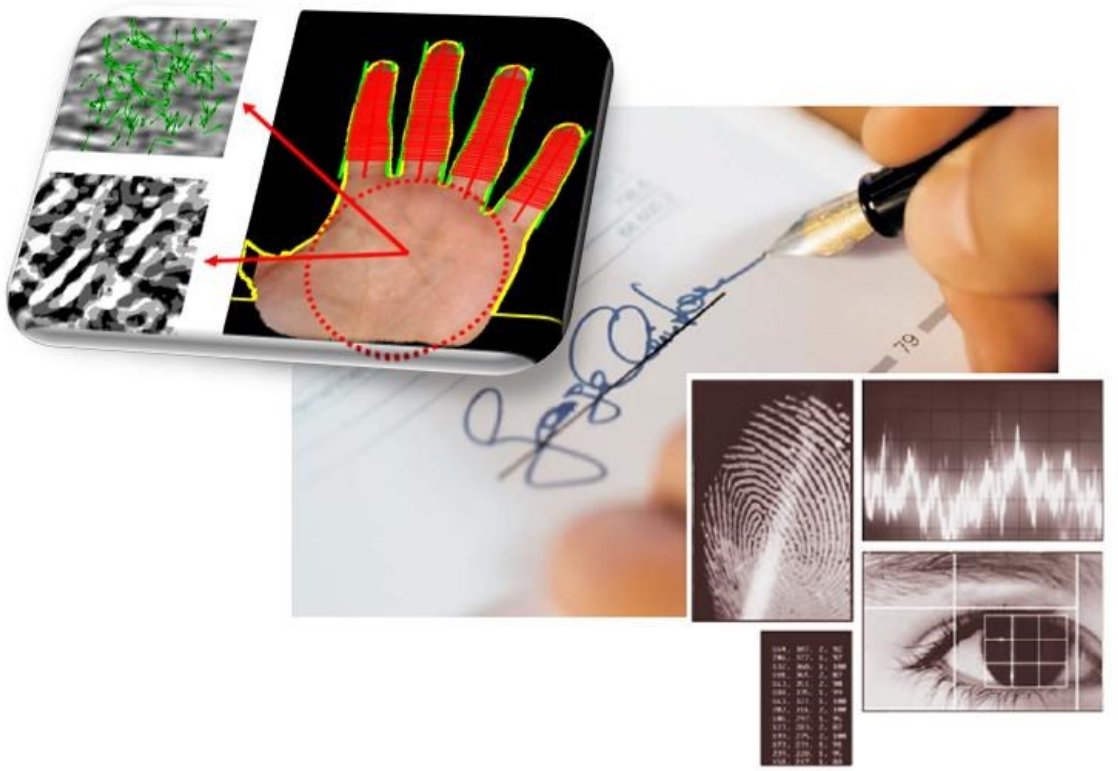
Universidad de Las Palmas de Gran Canaria
Escuela de Ingeniería de Telecomunicación y Electrónica

Procesado Digital de la Señal
para graduados en Telecomunicación

2ª edición



Autores:
Miguel Ángel Ferrer Ballester
Aythami Morales Moreno
Jesús Bernardino Alonso Hernández

IDeTIC®
Instituto para el Desarrollo Tecnológico y la Innovación en Comunicaciones

Prólogo

Este libro trata de ofrecer una primera aproximación al contenido y desarrollo de la asignatura *Procesado de la Señal* que se imparte en el segundo semestre del tercer curso (3B) del título del Grado en Ingeniería en tecnologías de la Telecomunicación. Forma parte del módulo de Tecnología Específica mención Sistemas de Telecomunicación y constituye, como asignatura única, la materia de Tratamiento de la señal. La segunda edición recoge las modificiones fruto de la experiencia del primer año de impartición de la asignatura. Concisamente se ha agrupado todo el procesado de imagen y clasificación en un capítulo final y se han añadido al final de cada capítulo algunos problemas.

Si se tuviera que dar una definición del título de la asignatura Procesado de la señal, podríamos decir que es el conjunto de técnicas para manipular muestras de señales normalmente continuas, con el fin de lograr determinados propósitos. Se han excluido de la anterior definición las señales estrictamente discretas debido a que en el ámbito de las comunicaciones la mayoría de las señales tienen naturaleza analógica.

Cabe resaltar que el Procesado de la Señal está cada vez más presente en innumerables aplicaciones, pero conviene tener presente que estas técnicas se encuentran normalmente en la base del funcionamiento de dichas aplicaciones, por lo que normalmente suelen pasar desapercibidas para el usuario. Para comprender esta afirmación piénsese en un sistema como la telefonía móvil celular digital; el que dicho sistema pueda funcionar es, en gran medida, debido al hecho de que existe un algoritmo eficiente de compresión de la señal vocal, a que existen técnicas eficientes de modulación digital cuyos moduladores se implementan de forma discreta la mayoría de las veces, y para lograr una calidad óptima hay que recurrir a sofisticados sistemas de ecualización automática implementados mediante técnicas de procesado digital de señal. Dichos aspectos pasan normalmente desapercibidos por el usuario.

El enfoque que se le quiere dar a la asignatura pretende ser doble:

- Aspectos matemáticos: La materia Procesado de la Señal tiene sólidas bases matemáticas. Dichas bases permitirían el planteamiento de una asignatura basada en la

formulación matemática, las demostraciones, etc. Sin embargo la asignatura pretende huir de dicha aproximación, y si bien la formulación matemática se hace a veces imprescindible. Se pretende desarrollar más la capacidad de razonar del alumno así como que conozca las herramientas de cálculo numérico que permiten en muchos casos prescindir de laboriosas soluciones analíticas.

- Aspectos tecnológicos: Muchas de las realizaciones prácticas de los conceptos que se ven en la asignatura precisan tecnología específica para ser llevadas a cabo. En la asignatura no se entra en aspectos tecnológicos particulares de ningún fabricante, en primer lugar por la rápida obsolescencia de estos conocimientos y en segundo lugar porque la gran diversidad de fabricantes existentes haría imposible estudiar el tema con mediana profundidad. Por contra, se presentan en abstracto aquellos aspectos de la tecnología que pueden influir en el diseño de algoritmos de procesado de señal, más en concreto los que hacen referencia al uso de aritmética de coma fija. Los aspectos tecnológicos son estudiados en otras asignaturas de la carrera, tales como Electrónica y Digital o Sistemas Digitales y Microprocesadores del primer cuatrimestre del mismo curso.

La presentación de la asignatura viene ilustrada con numerosos ejemplos de aplicaciones reales de las ideas vistas en la misma. Los problemas normalmente constituyen la excusa para el estudio de dichas aplicaciones de ingeniería.

Las Palmas de Gran Canaria a 8 de enero de 2013

Competencias

Las Competencias Básicas y Generales de la asignatura Procesado de la Señal, dentro del Perfil del Egresado del Título de Grado en Ingeniería en Tecnologías de la Telecomunicación, son las siguientes:

- CB-1: Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
- CB-2: Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
- CB-3: Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
- CB-4: Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
- CB-5: Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.
- CG-3: Capacidad para concebir, diseñar, desplegar, organizar y gestionar sistemas y servicios de telecomunicación en línea y radioeléctricos, infraestructuras de telecomunicación y sistemas de hogar digital.
- CG-4: Capacidad para diseñar e implementar sistemas de adquisición y procesado de señales.

Las Competencias Transversales de la asignatura Procesado de la Señal, dentro del Perfil del Egresado del Título de Grado de Ingeniería en Tecnología de la Telecomunicación, son las siguientes:

- CT-1: Comunicarse de forma adecuada y respetuosa con diferentes audiencias (clientes, colaboradores, promotores, agentes sociales, etc.), tanto en castellano como en

inglés, utilizando los soportes y vías de comunicación más apropiados (especialmente las nuevas tecnologías de la información y la comunicación) de modo que pueda llegar a comprender los intereses, necesidades y preocupaciones de las personas y organizaciones, así como expresar claramente el sentido de la misión que tiene encomendada y la forma en que puede contribuir, con sus competencias y conocimientos profesionales, a la satisfacción de esos intereses, necesidades y preocupaciones.

- CT-2: Cooperar con otras personas y organizaciones en la realización eficaz de funciones y tareas propias de su perfil profesional, desarrollando una actitud reflexiva sobre sus propias competencias y conocimientos profesionales y una actitud comprensiva y empática hacia las competencias y conocimientos de otros profesionales.
- CT-3: Contribuir a la mejora continua de su profesión así como de las organizaciones en las que desarrolla sus prácticas a través de la participación activa en procesos de investigación, desarrollo e innovación.
- CT-4: Comprometerse activamente en el desarrollo de prácticas profesionales respetuosas con los derechos humanos así como con las normas éticas propias de su ámbito profesional para generar confianza en los beneficiarios de su profesión y obtener la legitimidad y la autoridad que la sociedad le reconoce.
- CT-5: Participar activamente en la integración multicultural que favorezca el pleno desarrollo humano, la convivencia y la justicia social

Las Competencias Específicas Comunes de la asignatura Procesado de la Señal, dentro del Perfil del Egresado del Título de Grado de Ingeniería en Tecnología de la Telecomunicación, son las siguientes

- CR-1: Capacidad para aprender de manera autónoma nuevos conocimientos y técnicas adecuados para la concepción, el desarrollo o la explotación de sistemas y servicios de telecomunicación.
- CR-2: Capacidad de utilizar aplicaciones de comunicación e informáticas (ofimáticas, bases de datos, cálculo avanzado, gestión de proyectos, visualización, etc.) para apoyar el desarrollo y explotación de redes, servicios y aplicaciones de telecomunicación y electrónica.

- CR-3: Capacidad para utilizar herramientas informáticas de búsqueda de recursos bibliográficos o de información relacionada con las telecomunicaciones y la electrónica.

Las Competencias Específicas de la asignatura Procesado de la Señal, dentro del Perfil del Egresado del Título de Grado de Ingeniería en Tecnología de la Telecomunicación en la mención de Sistemas de Telecomunicación, son las siguientes

- CEST-1: Capacidad para construir, explotar y gestionar las redes, servicios, procesos y aplicaciones de telecomunicaciones, entendidas éstas como sistemas de captación, transporte, representación, procesado, almacenamiento, gestión y presentación de información multimedia, desde el punto de vista de los sistemas de transmisión.
- CEST-6: Capacidad para analizar, codificar, procesar y transmitir información multimedia empleando técnicas de procesado analógico y digital de señal.

Objetivos

Los objetivos de la asignatura podrían enunciarse como sigue siguiendo el plan de estudios aprobado:

OBJ-1: Que el alumno domine los conceptos subyacentes en las dualidades Continuo/Discreto y Tiempo/Frecuencia

- Continuo/Discreto: es importante que el alumno comprenda que cualquier manipulación que se pretenda realizar sobre una señal continua, puede ser realizada tanto de forma continua como discreta. La implementación de forma discreta puede suponer, en muchos casos, ventajas. Es crucial comprender perfectamente la relación entre las operaciones realizadas sobre las muestras y el efecto que provocan en su señal analógica asociada. Aunque existen sistemas totalmente analógicos, no existen sistemas totalmente digitales, siendo misión del ingeniero el determinar en cada situación cuál es el punto más ventajoso para realizar la digitalización de la señal, teniendo en cuenta aspectos como calidad de la señal, disponibilidad de tecnología, coste etc.
- Tiempo/Frecuencia: Toda señal admite dos representaciones una temporal y otra frecuencial. También cualquier sistema lineal e invariante puede analizarse en el dominio del tiempo y de la frecuencia. Se pretende igualmente que el alumno razone con naturalidad en ambos dominios aprendiendo a distinguir en cuál de ellos resulta más sencilla la interpretación de señales o algoritmos.

OBJ-2: Entender cómo funcionan los sistemas o subsistemas que incorporan procesado digital de señal con el fin de integrarlos en sistemas complejos.

OBJ-3: Implementar sistemas de procesado discreto de señal.

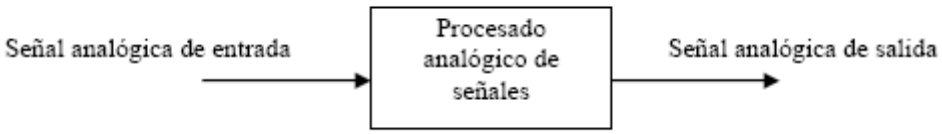
Contenidos

Capítulo I. Introducción al procesado digital de la señal

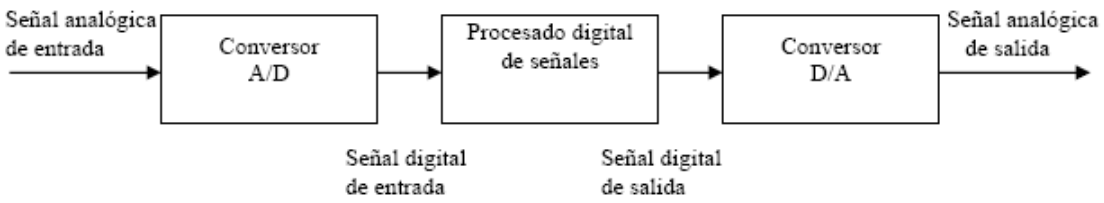
El objetivo de éste capítulo es capacitar al alumno para aplicar el procesado digital de señales discretas definiendo secuencias, implementando diagramas de bloques sencillos, calculando transformadas de Fourier, y aplicando sus propiedades.

1.1 Diagrama de bloques del procesado digital de señales continúas

La mayor parte de las señales que aparecen en los ámbitos de la ciencia y la ingeniería son de naturaleza analógica, es decir, las señales son funciones de una variable continua, como el tiempo o el espacio y normalmente toman valores en un rango continuo. Tales señales pueden ser procesadas directamente por sistemas analógicos adecuados (como filtros o analizadores de frecuencia) o multiplicadores de frecuencia con el propósito de cambiar sus características o extraer cualquier información deseada. En tal caso, decimos que la señal ha sido procesada directamente de forma analógica, como podemos ver en la figura tanto la entrada como la salida están de forma analógica.



El procesado digital de señales proporciona un método alternativo para procesar una señal analógica, como se ilustra en la figura siguiente:



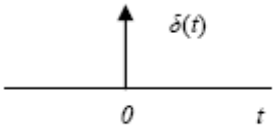
Para realizar el procesado digitalmente se necesita un interfaz entre al señal analógica y el procesador digital. Este interfaz se denomina conversor analógico-digital (A/D). La salida del conversor analógico-digital es una señal adecuada como entrada al procesador digital. El procesador digital de señales puede ser un ordenador digital programable o un pequeño microprocesador programado para realizar las operaciones deseadas sobre la señal de entrada. Las máquinas programables proporcionan la flexibilidad de cambiar las operaciones de procesado de señales mediante un cambio del software. En consecuencia, los procesadores de señales programables son de uso muy frecuente. Por otro lado, cuando las operaciones de procesado de señales están bien definidas, se puede optimizar la implementación cableada de las operaciones, resultando un procesador más barato y, habitualmente, más rápido que su equivalente programable. En aplicaciones donde la salida digital del procesador digital de señales se ha de entregar en forma analógica, como en comunicaciones digitales, debemos proporcionar otro interfaz desde el dominio digital al analógico. Tal interfaz se denomina conversor digital-analógico (D/A). De este modo, la señal se entrega al usuario de forma analógica. No obstante, existen otras aplicaciones prácticas que requieren análisis de señales en las que la información deseada se encuentra en formato digital y no se requiere ningún conversor D/A. Por ejemplo, en el procesado digital de señales radar, la información extraída de la señal radar, como la posición de la nave y su velocidad, se pueden imprimir directamente sobre papel. En este caso no hay necesidad de conversor D/A.

1.2 Descripción de señales continuas: transformada de Fourier y Laplace

Una técnica clásica de trabajo en procesado de señal es descomponer la señal en combinación lineal de señales sencillas o básicas. Así bastará conocer cómo responde el sistema a dichas señales básicas para conocer la respuesta del sistema a cualquier entrada descompuesta en dichas señales básicas.

Las señales básicas con las que se trabajan son dos: la delta (o delta de Dirac) y la exponencial compleja.

La delta se define gráficamente como



Una de las propiedades que hace más interesante a la delta es que cualquier señal puede descomponerse como suma de deltas según la expresión:

$$x(t) = \int x(\tau)\delta(t-\tau)d\tau = x(t) * \delta(t) \qquad \text{[Ec.1]}$$

Exponencial compleja

$$e^{j\Omega_o t} = \cos\Omega_o t + j\, sen\Omega_o t \qquad \text{[Ec.2]}$$

Cuya principal propiedad es la periodicidad en el tiempo. Para calcular su periodo *T*:

$$e^{j\Omega_o t} = e^{j\Omega_o (t+T)} = e^{j\Omega_o t} \cdot e^{j\Omega_o T} \qquad \text{[Ec.3]}$$

para que ambos lados sean iguales se debe cumplir $e^{j\Omega_o T} = 1$

de donde: $\Omega_o T = 2\pi k \quad \Rightarrow \quad T = \frac{2\pi k}{\Omega_o}$. Al menor periodo T_0 con k=1 se le denomina periodo fundamental.

Exponenciales armónicamente relacionadas son aquellas que tiene un periodo común $T = \frac{2\pi}{\Omega_o}$. Su expresión es $\Theta_k(t) = e^{jk\Omega_o t}$.

Desarrollo en serie de Fourier

Una señal $x(t)$ periódica de periodo T que cumpla:

1. Tener un número finito de máximos y mínimos.
2. Tener un número finito de discontinuidades.
3. Ser absolutamente integrable.

puede representarse como una combinación lineal de exponenciales armónicamente relacionadas con periodo común *T*, esto es:

$$x(t) = \sum_{k=-\infty}^{+\infty} a_k \cdot e^{jk\Omega_o t} \qquad \text{[Ec.4]}$$

donde los coeficientes se calculan:

$$a_k = \frac{1}{T}\int_T x(t) \cdot e^{-jk\Omega_o t} dt \qquad \text{[Ec.5]}$$

denominándose los a_k coeficientes del desarrollo en serie de Fourier.

Para el caso de $x(t)$ no periódica, supongo T_0=∞, obteniendo la transformada de Fourier:

Formula de análisis: $$X(\Omega) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-j\Omega t} dt \qquad \text{[Ec.6]}$$

Fórmula de síntesis : $$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\Omega) \cdot e^{j\Omega t} d\Omega \qquad \text{[Ec.7]}$$

La transformada de Fourier sería la envolvente de los a_k.

El cálculo de la transformada de Fourier de una señal $x(t)$ periódica de periodo T se realiza aplicando la siguiente formula:

$$X(\Omega) = \sum_{k=-\infty}^{+\infty} 2\pi a_k \delta(\Omega - k\Omega_o) \qquad \text{[Ec.8]}$$

siendo $\Omega_o = \frac{2\pi}{T}$ y a_k los coeficientes del desarrollo en serie de Fourier de $x(t)$

Propiedades de la Transformada de Fourier

1. Linealidad

Si $x_3(t) = \alpha x_1(t) + \beta x_2(t)$

entonces $X_3(\Omega) = \alpha X_1(\Omega) + \beta X_2(\Omega)$

2. Simetría

Si $x(t)$ es real $\Rightarrow$ $X(-\Omega) = X^*(\Omega)$

Esto es: su módulo cumple: $|X(\Omega)| = |X(-\Omega)|$

y su fase: $\angle X(\Omega) = -\angle X(-\Omega)$

3. Desplazamiento

si $x(t) \rightarrow X(\Omega)$

entonces $x(t - t_0) \rightarrow e^{-j\Omega t_0} \cdot X(\Omega)$

de donde se deduce que el módulo de la T.F. es invariante frente al desplazamiento, sólo cambia la fase pues $\left|e^{-j\Omega to}\right| \cdot \left|X(\Omega)\right| = \left|X(\Omega)\right|$

4. Parseval

$$\int_{-\infty}^{\infty} \left|x(t)\right|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left|X(\Omega)\right|^2 d\Omega \qquad \text{[Ec.9]}$$

La transformada de Fourier no altera la energía de la señal.

5. Convolución

si $x_3(t) = x_1(t) * x_2(t) \rightarrow X_3(\Omega) = X_1(\Omega) X_2(\Omega)$

6. Modulación

Si $x_3(t) = x_1(t) \cdot x_2(t) \rightarrow X_3(\Omega) = \frac{1}{2\pi} X_1(\Omega) * X_2(\Omega)$

La anterior transformada se puede generalizar en la Transformada de Laplace (T.L.)

$$X(s) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-st} dt \qquad \text{[Ec.10]}$$

donde $s = \sigma + j\Omega$, con propiedades similares.

Téngase en cuenta que si σ=0 $\Rightarrow$ $X(s)\,\big|_{\sigma=0} = X(\Omega)$, es decir, para σ=0, la T.F. y la T.L. coinciden.

1.3 Descripción de señales discretas: señales básicas, transformada de Fourier y Z

Una señal discreta se puede definir como una secuencia o conjunto de números ordenados por un índice.

Es importante hacer notar que aquí lo que hay entre muestra y muestra no es ni tiempo ni espacio ni cualquier otra magnitud; lo importante es el orden entre las muestras. Al igual que hicimos con las señales continuas, las secuencias se describen como combinación lineal de señales básicas, que suelen ser dos:

Impulso: $\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & resto \end{cases}$ de donde $x[n] = \sum_{k=-\infty}^{+\infty} x[k]\delta[n-k]$

Exponencial compleja

$$e^{j\omega_o n} = \cos \omega_o n + j\, sen \omega_o n \qquad \text{[Ec.11]}$$

tiene dos tipos de periodicidad, en ω y en n (mientras que en el dominio continuo sólo eran periódicas en tiempo).

Periodicidad en pulsación: las exponenciales complejas discretas son periódicas en pulsación con periodo 2π, pues $e^{j\omega n} = e^{j(\omega + 2\pi k)n} = e^{j\omega n} e^{j2\pi k n} = e^{j\omega n}$ ya que $e^{j2\pi k n} = 1$.

Así, al contrario que en el dominio continuo, aumentar la pulsación no implica aumentar la velocidad de oscilación o variabilidad de la señal. La máxima variabilidad se tiene en $\omega = \pm\pi, \pm 3\pi, \pm 5\pi ...$

Esto explica la limitación de ancho de banda en procesado digital de la señal y los efectos de solapamiento espectral en muestreo.

Periodicidad en n: a pesar que en el dominio continuo todas las exponenciales complejas son periódicas en *t*, no todas las exponenciales complejas discretas son periódicas en *n*, pues

$$e^{j\omega n} = e^{j\omega(n+N)} = e^{j\omega n}e^{j\omega N} = e^{j\omega n} \rightarrow e^{j\omega N} = 1 \rightarrow \omega N = 2\pi k$$

Así, sólo son periódicas en *n* con periodo *N* las exponenciales con pulsación $\omega = \frac{2\pi k}{N}$.

Exponenciales armónicamente relacionadas son aquellas que tienen un periodo común *N* en *n*. Su expresión es:

$$\Theta_k[n] = e^{j\frac{2\pi kn}{N}} \quad k=0,1,2,...,N\text{-}1 \qquad \text{[Ec.12]}$$

Debido a la periodicidad en pulsación, sólo hay *N* exponenciales complejas discretas armónicamente relacionadas.

Series discretas de Fourier

Una secuencia *x*[*n*] periódica de periodo *N*, absolutamente sumable en un periodo, se puede representar como combinación lineal de exponenciales armónicamente relacionadas:

$$x[n] = \sum_{k=0}^{N-1} a_k \cdot e^{j\frac{2\pi kn}{N}} \qquad \text{[Ec.13]}$$

Siendo los coeficientes del desarrollo en serie de Fourier :

$$a_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{N}} \qquad \text{[Ec.14]}$$

Para señales no periódicas, se halla el límite $N \rightarrow \infty$ obteniendo la transformada de Fourier, cuyas fórmulas son:

$$\textit{Formula de análisis:} \qquad X(\omega) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n} \qquad \text{[Ec.15]}$$

dónde $X(\omega) \in \text{Complejos} : X(\omega) = \begin{cases} \textit{módulo} & |X(\omega)| \\ \textit{fase} & \angle X(\omega) \end{cases}$

Formula de síntesis: $$x[n]=\frac{1}{2\pi}\int_{2\pi}X(\omega)\cdot e^{j\omega n}d\omega \qquad \text{[Ec.16]}$$

Para calcular la transformada de Fourier de una señal $x[n]$ periódica de periodo N, se calcula

$$X(\omega)=\sum_{k=-\infty}^{+\infty}2\pi a_k\delta(\omega-\frac{2\pi k}{N}) \qquad \text{[Ec.17]}$$

siendo los a_k los coeficientes del desarrollo en serie de Fourier de $x[n]$.

Propiedades de la Transformada de Fourier

1. $X(\omega)=\sum_{n=-\infty}^{+\infty}x[n]\cdot e^{-j\omega n}$ *periódica* (2π)

2. Linealidad.

Si $x_3[n]=\alpha x_1[n]+\beta x_2[n]$

entonces $X_3(\omega)=\alpha X_1(\omega)+\beta X_2(\omega)$

3. Simetría.

a) si $x[n]$ real → X(ω) = X*(-ω): módulo par y fase impar

b) si $x[n]$ par → X(ω) real

c) si $x[n]$ impar → X(ω) imaginaria

4. Desplazamiento.

$$x[n-n_o]\longrightarrow e^{-j\omega n_o}X(\omega) \qquad \text{[Ec.18]}$$

$$e^{j\omega_o n}\cdot x[n]\longrightarrow X(\omega-\omega_o) \qquad \text{[Ec.19]}$$

5. Parseval

$$\underbrace{\sum_{n=-\infty}^{n=\infty}|x[n]|^2}_{Energía\ en\ n}=\underbrace{\frac{1}{2\pi}\int_{2\pi}|X(\omega)|^2\,d\omega}_{Energía\ en\ \omega} \qquad \text{[Ec.20]}$$

La energía de la señal se conserva al hallar su T.F.

6. Convolución

$$\text{Si } x[n]=x_1[n]*x_2[n]\rightarrow X(\omega)=X_1(\omega)\cdot X_2(\omega) \qquad \text{[Ec.21]}$$

7. Modulación

Convolución circular de dos señales continuas en un período:

$$x[n]=x_1[n]\cdot x_2[n]\rightarrow X(\omega)=\frac{1}{2\pi}\int_{2\pi}X_1(\theta)\cdot X_2(\omega-\theta)d\theta \qquad \text{[Ec.22]}$$

8. Correlación

$$x[n]=\sum_{m=-\infty}^{m=+\infty}x_1[m+n]\cdot x_2[m]\longrightarrow X(\omega)=X_1(\omega)\cdot X^*{}_2(\omega) \qquad \text{[Ec.23]}$$

9. Fenómeno de Gibbs

Si $X(\omega)=\sum_{n=-\infty}^{n=+\infty}x[n]\cdot e^{-j\omega n}$ y $X_M(\omega)=\sum_{n=-M}^{n=+M}x[n]\cdot e^{-j\omega n}$

Entonces $\lim_{M\rightarrow\infty}\int_{2\pi}|X(\omega)-X_M(\omega)|^2 d\omega\longrightarrow 0$

Cuanto mayor sea *M*, menor será la diferencia entre las energías de ambas señales.

La anterior transformada puede generalizarse en la transformada *Z* definida como

$X(z)=\sum_{n=-\infty}^{n=+\infty}x[n]\,z^{-n}$ donde $z=re^{j\omega}$ $\rightarrow$ por tanto $X(\omega)=X(z)\big|_{r=1}$

sus propiedades son similares a las de la trasformada de Fourier.

1.4 La transformada discreta de Fourier: definición, práctica y propiedades

La transformada de Fourier es una herramienta de gran utilidad, pero el hecho de ser $X(\omega)$ continua dificulta su implementación en un ordenador. Por ello que se recurre entonces a la DFT, con los inconvenientes que a continuación se enumeran:

Se muestrea la TF en $\omega=2\pi k/N$, $k=0,1,...,N-1$

Transformada de Fourier		equivale a tomar *N* muestras de $X(\omega)$		
secuencia $x[n]$	$\longleftrightarrow$	$X(\omega)$	$\longrightarrow$	$X[k]=X(\omega)\big\|_{\omega=\frac{2\pi k}{N}}$

A $X[k]$ se le conoce como transformada discreta de Fourier (DFT)

Aquí surgen dos cuestiones:

Primera cuestión: ¿Cómo puedo obtener $X[k]$ a partir de $x[n]$?

$$X[k]=X(\omega)\big|_{\omega=\frac{2\pi k}{N}}=\sum_{n=-\infty}^{n=\infty}x[n]\,e^{-j\omega n}\Bigg|_{\omega=\frac{2\pi k}{N}}\Rightarrow X[k]=\sum_{n=-\infty}^{+\infty}x[n]\cdot e^{-\frac{j2\pi kn}{N}}$$

Que es la fórmula de análisis de la DFT.

Como $X(\omega)$ es periódica (2π), $X[k]$ es periódica de período N (k=0,....,N-1), esto es:

$X[k] = X[k+N]$.

Téngase en cuenta que la fórmula de análisis de la DFT es similar a la fórmula de análisis del desarrollo en serie de Fourier de señales periódicas de periodo N, cuya expresión se recuerda es: $a_k=\frac{1}{N}\sum_{n=0}^{N-1}\hat{x}[n]e^{-j\frac{2\pi kn}{N}}$.

La diferencia radica en los límites del sumatorio. Si la $x[n]$ es finita de longitud N ambas expresiones coinciden y $X[k]=Na_k$. Y, el hecho de que los a_k representen una señal periódica induce a pensar que la DFT no representa a la $x[n]$ sino a una versión periódica de $x[n]$ a la que denominaremos $\hat{x}[n]$.

Segunda cuestión: ¿Cómo puedo recuperar $x[n]$ a partir de $X[k]$?

Si obtenemos $X[k]$ mediante la fórmula de análisis del desarrollo en serie de Fourier, la vuelta al dominio n se realizará mediante la fórmula de síntesis del desarrollo en serie de Fourier.

$$\hat{x}[n]=\sum_{k=0}^{N-1}a_k\cdot e^{\frac{j2\pi kn}{N}}=\frac{1}{N}\sum_{k=0}^{N-1}X[k]\cdot e^{\frac{j2\pi kn}{N}} \qquad \text{[Ec.24]}$$

Otra forma de llegar al mismo resultado es discretizando la fórmula de síntesis de la transformada de Fourier: $x[n]=\frac{1}{2\pi}\int_{2\pi}X(\omega)\cdot e^{j\omega n}d\omega$

Teniendo en cuenta que $\hat{x}[n]$ es el resultado de una suma de N señales periódicas de periodo N, $\hat{x}[n]$ será una señal periódica de periodo N. Por tanto, el resultado de la anterior expresión no es $x[n]$ sino una señal $\hat{x}[n]$ periódica de período N.

