2.3. Cepstrum

Sea la señal temporal discreta s_n, cuya transformada *Z* notamos *S(z)*. Suponemos que s_n es una secuencia estable, de modo que *S(z)* converge en el círculo unidad según (Proakis, J.G.; Manolakis, D.G., 1998). El cepstrum complejo de la secuencia s_n se define como la secuencia c_n, que es la inversa de la transformada Z de *C(z)*, donde:

$$C(z) = \ln S(z) \qquad [2.62]$$

La función de transferencia de un sistema en tiempo discreto se define como el cociente de las transformadas Z de la señal salida y la señal entrada:

$$H(z) = \frac{S(z)}{U(z)} \Rightarrow S(z) = U(z) \cdot H(z) \qquad [2.63]$$

Aplicando [2.62]:

$$C(z) = \ln S(z) = \ln U(z) + \ln H(z) \qquad [2.64]$$

Se observa que el primer sumando del segundo miembro de la expresión [2.64] se refiere a la fuente (entrada al sistema), mientras que el segundo sumando se refiere a la transmisión a lo largo del sistema (función de transferencia del mismo).

El cepstrum complejo existe si *C(z)* converge en la región anular $r_1<z<r_2$, con $0<r_1<1$ y $r_2>1$. Dentro de esta región de convergencia, *C(z)* puede representarse mediante la serie de Laurent.

$$C(z) = \ln S(z) = \sum_{n=-\infty}^{\infty} c_n z^{-n} \qquad [2.65]$$

donde

$$c_n = \frac{1}{2\pi j} \int_C \ln S(z) z^{n-1} dz \qquad [2.66]$$

C es una curva cerrada simple que incluye al origen y que pertenece a la región de convergencia. Claramente, si *C(z)* se puede representar como en [2.65], la secuencia cepstrum complejo c_n es estable. Además, si el cepstrum complejo existe, *C(z)* converge en el círculo unidad, y así, tenemos:

$$C(\omega) = \ln S(\omega) = \sum_{n=-\infty}^{\infty} c_n e^{-j\omega n} \qquad [2.67]$$

donde c_n es la secuencia que se obtiene a partir de la transformada de Fourier inversa de ln $S(\omega)$, esto es:

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln S(\omega) \cdot e^{j\omega n} d\omega \qquad [2.68]$$

Se observa que la definición [2.68] corresponde al cepstrum complejo. En procesado de señales de voz, la parte imaginaria del cepstrum no interesa, ya que las aplicaciones para las cuales se utiliza el cepstrum no requieren la posterior reconstrucción de la señal en el dominio del tiempo. De modo que se utiliza generalmente la parte real del cepstrum.

Si expresamos $S(\omega)$ en términos de módulo y fase:

$$S(\omega) = |S(\omega)| e^{j\theta(\omega)} \qquad [2.69]$$

entonces

$$\ln S(\omega) = \ln |S(\omega)| + j\theta(\omega) \qquad [2.70]$$

Sustituyendo [2.70] en [2.68] obtenemos el cesptrum complejo en la forma

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[\ln |S(\omega)| + j\theta(\omega) \right] \cdot e^{j\omega n} d\omega \qquad [2.71]$$

Podemos separar la transformada de Fourier inversa en las transformadas de Fourier inversas correspondientes a las partes real e imaginaria:

$$c_{n(módulo)} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln\left|S(\omega)\right| \cdot e^{j\omega n} d\omega$$
$$c_{n(\,fase)} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \theta(\omega) \cdot e^{j\omega n} d\omega \qquad [2.72]$$

Capítulo 3: Reconocimiento de locutores

Etapa de procesado de la señal
Comparación de modelos
Etapa de clasificación

3. Reconocimiento de locutores

En este apartado desarrollaremos el esquema de un sistema de reconocimiento automático dado en la figura 1.12. Recordemos que el mismo se divide en tres etapas: Procesado de la señal, comparación de modelos y etapa de decisión lógica. Veamos a continuación el contenido de cada una de estas etapas según el sistema que hemos implementado a lo largo de este proyecto.

3.1. Etapa de procesado de la señal

Esta etapa a su vez queda dividida en diversas fases según vemos en la figura:

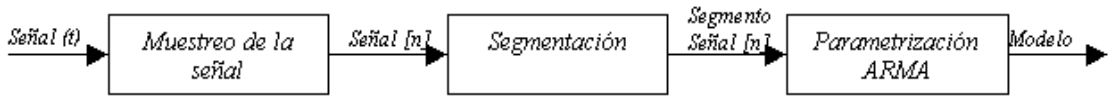


Figura 3.1: *Etapa de procesado de señal*

3.1.1. Muestreo de la señal

El muestreo de la señal se hará de acuerdo a lo visto en el apartado 1.2. La frecuencia de muestreo será la adecuada para que no se produzca una pérdida de información. En nuestro caso, las señales serán muestreadas a una frecuencia de 16000 Hz.

3.1.2. Segmentación

La segmentación nos permite aislar eventos de la señal del habla que sean fuertemente dependientes del locutor y así usar la información que contienen para el reconocimiento (Rosenberg,1976). En nuestro caso caracterizamos la señal de entrada a través de los métodos de la dinámica no lineal que hemos visto en el apartado 2.1., y tomamos los segmentos transicionales, ya que como veremos es el tramo de señal más dependiente del locutor. En concreto tomaremos transiciones con coarticulación nasal. Para diversos autores, la coarticulación se debe a que en el habla conectado las articulaciones del sistema fonador se mueven lentamente y la forma del tracto vocal en un momento dado no sólo depende del fonema que estemos emitiendo, si no también de los vecinos (Atal, B. S., 1976). La coarticulación que se produce en las transiciones consonante nasal – vocal tiene una fuerte dependencia del locutor y puede ser usada en sistemas de reconocimiento de locutores (Li, 1969). Estas transiciones están controladas por impulsos neuromusculares de tal forma que se producen inconscientemente, es decir, no se puede controlar por el locutor.

3.1.3. Parametrización

En esta etapa obtenemos un modelo de la señal de entrada. Usaremos modelización ARMA, ya que como vimos en el apartado 2.2, era la que mejor se adaptaba a las transiciones. Para poder realizar el modelo, es necesario asumir la condición de estacionalidad del segmento que modelamos. Esta la podemos asegurar modelando tramos de señal pequeños (32 ms), en caso de que la señal de entrada la dividimos en pequeños segmentos de 32 ms y modelamos cada uno de ellos. Los modelos ARMA se obtendrán de acuerdo al método de Shanks según vimos en el apartado 2.2.

3.2. Comparación de modelos

3.2.1. Obtención de los modelos de referencia de los locutores

Para clasificar la señal de entrada como perteneciente a alguno de los locutores de la base de datos, es necesario tener un modelo que sea representativo de cada locutor para el tipo de señal de entrada (sistemas dependientes del texto). Para crear un modelo de referencia, que denotaremos r_{ij} , tomamos N muestras del locutor j y creamos el modelo de cada una de ellas. Realizando las media entre los distintos modelos obtenemos un el modelo de referencia m_j, que matemáticamente lo expresamos como:

$$m_j = \frac{1}{N}\sum_{i=1}^{N} r_{ij} \qquad [3.1]$$

3.2.2. Cálculo de la distancia entre modelos

A la hora de realizar la comparación entre dos modelos, necesitaremos encontrar una métrica que nos indique el grado de similitud o distancia entre ellos. A lo largo de la historia se han propuesto muchas técnicas para calcular distancias entre modelos de tal forma que permitan su aplicación en la clasificación de modelos. La más conocida y aplicada es el Análisis Discriminante (AD). Junto a esta técnica clásica, en este proyecto presentamos una nueva métrica que establece una distancia entre modelos ARMA a partir de los coeficientes cepstrales correspondientes.

3.2.2.1. Análisis Discriminante

El análisis discriminante trata de decidir si un ''individuo'' sobre el que se han estudiado una serie de características pertenece a una de las poblaciones (normales) existentes ''a priori''. Para ello construiremos una función (discriminante) que servirá

para decidir en qué población incluimos cada sujeto. Las variables estudiadas sobre los individuos deben ser numéricas (normales multivariantes). Con el fin de construir un modelo para cada una de las poblaciones en las que se pueden clasificar los individuos completamente, extraeremos una muestra de individuos de cada una de ellas.

Básicamente, veremos que esta clasificación se basa en la distancia de Mahalanobis del individuo a cada una de las poblaciones. La utilización de esta distancia es equivalente bajo normalidad a la utilización del criterio de máxima verosimlitud que clasificará a un individuo donde sean ''más probables'' sus características. Este segundo criterio permitirá la extensión de dicha clasificación a más de dos poblaciones con diferentes matrices de covarianzas, incluso relajando la hipótesis de la normalidad de las mismas.

Recordemos que si tenemos una población caracterizada por una v.a. *X* de dimensión *k*, con vector de medias μ y matriz de covarianzas $V = (\sigma_{ij})$ definida positiva, se define la distancia de Mahalanobis de una observación $x \in R^k$ a la población *X* como:

$$\Delta(x,\mu) = \sqrt{(x-\mu)'V^{-1}(x-\mu)} \qquad [3.2]$$

Supongamos que $X = (x_1,\ldots,x_k)$ e $Y = (y_1,\ldots,y_k)$ son dos v.a. normales k dimensionales con vectores de medias μ_x y μ_y y matriz de covarianzas común V definida positiva. Supongamos que $z = (z_1,\ldots,z_k)$ representa las medidas obtenidas para un individuo y que *z* proviene de *X* o de *Y*. Es decir, *z* provendrá de una v.a. normal k dimensional con media igual a μ_x o μ_y y matriz de covarianzas *V*. En la práctica *z* será un punto de R^k que debemos clasificar en *X* o en *Y*.

La idea desarrollada por Fisher (1936) es usar una función discriminante *D* unidimensional lineal basada en *z*. De esta forma clasificaremos la observación como perteneciente al locutor que se encuentre más cerca. Esto es, z será clasificada en X si :

$$\Delta(z,\mu_X) < \Delta(z\mu_Y)$$

siendo μ_x y μ_y los vectores de medias de las poblaciones *X* e *Y* respectivamente. Este resultado es equivalente a encontrar aquel vector *a* que:

$$\max_a \frac{(a'\mu_X - a'\mu_Y)^2}{a'Va} \qquad [3.3]$$

Nótese que el objetivo es alejar las medias $a'\mu_X - a'\mu_Y$ y disminuir la varianza común $\sigma^2 = a'Va$.

La solución general al problema anterior es:

$$a = V^{-1}(\mu_X - \mu_y)$$

y el máximo valor de la expresión [3.3] es precisamente el cuadrado de la distancia de Mahalanobis entre ambas poblaciones $(\Delta^2(\mu_X, \mu_Y))$.

Otro aspecto importante es dar una medida de la clasificación realizada, puesto que los errores vistos anteriormente competen al método globalmente, y no a cada observación de manera individual. Así, dada una observación *z* podemos calcular la probabilidad de encontrarnos en una u otra población utilizando el teorema de Bayes, teniendo en cuenta que la función de verosimilitud, L_x o L_Y , asociada a cada población se corresponderá con la densidad asociada a una distribución normal k-dimensional con parámetros μ_x o μ_y (según la población) y la matriz de covarianzas *V*, evaluada en el punto z, que suponemos común a ambas poblaciones. Así:

$$\begin{aligned} L_X &= L(z, \mu_X) \\ L_Y &= L(z, \mu_Y) \end{aligned} \qquad [3.4]$$

siendo

$$L(z, \mu) = \frac{1}{\sqrt{|V|(2\pi)^k}} \exp\left\{-\frac{1}{2}(z-\mu)'V^{-1}(z-\mu)\right\} \qquad [3.5]$$

Por tanto, aplicando el teorema de Bayes y el teorema de la Probabilidad total, obtenemos la probabilidad de que el vector observado *z* pertenezca a la población *X* o *Y*.

$$\Pr(X/z) = \frac{\Pr(z/X)\cdot\Pr(X)}{\Pr(z/X)\cdot\Pr(X)+\Pr(z/Y)\cdot\Pr(Y)} = \frac{L(z,\mu_X)\cdot\Pr(X)}{L(z,\mu_X)\cdot\Pr(X)+L(z,\mu_Y)\cdot\Pr(Y)}$$

$$\Pr(Y/z) = \frac{\Pr(z/Y)\cdot\Pr(Y)}{\Pr(z/X)\cdot\Pr(X)+\Pr(z/Y)\cdot\Pr(Y)} = \frac{L(z,\mu_Y)\cdot\Pr(Y)}{L(z,\mu_Y)\cdot\Pr(Y)+L(z,\mu_X)\cdot\Pr(X)}$$

Suponiendo que las probabilidades a priori de pertenencia a cada población son iguales, las expresiones anteriores se transforman en:

$$\Pr(X/z) = \frac{\Pr(z/X)}{\Pr(z/X)\cdot+\Pr(z/Y)} = \frac{L(z,\mu_X)}{L(z,\mu_X)+L(z,\mu_Y)}$$

$$\Pr(Y/z)=\frac{\Pr(z/Y)}{\Pr(z/X)+\Pr(z/Y)}=\frac{L(z,\mu_Y)}{L(z,\mu_Y)+L(z,\mu_X)}$$

En el caso de tener más de dos locutores, la expresión general para obtener las probabilidades de pertenencia a un determinado locutor será:

$$\Pr(locutor\ \ i/z)=\frac{L(z,\mu_i)}{\sum_{j=1}^{n^o locutores} L(z,\mu_{ji})}=\frac{e^{-\frac{1}{2}D_i^2}}{\sum_{j=1}^{n^o locutores} e^{-\frac{1}{2}D_j^2}} \qquad [3.6]$$

donde D_i será la distancia de Mahalanobis de la observación z al locutor *i.*

Sin embargo, esta técnica presenta un inconveniente al aplicarla a modelos ARMA. En los modelos ARMA, existe una alta correlación entre los coeficientes. Debido a esto, es posible que tengamos problemas a la hora de invertir la matriz de covarianzas, ya que en la práctica el valor de su determinante puede ser próximo a 0. Por tanto para aplicar el AD es necesario dar un paso previo en el que se elimine la información redundante y acabar con la correlación entre los coeficientes. Este proceso se realizará por medio de una técnica clásica conocida como Análisis de Componentes Principales (ACP).

3.2.2.2. Análisis de Componentes Principales

El análisis de componentes principales (ACP) suele ser casi siempre el primer paso a la hora de estudiar un conjunto de datos en los que intervienen numerosas variables. Supongamos que $\vec{X}=(X_1,X_2,\ldots,X_k)$ es una v.a. k dimensional con matriz de covarianzas *V* semidefinida positiva. Entonces la primera componente principal será la v.a. unidimensional $Y_1=t_{11}X_1+\ldots+t_{k1}X_k$ cuya varianza es máxima. Nuestro problema reside en encontrar el vector $\vec{t}_1=(t_{11},\ldots,t_{k1})'$, por comodidad a partir de este momento denotaremos por $t_1=t'_1$ y por $X=\vec{X}$. Por consiguiente el problema que debemos resolver será:

$$\left.\begin{array}{l} max\ \ Var(t'_1X) \\ s.a.\ \ t'_1t_1=1 \end{array}\right\} \qquad [3.7]$$

ahora bien, como:

$$Var(Y_1)=Var(t'_1X)=t'_1Var(X)t_1=t'_1Vt_1 \qquad [3.8]$$

Con el fin de maximizar esta función, utilizando multiplicadores de Lagrange, definimos la siguiente función auxiliar:

$$L(t_1, \lambda) = t'_1 V t_1 - \lambda(t'_1 t_1 - 1) \qquad [3.9]$$

Para poder maximizarla con respecto al vector t_1 debemos calcular las parciales con respecto de cada componente, para lo cual nos apoyaremos en el siguiente resultado:

$$\frac{\delta L}{\delta t_1} = 2Vt_1 - 2\lambda I t_1 \qquad [3.10]$$

siendo *I* la matriz identidad.

Por tanto, aquellos valores t_1 que maximizan la varianza de la variable Y_1 serán los que verifiquen:

$$\frac{\delta L}{\delta t_1} = 2Vt_1 - 2\lambda I t_1 = 0 \qquad [3.11]$$

Por tanto:

$$Vt_1 - \lambda I t_1 = 0 \Rightarrow Vt_1 = \lambda I t_1 \qquad [3.12]$$

lo que indica que t_1 es un vector propio asociado a la matriz V. Además, como:

$$Var(Y_1) = t'_1 V t_1 = t'_1 \lambda t_1 = \lambda \qquad [3.13]$$

el valor propio λ, se corresponde con la varianza de la componente principal. Un resultado importante de álgebra lineal nos garantiza por la naturaleza de la matriz *V* que las componentes principales existen. Puesto que *V* es una matriz simétrica y semidefinida positiva (con valores propios mayores o iguales a cero), existe una matriz ortogonal *T* tal que $T'VT = D = diag(\lambda_1, \ldots, \lambda_k)$ con $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{k1} \geq 0$ (Burgos,1994). De esta forma si:

$$Y = (Y_1, \ldots, Y_k) = T'X = \begin{pmatrix} t_{11} & \cdots & t_{k1} \\ \ldots & \ldots & \ldots \\ t_{1k} & \ldots & t_{kk} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \ldots \\ X_k \end{pmatrix}, \quad [3.14]$$

entonces $Y_1, \ldots, Y_k$ son las componentes principales ya que

$$Cov(Y) = Cov(T'X) = T'Cov(X)T = T'VT = \begin{pmatrix} \lambda_1 & \ldots & 0 \\ \ldots & \ldots & \ldots \\ 0 & \ldots & \lambda_{k1} \end{pmatrix} \quad [3.15]$$

$$(Cov(Y_i, Y_j) = 0 \quad y \quad Var(Y_j) = \lambda_j \quad [3.16]$$

Además, si $\lambda_1 > \lambda_2 > \ldots > \lambda_k$, entonces las componentes principales son únicas salvo signo.

Observar que en nuestro caso no tiene sentido que $\lambda_i = 0$, puesto que esto implicaría $Var(Y_i) = 0$, lo que se traduce en que Y_i no es una v.a., sino que se trataría de una constante.

Otro problema importante que se nos presenta es decidir el número de componentes significativas, esto es, que aporten información sobre las v.a. originales. La respuesta más sencilla, es que si queremos mantener toda la información, deberemos quedarnos con todas las componentes principales. En la práctica, se utiliza el siguiente criterio: Si queremos mantener un porcentaje prefijado *p* de la variabilidad inicial deberemos quedarnos con las primeras *m* componentes, que verifiquen:

$$\frac{\lambda_1 + \ldots + \lambda_m}{\lambda_1 + \ldots + \lambda_k} \geq \frac{p}{100} \quad [3.17]$$

En nuestro caso, nos conformamos con mantener un 99% de la información, por tanto, la decisión será buscar el menor índice *m* tal que:

$$\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{k} \lambda_i} \geq 0.99 \quad [3.18]$$

Gracias al ACP, obtenemos un conjunto de v.a. $(Y_1, \ldots, Y_m)$ independientes que contienen la totalidad de la variabilidad del conjunto de datos original salvo en un 1%.

Notar que al tratarse de v.a. independientes la matriz de covarianzas será una matriz diagonal invertible de forma inmediata.

3.2.2.3. Métrica ARMA

A todo modelo ARMA correspondiente a una serie temporal se puede asociar una función racional compleja no nula, denominada función de sistema o de transferencia, H(z).

El cepstrum, tal y como vimos, es la transformada inversa de Fourier del logaritmo del espectro de potencia, y para modelos ARMA se reducirá a:

$$\log\left[H(z)H^{*}(1/z)\right]=\sum_{n=-\infty}^{+\infty}c_n z^{-n} \qquad [3.19]$$

Los términos c_n son los coeficientes cepstrales, los cuales forman una sucesión compleja hermitiana, es decir, $c_n^{*}=c_{-n}$, $\forall n\in Z$.

A continuación, presentamos distintas definiciones de una métrica *d* entre modelos ARMA.

Definición 1: Para dos modelos ARMA *M, M´* con coeficientes cepstrales (c_n), (c'_n) definimos la métrica *d* como (Martin, 2000):

$$d(M,M')=\left(\sum_{n=1}^{\infty}n\left|c_n-c'_n\right|^2\right)^{\frac{1}{2}} \qquad [3.20]$$

Esta métrica *d* que presentamos tiene las siguientes propiedades:

- *d* es en realidad una pseudométrica, porque si (c_n), (c'_n) son iguales $\forall n\in Z-\{0\}$, entonces $d(M,M')=0$.
- *d* es invariante respecto a las traslaciones, es decir:

$$d(MM'',M'M'')=d(M,M') \qquad [3.21]$$

Las siguientes definiciones las referimos a modelos AR, teniendo en cuenta el siguiente comentario: Dados dos modelos ARMA con función de sistema $M=\frac{B}{A}$,

$M' = \frac{B'}{A'}$, se pueden construir dos modelos AR $N = \frac{1}{AB'}$, $N' = \frac{1}{BA'}$ y observar simplemente que $d(M,M') = d(N,N')$, lo que se sigue de la propiedad de invarianza a traslaciones, haciendo M´´= $\frac{1}{BB'}$.

Definición 2: Para dos modelos AR estables *M, M´* de orden *p, p´* con parámetros (a_j), (a'_j), definimos los polinomios asociados (Martin, 2000):

$$A(z) = \sum_{j=0}^{p} a_j z^{p-j}\ ,\ A^{+}(z) = \sum_{j=0}^{p} a_j^{*} z^{j} \qquad [3.22]$$

y análogamente para A´, A´$^{+}$; entonces

$$d(M.M') = \left[\log \frac{res(A, A'^{+}) \cdot res(A', A^{+})}{res(A, A^{+}) \cdot res(A', A'^{+})} \right]^{\frac{1}{2}} \qquad [3.23]$$

Con *res(A,B)* denotamos la resultante de los polinomios *A, B* que se define como,

$$res(A,B) = \begin{vmatrix} a_0 & a_1 & \dots & a_m & & \\ & a_0 & a_1 & \dots & a_m & \} n \cdot filas \\ & & \dots & \dots & \dots & \\ b_0 & b_1 & \dots & b_n & \dots & \\ & b_0 & b_1 & \dots & b_n & \} m \cdot filas \\ & & \dots & \dots & \dots & \end{vmatrix} \qquad [3.24]$$

en el que los espacios vacíos se llenan con ceros, su diagonal principal viene dada por:

$$\left(\overbrace{a_0, \dots, a_0}^{n}, \overbrace{b_n, \dots, b_n}^{m} \right) \text{ y } A(z) = a_0 z^m + a_1 z^{m-1} + \dots + a_m\ ,\ B(z) = b_0 z^n + b_1 z^{n-1} + \dots + b_n$$

Definición 3: Para dos modelos AR estables *M, M´* de orden *p, p´* con polos (α_j), (α'_j) es (Martin, 2000):

$$d(M.M') = \left[\log \frac{\prod_{i=1}^{p}\prod_{j=1}^{p'}(1-\alpha_i\alpha'^{*}_{j})\prod_{i=1}^{p'}\prod_{j=1}^{p}(1-\alpha'_{i}\alpha^{*}_{j})}{\prod_{i=1}^{p}\prod_{j=1}^{p}(1-\alpha_i\alpha^{*}_{j})\prod_{i=1}^{p'}\prod_{j=1}^{p'}(1-\alpha'_{i}\alpha'^{*}_{j})} \right]^{\frac{1}{2}} \qquad [3.25]$$

La equivalencia de estas definiciones es fácil de probar:

- Para las definiciones 1 y 2 los coeficientes cepstrales AR pueden ser obtenidos por $c_n = \sum_{j=1}^{p} \frac{\alpha_j^n}{n}$, $\forall n \in N$. Así, para dos modelos AR estables *M, M´* de órdenes *p, p´* con polos (α_j), (α'_j) se puede escribir

$$\delta(M,M') = \left(\sum_{n=1}^{\infty} n|c_n - c'_n|^2 \right)^{\frac{1}{2}} = \left(\sum_{n=1}^{\infty} n(c_n - c'_n)(c_n^* - c'^*_n) \right)^{\frac{1}{2}}$$

$$= \left(\sum_{n=1}^{\infty} n \left(\sum_{j=1}^{p} \frac{\alpha_j^n}{n} - \sum_{j=1}^{p'} \frac{\alpha'^n_j}{n} \right) \left(\sum_{j=1}^{p} \frac{\alpha^{*n}_j}{n} - \sum_{j=1}^{p'} \frac{\alpha'^{*n}_j}{n} \right) \right)^{\frac{1}{2}} =$$

$$= \left(\sum_{n=1}^{\infty} \left(\sum_{i=1}^{p}\sum_{j=1}^{p} \frac{\alpha_i^n \alpha^{*n}_j}{n} + \sum_{i=1}^{p'}\sum_{j=1}^{p'} \frac{\alpha'^n_i \alpha'^{*n}_j}{n} - \sum_{i=1}^{p}\sum_{j=1}^{p'} \frac{\alpha_i^n \alpha'^{*n}_j}{n} - \sum_{i=1}^{p}\sum_{j=1}^{p'} \frac{\alpha^{*n}_i \alpha'^n_j}{n} \right) \right)^{\frac{1}{2}}$$

Utilizando ahora la relación: $\sum_{n=1}^{\infty} \frac{\alpha^n}{n} = -\log(1-\alpha)$, $(|\alpha| < 1)$, se sigue de forma inmediata [3.25].

Nótese que la condición de estabilidad que se exige a los modelos AR, se debe a que la anterior relación sólo se verifica si α está en el interior del círculo unidad.

Es importante también observar que [2.79] permite calcular algorítmicamente *la métrica ARMA como un producto finito en el dominio polo-cero*, probando así que la suma infinita $\left(\sum_{n=1}^{\infty} n|c_n - c'_n|^2 \right)^{\frac{1}{2}}$ converge.

- La equivalencia entre las definiciones 2 y 3, se puede mostrar a través de las propiedades elementales de la resultante. Para un modelo AR estable *M* de orden

p y polos *(α_j),* su cepstrum se puede definir, [], $c_n = \sum_{j=1}^{p} \frac{\alpha_j^n}{n}$, $\forall n \in N$. Así, para dos modelos AR estables *M, M´* de órdenes *p, p´* y polos (α_j), (α'_j) se puede escribir

$$f(z) = f_0 z^m + f_1 z^{m-1} + \ldots + f_m$$
$$g(z) = g_0 z^n + g_1 z^{n-1} + \ldots + g_n \qquad [3.26]$$

la resultante viene dada por

$$res(f,g) = \begin{vmatrix} f_0 & f_1 & \cdots & f_m & & \\ & f_0 & f_1 & \cdots & f_m & \} n \cdot filas \\ & & \cdots & \cdots & \cdots & \\ g_0 & g_1 & \cdots & g_n & \cdots & \\ & g_0 & g_1 & \cdots & g_n & \} m \cdot filas \\ & & \cdots & \cdots & \cdots & \end{vmatrix} \qquad [3.27]$$

Factorizando los polinomios *f* y *g*

$$f(z) = f_0 \prod_{i=1}^{m} (z - u_i)$$
$$g(z) = g_0 \prod_{j=1}^{n} (z - v_j) \qquad [3.28]$$

se puede escribir

$$res(f,g) = f_0^n g_0^m \prod_{i=1}^{m} \prod_{j=1}^{n} (u_i - v_j) \qquad [3.29]$$

lo que evidencia el siguiente resultado: *res(f,g)=0 si y sólo si f y g tienen un factor en común.* Algebraicamente, la importancia de este resultado estriba en que muestra la presencia de un factor común sin necesidad de factorizar los polinomios

A partir de la expresión anterior de *res(f,g),* se sigue de forma inmediata

$$res(f, g^{+}) = f_0^{n} g_0^{*m} \prod_{i=1}^{m} \prod_{j=1}^{n} (1 - u_i v_j^{*})$$

De estos resultados se deduce directamente la equivalencia entre las definiciones 2 y 3.

En el capítulo 5 presentaremos una comparación de resultados obtenidos entre estas dos técnicas de comparación.

3.2.3. Comparaciones según el tipo de reconocimiento: Verificación o Identificación.

El número de comparaciones que habrá que realizar será distinto según se trate de un sistema de verificación o identificación. Veamos por lo separado.

3.2.3.1. Verificación de locutores.

Recordemos que en los sistemas de verificación de locutores una identidad es reclamada por el usuario del sistema, y a este se le requiere tomar una decisión estrictamente binaria, aceptar o rechazar la identidad reclamada (Rabiner, L. y Schafer, R., 1978). Tradicionalmente en este tipo de sistemas lo que se ha hecho es realizar una sola comparación entre el modelo del registro de entrada y el patrón de referencia del locutor cuya identidad se reclama para tomar la decisión final de aceptar o rechazar la identidad reclamada. En función del coste que tenga el error del sistema, se seleccionaba un umbral apropiado. Este método no parece ser el más objetivo, ya que deja prácticamente la decisión en manos del perito que maneja el sistema.

En nuestro sistema de verificación crearemos dos modelos de referencia. El primero al igual que en el método clásico perteneciente al locutor cuya identidad se reclama, y el segundo formado a partir de todas las muestras del resto de locutores. Finalmente el modelo de entrada se comparará con estos dos modelos de referencia y en función de su distancia con ellos, en la etapa de decisión se aceptará o rechazará la identidad reclamada.

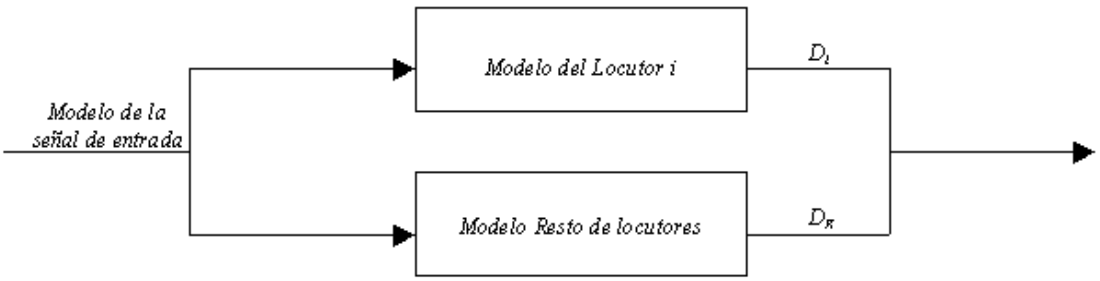


Figura 3.2: *Etapa de comparación de un sistema de verificación*

3.2.3.2. Identificación de locutores.

En este caso al sistema se le pide que clasifique la señal de entrada como perteneciente a alguno de los posibles *N* locutores. En este caso se requerirán N comparaciones entre el modelo de la señal de entrada y los *N* modelos de referencia de los locutores almacenados en memoria (Rabiner, L. y Schafer, R., 1978).

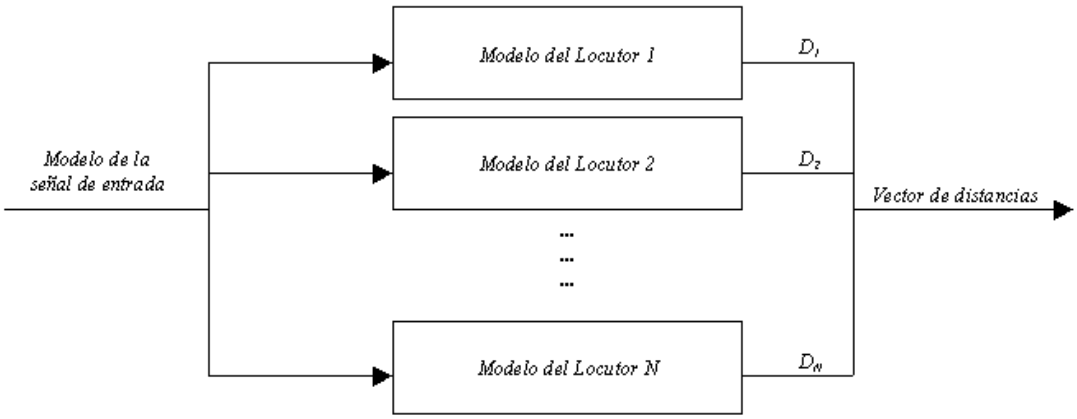


Figura 3.3: *Etapa de comparación de un sistema de identificación*

3.3. Etapa de clasificación

La regla de decisión o clasificación más simple y utilizada es la del vecino más cercano o la de la distancia mínima (Rosenberg, A.,1976). En el sistema de identificación la señal de entrada será clasificada como perteneciente al locutor con el que tenga un mayor grado de similitud o lo que es lo mismo le separe una menor distancia. En el sistema de verificación se aceptará la identidad si la distancia calculada es menor respecto al modelo del locutor cuya identidad se reclama, y en caso contrario se clasificará como perteneciente a alguno de los otros locutores sin especificar cual, simplemente la identidad es rechazada.

Capítulo 4: Introducción a LabVIEW®

El entorno de programación LabVIEW®
Descripción de LabVIEW®

4. Introducción a LabVIEW®

4.1. El entorno de programación LabVIEW®

LabVIEW es un entorno gráfico de programación orientada a objetos, esto es, un lenguaje de programación, como también lo son BASIC, Pascal ó C/C++, pero con unas singularidades que lo diferencian de todos ellos y lo hacen recomendable para la simulación de aparatos de instrumentación y el desarrollo de aplicaciones de adquisición automática de datos. Sus principales características son:

a) Programación orientada a objetos.

La evolución de la ingeniería del software desde sus comienzos ha estado encaminada a soportar la complejidad creciente de los programas.

En las primeras computadoras se programaba directamente en el lenguaje que entendía la máquina, es decir, el lenguaje binario (compuesto por ceros y unos); después se pasó al lenguaje ensamblador (que no es más que una codificación del primero), y más tarde, a los llamados lenguajes de alto nivel no estructurados), finalmente se diseñaron lenguajes estructurados, como Pascal.

En cada uno de estos pasos, se han conservado las mejores características de los existentes, y se han añadido otras para crear los nuevos métodos de programación.

Tanto el lenguaje C como el C++ no son técnicamente estructurados, ya que no permiten crear funciones desde el interior del cuerpo de una función, pero generalmente son considerados lenguajes estructurados.

Como variante de los lenguajes estructurados, existen los lenguajes modulares, como LabVIEW; son lenguajes estructurados en los que el proceso de declaración de funciones, procedimientos, variables, son transparentes al usuario.

La última generación de lenguajes de programación, utilizan el concepto de Programación Orientada a Objetos (POO), ésta permite descomponer fácilmente un problema en subgrupos de partes relacionadas (conserva esta ventaja de la programación estructurada), y traducir estos subgrupos en unidades auto-contenidas llamadas objetos.

Se entiende por *objeto* una entidad lógica que contiene datos y código que manipula estos datos; por ejemplo, en LabVIEW es posible utilizar una clase de objetos denominada "waveform graph" que es capaz de almacenar, manipular y presentar en pantalla una gráfica, en la figura 4.1 se presenta el control de un objeto como el comentado, y en la figura 4.2 se presenta el código del objeto.

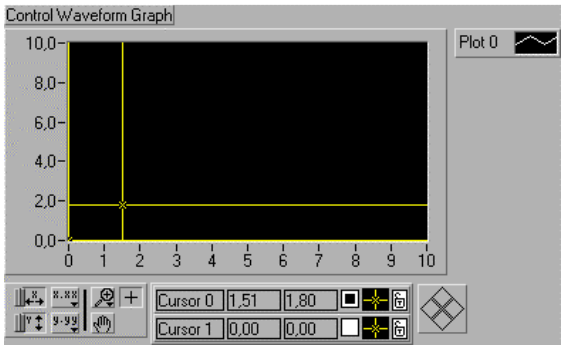


Figura 4.1:*Control del objeto Waveform Graph*

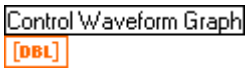


Figura 4.2: *Código del objeto Waveform Graph*

b) Programación gráfica.

La presentación gráfica de un interface usuario-código en un programa, no solamente interesa desde el punto de vista estético, también afecta a la facilidad de manejo y a la claridad con que los resultados del programa son mostrados al usuario final de la aplicación.

Por ello se han desarrollado entornos de programación gráfica, ya sea basados en lenguajes existentes anteriormente y orientados a objetos (como eran C/C++ o Pascal), otros se han actualizado a la programación orientada a objetos (es el caso del BASIC), y han surgido también lenguajes nuevos, como LabVIEW.

Quizá la mayor particularidad que se puede resaltar de LabVIEW con respecto a otros lenguajes orientados a objetos con interface gráfico, como pueden ser Visual BASIC, Delphi o Visual C++, es que no sólo el interface que se presenta al usuario final de la aplicación es gráfico, sino que incluso el código lo es, a diferencia de los anteriormente citados.

El código del programa se desarrolla en un lenguaje denominado G (*Graphical Programming Language*), que podemos describir como un diagrama de bloques considerablemente detallista, en el que, mediante símbolos gráficos, podemos especificar las acciones que ha de realizar el programa.

c) Librerías de funciones.

Otra característica importante de LabVIEW reside en su extremada jerarquización, que permite la revisión del código desde una visión global de la aplicación hasta el más pequeño detalle, y un control preciso del flujo del programa.

Pero la potencia de LabVIEW como lenguaje de programación en aplicaciones de adquisición de datos y simulación de instrumentos se apoya en las librerías de funciones que lo componen.

Desde la más simple de las funciones hasta el más complicado de los programas mantienen la filosofía de "caja negra" característica de los lenguajes modulares y estructurados que consiste en tratar a los procedimientos y funciones de un programa como "cajas" con entradas y salidas, de modo que se pueden construir aplicaciones de gran complejidad utilizando funciones y procedimientos cuyo código no se ha programado especialmente para dicha aplicación, o incluso no ha

programado la misma persona; Estas funciones pueden a su vez contener otras, creadas anteriormente, con el fin de jerarquizar el desarrollo y simplificar el código.

Dichas funciones se reúnen en conjuntos llamados "Librerías de Funciones", que suelen agrupar funciones relacionadas con una misma actividad, como la Librería de Funciones Matemáticas, la Librería DAQ, de adquisición de datos, o bien, las relacionadas con un proyecto particular, con el fin de diferenciarlas del resto y agilizar el proceso de transformación del programa a un archivo ejecutable, o también por una simple cuestión organizativa.

Es posible crear nuevas librerías de funciones en LabVIEW a partir de las existentes, o introduciendo código nativo de otro lenguaje, como C (en particular el dialecto desarrollado por *National Instruments*, Watcom-C), o llamadas al sistema operativo, o al entorno del sistema operativo, de este modo, es posible utilizar librerías de funciones de *Windows* (recopiladas en los archivos de extensión (.DLL)), para crear, a su vez, funciones que integren librerías de LabVIEW (archivos de extensión (.LLB)).

Existen multitud de librerías de LabVIEW diseñadas para propósito general, y también librerías especializadas en el control de hardware específico, que son proporcionadas por la casa *National Instruments*.

4.2. Descripción de LabVIEW

Los programas creados bajo LabVIEW se llaman Instrumentos Virtuales (*Virtual Instruments*, *VIs*), porque imitan a los instrumentos reales en su apariencia y modo de manejo. Cada VI posee tres partes, el Panel Frontal (*Front Panel*), el Diagrama de Bloques (*Block Diagram*), y el Icono/Conector.

Es posible cambiar la ventana en que se trabaja con la opción "Show Diagram", si se quiere pasar al Diagrama de Bloques, o la opción "Show Panel", para el Panel Frontal, que aparecen en el menú "Windows" según la situación de trabajo actual, pero se accede al Icono/Conector mediante doble-click o click-derecho sobre el mismo, siempre desde la ventana "Panel".

a) El Panel Frontal (*Front Panel*)

Se denomina así al interface gráfico que el usuario de la aplicación manejará de modo interactivo.

La apariencia del Front Panel vacío es la de la figura 4.3, y la de un Front Panel de una aplicación puede ser la de la figura 4.4. En el Panel Frontal es posible situar botones, LEDs, indicadores, gráficos y otros objetos con los que el usuario interactuará y a través de los cuales obtendrá los resultados de los procesos para los cuales fue diseñado el código.

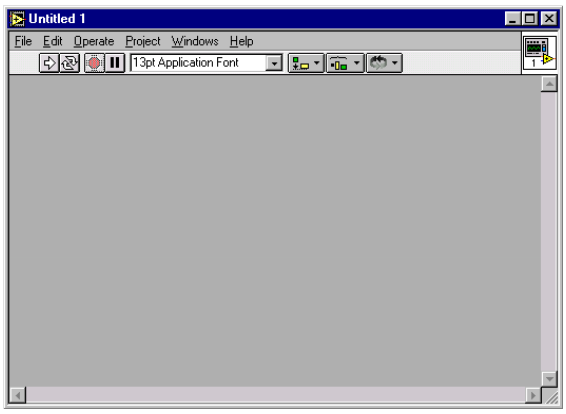


Figura 4.3: *Panel Frontal vacío*

En el Panel Frontal pueden existir dos clases de elementos, los controles, mediante los cuales, el usuario de la aplicación puede introducir datos necesarios para el proceso, y los indicadores, a través de los cuales, el programa proporciona resultados al usuario; ambos poseen una forma en el Panel Frontal y el programa accede a los datos que proporcionan (los controles) o se les suministra (en el caso de los indicadores) desde el Diagrama de Bloques a través de un icono llamado terminal.

Existen multitud de controles e indicadores distintos, desde botones que manejan variables booleanas hasta "cajas de ruta" (*path boxes*) que manejan rutas de archivos, y todos ellos están incluidos en la Paleta de Controles (*Controls Palette*), que es posible visualizar desde la opción "Show Controls Palette" del menú "Windows", siempre que estemos en la ventana "Panel".

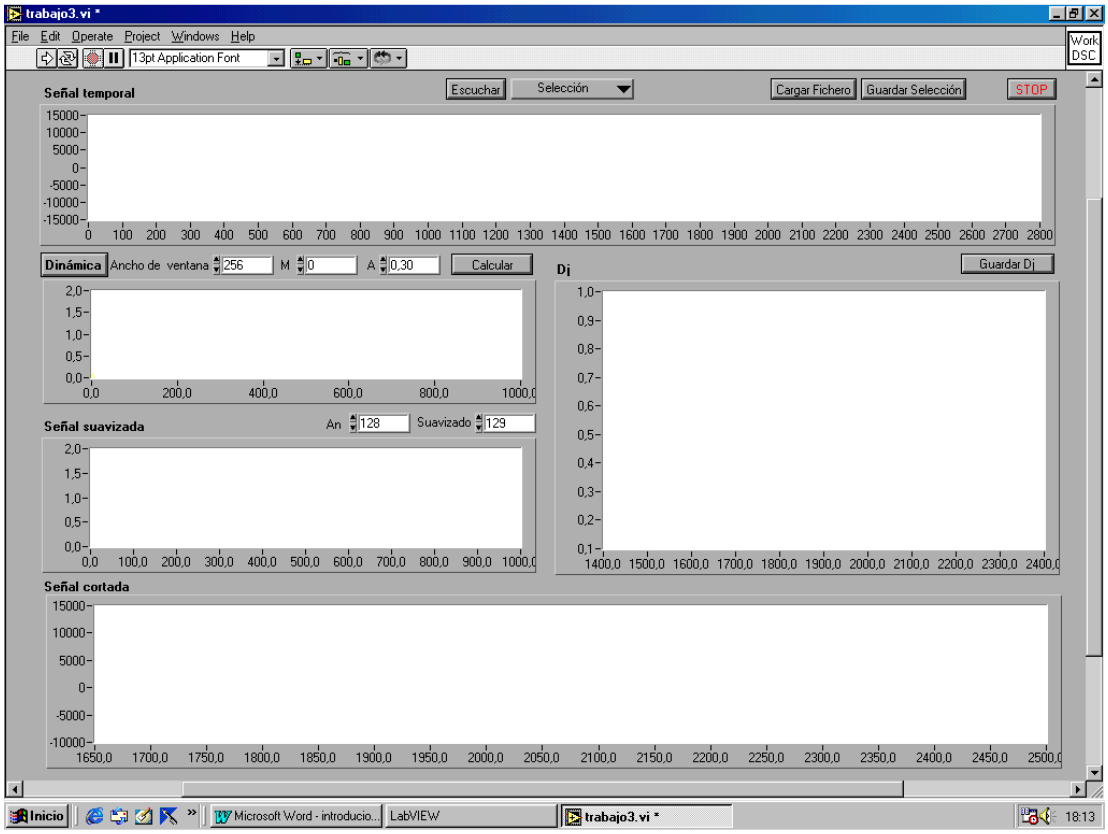


Figura 4.4*: Ejemplo de Panel Frontal (Programa1.23.vi).*

b) El Diagrama de Bloques (*Block Diagram*)

Es la ventana de LabVIEW en la cual el programador construye la aplicación mediante un código denominado lenguaje G.

La apariencia del Diagrama de Bloques vacío es la de la figura 5, y la de un Diagrama de Bloques con Código G es la de la figura 6

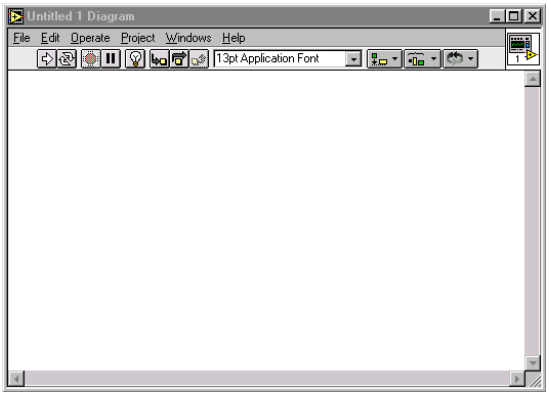


Figura 4.5: *Diagrama de Bloques vacío.*

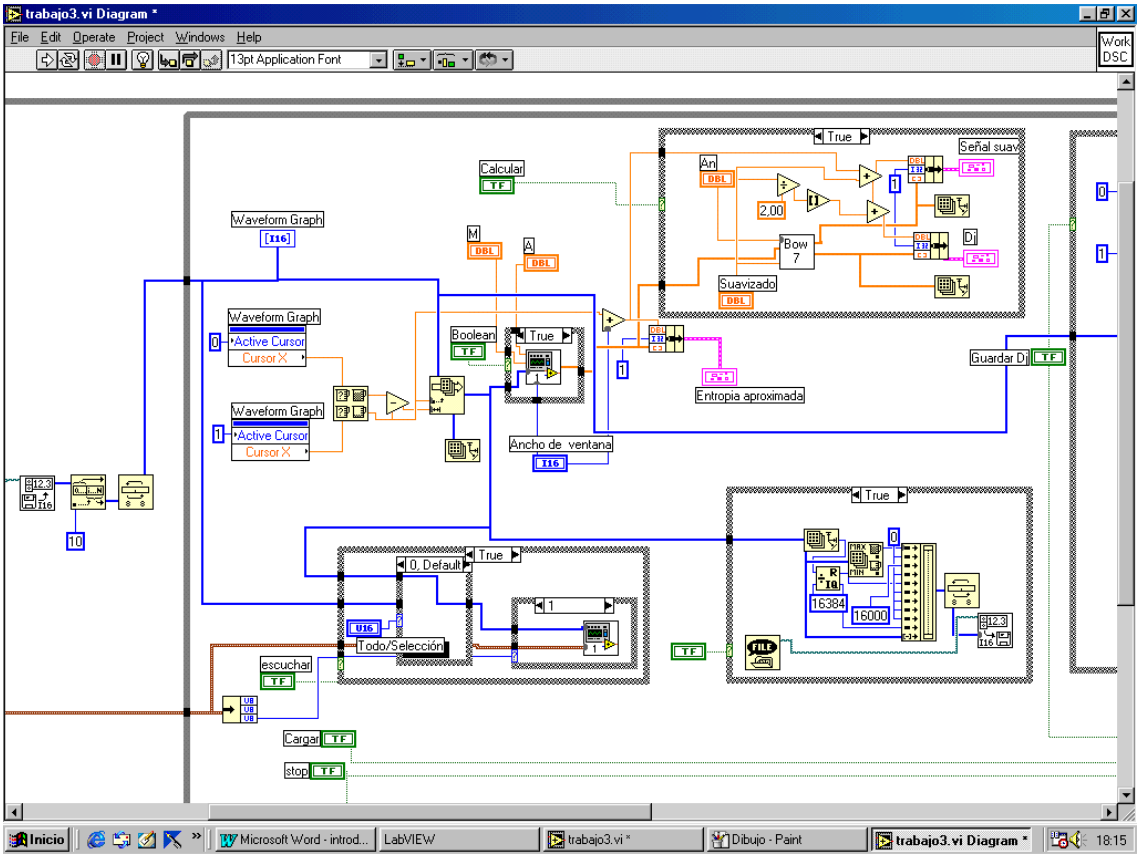


Figura 4.6 *Ejemplo de Diagrama de Bloques.*

Los Diagramas de Bloques no tienen por qué ser complicados en principio, si se ha diseñado la aplicación correctamente, usando la jerarquización y la filosofía de "cajas negras" para las funciones, y una organización lógica del código; también es posible colocar comentarios para aclarar el código a otras personas o para futuras revisiones.

Sin embargo, un diagrama puede ser extremadamente complicado si no ha sido diseñado correctamente.

Todas las funciones que componen el código G son accesibles desde la Paleta de Funciones (*Functions Palette*), que es posible visualizar desde la opción "Show Functions Palette" del menú "Windows", siempre que estemos en la ventana "Diagram".

c) El Icono/Conector

Como ya se ha explicado, es posible utilizar VIs dentro de otros VIs, entonces a los primeros se les denomina subVIs, y se representan en el diagrama de bloques mediante un Icono similar al de cualquier función nativa de una librería LabVIEW. Es posible conectar estos Iconos como cualquier función nativa, pasándoles los parámetros que requieran, siempre que hayan sido adecuadamente preparados para ello.

Así, es posible dividir una aplicación en una serie de tareas, que se pueden dividir de nuevo, hasta conseguir que la complicada aplicación original se reduzca a una serie de simples procedimientos.

Gracias a la filosofía de "cajas negras" y a la jerarquización de VIs, es posible para un programador utilizar funciones de una librería creada por otro programador sin necesidad de conocer cómo han sido programadas, únicamente sabiendo los parámetros que la función precisa y los que devuelve. Esta información generalmente se asocia al Icono mediante la opción "Show VI Info" del menú "Windows" de la barra de comandos de LabVIEW.

Con el fin de simplificar futuras utilizaciones de las funciones creadas, es aconsejable la organización, jerarquización e información en una librería de funciones.

En la figura 3.7 está representado el iconos de una función nativa de LabVIEW (la función Index Array.vi) y en la figura 4.8 el de una aplicación desarrollada a partir de las funciones naticas

Figura 4.7 *Icono de Index Array.vi*

Figura 4.8 *Icono de Ejemplo.vi*

Capítulo 5: Programación de Algoritmos

Algoritmos de la dinámica no lineal
Algoritmos para la obtención de un modelo ARMA
Algoritmo de cálculo de la distancia ARMA
Algoritmos para el reconocimiento de locutores

5. Programación de algoritmos

En este capítulo presentaremos los programas implementados en el entorno LabVIEW durante el desarrollo de este proyecto. La forma de presentarlos será la siguiente: En un primer cuadro se indicarán las entradas y salidas de cada programa y de que tipo son (real, entero,..). Además en esta tabla se incluirá el icono / conector por el cual lo reconoceremos. Seguidamente explicaremos el diagrama (código) del programa incluyendo las ventanas Diagrama de cada programa.

A continuación presentamos los programas desarrollados, los cuales hemos divido en dos grandes grupos:

El primero de ellos contendrá todos aquellos relacionados con la caracterización de las señales del habla a partir de los métodos de la dinámica no lineal, estos son:

- Algoritmo de cálculo de la Entropía de Renyi
    - Rutina para el cálculo de la Correlación Acumulada
- Algoritmo de cálculo de la Entropía Aproximada
- Algoritmo de cálculo rápido de la Entropía Aproximada
- Algoritmo de cálculo da la Dinámica Simbólica

El segundo grupo estará formado por todos aquellos programas relacionados con el modelado de señales del habla (modelado ARMA) y con el reconocimiento de locutores:

- Algoritmo para la obtención del modelo ARMA.
    - Rutina para el cálculo de los coeficientes AR.
    - Rutina para el cálculo del espectro de potencia.
    - Rutina para el cálculo del factor de ganancia LPC.
    - Rutina para el cálculo de los coeficientes MA.
    - Rutina para la obtención de la envolvente ARMA.
- Algoritmo para el cálculo de la distancia ARMA entre dos modelos.
    - Rutina para el cálculo de la resultante.
    - Rutina para la reflexión de las raíces.
    - Rutina para el cálculo de la distancia.
- Algoritmos para el reconocimiento de locutores.
    - Algoritmo para la identificación de locutores.
    - Algoritmo para la verificación de locutores.

5.1. Algoritmos de la dinámica no lineal

5.1.1. Algoritmo para el cálculo de la Entropía de Renyi de orden 2

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *Tamaño Ventana [Entero, 16 bits]* *a [Real, DBL]* *d [Entero, 8 bits]*	*Dimensión de correlación [Real, DBL]* *Entropía [Real, DBL]*	

El algoritmo para caracterizar una señal mediante el cálculo de la *Entropía de Renyi de orden 2* (K_2) se basa en desplazar a lo largo de ella una ventana y calcular el valor de la Entropía para el tramo de señal contenido en la ventana. La entropía se calculará según lo expuesto en el apartado [2.1.2.1.]. También calcularemos el valor de la *Dimensión de Correlación*, ya que como dijimos puede ser utilizada para caracterizar la señal, y se trata de un paso intermedio para calcular la Entropía. Una vez calculados los valores de *Entropía de Renyi* y *Dimensión de Correlación* se desplazará la ventana un punto y se repite la operación. Esta operación se repetirá un número de veces dependiente del tamaño de la señal (N_s) y del ancho de la ventana (L) que se desplaza (concretamente N_s-L+1), por esto es necesario incluirla dentro de un bucle.

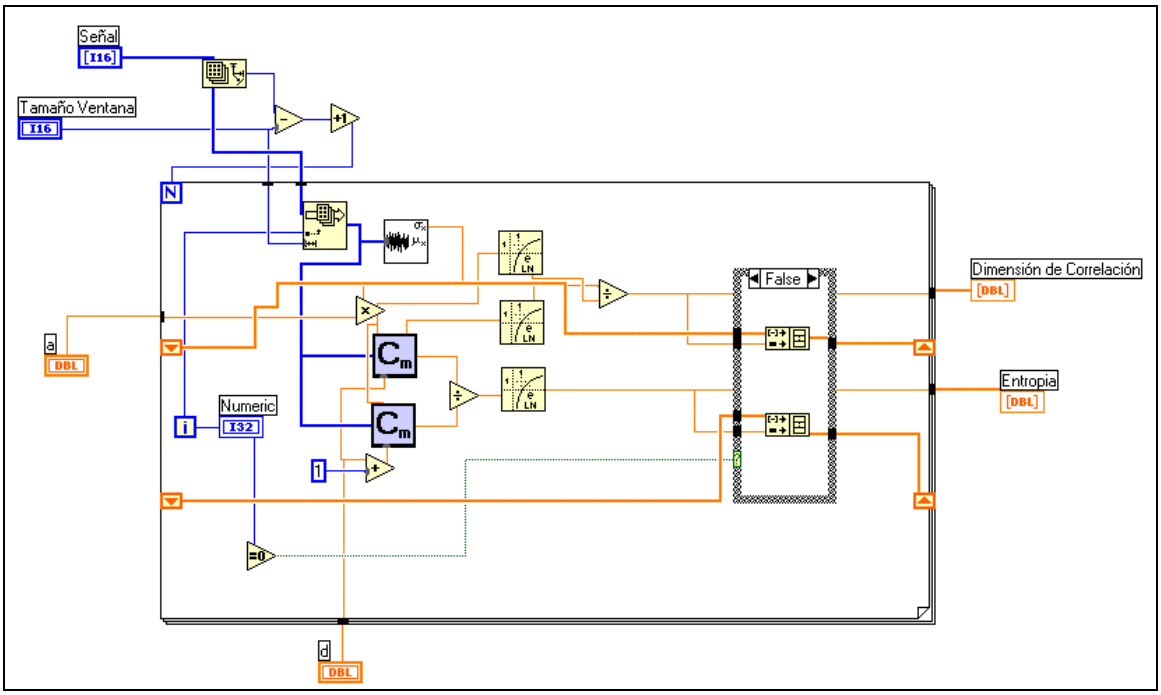


Figura 5.1: *Diagrama de Entropia de Renyi+dimension correlacion.vi*

Como vemos en el anterior diagrama se calculan por separado las Correlaciones acumuladas C_m y C_{m+1} mediante la rutina *ErgSim.vi* que más tarde describiremos. El umbral de comparación lo obtendremos al multiplicar la desviación típica (*SD*) de la ventana por un factor *a*. A partir de la fórmula 2.1 obtenemos la *Dimensión de Correlación* dividiendo C_m por el logaritmo neperiano del umbral. La obtención del valor de *entropía* para la ventana es una aplicación directa de la fórmula 2.11. Todas estas operaciones se engloban dentro de un bucle *For* que permite repetir las operaciones desplazando la ventana por toda la señal.

5.1.1.1. Rutina para el cálculo de la Correlación Acumulada

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *d [Entero, 8 bits]* *Umbral [Real, DBL]*	*Correlación acumulada [Real, DBL]*	

El algoritmo para el cálculo de la *Correlación Acumulada* (C_m) se basa en la fórmula 2.3. Si analizamos la fórmula llegamos a la conclusión de que al comparar los vectores obtendremos una matriz cuadrada y triangular, estando la diagonal principal formada por ceros. Así pues para que el computo sea más rápido es posible obtener el resultado final calculando sólo la parte triangular superior de la matriz y asignando ceros a la parte inferior. Una vez contabilizados todos los puntos que son inferiores al umbral (suma de todos los 1 de la matriz) multiplicamos por dos (compensamos la parte inferior que no hemos calculado) y restamos un número equivalente al tamaño de la diagonal principal (que siempre estará formada por 1, ya que resulta de restar un vector por sí mismo lo que dará 0, y al compararlo con el umbral siempre será él). Finalmente sólo resta dividir el número obtenido por el tamaño de la señal al cuadrado. El diagrama correspondiente a este algoritmo lo tenemos en la siguiente figura:

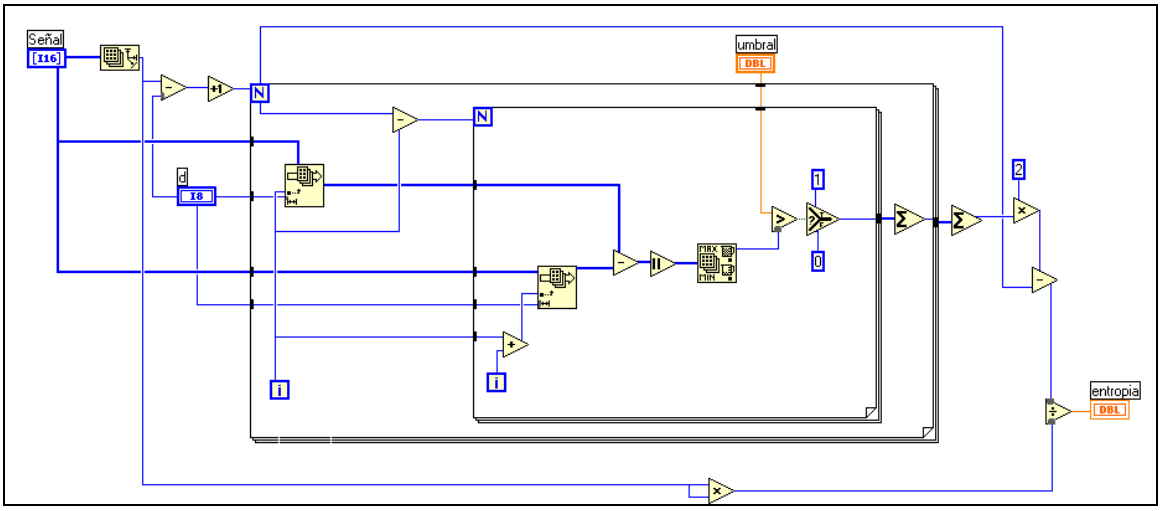


Figura 5.2: *Diagrama de ErgSim.vi*

5.1.2. Algoritmo para el cálculo de la Entropía Aproximada

Entradas	Salidas	Icono
Entrada [Entero, 16bits] *Ancho de ventana [Entero, 16 bits]* *A [Real, DBL]* *M [Entero, 8 bits]*	*Salida [Real, DBL]*	

El algoritmo para el cálculo de la entropía aproximada se basa en desplazar a lo largo de la señal una ventana. Para esa ventana calculamos su valor de entropía según lo expuesto en el apartado 2.1.2.2. Una vez calculado este valor, se desplaza la ventana un punto y se repite la operación. Esta operación se repetirá, dentro de un bucle, un número de veces dependiente del tamaño de la señal (N_s) y del ancho de la ventana (L) que se desplaza, concretamente N_s-L+1.