La relación entre $x[n]$ y $\hat{x}[n]$ se puede obtener:

$$\hat{x}[n]=\frac{1}{N}\sum_{k=0}^{k=N-1}X[k]\;\;e^{j\frac{2\pi kn}{N}}=\frac{1}{N}\sum_{k=0}^{k=N-1}\left[\sum_{m=-\infty}^{m=+\infty}x[m]e^{-j\frac{2\pi km}{N}}\right]e^{j\frac{2\pi kn}{N}}$$

reordeno:

$$\hat{x}[n]=\sum_{m=-\infty}^{m=+\infty}x[m]\frac{1}{N}\sum_{k=0}^{k=N-1}e^{j\frac{2\pi kn}{N}}e^{-j\frac{2\pi km}{N}}=\sum_{m=-\infty}^{m=+\infty}x[m]\frac{1}{N}\sum_{k=0}^{k=N-1}e^{j\frac{2\pi k(n-m)}{N}}$$

lo cual es la convolución de dos secuencias ($x[n]*y[n]=\sum_{m=-\infty}^{m=+\infty}x[m]y[n-m]$)

$$\hat{x}[n]=x[n]*\frac{1}{N}\sum_{k=0}^{k=N-1}e^{j\frac{2\pi kn}{N}}$$

y como: $$\sum_{r=-\infty}^{r=+\infty}\delta[n+rN]=\frac{1}{N}\sum_{k=0}^{k=N-1}e^{j\frac{2\pi kn}{N}}$$

se tiene:

$$\hat{x}[n]=x[n]*\sum_{r=-\infty}^{r=+\infty}\delta[n+rN]=\sum_{r=-\infty}^{r=+\infty}x[n+rN]$$

donde se observa cómo $\hat{x}[n]$ es una versión periodificada con periodo N de $x[n]$, lo cual corresponde a aplicar una fórmula para señales periódicas (fórmula de la DFT similar a la fórmula de análisis del desarrollo en serie de Fourier) a una señal no periódica.

¿Se puede obtener $x[n]$ a partir de la recuperada $\hat{x}[n]$?

Respuesta: $x[n]$ de longitud finita M se podrá recuperar a partir de $\hat{x}[n]$ sólo si el número de muestras tomado de la transformada de Fourier N es mayor o igual que la longitud de la señal $x[n]$: $N \geq M$ (aunque un valor de N mayor a M no aporta nada en la recuperación de $x[n]$, por motivos computacionales puede ser conveniente), ya que en este caso $\hat{x}[n]$ es una extensión periódica de $x[n]$ (no hay solapamiento), esto es, $\hat{x}[n]$ es una secuencia periódica de periodo N cuyo periodo es $x[n]$ más los ceros necesarios hasta las N muestras del periodo.

En cambio si $N<M$, el periodo de N muestras de $\hat{x}[n]$ no corresponderá con las M muestras de $x[n]$ y no será posible recuperar $x[n]$.

Por esta razón la fórmula de análisis de la DFT queda:

$$X[k]=\sum_{n=0}^{M-1}x[n]\,e^{-j\frac{2\pi kn}{N}} \quad k=0,1,2,..,N, \text{ y } N\geq M$$

donde M es la longitud de $x[n]$ y N es el número de muestras de la transformada de Fourier.

Apuntes prácticos

El cálculo de DFTs e IDFTs en Matlab se realiza como sigue:

```
x=[1 2 3 4] %se define la señal x[n]
M=length(x);
X=fft(x)      % calculo la DFT de x[n]-> X[k]
x=ifft(X)     % calculo la IDFT de X[k]-> x[n]
% Para utilizar un N≥M se hace
N=6;
X=fft(x,N);
x=ifft(X); % se obtiene x=[1 2 3 4 0 0];
% Para utilizar N<M es necesario
N=2;
xs=x(1:2)+x(3:4)
X=fft(xs);
% o calcular N=múltiplo de N>M y muestrear
X=fft(x,2*N);
X=X(1:2:end);
```

Nota sobre la fft

Si se hace X=*fft*(x,N) la función supone que $x(1)$ es el valor de $x[n]$ en n=0, y $X(1)$ corresponde a k=0 .

Si queremos $X[k]$ entre k=–N/2 y k=N/2-1 (caso N par) o k=-(N-1)/2 y k=(N-1)/2 (caso N impar) hay que hacer X= *fftshift*(*fft*(x,N)).

Nota: Muestrear la transformada $X(\omega)$ de Fourier N veces obteniendo la DFT $X[k]$ de $x[n]$ de longitud M<N, equivale a calcular la transformada de Fourier $X_e(\omega)$ de la secuencia extendida $x_e[n]$ obtenida al rellenar $x[n]$ con ceros hasta hacerla de longitud N. A esta operación de relleno con ceros se le llama zero padding. La función de Matlab realiza zero padding.

Ejemplo, tomando las señales del ejemplo anterior:

```
N=5               %número de muestras de la DFT
x1=[1 2 1];
M=length(x1)      %N>M;
x1e=[1 2 1 zeros(1,N-M)];
% DFT muestreando N veces
X=fft(x1,N)
% DFT de la secuencia extendida muestreando N veces
Xe=fft(x1e)
% Xe debe ser igual a X
```

Como se ha comentado anteriormente, tratamos de trabajar con la T.F., pero como un microprocesador no puede trabajar con señales continuas, hacemos una aproximación, y

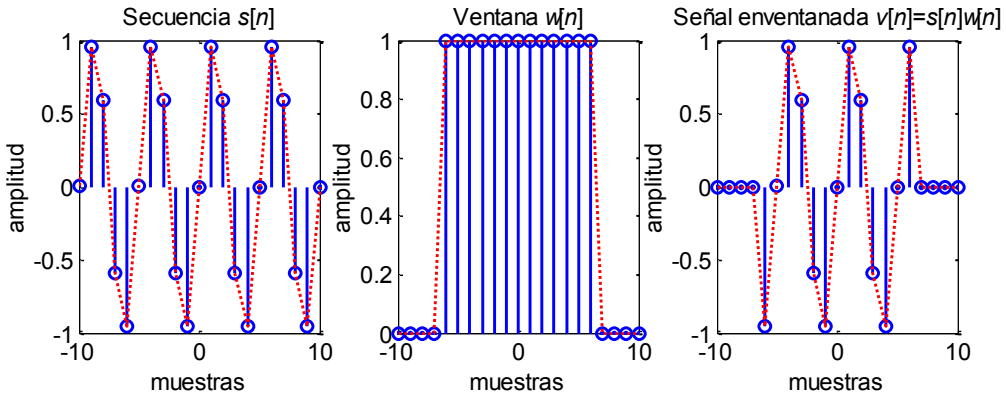
calculamos muestras de la T.F: estas muestras sólo representan a la T.F. si la señal es finita. Si en lugar de tener una señal finita tengo una señal infinita habría que hacer una segunda aproximación:

Supongamos que quiero calcular la T.F. de un coseno. Como se trata de una señal infinita se realiza el siguiente procedimiento:

Sea la señal $s[n] = \cos(\omega_0 n) \rightarrow S(\omega) = \pi(\delta(\omega - \omega_0) + \delta(\omega + \omega_0))$

1- Hago la señal $s[n]$ finita multiplicándola por una ventana $w[n]$ de longitud M:

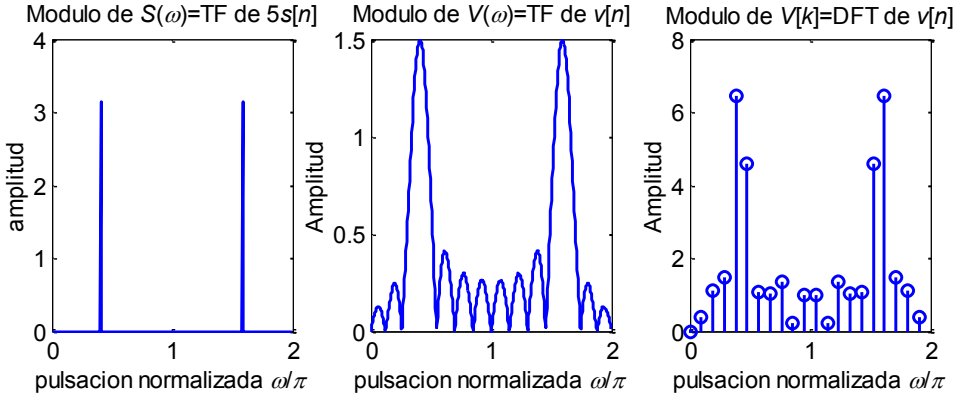
$$v[n] = w[n] \cdot s[n] \rightarrow V(\omega) = \frac{1}{2\pi} W(\omega) * S(\omega) = \frac{1}{2}(W(\omega - \omega_0) + W(\omega + \omega_0))$$



Así, suponiendo que $w[n]$ es rectangular con transformada de Fourier igual a una *sinc,* la primera aproximación cambia las dos deltas de la transformada de Fourier de $s[n]$ por dos *sinc* centradas en la frecuencia de la sinusoide. Cuanto más se aproxime la transformada de Fourier de la ventana a una delta, mejor será la aproximación.

2- Calculo la DFT de la señal finita que proviene de la señal infinita con $N \geq M$

$$V[k] = V(\omega)\big|_{\omega = \frac{2\pi k}{N}}$$



Y el resultado final son muestras de las dos *sinc* centradas en la frecuencia de la sinusoide. Por tanto la ventana provoca: 1) Ensanchamiento del espectro: pérdida de resolución y 2) Fugas de energía.

Obsérvese que si la frecuencia de la sinusoide puede escribirse como $\omega_0 = \frac{2\pi k}{N}$, y la DFT toma N=M muestras, entonces estoy muestreando en los ceros y en el pico de las *sinc*, así la apariencia en pantalla es de dos deltas.

Propiedades de la DFT a resaltar

1. *Linealidad.*

$x_1[n] \longleftrightarrow X_1[k]$

$x_2[n] \longleftrightarrow X_2[k]$

$ax_1[n] + bx_2[n] \longleftrightarrow aX_1[k] + bX_2[k]$; con a y b constante complejas

2. *Desplazamiento circular.*

Si en la transformada de Fourier se cumple

$si\ x[n] \to X(\omega)$ entonces $e^{-j\omega m} X(\omega) \to x[n-m]$

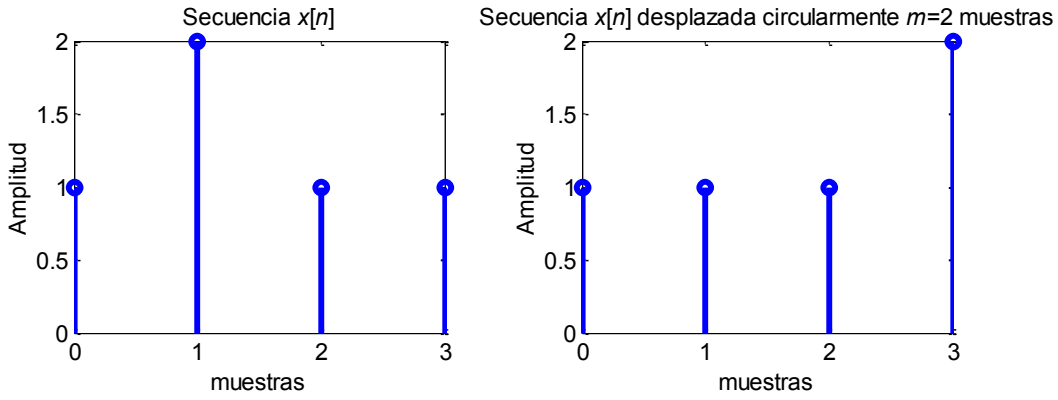
En la transformada discreta de Fourier se tiene que

$si\ x[n] \to X[k]$ entonces $e^{-j\frac{2\pi km}{N}} X[k] \to \hat{x}[n-m] = x[(n-m)_N]$

donde $x[(n-m)_N]$ indica un desplazamiento circular o un desplazamiento de $x[n]$ periodificada con periodo N. Notar que cualquier operación sobre $X[k]$ afectará, no a $x[n]$, si no a $\hat{x}[n]$

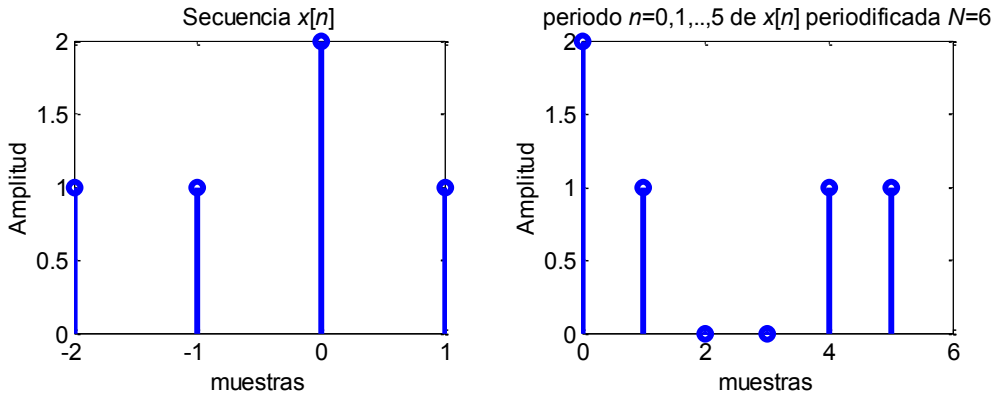
Ejemplo 1*:* calcular la DFT de $x[n]$, calcular $e^{-j\frac{2\pi k}{N}m} X[k]$ con m=2, y observar el efecto de dicho desplazamiento circular.

```
n=[0 1 2 3];
x=[1 2 1 1];
M=length(x);
N=M;
m=2;
k=[0:N-1];
X=fft(x,N);
E=exp(-j*2*pi*k*m/N);
Xd=E.*X;
xd=ifft(Xd);
```



Ejemplo 2: Calcular la DFT de la secuencia $x[n]$ con N=6

```
n=[-2 -1 0 1];
x=[ 1  1 2 1];
xc=[2 1 0 0 1 1];
N=6;
X=fft(xc,N);
% Otro método para obtener el mismo resultado es
m=-2;k=[0:N-1];
E=exp(-j*2*pi*k*m/N)
X=fft(x,6).*E;
```



Igualmente ocurre con la propiedad inversa $DFT(e^{-j\frac{2\pi l}{N}n}x[n]) = X[(k-l)_N]$

3. Dualidad.

$$x[n] \longleftrightarrow X\ [k]$$

$$X[n] \longleftrightarrow \frac{1}{N} x \cdot [(-k)_N] \quad , \quad 0 \le k \le N-1 \qquad \text{[Ec.25]}$$

4. Convolución circular

La propiedad se puede expresar como:

$$x_3[n] = x_1[n] * x_2[n] \xrightarrow{\text{Paso al dominio de de Fourier}} X_3(\omega) = X_1(\omega)X_2(\omega)$$

$$\downarrow \text{N muestras}$$

$$\hat{x}_3[n] = \sum_{r=-\infty}^{r=\infty} x_3[n+rN] \xleftarrow{\text{Paso al dominion}} X_3[k] = X_1[k]X_2[k]$$

y se puede demostrar que:

$$\hat{x}_3[n] = \sum_{r=-\infty}^{r=\infty} x_3[n+rN] = \sum_{m=0}^{N-1} x_1[m]x_2[(n-m)_N] = x_1[n] \otimes x_2[n]$$

que es la expresión de la convolución circular. El producto de las DFTs de dos secuencias es equivalente a la convolución circular de las dos secuencias en el dominio *n*. Notar que la convolución circular es diferente a la convolución lineal, cuya fórmula se recuerda es:

$$x_1[n] * x_2[n] = \sum_{m=-\infty}^{m=+\infty} x_1[m] \cdot x_2[n-m]$$

Para ver clarificar la relación entre ambas, véase el siguiente ejemplo con N=3:

```
x1=[1 2 1];
x2= [3 1 2];
L=length(x1);
P=length(x2);
% convolución lineal
x3lineal=conv(x1,x2);
%convolución circular
x3circular=ifft(fft(x1).*fft(x2));
```

Los resultados son:

```
x3lineal=[3 7 7 5 2]; %longitud L+P-1
x3circular=[8 9 7]; %longitud L=P
```

donde se cumple que $x3circular[n] = \sum_{r=-\infty}^{r=-\infty} x3lineal[n+rN]$ pues:

```
x3circular=x3lineal(1:3)+[x3lineal(4:5) 0]
```

¿Cómo puedo obtener el resultado de la convolución lineal haciendo un convolución circular?: como $x_3[n]$, convolución lineal de $x_1[n]$ (de longitud L) y $x_2[n]$ (de longitud P), tiene longitud $L+P-1$, su DFT $X_3[k]$ tendrá que hacerse con $L+P-1$ muestras para que un periodo de la $IDFT\{X_3[k]\}$ coincida con $x_3[n]$. Puesto que $X_3[k]=X_1[k]X_2[k]$, esto implica calcular las DFT $X_1[k]$ y $X_2[k]$ de $x_1[n]$ y $x_2[n]$ con $L+P-1$ muestras. En este caso, un periodo de la convolución circular coincidirá con la convolución lineal.

Concretando: La convolución lineal coincide con un periodo de la convolución circular si la convolución circular se realiza con un periodo N igual a la longitud de la convolución lineal $L+P-1$.

En el caso anterior $L+P-1=5$ sería:

$$x_1[n] * x_2[n] = conv(x_1, x_2) = ifft(fft(x_1,5).*fft(x_2,5))$$

Consideraciones computacionales

La DFT juega un importante papel en el análisis, diseño e implementación de sistemas y algoritmos de procesado de señales en tiempo discreto. Las propiedades ya vistas de la DFT hacen conveniente su análisis y diseño en el dominio de Fourier, pero el número de operaciones que se debe realizar para calcular la DFT disuaden de su uso.

Para calcular el número de operaciones que requiere el procedimiento clásico de cálculo de la DFT:

$$X[k] = \sum_{n=0}^{n=N-1} x[n]\, e^{-j\frac{2\pi kn}{N}} = \sum_{n=0}^{n=N-1} x[n]\, W_N{}^{kn}$$

Se tiene que para cada obtener cada $X[k]$ hay que hacer N multiplicaciones complejas y N sumas complejas. Así, para calcular todas las $X[k]$, $k=0,1,2,..,N-1$ hay que realizar N^2 multiplicaciones complejas y N^2 sumas complejas. Como una suma compleja son $2N$ sumas reales y una multiplicación compleja son $2N$ sumas reales y $4N$ multiplicaciones reales, necesitamos realizar $4N^2$ sumas reales y $4N^2$ multiplicaciones reales.

Para reducir el número de operaciones se han desarrollado una serie de algoritmos que reducen el número de operaciones a realizar para calcular la DFT llamados FFT (Fast Fourier Transform). Por ejemplo, la DFT por diezmado en el tiempo divide una DFT de $N=2^{\nu}$ muestras en dos DFT de $N/2$ y éstas en 4 DFTs de $N/4$ muestras hasta llegar a DFTs de 2 muestras. En este caso se precisan $\left(\frac{N}{2}\right)\log_2 N$ multiplicaciones y $N\log_2 N$ suma complejas.

1.5 Problemas de Aula

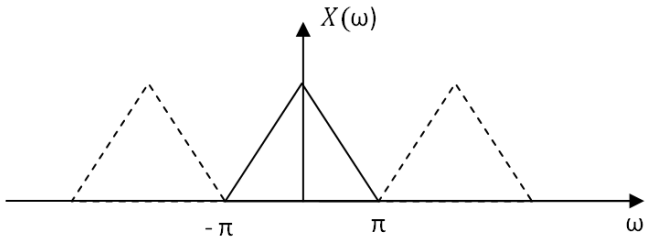
-Problema 1.1

Dada la señal $X(\omega)$, dibujar la transformada de Fourier de:

a) $y_s[n] = \begin{cases} x[n] & para\ n\ par \\ 0 & para\ n\ impar \end{cases}$

b) $y_d[n] = x[2n]$

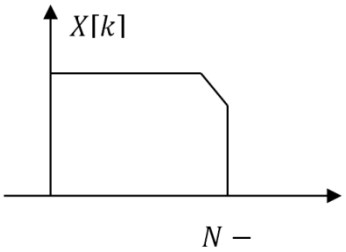
c) $y_e[n] = \begin{cases} x[n/2] & para\ n\ par \\ 0 & para\ n\ impar \end{cases}$



-Problema 1.2

Sea $X[k]$ una secuencia finita tal que:

$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}nk}$



Calcular la transformada discreta de Fourier de:

a) $g_1[n] = x[N-1-n]$

b) $g_2[n] = (-1)^n x[n]$

c) $g_3[n] = \begin{cases} x[n] & 0 \le n \le N-1 \\ x[n-N] & N \le n \le 2N-1 \\ 0 & resto \end{cases}$

d) $g_4[n] = \begin{cases} x[n] + x[n+N/2] & 0 \le n \le N/2-1 \\ 0 & resto \end{cases}$

e) $g_5[n] = \begin{cases} x[n] & 0 \le n \le N-1 \\ 0 & N \le n \le 2N-1 \\ 0 & resto \end{cases}$

f) f.- $g_6[n] = \begin{cases} x[n/2] & para\ n\ par \\ 0 & para\ n\ impar \end{cases}$

g) $g_7[n] = x[2n]$

-Problema 1.3

Sea $xc(t)=A_0 e^{j2\pi f_0 t}$ con $0<f_0<1KHz$ una señal continua de amplitud y frecuencia desconocida. Se quiere saber cuál es su amplitud máxima A_0 y su frecuencia f_0. Para ello se dispone de un sistema que realiza las siguientes operaciones sobre la señal de entrada:

- Muestrea $x_c(t)$ mediante un conversor C/D a una frecuencia de muestreo f_s, obteniendo una secuencia $x[n]$.
- Escoge las N primeras muestras de $x[n]$, generando la secuencia $r[n]$.
- Calcula el módulo de la DFT de $|R[k]|$ (sin rellenar por ceros)
- Selecciona el índice k_0 en que $|R[k]|$ es máximo y obtiene el valor $|R[k_0]|$.

Considere a partir de ahora que se utiliza la f_s obtenida en el apartado anterior.

a) ¿Cómo se puede estimar f_0 a partir del índice k_0?

b) ¿Cuál es el máximo error en función de N que se comete al estimar la frecuencia f_0? Calcule N para que el error máximo sea menor que 1 Hz.

c) Indique como podemos estimar A_0 a partir del valor $|R[k_0]|$ en el caso de que Nf_0/f_s sea entero.

d) Cuál es el máximo error (en función de N) que se puede cometer al estimar la amplitud A_0 en el caso de f_0 arbitrario entre 0 y 1 kHz?

e) Proponga y justifique un método para disminuir el error a la hora de estimar A_0 sin aumentar N, con respecto al caso del apartado anterior.

-Problema 1.4

Supongamos que disponemos de un programa FFT para evaluar la DFT de una secuencia compleja. Si queremos calcular la DFT de una secuencia real, simplemente debemos igualar a cero la parte compleja antes de introducir la secuencia en el programa. Sin embargo, la simetría de la DFT de una secuencia real puede usarse para reducir la carga computacional.

a.- Sea $x[n]$ una secuencia real de N valores, y sea $X[k]$ su Transformada Discreta de Fourier con parte real e imaginaria denominadas $X_R[k]$ y $X_I[k]$ respectivamente:

$$X[k] = X_R[k] + jX_I[k]$$

Demostrar que si $x[n]$ es real, entonces $X_R[k] = X_R[N-k]$ y $X_I[k] = -X_R[N-k]$ para $k = 1, \dots, N-1$.

-Problema 1.5

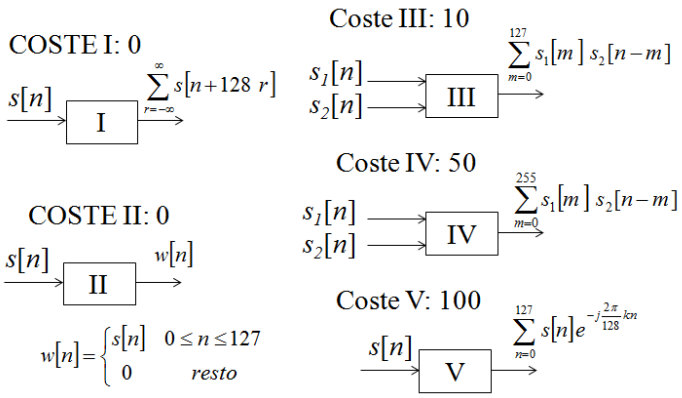
Sea:

$$x[n] > 0 \quad 0 \le n \le 255$$
$$y[n] > 0 \quad 0 \le n \le 255$$
$$x[n] = y[n] = 0 \quad resto$$

Suponiendo que:

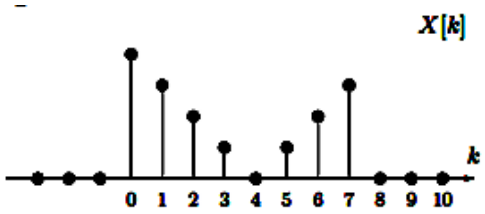
$$r[n] = x[n] * y[n]$$
$$R(w) = TF\{r[n]\}$$
$$R_s[k] = R(w)\big|_{w=\frac{2\pi k}{128}}$$

Se pide: a partir de $x[n]$ e $y[n]$ obtener $R_s[k]$ con el mínimo coste computacional utilizando los siguientes módulos:



-Problema 1.6

Sea $x[n]$ una secuencia de duración 8 que tiene la DFT de 8 puntos mostrada en la siguiente figura:

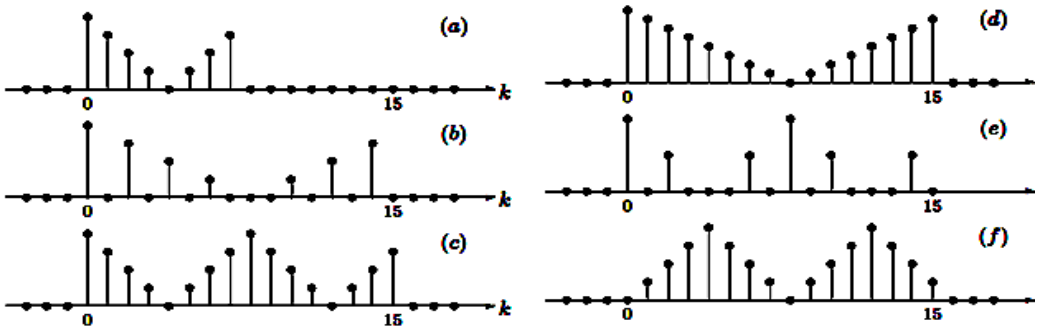


Se definen 2 secuencias $y[n]$ y $z[n]$, de 16 muestras de la siguiente manera

$$y[n] = \begin{cases} x[n/2] & para\ n\ par \\ 0 & para\ n\ impar \end{cases}$$

$$z[n] = \begin{cases} x[n] & 0 \le n \le 8 \\ 0 & 8 \le n \le 16 \end{cases}$$

Determine razonablemente cuál de las siguientes DFT: a,b,c,d,e o f corresponde a $y[n]$ y cual corresponde a $z[n]$



-Problema 1.7

Considere una señal discreta de 20 muestras de duración, de modo que $x[n]$=0 fuera del intervalo $-2 \le n < 17$. Sea $X(\omega)$ la transformada de Fourier de dicha señal.

Se desea evaluar $X(\omega)$ en $\omega = 5\pi/6$ calculando una única FFT de *M* puntos.

a) Determine el valor mínimo de M (número de puntos de la FFT) necesario.

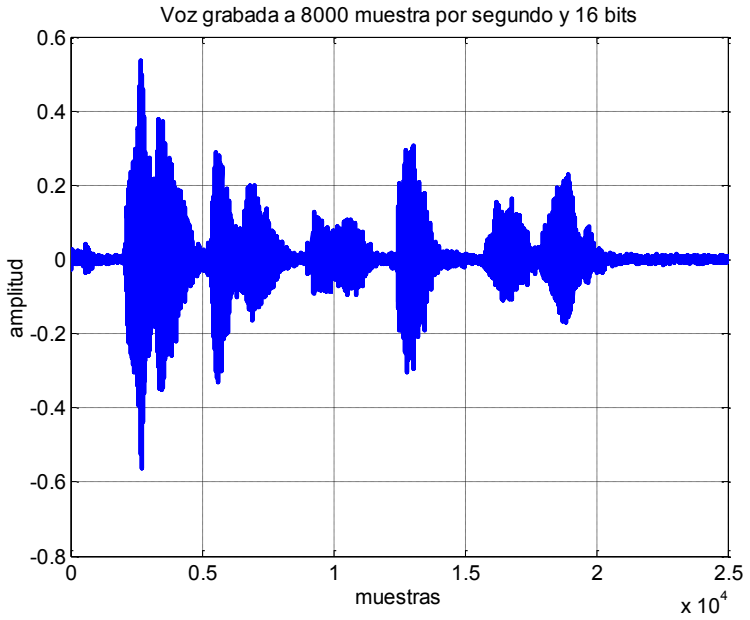
b) Determine el procedimiento a seguir, es decir, indicar exactamente de qué secuencia se calcula la FFT con M mínima y qué valor del resultado de la FFT con M mínima se toma como $X(\omega = 5\pi/6)$.

c) Repita el apartado para $\omega = 8\pi/27$ y suponiendo que x[n]=0 fuera del intervalo $10 \le n < 29$.