El algoritmo para calcular la entropía aproximada para cada ventana se realizará de la siguiente manera:

- Tomamos una secuencia formada por los primeros *M* puntos de la ventana y mediante un bucle *For* calculamos su diferencia punto a punto respecto a todas las posibles secuencias de *M* puntos hasta aquella que empieza en el (*L-M*)-*iesimo* punto.
- Comparamos la diferencia máxima (en valor absoluto) de cada secuencia de diferencias con un umbral formado por la desviación típica de la señal (*SD*) multiplicada por un factor *a* y contabilizamos el número de veces que la diferencia es menor que el umbral. En este paso establecemos como de repetitiva es la señal.
- Paralelamente se repite la misma operación para *M+1*. En este paso establecemos con que bondad cada acierto obtenido en el paso anterior predice el valor siguiente de la secuencia.
- Dividimos el número de aciertos en *M+1* por el número de aciertos en *M* y obtenemos el logaritmo del cociente .
- Mediante un bucle *For* repetimos las operaciones anteriores pero empezando las secuencias de comparación en el segundo, tercero,… punto hasta el (*L-M*)-*iesimo* punto.
- Finalmente sumamos todos los logaritmos obtenidos, dividimos la suma por (L-M) y cambiamos el signo del resultado final. Este cambio de signo es debe a que el cociente entre aciertos en *M+1* y *M* dará siempre un número entre 0 y 1, siendo su logaritmo neperiano un número negativo.

El resultado obtenido será el valor de la *Entropía Aproximada (M,a,L).* Desplazamos la ventana un punto y repetimos todas las operaciones hasta obtener la secuencia de valores de entropía aproximada de la señal. En la siguiente figura tenemos el diagrama del algoritmo implementado en *Labview 5.1*:

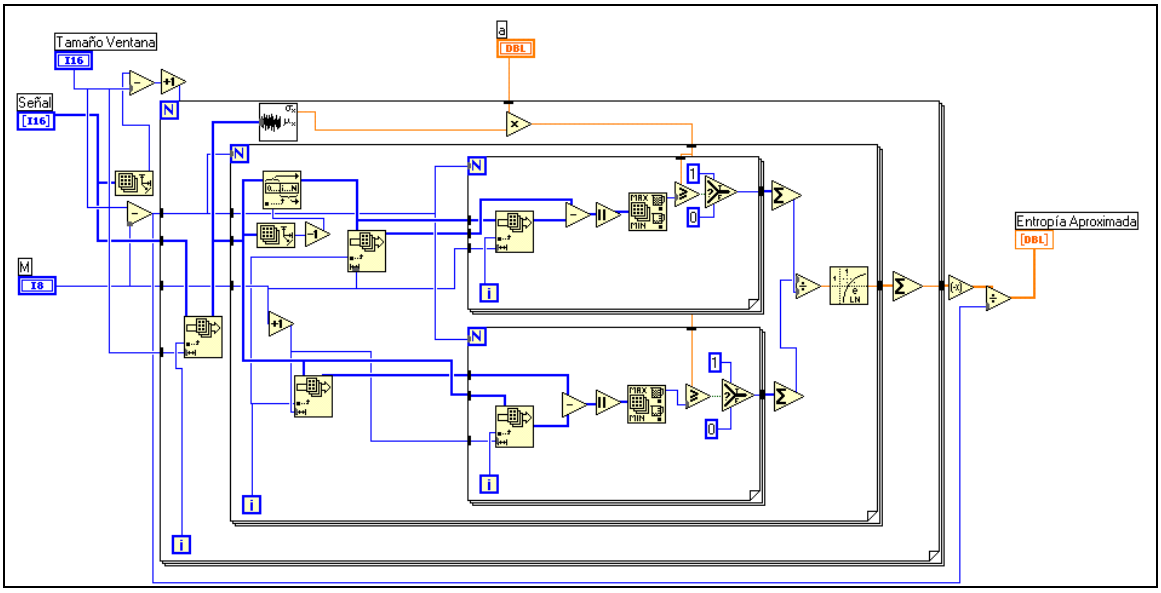


Figura 5.3: *Diagrama de Entropiamodificada.vi*

5.1.3. Algoritmo rápido para el cálculo de la Entropía Aproximada

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *Ancho de ventana [Entero, 16 bits]* *A [Real, DBL]* *M [Entero, 8 bits]*	*Entropía [Real, DBL]*	ER

El algoritmo rápido que aquí presentamos se basa en el desplazamiento de una ventana a lo largo de la señal. A través de una rutina, que a continuación explicaremos, calculamos el valor de la entropía aproximada para esa ventana. Una vez calculado este valor, la ventana se desplaza un punto sobre la señal y se calcula un nuevo valor de entropía aproximada.

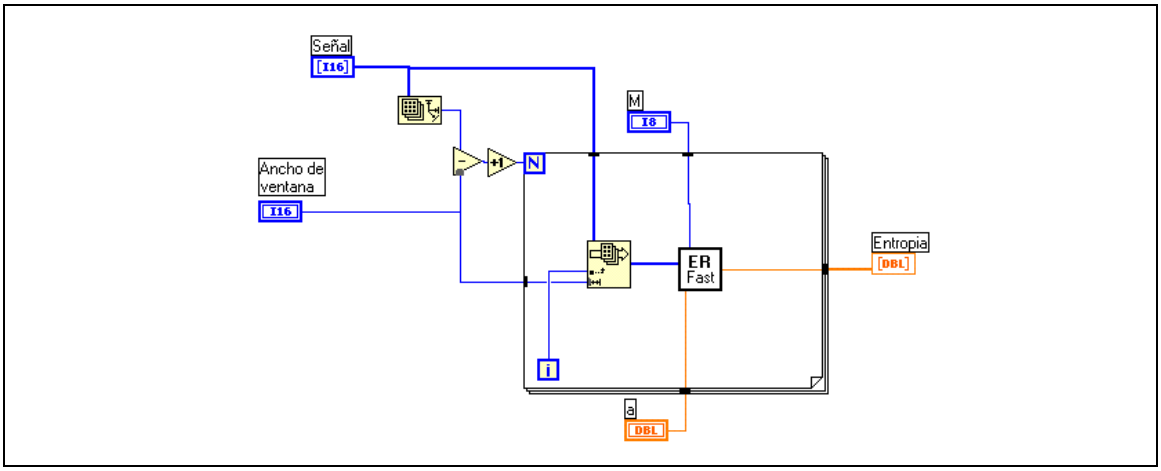


Figura 5.4: *Diagrama de Entropía rápida general.vi*

En la figura 5.4 vemos el esquema general, en el cual la ventana se desplaza gracias a un bucle *For* y desde el se llama a la rutina de cálculo de la entropía aproximada.

5.1.3.1. Rutina para el cálculo rápido de la Entropía Aproximada de una ventana

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *Ancho de ventana [Entero, 16 bits]* *A [Real, DBL]* *M [Entero, 8 bits]*	*Entropía [Real, DBL]*	

El presente algoritmo se basa en la organización de las operaciones que había que realizar para calcular la entropía aproximada. Tal y como vimos en el apartado anterior es necesario realizar una serie de diferencias entre secuencias, muchas de las cuales se repiten. Un método para calcular estas diferencias una sola vez es crear una matriz *D* cuadrada (*L x L*) y simétrica ($D_{ij}=D_{ji}$) donde almacenamos las diferencias (en valor absoluto) entre todos los puntos de la ventana. Para mayor ahorro comparamos las diferencias con el umbral indicado en el apartado anterior (*SD x a*). Si la diferencia es inferior ponemos un 1 en su posición y si es superior 0.

Una vez calculada la matriz *D* formada por unos y ceros que me relaciona la diferencia entre puntos con el valor umbral, para comparar una secuencia de tamaño *M* cualquiera, que comienza en el punto x_i con todas las posibles secuencias, consultaré las diagonales de la matriz *D* de tamaño *M* y que comienzan en la fila *i-esima*. Por ejemplo, para *M=2* consultaré los valores $D_{i,j}$ y $D_{(i+1),(j+1)}$. Si se da el caso de que ambos valores sean 1, esto significa que de haber realizado la diferencia entre secuencias, ninguno de los valores obtenidos hubiera superado el valor de umbral. En el apartado anterior habríamos introducido un 1 al sumatorio. Si alguno de los valores hubiera sido cero, esto significa que de haber realizado la diferencia entre secuencias, alguno de los valores obtenidos hubiera superado el valor de umbral. En el apartado anterior habríamos introducido un 0 al sumatorio.

En este algoritmo introducimos otra medida de ahorro computacional al calcular el caso *M+1* . No calcularemos C_m y C_{m+1} por separado y en paralelo, si no consecutivamente. Si al comparar dos secuencias de tamaño *M* obtenemos que alguna de las diferencias pasa del umbral es evidente que al estudiar estas mismas secuencias pero con un elemento más (caso *M+1*), seguirá existiendo esa diferencia superior al umbral, luego no es necesario estudiar ese caso. En los sumatorios para el calculo de C_m y C_{m+1} introduciremos 0. Sin embargo en el caso de que todos los elementos estudiados sean 1, será necesario consultar un elemento más de la diagonal. Concretamente en el caso visto ahora evaluaríamos *M=3*, luego deberíamos estudiar el elemento $D_{(i+2),(j+2)}$. Si este es 0 en el sumatorio para calcular C_{m+1} introduciríamos un 0, ya que la comparación introducida sobrepasa el umbral. Si es 1 introducimos un 1 al sumatorio, ya que significaría que ninguna de las comparaciones sobrepasa el umbral.

:

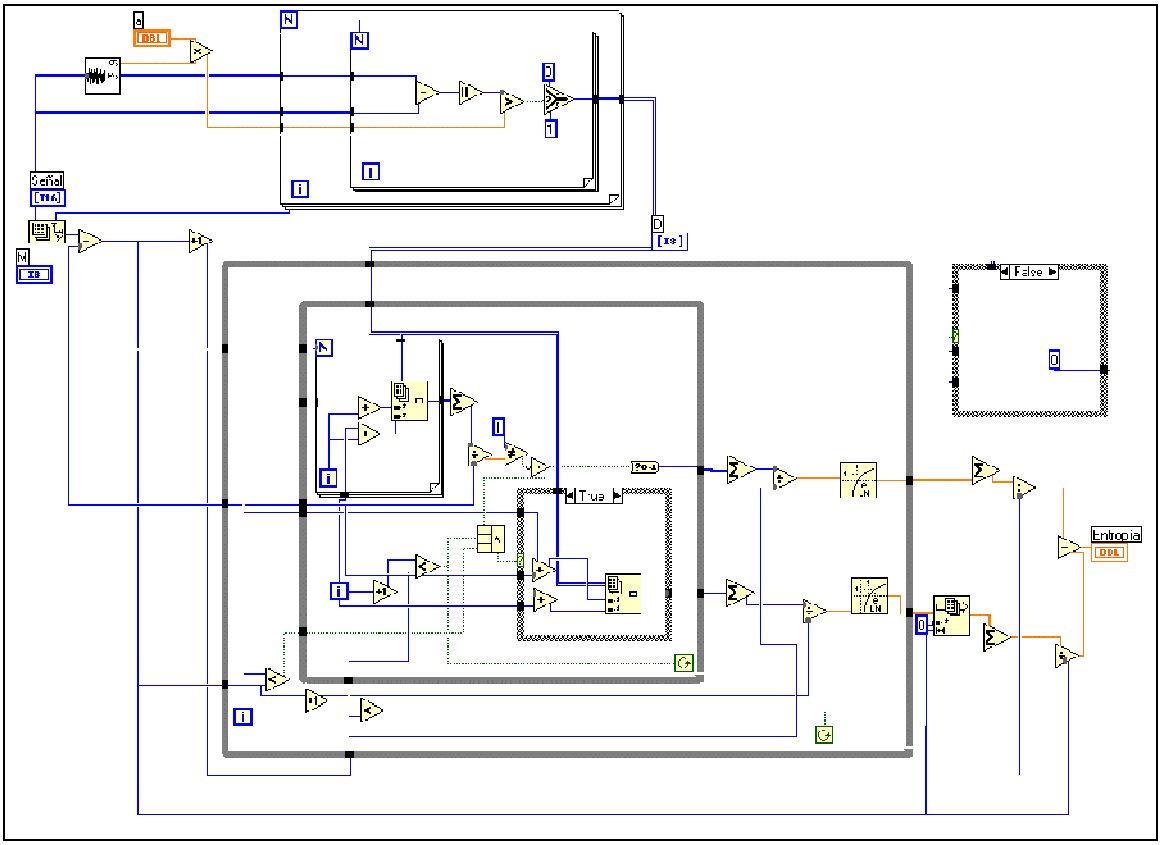


Figura 5.5: *Diagrama Entropía rápida.vi*

5.1.4. Algoritmo para el cálculo de la Dinámica Simbólica

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *Tamaño de ventana [Entero, 16 bits]* *A [Real, DBL]*	*Dinámica Simbólica [Real, DBL]*	

El algoritmo para el cálculo de la dinámica simbólica se basa en desplazar a lo largo de la señal una ventana. Para esa ventana calculamos su valor simbólico según lo expuesto en el apartado 2.1.3. Una vez calculado el valor de dinámica simbólica se desplaza la ventana un punto y se repite la operación. Esta operación se repetirá, dentro de un bucle, un número de veces dependiente del tamaño de la señal (N_s) y del ancho de la ventana (L) que se desplaza, concretamente N_s-L+1. El cálculo del valor simbólico consiste en calcular la diferencia entre los puntos sucesivos de la ventana y contabilizar el número de veces que esa diferencia (en valor absoluto) supera un umbral. De esta manera se obtiene información de la variabilidad del tramo de señal contenido en la ventana. El umbral lo fijaremos multiplicando la desviación típica por un factor a. En la siguiente figura vemos el diagrama correspondiente al programa:

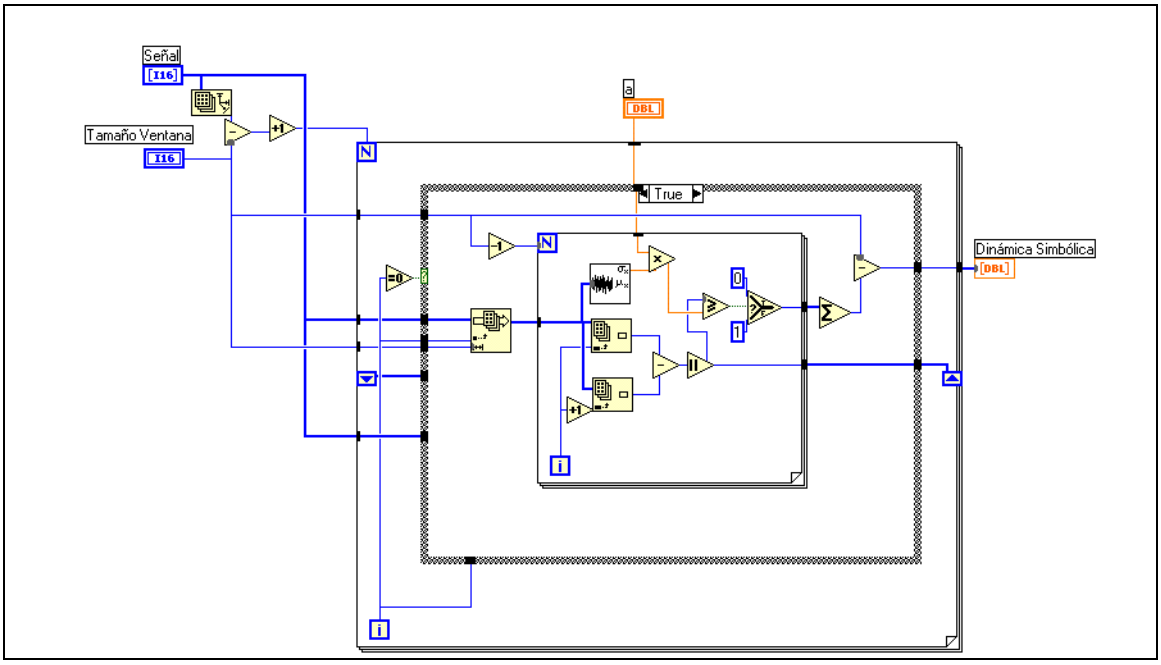


Figura 5.6: *Diagrama de Dinsin2.vi-Case True*

Como hemos dicho desplazamos la ventana punto a punto, es decir, los puntos contenidos en la ventana serán los mismos excepto el primero que ya no entra en la nueva ventana y el último que es el único nuevo. Debido a esto las diferencias calculadas en la iteración anterior son válidas en la nueva iteración con excepción de la primera que la desechamos y la última que es la única que debemos calcular. Así pues si tenemos esto en cuenta podemos obtener un considerable ahorro computacional. Creamos una estructura *Case* siendo la condición de selección el número de iteración. En la primera iteración si es necesario calcular todas las diferencias (*Case True*) es el caso que aparece reflejado en la figura []. A partir de la primera iteración (*Case False*) sólo habrá que calcular la última diferencia, luego el programa se completa con el siguiente diagrama:

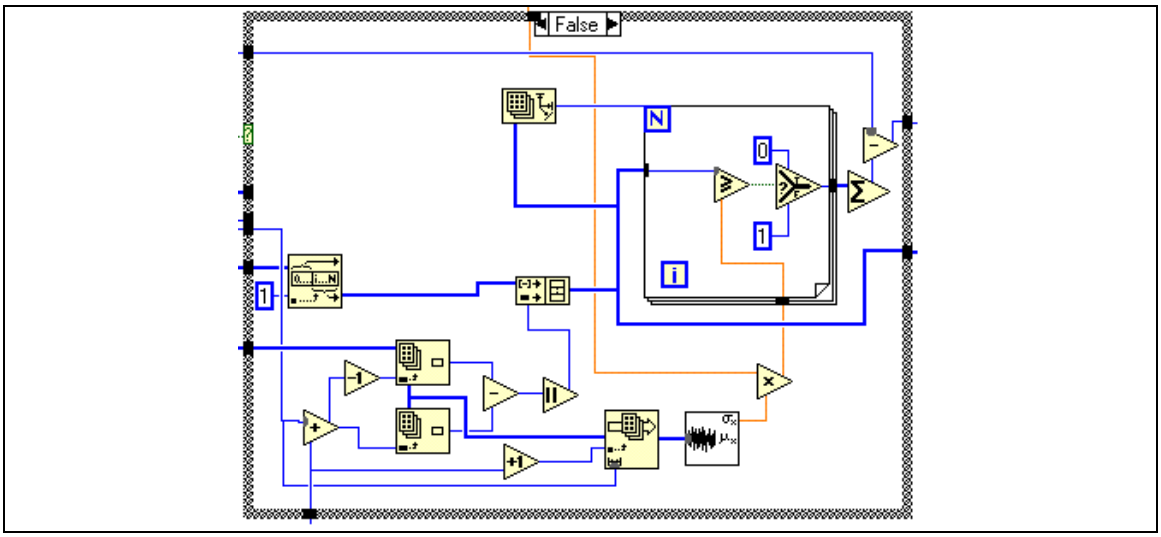


Figura 5.7: *Diagrama de Dinsin2.vi-Case False*

5.2. Algoritmo para la obtención de un modelo ARMA

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *Nºcoef AR [Real, DBL]* *Nºcoef MA [Real, DBL]*	*AR [Real, DBL]* *Ganancia LPC [Real, DBL]* *MA [Real, DBL]* *Ganancia MA [Real, DBL]*	

El algoritmo para el cálculo del modelo ARMA se basará en el Método de Shanks según lo expuesto en el apartado 2.2. Tal y como vemos en la siguiente figura, en una primera etapa calculamos los coeficientes AR o LPC mediante el método de Autocorrelación. Una vez calculados estos se calculan los coeficientes MA mediante el método propuesto por Shanks. No se pueden calcular los coeficientes en paralelo, ya que como vimos los coeficientes MA se calculan a partir de los AR.

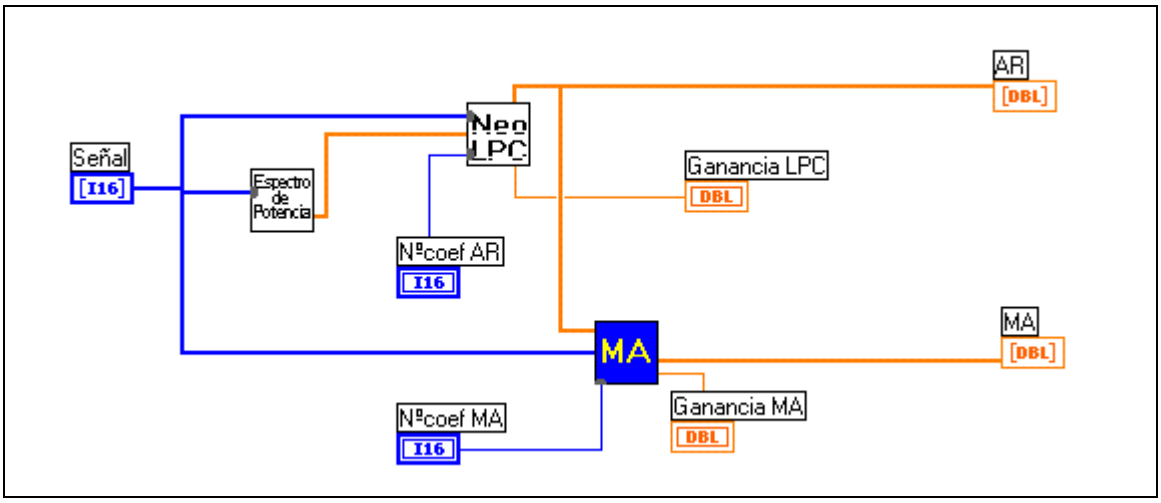


Figura 5.8: *Diagrama de AR+MA.vi*

5.2.1. Rutina de cálculo de LPC (AR). Método de autocorrelación

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *Espectro [Real, DBL]* *Número de polos [Real, DBL]*	*Coeficientes NeoLPC [Real, DBL]* *Ganancia [Real, DBL]*	

El algoritmo de cálculo de los coeficientes LPC (AR) utilizado en el desarrollo del Proyecto se basa en el método de mínimos cuadrados. Tal y como se describe en el apartado 2.2.1, minimizando el error cuadrático medio se obtiene un sistema de

ecuaciones normales que se presenta analíticamente en la expresión [2.28], este sistema proporciona el conjunto de parámetros del modelo LPC.

En la implementación del algoritmo, que se presenta en la figura 5.9, se observa cómo se construye la matriz de autocorrelaciones a partir de la fórmula 2.25 de la función de autocorrelación de la señal.

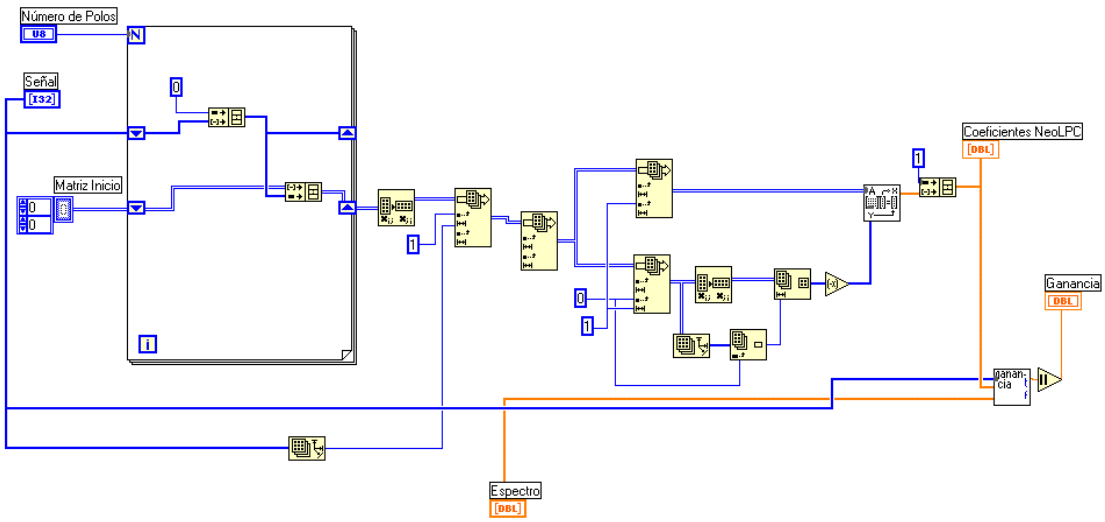


Figura 5.9: *Código de NeoLPC.vi*

La matriz de coeficientes del sistema y los términos independientes son las autocorrelaciones de la señal s_n. La primera columna de la matriz es el vector de términos independientes del sistema de ecuaciones normales, por ello se separa del resto y se cambia de signo. El primer coeficiente, a_0, correspondiente a la salida actual, es igual a 1 según se observa en la expresión 2.22, y se incluye en el vector formado tras el cálculo de los coeficientes a_k ($1 \leq k \leq p$).

5.2.1.1. Rutina de cálculo del Espectro de Potencia

Entradas	Salidas	Icono
Señal [Real, 16 DBL]	*Espectro de Potencia [Real, DBL]* *Espectro de Potencia, logaritmo [Real, DBL]*	

Esta función realiza el cálculo del espectro de potencia y del espectro de potencia logarítmico. Se observa en el código presentado en la siguiente figura que la función utilizada para calcular la transformada de Fourier es Real FFT.vi, que calculará la FFT de la “Señal” cuando el número de puntos sea potencia de 2 y la DFT cuando no lo sea. Tras el cálculo de la transformada de la señal real, se determina su módulo, se eleva al cuadrado y se prescinde de la parte derecha del espectro (recordemos que es simétrico).

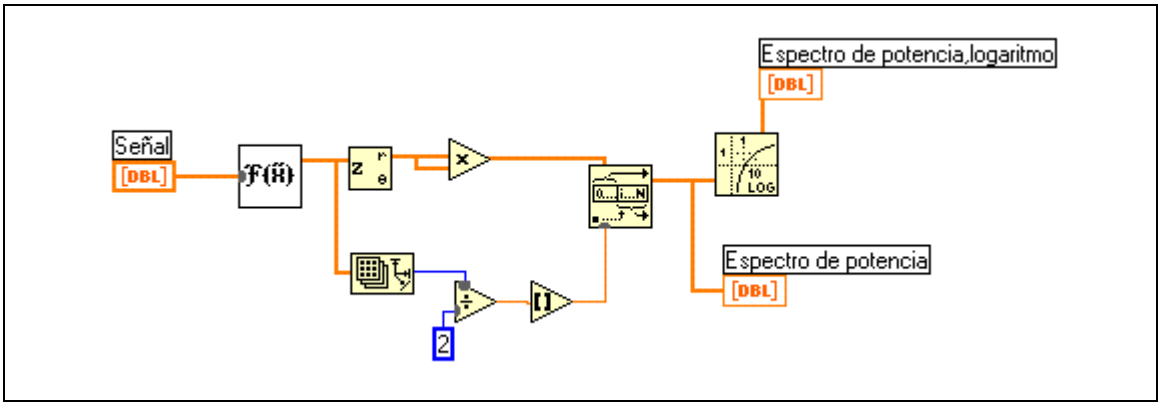


Figura 4.10: *Diagrama para el cálculo del Espectro de Potencia*

5.2.1.2. Rutina de cálculo del factor de ganancia LPC

Entradas	Salidas	Icono
Señal [Entero, 16 bits] *AR [Real, 16bits]* *Espectro [Real, 16 bits]*	*Ganancia (t) [Real, DBL]* *Ganancia (f) [Real, DBL]*	

Puede observarse en la figura 5.9 que el cálculo de la ganancia se realiza en un módulo separado, donde se calcula dicho parámetro del modelo mediante dos métodos alternativos: el primero en el dominio del tiempo, descrito en el apartado 2.2.1.3.

El código perteneciente al cálculo del factor de ganancia en el dominio del tiempo se presenta en la figura 5.11. Para calcular el factor de ganancia únicamente son necesarios los coeficientes de predicción AR y los datos temporales, es decir, las muestras de la señal.

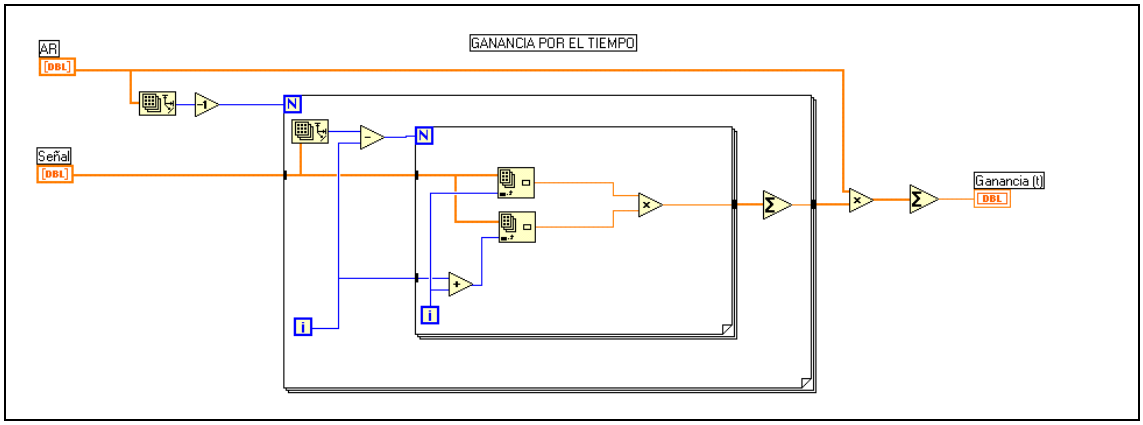


Figura 5.11: *Cálculo de la ganancia en el dominio del tiempo*

En el caso del cálculo en el dominio de la frecuencia, se precisan los coeficientes AR y el espectro de potencia de la señal, tal y como se muestra en la figura 5.12, que presenta el código correspondiente a dicho cálculo.