1.6 Ejercicios Prácticos

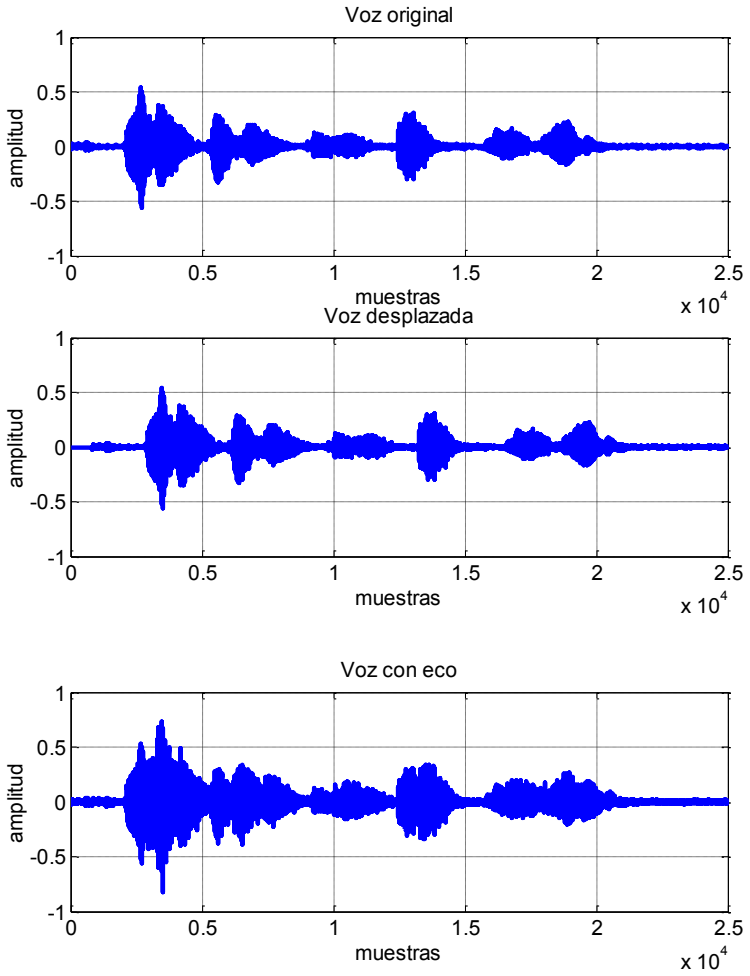
Grabación de una señal de voz

Grabe una señal de tres segundo de voz a *fs*=8000 muestras por segundo y 16 bits por muestra en formato *double*, en donde la voz ocupe dos segundos. Elimine la media de la grabación (*voz=voz-mean(voz)*). Asígnele a la señal grabada el nombre *voz*, y grábela en el fichero *voz.wav*. Para asegurar que la voz está en medio de la locución, represente la voz grabada. Si la voz no está centrada, repita la grabación.



Desplazamiento de una señal

Desplace la voz 100 mseg utilizando la función *conv*. Represente a la vez y en gráficas separadas la voz original, la desplazada y la suma de ambas (voz con eco). Escuche la suma de ambas a *fs* muestras por segundo.



Inversión de espectro

Los inversores de frecuencia se han utilizado durante muchos años en secrafonía (embrollado de la voz); en efecto, una señal de voz $x[n]$ resulta ininteligible si se invierte su espectro tal como se muestra en la figura.

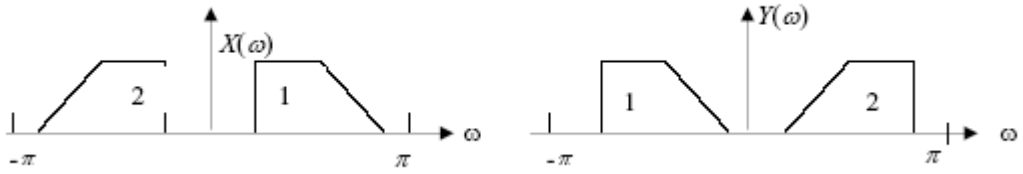


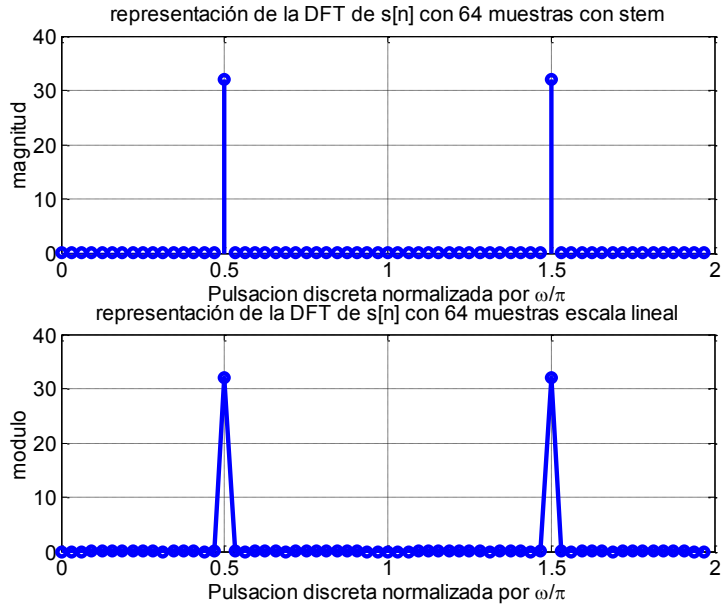
Figura. Izquierda: Espectro original; Derecha: Espectro invertido en frecuencia.

Para efectuar este procedimiento con una señal discreta basta con desplazar el espectro π y éste queda invertido por la propiedad de periodicidad 2π de los espectros de señales discretas. Esta operación se puede realizar en el dominio del tiempo multiplicado la voz por la secuencia $e^{j\pi n}=(-1)^n$.

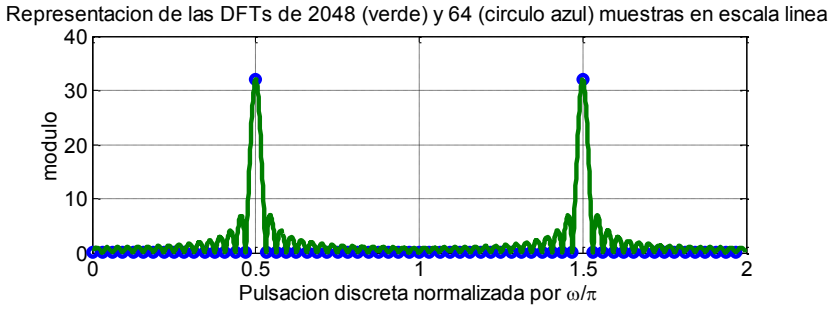
Invierta en frecuencia la voz obtenida en el apartado 1. Escuche el resultado obtenido. Invierta de nuevo en frecuencia la voz con el espectro invertido. Escúchela y verifique que coincide con la original.

Efectos en la representación de la magnitud de la DFT

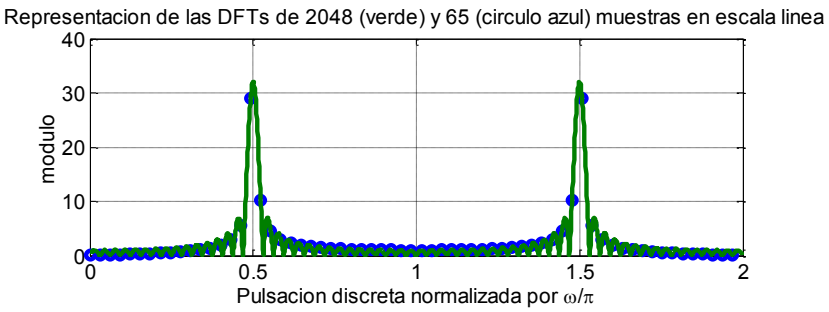
Genere $L=64$ muestras de un seno $s[n]$ de pulsación $\omega=0.5\pi$ y amplitud 1. Siendo $S(\omega)$ la transformada de Fourier de s[n], y $S_{64}[k] = S(\omega)\big|_{\omega=2\pi k/64}$ su DFT de *64* muestras. Represente el módulo $\left|S_{64}[k]\right|$, en escala lineal con *stem* y *plot*. Verifique que la amplitud de los picos de las *sinc* es (L/2).



La DFT del seno aparenta dos *deltas* cuando en teoría se ha estudiado que deberían ser dos *sinc* por el efecto del enventanado. Para ver las dos *sinc* hay que tomar más muestras de la DFT. Para ello partiendo de la misma señal $s[n]$, calcule el módulo de su DFT de 2048 muestras $S_{2048}[k]=S(\omega)\big|_{\omega=2\pi k/2048}$. Represente el módulo de $S_{2048}[k]$ solapadas con las muestras de $S_{64}[k]$ en escala lineal.



Para visualizar el efecto de obtener la DFT muestreando la transformada de Fourier en frecuencias no submúltiplo de la frecuencia de la señal $s[n]$ generada en el apartado A, obtenga su DFT de 65 muestras $S_{65}[k]=S(\omega)\big|_{\omega=2\pi k/65}$. Represente el módulo de $S_{2048}[k]$ solapadas con las muestras de $S_{65}[k]$ en escala lineal. ¿Por qué no se muestrea el pico dela s*inc*?¿Cual debería ser la frecuencia de la sinusoide para muestrear el pico?

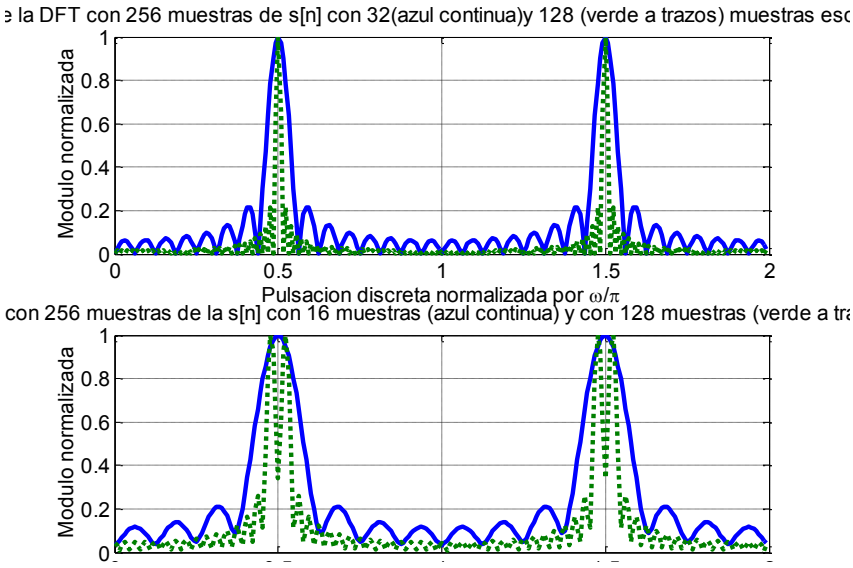


Se procurará visualizar los efectos de aumentar la longitud de la señal $s[n]$. Defina $sc1[n]$ como 32 muestras de un seno de pulsación $\omega=0.5\pi$ y $sc2[n]$ como 128 muestras del mismo seno. Calcule los módulos de las DFT de 256 muestras de $scl[n]$ y $sc2[n]$. Represente de forma solapada en la misma gráfica ambos módulos normalizados en amplitud a 1 (utilice la función *max* de Matlab). Comente los resultados.

Como aplicación práctica de lo anterior, genere dos versiones de la señal:

$$sc[n]= \sin(0.48\pi n) + \sin(0.52\pi n)$$

La primera versión con 16 muestras y la segunda versión con 128 muestras. Calcule sus módulos con DFTs de 256 muestras y represente normalizados a uno, a la vez y en gráficas separadas los módulos de sus DFTs. Comente los resultados.

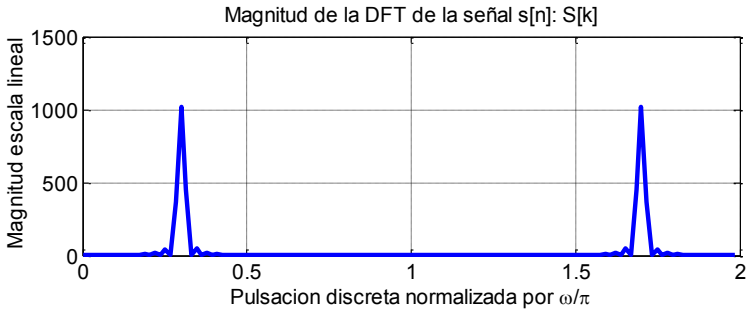


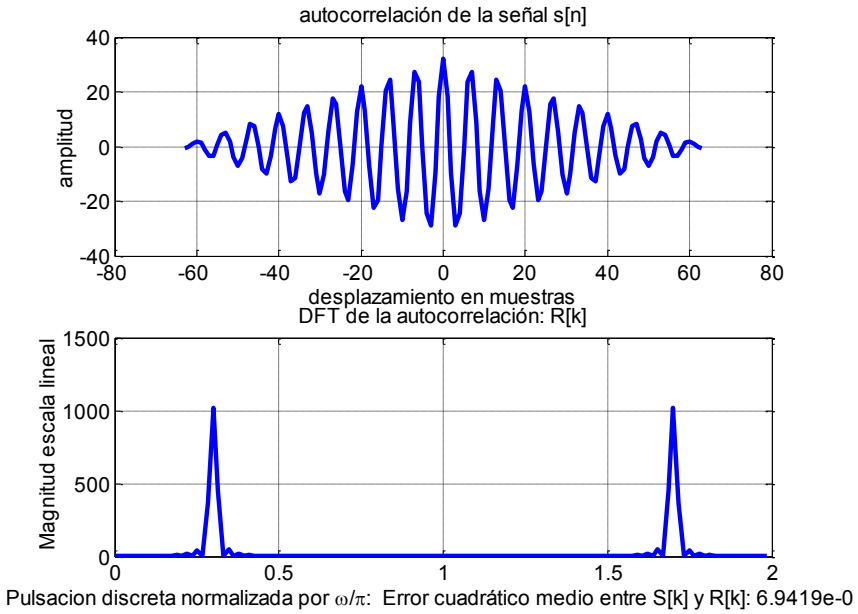
Calculo de la magnitud de la DFT mediante la autocorrelación

Se comparan dos procedimientos de cálculo de la magnitud de la DFT de una señal: 1) aplicación directa de la transformada de Fourier a la señal, y 2) mediante la transformada de Fourier de la autocorrelación de la señal.

Para ello realice el siguiente procedimiento

1. Genere 64 muestras de un seno $s[n]$ de pulsación $\omega=0.3\pi$
2. Calcule la magnitud de la transformada discreta de Fourier $S[k]$ del punto anterior mediante la función *fft* con 127 muestras. Represéntela en pantalla con *plot*.
3. Obtenga la magnitud de la transformada de Fourier de la señal $s[n]$ a través de la autocorrelación, esto es:
    i. Represente mediante *plot* la autocorrelación $r[m]$ de la secuencia $s[n]$ (puede utilizar la función *xcorr* de Matlab).
    ii. Dibuje mediante plot la transformada discreta de Fourier $R[k]$ de la secuencia de autocorrelación $r[m]$. Tenga en cuenta que la autocorrrelación es una secuencia par que comienza en índice negativo.
    iii. Para comprobar que ambos procedimientos son equivalentes, dibuje a la vez y en gráficas separadas las secuencias S[k] y R[k]. ¿Son ambas secuencias iguales?. Calcule el error cuadrático medio entre ambas secuencias: *sum((S-R).^2)*.



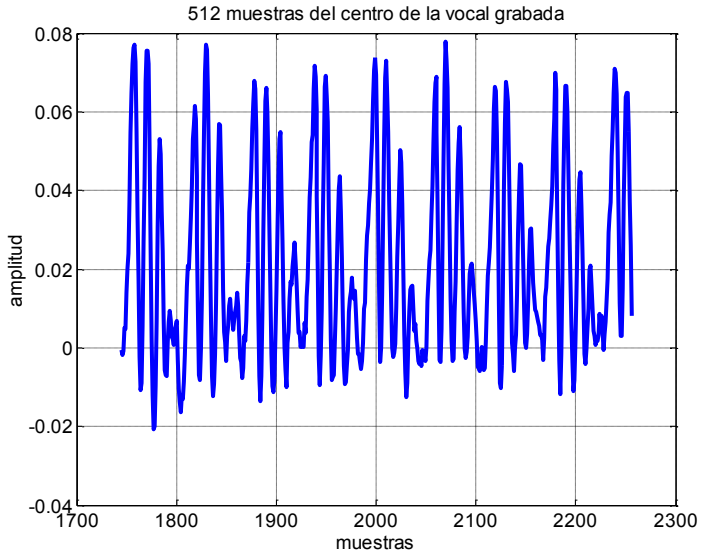


Repita éste apartado mediante DFTs de 256 muestras.

Repita de nuevo éste apartado con DFTs de 64 muestras.

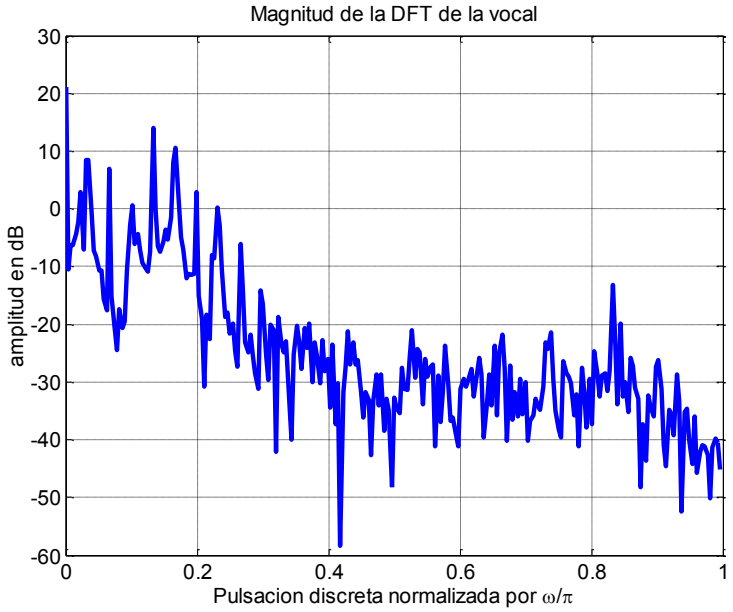
Cálculo de la DFT de una señal de voz

En este apartado se trata de calcular y representar la magnitud de la DFT de una señal real como la voz. Para ello grabe una vocal sostenida de aproximadamente medio segundo de duración (inicie y finalice la grabación durante la fonación). Represente gráficamente las 512 muestras centrales de la grabación.



Con ayuda del comando *ginput* obtenga el periodo de la señal.

De la vocal aislada del apartado anterior, coja las 512 muestras centrales y represente la DFT de dichas muestras en el rango [0 π]. Anote las propiedades de la voz que observe en la gráfica representada.



Obtenga el periodo de esta señal sobre el espectro y verifique que coincide con el obtenido en la señal temporal.

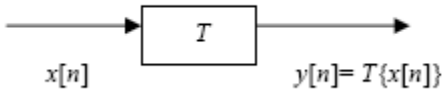
Capítulo II. Análisis de Sistemas Discretos

Vistas ya las señales continuas y discretas y las herramientas básicas de trabajo, como son la transformada de Fourier y la transformada discreta de Fourier, comenzamos en este capítulo el estudio de los sistemas discretos que modifican o transforman las señales discretas.

Como objetivo se pretende que el alumno sea capaza de calcular, manejar y relacionar las diferentes representaciones de un sistema racional lineal, causal e invariante en el tiempo.

2.1 Sistemas discretos lineales e invariantes en el tiempo (LTI)

Se define sistema como un conjunto de operaciones matemáticas que transforman una secuencia en otra.



Los sistemas pueden clasificarse según las siguientes propiedades:

1. *Sin memoria:* la salida depende de la entrada en el momento actual.

   Ejemplos: $y[n]=x^2[n] \rightarrow$ Sin memoria

   $y[n]=x[n-m] \rightarrow$ Con memoria

2. *Linealidad*: cumplen el principio de superposición, esto es, la transformada de la suma es igual a la suma de las transformadas.

   $$T\{\alpha \cdot x_1[n]+\beta \cdot x_2[n]\}=\alpha \cdot T\{x_1[n]\}+\beta T\{x_2[n]\} \qquad \text{[Ec.26]}$$

3. *Invarianza.* El sistema no depende del origen de tiempos.

   $x[n] \rightarrow y[n]$

   $x[n\text{-}n_o] \rightarrow y[n\text{-}n_o]$

4. *Causalidad*: la salida $y[n_o]$ sólo depende de $x[n]$ $n \leq n_o$ (no depende de ninguna muestra que venga después).

   La derivada forward es no causal $\rightarrow y[n]=x[n+1]-x[n]$

   La derivada backward es causal $\rightarrow y[n]=x[n]-x[n-1]$

5. *Estabilidad:* un sistema en estable en sentido BIBO cuando la salida está limitada si la entrada también los está: $|x[n]|<\infty \rightarrow |y[n]|<\infty$

   Para que un sistema no se sature debe ser estable.

Esta asignatura se centra en los sistemas que cumplen la propiedad de linealidad e invarianza, es decir, sistemas LTI.

Un sistema LTI se puede definir o caracterizar (describir su comportamiento) de varias maneras. En este capítulo veremos dos:

1. Mediante la respuesta al impulso.
2. Mediante ecuaciones en diferencias.

2.2 Análisis de sistemas LTI descritos mediante su respuesta al impulso

Si h[n] es la respuesta del sistema al impulso de un sistema LTI:

$$h[n]=T\{\delta[n]\} \qquad \text{[Ec.27]}$$

entonces, $h[n]$ define al sistema, esto es, a partir de la $h[n]$ puedo calcular la salida a cualquier entrada pues:

$$y[n]=T\{x[n]\} \qquad \text{[Ec.28]}$$

se descompone la $x[n]$ como suma de $\delta[n]$:

$$y[n]=T\left\{\sum_{k=-\infty}^{k=\infty} x[k]\delta[n-k]\right\} \qquad \text{[Ec.29]}$$

se aplica linealidad, teniendo en cuenta que la $T\{\}$ sólo afecta a la variable *n:*

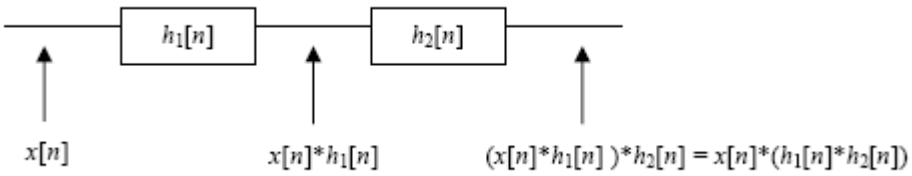
$$y[n]=\sum_{k=-\infty}^{k=-\infty} T\{x[k]\delta[n-k]\}=\sum_{k=-\infty}^{k=-\infty} x[k]\,T\{\delta[n-k]\} \qquad \text{[Ec.30]}$$

y aplicando invariabilidad

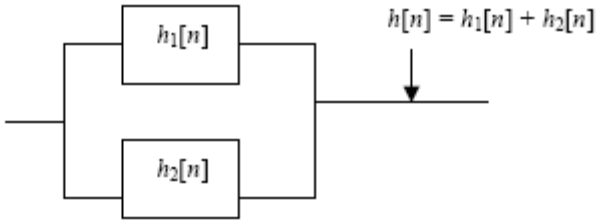
$$y[n]=\sum_{k=-\infty}^{k=-\infty} x[k]\cdot h[n-k]=x[n]*h[n] \qquad \text{[Ec.31]}$$

Esto es, cualquier salida se obtiene convolucionando la entrada con la respuesta al impulso, es decir, siempre que conozca la $h[n]$ podré saber cómo va a actuar el sistema.

Evidentemente, las propiedades del operador convolución también lo serán de los sistemas LTI, por ejemplo: si se tienen dos sistemas con respuesta al impulso $h_1[n]$ y $h_2[n]$ en cascada, la respuesta al impulso del sistema resultante será la convolución de sus respuestas al impulso $h_1[n]* h_2[n]$.



Y si los dos sistemas se combinan en paralelo, la respuesta al impulso del sistema equivalente será $h_1[n]+h_2[n]$.



Propiedades del sistema que se pueden conocer a partir de la $h[n]$ son:

- ✓ Si se cumple que $h[n]$ es absolutamente sumable

  $\sum_{k=-\infty}^{+\infty} |h[k]| < \infty \quad \rightarrow$ El sistema es *Estable*

- ✓ Si $h[n]=0, \quad n<0 \rightarrow$ El sistema es *Causal*.

  Esto es, si la respuesta al impulso comienza en cero, el sistema es siempre causal: la salida no se va a adelantar nunca a la entrada.

- ✓ Si $h[n]$ es finita, el sistema es *FIR* (respuesta al impulso finita).

  Los sistemas FIR son siempre estables

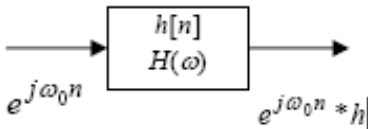
  Los sistemas FIR siempre se pueden hacer causales

- ✓ Si $h[n]$ es infinita, el sistema es IIR (respuesta al impulso infinita).

De lo expuesto anteriormente se deduce que a partir de la $h[n]$ es posible determinar si el sistema es FIR o IIR, si es causal o si es estable.

Conocer más propiedades de un sistema a partir de su $h[n]$ es complicado. Más sencillo es hacerlo a partir de su función de trasferencia $H(\omega)=\mathrm{TF}\{h[n]\}$ pues las exponenciales complejas $e^{j\omega_0 n}$ son autofunciones de un sistema lineal cuyo autovalor

es $H(\omega_0)$, esto es, la salida de un sistema LTI a una exponencial compleja es la misma exponencial compleja con módulo y fase diferente.



$$e^{j\omega_0 n} * h[n] = \sum_m h(m) e^{j\omega_0 (n-m)} = e^{j\omega_0 n} \sum_{m=-\infty}^{+\infty} h(m) e^{-j\omega_0 m} = e^{j\omega_0 n} H(\omega_0)$$

así, a la salida del sistema LTI nunca obtendremos señales de frecuencias distintas de las de entrada.

Los parámetros de la función de transferencia $H(\omega)$ con los que se trabaja son:

$$H(\omega) = H_R(\omega) + jH_I(\omega) = \begin{cases} Módulo & |H(\omega)| = \sqrt{H_R^2(\omega) + H_I^2(\omega)} \\ Fase & \angle H(\omega) = arctg\left[\dfrac{H_I(\omega)}{H_R(\omega)}\right] \end{cases}$$

Transformaciones típicas del módulo son:

- ✓ la Ganancia en dB = $20\log_{10}|H(\omega)|$
- ✓ la Atenuación en dB = $-20\log_{10}|H(\omega)|$
- ✓ la Magnitud $\rightarrow |H(\omega)|^2$

La magnitud indica el carácter selectivo del filtro en frecuencia, esto es, la amplificación o atenuación que el filtro da a cada exponencial compleja (frecuencia). Según la forma de la magnitud, el filtro puede ser pasobajo, pasoalto, pasobanda o banda eliminada.

La fase, está íntimamente ligada al retraso que introduce el sistema en cada frecuencia. Supuesto que cualquier entrada $x[n]$ puede descomponerse como una combinación lineal de exponenciales complejas, la fase indica cuanto retrasa el sistema cada exponencial compleja en radianes

Cuando se calcula la fase de forma numérica con un ordenador (en vez de calcularla de forma analítica como en el anterior párrafo), éste devuelve el valor principal de la fase en el rango $[-\pi +\pi]$ obteniéndose una función de fase discontinua (denominada $ARG[H(\omega)]$) equivalente a la fase continua (denominada $arg[H(\omega)]$) obtenida en el cálculo analítico.

Por tanto, en el cálculo numérico de fase, se deben realizar dos pasos: primero calcular la fase discontinua (función *angle* de Matlab) y a partir de ella obtener la fase continua (función *unwrap* de Matlab).

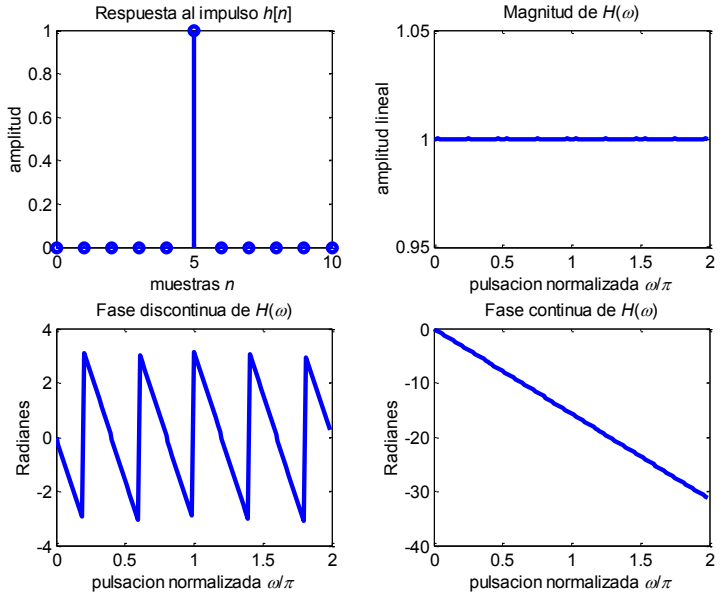
Ejemplo: Sea el sistema $h[n]=\delta[n-n_0]$, con n_0=5.

Su función de transferencia es: $H(\omega)=e^{-j\omega n_0}$

de donde su fase vale: $\angle H(\omega)=-\omega n_0$ y su magnitud: $|H(\omega)|^2=1$)

La fase se puede representar como una recta de pendiente $-n_0$. Esto es, en $\omega=\pi$ el sistema desfasa la frecuencia $-\pi n_0$.

```
h=[zeros(1,5),1,zeros(1,5)];
H=fft(h,128);
MagH=H.*conj(H);
ARGH=angle(H)
argH=unwrap(ARGH); %argH=phase(H)
```



En el ejemplo anterior se puede observar cómo cuando la fase del sistema es una línea recta de pendiente n_0=-5, lo que hace el sistema $h[n]=\delta[n-5]$ es retrasar 5 muestras la señal de entrada.

Existe un efecto indeseado llamado *distorsión de fase* que consiste en el cambio de la relación de fase que existe entre las exponenciales complejas de la entrada y entre las exponenciales complejas de la salida. Un ejemplo de este fenómeno es el siguiente:

Sea la señal $x[n]=sin(\omega_1 n)+sin(\omega_2 n)$

Se pasa por un sistema $H(\omega)\, con \begin{cases} módulo & |H(\omega)| = 1 \\ fase & \begin{cases} \arg[H(\omega_1)] = -\pi \\ \arg[H(\omega_2)] = -\pi/2 \end{cases} \end{cases}$

La salida del sistema a $x[n]$ será $y[n]= \sin(\omega_1 n-\pi)+\sin(\omega_2 n-\pi/2)$

donde las dos sinusoides de frecuencia ω_1 y ω_2 de $y[n]$ no tienen entre ellas la misma relación de fase que tenían en la entrada $x[n]$.

Para evitar dicha distorsión el sistema debe cumplir

- ✓ Bien que la fase del sistema sea cero, lo cual no cambia la fase de ninguna exponencial manteniéndose la relación de fase entre ellas.