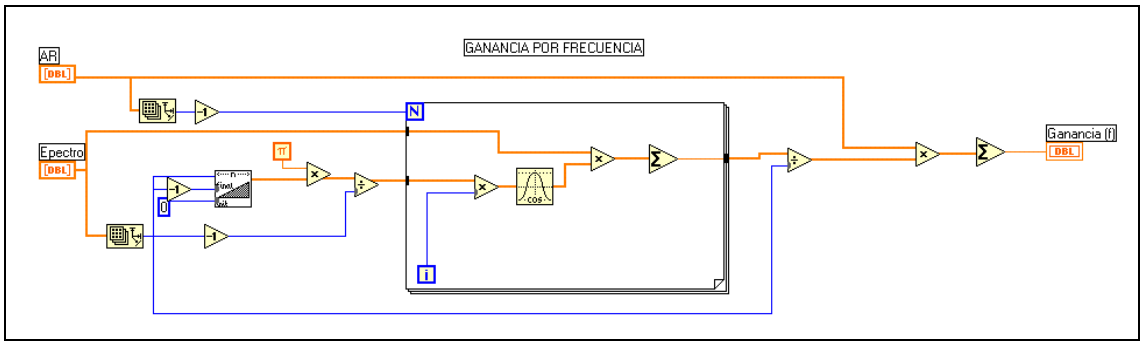


Figura 5.12: *Cálculo de la ganancia en el dominio de la frecuencia*

5.2.2. Algoritmo de cálculo de coeficientes MA. Método de Shanks.

.

Entradas	Salidas	Icono
AR [Real, DBL] *Señal [Entero, 16 bits]* *Nº coeficientes MA [Entero, 8 bits]*	*Ha [Real, DBL]* *MA [Real, DBL]* *Ganancia [Real, DBL]*	MA

El algoritmo de cálculo de los coeficientes MA utilizado en el desarrollo del Proyecto se basa en el Método de Skanks. Tal y como se describe en el apartado 2.2.2, a partir de los coeficientes AR calculados previamente, obtenemos un sistema parcial del cual obtenemos su respuesta ante una entrada impulso a partir de la expresión 2.51. Una vez tenemos la respuesta impulso creamos la matriz $H_a(z)$ y planteamos el sistema de ecuaciones 2.47. La resolución de este sistema de ecuaciones nos permite obtener el conjunto de coeficientes MA.

En la implementación del algoritmo, que se presenta en la siguiente figura, se observa cómo se construye en un bucle *For* la respuesta impulso del sistema a partir de los coeficientes AR. Una vez tenemos un vector con esta respuesta se construye la matriz $H_a(z)$ en otro bucle *For*. Solo quedaría resolver el sistema de ecuaciones y obtenemos los coeficientes MA. Por último para que el primero de los coeficientes sea 1 dividimos el vector de coeficientes por el valor de este. Este valor por el que dividimos pasará a formar parte de la ganancia del sistema

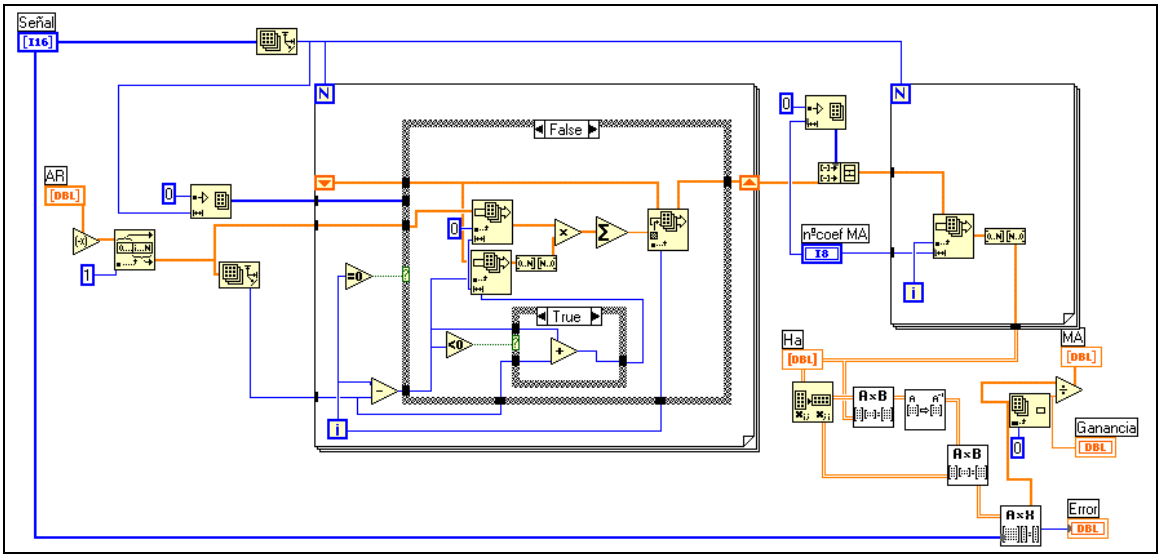


Figura 5.13: *Diagrama para el cálculo de los coeficientes MA*

5.2.3. Rutina para el cálculo de la envolvente ARMA

Entradas	Salidas	Icono
Coeficientes AR [Real, DBL] *Coeficientes MA [Real, DBL]* *Nº puntos del espectro [Entero, 32bits]* *Ganancia [Real, DBL]*	*Envolvente ARMA [Real, DBL]*	Envol ARMA

Implementando con LabVIEW la expresión 2.57 se obtendrá la aproximación al espectro resultante del modelo ARMA. Como vemos en la siguiente figura las respuestas en frecuencia correspondientes al numerador (parte todo-cero) y al denominador (parte todo-polo) de la envolvente (expresion 2.58) se calculan en un módulo aparte que presentaremos en el siguiente apartado.

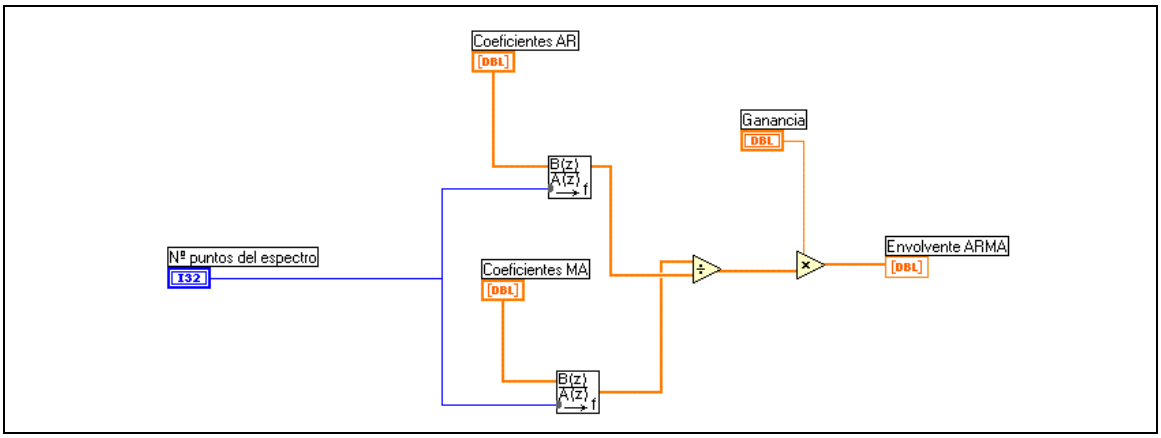


Figura 5.14: *Diagrama correspondiente a envolvente ARMA.vi*

5.2.3.1. Rutina para el cálculo de la respuesta en frecuencia de un polinomio

Entradas	Salidas	Icono
Nº puntos del espectro [Entero, 32bits] *Coeficientes [Real, DBL]*	*Componentes [Real, DBL]*	

Con este algoritmo calculamos la respuesta en frecuencia de un polinomio. Para ello, basta con aplicar los coeficientes del polinomio en la expresión 2.61.

Para recorrer el intervalo [0, π] sobre el cual se realiza el sumatorio, se crea un vector [0, *N*] donde *N* es el número de puntos del espectro y se multiplica por la constante π / N_S . Para obtener tanto el vector anterior como otro [0, *p*] donde *p* es el número total de coeficientes, se utiliza la función Ramp.vi.

Posteriormente, mediante un bucle, se recorren los intervalos calculando los elementos de los sumatorios de la expresión 2.61 para cada coeficiente y sumándolos, tras ello, se eleva al cuadrado y se suma.

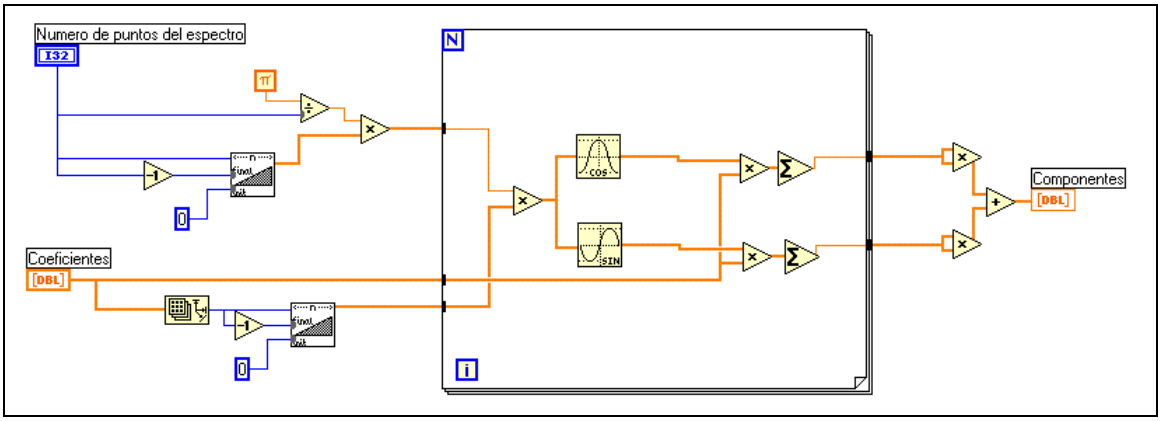


Figura 5.15: *Cálculo de la respuesta en frecuencia de un polinomio*

5.3. Algoritmo de cálculo de la distancia ARMA

Entradas	Salidas	Icono
AR_señal1 [Real, DBL] *MA_señal1 [Real, DBL]* *AR_señal2 [Real, DBL]* *MA_señal2 [Real, DBL]*	*N [Real, DBL]* *N' [Real, DBL]* *Distancia [Real, DBL]*	

En este algoritmo calcularemos la distancia existente entre dos modelos ARMA. Para ello implementaremos la métrica para modelos ARMA expuesta en el apartado 2.5. Como vemos en la siguiente figura este es un programa en el que simplemente ordenamos las etapas necesarias para poder aplicar la métrica.

Tendremos como entradas dos polinomios AR y dos polinomios MA, correspondientes a dos modelos ARMA distintos. El primer paso es asegurar que todos los polinomios tengan todas sus raíces dentro del círculo unidad, es decir, que sean estables. Esta condición la tenemos asegurada en los polinomios AR, ya que el método implementado aseguraba esta propiedad. Sin embargo el método de Shanks por el que obtenemos los coeficientes MA, no lo asegura. Los polinomios MA se introducen en la rutina *Refleja.vi* donde las raíces de los polinomios MA serán reflejadas según lo expuesto en el apartado 2.3.3 en caso de que sea necesario.

Una vez asegurada la estabilidad de los polinomios obtenemos los dos modelos AR estables a partir de los polinomios AR y MA de distintos modelos. Los modelos AR se calcularán en la rutina *MulPoli.vi* que más tarde explicaremos.

Finalmente en una nueva rutina llamada *Distancia.vi* cálculamos la distancia entre los dos modelos según la fórmula 2.77.

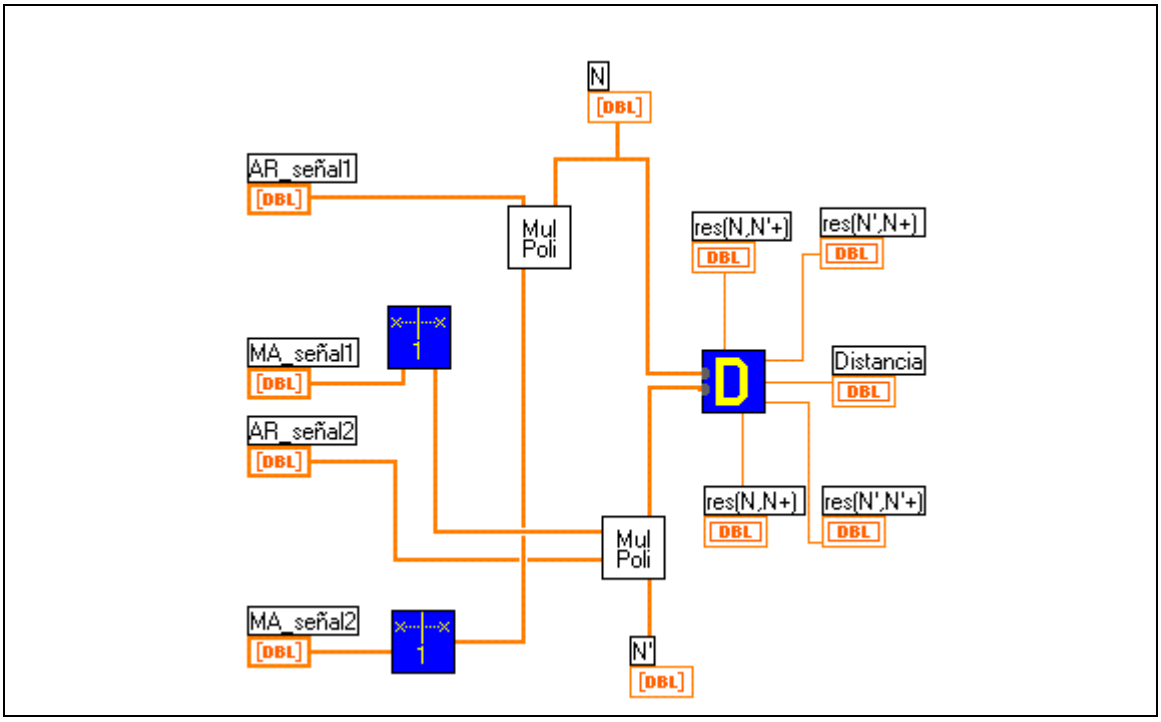


Figura 5.16: *Diagrama de Distancia Subvi.vi*

5.3.1. Rutina para la reflexión de raíces

Entradas	Salidas	Icono
Poli in [Real, DBL]	*Polos reflejados [Real, DBL]*	

Tal y como dijimos en el apartado anterior en esta rutina reflejamos las raíces de los polinomios MA, en caso de que sea necesario, según lo expuesto en el apartado 2.3.3.

El primer paso es calcular las raíces del polinomio. Una vez tenemos sus valores en un bucle *For* comprobamos si hay alguna cuya módulo sea mayor que uno (fuera del círculo unidad). Este algoritmo está previsto utilizarlo para polinomios de grado dos, luego tendremos cuatro posibilidades: ninguna raíz fuera del círculo unidad, la primera raíz fuera, la segunda raíz fuera y las dos raíces fuera. Cada una de esta posibilidades habilitará una de las opciones de una estructura *Case*, donde realizaremos (o no) los cambios de coeficientes correspondientes según las expresiones 2.54 y 2.56.

En la siguiente figura vemos el diagrama donde se implementa lo explicado anteriormente. En ella aparecerá en la estructura *Case* la opción correspondiente a dos raíces fuera del circulo unidad. El resto de ventanas del *Case* lo tenemos en la figura 4.18.

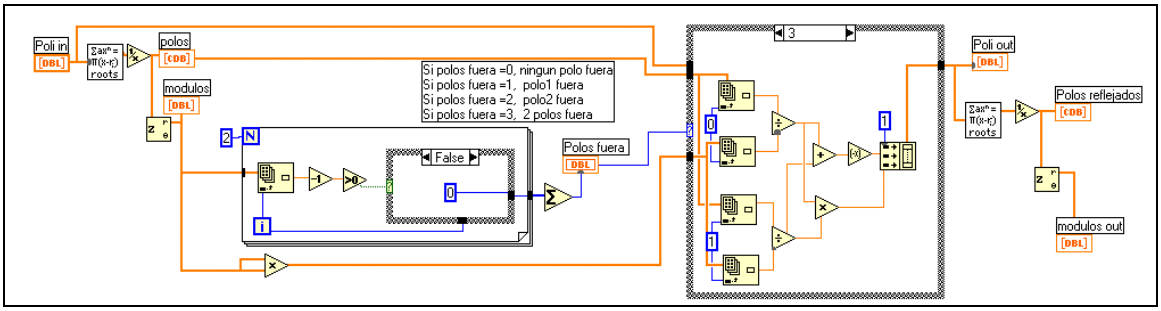


Figura 5.17: *Diagrama de Refleja .vi*

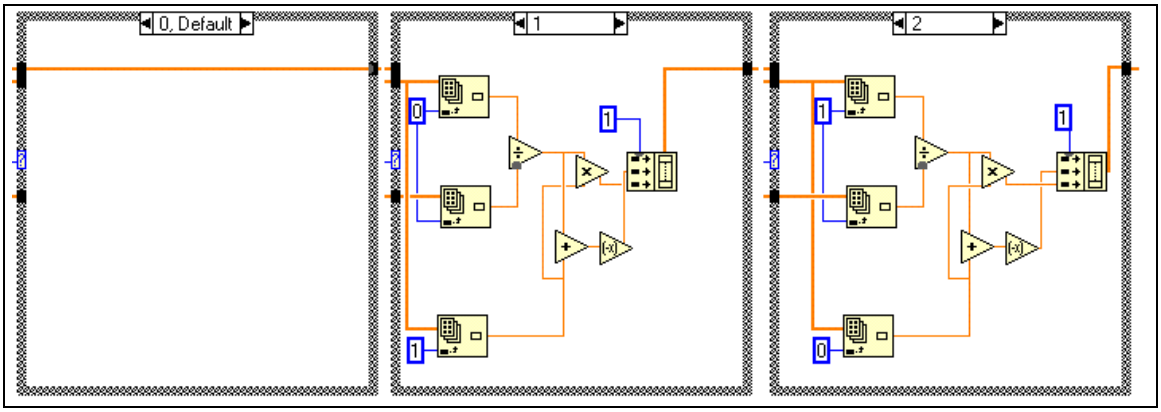


Figura 5.18: *Ventanas 0,1 y 2 de la estructura Case de Refleja.vi*

5.3.2. Rutina para la multiplicación de polinomios

Entradas	Salidas	Icono
Poli 1 [Real, DBL] *Poli 2 [Real, DBL]*	*Poli1 x Poli2 [Real, DBL]*	

En este algoritmo obtenemos un polinomio como resultado de multiplicar otros dos que serán nuestras entradas Para ello, en un bucle *For* multiplicamos uno de los polinomios por cada uno de los coeficientes del otro, desplazando el resultado de cada multiplicación parcial tantas posiciones a la izquierda como sea el número de la iteración. Finalmente obtenemos una pila de resultados parciales, la cual sumamos columna a columna en otro bucle *For*, obteniéndose el polinomio final. En la siguiente figura vemos el código correspondiente a este algoritmo.

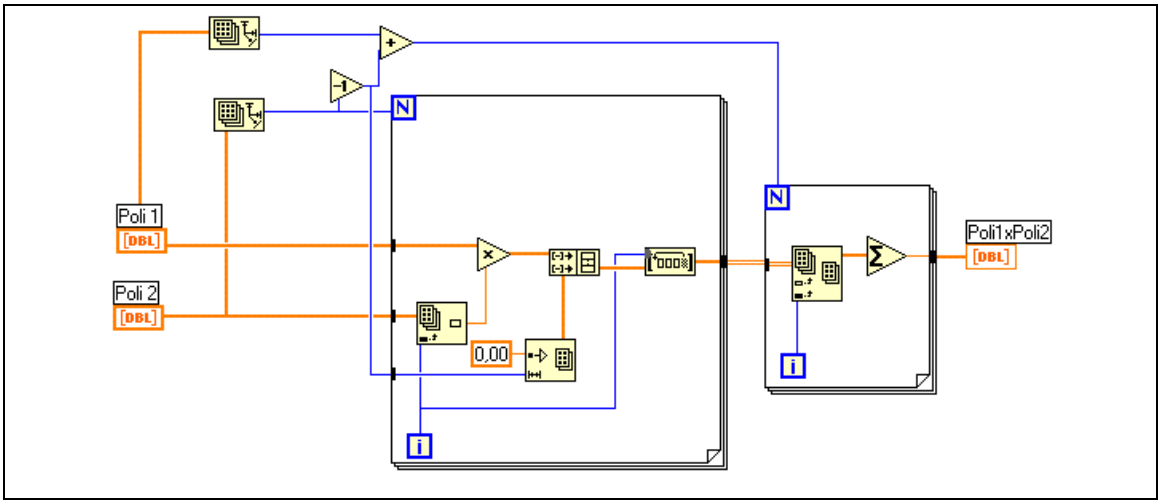


Figura 5.19: *Diagrama de MulPoli.vi*

5.3.3. Rutina de cálculo de distancia entre dos modelos AR

Entradas	Salidas	Icono
A [Real, DBL] *A' [Real, DBL]*	*Distancia [Real, DBL]*	

Este algoritmo será el encargado de obtener la distancia entre lo dos modelos. Para ello solamente hemos de implementar la ecuación 2.77. Para el cálculo de las resultantes hemos desarrollado la rutina *Resultante.vi* que explicaremos en el apartado siguiente. En la figura 5.20 tenemos el código correspondiente de esta rutina.

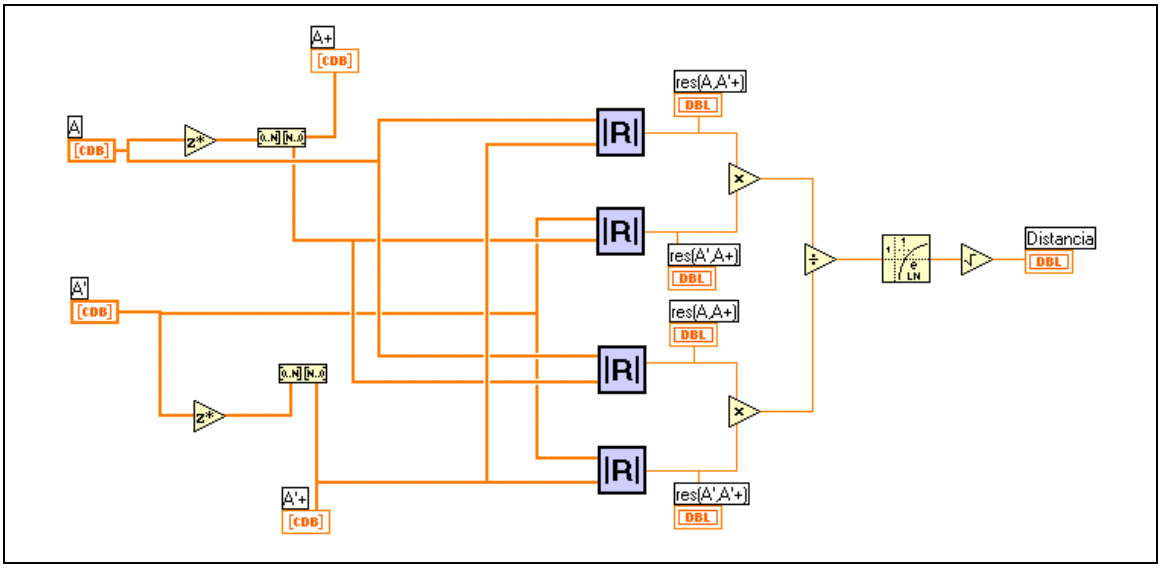


Figura 5.20: *Diagrama de Distancia.vi*

5.3.3.1. Rutina de cálculo de la resultante entre dos polinomios

Entradas	Salidas	Icono
A [Real, DBL] *A' [Real, DBL]*	*Determinante [Real, DBL]* *Resultante [Real, DBL]*	

Este algoritmo será el encargado de obtener la resultante entre dos polinomios. Para ello solamente hemos de implementar la ecuación 2.81. Como vemos el determinante puede dividirse en dos partes. En cada una de estas partes un mismo polinomio se desplaza en cada fila una posición a la derecha, rellenándose los huecos con ceros. Cada submatriz tendrá un número de filas igual al número de polos. Estas submatrices las creamos en sendos bucles *For*. Una vez obtenidas las dos submatrices las juntamos y creamos la matriz global y calculamos el determinante de la misma. En la siguiente figura tenemos el código de esta rutina.

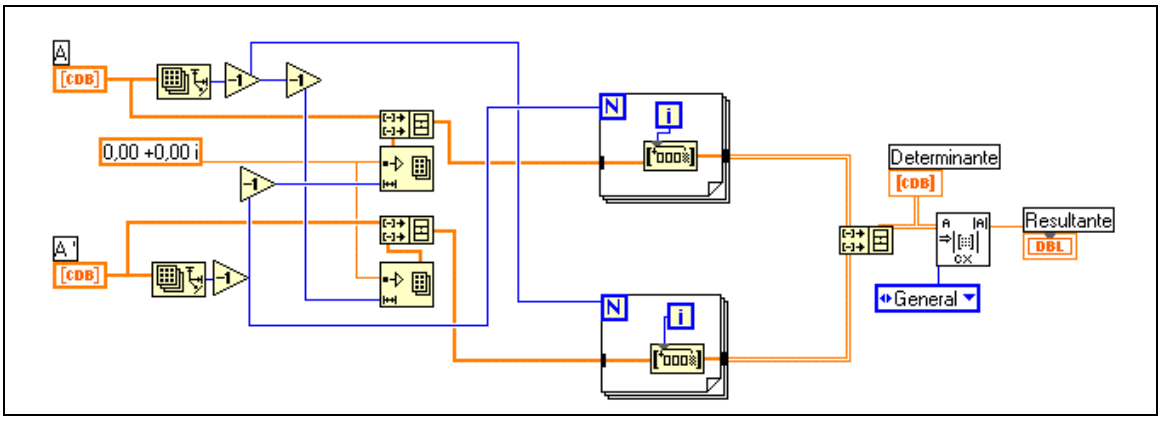


Figura 5.21: *Diagrama de Resultante.vi*

5.4. Algoritmos para el reconocimiento de locutores

Dentro de este apartado presentamos una tres aplicaciones desarrolladas para el reconocimiento de locutores. Tendremos una aplicación para la identificación de locutores y otra para la verificación.

5.4.1. Rutina para la creación de pilas de datos.

Con esta aplicación lo que hacemos es obtener un modelo de la señal de entrada y guardarlo dentro de un fichero de datos donde se encontrarán todos los modelos de las señales que intervendrán en el proceso de reconocimiento de locutores. A continuación explicamos paso por paso el programa:

- Tomamos la señal del fichero donde se encuentre almacenada.

- La señal tendrá una longitud de 2560 puntos. Para que el modelo sea más preciso dividimos la señal en 10 segmentos de 256 puntos y creamos el modelo ARMA(12,2) de cada uno de ellos. Para realizar esta segmentación nos valemos de un bucle *For* , dentro del cual el modelo de cada segmento se ñade a la cola de un vector que será el modelo final (quitamos de los modelos los coeficientes a_0 y b_0 iguales a 1).

- El modelo final de la señal estará formado por un vector que tendrá 140 coeficientes. Sólo falta añadir en la primera posición del vector un número de cabecera que indicará el locutor al que pertenece la señal. Esta cabecera, nos servirá para poder crear los modelos de cada locutor a partir de todas aquellas señales que pertenecen a él. Además gracias a ella, podremos saber si la clasificación realizada es correcta o no.

- Finalmente el modelo de esta señal (vector de 1+140 coeficientes) es almacenado en un fichero TXT el cual contendrá todos los modelos de todas las señales de todos los locutores. Cada uno de estos ficheros se creará para un determinado tipo de sonido, por ejemplo, *pila de datos _oma.txt*.