  El problema es que si la fase del sistema es cero $\arg[H(\omega)] =0$, entonces

  $H(\omega) = H_R(\omega) + j\, H_I(\omega) = H_R(\omega) + j0$

  Al ser $H(\omega)$ real, $h[n]$ será par (simétrica) lo cual implica un sistema no causal

- ✓ Bien que la fase del sistema sea lineal, es decir, del tipo: $\arg[H(\omega)] = -\omega n_0$

  esta fase aplicada al sistema anterior, da una salida:

  $y[n]=sin(\omega_1 n+arg[H(\omega_1)])+sin(\omega_2 n+arg[H(\omega_2)])= sin(\omega_1 n-\omega_1 n_0)+sin(\omega_2 n-\omega_2 n_0)=$

  $y[n]= sin(\omega_1(n-n_0))+sin(\omega_2(n-n_0))$

  Como puede verse, un sistema con fase lineal tan sólo retrasa la señal manteniendo la relación de fase entre las exponenciales de las diferentes frecuencias Por tanto un sistema de fase lineal no tiene distorsión de fase.

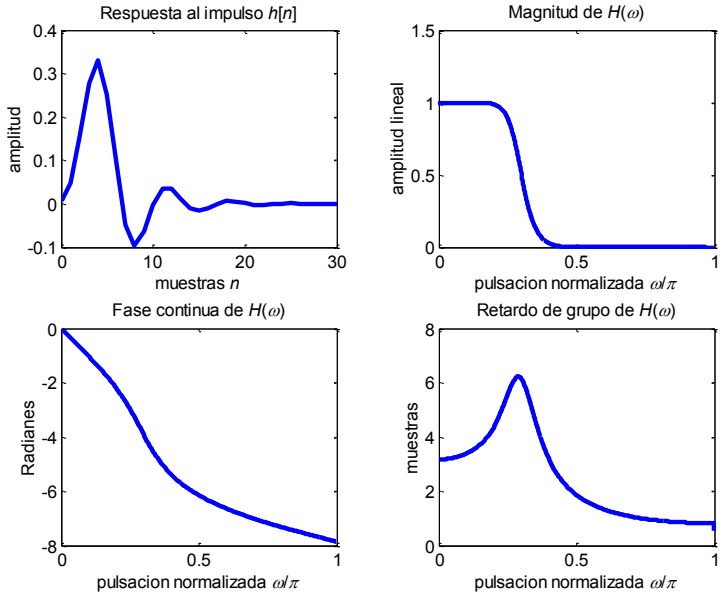
Así puede ser útil una medida de la linealidad de la fase. Esta media se denomina *Retardo de Grupo* y se define como la derivada de la fase continua:

$$\tau(\omega) = -\frac{d}{d\omega}\arg[H(\omega)] \qquad \text{[Ec.32]}$$

Si la fase es lineal, $\tau(\omega)$ es constante y de valor igual al retardo (la n_0 del ejemplo anterior) que produce el sistema.

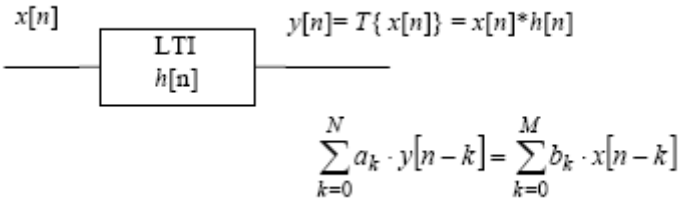
Si el sistema no es de fase lineal e introduce distorsión de fase se puede ver en qué frecuencias hay más distorsión de fase y en qué frecuencias la distorsión de fase es más reducida.

Ejemplo: En la gráfica tenemos la respuesta al impulso, magnitud, fase y retardo de grupo de un sistema paso bajo. Como puede observarse, a pesar de que la fase puede parecer lineal a tramos. El retardo de grupo revela una fuerte distorsión de fase en la banda de transición. Al final de la banda atenuada el sistema es de fase casi lineal, aunque esto no reporta ninguna ventaja pues dichas exponenciales son atenuadas



2.3 Análisis de sistemas LTI descritos mediante sus ecuaciones en diferencias.

Un sistema descrito por una ecuación en diferencias es aquel cuya relación entre la entrada y salida puede describirse mediante una ecuación en diferencias, esto es:



$$\sum_{k=0}^{N} a_k \cdot y[n-k] = \sum_{k=0}^{M} b_k \cdot x[n-k]$$

donde la salida del sistema definido por la ecuación en diferencias se calcula como:

$$y[n] = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k) \text{ suponiendo } a_0=1 \qquad \text{[Ec.33]}$$

por lo tanto, estos sistemas se consideran definidos por los coeficientes de la ecuación en diferencias $\{a_k\}_{k=0}^{N}$ y $\{b_k\}_{k=0}^{M}$. Un sistema definido por una ecuación en diferencias cumple las propiedades de un sistema LTI.

La ventaja de esta representación es la de poder utilizar filtros discretos IIR, lo cual no puede realizarse con la $h[n]$ pues no es posible programar una convolución con una secuencia infinita.

Para analizar un sistema definido por su $h[n]$ hemos recurrido a la transformada de Fourier, en cambio, para analizar un sistema definido por su ecuación en diferencias se recurre a la Transformada Z.

Aplicación de la TZ a la ecuación en diferencias

Se parte de la ecuación en diferencias:

$$\sum_{k=0}^{N} a_k \cdot y[n-k] = \sum_{k=0}^{M} b_k \cdot x[n-k] \qquad \text{[Ec.34]}$$

El resultado de calcula la transformada Z de la ecuación en diferencias es:

$$TZ\left\{\sum_{k=0}^{N} a_k y[n-k]\right\} = TZ\left\{\sum_{k=0}^{M} b_k x[n-k]\right\} \qquad \text{[Ec.35]}$$

Por ser la transformada Z un operador lineal, se tiene:

$$\sum_{k=0}^{N} a_k TZ\{y[n-k]\} = \sum_{k=0}^{M} b_k TZ\{x[n-k]\} \qquad \text{[Ec.36]}$$

Si llamamos $X(z)=TZ\{x[n]\}$, y $Y(z)=TZ\{y(n)\}$, aplicamos propiedades de la TZ y se obtiene:

$$\sum_{k=0}^{N} a_k z^{-k} Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z) \qquad \text{[Ec.37]}$$

De donde se puede definir una relación entre la salida y entrada:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} = Cte\,\frac{\prod_{k=1}^{M} \left(1 - c_k \cdot z^{-1}\right)}{\prod_{k=1}^{N} \left(1 - d_k \cdot z^{-1}\right)} = \frac{Q(z)}{P(z)} \qquad \text{[Ec.38]}$$

Dicha relación se denomina función de transferencia del sistema.

Relación de la ecuación en diferencias con la respuesta al impulso

Téngase en cuenta que la salida de un sistema representado por la respuesta a impulso se calcula: $y[n] = x[n] * h[n]$.

Si se aplica la TZ a dicha ecuación, se tiene $Y(z) = X(z)H(z)$ de donde

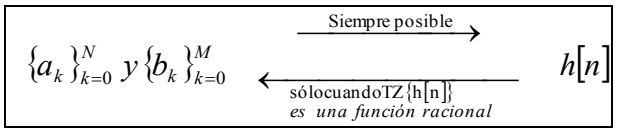
$$H(z) = TZ\{h[n]\} = \frac{Y(z)}{X(z)}$$

esto es, la función de transferencia coincide con la *TZ* de la respuesta al impulso.

Notar que:

$$H(z)\big|_{r=1} = \frac{Y(z)\big|_{r=1}}{X(z)\big|_{r=1}} = \frac{Y(\omega)}{X(\omega)} = H(\omega) = TF\{h[n]\} \qquad \text{[Ec.39]}$$

por tanto, pasar de la ecuación en diferencias $\{a_k\}_{k=0}^{N}$ y $\{b_k\}_{k=0}^{M}$ a $h[n]$ mediante la transformada *Z* inversa de la función de transferencia siempre es posible mientras que el paso contrario sólo es posible cuando la $TZ\{h[n]\}$ pueda expresarse como una función racional. Por tanto sólo se podrán implementar sistemas IIR cuya $h[n]$ tenga como transformada *Z* una función racional



Análisis de un sistema definido por ecuación en diferencias

El análisis de los sistemas definidos por ecuación en diferencias se realiza a partir de las raíces de los polinomios $Q(z)$ y $P(z)$. A las raíces c_k del polinomio del numerador $Q(z)$ se les denomina ceros del sistema, y a las raíces d_k del polinomio del denominador $P(z)$ se les denomina polos del sistema.

Si el sistema tienen polos es un sistema IIR. Si no tienen polos (exceptuando los polos en $z=0$ y $z=\infty$), el sistema es FIR.

La causalidad y estabilidad del sistema se puede determinar a partir de la región de convergencia de la función de transferencia $H(z)$ (regiones concéntricas que excluyen los polos): si la región de convergencia utilizada incluye el circulo unidad, el sistema es estable; si la región de convergencia es la más exterior, el sistema es causal; si la región de convergencia es la más interior, el sistema es anticausal; si la región de convergencia es un anillo; la repuesta al impulso va desde $-\infty$ hasta $+\infty$. Si el sistema no tienen polos, la región de convergencia abarca todo el plano Z y el sistema es FIR.

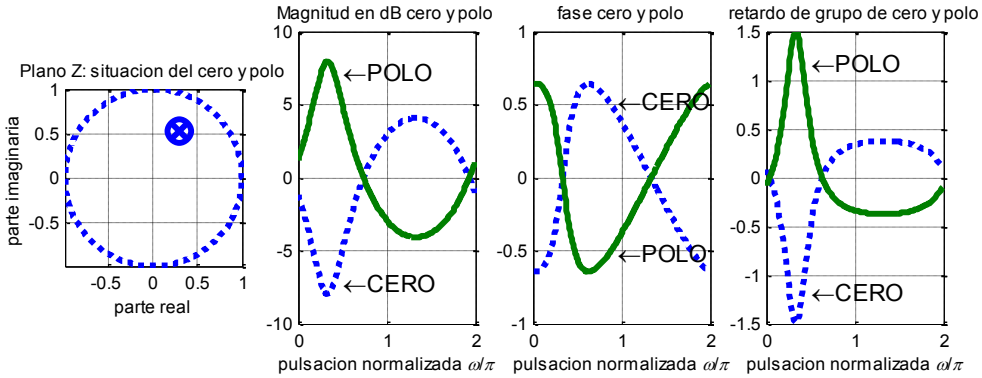
Por esa razón si el sistema tiene todos los polos dentro del círculo unidad, la región de convergencia más exterior incluye el círculo unidad y el sistema puede ser causal y estable.

Vamos a ver cómo se obtienen la magnitud y la fase del sistema a partir de los polos y ceros:

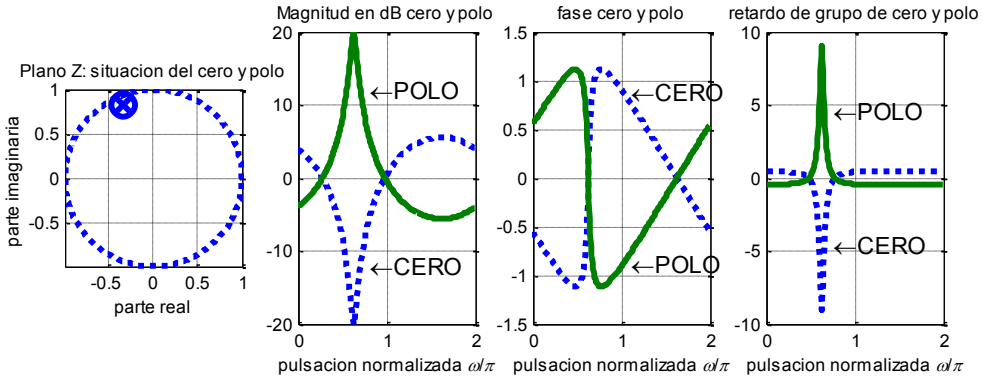
Ejemplos para obtener la magnitud y la fase a partir de los polos y ceros:

Para un polo en $c_k = re^{j\theta} = 0.6\, e^{j\pi/3}$

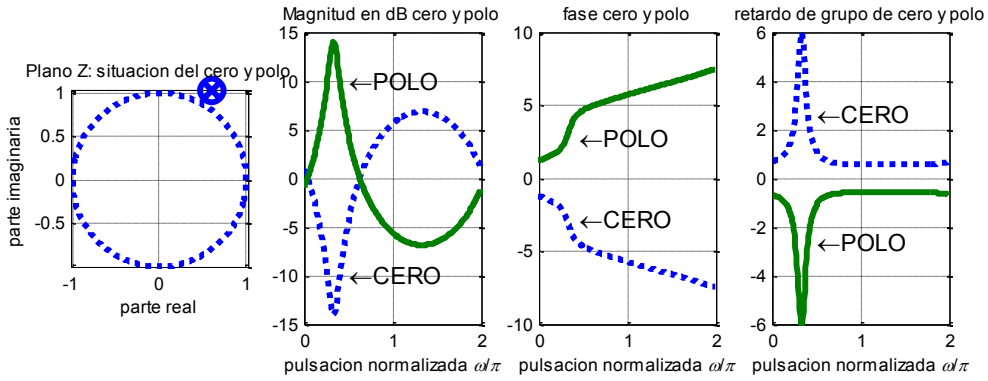
```
r=0.6;          % radio del cero o polo
O=pi/3;       % ángulo del cero o polo
a=r*exp(j*O);   % valor del cero o polo
% Calculo de la función de transferencia
[Hpolo,w]=freqz(1,poly(a),256,'whole');
[Hcero,w]=freqz(poly(a),1,256,'whole');
% calculo del retardo de grupo
Gdpolo=grpdelay(1,poly(a),256,'whole');
Gdcero=grpdelay(poly(a),1,256,'whole');
% representación gráfica
subplot(141),zplane(poly(a),poly(a)),title('Situación del polo/cero')
subplot(142),semilogy(w/pi,abs(Hcero).^2,w/pi,abs(Hpolo).^2);grid
title('Magnitud cero/polo')
subplot(143),plot(w/pi,phase(Hcero),w/pi,phase(Hpolo));grid
title('fase cero/polo')
subplot(144);plot(w/pi,Gdcero,w/pi,Gdpolo);grid
title('retardo de grupo de cero/polo')
```



si se mueve el polo de θ=*pi*/3 a θ=5**pi*/8 y el radio a r=0.9 se tiene:



Notar que las anteriores gráficas son sólo ciertas para ceros y polos dentro del circulo unidad. Si el polo o cero esta fuera del circulo unidad, la magnitud no varía; la fase varía pues en vez de presentar un punto de inflexión $\omega=\theta$ presenta un aumento de pendiente (téngase en cuenta que en este caso en $\omega=\theta$ el denominador de la función *atan* con la que se calcula la fase cambia de signo). El retardo de grupo varía siguiendo la derivada de la fase. Ejemplo con r=1.2 y θ=*pi*/3.



En estas gráficas se puede observar cómo a partir de la posición de los polos y ceros se puede reconstruir la magnitud, fase o retardo de grupo del sistema.

De las anteriores gráficas puede deducirse que un cero apenas introduce retardo de grupo en la banda de paso del filtro mientras que un polo introduce mucha distorsión de fase dentro de la banda de paso.

Ejemplo de uso de la transformada Z aplicada a ecuaciones en diferencias:

Suponga $h[n]=a^n u[n]$, entonces su transformada Z es

$$H(z)=\sum_{n=-\infty}^{n=+\infty} a^n u[n] z^{-n} = \frac{1}{1-az^{-1}} = \frac{Y(z)}{X(z)}$$

de donde la ecuación en diferencias del sistema es:

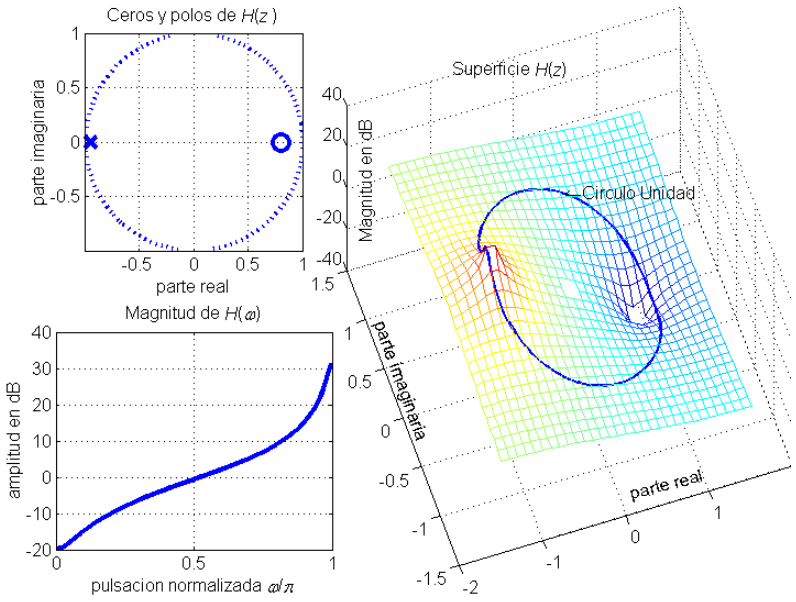
$y[n]=x[n]+ay[n-1]$

ecuación fácilmente programable en un microprocesador:

Adicionalmente, la $H(z)$ nos facilita el análisis del sistema:

1. Cómo el denominador tienen raíces→ tiene polos→el sistema es IIR.
2. Cómo el polo está en $z=a$, si $abs(a)<1$, entonces el circulo unidad está dentro de la región de convergencia hacia el exterior del polo y el sistema puede ser causal y estable.
3. Es sencillo determinar el tipo de sistema (pasoalto, bajo, etc.) a partir de sus polos y ceros. Por ejemplo, obsérvese cómo si a tiene fase cero el filtro es paso bajo, y conforme aumenta su ángulo hacia π el filtro se transforma en paso banda hasta paso alto.

Ejemplo de determinación de magnitud de $H(z)=\frac{1-0.8z^{-1}}{1+0.95z^{-1}}$



Uso en Matlab, suponiendo que tenemos los coeficientes del numerador y denominador en los vectores $b=[b_0\ b_1,....,b_M]$, y $a=[a_0\ a_1,....,a_N]$

- ✓ sus polos y ceros se pueden representar mediante *zplane*(*b,a*)
- ✓ sus polos y ceros se pueden obtener a partir de *b* y *a* con la función *roots*
- ✓ los coeficientes *a* y *b* pueden obtenerse a partir de los polos y ceros con la función *poly*
- ✓ su magnitud y fase se puede dibujar mediante la función *freqz*(*b,a*)
- ✓ su retardo de grupo con la función *grpdelay*(*b,a*)

- ✓ y se puede filtrar una entrada $x[n]$ por dicho filtro mediante la función $y=filter(b,a,x)$

Las funciones *freqz* y *grpdelay* devuelven muestras de la función de transferencia y retardo de grupo. Hay que tener en cuenta que si la función de transferencia o retardo de grupo tienen alguna discontinuidad o valor infinito (caso de cero sobre el circulo unidad) y se solicita el valor en dicho punto ambas funciones devolverán mensajes de *warning*.

2.4 Sistemas racionales particulares: Paso Todo, Fase Mínima y de Fase Lineal.

Se trata de sistemas descritos por la $H(z)$ que presentan unas características muy útiles como veremos a continuación.

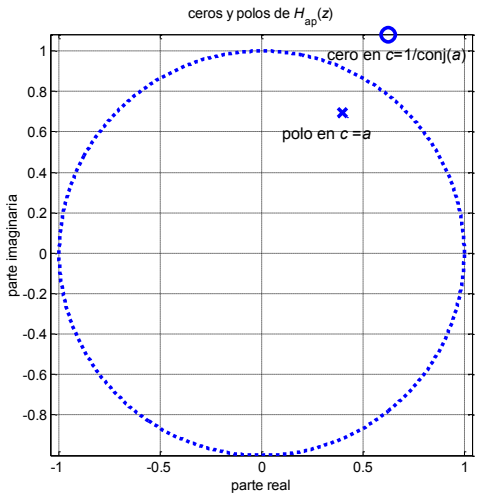
Sistema Paso Todo

Son sistemas que no varían la amplitud de ninguna exponencial compleja, es decir, dejan pasar todas las frecuencias por igual. Sólo modifica la fase. Su función de transferencia es de la forma

$$H_{ap}(z)=\frac{z^{-1}-a^{*}}{1-az^{-1}} \text{ que tiene un polo en } z=a \text{ y un cero en } z=1/a^{*} \qquad \text{[Ec.40]}$$

Gráficamente, los polos y ceros de un sistema paso todo se representan:

```
r=0.8;
angulo=pi/3;
polo=r*exp(j*angulo);
zplane([-conj(polo) 1],[1 -polo])
title('ceros y polos de Hap(z)')
```



La expresión del mismo sistema en función de su cero $c=1/a^{*}$ es:

$$H_{ap}(z)=\frac{z^{-1}-a^{*}}{1-az^{-1}}=\frac{c^{*}}{c}\cdot\frac{1-cz^{-1}}{z^{-1}-c^{*}} \qquad \text{[Ec.41]}$$

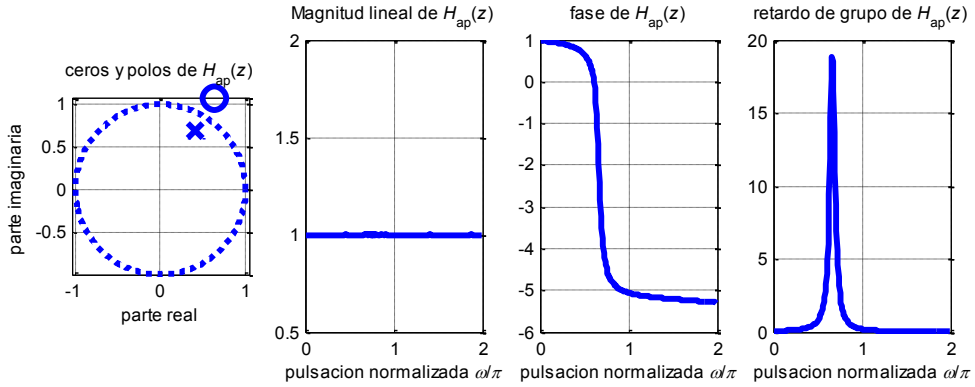
Al dejar pasar todas las frecuencias por igual, se debe cumplir que el módulo de $H(\omega)$ sea igual a $\left|H(z)\big|_{z=e^{j\omega}}\right|=\left|H(\omega)\right|==1$ [Ec.42]

Entonces, si tengo un sistema $H_1(z)= H(z)\cdot H_{ap}(z)$, $H_1(z)$ será un sistema con un polo y un cero más que $H(z)$ pero con la misma magnitud que $H(z)$ ya que el sistema paso todo no afecta a la magnitud.

En cambio, los sistemas paso todo si afectan a la fase. Se puede afirmar que la fase continua de un sistema paso todo tiene pendiente negativa y que su retardo de grupo es siempre positivo.

Un ejemplo de lo anterior lo tenemos en:

```
a=0.9*exp(2*j*pi/3);
% calculo de la magnitud y fase
[Hap,w]=freqz([-conj(a) 1],[1 -a],256,'whole');
% calculo del retardo de grupo
Gdap=grpdelay([-conj(a) 1],[1 -a],256,'whole');
```



Sistema de Fase Mínima

Un sistema de fase mínima $H_{min}(z)$ es aquel sistema cuyos polos y ceros están dentro del circulo unidad, lo cual garantiza mínimo retardo o fase mínima. Éstos sistemas cumplen la propiedad de que el sistema $H_{min}(z)$ y su inverso $1/H_{min}(z)$ son causales y estables.

Todo sistema $H(z)$ tienen un equivalente $H_{min}(z)$: misma magnitud, mínimo retardo. Su cálculo es muy usado para la ecualización de canales de comunicación, donde el canal

se modela con un filtro de fase mínima para que su inverso en la ecualización del receptor sea causal y estable.

¿Cómo se puede separar de $H(z)$ el factor $H_{\min}(z)$?. Supongamos que:

$$H(z) = H_{\min}(z)H_{ap}(z) = H_1(z)(1-cz^{-1})$$

con $|c|>1$ (un cero fuera del circulo unidad). Operando se obtiene

$$H(z) = H_1(z)(1-cz^{-1}) = H_1(z)(1-cz^{-1})\frac{(1-z^{-1}/c^*)}{(1-z^{-1}/c^*)} = H_1(z)(1-\frac{z^{-1}}{c^*})\frac{(1-cz^{-1})}{(1-z^{-1}/c^*)}$$

$$H(z) = H_1(z)(1-z^{-1}/c^*)(-c)\frac{(1-cz^{-1})}{(z^{-1}-c^*)}\frac{c^*}{c}$$

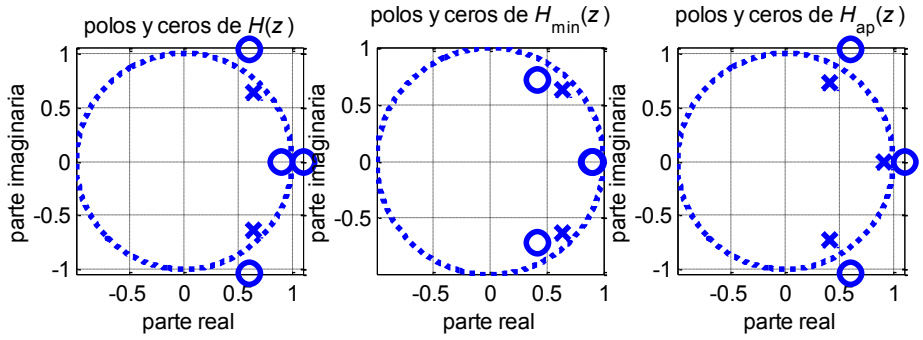
de donde por similitud con la fórmula del sistema paso todo

$$H_{\min}(z) = H_1(z)(1-\frac{z^{-1}}{c^*})(-c) \quad \text{y} \quad H_{ap}(z) = \frac{(1-cz^{-1})}{(z^{-1}-c^*)}\frac{c^*}{c}$$

Como puede verse, el efecto es que el cero de $H(z)$ situado en c se convierte en un cero en $1/c^*$ en el sistema de fase mínima. De esta forma la magnitud de $H(z)$ no cambia.

Ejemplo de separar $H(z)$ en $H_{\min}(z)$ y $H_{ap}(z)$

```
% Sistema de fase mínima
ceros=[1.2*exp(-j*pi/3) 1.2*exp(j*pi/3) 0.9 1.1];
polos=[0.9*exp(-j*pi/4) 0.9*exp(j*pi/4)];
b=poly(ceros);
a=poly(polos);
% sistema de fase mínima
amin=a;
ind=find(abs(ceros)>1);
g=prod(-ceros(ind));
ceros(ind)=conj(1./ceros(ind));
bmin=g.*poly(ceros)*b(1);
% sistemas paso todo unidos en uno
aap=poly(ceros(ind));
bap=poly(1./conj(ceros(ind)));
```



Sistema de Fase Lineal

Se trata de sistemas que no introducen distorsión de fase. Como vimos en el apartado anterior, para que un sistema no introduzca distorsión de fase debe ser de fase cero o de fase lineal. Un sistema de fase cero, tiene la respuesta al impulso $h[n]$ par, esto es $h[n]=h[-n]$, y función de transferencia $H(\omega)=|H(\omega)|e^{j0}$

Un sistema con respuesta al impulso igual a la $h[n]$ anterior retardada n_d muestras, esto es $h[n-n_d]$, será simétrica con eje de simetría en n_d y función de transferencia $H(\omega)e^{-j\omega n_d} = |H(\omega)|e^{-j\omega n_d}$

de donde su fase y retardo de grupo valdrán $\begin{cases} fase & \arg[H(\omega)] = -\omega n_d \\ retardo\,de\,grupo & \tau(\omega) = n_d \end{cases}$

donde se aprecia que es de fase lineal (sin distorsión de fase) y que el retardo de grupo coincide con el eje de simetría de la respuesta al impulso.

Se puede realizar el mismo razonamiento en sentido contrario llegando a la conclusión que todo sistema con $h[n]$ simétrica es de fase lineal.

Para que mi sistema sea de fase lineal y causal, se deben cumplir dos condiciones:

1. $h[n]$ debe ser simétrica
2. $h[n]=0$ $n<0$.

de donde un sistema de fase lineal y causal será FIR.

Un sistema FIR causal de fase lineal con respuesta al impulso de M+1 muestras, esto es: $h[n]=0$ $n<0$, y $n>M$, con $M\in\aleph$ (número entero positivo) tendrá una $h[n]$ simétrica con eje de simetría en M/2, esto cumplirá $h[n]=h[M-n]$

y tendrá $\begin{cases} fase & \arg[H(\omega)] = -\omega\dfrac{M}{2} \\ retardo\,de\,grupo & \tau(\omega) = \dfrac{M}{2} \end{cases}$

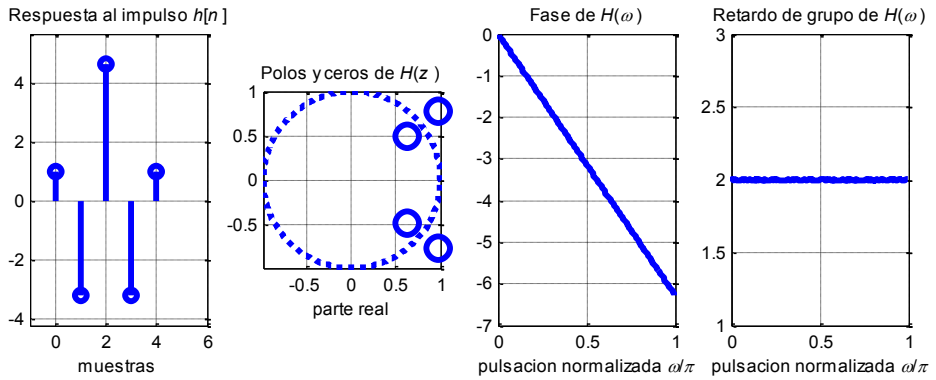
Dependiendo del tipo de simetría de $h[n]$ (par o impar) y del valor de M (par o impar), podemos definir cuatro tipos de sistemas FIR de fase lineal.