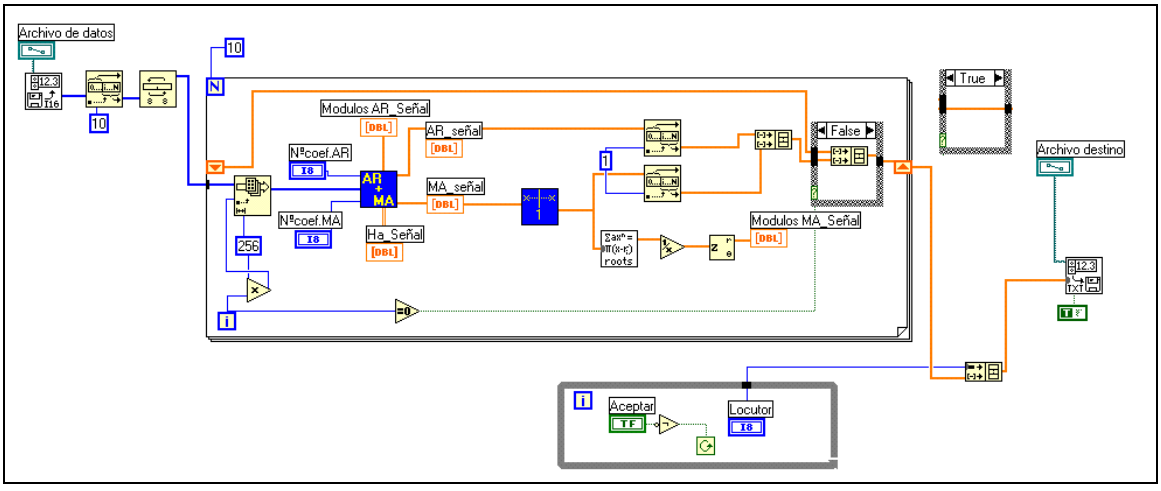


***Figura 5.22**: Diagrama de Acumulador.vi*

5.4.2. Algoritmo para la identificación de locutores

El objetivo de este algoritmo es clasificar el vector de coeficientes que hemos obtenido a partir de una señal como perteneciente al locutor que la ha generado. Para ello compararemos cada uno de los vectores con cada uno de los modelos generados para cada locutor, y será clasificado como perteneciente a aquel cuya distancia sea menor. A continuación explicamos el algoritmo generado por partes:

Las muestras son extraídas de un fichero que contendrá una matriz de datos. Cada fila de esta matriz corresponderá a una muestra (modelo) obtenida a partir de una señal con el *Acumulador.vi* . El primer paso es colocar una cabecera que indique su posición de entrada, esta nos servirá para ordenarlos por locutores. A continuación pasan a la rutina *Ordena Vectores.vi* donde son ordenados por locutores y orden de entrada, es decir, todas las muestras del locutor 1 son situadas en filas consecutivas, y así para todos lo locutores. Una vez ordenadas las muestras y que conocemos el número de locutores pasamos a calcular las distancias entre las muestras y los modelos de cada locutor. Esto lo hacemos en un bucle *For* que se repetirá tantas veces como muestras tengamos. Al igual que ocurría al crear el modelo de la señal calcularemos 10 distancias correspondientes a cada uno de los segmentos en los ue dividíamos la señal. Los modelos de cada locutor se obtienen como la media de todas las muestras pertenecientes a un mismo locutor. Una vez que tenemos la muestra y el modelo calculamos las distancias. Para cada muestra tendremos una matriz de distancias con un número de filas igual al número de locutores y diez columnas correspondientes a cada uno de los segmentos. El código del programa correspondiente a lo explicado hasta el momento lo tenemos en la siguiente figura:

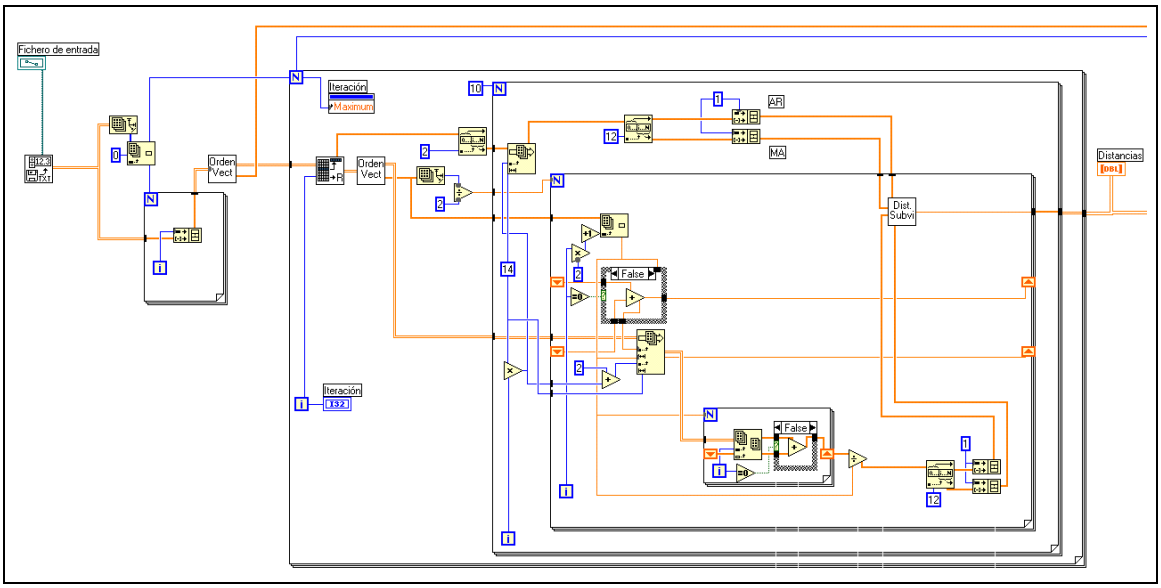


Figura 5.23: *Etapa de cálculo de distancias por segmentos en Identificación 10.vi*

El siguiente paso es combinar las distancias parciales de cada segmento en una distancia global entre muestra y modelo, quedando la matriz de distancias de cada muestra reducida a un vector de distancias de tamaño igual al número de locutores. Para ello hemos computado tres tipos de distancias:

- La distancia final será la suma de distancias parciales.
- La distancia final será la suma del cuadrado de distancias parciales.
- La distancia final será la mayor de las distancias parciales.

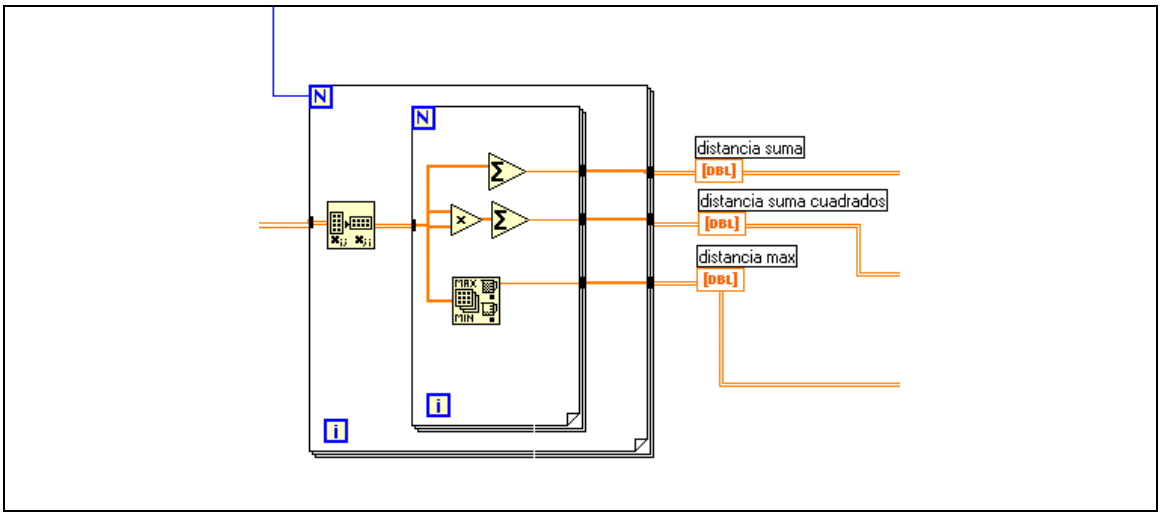


Figura 5.24: *Etapa de cálculo de distancias finales en Identificación 10.vi*

Finalmente llegamos a la etapa de clasificación. En esta etapa para cada muestra vemos en que posición del vector de distancias calculado en la etapa previa se encuentra el mínimo. La muestra queda clasificada como perteneciente a ese locutor. Para comprobar si la clasificación es correcta comparamos el resultado con la cabecera de locutor que llevaba la muestra en un principio. Si coinciden tendremos un acierto, si no acumularemos un error. Finalmente como salida del algoritmo además de las distancias entre muestras y modelos de locutores tendremos el número de errores cometidos y el índice de aciertos. Todo lo comentado en esta etapa se realizará para los tres tipos de distancias desarrolladas. El código del programa correspondiente a esta última etapa lo tenemos en la siguiente figura:

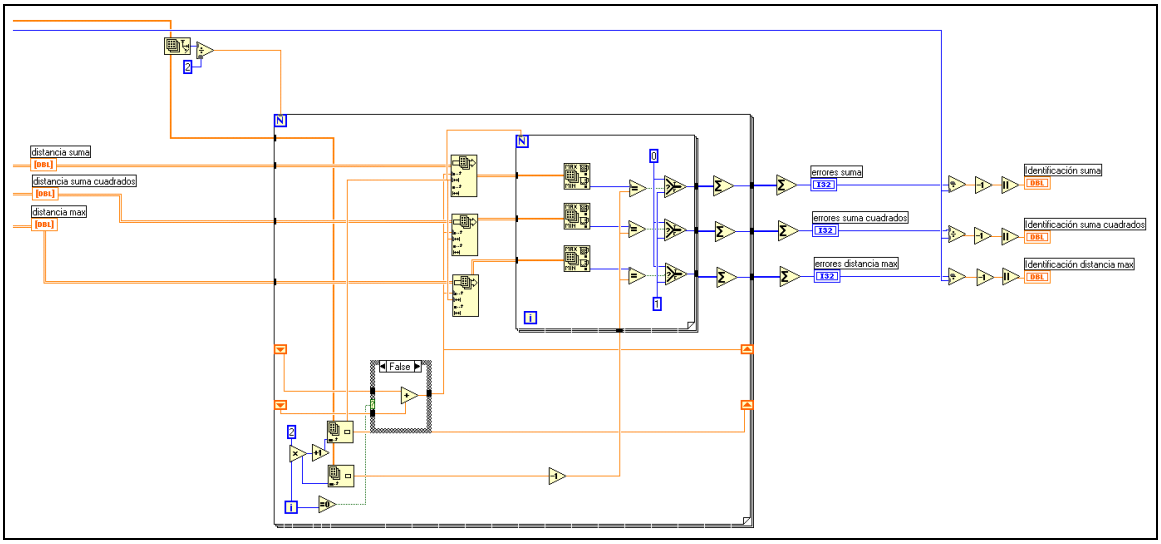


Figura 5.25: *Etapa de clasificación en Identificación 10.vi*

5.4.3. Algoritmo para la verificación de locutores

El objetivo de este algoritmo es obtener una serie de estadísticos (sensibilidad, especificidad, valor predictivo positivo y valor predictivo negativo) que me indican la bondad del método de clasificación que empleamos. Para ello compararemos cada muestra con dos modelos, uno correspondiente a una población (calculado a partir de las muestras pertenecientes a un mismo locutor) y otro correspondiente al resto de las poblaciones (calculado a partir del resto de muestras, es decir, todas las muestras menos las usadas para calcular el modelo de la población con la que ensayemos). Si la distancia es menor respecto al modelo del locutor la muestra pertenecerá a ese locutor y si por el contrario la distancia es menor respecto al segundo modelo, la muestra no pertenecerá al locutor ensayado. Este proceso en el que no clasificamos una muestra si no que decimos simplemente si pertenece a una población o no, es lo que llamábamos en el capítulo 3 verificación.

Dividimos el algoritmo por etapas:

- En una primera etapa las muestras son tomadas de un fichero, y tras introducirles una cabecera con su orden de entrada son ordenadas por poblaciones en la rutina *Ordena Vectores.vi* que en el siguiente apartado explicaremos.

- En segundo lugar se extraerá por un lado las muestras correspondientes a un mismo locutor y por otro lado el resto de muestras. Esto lo haremos mediante la rutina *Separa Submatriz.vi*. A partir de estas dos matrices de datos creamos los modelos realizando una media entre coeficientes con la rutina *Media+Desv_M2.vi*. Ambas rutinas se comentarán en el siguiente apartado.

- En tercer lugar pasamos a realizar la verificación según explicábamos anteriormente. Tomaremos una muestra y la compararemos (mediante el cálculo de la distancia entre modelo) con los dos modelos calculados en el paso previo. La distancia entre modelos se realizará como vimos en el apartado anterior, es decir, se calcularán diez distancias parciales (mediante un bucle *For*) correspondientes a cada uno de los segmentos en los que dividíamos la señal y una total que será la suma de las parciales. Las distancias se calcularán con la rutina *Distancia Subvi.vi*, que ya vimos en el apartado 5.3. En el caso de que la distancia entre la muestra y el modelo del locutor sea menor o igual que respecto al modelo del resto de muestras asignaremos a la comparación un 1 (acierto) y en caso contrario un 0 (rechazo).

- Los pasos dos y tres se realizarán con todas las poblaciones existentes con la ayuda de un bucle *For* que se repetirá tantas veces como poblaciones tengamos. Finalmente para cada muestra tendremos un vector de 1 y 0 de tamaño igual al número de locutores. Es de desear que en el vector solo existiera un 1 en la posición correspondiente a la población a la que realmente pertenece la muestra y 0 en el resto de posiciones.

- Todos estos pasos se repiten para todas las muestras obteniéndose finalmente una matriz de 1 y 0 con un número de filas igual al número de locutores y un número de columnas igual al número de muestras.

El código correspondiente a estos pasos lo tenemos en la figura 5.26.

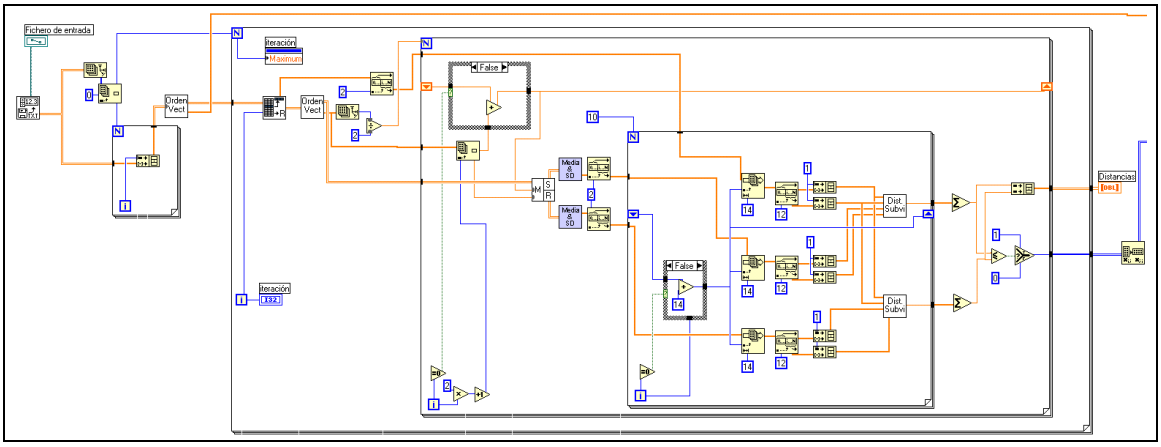


Figura 5.26: *Etapas de cálculo de distancias y comparación en Verificación 10.vi*

En la siguiente etapa del algoritmo será donde obtendremos una serie estadísticos que nos permitirán cuantificar los resultados. Todos ellos se calcularán a partir de combinar los verdaderos positivos (TP, 1 en su lugar correcto), falsos negativos (FN, 0 en lugar incorrecto), verdaderos negativos (TN, 0 en lugar correcto) y falsos positivos (FP, 1 en lugar incorrecto). En nuestro caso al tener la matriz de resultados ordenada por poblaciones será fácil. Tomamos la submatriz formada por aquellas columnas pertenecientes a una misma población y la dividimos en dos partes. Por un lado tomamos la fila correspondiente a la población a la que pertenecen las muestras. Todos los 1 que encontremos serán TP y los 0 son FN. Por otro lado, los 0 que encontremos en el resto de submatriz serán TN y los 1 FP. Combinando los resultados obtenidos calculamos la sensibilidad, especificidad, valor predictivo positivo y valor predictivo negativo. El código correspondiente a este paso lo tenemos en la siguiente figura.

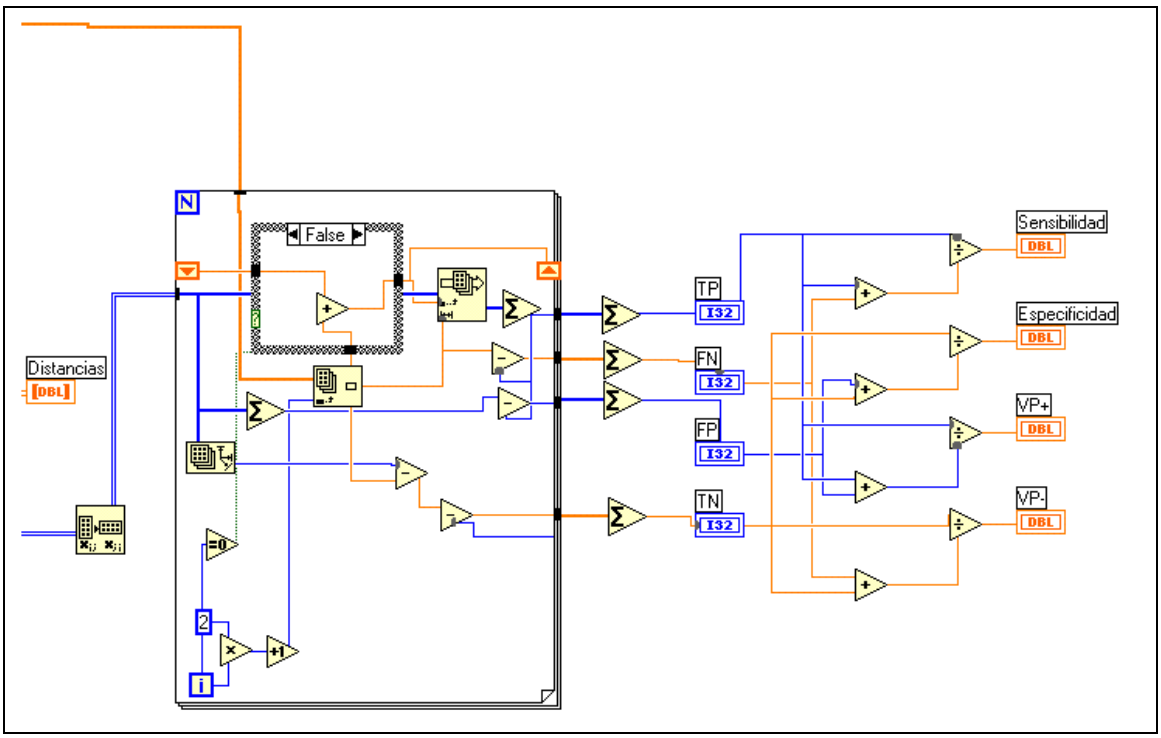


Figura 5.27: *Etapa de clasificación en Verificación10.vi*

5.4.4. Rutinas utilizadas en el reconocimiento de locutores

Tal y como hemos visto en los apartados anteriores los algoritmos de reconocimiento llamaban a una serie de rutinas para su funcionamiento. En el presente apartado las presentamos.

5.4.4.1. Rutina para ordenar vectores

Entradas	Salidas	Icono
Entrada vectores [Real,DBL]	*Vectores Ordenados [Real,DBL]* *Locutores / cantidad [Real,DBL]*	

Con este algoritmo ordeno la matriz de muestras por locutores. La salida de este algoritmo será por tanto la matriz de datos ordenada, pudiendo dividirla por así decirlo en pequeñas submatrices dentro de las cuales sólo existirían muestras pertenecientes a un mismo locutor.

El primer paso es detectar el número de locutores que existen en la matriz de entrada. Para ello examino la cabecera de cada muestra. En ella el primer coeficiente me dirá su número de fila y el segundo el locutor al que pertenece. Examinando estos dos coeficientes y con ayuda de un bucle *For*, creo una matriz que contendrá tantas filas como locutores y en cada una de las filas cuales de las muestras de entrada pertenecen a ese locutor. Finalmente a partir de esta matriz se ordenarán las muestras de entrada en pequeñas submatrices de muestras pertenecientes a un mismo locutor y todas juntas formarán la salida del programa (Vectores Ordenados). La rutina tendrá otra salida en la que alternativamente se indicará locutor y número de muestras que le pertenecen. Esta salida será de utilidad en los algoritmos de reconocimiento para poder extraer las submatrices y crear los modelos de cada locutor. En la siguiente figura tenemos el código de esta rutina:

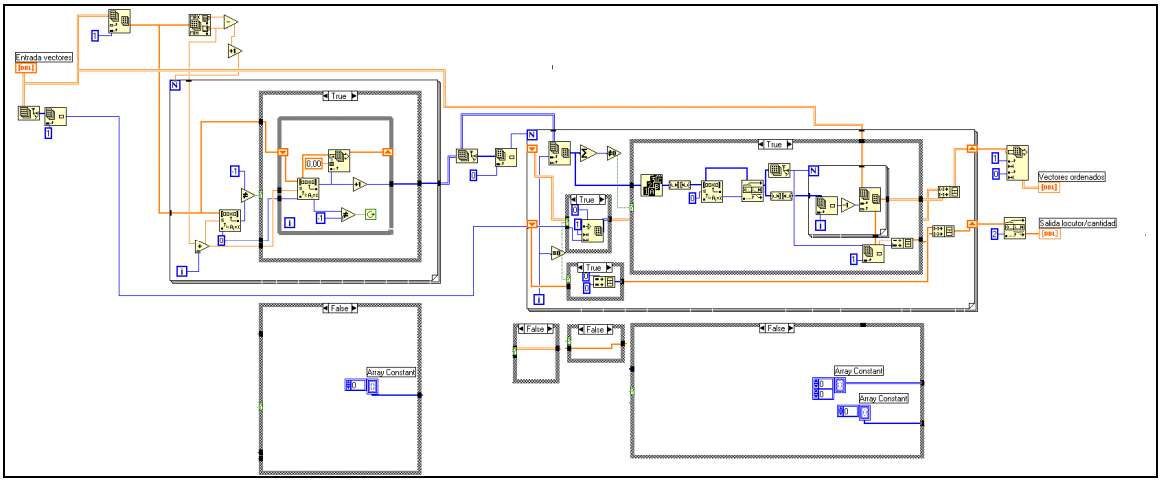


Figura 5.28: *Diagrama de Ordena vectores subvi.vi*

5.4.4.2. Rutina para el cálculo de media y desviación típica

Entradas	Salidas	Icono
Entrada [Real,DBL]	*Desviación Típica [Real,DBL]* *Media [Real,DBL]*	

Esta rutina la aplicaremos para obtener dos modelos a partir de una matriz de datos de entrada. Un modelo será la media de los datos de entrada y el otro la desviación típica. Cada columna de la matriz dará lugar a un coeficiente de cada uno de los modelos. Para el cálculo de estos se usa el módulo *Sample Variance.vi* de las librerías de LabVIEW. El código de esta rutina lo tenemos en la siguiente figura:

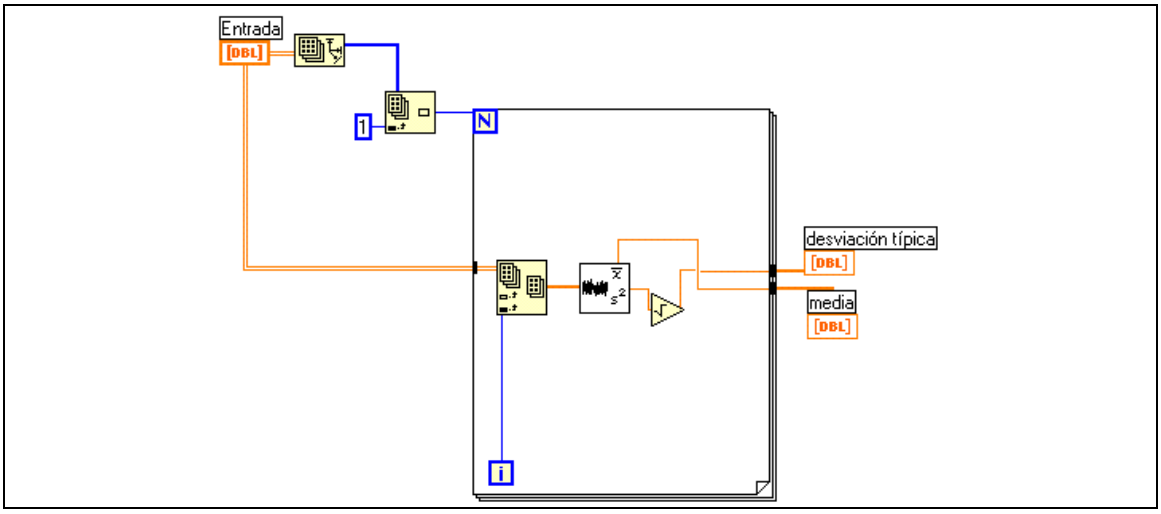


Figura 5.29: *Diagrama de Media+Desv_M2.vi*

5.4.4.3. Rutina para la extracción de filas

Entradas	Salidas	Icono
Matriz [Real,DBL] *Fila [Entero, 32 bits]*	*Matriz sin fila [Real,DBL]* *Fila [Real,DBL]* *Error [Cadena de texto]*	

El objetivo de esta rutina es que a partir de una matriz de datos obtener dos salidas: Por un lado una fila especificada y por otra parte el resto de la matriz de datos. El primer paso es comprobar que el número de fila solicitada sea positivo y que no sea mayor que la dimensión de la matriz. Si no se cumple alguna de estas condiciones se dará un mensaje de error. Si se cumplen extraeremos de la matriz por un lado la fila solicitada y por otra las submatrices que quedan por encima y por debajo de esta fila, juntándolas posteriormente. El código de esta rutina lo tenemos en la siguiente figura:

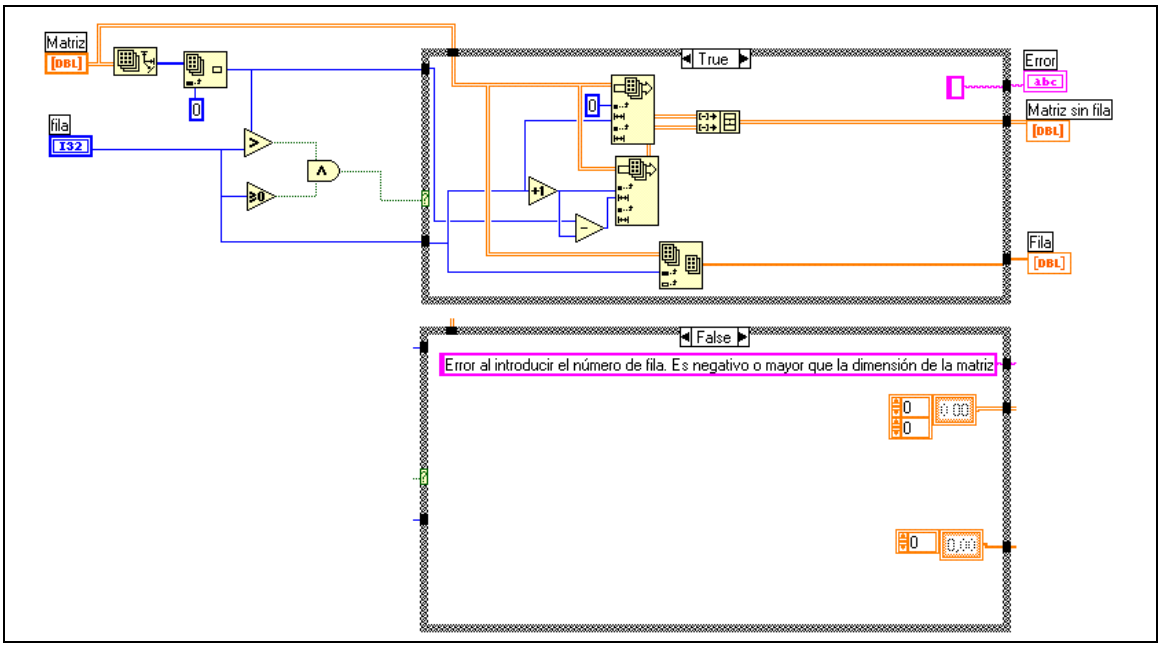


Figura 5.30: *Diagrama de Separa_fila+resto.vi*

5.4.4.4. Rutina para la extracción de submatrices

Entradas	Salidas	Icono
Matriz de entrada [Real,DBL] *Ancho [Entero, 32 bits]* *Index [Entero, 32 bits]*	*Submatriz [Real,DBL]* *Resto [Real,DBL]*	

El objetivo de esta rutina es que a partir de una matriz de datos obtener dos salidas: Por un lado una submatriz (especificada por un índice y un ancho) y por otra parte el resto de la matriz de datos. En un primer paso se extrae la submatriz de la matriz de datos de netrada con el módulo *Array Subset* de las librerías de LabVIEW. Por otra lado, extraemos las submatrices que quedan por encima y por debajo de esta fila, juntándolas posteriormente. El código de esta rutina lo tenemos en la siguiente figura:

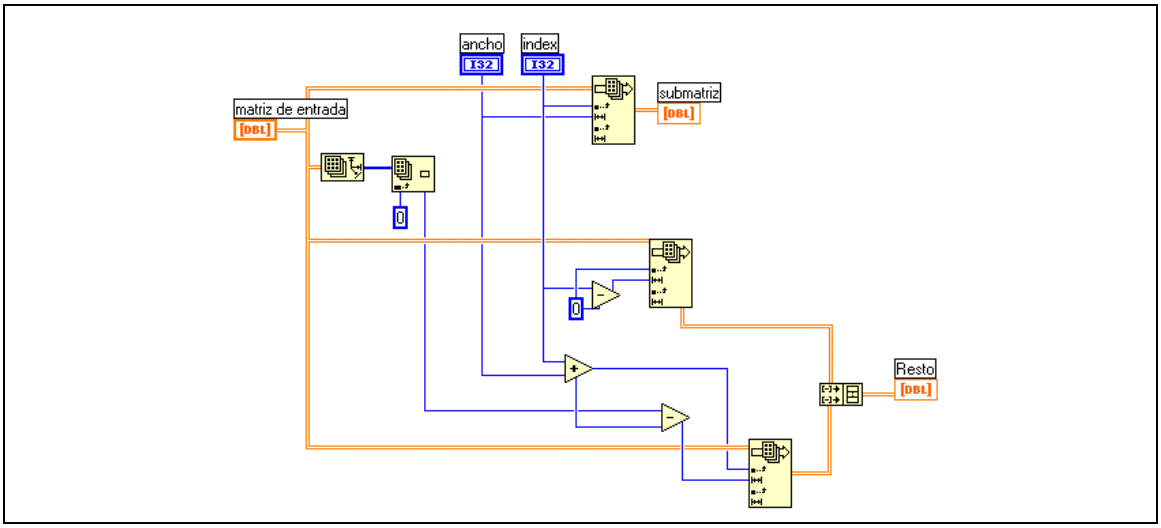


Figura 5.31: *Diagrama de Separa Submatriz.vi*

Capítulo 6: Resultados

Dinámica no lineal
Modelo ARMA. Número óptimo de polos
Reconocimiento de locutores

6. Resultados

Tal y como hemos visto en el capítulo anterior para la aplicación de los distintos algoritmos es necesario ajustar una serie de parámetros de entrada. En el presente capítulo veremos cómo se han obtenido los valores óptimos de estos parámetros para que los algoritmos tengan un mejor comportamiento.