		Simetría	
		Par	Impar
Valor de M	Par	Tipo I	Tipo III
	Impar	Tipo II	Tipo IV

Ejemplos de cada uno:

Tipo I. M par y simetría par

```
ceros=[0.8*exp(-j*pi*2/3),0.8*exp(j*pi*/3),1/0.8*exp(-j*2*pi/3),
1/0.8*exp(j*pi*2/3)];
h=poly(ceros);
[H,w]=freqz(h,1,255);
Gd=grpdelay(h,1,255);
```

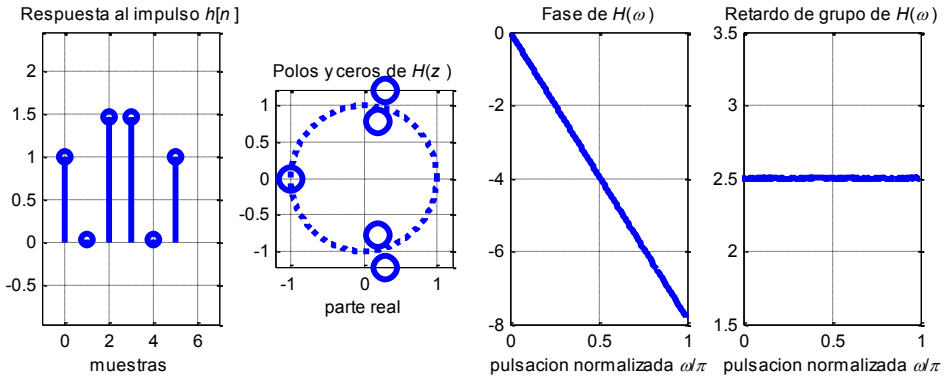


TipoII. M impar y simetría par

Tiene un cero en z=-1. Consecuencia práctica de lo anterior es la imposibilidad de realizar filtros paso alto de fase lineal tipo II.

Ejemplo

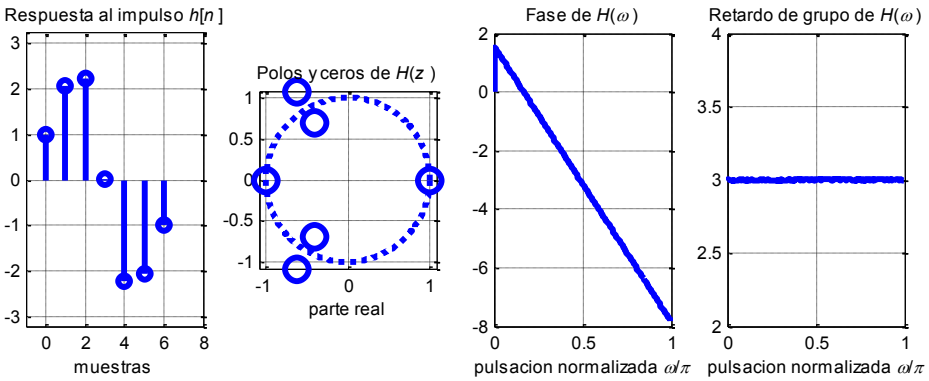
```
ceros=[0.8*exp(-j*pi*4/3),0.8*exp(j*pi*4/3),1/0.8*exp(-j*pi*4/3),
1/0.8*exp(j*pi*4/3) -1];
h=poly(ceros);
[H,w]=freqz(h,1,255);
Gd=grpdelay(h,1,255);
```



TipoIII. M par y Simetría impar

En este caso tiene ceros en z=1 y z=-1.

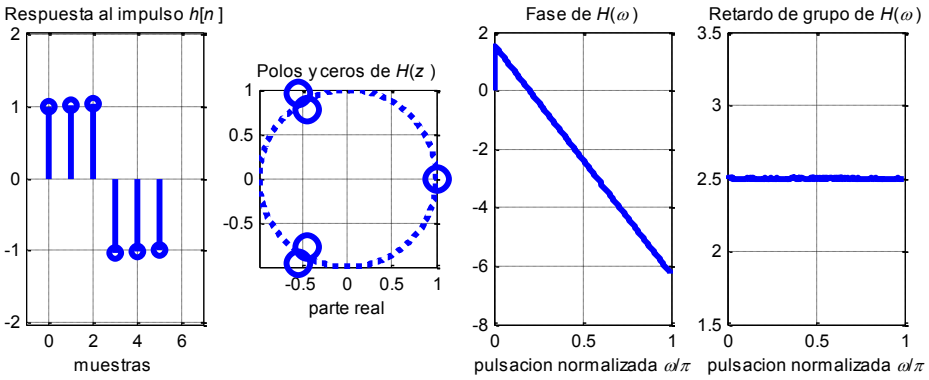
```
ceros=[0.8*exp(-j*4*pi/3),0.8*exp(j*4*pi/3),1/0.8*exp(-j*4*pi/3),
1/0.8*exp(j*4*pi/3) 1 -1];
h=poly(ceros);
[H,w]=freqz(h,1,255);
Gd=grpdelay(h,1,255);
```



TipoIV. M impar y Simetría impar

Tiene un cero en z=1, por lo tanto no es recomendable realizar filtros paso bajo de fase lineal Tipo IV.

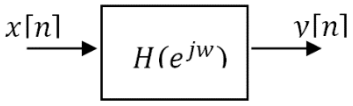
```
ceros=[0.9*exp(-j*2*pi/3)  0.9*exp(j*2*pi/3)  1/0.9*exp(-j*2*pi/3)
1/0.9*exp(j*2*pi/3) 1];
h=poly(ceros);
[H,w]=freqz(h,1,255);
Gd=grpdelay(h,1,255);
```

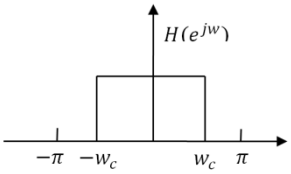


2.5 Problemas de Aula

-Problema 2.1

Sea el sistema:





¿Existe alguna $x[n]$ para que:

$$y[n] = \begin{cases} 1 & 0 \le n \le 10 \\ 0 & resto \end{cases}$$ con $M = 11$ muestras?

- Problema 2.2

Para obtener la distorsión en frecuencia H(z) del canal de transmisión, comprobamos que cuando transmitimos una delta:

$$x[n] = [1; 0; 0; 0; 0; 0],$$

se recibe $\Rightarrow t[n] = [1; 1.5 \cdot sqrt(2); 2.25; 0; 0; 0]$

a) Calcular la H(z) del canal de transmisión. ¿es FIR o IIR?

b) Dibujar su diagrama de polos y ceros. ¿Es de fase mínima?

c) Dibuje la magnitud y fase del sistema H(z). ¿Es pasobajo, pasoalto...?

NOTA: recuerde que:

	0	45	90	135	180
cos	0	sqrt(2)/2	1	-sqrt(2)/2	0
sin	1	sqrt(2)/2	0	sqrt(2)/2	0

- Problema 2.3

Un proceso de distorsión de señal convolutivo se puede modelar como:

$$x[n] \rightarrow \boxed{h[n] = \begin{cases} 1 & 0 \le n \le M-1 \\ 0 & resto \end{cases}} \rightarrow y[n]$$

Para recuperar la señal $x[n]$ a partir de $y[n]$:

a) Un primer procedimiento sería calcular el filtro inverso de $h[n]$ y filtrar $y[n]$ por el filtro inverso. ¿Cuáles son los problemas prácticos de esta aproximación?. Demuestre sus afirmaciones.

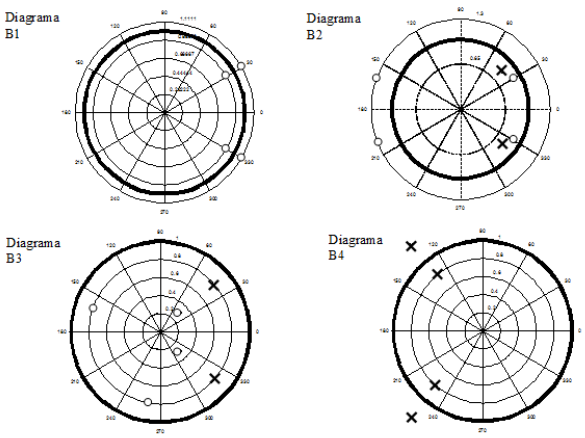
b) Se sugiere una segunda aproximación

$$y[n] \rightarrow \boxed{h_1[n] = \sum_{k=0}^{Q} \delta[n - kM]} \rightarrow \boxed{h_2[n] = \delta[n] - \delta[n-1]} \rightarrow w[n]$$

¿bajo qué condiciones (valor de Q y M) puedo recuperar $x[n]$ a partir de $w[n]$?. Demuestre sus afirmaciones.

- Problema 2.4

De los diagramas de polos y ceros B1, B2, B3 y B4, indique razonadamente si corresponden a filtros FIR o IIR, si son de fase lineal o no, si son de fase mínima o no, si son de coeficientes reales o no, y dibuje la región de convergencia del H(z) causal y estable



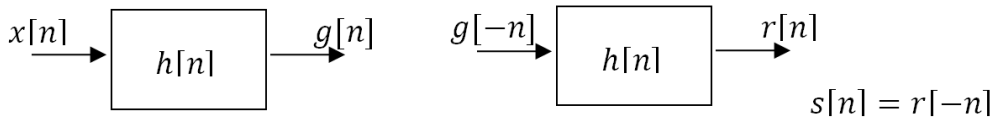
- Problema 2.5

Los métodos clásicos de diseño de filtros sólo tienen en cuenta la magnitud, sin tener en cuenta la fase. Sin embargo, casi siempre quieres tener fase nula o lineal.

En este problema se presentan varios métodos de una técnica utilizada para resolver el anterior problema cuando la secuencia de datos es de duración finita y está almacenada.

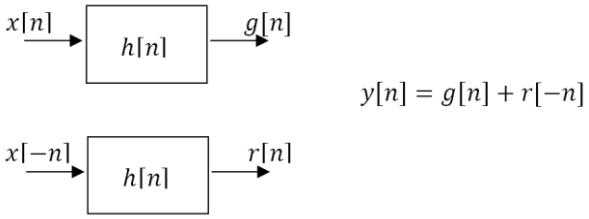
La técnica consiste en utilizar el mismo filtro dos veces y los métodos los siguientes. Asumiendo $h[n]$ la respuesta al impulso de un filtro causal, real y fase arbitraria con transformada de Fourier $H(w)$.

Método A:



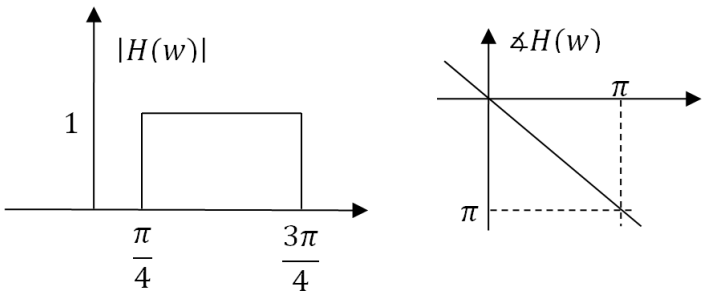
a) Calcular la respuesta impulsiva $h_1[n]$ que relaciona s[n] con $x[n]$. Mostrar $H_1(w)$ que es de fase cero.

Método B:



b) Calcular la respuesta global $h_2[n]$ que relaciona y[n] con $x[n]$.

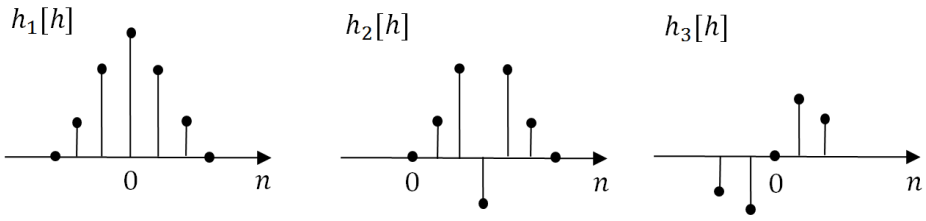
Si tenemos $h[n]$ tal que:



c) Dibujar $|H_1(w)|$ y $|H_2(w)|$ de acuerdo a los métodos A y B.

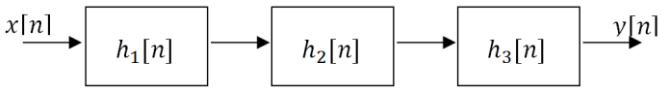
- Problema 2.6

Sean:

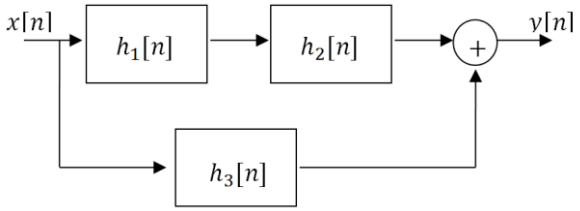


Determinar si son de fase lineal generalizada los siguientes sistemas:

a)



b)



2.6 Ejercicios Prácticos

Método para cifrado de voz

Los métodos de cifrado para modificar la voz y hacerla ininteligible has sido motivo de investigación durante varias décadas. Un método sencillo de cifrar la voz es modificar el espectro de la voz mediante un filtro lineal. En éste apartado se estudia un filtro lineal útil para cifrado. El filtro de cifrado es $H_{cifrado}(z)$

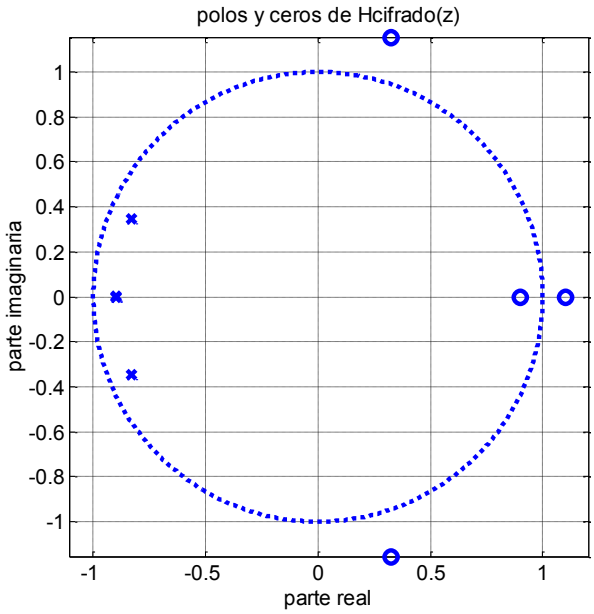
$$H_{cifrado}(z) = \frac{\sum_{k=0}^{M} b_k^{cifrado} z^{-k}}{\sum_{k=0}^{N} a_k^{cifrado} z^{-k}}$$

$b^{cifrado} = [1.0000 \quad -2.6515 \quad 3.7329 \quad -3.5249 \quad 1.4256]$, y

$a^{cifrad} = [1.0000 \quad 3.4630 \quad 4.6134 \quad 2.8050 \quad 0.6561]$

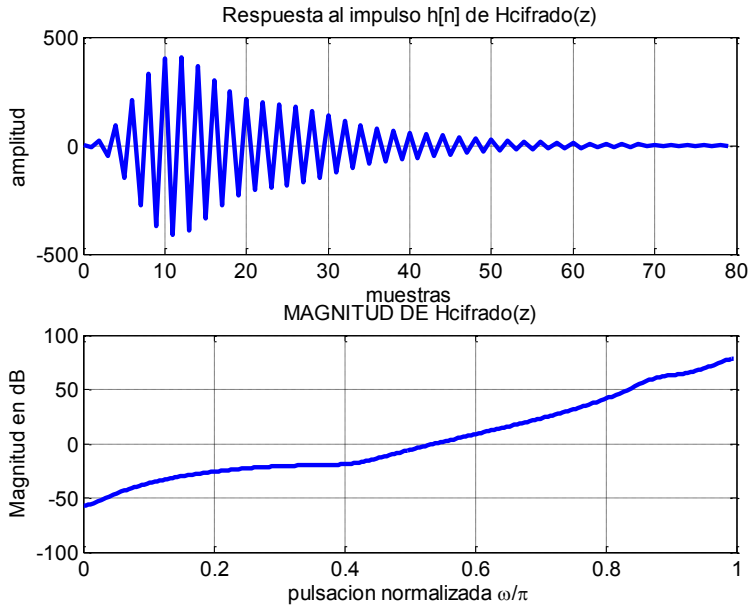
Para entender cómo $H_{cifrado}(z)$ modifica la voz se procede a su análisis. Obtenga sus ceros y polos (módulo y fase en grados) con la ayuda de las funciones *roots*, *abs* y *angle* de Matlab y sin utilizar la función *tf2zp*. Dibuje el diagrama de polos y ceros de $H(z)$, utilizando la función *zplane*.

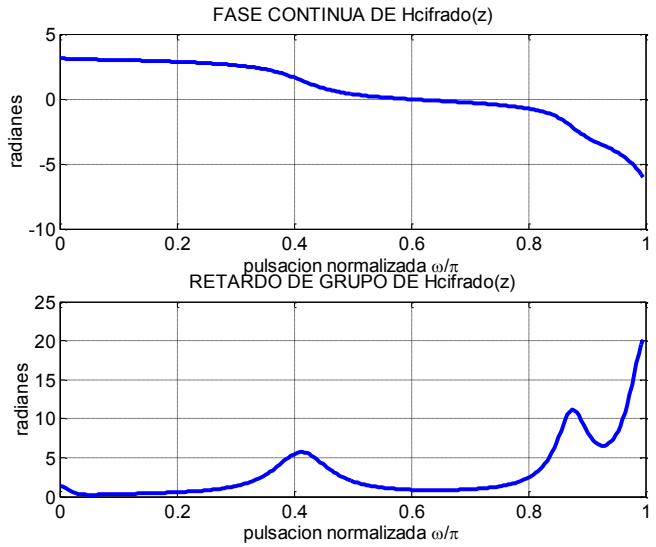
```
modulo y fase de los ceros de Hcifrado(z)
    1.2000   74.2500
    1.2000  -74.2500
    1.1000         0
    0.9000         0
modulo y fase de los polos de Hcifrado(z)
    0.9000  157.5000
    0.9000 -157.5000
    0.9000  180.0000
    0.9000 -180.0000
```



Calcule la respuesta al impulso $h[n]$ de $H(z)$ utilizando la función *filter* de Matlab. Calcule la magnitud, fase continua y retardo de grupo de $H(z)$. Puede utilizar, entre otras, las funciones *freqz*, *grpdelay* y *unwrap* de Matlab.

Dibuje a la vez y en gráficas diferentes la respuesta al impulso, la magnitud, fase continua y retardo de grupo de $H(z)$. ¿Qué operación realiza el filtro de cifrado sobre la voz?.



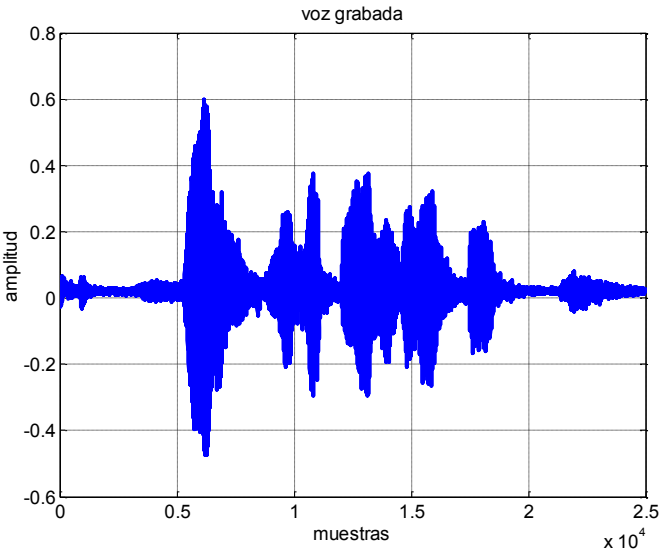


Tenga en cuenta que el filtro es IIR, por lo tanto la transformada de Fourier debe calcularse con la función *freqz*. Compare la transformada de Fourier obtenida con *freqz* y la obtenida mediante la *fft* de $h[n]$ con distintos número de muestras.

Grabación de la señal de voz a cifrar

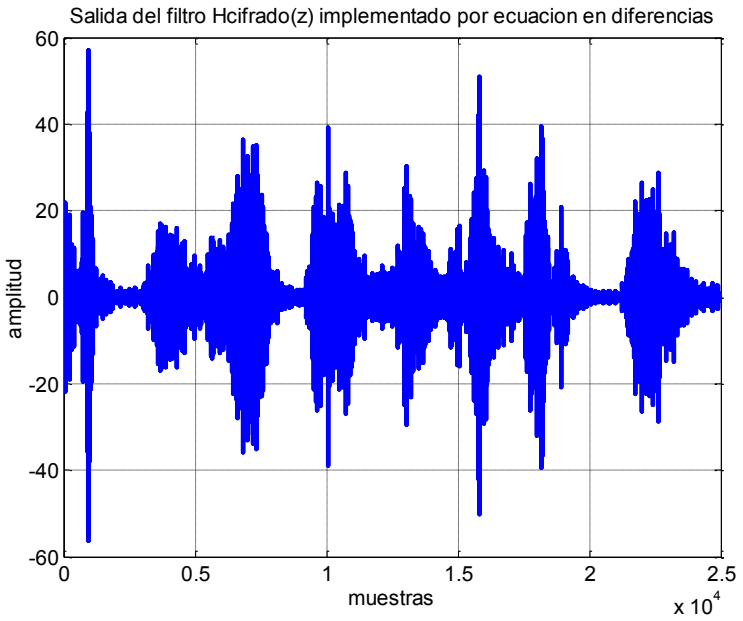
Para realizar la operación de filtrado, debe filtrar la voz grabada por el filtro $H_{cifrado}(z)$. En la teoría se han visto tres procedimientos diferentes para realizar la operación de filtrado: mediante la convolución, implementando la ecuación en diferencias, y mediante el modelo de variables de estado.

Para ello, y mediante la correspondiente utilidad de Matlab, grabe una frase de voz de aproximadamente 3 segundos. Realice la grabación con frecuencia de muestreo 8000 muestra por segundo, con 16 bits por muestra. Verifique su correcta grabación escuchándola con el comando *soundsc*. Asigne al resultado la variable *voz*.



Realización de la operación de cifrado mediante ecuación en diferencias

Calcular la salida *vozcifradaec_dif*[*n*] del filtro *Hcifrado*(*z*) cuando la entrada es *voz* utilizando su definición en ecuación en diferencias. Para ello programe la función *mifilter* que implemente la ecuación en diferencias la cual debe funcionar como la función *filter* de Matlab. Dibuje y escuche la salida del filtro.



Calculo del sistema de fase mínima para descifrado

El sistema $H_{descfrado}(z)$ de descifrado, que recupere la señal *voz* de la voz cifrada *vozcifradaec_dif* no será más que el sistema inverso del sistema $H_{cifrado}(z)$. El problema es que el sistema inverso

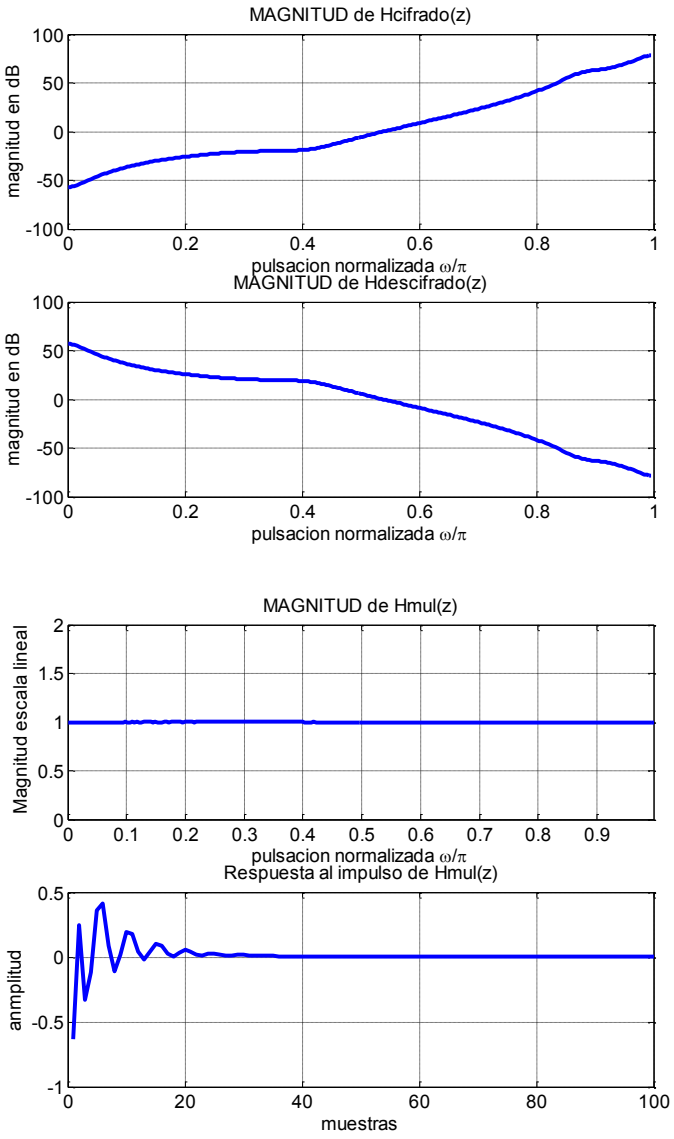
$$H_{descifrado}(z)=\frac{\sum_{k=0}^{M} b_k^{descifrado} z^{-k}}{\sum_{k=0}^{N} a_k^{descifrado} z^{-k}}=\frac{1}{H_{cifrado}(z)}=\frac{\sum_{k=0}^{N} a_k^{cifrado} z^{-k}}{\sum_{k=0}^{M} b_k^{cifrado} z^{-k}}$$

no es causal y estable pues $H_{cifrado}(z)$ tiene ceros fuera del circula unidad. La solución es calcular el sistema fase mínima de $H_{cifrado}(z)$ y utilizar como sistema de descifrado el inverso al fase mínima de $H_{cifrado}(z)$. Para verificar que esta solución es válida

Calcule el sistema de fase mínima $Hmin_{cifrado}(z)$ del sistema $H_{cifrado}(z)$. Implemente para ello la función *h2hmin* que realice esta operación de forma general (puede necesitar las funciones *poly*, *prod*, *find*, y *conj* de Matlab pero no utilice la función *polystab*).

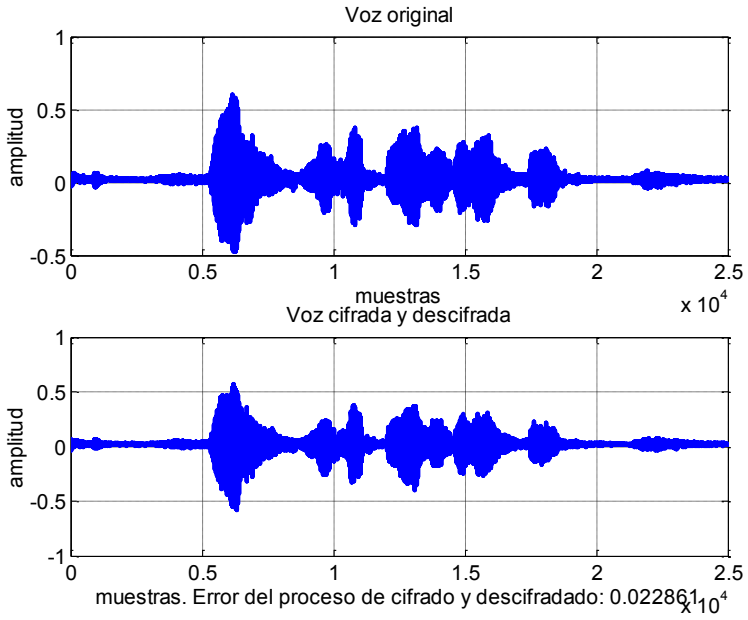
Sistema de descifrado

A partir de $Hmin_{cifrado}(z)$ obtenga $H_{descifrado}(z)$ (los coeficientes $a^{descifrado}$ y $b^{descifrado}$) como el sistema inverso de $Hmin_{cifrado}(z)$. Dibujar simultáneamente en gráficas separadas la magnitud de $H_{cifrado}(z)$, la magnitud de $H_{desifrado}(z)$. Compruebe que el sistema multiplicación $H_{mul}(z)= H_{cifrado}(z)\ H_{desifrado}(z)$ es un sistema paso todo con magnitud igual a uno (verifique que a pesar de ser $|H_{mul}(z)|=1$ su respuesta al impulso e diferente a una $\delta[n]$). En la práctica la magnitud de $H_{mul}(z)$ se obtienen filtrando una $\delta[n]$ por el sistema en cascada $H_{cifrado}(z)\ H_{desifrado}(z)$ y calculando su magnitud o multiplicado las magnitudes de $H_{cifrado}(z)$ y $H_{desifrado}(z)$, aunque en este caso pueden calcularse los coeficientes de $H_{mul}(z)$ convolucionando $a^{descifrado}$ con $a^{cifrado}$ y $b^{descifrado}$ con $b^{cifrado}$. Compare el resultado de ambos métodos.



Descifrado de la voz

Filtre la *vozcifradaec_dif* por el filtro $H_{desifrado}(z)$ para obtener la señal *vozdescifrada*. ¿se recupera la señal de voz original?. Dibuje simultáneamente *voz* y *vozdescifrada*. Obtenga el error entre ellas.



Capítulo III. Implementación de sistemas discretos LTI

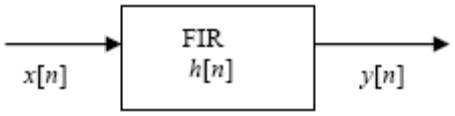
En este capítulo se abordan las técnicas clásicas de programación de filtros. Se tratarán filtros definidos por su respuesta al impulso $h[n]$ y por su ecuación en diferencias. Puesto que existen microprocesadores especializados para el procesado de señales digitales, se expondrá brevemente su arquitectura y funcionamiento y se resaltarán los aspectos a tener en cuenta al trabajar con estos microprocesadores en punto fijo.

Téngase en cuenta que al programar un filtro, el que el algoritmo funcione (lo cual es condición necesaria) no es el único aspecto a considerar. También hay que tener presente el número de operaciones que realiza el algoritmo, la memoria necesaria para las variables del algoritmo y su robustez al utilizarlo en un microprocesador de punto fijo.

Como objetivo se pretende que el alumno sea capaz de programar las diferentes estructuras de filtros FIR e IIR tanto en microprocesadores de punto flotante como en punto fijo.

3.1 Sistemas descritos por su respuesta al impulso

Éste apartado se dedica al estudio de técnicas para programar filtros definidos por su respuesta al impulso $h[n]$. La primera forma de calcular la salida es mediante convolución directa (aplicando la fórmula de la convolución suma)



con $y[n] = x[n] * h[n] = \sum_{m=0}^{L-1} x[m] \cdot h[n-m] = \sum_{m=0}^{P-1} h[m] \cdot x[n-m]$

y $h[n]$ FIR de P muestras y $x[n]$ de L muestras.