6.1. Algoritmos de la Dinámica no lineal

Para el ajuste de los parámetros de entrada en los métodos de la dinámica no lineal, cuya finalidad era caracterizar transiciones, ejecutaremos los algoritmos sobre dos tipos situaciones muy frecuentes en el habla natural. Estas señales que nos servirán de estándar son: una transición consonante nasal – vocal (concretamente /m~a/, figura 6.1), donde tenemos coarticulación nasal y una transición consonante fricativa – vocal (concretamente /s~i/, figura 6.2).

Utilizaremos estas dos señales ya que como veremos al caracterizar la evolución de las transiciones, en la señal de la figura 6.1, ésta se produce de manera más sostenida que la correspondiente a la de la figura 6.2 que es muy corta.

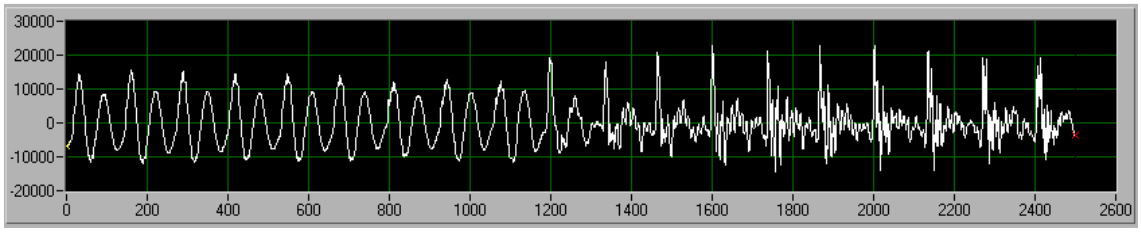


Figura 6.1: *Señal correspondiente al registro /ma/*

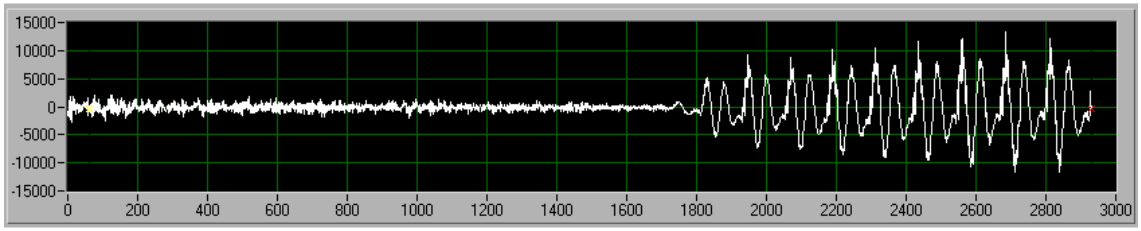


Figura 6.1: *Señal correspondiente al registro /si/*

6.1.1. Entropía Aproximada

Para caracterizar correctamente una señal mediante la evolución de la Entropía Aproximada es necesario la estimación de los valores óptimos de algunos parámetros. Estos son:

- El número de valores previos para la predicción del valor subsiguiente, m.
- Factor de modulación (α) del umbral, ε.
- La longitud de la ventana usada, *N*.

En primer lugar vamos a estudiar como afecta la variación de α a la entropía aproximada. Para ello aplicamos el algoritmo con tres valores de α distintos (*0.1, 0.3* y *0.5*) a la señal que apare representada en la figura 6.1, manteniendo los valores *m=1* y *N=256*.

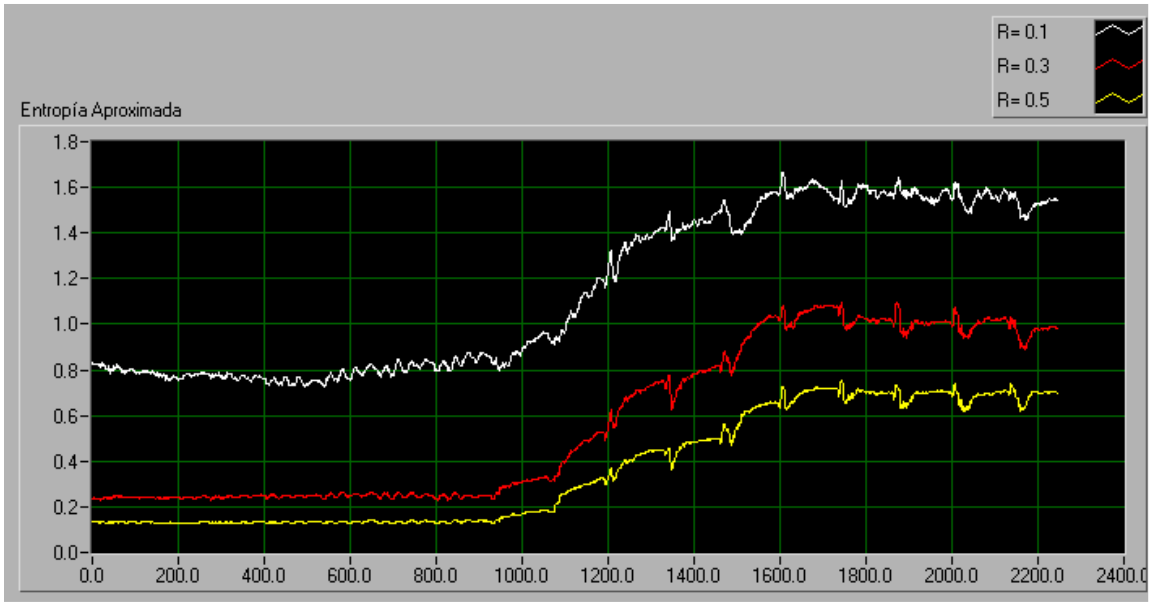


Figura 6.3: *Evolución de la Entropía Aproximada a lo largo del registro /ma/ para distintos valores de* α

En la figura 6.3, podemos ver que la señal queda perfectamente caracterizada por cualquiera de las tres gráficas, si bien observándola con atención vemos que la diferencia entre los valores que adopta la entropía (amplitud) entre las dos zonas estables es mayor para α*=0.1* y *0.3* que para *0.5*. Además se puede observar que existe menos rizado en las zonas correspondientes a sonidos puros para α*=0.3* y *0.5* que para *0.1*. Luego todo parece indicar que la mejor opción es tomar α*=0.3*

Si repetimos el experimento para la señal de la figura 6.2 llegaremos a la misma conclusión, siendo en este caso más claro el mayor salto entre sonidos puros para α*=0.3.*

Figura 6.4: *Evolución de la Entropía Aproximada a lo largo del registro /si/ para distintos valores de* α

A continuación, estudiaremos como responde la entropía ante la variación del tamaño del vector de comparación (m). Para ello aplicamos el algoritmo con tres valores de *m* distintos (1, 2 y 3) a la señal que aparece representada en la figura 6.1, manteniendo los valores α*=0.3* y *N=256*.

Como vemos en la figura 6.5, aplicando cualquiera de los tres valores, la señal quedaría correctamente caracterizada. El valor de la entropía permanece dentro de un mismo intervalo (existe un cierto rizado) en los tramos correspondientes a sonido puro. Por el contrario, la transición transcurre manteniendo una pendiente estable. Sin embargo, para un valor *m=1* el salto entre valores estables de entropía es mayor. Esta característica es importante si aplicamos este algoritmo como una fase previa en un algoritmo de segmentación, ya que los tramos estables quedan más diferenciados y su detección sería más fácil.

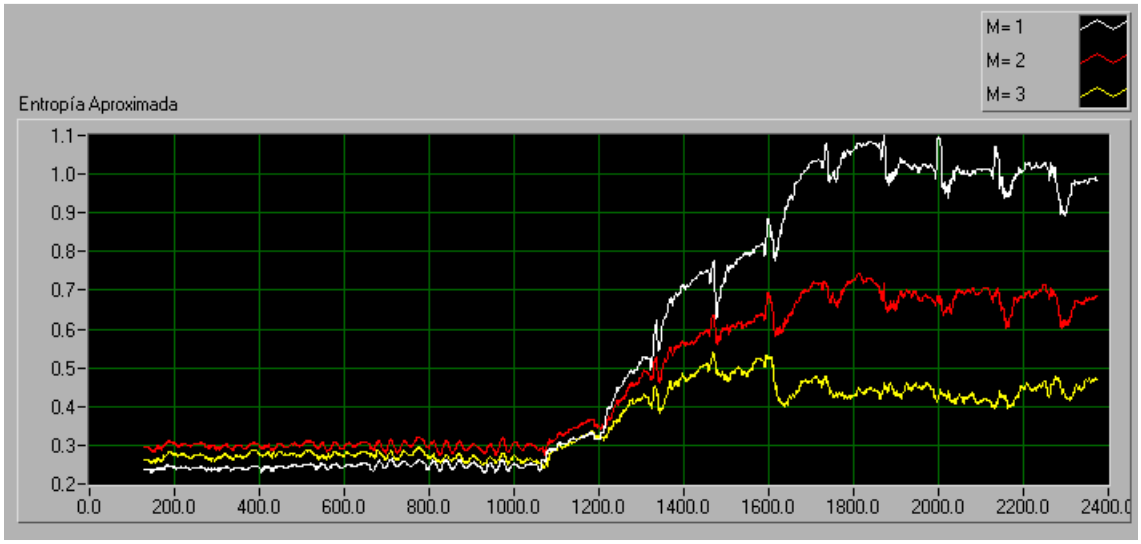


Figura 6.5: *Evolución de la Entropía Aproximada a lo largo del registro /ma/ para distintos valores de m*

Repitiendo el experimento para el registro /si/ obtenemos la Figura 6.6. Observamos que para *m=1* y *2*, la señal se caracteriza bien a través de la entropía aproximada, siendo ésta peor para *m=3*. Además, al igual que ocurría con el registro /ma/, el mayor salto entre los sonidos puros se da en el caso *m=1*, luego será este el valor que tomaremos como óptimo.

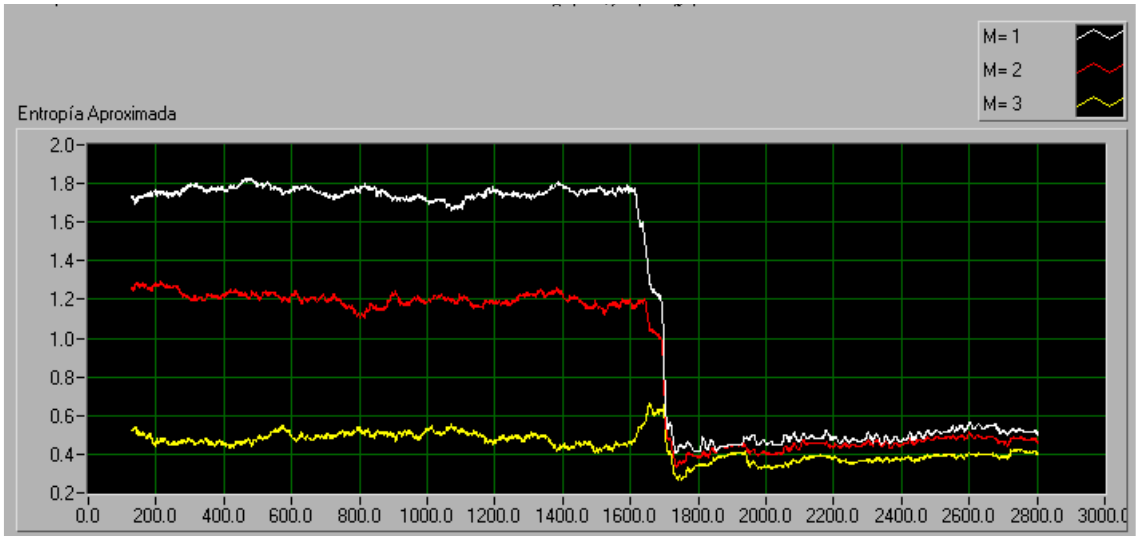


Figura 6.6: *Evolución de la Entropía Aproximada a lo largo del registro /si/ para distintos valores de m*

Por último, buscamos el valor óptimo del tamaño de la ventana (*N*) que desplazamos para calcular la entropía. Como en los casos anteriores aplicamos el algoritmo con tres valores de *N* distintos (128, 256 y 512) al registro /ma/, manteniendo los valores α*=0.3* y *m=1*, el resultado se presenta en la siguiente figura:

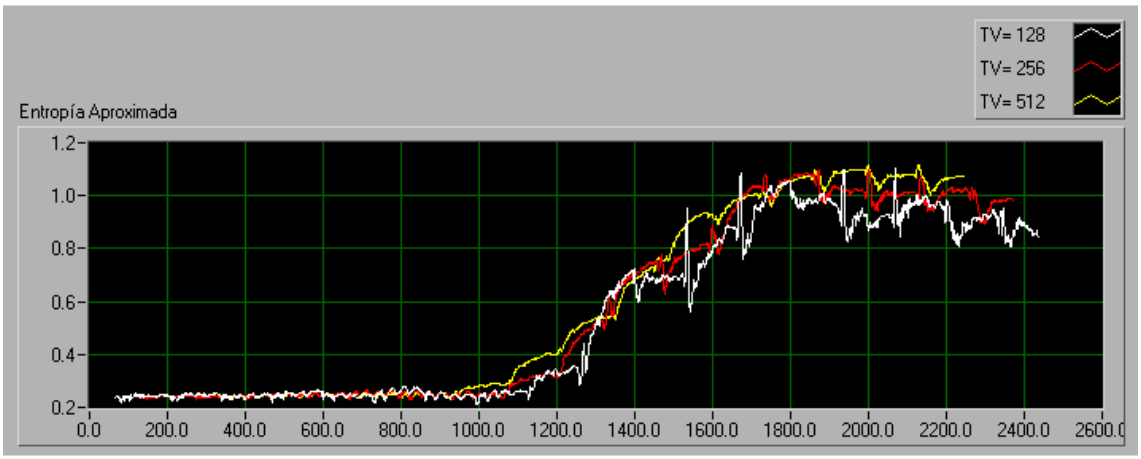


Figura 6.7: *Evolución de la Entropía Aproximada a lo largo del registro /ma/ para distintos valores de N*

Como vemos en la figura anterior para *N=256* y *N=512*, el rizado tanto en la transición como en la zona correspondiente al segundo sonido puro (el correspondiente al sonido /a/) es menor que para *N=128*. Por otro lado, dejando a un

lado este rizado, no se aprecia que al variar el tamaño de la ventana se vean afectados los niveles promedio de entropía, ni la pendiente de la zona de transición.

A continuación, repetimos el experimento sobre el registro /si/ y representamos los resultados en la figura 6.8. En ella observamos que al igual que ocurría en la figura anterior para *N=256* y *N=512* el rizado es mucho menor que para *N=128*. Sin embargo en esta figura si podemos decir que con *N=128* y *N=256* se detecta la transición con mayor precisión. Si observamos la figura 6.2 vemos que la transición se da de manera muy rápida, apenas *100* puntos, luego las gráficas correspondientes a *N=128* y *N=256* que presentan una mayor pendiente en la zona de transición, la caracterizan correctamente. Por tanto si queremos tener un rizado aceptable en la serie y que además responda bien ante variaciones bruscas, el valor óptimo de la ventana será *256.*

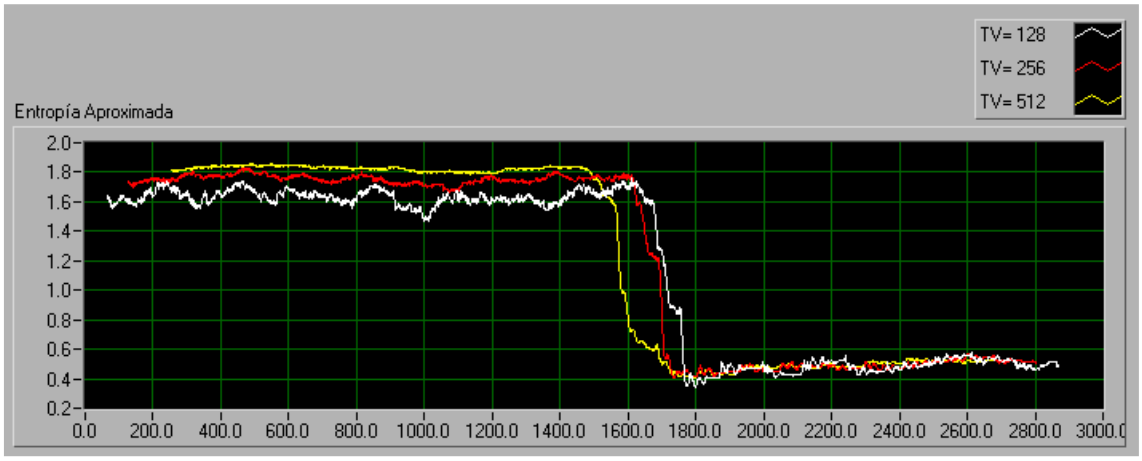


Figura 6.8: *Evolución de la Entropía Aproximada a lo largo del registro /si/ para distintos valores de N*

En resumen, los valores óptimos de entrada para el algoritmo de cálculo de la Entropía Aproximada serán:

- *m=1*
- *α=0.3*
- *N = 256*

6.1.2. Comparación entre Entropía de Renyi y Entropía Aproximada

Tal y como veíamos en el capítulo 2 al estudiar la expresión [2.5] para el cálculo de la Entropía de Renyi de Orden q (K_q), decíamos que las posibles entropías K_i que se podían calcular presentaban la siguiente propiedad:

$$K_q > K_{q'} \text{ para todo } q' > q \, .$$

Considerando que la expresión para calcular la Entropía Aproximada es un caso particular de la Entropía de Renyi de orden q, concretamente para *q=1*, y que la

Entropía de Renyi que hemos computado es la de orden 2 (K_2), es de suponer que si aplicamos los algoritmos creados a dos señales y comparamos las dos series de resultados, la evolución de la Entropía Aproximada será superior a la de la Entropía de Renyi, a igualdad de parámetros de entrada.

Si aplicamos el anterior comentario a las señales con las que hemos ido trabajando a lo largo de este capítulo verificamos que se cumple la propiedad anterior.

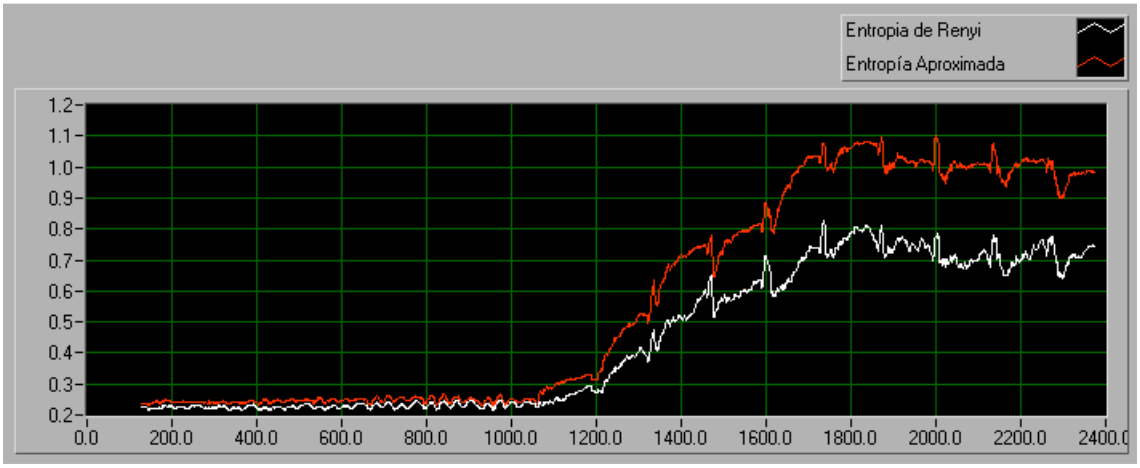


Figura 6.9: *Comparación entre Entropía de Renyi y Entropía Aproximada para el registro /ma/*

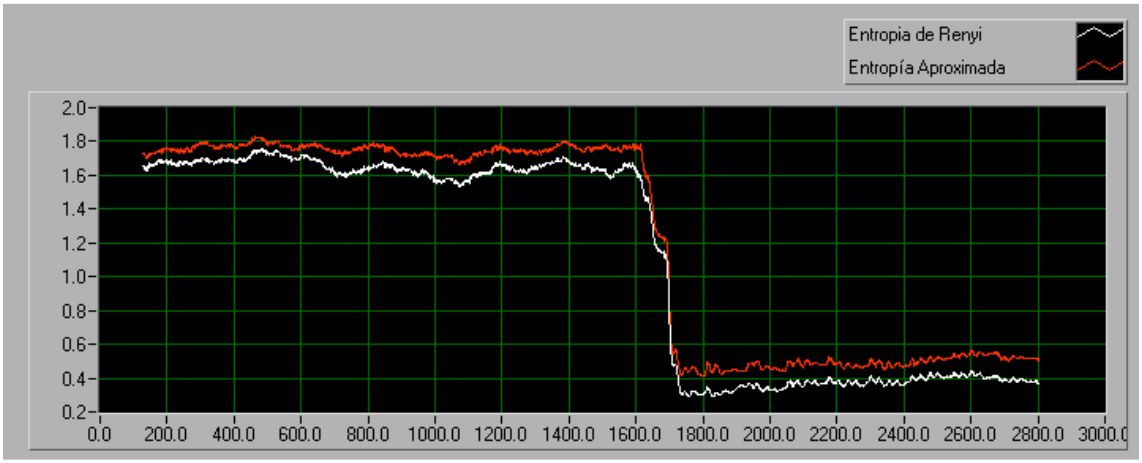


Figura 6.10: *Comparación entre Entropía de Renyi y Entropía Aproximada para el registro /si/*

6.1.3. Dinámica simbólica

Al igual que ocurría con el algoritmo para el cálculo de la Entropía Aproximada, para la Dinámica Simbólica es necesario la estimación de los valores óptimos de algunos parámetros. En este caso son:

- Factor de modulación (α) del umbral, ε.

- La longitud de la ventana usada, *N*.

El primer paso será observar como afecta la variación de α a la dinámica simbólica. Para ello aplicamos el algoritmo con tres valores de α distintos (*0.1, 0.3* y *0.5*) a la señal que apare representada en la figura 5.1, manteniendo *N=256*.

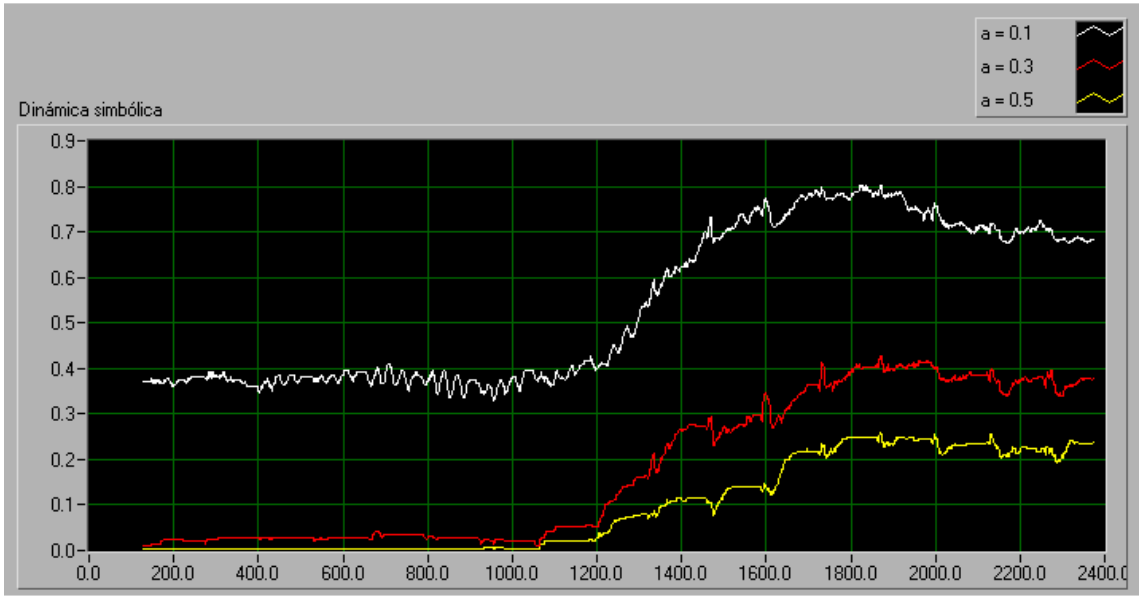


Figura 6.11: *Evolución de la Dinámica Simbólica a lo largo del registro /ma/ para distintos valores de* α

Tal y como se aprecia en la figura anterior, la señal queda caracterizada correctamente por cualquiera de las tres gráficas, si bien observándolas con atención comprobamos que la diferencia entre los valores que adopta la dinámica entre las dos zonas estables es mayor para α*=0.1* y α*=0.3* que para *0.5*. Además, existe menos rizado en la zona correspondiente al primer sonido puro en los casos α*=0.3* y α*=0.5* que cuando α*=0.1*. Además el valor de la dinámica simbólica en el segundo tramo de sonido puro permanece más estable, ya que para α*=0.1* se aprecia una pendiente descendente tras la transición Luego todo parece indicar que la mejor opción es tomar α*=0.3.*

Si repetimos el experimento para la señal de la figura 6.2 llegaremos a la misma conclusión, ya que para α*=0.3* y α*=0.5* el valor de la dinámica simbólica en las zonas correspondientes a sonidos puros es prácticamente constante, mientras que para α*=0.1* no. En este caso las diferencias en amplitud entre las dos zonas estables de la serie son prácticamente idénticas para α*=0.3* y α*=0.5*, a diferencia de lo que ocurría para el registro /ma/, donde esta diferencia era mayor para *0.3* que para *0.5*. Debido a esto parece lógico pensar que el valor óptimo de α debe ser *0.3*, ya que se adapta mejor a los dos tipos de transiciones.

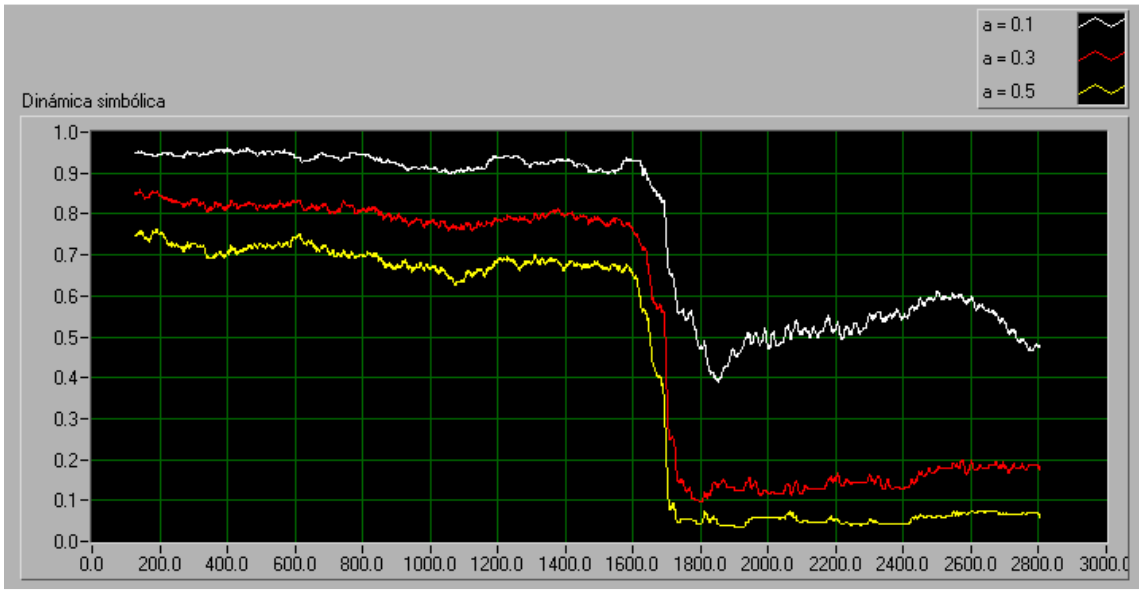


Figura 6.12: *Evolución de la Dinámica Simbólica a lo largo del registro /si/ para distintos valores de* α

Por último fijamos el valor óptimo del tamaño de la ventana (*N*). Aplicamos el algoritmo con tres valores de *N* distintos (*128, 256* y *512*) al registro /ma/, manteniendo α*=0.3,* el resultado lo tenemos en la siguiente figura:

Como vemos en la figura 6.13, para *N=256* y *N=512*, el rizado tanto en la transición como en la zona correspondiente al segundo sonido puro (el correspondiente al sonido /a/) es menor que para *N=128*. Por otro lado, dejando a un lado este rizado, no se aprecia que el variar el tamaño de la ventana afecte a los niveles promedio de Dinámica, ni a la pendiente de la zona de transición.

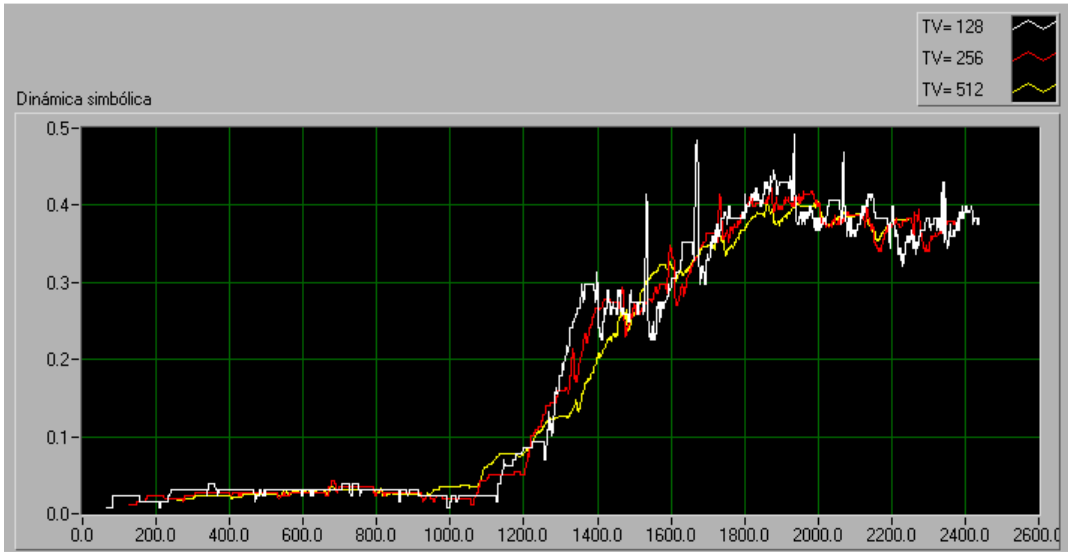


Figura 6.13: *Evolución de la Dinámica Simbólica a lo largo del registro /ma/ para distintos valores de N*

A continuación, repetimos el experimento sobre el registro /si/ y representamos los resultados en la figura 6.14. En ella observamos que al igual que ocurría en la figura anterior para *N=256* y *N=512* el rizado es mucho menor que para *N=128*. Sin embargo en esta figura si podemos decir que con *N=128* y *N=256* se detecta la transición con mayor precisión. Si observamos la figura 6.2 vemos que la transición se da de manera muy rápida, apenas 100 puntos, luego la gráficas correspondientes a *N=128* y *256* que presentan una mayor pendiente en la zona de transición, la caracterizan correctamente. Por tanto si queremos tener un rizado aceptable en la serie y que además responda bien ante variaciones bruscas, el valor óptimo de la ventana será *256*.