Téngase en cuenta que el resultado de la convolución son L+P-1 muestras. Entonces, el número máximo de operaciones realizado, suponiendo $L>P$, son:

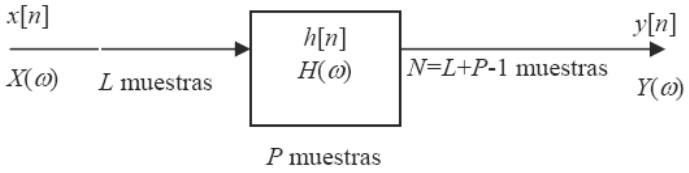
- ✓ Para cada $y[n]$ hay que realizar P multiplicaciones y P sumas, pues $y[n]=\sum_{m=0}^{P-1} h[m]\cdot x[n-m]$
- ✓ Para las L+P-1 muestras de $y[n]$, el número total de operaciones será $P(L+P-1)$ multiplicaciones y $P(L+P-1)$ sumas.

En el caso de $P>L$, el número de operaciones será $L(L+P-1)$ multiplicaciones y sumas.

Por tanto se puede decir que el número de operaciones (multiplicaciones más sumas) que se realizan en una operación de filtrado son: $2\cdot\min(L,P)\cdot(L+P-1)$.

La memoria necesaria para implementar este algoritmo son: $L+P$ datos (corresponde a guardar los P coeficientes del filtro más las L muestras de la $x[n]$ necesarias para realizar la convolución).

Una alternativa al anterior método es realizar la operación de convolución suma propia del filtrado mediante una convolución circular en el dominio de la frecuencia



Mediante la propiedad de la convolución tenemos:

$$y[n]=x[n]*h[n] \qquad \text{[Ec.43]}$$

$$Y(\omega)=X(\omega)\cdot H(\omega) \qquad \text{[Ec.44]}$$

$$Y[k]=X[k]\cdot H[k] \qquad \text{[Ec.45]}$$

Luego la IDFT de la convolución circular es:

$$IDFT \rightarrow \hat{y}[n]=\sum_{r=-\infty}^{+\infty} y[n+rN] \qquad \text{[Ec.46]}$$

donde N es el número de muestras de la DFT.

Si $N=L+P-1$, entonces un periodo de la convolución circular coincide con la convolución suma.

Esta operación utilizando la fórmula de la DFT como algoritmo de cálculo de la DFT nunca será rentable desde el punto de vista computacional. Si queremos utilizar el algoritmo FFT, que es el que hace computacionalmente rentable la utilización de la DFT en el proceso de cálculo de la convolución lineal, hemos de utilizar DFTs de longitud N igual a la siguiente potencia de 2 del valor de $L+P-1$

Luego el procedimiento de cálculo sería:

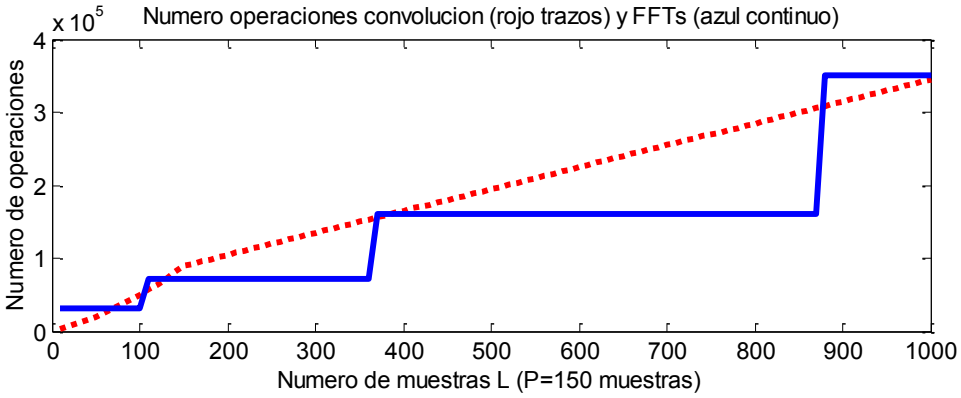
```
L=length(x);
P=length(h);
N=L+P-1;
%computacionalmente rentable si usamos FFT ⇒ M potencia de 2
N=2^nextpow2(N);
X=fft(x,N);
H=fft(h,N);
Y=X .*H;
y=real(ifft(Y));
y= y(1:L+P-1);  % este resultado debe coincidir con y=conv(x,h)
```

Veamos el número de operaciones necesarias para realizar el filtrado con FFTs. Para ello, y una vez calculado N=2^*nextpow*2($L+P$-1), veamos las operaciones que realizan cada uno de los pasos a realizar, que son:

1. La FFT directa de N muestras de la secuencia $x[n]$: $\frac{N}{2}\log_2 N$ multiplicaciones complejas y $N\log_2 N$ sumas complejas lo cual implica $2N\log_2 N$ multiplicaciones reales y $3N\log_2 N$ sumas reales, esto es: $5N\log_2 N$ operaciones.
2. La FFT directa de N muestras de $h[n]$: $5N\log_2 N$ operaciones.
3. Multiplicar 2 DFT de N muestras, lo cual implica $4N$ multiplicaciones reales y $2N$ sumas reales: $6N$ operaciones.
4. Una FFT inversa de N muestras: $5N\log_2 N$ operaciones.

Total son $15N\log_2 N + 6N$ operaciones frente a los $2\cdot\min(L,P)\cdot(L+P-1)$ de la convolución suma. Una comparativa gráfica de ambas cargas computacionales se tiene a continuación.

```
P=150;
L=[10:10:1000];
h=randn(P,1);
nopconv=zeros(size(L));
nopFFT=zeros(size(L));
for i=1:length(L),
   x=randn(L(i),1);
   conv(x,h);
   nopconv(i)=2*min(L(i),P)*(L(i)+P-1);
   N=2^nextpow2(L(i)+P-1);
   ifft(fft(x,N).*fft(h,N));
   nopFFT(i)=15*N*log2(N)+6*N;
end;
```



Como puede observarse, casi siempre compensa realizar la FFT y operar en el dominio de la DFT: a mayor longitud de L+P-1, mayor beneficio computacional.

Generalmente, la *fft*(h,N) se suele calcular en la fase de diseño del filtro y se almacena en memoria (el número de operaciones para obtener $H[k]$ no debería contabilizarse como operaciones a realizar en el proceso de filtrado, aunque en este capítulo si los contabilizamos para considerar el caso peor).

Para reducir operaciones se puede calcular eficientemente $X[k]$ si utilizamos el algoritmo de cálculo de una secuencia real de longitud N como una DFT de una secuencia compleja de longitud $N/2$, aunque esto no se considerará en los cálculos que realicemos en este capítulo.

En cuanto a la memoria utilizada, es mayor pues requiere los N datos de la $x[n]$, los $2N$ datos de la $X[k]$ (son número complejos), los $2N$ datos de la $H[k]$, los $2N$ datos de la $Y[k]$; y los N datos de la $y[n]$; total: $8N$ datos.

Cabe preguntarse ahora ¿qué pasaría cuando $x[n]$ es ilimitada? ¿Se debe esperar a que llegue toda la señal $x[n]$ para comenzar a filtrar ?. La solución más sencilla sería trocear $x[n]$, filtrar cada trozo por el filtro $h[n]$, y unir las salidas de cada trozo filtrado.

Para ello existen diversos métodos. En estos apuntes se describe el solape-suma mediante un ejemplo.

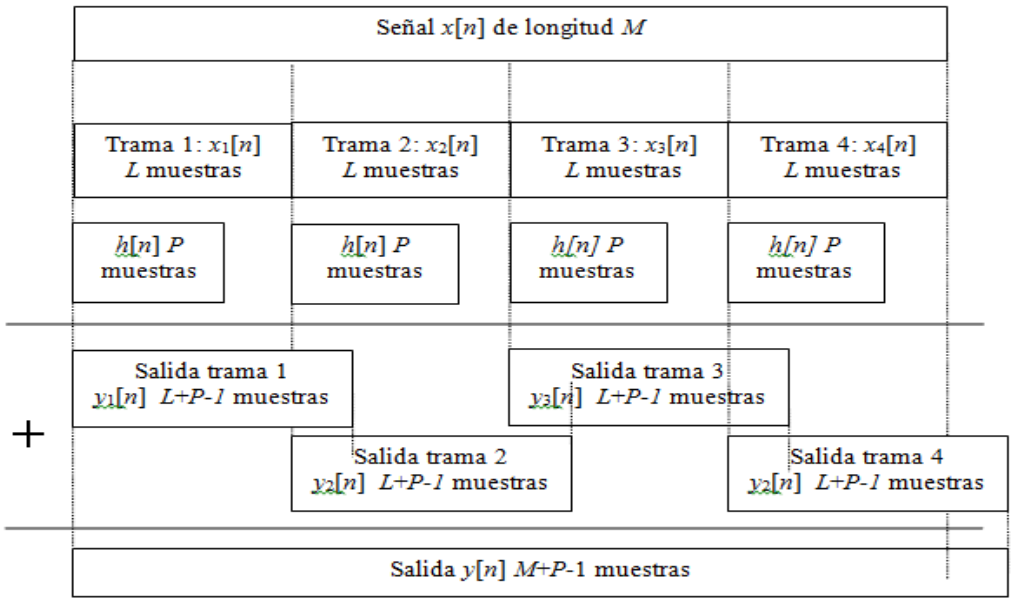
Como referencia, suponga que filtra la secuencia $x[n]=[x[0]\ \ x[1]\ \ x[2]\ \ x[3]\ \ x[4]\ \ x[5]]=[1\ 2\ 1\ 3\ 5\ 2]$ de longitud $M=6$ por un filtro con respuesta al impulso $h[n]=[1\ 2\ 1]$ de longitud $P=3$.

La salida $y[n]=x[n]*h[n]=\sum_{k=-\infty}^{k=+\infty}x[k]h[n-k]$ serán las siguientes $M+P-1=8$ muestras

$y[0]=h[0]x[0]$
$y[1]=h[0]x[1]+h[1]x[0]$
$y[2]=h[0]x[2]+h[1]x[1]+h[2]x[0]$
$y[3]=h[0]x[3]+h[1]x[2]+h[2]x[1]$
$y[4]=h[0]x[4]+h[1]x[3]+h[2]x[2]$
$y[5]=h[0]x[5]+h[1]x[4]+h[2]x[3]$
$y[6]=\qquad h[1]x[5]+h[2]x[4]$
$y[7]=\qquad\qquad h[2]x[5]$

El método solape suma divide la señal $x[n]$ de longitud M en trozos (tramas) de longitud L y calcula la salida (convolución lineal) de cada trama de forma independiente. Evidentemente, la salida de cada trama tendrá $L+P-1$ muestras. Hay que estudiar cómo concatenar las salidas de cada trama para que el resultado final $y[n]$ sea de longitud $M+P-1$.

Parece lógico pesar que si de cada L muestras de la trama se obtienen $L+P-1$ muestras, para que el resultado final no exceda las $M+P-1$ muestras habrá que solapar las salidas de tramas consecutivas $P-1$ muestras.



Para verificar la anterior afirmación con el ejemplo: se trocea la señal de entrada $x[n]$ en tramas no solapadas de longitud L=3

Primer tramo $x_1[n]=[x[0]\ x[1]\ x[2]]=[1\ 2\ 1]$

Se calcula la convolución lineal $x_1[n]*h[n]=y_1[n]$

$y_1[0]=h[0]x[0]$
$y_1[1]=h[0]x[1]+h[1]x[0]$
$y_1[2]=h[0]x[2]+h[1]x[1]+h[2]x[0]$
$y_1[3]=h[1]x[2]+h[2]x[1]$
$y_1[4]=h[2]x[2]$

Segundo tramo $x_2[n]=[x[3]\ x[4]\ x[5]]=[3\ 5\ 2]$

Se calcula la convolución lineal $x_2[n]*h[n]=y_2[n]$

$y_2[0]=h[0]x[3]$
$y_2[1]=h[0]x[4]+h[1]x[3]$
$y_2[2]=h[0]x[5]+h[1]x[4]+h[2]x[3]$
$y_2[3]=h[1]x[5]+h[2]x[4]$
$y_2[4]=h[2]x[5]$

Hay que obtener $y[n]$ a partir de $y_1[n]$ e $y_2[n]$. Para ello se solapan P-1 muestras y suman obteniendo:

	$y_1[0]$	$y_1[1]$	$y_1[2]$	$y_1[3]$	$y_1[4]$			
+				$y_2[0]$	$y_2[1]$	$y_2[2]$	$y_2[3]$	$y_2[4]$
	$y[0]$	$y[1]$	$y[2]$	$y[3]$	$y[4]$	$y[5]$	$y[6]$	$y[7]$

pues los términos de $y[3]$ que le faltan a $y_1[3]$ los tiene $y_2[0]$ y los términos de $y[4]$ que le faltan a $y_1[4]$ los tiene $y_2[1]$. Por lo tanto el procedimiento solape suma queda de la siguiente manera: se trocea la señal $x[n]$ en tramas consecutivas no solapadas de longitud L, se filtra cada trama de longitud L por el filtro $h[n]$ de longitud P, obteniendo la salida de cada trama de longitud $L+P$-1. Las diferentes salidas de cada trama se concatenan solapándose y sumándose P-1 muestras. Generalizando:

1. Se trocea la señal en tramas $x_k[n]$ no solapadas de longitud L, donde

   $x_k[n]=x[kL:(k+1)L-1]$ (se supone $x[n]=0$, $n<0$)

2. Se calcula la convolución lineal

   $y_k=conv(x_k,h)$ ó $y_k=ifft(fft(x_k,L+P-1).*fft(h,L+P-1))$

3. Se forma la salida solapando y sumando las salidas de cada trama

$$y[n] = \sum_{k=0}^{k=+\infty} y_k[n - kL]$$

3.2 Implementación de sistemas lineales a partir de su ecuación en diferencias

Método de programación de la ecuación en diferencias

Se presenta el método a utilizar mediante un ejemplo sencillo:

Suponga que debe programar el filtro con función de transferencia

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + az^{-1}}$$

su ecuación en diferencias es:

$y[n]=b_0x[n]+b_1x[n\text{-}1]\text{-}ay[n\text{-}1]$

Este algoritmo se puede programar en Matlab de la siguiente manera:

```
load ficherodatos.mat b a x
xini=0;
yini=0;
for n=1:length(x)
      y[n]=b(1)*x[n]+b(2)*xini-a*yini;
      xini=x[n];
      yini=y[n];
end
```

En este algoritmo es fácil ver que el programa utiliza tan sólo tres operadores para calcular la salida: multiplicación, suma, y retardo (ocupación de posiciones en memoria)

Con la finalidad de programar el mismo sistema con otro algoritmo de manera que se reduzca el número de operaciones, los requerimientos de memoria, y sea más robusto (se degrade menos) al pasarlo a punto fijo, se recurre a los diagramas de flujo mediante el siguiente proceso:

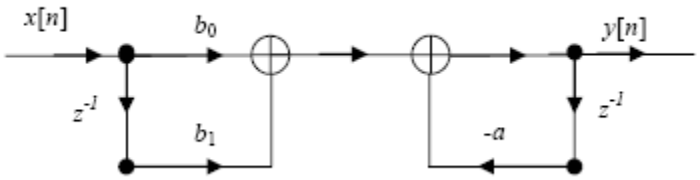
1. Se define $H(z)$.
2. Se obtiene su ecuación en diferencias.
3. Se dibuja su diagrama de flujo.
4. Se modifica el diagrama de flujo de forma que el diagrama resultante realice la misma función que el original ahorrando operaciones y posiciones de memoria.
5. Se programa el nuevo diagrama de flujo comprobando que ahorra operaciones y posiciones de memoria.
6. Se comprueba que el sistema implementado por el nuevo diagrama de flujo mantiene la $H(z)$ original.

Ejemplo del proceso definido

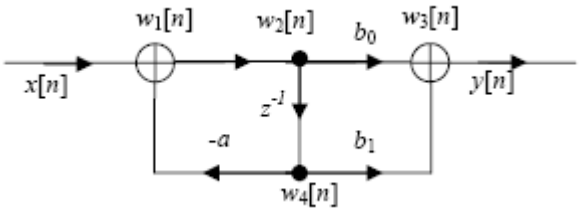
Se parte de $H(z)=\dfrac{b_0+b_1z^{-1}}{1+az^{-1}}$

Su ecuación en diferencias es $y[n]=b_0x[n]+b_1x[n\text{-}1]\text{-}ay[n\text{-}1]$

Su diagrama de flujo se puede dibujar como sigue



Dicho diagrama se modifica para obtener el siguiente:



Para programarlo se escriben sus ecuaciones:

$w_1[n]=x[n]\text{-}aw_4[n]$
$w_2[n]=w_1[n]$
$w_3[n]=b_0w_2[n]+b_1w_4[n]$
$w_4[n]=w_2[n\text{-}1]$
$y[n]=w_3[n]$

se opera con el fin de calcular su algoritmo (cuidado con no modificar el diagrama de flujo), obteniéndose:

$w_2[n]=w_1[n]=x[n]\text{-}aw_4[n]$
$y[n]=w_3[n]=b_0w_2[n]+b_1w_4[n]$
$w_4[n]=w_2[n\text{-}1]$

el nuevo algoritmo se programa:

```
% condiciones iniciales
w2=0;
% recursión
for n=1:length(x)
      w4=w2;
      w2=-a*w4+x[n];
      y(n)=b0*w2+b1*w4;
end
```

Para comprobar que la $H(z)$ del sistema no ha sido modificada, se sigue operando con las ecuaciones del diagrama de flujo con el fin de obtener la $H(z)$ del diagrama de flujo

$$w_2[n]=x[n]-aw_4[n]= x[n]-aw_2[n-1]$$
$$y[n]= b_0w_2[n]+b_1w_4[n]= b_0w_2[n]+b_1w_2[n-1]$$

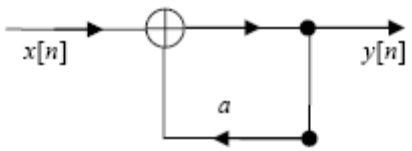
se calcula su Transformada Z y se obtiene

$$W_2(z)= X(z)-az^{-1}W_2(z)$$
$$Y(z)= b_0W_2(z)+b_1z^{-1}W_2(z)$$

Sustituyendo la primera ecuación en la segunda se obtiene

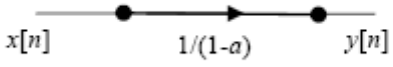
$$H(z)=\frac{b_0+b_1z^{-1}}{1+az^{-1}}$$

Precaución: Si un diagrama tiene un lazo sin retardo, su algoritmo no se puede implementar



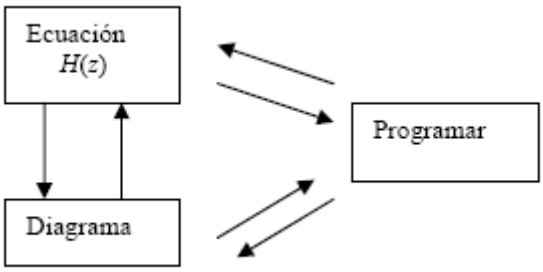
pues su ecuación en diferencias $y[n]=ay[n]+x[n]$ no se puede programar.

Ojo, pues si manipulamos la anterior ecuación y la programamos como y[n]=$x[n]/(1-a)$ hemos cambiado el diagrama de flujo:



Norma para no cambiar el diagrama de flujo: Se puede sustituir pero no pasar al otro lado de la igualdad.

En resumen, hay que saber



En lo que resta de capítulo veremos cómo se puede modificar un diagrama de flujo sin cambiar la $H(z)$ del sistema al que representan.

Estructuras básicas para sistemas IIR: forma directa I, II y traspuestas

Forma directa I

Partimos del sistema general:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \qquad \text{[Ec.47]}$$

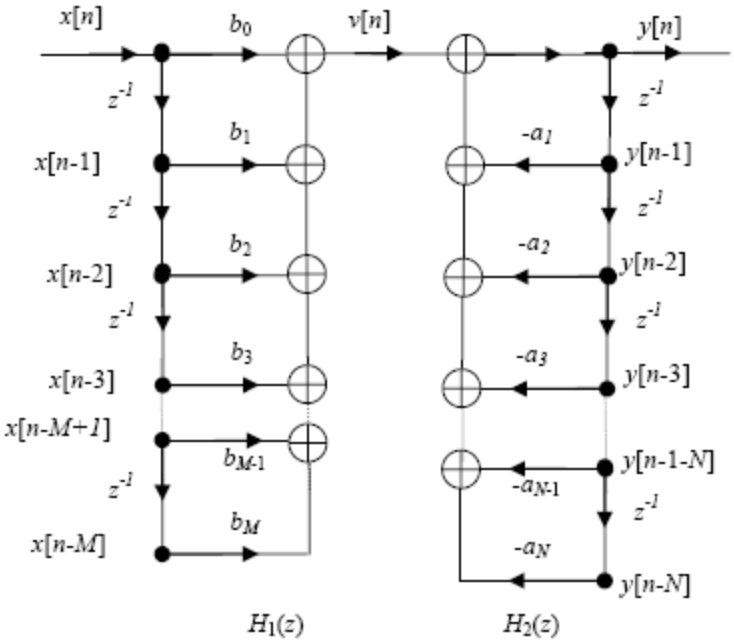
Cuya ecuación en diferencias se puede escribir como:

$$y[n] + \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k] \qquad \text{[Ec.48]}$$

de donde:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k] \qquad \text{[Ec.49]}$$

representamos su diagrama de flujo:



A este diagrama de flujo se le denomina forma directa I. Como se puede obtener del diagrama, requiere $N+M$ retardos (posiciones de memoria ocupadas por las condiciones iniciales) y $N+M+1$ multiplicaciones y sumas reales para programarse.

Forma directa II

Con la finalidad de ahorrarnos posiciones de memoria, escribimos las ecuaciones del diagrama de flujo de la siguiente manera

$$v[n]=\sum_{k=0}^{M} b_k x[n-k]$$

$$y[n]=v[n]-\sum_{k=1}^{N} a_k y[n-k]$$

Calculamos su transformada Z obteniendo

$$V(z)=X(z)\sum_{k=0}^{M} b_k z^{-k} = X(z)H_1(z)$$

$$Y(z)=V(z)\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}} = V(z)H_2(z)$$

donde $H_1(z)$ y $H_2(z)$ tienen su correspondencia en el gráfico anterior.

Sustituyendo la primera ecuación en la segunda se tiene

$Y(z)=X(z)H_1(z)H_2(z)$

utilizando las propiedades de los sistemas LTI

$Y(z)=X(z)H_2(z)H_1(z)$

lo cual sugiere la posibilidad de reordenar el diagrama de flujo de tal manera que obtenemos un algoritmo de cálculo diferente para implementar el mismo sistema, pues la última ecuación se puede escribir

$Y(z)=W(z)\ H_1(z)$ *s*iendo $W(z)=X(z)H_2(z)$

de donde

$$W(z) = X(z)H_2(z) = X(z)\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}$$

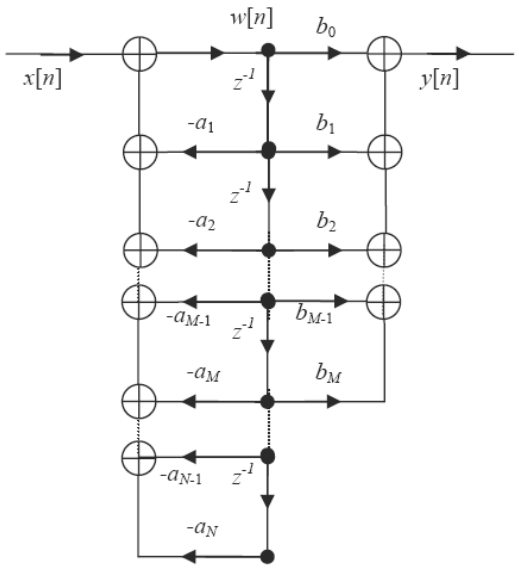
$$Y(z) = W(z)H_1(z) = W(z)\sum_{k=0}^{M} b_k z^{-k}$$

realizando la transformada Z inversa

$$w[n] = x[n] - \sum_{k=1}^{N} a_k w[n-k]$$

$$y[n] = \sum_{k=0}^{M} b_k w[n-k]$$

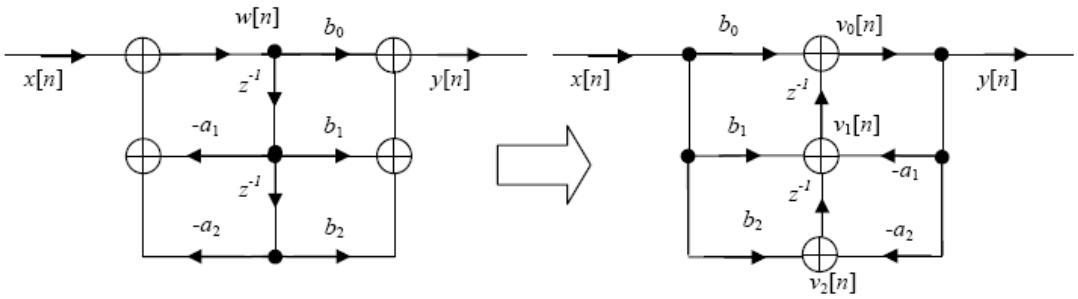
Y al representar estas ecuaciones, obtenemos el diagrama de flujo denominado forma directa II o forma canónica:



Que realiza la misma operación que la forma directa I pero con tan solo *max*(*M,N*) retardos. Así ahorramos posiciones de memoria sin alterar el número de operaciones.

Formas Traspuestas

Aprovechando la teoría de grafos empleamos procedimientos para transformar los diagramas de flujo sin variar su salida. Uno de estos procedimientos es el *diagrama de flujo traspuesto*: si en un diagrama de flujo se cambian la entrada por la salida y la dirección de todas las flechas, el sistema resultante realiza la misma función que el sistema original. Ejemplos:



Sus ecuaciones son las siguientes:

$v_0[n]=b_0[n]x[n]+v_1[n-1]$
$y[n]=v_0[n]$
$v_1[n]=-a_1y[n]+b_1x[n]+v_2[n-1]$
$v_2[n]=-a_2y[n]+b_2[n]x[n]$

Que se pueden pasar a programa de la siguiente manera:

```
% Condiciones iniciales
v1=0;v2=0;
for n=1:length(x)
      y(n)=x(n)*b0+v1;
      v1=b1*x(n)-a1*y(n)+v2;
      v2=b2*x(n)-a2*y(n);
end
```

En el diagrama de flujo se puede observar como la forma directa II realiza primero los polos y luego los ceros, mientras que la forma directa II traspuesta implementa primero los ceros y luego los polos.

Aquí, los requerimientos de memoria y de cómputo son los mismos para los diagramas de flujo original y traspuesto. La única diferencia entre ambos casos se tiene cuando se opera con aritmética de precisión finita.

La estructura que implementa el algoritmo filter de Matlab es una forma directa II traspuesta.

Forma en Cascada

Otra manera de programar un filtro es descomponer la función racional en multiplicación de funciones racionales de orden 2 e implementarlo como una cascada de filtros.

Matemáticamente H(z) puede expresarse como:

$$H(z)=\frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}}=A\frac{\prod_{k=1}^{M_1}(1-g_k z^{-1})\prod_{k=1}^{M_2}(1-h_k z^{-1})(1-h_k^* z^{-1})}{\prod_{k=1}^{N_1}(1-c_k z^{-1})\prod_{k=1}^{N_2}(1-d_k z^{-1})(1-d_k^* z^{-1})} \qquad \text{[Ec.50]}$$

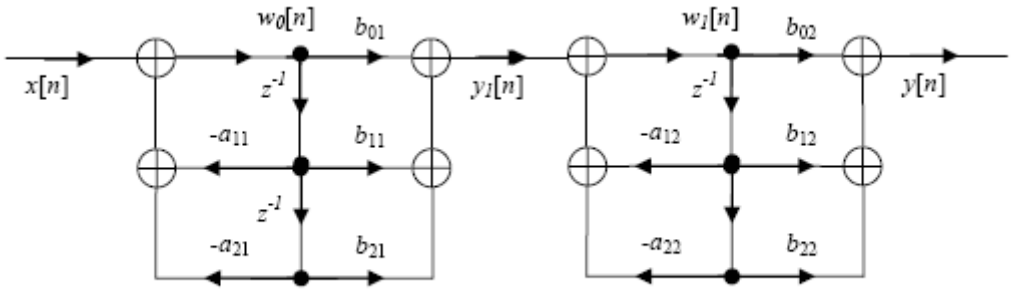
con $M=M_1+2M_2$, $N=N_1+2N_2$, g_k los ceros reales, h_k los ceros complejos, c_k los polos reales, y d_k los polos complejos

De la anterior expresión se puede obtener $H(z)$ como una cascada de sistemas de segundo orden como sigue:

$$H(z)=\prod_{k=1}^{N_s}\frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}} \qquad \text{[Ec.51]}$$

siendo $N_s=\begin{cases}\frac{N}{2} & N \quad par\\ \frac{N+1}{2} & N \quad impar\end{cases}$

Cada sistema de la cascada se puede implementar con la forma que se desee: directa I, directa II o traspuestas. Así, un sistema de $M=N=4$ implementando en cascada donde cada sistema de la cascada en la forma directa II quedaría



En este caso, cada sistema de la cascada requiere 5 multiplicaciones y sumas y 2 retardos, lo cual hace un total de (para $N=M$ y N par) de $5N/2$ multiplicaciones y sumas, y N retardos. Esto supone un aumento de carga computacional de la forma en cascada

respecto a las formas directas, aunque su robustez frente a su implementación en punto fijo la hacen ampliamente aplicable y utilizada.

Estructuras básicas para sistemas FIR

Un filtro FIR responde a la ecuación en diferencias con N=0

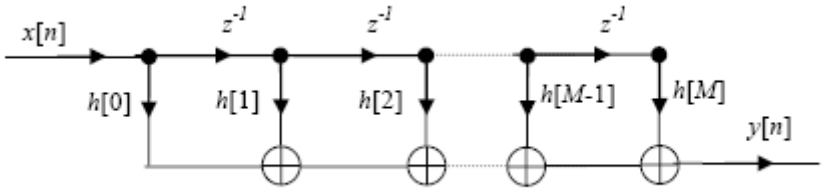
$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \qquad \text{[Ec.52]}$$

de donde su respuesta es $h[n] = \begin{cases} b_n & n = 0,1,...,M \\ 0 & resto \end{cases}$

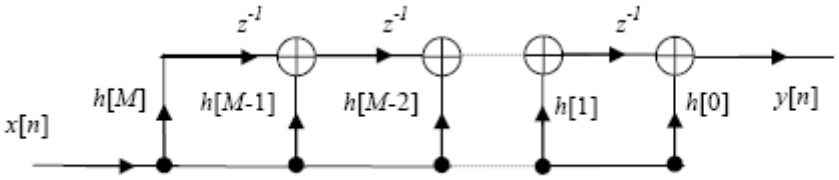
En los filtros FIR, al no tener polos, la forma directa I y II coinciden.

Si dibujamos su diagrama de flujo obtenemos:

Forma directa



A esta estructura se les conoce como línea de retardos o filtro transversal. Su forma es traspuesta



También se pueden descomponer en cascada o paralelo

3.3 Implementación en microprocesadores para procesado digital de señal

Generalmente las operaciones de filtrado se realizan en tiempo real con microprocesadores específicos para ello, denominados DSP (Digital Signal Processor). Estos microprocesadores tienen una aritmética y una arquitectura pensada para filtrar una secuencia de números proveniente de un conversor continuo/discreto y enviar el resultado al conversor discreto/continuo. Por su concepción, incluyen memoria interna y

todo el hardware necesario para que las tarjetas que lo utilizan apenas incluyan la alimentación, señal de reloj y los conversores.

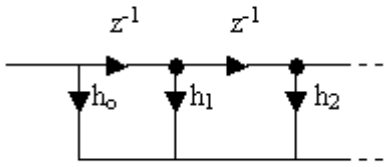
Introducción a los µDSP (Microprocesadores para procesado de señales)

Los DSP se diferencian de los microsprocesadores de propósito general en su aritmética y arquitectura, pues están pensados para programar filtros.

Respecto a la arquitectura están preparados para realizar la operación del filtro transversal:

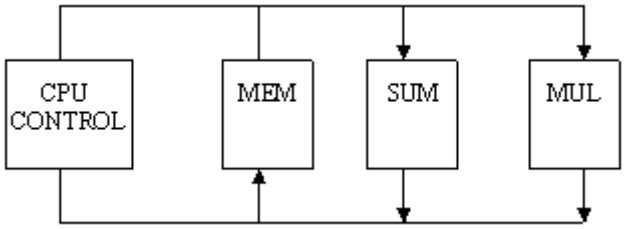
El filtro se basa en la operación:

$$a_o = a_o + m_2 * m_3$$



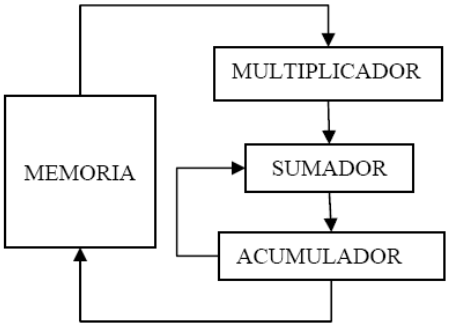
Requiere:
- -Buscar instrucción.
- -Buscar operandos.
- -Multiplicar. → 4 Accesos a memoria
- -Sumar. + MUL.
- -Desplazar. + SUM.

Partiendo de la arquitectura clásica



se requieren 6 ciclos (4 de acceso a memoria + 2 de operación).

Para reducir el tiempo MUL/SUM se modifica de la siguiente manera:



reduciéndose la anterior operación a 5 ciclos (4 de acceso a memoria + 1 de operación), que se reducen a 1 añadiendo una memoria en paralelo, permitiendo 2 accesos a memoria por ciclo, introduciendo la caché de instrucción y el modo de direccionamiento circular y pipeline que(distribuye la ejecución de una instrucción en varios ciclos. La filosofía inicial con el *pipeline* fue que fuera transparente para él y cuando haya un conflicto, el hardware retrasará la ejecución de una instrucción. Esto se llama *interlocking* (antibloqueo). Aunque existen estrategias para dar al programador mayor control sobre el *pipeline*: el *time-stationary coding* (codificación estacionaria en el tiempo) o el *Data-Stationary Coding* (codificación estacionaria en datos).

Respecto a la aritmética, los DSP pueden dividirse, según su aritmética en DSP de punto fijo y punto flotante. Los de punto fijo son más rápidos, más baratos y consumen menos. Los de punto flotante presentan la ventaja de no preocupar la magnitud de los números obtenidos en la programación de los filtros. Los de punto fijo tienen una longitud de palabra de *B*+1 bits. En estos casos el sumador suele ser de 2B bits, y el acumulador de aproximadamente 2B+B/3 para reducir los errores de cuantificación y probabilidad de desbordamiento.

Efectos a tener en cuenta al programar filtros en DSPs de punto fijo

A la hora de programar un filtro definido por:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \qquad \text{[Ec.53]}$$

en aritmética de punto fijo, se han de tener en cuenta los siguientes aspectos

1. Efectos sobre el filtro de la cuantificación de los coeficientes a_k y b_k. La respuesta en frecuencia cambia mucho al cuantificar si hay polos cercanos al círculo unidad o polos muy cercanos entre sí. Por eso se suele recurrir a la implementación en cascada.
2. Cuando se multiplican dos números en punto fijo de B+1 bits, el resultado tiene 2B+1 bits. Estos resultados hay que cuantificarlos todos. Así hay que estudiar el efecto del truncamiento o redondeo de los resultados de las multiplicaciones sobre el filtro
3. Cuando se suman los resultados de las multiplicaciones se puede ocasionar un desbordamiento u overflow. Hay que estudiar cómo evitar el desbordamiento.

El estudio de estos aspectos nos dará unos criterios para la elección de la estructura más conveniente.

Matlab ofrece una herramienta interesante para estudiar los efectos del punto fijo en los filtros denominada *fdatool*.

3.4 Problemas de Aula

-Problema 3.1

Sean las dos señales:

- $x[n]$ de longitud $\frac{N}{2} = 2^{\vartheta}$
- $h[n]$ de longitud $\frac{N}{2} = 2^{\vartheta}$

Se puede obtener la convolución de ambas señales de 2 maneras:

a) $y[n] = conv(x, h)$

b) $y[n] = ifft(fft(x, N).* fft(h, N))$

Obtener el número de operaciones de cada caso. ¿A partir de qué N compensa una u otra?

- Problema 3.2

Se quiere hacer una convolución lineal de:

- Secuencia de 10000 puntos.
- Con la $h[n]$ de un FIR de 100 puntos.

a) Si se utiliza el método solape-suma. ¿Cuántas DFTs e IDFTs hacen falta?

b) Calcular el número de operaciones por trama ($N = 256$)

- Problema 3.3

Algunas aplicaciones requieren calcular convoluciones circulares de $N = 63$ de dos secuencias de 63 puntos ($x[n]$ y $h[n]$).

a) Si solo se dispone de: Multiplicadores, Sumadores y algoritmos DFT de 128 muestras. Calcular $CC[n] = x[n] \otimes h[n]$.

b) Calcule ahora con DFT de 64 muestras (sugerencia: utilice el solape-suma).

c) ¿Qué es más eficiente?¿Hacer la convolución circular directamente o con DFT?

- Problema 3.4

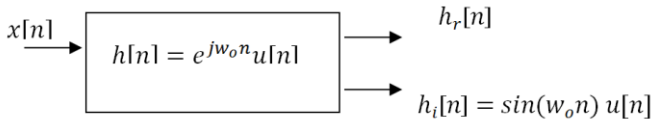
Para el filtrado de señales bidimensionales $x[n,m]$ por un filtro LTI de respuesta al impulso $h[n,m]$ hay que realizar la convolución bidimensional definida por:

$$y[i,j] = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h[n,x]x[i-n,j-m]$$

Realice un programa en Matlab que a partir de la señal $x[n,m]$ y de la respuesta al impulso $h[n,m]$ obtenga el resultado de la convolución bidimensional $y[i,j]$.

- Problema 3.5

A veces es necesario que un sistema genere una secuencia sinusoidal. Una manera de hacerlo es con un sistema como el siguiente:



Calcular la ecuación en diferencias compleja del sistema y dibujar su diagrama de flujo (con coeficientes reales).

- Problema 3.6

A partir del siguiente programa que implementa un filtro lineal e invariante en el tiempo

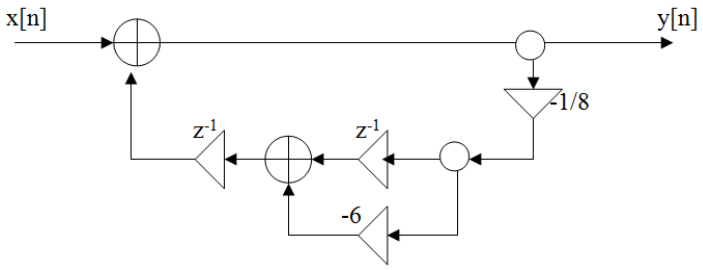
```
A=0;
B=0;
C=0;
for n=1:length(entrada)
        aux=B;
        B=A;
        D=B+C;
        C=aux+a*B;
        salida(n)=entrada(n)+D;
        A=-1/2*salida(n);
end
```

a) Calcular su H(z)

b) Dibujar su diagrama de flujo

- Problema 3.7

Del diagrama de flujo representado en la figura,

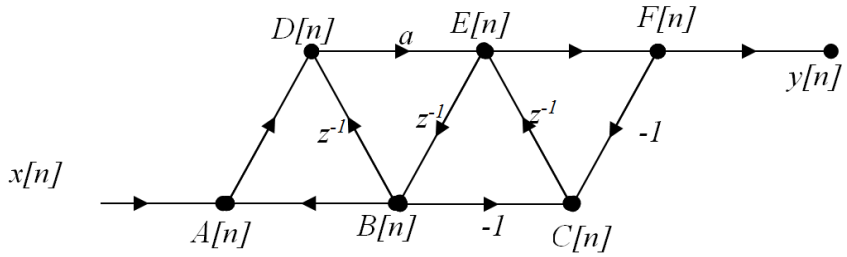


a) Realizar el programa de este filtro.

b) Calcular su H(z).

- Problema 3.8

Considere el filtro de la figura, donde las flechas sin etiqueta representan multiplicadores por uno:



Se pide

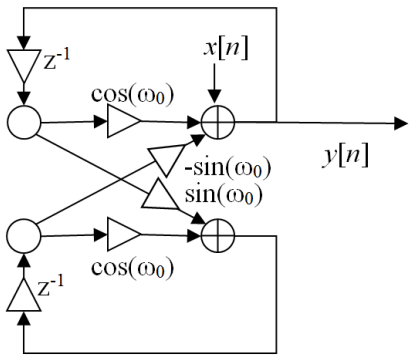
a) Analizar dicho filtro e indicar razonadamente si es realizable, y si es FIR o IIR.

b) Realizar un programa en Matlab que implemente dicha estructura con el número mínimo de operaciones y condiciones iniciales. El nombre de cada nodo está indicado en la figura.

c) Determinar la función de transferencia *H(z)* del sistema para un valor de *a* arbitrario. Verifique que si *a=1*, entonces *H(z)=1*.

- Problema 3.9

Del siguiente filtro:



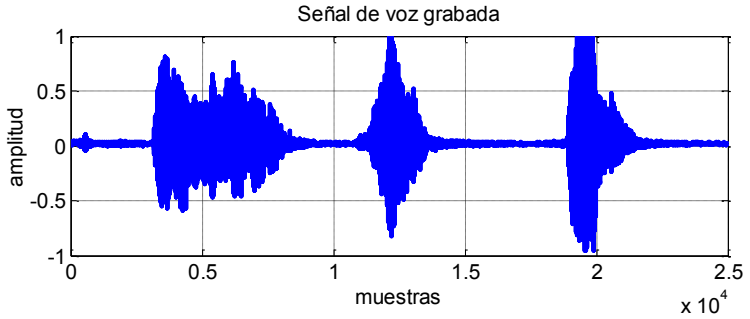
a) Realizar un programa en MATLAB que implemente dicho filtro con el mínimo número de condiciones iniciales.

b) Calcular su H(z).

3.5 Ejercicios Prácticos

Obtención de la envolvente de una señal

Para ello, mediante la correspondiente utilidad, de Matlab grabe un fichero que contenga tres palabras separadas de silencios. Para facilitar la práctica, procure decir palabras de sólo una o dos sílabas. Realice la grabación con frecuencia de muestreo 8000 muestra por segundo, con 16 bits por muestra (comando de Matlab *wavrecord*). Verifique su correcta grabación escuchándola con el comando *soundsc* y dibujándola. Asegúrese que la media es nula y normalice la voz a amplitud máxima igual a 1. Asígnele la variable *voz*.



Implementación del detector de actividad de voz con un filtro FIR

El método que seguiremos para el cálculo de la envolvente de la voz que se seguirá en esta práctica es filtrar el **módulo de la voz** (en Matlab abs(voz)) por un filtro paso bajo $H_{DAV}(z)$ definido como:

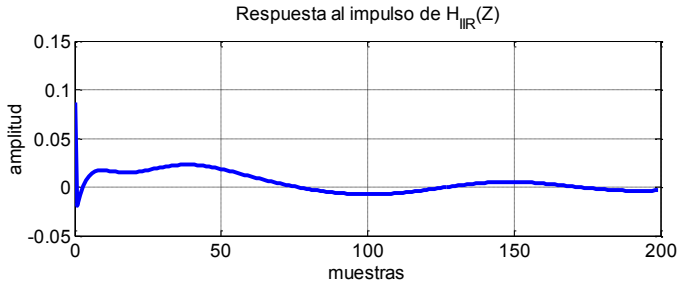
$$H_{IIR}(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}}$$

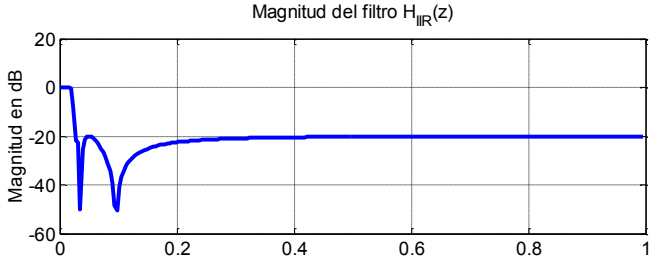
cuyos coeficientes son

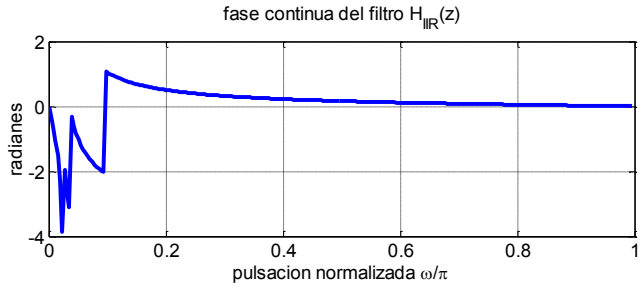
b_k=[0.0873 -0.5142 1.2714 -1.6889 1.2714 -0.5142 0.0873] y

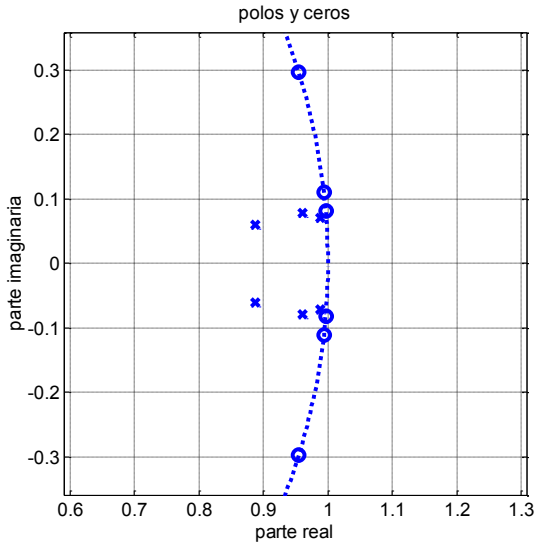
a_k=[1.0000 -5.6727 13.4173 -16.9370 12.0343 -4.5634 0.7215]

Dibujar simultáneamente y en gráficas distintas su respuesta al impulso, magnitud y fase del filtro $H_{IIR}(z)$. ¿Qué función realiza el filtro? Dibujar su diagrama de polos y ceros.



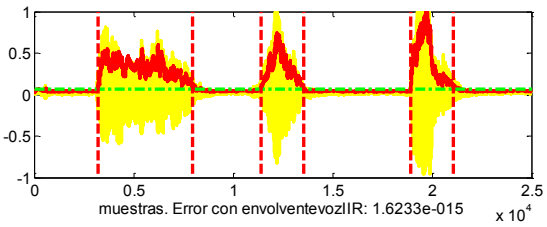






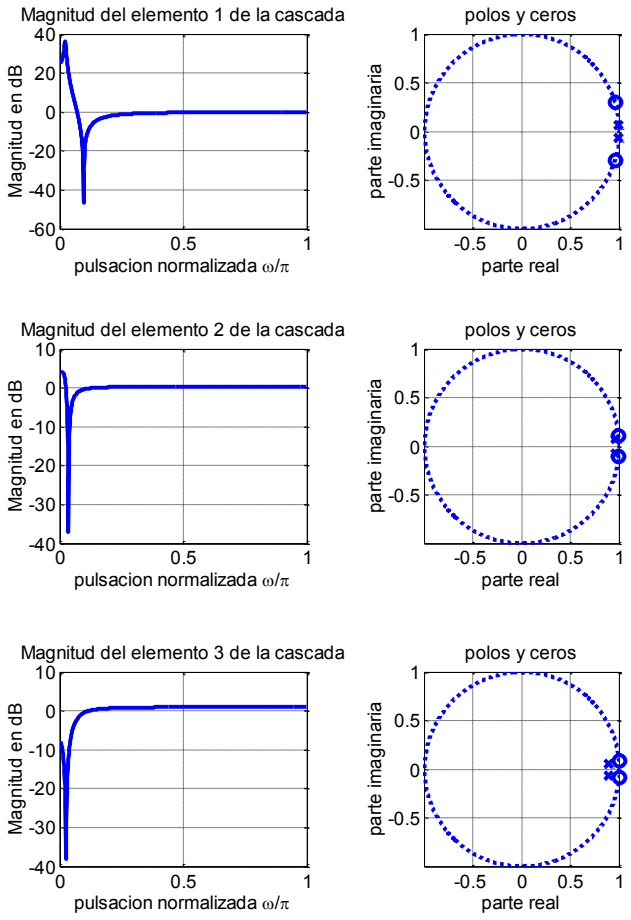
Programación con un filtro en forma directa I, II o traspuesta.

Obtenga la envolvente de la voz utilizando el filtro $H_{IIR}(z)$ implementado mediante la forma directa I, II o sus traspuesta. Compárelo con el resultado obtenido on *filter*. Dibuje la secuencia *voz* y su envolvente

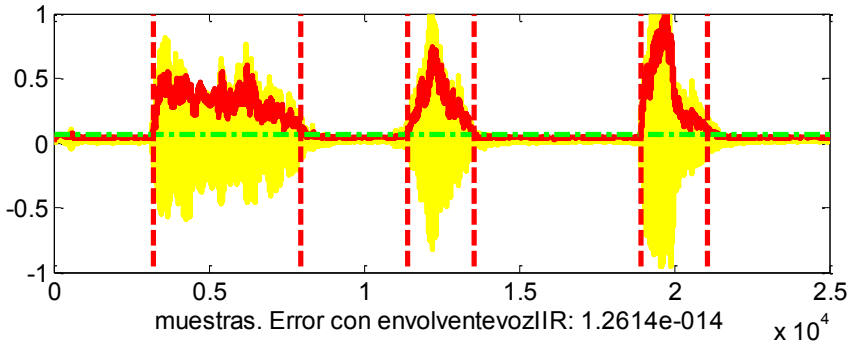


Programación del sistema en cascada

Para estudiar las ventajas de la descomposición del filtro en factores de orden 2, descomponer el filtro anterior en factores de orden 2 en cascada (Para ello puede servirse de las funciones *roots* y *poly* de Matlab). Una vez obtenidos los coeficientes de cada factor de la cascada visualice su magnitud y los polos y ceros de cada factor

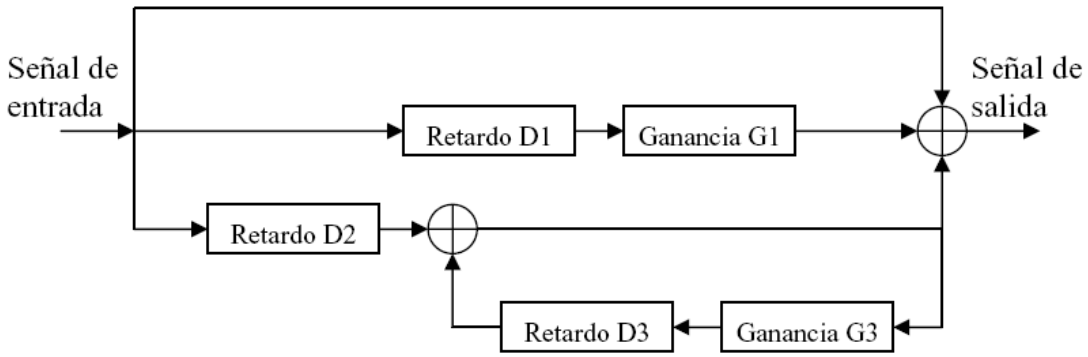


Obtenga la envolvente de la voz utilizando el filtro $H_{IIR}(z)$ implementado mediante la anterior estructura en cascada de sistemas de orden 2. Dibuje el resultado y compárelo con el anterior.



Aplicación a audio

La reverberación es la combinación de un sonido directo, un retorno rápido, y una cola de reverberación. En el ejemplo de dar palmas en una sala, el sonido directo es el aplauso inicial. El retorno rápido sería el eco primario que se oiría. La cola de reverberación sería todos los otros ecos, excepto que serían suficientemente cercanos y no serían muy distintos. Un sistema simple de reverberación es el mostrado en la siguiente Figura.



Sistema que implementa reverberación.

Con la selección apropiada de los parámetros en el sistema propuesto en la anterior, se puede simular la respuesta de cualquier sala desde un velatorio (suele ser una sala con mucho aislamiento sobre las paredes) a lugares sumamente vivaces (como un estadio).

Por supuesto, la palabra clave aquí es "apropiado". De los datos disponibles sobre diversos espacios acústicos, dos de los parámetros son el tiempo inicial de demora (TID) y tiempo de reverberación. El TID es el tiempo transcurrido entre el instante en el que se genera un sonido y cuando se oye el primer eco. El tiempo de reverberación es el tiempo en el cual un sonido atenúa su amplitud a un valor 1/1000 de su amplitud inicial.

Relacionar el TID y el tiempo de reverberación con los valores que se necesitan poner en el sistema es una cuestión de diseño. Las reglas básicas para elegir los diversos valores son como se indica a continuación:

D1: el TID

G1: el mismo que G3

D2: dos veces el TID

D3: 50 mseg.

G3: $10^{-3*D3/\text{tiempo de reverberación}}$

Valores típicos del TID y tiempo de reverberación son:

Lugar	TID (mseg)	Tiempo de reverberación (s)
Estudio	20	0,4
Auditorio	40	0,75
Sala concierto	15	1,80
Iglesia	50	3,25

Implemente el filtro que modela la reverberación del lugar que estime oportuno siguiendo el modelo de la anterior figura y los valores de la anterior tabla. Filtre la voz por dicho filtro y escuche el resultado. Dibuje la voz con reverberación.

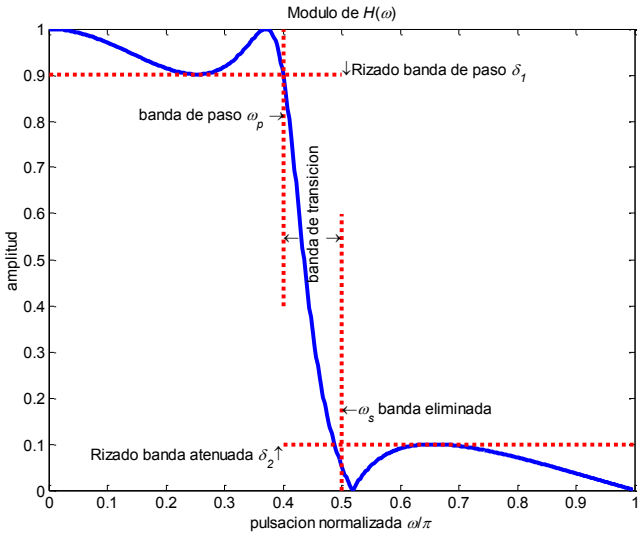
Capítulo IV. Diseño de Filtros Discretos

En este capítulo estudiaremos las dos formas que existen de diseñar filtros discretos: 1) a partir de las especificaciones que definen qué tiene que hacer el filtro, y 2) a partir de la salida deseada para una entrada dada.

Por tanto, este capítulo se propone habilitar al alumno para el diseño de filtros discretos IIR, FIR utilizando los métodos clásicos y adaptativos para realizar una función deseada. El capítulo comienza con el diseño de filtros a partir de sus especificaciones, las cuales son:

1. Tipo de respuesta al impulso (FIR o IIR). Se supone que debe ser causal y estable
2. Si el filtro es FIR, si debe ser de fase lineal
3. Las banda de frecuencia a dejar pasar (banda de paso), rechazar (banda eliminada) y sus tolerancias.

Por tolerancias entendemos tanto el ancho de las bandas de transición de banda de paso a banda eliminada como el rizado de las bandas de paso y eliminada. Un ejemplo de dichas especificaciones puede observarse en el siguiente gráfico:



4.1 Diseño de filtros discretos IIR a partir de sus especificaciones.

Las herramientas de diseño de filtros IIR analógicos han sido muy desarrolladas y son bien conocidas, habiéndose llegado a fórmulas cerradas de gran simplicidad. Puesto que dichas fórmulas no se pueden pasar al dominio discreto, el procedimiento de diseño de filtros discretos IIR utilizará dicho conocimiento obteniendo los filtros discretos como transformaciones de los filtros continuos.

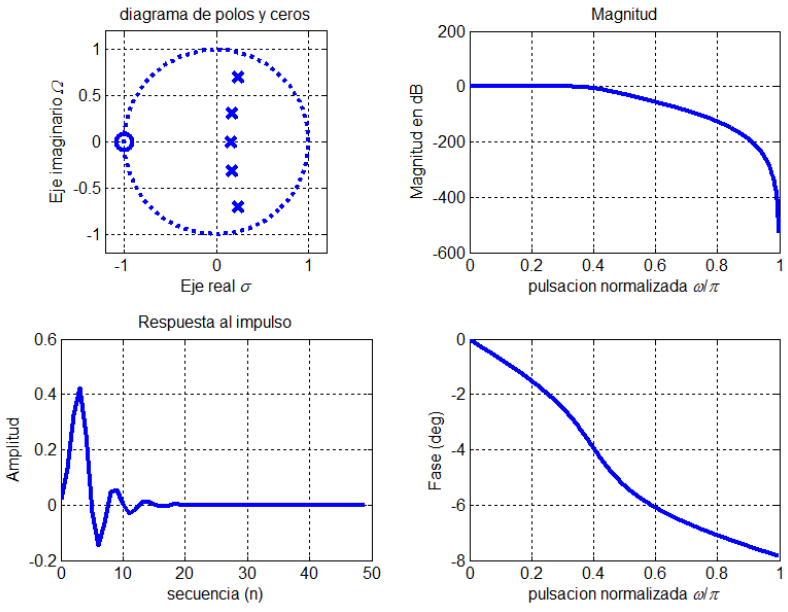
Se describen los filtros IIR discretos más utilizadas.

Filtros de Butterworth

Se define por la propiedad de que la magnitud es monótonamente decreciente y tan sólo depende del orden *N* del filtro y la frecuencia de corte.

Un ejemplo lo tenemos a continuación:

```
N=5;
wc=0.4;
[b,a]=butter(N,wc);
[H,w]=freqz(b,a);
Mag=H.*conj(H);
fase=unwrap(angle(H));
c=roots(b); %ceros
d=roots(a); %polos
h=filter(b,a,[1; zeros(149,1)]);
```



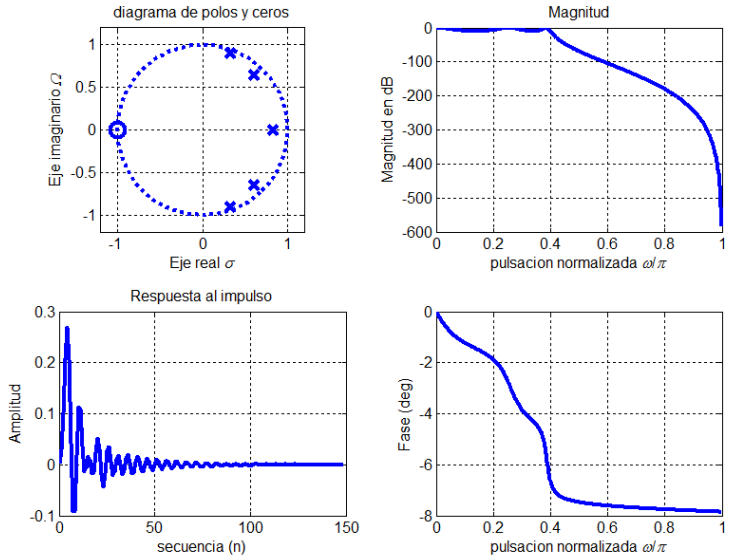
Según aumenta el parámetro *N*, aumenta la pendiente de la magnitud de la banda atenuada.

Filtros de Chebyshev

En este tipo de filtros la magnitud tiene rizado en la banda de paso y es decreciente en la banda atenuada (tipo I), o decreciente en la banda de paso y rizado en la banda atenuada (tipo II). Así depende del orden del filtro, de la frecuencia de corte y del rizado de la banda de paso (tipo I) o la banda atenuada (tipo II).