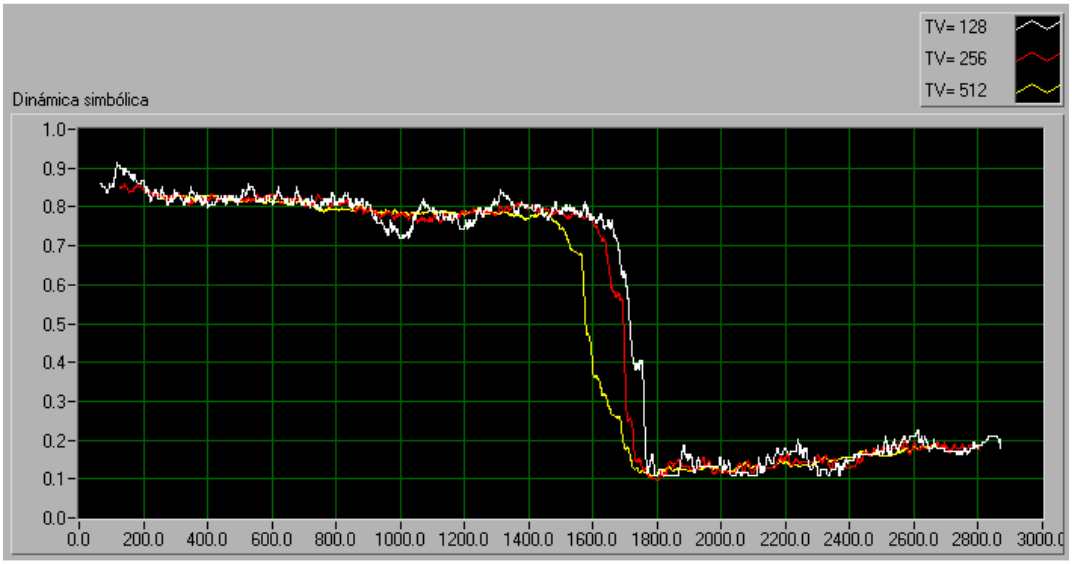


Figura 6.14: *Evolución de la Dinámica Simbólica a lo largo del registro /si/ para distintos valores de N*

En resumen, los valores óptimos de entrada para el algoritmo de cálculo de la Dinámica Simbólica serán:

- $\alpha=0.3$
- $N = 256$

Por último, destacaremos que la principal ventaja de la Dinámica Simbólica es su velocidad de cálculo frente al resto de algoritmos presentados. En la siguiente tabla se presentan los tiempos de cálculo necesarios para obtener la caracterización de una señal de 2400 puntos de longitud. Los parámetros de entrada para todos los algoritmos serán: *N=256, a=0.3* y *d=1*. El procesador en el que se ejecutan los algoritmos será un Pentium IV a 1.4 GHz.

MÉTODO	TIEMPO (s)
Dinámica Simbólica	0.103
Entropía Aproximada	372.551
Entropía Aproximada Rápida	77.297
Entropía de Renyi	200.569

6.2. Modelo ARMA. Número óptimo de polos y ceros.

En este partado trataremos de seleccionar el número óptimo de polos y ceros para la modelización ARMA de sonidos del habla natural. Para ello calcularemos el error cuadrático medio cometido al aproximar el espectro de potencia de una señal mediante su envolvente ARMA. La señal que utilizaremos será un segmento correspondiente a un sonido vocálico con un tamaño de 512 puntos equivalentes a 32ms.

En la Figura 6.15. se representa el MSE (Mean Square Error) variando el número de polos desde *1* a *17* y el número de ceros desde *0* (modelo AR) a *4.* Para un número de ceros mayor que *4* resulta un filtro inestable.

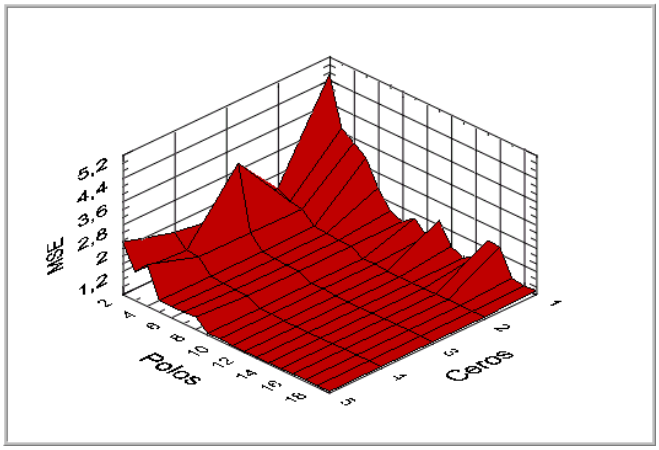


***Figura 6.15**: Análisis del MSE para AR y ARMA y diferente número de polos y ceros en la vocal "a".*

Con el fin de poder apreciar los cambios relativos al variar el número de polos, en la Figura 6.16, se parte de un número de polos *p=6*. Se observa que un aumento del número de coeficientes a partir de *p=10* no proporciona una disminución significativa del MSE.

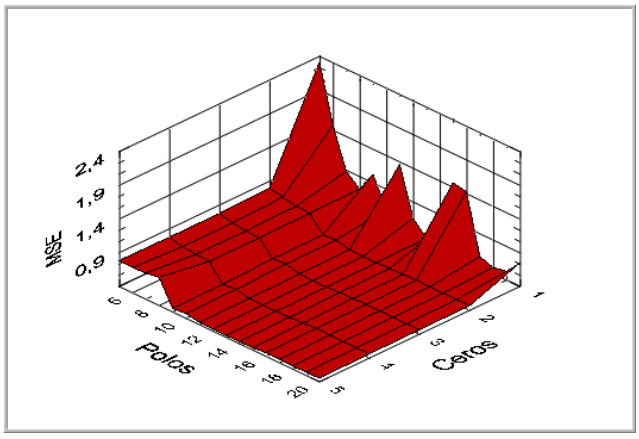


Figura 6.16: *Análisis del MSE para ARMA y diferente número de polos y ceros en la vocal "a".*

Si se representa esta misma gráfica a partir de un número de ceros *q=2*, se observan mejor los posibles cambios en el MSE respecto a la variación del número de ceros. En la Figura 6.17 se puede obsevar que estas variaciones no resultan significativas.

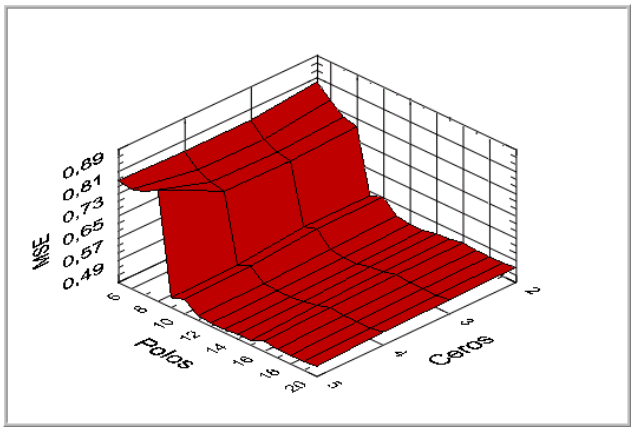


Figura 6.17: *Análisis del MSE para ARMA y diferente número de polos y ceros en la vocal "a".*

Aplicando el anterior criterio para la selección del orden de un modelo ARMA sobre distintas situaciones fonológicas presentes en el habla natural, se ha optado por trabajar con el modelo ARMA(12,2).

En las siguientes figuras presentamos una comparación entre las envolventes calculadas a partir de modelos ARMA y el espectro de potencia de distintos segmentos del habla conectada de diferente naturaleza fonética. Los segmentos estudiados serán de 512 puntos muestreados a 16000 Hz (32 ms), y los modelos ARMA tendrán 12 polos y 2 ceros:

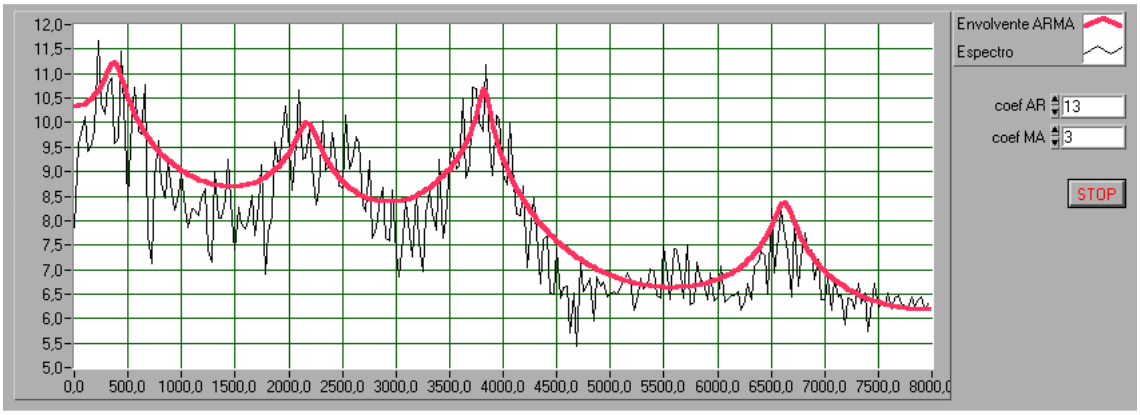


Figura 6.18: *Análisis frecuencial con espectro de potencia y envolvente ARMA del sonido /e/*

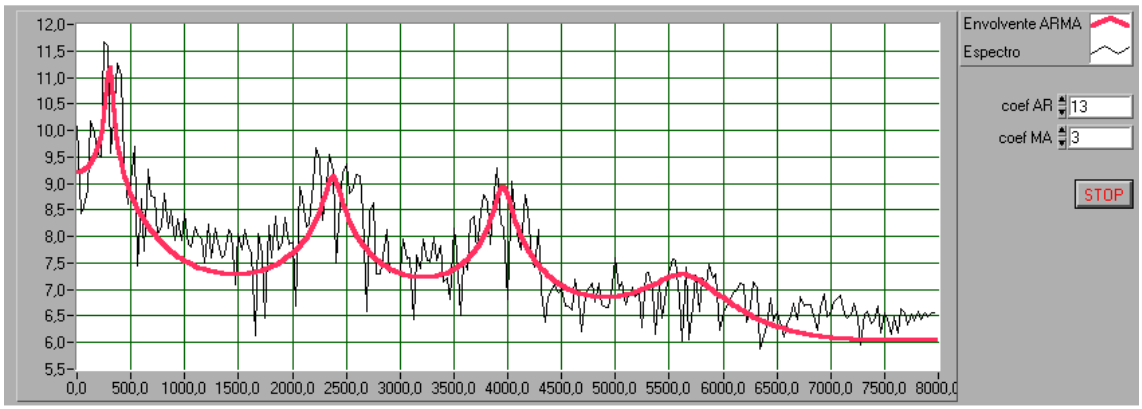


Figura 6.19: *Análisis frecuencial con espectro de potencia y envolvente ARMA de la transición /ci/*

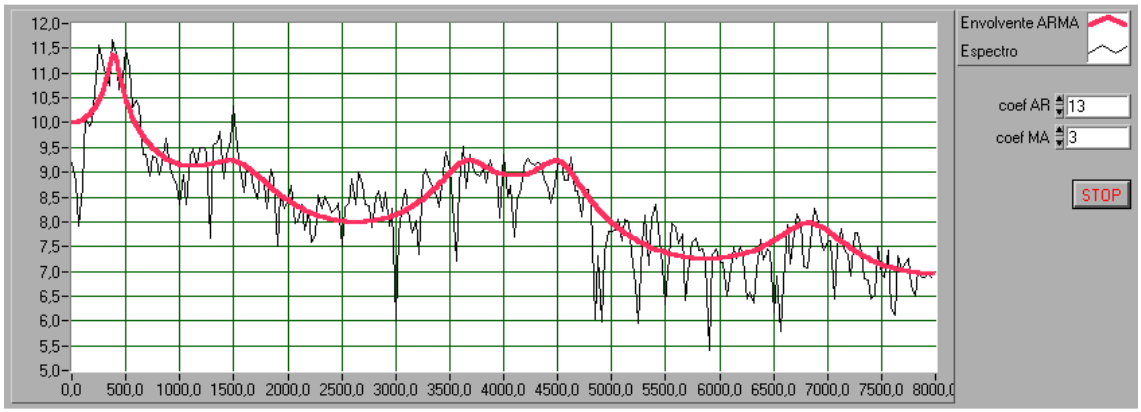


Figura 6.20: *Análisis frecuencial con espectro de potencia y envolvente ARMA de la transición /to/*

6.3. Reconocimiento de Locutores

En este apartado presentamos los resultados obtenidos con nuestro sistemas de reconocimiento de locutores, en cuya etapa de comparación utilizaremos la métrica ARMA presentada en el apartado 3.2.2.3. Estos resultados serán comparados con los obtenidos para las mismas muestras mediante la técnica clásica de clasificación del Análisis discriminante (apartado 3.2.2.1).

Los resultados obtenidos los expresamos en función de unos estadísticos que a continuación presentamos:

- **Eficiencia:** Es la proporción entre clasificaciones correcta y el número total de clasificaciones realizadas. En la práctica podemos expresarla como:

$$Eficiencia = \frac{TP + TN}{TP + TN + FP + FN} \cdot 100$$

Siendo:

- *TN*: Verdadero negativo.
- *TP*: Verdadero positivo.
- *FP*: Falso positivo.
- *FN*: Falso negativo.

- **Sensibilidad:** Es la proporción entre el número de observaciones a las que se le ha aceptado correctamente la identidad reclamada y el número de observaciones que reclamaban una identidad correcta. En la práctica:

$$Sensibilidad = \frac{TP}{TP + FN} \cdot 100$$

- **Especificidad:** Es la proporción entre el número de observaciones a las que se le ha rechazado correctamente la identidad reclamada y el número de observaciones que reclamaban una identidad falsa. En la práctica:

$$Especificidad = \frac{TN}{TP + FP} \cdot 100$$

- **Valor predictivo negativo:** Representa la proporción entre el número de observaciones a las que correctamente se le ha rechazado la identidad reclamada y el número total de observaciones a las que se le ha rechazado la identidad reclamada. En la práctica:

$$Valor\ predictivo\ negativo = \frac{TN}{TN + FN} \cdot 100$$

6.3.1. Comparación de resultados entre AD y Métrica ARMA

En este apartado, se presentan los resultados obtenidos para las dos técnicas de clasificación anteriormente mencionada. Estas han sido aplicadas sobre segmentos transicionales con coarticulación nasal, extraídos de la base de datos Ahumada. En concreto los segmentos tendrán una duración de 80 ms (1280 puntos muestreados a 16000 Hz). El número de locutores empleados es 18 y el modelo de cada locutor se obtendrá a partir de seis observaciones del mismo.

Los resultados obtenidos para el Análisis Discriminante son:

Transición	MA	MI	ÑA
Sensibilidad	89.72%	90.89%	90.12%
Especificidad	82.32%	81.08%	82.18%
Valor Predictivo Negativo	99.34%	99.34%	99.29%
Eficiencia	82.69%	81.62%	82.62%

Tabla 1

Los resultados obtenidos para la Métrica ARMA son:

Transición	MA	MI	ÑA
Sensibilidad	89.16%	91.66%	91.66%
Especificidad	84.21%	83.38%	85.07%
Valor Predictivo Negativo	99.41%	99.41%	99.42%
Eficiencia	84.45%	83.84%	84.4%

Tabla 2

Como vemos los resultados son similares para ambas técnicas, si bien se observa una ligera mejoría al aplicar la métrica ARMA. Esta mejora se aprecia sobre todo en la especificidad y en la eficiencia donde los porcentajes suben en 2 puntos

A continuación y tras ver que la nueva métrica ARMA que presentamos en este proyecto ofrece mejores resultados que el AD tratamos de mejorar el sistema de clasificación. Para ello, tomaremos tramos de señal de mayor longitud, los cuales contendrán tanto tramos transicionales como sonidos sostenidos, en concreto tomaremos tramos de 160 ms (2560 puntos).

Para modelar mejor la señal, la dividiremos en 10 segmentos y obtendremos un modelo M_K para cada segmento. Aplicaremos la métrica segmento a segmento, y finalmente tendremos el grado de similitud entre ambas señales mediante la siguiente métrica:

$$D(s,s') = \sum_{K=1}^{10} d(M_K M'_K) ,$$

donde *s* y *s'* son las señales del habla comparadas

La siguiente tabla presenta los resultados obtenidos para tres registros distintos. El número de locutores empleados en este caso será 20 y el número de observaciones por locutor 6.

Registro	ANA	OMA	MAN
Sensibilidad	97.5%	98.3%	96.7%
Especificidad	95.1%	95.4%	96.3%
Valor Predictivo Negativo	99.8%	99.9%	99.8%
Eficiencia	97.5%	99.2%	96.7%

Tabla 3

Como se puede apreciar si comparamos estos últimos resultados con los que aparecen en la tabla 2, los porcentajes correspondientes son muy superiores a pesar de que el número de poblaciones ha aumentado (pasa de 18 a 20), con lo que la clasificación en principio es más difícil. Esto indica que el modelado de la señal es fundamental en un sistema de reconocimiento de locutores. Conforme mejor sea el modelado de la señal, los porcentajes de acierto que obtendremos al realizar la clasificación serán mayores.

Por último, si aplicamos la métrica ARMA a tramos transicionales sin coarticulación nasal bajo las mismas condiciones del experimento anterior, obtenemos los siguientes resultados:

Registro	BE	SA
Sensibilidad	95.83%	94.16%
Especificidad	94.86%	97.8%
Valor Predictivo Negativo	99.77%	99.68%
Eficiencia	95.83%	95.83%

Tabla 4

Si comparamos los resultados con los reflejados en la tabla 3 vemos que disminuyen los porcentajes. Esto confirma la idea de que los tramos con transiciones en las que exista coarticulación nasal tienen una mayor dependencia del locutor, y por tanto son preferibles en los sistemas de reconocimiento de locutores frente a otros tipos de transiciones.

Capítulo 7: Conclusiones

7. Conclusiones

Teniendo en cuenta los resultados obtenidos en el desarrollo del presente proyecto, podemos destacar las siguientes conclusiones:

- Se ha constatado la naturaleza caótica de las señales del habla a través del diagrama de espacio - estado, observando la presencia de atractores extraños en diferentes sonidos del habla natural (vocales, nasales, fricativos,...).

- La medida de la entropía supone un parámetro muy adecuado para el estudio de los cambios en la complejidad de la señal del habla a lo largo de un discurso continuo; lo cual nos lleva a pensar en esta medida como un criterio alternativo a los que tradicionalmente se han utilizado en los sistemas de segmentación. Esta última afirmación cobra mayor importancia si pensamos en aquellas señales del habla natural que están implicadas transiciones entre sonidos de distinta naturaleza.

- Asimismo, la Dinámica Simbólica constituye otra técnica de la dinámica no lineal que nos ofrece similares resultados para el estudio de la regularidad de las señales del habla natural, que los obtenidos a través de la aplicación de los algoritmos de Entropía. Frente a estos algoritmos, la aplicación de la Dinámica Simbólica presenta como ventaja su mayor rapidez de cómputo.

- En este sentido, en la actualidad nuestras investigaciones se centran en la Entropía de Renyi de orden q, en concreto, en la selección de los parámetros implicados en su formulación; con el fin, de que la aplicación de esta medida más general de la entropía a las señales del habla natural mejore los resultados obtenidos hasta el momento.

- Teniendo en cuenta que uno de los objetivos principales de este proyecto es el reconocimiento de locutores por su voz, la etapa del modelado de las señales es una etapa crítica en este problema. En este sentido, la elección del modelo ARMA para la caracterización de señales del habla, especialmente en aquellos tramos en los que están presentes transiciones entre sonidos de distinta naturaleza, se presenta muy adecuado teniendo en cuenta los resultados obtenidos. El cálculo de los coeficientes MA del modelo se ha llevado a cabo mediante un método directo denominado Método de Shanks.

- Se ha implementado una nueva métrica entre modelos ARMA, obtenidos a partir de señales del habla, que supone una alternativa a las clásicas distancias cepstrales, al calcularse directamente a partir de los coeficientes del modelo.

- Planteando un sistema de reconocimiento de locutores que en su etapa de comparación utilice la métrica ARMA introducida, se obtiene un alto índice de aciertos tanto en identificación como en la verificación de locutores superior a los obtenidos a través de la técnica clásica del Análisis Discriminante.

- La hipótesis que asumimos en el planteamiento de este proyecto sobre la utilización de las señales del habla natural que presentan coarticulación nasal,

como señales adecuadas para la comparación de locutores ha quedado confirmada de acuerdo a los resultados obtenidos.

- Los resultados obtenidos en el estudio del reconocimiento de locutores por sus voces en el presente proyecto, nos inducen a pensar en el desarrollo de un sistema automático de reconocimiento de locutores con las dos especificaciones siguientes:
    - Trabajar fundamentalmente con sonidos del habla natural que contengan tramos transicionales con presencia de coarticulación nasal
    - La etapa de decisión se realizará comparando señales mediante la métrica ARMA.

Bibliografía

Bibliografía

Libros:

- **Box,G.; Jenkins, G.**, "*Time Series Analysis: Forecasting and Control*".
  Ed. Prentice-Hall, 1986.

- **González Fuentes, F.,** "*Modelado no lineal de señales del habla*".
  Proyecto Fin de Carrera, 2001.

- **Holmes, J.N.,** "*Speech syntesis and recognition*".
  Ed. Chapman & Hall, 1988.

- **Klevans, R.; Rodman, R.,** "*Voice Recognition*".
  Ed. Artech House Publishers, 1997.

- **Oppenheim, A.; Willsky, A.,** "*Señales y Sistemas*".
  Ed. Prentice Hall, 1994.

- **Proakis, J.; Manolakis, D.**, "*Tratamiento digital de señales*".
  Ed. Prentice Hall, 1988.

- **Rabiner, L.; Schafer, R.**, "*Digital Processing of Speech Signals*".
  Ed. Prentice Hall, 1978.

- **Therrien, C.**, "*Discrete Random Signals and Statistical Signal Processing*".
  Ed. Prentice Hall, 1992.

Artículos:

- **Atal, B.,** "*Automatic Recognition of Speakers from their Voices*".
  Proceedings IEEE, 1976.

- **Atal, B.; Hanauer, S.,** "*Speech Analysis and Synthesis by Linear Prediction of the Speech Wave*"
  The Journal of the Acoustical Society of America, 1971.

- **Grassberger, D.; Procaccia, I.,** "*Measuring the strangeness of strange attractors*"
  Physica D9: 189-208, 1983.

- **Grassberger, D.; Procaccia, I.,** "*Estimation of the Kolmogorov entropy from a chaotic signal*"
  Phys. Rev, A28, pp. 2591-2593, 1983.

- **Grassberger, D.; Procaccia, I.,** "*Characterizacion of Strange Attractors*".
  Phys Rev Let, vol 50 pp: 346-349, 1983.

- **Guillén, P.; Jensen, E.; Litvan, H.; Vallverdú, M.; Caminal, P.,** "*Entropía aproximada y Dinámica Simbólica para cuantificar regularidad en señales electroencefalográficas de pacientes anestesiados*".
  Actas CASEIB 2000.

- **Jörgen, M.; Heiko, M.; Andreas, M.,** "*Approximate Entropy as an Electroencephalografic Measure of Anesthetic Drug Effect during Desflurane Anesthesia*".
  Proceedings American Society of Anestheosiologists, 2000.

- **Makhoul, J.,** "*Linear Predictions: A tutotial review*".
  Proceedings IEEE, 1976.

- **Martin, R.,** "*A Metric for ARMA Processes*".
  Proceedings IEEE, 2000.

- **Martínez González, F.; García Sánchez, A.; Guillamón Frutos, A.; González Fuentes, F.,** "*Modelado ARMA de señales del habla*".
  Actas CASEIB 2000.

- **Martínez González, F.; García Sánchez, A.; Guillamón Frutos, A.; Paredes Hernández, S.; González Fuentes, F.,** *"Desarrollo de un sistema de segmentación automático de señales del habla*".
  Actas SEAF 2000.

- **Pritchard, W.; Duke, D.,** "*Measuring chaos in the brain: A Tutorial Review of EEG Dimension Stimation*".
  Academic Press, Inc. 1995.

- **Shanks, J.,** "*Recursion filters for digital processing*".
  Proceedings Geophysics, 1967.

- **Takens, F.,** "*Invariants related to Dimension and Entropy*" .
  Proceedings of the 13th Coloquio Brasileiro de Matemáticas, 1983.

- **Grigorias, F.; Teodorescu, H.; Apopei, V.,** *"Nonlinear and Nonstationary Processes in Speech Production*".
  Int. J. of Chaos Theor. And Appl. Vol.2, pp 1453-1457, 1997.

Anexos

Anexo I: Base de Datos Ahumada
Anexo II: Archivos de Sonido

ANEXO I: Base de Datos Ahumada

Se trata de una Base de Datos de Locutores (BDDL) en idioma español, desarrollada en el Departamento de Ingeniería Audiovisual y Comunicaciones de la E.U.I.T. de Telecomunicaciones de la Universidad Politécnica de Madrid (García Jiménez R. Y Díaz Gómez J., 1998).

Es una base de datos donde se controlan las condiciones de grabación evitando los agentes externos que alterarían los parámetros identificativos de la grabación. Se compaginan tomas microfónicas (en laboratorio) con tomas telefónicas (con líneas urbanas e interurbanas).

Sus principales características son:

- 103 locutores varones.
- Base de datos multisesión (tres sesiones por grabación distanciadas entre sí al menos quince días).
- Cada sesión de grabación dividida en dos partes:

    - A. Grabación Microfónica
    - B. Grabación Telefónica

Tareas realizadas por sesión

Cada locutor realiza la pronunciación de las siguientes secuencias por sesión de grabación:

1. Una serie de veinticuatro dígitos aislados. Dos repeticiones de cada dígito (0,1,2,3,4,5,6,7,8,9) repartidas aleatoriamente dentro de la serie. Añadiendo dos dígitos al inicio de la serie y otros dos al final para evitar inicios de lectura inadecuados y finales con "tonillos". La lectura de la serie es seguida, con una velocidad normal y con cada número pronunciado individualmente, separando un número del siguiente con una breve pausa. La serie es la misma para todos los locutores, siendo la que se especifica a continuación:

    23 // 78610496325874190237 // 64

2. Diez cadenas o series de diez dígitos cada una. Cada dígito de los diez existente repetidos de forma aleatoria. La lectura de la serie es seguida, con una velocidad normal y con cada número pronunciado individualmente, pero de manera concatenada, con una breve pausa entre serie y serie, evitando la entonación variante.

- Cinco series iguales para todos los locutores.
- Cinco series propias del locutor. Este tipo de series van confeccionadas de forma aleatoria, procurando que ningún locutor lea series iguales.

3. Diez frases cortas (entre ocho y diez palabras) equilibradas fonéticamente. Son leídas de forma consecutiva, haciendo pausa entre frase y frase.

4. Un texto común equilibrado silábica y fonéticamente, de unos setenta u ochenta segundos de duración. Este texto es común a todos los locutores, leído tres veces en la toma microfónica y una en la telefónica. Cada una de las tres lecturas de la toma microfónica se realiza a diferentes velocidades de locución.

5. Un texto propio extraído de un libro cuya duración será de unos sesenta o setenta segundos aproximadamente. La lectura se hará a una velocidad de locución normal. El texto será personal para cada locutor e independiente uno de otro.

6. Habla espontanea durante un minuto y medio aproximadamente, para obtener un minuto de habla real por lo menos.

Sistemas de almacenamiento

La toma microfónica para las tres sesiones se ha materializado en 64 cintas DAT, que constituyen las cintas originales o master. De ellas 37 han registrado las tomas “IN SITU” o en el estudio de grabación, siendo 28 de 64 minutos y 9 de 94 minutos. En las otras 27 cintas se han grabado las muestras telefónicas. Hay 19 de 64 minutos y 8 de 94 minutos. La frecuencia de muestreo es de 48 KHz. para las tomas en directo con los micrófonos del estudio (solapa o de sobremesa) y de 44,1 KHz. para los registros telefónicos. Los formatos de cintas utilizados han sido: DAT SONY DT-60 RN, TDK DA-R STUDIO 60, KAO KX-PRO 60 y AMPEX 467 R-64 ó R-94.

Todas estas cintas master han sido copiadas íntegramente en otras 49 cintas DAT AMPEX R-94.

Se construyen los ficheros de cada locutor procesando las muestras registradas en las cintas DAT con el conversor analógico/digital PROPORT, la tarjeta ARIEL y el *software* de HYPERSIGNAL. Esto permite muestrear nuevamente las grabaciones microfónicas a 16 KHz y las telefónicas a 8 KHz. El sistema de almacenamiento informático son CD’s. De ellos existe copia de seguridad.

Nomenclatura de los ficheros

Para almacenar los datos referentes a los 103 locutores, sabiendo en cada momento y a la vista del nombre del fichero a qué locutor pertenece (del 000 al 103), de qué sesión se trata (1,2,3), si es vía telefónica o microfónica y en qué tarea nos encontramos (A, B, ..., F), se ha adoptado por convenio el siguiente formato de denominación de ficheros:

A. Adoptaremos el máximo numero de caracteres que admite el sistema operativo MS-DOS, es decir, 8 caracteres, distribuidos como sigue :

- Tres primeros caracteres son números que nos indican de que locutor se trata.

- Caracteres **4º y 5º** con la siguiente nomenclatura :

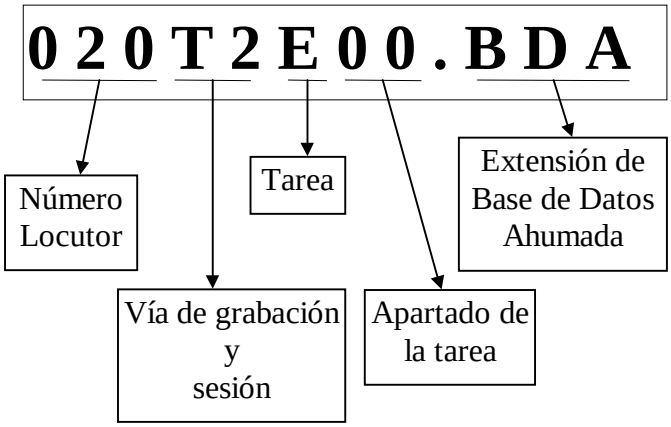
  - "T1" ⇒ Grabación telefónica, 1ª sesión, cuña telefónica. (Canal R).
  - "T2" ⇒ Grabación telefónica, 2ª sesión, cuña telefónica. (Canal R).
  - "T3" ⇒ Grabación tele-microfónica, 3ª sesión, cuña telefónica. (Canal L).
  - "M1" ⇒ Grabación microfónica, 1ª sesión, micrófono SONY de solapa. (Canal L).
  - "M2" ⇒ Grabación microfónica, 2ª sesión, micrófono SONY de solapa. (Canal L).
  - "M3" ⇒ Grabación microfónica, 3ª sesión, micrófono SONY de solapa. (Canal L).
  - "M4" ⇒ Grabación microfón., 1ª sesión, micrófono AKG de sobremesa. (Canal R).
  - "M5" ⇒ Grabación microfónica, 2ª sesión, micrófono AKG de cascos. (Canal R).
  - "M6" ⇒ Grabación microfón., 3ª sesión, micrófono TARGET de solapa. (Canal R).
  - "M7" ⇒ Grabación tele-microf., 3ª sesión, micrófono SONY de solapa. (Canal R).

- El **6º** carácter nos determina la tarea a realizar.

- Los dos últimos caracteres (el **7º y 8º**) nos determinan la subtarea dentro de las que son múltiples, poniendo ceros en las que sean simples y no posean apartados.

B. El nombre adoptado para la extensión son las siglas que dan nombre propio a la base de datos: "BDA" que significan "Base de Datos Ahumada".

Un ejemplo de todo lo anterior nos puede aclarar la explicación hecha de la nomenclatura de los ficheros de voz . Sea el fichero **020T2E00.BDA**:



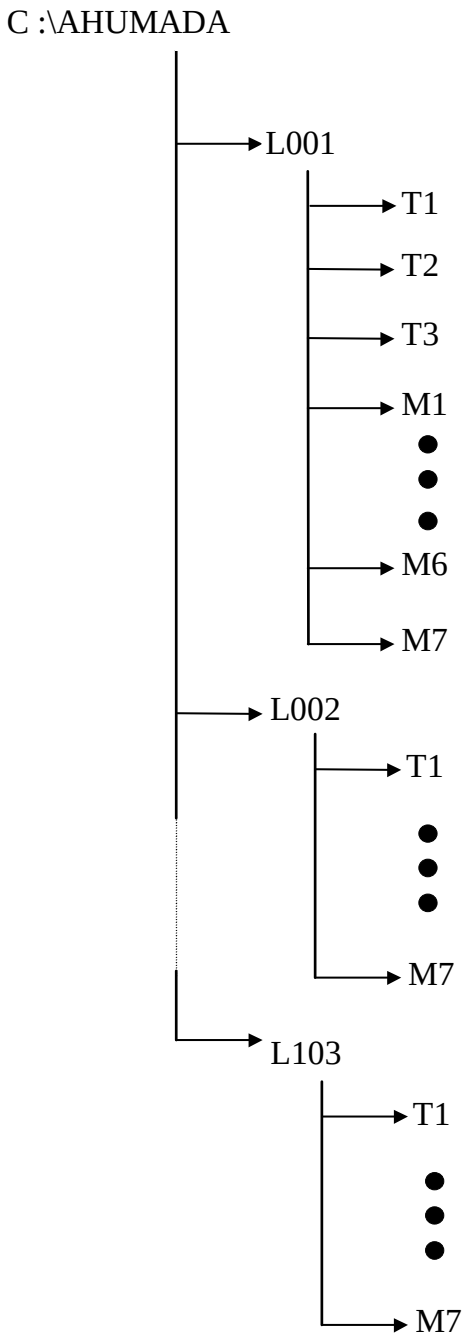
En este ejemplo, a la vista del esquema gráfico y según las directrices explicadas anteriormente se deduce :

- Se trata del locutor número 20 (tres primeros caracteres “020”).
- Estamos en la segunda sesión de grabación vía teléfono (dos siguientes caracteres “T2”).
- Según el sexto carácter (“E”) el locutor ha leído un texto propio (libro, periódico, etc.).
- Los dos caracteres que siguen a la “E” (“00”), nos indican que esta tarea no posee ningún apartado o subtarea.

Estructura del árbol de ficheros

A partir de un directorio raíz denominado “AHUMADA” cuelgan los subdirectorios de locutores. Es decir, de “AHUMADA” penden los 103 subdirectorios que se corresponden con otros tantos locutores y que se denominan con la letra “L” de locutor seguida del número a que corresponda. Por ejemplo: el directorio “L098” corresponde al locutor número 98. Siguiendo en esta línea y dentro de cada subdirectorio de locutor existen 9 subdirectorios que pertenecen a la sesión y vía de grabación. Estos subdirectorios son “T1”, “T2”, “T3”, “M1”, “M2”, “M3”, “M4”, “M5”, “M6” y “M7”. Con esta estructura los ficheros ya estarán donde les correspondan. Además es de saber que los nombres son únicos de cada tarea sesión y locutor ; nunca existirán dos nombres iguales dentro de toda la base de datos.

En la siguiente figura vemos como queda estructurado el árbol de directorios



En este esquema se ve la estructura organizativa de los CD's, donde se ha de procurar que estén los locutores completos a ser posible, en los mismos.

Sabiendo la etiquetación de los ficheros ya podemos saber el número total de ellos, dentro de la "base de datos AHUMADA".

- Para los ficheros en grabación microfónica existen 26 ficheros por canal en cada sesión.
- Al ser tres sesiones en estéreo son 26 ficheros por 6 canales diferentes resultan 156 ficheros por locutor en las tres sesiones microfónicas.
- Para los ficheros en grabación telefónica existen 24 ficheros por sesión.
- Son tres sesiones, pero la 3ª es por dos canales (uno para teléfono y otro para micrófono).
- Estas tres sesiones de 24 ficheros por 4 canales diferentes resultan 96 ficheros por locutor para teléfono.
- Por todo lo anterior cada locutor posee 252 ficheros.

Si se dispone de 103 locutores estamos ante la “friolera” de 25.956 ficheros para una base de datos de las características descritas en capítulos anteriores.

ANEXO II: Los archivos de sonido

El Formato WAV

Los archivos de sonido utilizados bajo el entorno gráfico *Windows*® son denominados archivos *Microsoft Waveform* poseen la extensión (.WAV), y siguen el formato Wave, que está estructurado según el estándar RIFF (*Research Interchange File Format*, Formato de Archivo de Intercambio de Recursos).

El formato RIFF

Este estándar fue desarrollado por *Microsoft* para los archivos de recursos multimedia sobre la base del formato IFF (*Interchange File Format*, Formato de Archivo de Intercambio), creado por la empresa *Electronic Arts*.

El estándar RIFF queda fijado por *Microsoft* en la convención C; según la misma, un archivo RIFF se compone de pequeñas unidades de archivo llamadas fragmentos o chunks.

Un chunk (abreviado ck), se define en dicha convención como una unidad de datos lógica que siempre tiene la misma estructura, independientemente de los datos que contenga.

El primer dato almacenado en un chunk es el nombre del propio chunk, el cual se basa en el tipo FOURCC (*Four Character Code*, Código de Cuatro Caracteres), es decir, un chunk siempre tiene que comenzar con una secuencia de cuatro caracteres ASCII que lo identifica. Tras el nombre, se indica la longitud del fragmento, en formato de 16 bits,*y la información a almacenar.

También existe la posibilidad de crear nuevos chunks, siempre que se atengan a las especificaciones de la convención C. En caso de crearse nuevos chunks, ampliando un formato, éstos han de registrarse en *Microsoft*, pero solamente se admiten aquellos cuyo nombre se componga de letras mayúsculas y números, con el fin de evitar confusiones entre distintos chunks.

La estructura de un chunk queda dividida, en general, en:

<nombre del chunk>, (ckID, 4 bytes)
<longitud del fragmento de datos> (ckSize , 4 bytes)
<fragmento de datos> (ckData , 4 bytes)

La longitud del fragmento de datos se indica en número de bytes a partir del ckSize, luego ésta no contiene los ocho bytes del ckID y el ckSize.

El elemento <fragmento de datos> puede estar, en general, compuesto por varios chunks, cada uno con su propia estructura.

Solamente existen dos tipos de chunks que pueden contener dentro de la parte de datos otros chunks. Éstos son los chunks LIST y RIFF.

El propio archivo RIFF se considera un chunk, por tanto, el primer dato almacenado en un RIFF será la cadena de cuatro caracteres “RIFF”, seguida de la longitud del fragmento de datos posterior y por último, los datos. Un archivo estructurado según el estándar RIFF siempre poseerá la estructura :

<ckID>,”RIFF”, (4 bytes)
<ckSize >, (4 bytes)
<ckData >, (4 bytes)

Definición del formato Wave

El formato Wave es un tipo de archivo estructurado según el estándar RIFF, dividido en dos chunks, que se denominan fragmento de formato y fragmento de datos. Es decir :

<nombre del tipo de archivo >,”WAVE”, (4 bytes)
<fragmento de formato> = fmt_.ck
<fragmento de datos> = data.ck

(Donde “_” significa “espacio”).

- Fragmento de formato

Es un chunk en el que se especifica el formato de los datos contenidos en el fragmento de datos. La estructura es la siguiente :

fmt*.ck

<ckID>,”fmt_”, (4 bytes)
<ckSize >, (4 bytes)
<ckData >, (4 bytes)

El nombre o ckID“fmt_” identifica al chunk como fragmento de formato.

La longitud ckSize se refiere a la longitud del fragmento de datos del fmt_.ck, que es invariablemente 16 bytes.

El fragmento de datos ckData indica los siguientes parámetros de formato:

- <etiqueta de formato>, (2 bytes), actualmente se utiliza el formato PCM (*Pulse Code Modulation*), de este dato depende la interpretación de los datos del data.ck. Dentro del entorno *Windows* se utiliza una constante llamada WAVE_FORMAT_PCM, que toma el valor “01”, para indicar el formato PCM.

- <número de canales>, (2 bytes), indica el número de canales de salida, tomará el valor “01” para ficheros mono y el valor “02” para estéreo.

- <número de*muestras por segundo>, (4 bytes), o frecuencia de muestreo a la que se reproducirá cada canal, y que está limitada a ciertos valores (11025 Hz, 22050 Hz y 44100 Hz) para las tarjetas SoundBlaster.

- <número de bytes por segundo>, (4 bytes), indica el número medio de bytes por segundo que se deben transmitir. Será igual a :

$$\text{Numero de bytes por segundo} = (\text{Numero de canales}) * (\text{Frecuencia}) * \frac{(\text{Numero de bits por muestra})}{8} \quad [4.1]$$

- <alineamiento de los bloques de datos>, (2 bytes), es el número de bytes que son necesarios para transcribir una muestra. Este dato es necesario, debido a que muchos programas de reproducción procesan un número fijo de bytes cada vez,*y deberán alinear los búferes dependiendo de los bytes utilizados por la muestra.

$$\text{Alineamiento de los bloques de datos} = (\text{Numero de canales}) * \frac{(\text{Numero de bits por muestra})}{8} \quad [4.2]$$

- <dato específico del formato>, (2 bytes), para el formato PCM es el número de bits que representan una muestra. Será igual a “1” para datos recogidos a 8 bits-mono, igual a “2” para 8 bits- estéreo y 16 bits-mono, e igual a “4” para 16 bits-estéreo.

- Fragmento de datos

Es el chunk que contiene los datos de sonido propiamente dicho. La estructura es la siguiente :

data.ck

<ckID>,"data", (4 bytes)
<ckSize >, (4 bytes)
<ckData >, (4 bytes)

El nombre o ckID "data" identifica al chunk como fragmento de datos.

La longitud ckSize se refiere a la longitud del fragmento de datos del data.ck, que variará, lógicamente, según la longitud del fichero y la frecuencia de muestreo.

El fragmento de datos ckData se compondrá de palabras de un byte; en el caso de almacenar sonido con una resolución en amplitud de 16 bits, se dividirá cada uno de los datos de 16 bits en dos palabras de 8 bits; podría pensarse en utilizar palabras dobles, es decir, de 16 bits, con lo cual se agilizaría la lectura de este tipo de archivos, pero el estándar Wave no lo permite.

En resumen, un archivo Waveform presentaría la siguiente estructura :

"RIFF"
<longTotal - longitud de la cabecera>
"WAVE"
"fmt_"
<longitud del fmt_.ck>
<etiqueta de formato>
<nº de canales>
<frecuencia>
<nº de bytes por segundo>
<alineamientos por bloque>
<especificación del formato>
"data"
<longitud del data_.ck>

. }
. }datos
. }

La longitud de la cabecera excluye el ck.ID y el ckSide del RIFF.

Se realiza un ejemplo para un sonido muestreado a 11025 Hz, con una resolución de 16 bits, en mono.

RIFF

<longTotal - 36>
WAVE
fmt_
16
1
1
11025
22050
2
16
data
<long de los datos = longTotal - 44>

. }
. }datos
. }

Todos estos datos se guardarán, como se ha dicho anteriormente, como palabras de*8 bits, con lo que deberá especificarse también la forma en que son “partidas” las palabras de 16 bits y “re-ordenadas” para su almacenamiento como bytes.

El Formato BDA

Se trata de archivos que siguen un formato propio del software *Hypersignal* de la casa *Hyperception*, muy utilizado para el procesado digital de señales (DSP), permitiendo la adquisición en tiempo real de las señales y su procesamiento.

Los ficheros de ondas creados en el entorno *Hypersignal* contienen dos secciones: una cabecera con los parámetros de la señal y formada por 10 enteros de 16 bits y una segunda sección que contiene los datos de la señal, también enteros de 16 bits.

A continuación se describe la cabecera de un archivo en formato BDA.

AMPLITUDE	Primer parámetro / atributo
FRAMESIZE	
SAMPLING FREQUENCY, REM	
FFT ORDER	
NUMBER OF DATA VALUES, REM	
FRAME OVERLAP	
DATA TYPE	
PARAMETER / ATTRIBUTE #8	No reservado, disponible para

	el usuario
SAMPLING FREQUENCY, DIV	
NUMBER OF DATA VALUES, DIV	Último parámetro / atributo
DATA VALUE 0	Primer dato (valor) de la onda
DATA VALUE 1	
.....	
DATA VALUE N-1	Ultimo dato, donde N es Number of data values

A continuación se muestra una breve descripción de cada uno de estos parámetros donde indicamos entre paréntesis con T o F si el parámetro es requerido en el dominio del tiempo o dominio de la frecuencia.

- AMPLITUDE (T,F): Máximo valor absoluto de los datos de la señal.
- FRAMESIZE (T,F): Número de muestras de la señal.
- SAMPLING FREQUENCY, REM (T,F): La frecuencia de muestreo es el ratio al cual los datos son muestreados (en Hz) y se calcula así:

Sampling Frequency,Div * 32767 + Sampling Frequency,REM

donde Sampling Frequency,Div es el parámetro número nueve.

- FFT ORDER (F): La longitud de la FFT (Fast Fourier Transform) se calcula:

$$\text{FFT length}=2^{\text{FFT*ORDER}}$$

En las señales en el dominio del tiempo, este parámetro se encuentra desactivado (no se interpreta).

- NUMBER OF DATA VALUES, REM (T,F): El número de datos (valores) de la señal (enteros de 16 bits) se calcula:

N=Number of Data Values, DIV * 16384 + Number of Data Values, REM

donde Number of Data Values,DIV*es el parámetro nº 10.

- FRAME OVERLAP: Indica la cantidad de solapamiento de la señal. Debe tener un valor distinto de cero en el dominio de la frecuencia.
- DATA TYPE (T,F): Atributo del tipo de datos, los bits 15-0 (bit 15 es el bit mas significativo, bit 0 es el bit menos significativo) se usan para definir el tipo de datos que alberga la señal como sigue:

Time Domain Source (bits 2-0)	000	arbitrary time data
	001	autocorrelation output
	010	FIR or IIR filter output
	...	reserved for *Hypersignal* use
	111	
Complex data attribute (bit 3)	0	real-sampled data (no imaginary data present)
	1	complex-sampled data (Imaginary data present in file)
LPC Order (bits 8-4)	0	waveform is not LPC autocorrelation output
	1-31	value of LPC order
Window Type (bits 11-9)	000	rectangular window (i.e. none)
	001	Hamming window
	010	Hanning window
	011	Blackman window
	100	Bartlett window
	101	
	...	user defined
	111	
Reserved (bits 15-12)		Reserved for future *Hypersignal* functions

- PARAMETER / ATTRIBUTE #8Disponible para aplicaciones de uso específico.

- SAMPLING FREQUENCY, DIVLa frecuencia de muestreo es el ratio al cual los datos son muestreados (en Hz) y se calcula así:

  Sampling Frequency,Div * 32767 + Sampling Frequency,REM

  donde Sampling Frequency,Div es el parámetro nº 3

- NUMBER OF DATA VALUES, DIVEl número de datos (valores) de la señal (enteros de 16 bits) se calcula:

  N=Number of Data Values, DIV * 16384 + Number of Data Values, REM

  donde Number of Data Values,DIV*es el parámetro nº 5.

Todos los ficheros de ondas de *Hypersignal* tienen el mismo formato. La interpretación de los datos contenidos dependerán de la extensión asignada.

Las extensiones válidas son las siguientes:

TIM Señal en Dominio del Tiempo
FRQ Datos de frecuencia
PHZ Datos de fase
PWR Espectro de potencia
REA Componente real
IMA Componente imaginaria

En el caso de la Base de Datos Ahumada se ha optado por nombrar a los ficheros con la extensión BDA.

ANEXO: Los archivos de sonido

El Formato WAV

Los archivos de sonido utilizados bajo el entorno gráfico *Windows*® son denominados archivos *Microsoft Waveform* poseen la extensión (.WAV), y siguen el formato Wave, que está estructurado según el estándar RIFF (*Research Interchange File Format*, Formato de Archivo de Intercambio de Recursos).

El formato RIFF

Este estándar fue desarrollado por *Microsoft* para los archivos de recursos multimedia sobre la base del formato IFF (*Interchange File Format*, Formato de Archivo de Intercambio), creado por la empresa *Electronic Arts*.

El estándar RIFF queda fijado por *Microsoft* en la convención C; según la misma, un archivo RIFF se compone de pequeñas unidades de archivo llamadas fragmentos o chunks.

Un chunk (abreviado ck), se define en dicha convención como una unidad de datos lógica que siempre tiene la misma estructura, independientemente de los datos que contenga.

El primer dato almacenado en un chunk es el nombre del propio chunk, el cual se basa en el tipo FOURCC (*Four Character Code*, Código de Cuatro Caracteres), es decir, un chunk siempre tiene que comenzar con una secuencia de cuatro caracteres ASCII que lo identifica. Tras el nombre, se indica la longitud del fragmento, en formato de 16 bits,*y la información a almacenar.

También existe la posibilidad de crear nuevos chunks, siempre que se atengan a las especificaciones de la convención C. En caso de crearse nuevos chunks, ampliando un formato, éstos han de registrarse en *Microsoft*, pero solamente se admiten aquellos cuyo nombre se componga de letras mayúsculas y números, con el fin de evitar confusiones entre distintos chunks.

La estructura de un chunk queda dividida, en general, en:

<nombre del chunk>, (ckID, 4 bytes)
<longitud del fragmento de datos> (ckSize , 4 bytes)
<fragmento de datos> (ckData , 4 bytes)

La longitud del fragmento de datos se indica en número de bytes a partir del ckSize, luego ésta no contiene los ocho bytes del ckID y el ckSize.

El elemento <fragmento de datos> puede estar, en general, compuesto por varios chunks, cada uno con su propia estructura.

Solamente existen dos tipos de chunks que pueden contener dentro de la parte de datos otros chunks. Éstos son los chunks LIST y RIFF.

El propio archivo RIFF se considera un chunk, por tanto, el primer dato almacenado en un RIFF será la cadena de cuatro caracteres “RIFF”, seguida de la longitud del fragmento de datos posterior y por último, los datos. Un archivo estructurado según el estándar RIFF siempre poseerá la estructura :

<ckID>,”RIFF”, (4 bytes)
<ckSize >, (4 bytes)
<ckData >, (4 bytes)

Definición del formato Wave

El formato Wave es un tipo de archivo estructurado según el estándar RIFF, dividido en dos chunks, que se denominan fragmento de formato y fragmento de datos. Es decir :

<nombre del tipo de archivo >,”WAVE”, (4 bytes)
<fragmento de formato> = fmt_.ck
<fragmento de datos> = data.ck

(Donde “_” significa “espacio”).

- Fragmento de formato

Es un chunk en el que se especifica el formato de los datos contenidos en el fragmento de datos. La estructura es la siguiente :

fmt*.ck

<ckID>,”fmt_”, (4 bytes)
<ckSize >, (4 bytes)
<ckData >, (4 bytes)

El nombre o ckID“fmt_” identifica al chunk como fragmento de formato.

La longitud ckSize se refiere a la longitud del fragmento de datos del fmt_.ck, que es invariablemente 16 bytes.

El fragmento de datos ckData indica los siguientes parámetros de formato:

- <etiqueta de formato>, (2 bytes), actualmente se utiliza el formato PCM (*Pulse Code Modulation*), de este dato depende la interpretación de los datos del data.ck. Dentro del entorno *Windows* se utiliza una constante llamada WAVE_FORMAT_PCM, que toma el valor “01”, para indicar el formato PCM.

- <número de canales>, (2 bytes), indica el número de canales de salida, tomará el valor “01” para ficheros mono y el valor “02” para estéreo.

- <número de*muestras por segundo>, (4 bytes), o frecuencia de muestreo a la que se reproducirá cada canal, y que está limitada a ciertos valores (11025 Hz, 22050 Hz y 44100 Hz) para las tarjetas SoundBlaster.

- <número de bytes por segundo>, (4 bytes), indica el número medio de bytes por segundo que se deben transmitir. Será igual a :

$$\text{Numero de bytes por segundo} = (\text{Numero de canales}) * (\text{Frecuencia}) * \frac{(\text{Numero de bits por muestra})}{8} \quad [4.1]$$

- <alineamiento de los bloques de datos>, (2 bytes), es el número de bytes que son necesarios para transcribir una muestra. Este dato es necesario, debido a que muchos programas de reproducción procesan un número fijo de bytes cada vez,*y deberán alinear los búferes dependiendo de los bytes utilizados por la muestra.

$$\text{Alineamiento de los bloques de datos} = (\text{Numero de canales}) * \frac{(\text{Numero de bits por muestra})}{8} \quad [4.2]$$

- <dato específico del formato>, (2 bytes), para el formato PCM es el número de bits que representan una muestra. Será igual a “1” para datos recogidos a 8 bits-mono, igual a “2” para 8 bits- estéreo y 16 bits-mono, e igual a “4” para 16 bits-estéreo.

- Fragmento de datos

Es el chunk que contiene los datos de sonido propiamente dicho. La estructura es la siguiente :

data.ck

<ckID>,”data”, (4 bytes)
<ckSize >, (4 bytes)
<ckData >, (4 bytes)

El nombre o ckID “data” identifica al chunk como fragmento de datos.

La longitud ckSize se refiere a la longitud del fragmento de datos del data.ck, que variará, lógicamente, según la longitud del fichero y la frecuencia de muestreo.

El fragmento de datos ckData se compondrá de palabras de un byte; en el caso de almacenar sonido con una resolución en amplitud de 16 bits, se dividirá cada uno de los datos de 16 bits en dos palabras de 8 bits; podría pensarse en utilizar palabras dobles, es decir, de 16 bits, con lo cual se agilizaría la lectura de este tipo de archivos, pero el estándar Wave no lo permite.

En resumen, un archivo Waveform presentaría la siguiente estructura :

“RIFF”
<longTotal - longitud de la cabecera>
“WAVE”
“fmt_”
<longitud del fmt_.ck>
<etiqueta de formato>
<nº de canales>
<frecuencia>
<nº de bytes por segundo>
<alineamientos por bloque>
<especificación del formato>
“data”
<longitud del data_.ck>

.
.
. } datos

La longitud de la cabecera excluye el ck.ID y el ckSide del RIFF.

Se realiza un ejemplo para un sonido muestreado a 11025 Hz, con una resolución de 16 bits, en mono.

RIFF
<longTotal - 36>
WAVE
fmt_
16
1
1
11025
22050
2
16
data
<long de los datos = longTotal - 44>

.
.
. } datos

Todos estos datos se guardarán, como se ha dicho anteriormente, como palabras de*8 bits, con lo que deberá especificarse también la forma en que son “partidas” las palabras de 16 bits y “re-ordenadas” para su almacenamiento como bytes.

El Formato BDA

Se trata de archivos que siguen un formato propio del software *Hypersignal* de la casa *Hyperception*, muy utilizado para el procesado digital de señales (DSP), permitiendo la adquisición en tiempo real de las señales y su procesamiento.

Los ficheros de ondas creados en el entorno *Hypersignal* contienen dos secciones: una cabecera con los parámetros de la señal y formada por 10 enteros de 16 bits y una segunda sección que contiene los datos de la señal, también enteros de 16 bits.

A continuación se describe la cabecera de un archivo en formato BDA.

AMPLITUDE	Primer parámetro / atributo
FRAMESIZE	
SAMPLING FREQUENCY, REM	
FFT ORDER	
NUMBER OF DATA VALUES, REM	
FRAME OVERLAP	
DATA TYPE	
PARAMETER / ATTRIBUTE #8	No reservado, disponible para el usuario
SAMPLING FREQUENCY, DIV	
NUMBER OF DATA VALUES, DIV	Último parámetro / atributo
DATA VALUE 0	Primer dato (valor) de la onda
DATA VALUE 1	
.....	
DATA VALUE N-1	Ultimo dato, donde N es Number of data values

A continuación se muestra una breve descripción de cada uno de estos parámetros donde indicamos entre paréntesis con T o F si el parámetro es requerido en el dominio del tiempo o dominio de la frecuencia.

- AMPLITUDE (T,F): Máximo valor absoluto de los datos de la señal.
- FRAMESIZE (T,F): Número de muestras de la señal.
- SAMPLING FREQUENCY, REM (T,F): La frecuencia de muestreo es el ratio al cual los datos son muestreados (en Hz) y se calcula así:

Sampling Frequency,Div * 32767 + Sampling Frequency,REM

donde Sampling Frequency,Div es el parámetro número nueve.

- FFT ORDER (F): La longitud de la FFT (Fast Fourier Transform) se calcula:

$$\text{FFT length}=2^{\text{FFT*ORDER}}$$

En las señales en el dominio del tiempo, este parámetro se encuentra desactivado (no se interpreta).

- NUMBER OF DATA VALUES, REM (T,F): El número de datos (valores) de la señal (enteros de 16 bits) se calcula:

  N=Number of Data Values, DIV * 16384 + Number of Data Values, REM

  donde Number of Data Values,DIV*es el parámetro nº 10.

- FRAME OVERLAP: Indica la cantidad de solapamiento de la señal. Debe tener un valor distinto de cero en el dominio de la frecuencia.

- DATA TYPE (T,F): Atributo del tipo de datos, los bits 15-0 (bit 15 es el bit mas significativo, bit 0 es el bit menos significativo) se usan para definir el tipo de datos que alberga la señal como sigue:

Time Domain Source (bits 2-0)	000	arbitrary time data
	001	autocorrelation output
	010	FIR or IIR filter output
	...	reserved for *Hypersignal* use
	111	
Complex data attribute (bit 3)	0	real-sampled data (no imaginary data present)
	1	complex-sampled data (Imaginary data present in file)
LPC Order (bits 8-4)	0	waveform is not LPC autocorrelation output
	1-31	value of LPC order
Window Type (bits 11-9)	000	rectangular window (i.e. none)
	001	Hamming window
	010	Hanning window
	011	Blackman window
	100	Bartlett window
	101	
	...	user defined
	111	
Reserved (bits 15-12)		Reserved for future *Hypersignal* functions

- PARAMETER / ATTRIBUTE #8Disponible para aplicaciones de uso específico.

- SAMPLING FREQUENCY, DIVLa frecuencia de muestreo es el ratio al cual los datos son muestreados (en Hz) y se calcula así:

  Sampling Frequency,Div * 32767 + Sampling Frequency,REM

  donde Sampling Frequency,Div es el parámetro nº 3

- NUMBER OF DATA VALUES, DIVEl número de datos (valores) de la señal (enteros de 16 bits) se calcula:

N=Number of Data Values, DIV * 16384 + Number of Data Values, REM

donde Number of Data Values,DIV*es el parámetro nº 5.

Todos los ficheros de ondas de *Hypersignal* tienen el mismo formato. La interpretación de los datos contenidos dependerán de la extensión asignada.

Las extensiones válidas son las siguientes:

TIM	Señal en Dominio del Tiempo
FRQ	Datos de frecuencia
PHZ	Datos de fase
PWR	Espectro de potencia
REA	Componente real
IMA	Componente imaginaria

En el caso de la Base de Datos Ahumada se ha optado por nombrar a los ficheros con la extensión BDA.