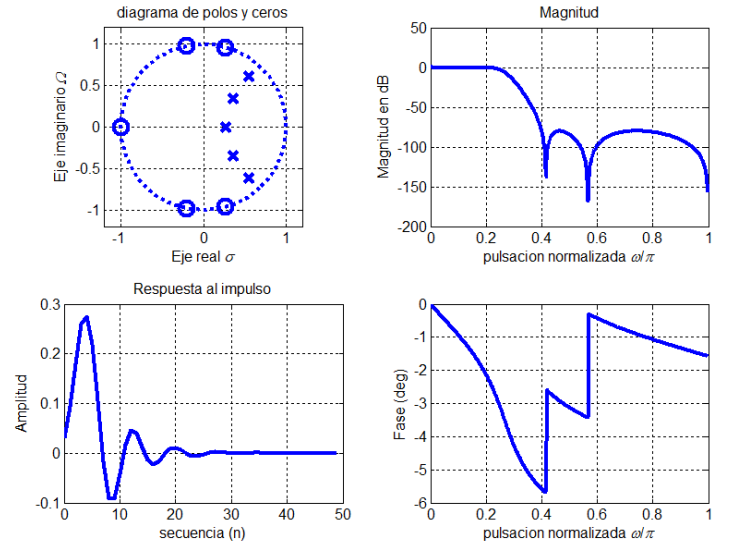
Un ejemplo es el siguiente:

```
N=5;
wc=0.4;
dlta1=5;
[b,a]=cheby1(N,dlta1,wc);
```



Y un ejemplo de tipo II:

```
N=5;
wc=0.4;
dlta2=40;
[b,a]=cheby2(N,dlta2,wc);
```

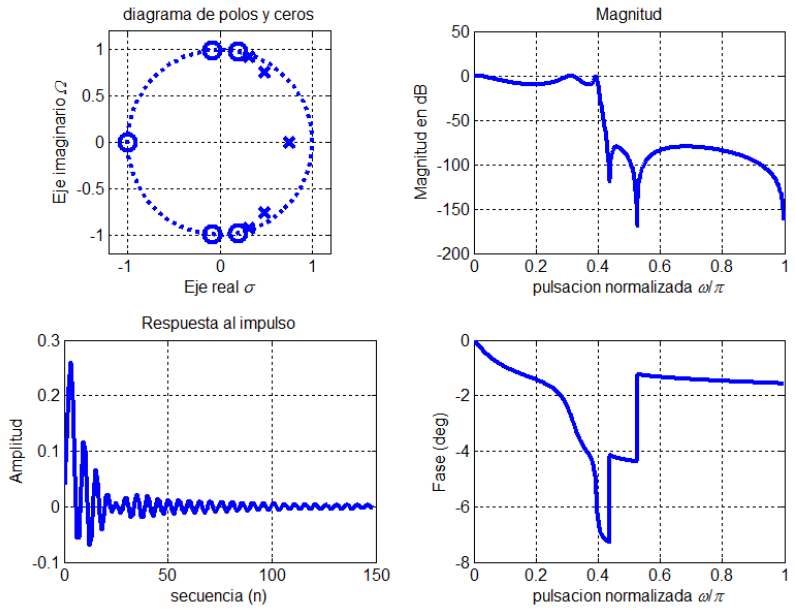


Al aumentar los grados de libertad en relación a los filtros de Butterworth, con un filtro de menor orden se consigue una mejor respuesta.

Filtros Elípticos

Son parecidos a los de Chebyshev, pero tienen rizado en la banda de paso y la banda atenuada. Un ejemplo de un filtro elíptico continuo es:

```
N=5;
wc=0.4;
dlta1=5;
dlta2=40;
[b,a]=ellip(N,dlta1,dlta2,wc);
```



Para unas especificaciones dadas, Chebyshev necesita un orden mayor que elípticos, pero menor que Butterworth.

4.2 Diseño de filtros FIR: método del enventanado

El método usual de diseño del filtros FIR difiere del método utilizado para filtros IIR.

En este caso, los pasos a realizar son:

1. Se especifica en discreto el filtro ideal a aproximar con sus márgenes ω_s, ω_p, δ_1, δ_2. Por ejemplo:

$$H_d(\omega) = \begin{cases} e^{-j\omega\frac{M}{2}} & |\omega| < \omega_c \\ 0 & resto \end{cases} \text{ de fase lineal}$$

2. Se calcula la respuesta del filtro ideal. En el caso del ejemplo tenemos:

$$h_d[n]=\frac{\sin\left(\omega_c\left(n-\frac{M}{2}\right)\right)}{\pi\left(n-\frac{M}{2}\right)}$$ simétrica de eje de simetría en $M/2$

3. Se aproxima la respuesta al impulso ideal, siendo el método más habitual de aproximar el enventanamiento, esto es:

$$h[n]=\begin{cases} h_d[n] & 0\le n\le M \\ 0 & resto \end{cases}=h_d[n]\cdot w[n] \text{ siendo } w[n]=\begin{cases} 1 & 0\le n\le M \\ 0 & resto \end{cases}$$

para que $h[n]$ mantenga la simetría propia de los filtros de fase lineal, $h_d[n]$ y $w[n]$ deben tener el mismo eje de simetría.

El problema a resolver es relacionar las especificaciones del filtro ω_s, ω_p, δ_1, δ_2. con la ventana (altura de los lóbulos laterales Δ y la anchura del lóbulo principal $\Delta\omega$) y la frecuencia de corte ω_c del filtro ideal. Para ello se parte de los siguientes hechos:

1. $\Delta\omega < \omega_s-\omega_p$
2. $\omega_c=\frac{\omega_p+\omega_s}{2}$
3. $\Delta=20\log_{10}\min(\delta_1,\delta_2)$

La ventana la podemos calcular mediante tabla. A partir de δ_1,y δ_2 obtenemos Δ, la cual determina en la tabla la ventana que se debe usar, y con la $\Delta\omega$ dada calculamos la longitud M+1 de la ventana. Un ejemplo de tabla se presenta a continuación:

Ventana	Matlab	$\Delta\omega$	Δ
rectangular	`W=rectwin(M+1)`	$4\pi/M$	-13dB
Barlett	`w=triang(M+1)`	$8\pi/M$	-21dB
Hanning	`w=hanning(M+1)`	$8\pi/M$	-31dB
Hamming	`w=hamming(M+1)`	$8\pi/M$	-41dB
Blackman	`w=blackman(M+1)`	$12\pi/M$	-57dB

Fíjese como al disminuir Δ aumenta $\Delta\omega$. Téngase también en cuenta que la amplitud de los lóbulos laterales no dependen de M. En cambio, aumentando M se consigue estrechar el lóbulo principal.

Ejemplo de uso de dichas tablas:

Dadas las especificaciones: paso bajo con $\omega_s=0.3\pi$, $\omega_p=0.2\pi$, $\delta_1=\delta_2=0.05$, se obtiene $\Delta\omega < 0.1\pi$, $\omega_c=0.25\pi$, y $\Delta=20 \log_{10} 0.05= -26.0206$

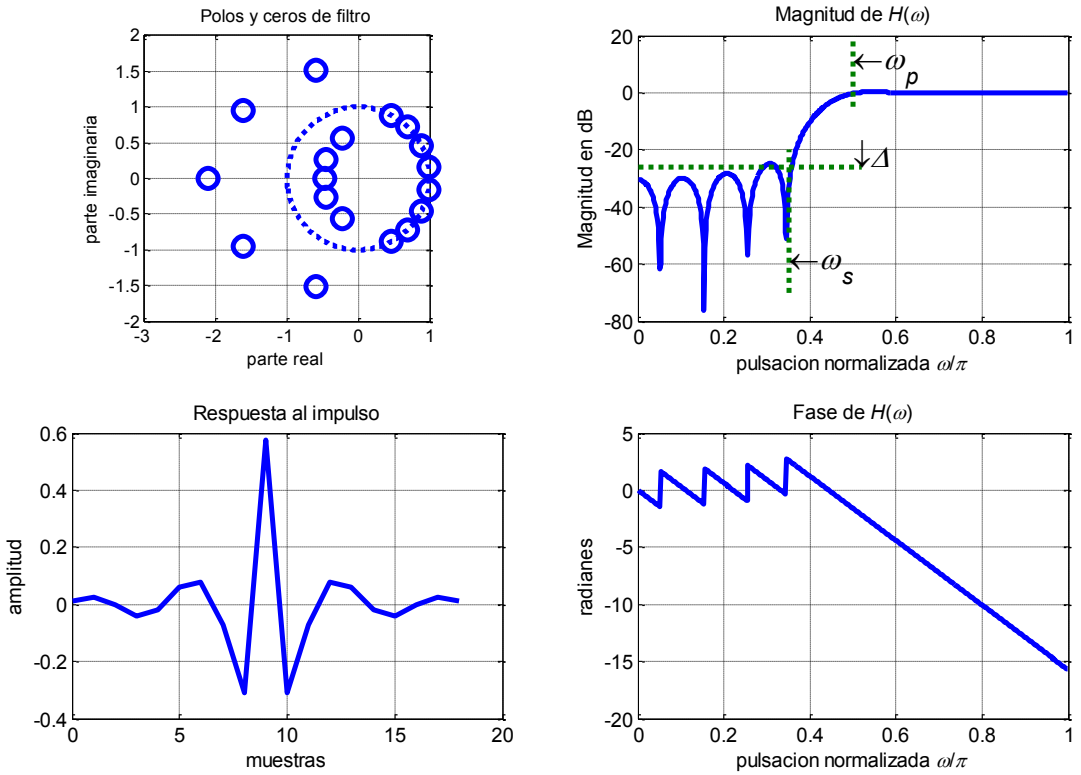
Por tanto, la ventana a utilizar debe ser la que tenga un Δ menor a –26.0206, esto es, una ventana de Hanning. Y el valor de M debe cumplir $8\pi/M<\omega_s-\omega_p=0.1\pi$ de donde $M>80$. Una vez obtenida la M hay que tener en cuenta qué tipo de filtro de fase lineal es para evitar, por ejemplo, filtros paso alto de tipo II.

En el caso anterior, puede observarse cómo el filtro siempre sobredimensiona la Δ. Para evitar este sobredimensionamiento se puede utilizar una ventana ajustable como la de Kaiser, con la cual se pueden controlar los valores de Δ y $\Delta\omega$.

El diseño puede realizarse en Matlab con la función *kaiserord*. La ventana de Kaiser puede obtenerse con la función *kaiser* de Matlab.

Una vez obtenida la ventana y la respuesta al impulso ideal, se multiplican ambas haciendo coincidir su centro de simetría para obtener la $h[n]$ buscada. En Matlab, esta operación puede realizarse con la función *fir*1.

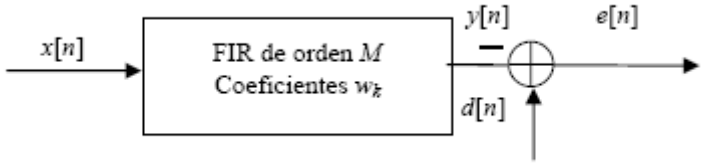
```
ws=0.35;
wp=0.5;
d=0.05;
[M,wn,beta,tipo]= kaiserord([0.35 0.5],[0 1],[d d],2)
h=fir1(M,wn,tipo,kaiser(M+1,beta),'noscale');
freqz(h,1)
% otra opcion sin controlar los valores de M y beta
%c= kaiserord([0.35 0.5],[0 1],[d d],2,'cell');
%h=fir1(c{:});
```



Obsérvese cómo la fase es lineal.

4.3 Diseño de Filtros FIR basado en el método de mínimos cuadrados

En este apartado damos por terminada la exposición de los métodos de diseño de filtros a partir de las especificaciones en frecuencia del filtro y pasamos a diseñar filtros a partir de la salida deseada para una entrada dada.



Para obtener los M+1 coeficientes w_k del filtro FIR que aproximen la salida $d[n]$ ante la entrada $x[n]$ calculamos la salida $y[n]$ real del filtro, dada por

$$y[n]=\sum_{k=0}^{M} w_k \cdot x[n-k] \qquad \text{[Ec.54]}$$

y calculamos los coeficientes que minimicen una función de coste del error cometido entre la salida deseada $d[n]$ y la salida real $y[n]$.

$$e[n]=d[n]-y[n] \qquad \text{[Ec.55]}$$

La función de coste de error de mínimos cuadrados es: $E = \sum_{n=-\infty}^{\infty} e^2[n]$

Se minimiza igualando a cero su primera derivada respecto a cada uno de los coeficientes del filtro

$$\frac{\partial E}{\partial w_k} = 0\,; \qquad k=0,1,2,...,M$$

Dicha derivada se calcula como:

$$\frac{\partial E}{\partial w_k} = \frac{\partial \sum_{n=-\infty}^{\infty} e^2[n]}{\partial w_k} = \sum_{n=-\infty}^{\infty} \frac{\partial e^2[n]}{\partial w_k} = \sum_{n=-\infty}^{\infty} 2 \cdot e[n] \frac{\partial e[n]}{\partial w_k}$$

como: $e[n] = d[n] - y[n] = d[n] - \sum_{k=0}^{M} w_k \cdot x[n-k]$

se tiene

$$\sum_{n=-\infty}^{\infty} 2 \cdot e[n] \frac{\partial e[n]}{\partial w_k} = -\sum_{n=-\infty}^{\infty} 2 \cdot e[n] \cdot x[n-k] = 0\,; \qquad k=0,1,2,.....,M \qquad \text{[Ec.56]}$$

La expresión $\sum_{n=-\infty}^{\infty} e[n] \cdot x[n-k] = 0$ se denomina principio de ortogonalidad. Significa que el error cometido $e[n]$ y la señal de entrada $x[n]$ no tienen nada en común. En otras palabras, la salida $y[n]$ se obtiene modificando la ponderación y fase de las sinusoides que componen la señal de entrada $x[n]$. Componentes de $d[n]$ que no estén presentes en $x[n]$ no podrán encontrarse en $y[n]$, y ese será el error cometido.

Seguimos, sustituyendo $e[n]$ por su expresión en el principio de ortogonalidad

$$\sum_{n=-\infty}^{\infty} \cdot \left(d[n] - \sum_{l=0}^{M} w_l \cdot x[n-l] \right) \cdot x[n-k] = 0\,; \qquad k=0,1,2,.....,M$$

de donde

$$\sum_{l=0}^{M} w_l \sum_{n=-\infty}^{\infty} x[n-l]x[n-k] - \sum_{n=-\infty}^{\infty} d[n]x[n-k] = 0\,; \qquad k=0,1,2,.....,M$$

Puesto que por definición:

$R_{xx}[l-k] = \sum_{n=-\infty}^{\infty} x[n-l]x[n-k]$ es la autocorrelación de $x[n]$ y

$R_{dx}[k] = \sum_{n=-\infty}^{\infty} d[n]x[n-k]$ es la autocorrelación cruzada de $d[n]$ y $x[n]$

se obtiene la solución, denominada solución de Wiener:

$$\sum_{l=0}^{M} w_l \cdot R_{xx}[l-k] = R_{dx}[k]; \quad k=0,1,2,.....,M$$

si se expande dicha expresión se obtienen las M+1 ecuaciones con M+1 incógnitas a resolver

```
para k=0: w0Rxx[0] +w1Rxx[1]   +w2Rxx[2]   +...+wMRxx[M]   = Rdx[0];
para k=1: w0Rxx[-1]+w1Rxx[0]   +w2Rxx[1]   +...+wMRxx[M-1]= Rdx[1];
     ...  ......................................
para k=M: w0Rxx[-M]+w1Rxx[1-M]+w2Rxx[2-M]+...+wMRxx[0]   = Rdx[M];
```

si se utiliza la propiedad: $R_{xx}[m] = R_{xx}[-m]$

y se representan las anteriores ecuaciones de forma matricial, se obtiene:

$$\begin{pmatrix} R_{xx}[0] & R_{xx}[1] & ... & R_{xx}[M] \\ R_{xx}[1] & R_{xx}[0] & ... & R_{xx}[M-1] \\ ... & ... & ... & ... \\ R_{xx}[M] & R_{xx}[M-1] & .... & R_{xx}[0] \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ ... \\ w_M \end{pmatrix} = \begin{pmatrix} R_{dx}[0] \\ R_{dx}[1] \\ ... \\ R_{dx}[M] \end{pmatrix}$$

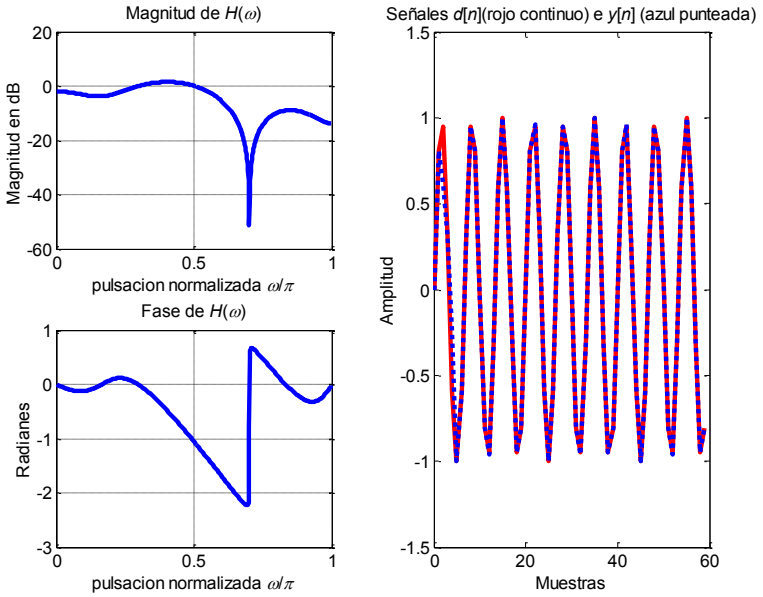
esto es:

$\overline{\overline{R_{xx}}} \cdot \overline{w} = \overline{R_{dx}}$ de donde la solución de wiener es: $\overline{w} = \left(\overline{\overline{R_{xx}}}\right)^{-1} \cdot \overline{R_{dx}}$

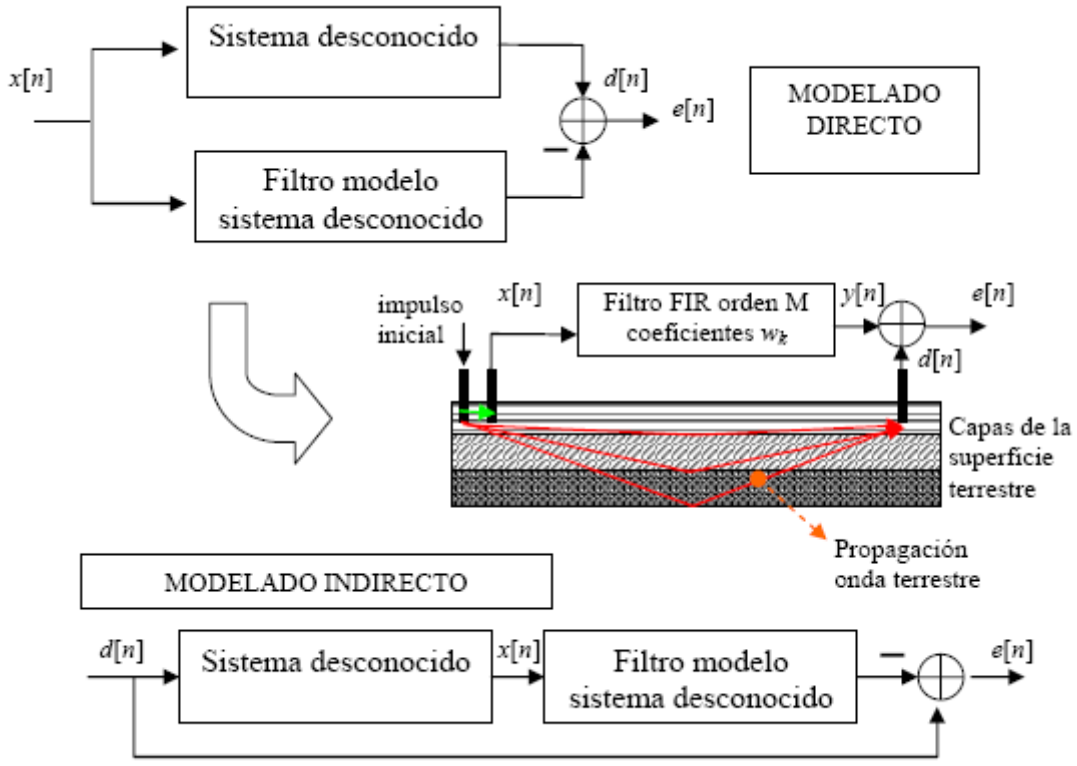
siendo la matriz de autocorrelación $\overline{\overline{R_{xx}}}$ una matriz Toeplitz.

Un ejemplo en Matlab de diseño de filtro que elimina un tono interferente diseñado mediante error cuadrático mínimo es:

```
n=[0:40];
x=sin(0.3*pi*n)+sin(0.7*pi*n); % señal de entrada
d=sin(0.3*pi*n);               % señal deseada
M=6;                           %orden del filtro
Rxx=xcorr(x,x,M);              %se calcula Rxx[m] m=0,1,..,M
Rdx=xcorr(d,x,M);              %se calcula Rydx[m] m=0,1,...,M
MRxx= toeplitz(Rxx(M+1:end));  %Matriz de autocorrelacion
w=inv(MRxx)*Rdx(M+1:end)'
freqz(w,1);            %el filtro debe eliminar el seno de 0.7pi
y=filter(w,1,x);       %se calcula la salida del filtro diseñado
w
    0.5000    0.1756   -0.0000   -0.4197
```



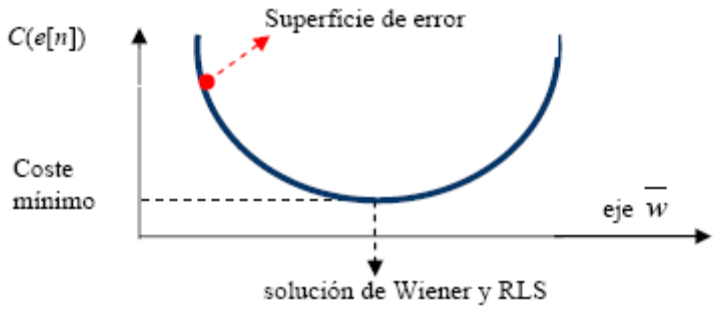
Este método de diseño de filtros es muy útil para modelado de sistemas estáticos donde la entrada y salida son bien conocidas. Este modelado puede ser tanto directo como indirecto. Un ejemplo de sistema directo es el modelado de la superficie terrestre que se realiza en geología al recoger las vibraciones producidas por una explosión en otro punto, tal y como se observa en la figura.



El problema de esta técnica surge cuando el sistema desconocido a modelar varía con el tiempo. En estos casos tengo dos soluciones: 1) cada determinado número de muestras se recalcula el filtro, o 2) se monitoriza el error actualizando los coeficientes del filtro cada muestra *n*. El algoritmo que actualiza el filtro que modela el sistema desconocido se denomina algoritmo iterativo de mínimos cuadrados.

4.4 Algoritmos adaptativos de máxima pendiente

Hasta el momento hemos diseñado los filtros minimizando el error cuadrático cometido entre la salida del filtro y la señal deseada. Esto es, si dibujamos el error cometido en función de los coeficientes del filtro $\bar{w}[n]$ tenemos que la solución de Wiener obtiene el mínimo mediante una fórmula cerrada.



En el caso adaptativo se busca el mínimo de la función de coste de error comenzando en un punto de la superficie de error y deslizándonos hacia el mínimo iterativamente.

Ésta idea se materializa mediante el siguiente procedimiento:

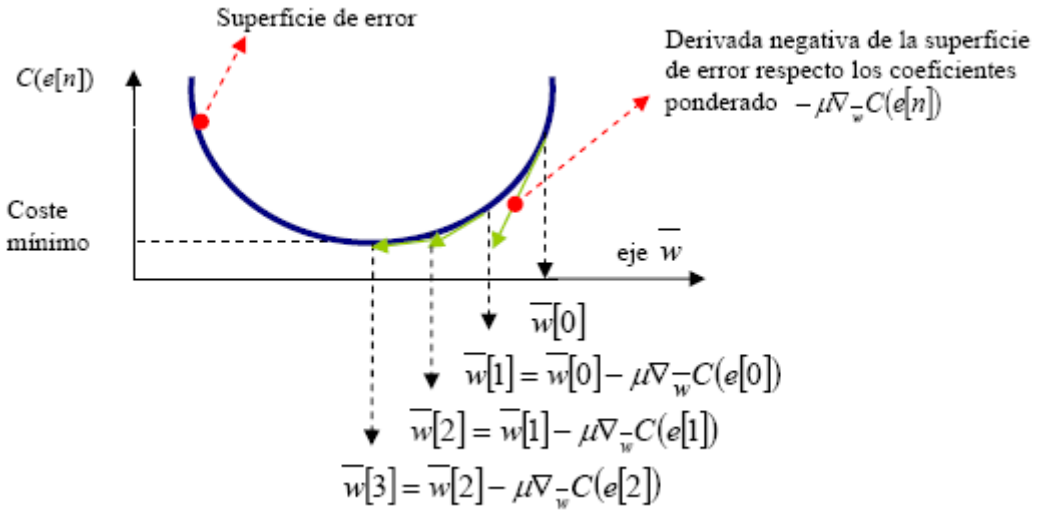
1. Se inicia el algoritmo en una posición arbitraria de la superficie de error $\bar{w}[0]$ y n=0
2. Se calcula el gradiente de la superficie de error en dicho punto $\nabla_{\bar{w}} C(e[n])$, siendo:

$$\nabla_{\bar{w}} C(e[n]) = \left(\frac{\partial C(e[n])}{\partial w_0}, \quad \frac{\partial C(e[n])}{\partial w_1}, \quad \ldots \quad \frac{\partial C(e[n])}{\partial w_M} \right)^T$$

3. Se modifican los coeficientes con un movimiento proporcional al gradiente y en sentido opuesto $\bar{w}[n+1] = \bar{w}[n] - \mu \nabla_{\bar{w}} C(e[n])$

   donde μ es el paso de adaptación y regula la velocidad de convergencia hacia el coste mínimo

4. se vuelve al punto 2 con n=n+1.



Es sencillo ver en la figura que si el paso de adaptación μ es muy pequeño, la convergencia es muy lenta, y si es muy grande el algoritmo puede no converger al coste de error mínimo.

Se estudian algoritmos de máxima pendiente con diferentes funciones de coste:

Función de coste cuadrática: $C(e[n])=e^2[n]$

En este caso:

$$\nabla_{\bar{w}}C(e[n])=\frac{\partial e^2[n]}{\partial \bar{w}}=2e[n]\frac{\partial e[n]}{\partial \bar{w}}$$

puesto que

$$e[n]=d[n]-y[n]=d[n]-\bar{w}^T\bar{x}[n]$$

se tiene

$$\frac{\partial e[n]}{\partial \bar{w}}=-\bar{x}[n]$$

de donde

$$\nabla_{\bar{w}}C(e[n])=-2e[n]\bar{x}[n]$$

por tanto, el algoritmo adaptativo queda

$$\bar{w}[n+1]=\bar{w}[n]+\mu e[n]\bar{x}[n]$$

Algoritmo denominado LMS. El valor máximo de μ para que el sistema converja es:

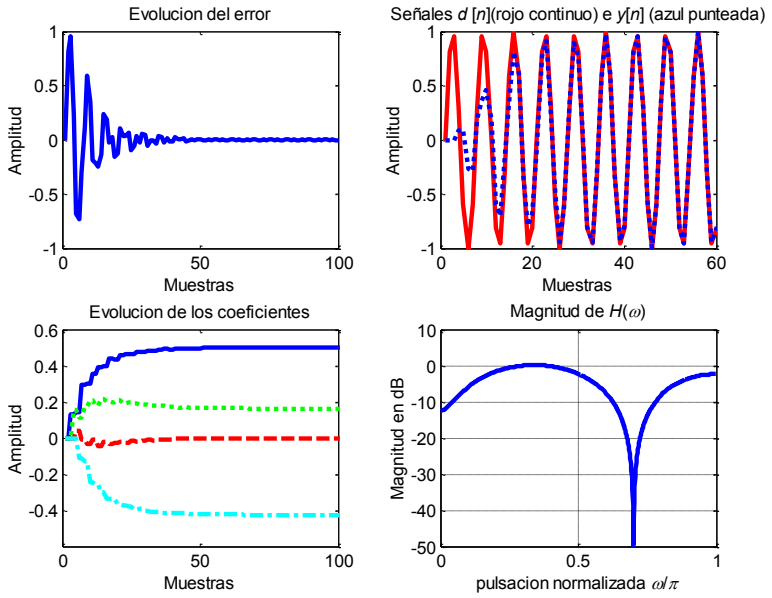
$0<\mu<\dfrac{2}{(M+1)P_x}$ donde P_x es la potencia de la secuencia $x[n]$ estimada como

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1} x^2[n]$$

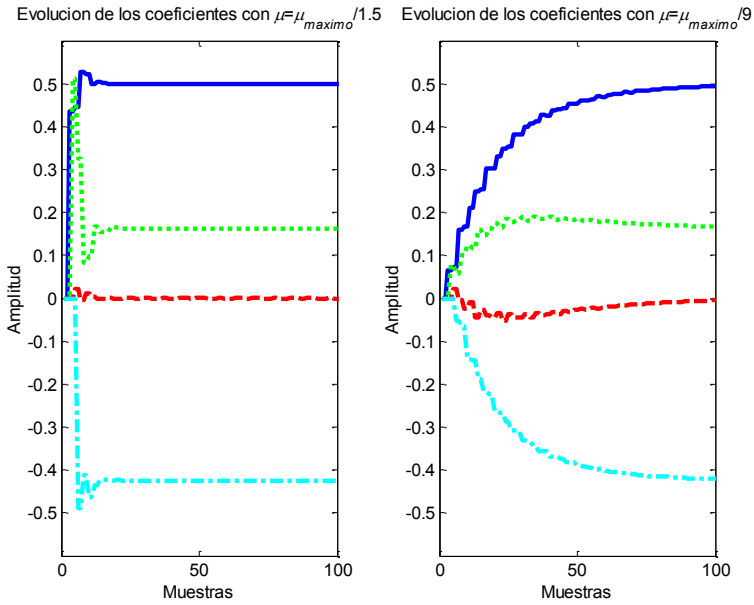
Éste algoritmo reduce en gran medida el número de operaciones del RLS (no necesita calcular el vector $\bar{g}[n]$) y es más robusto a su implementación en aritmética de punto fijo. El inconveniente es que necesita más muestras de entrada para llegar al mínimo de la función de coste de error

Un ejemplo de algoritmo LMS con el caso anterior

```
x=sin(0.3*pi*n)+sin(0.7*pi*n); % señal de entrada
d=sin(0.3*pi*n);               % señal deseada
M=3;                           %orden del filtro
r0=xcorr(x,x,0,'biased');      %se estima la mu maxima
mumax=2/((M+1)*r0);
u=zeros(1,M+1);   %condiciones iniciales del filtro
w=zeros(M+1,1);   %coeficientes del filtro
for n=1:length(x)
   u=[x(n) u(1:length(u)-1)];
   y(n)=u*w                   %salida del filtro
   e(n)=d(n)-y(n);            %error
   w=w+(mumax/5)*u'*e(n);    %actualización de coeficientes
end
w(:,end)
0.4996    0.1629   -0.0003   -0.4250
```



Obsérvese cómo el filtro final es similar al filtro de Wiener. Con el valor de μ utilizado ($\mu = \mu_{máximo}/5$) el filtro tarda unas 60 muestras en converger. Si utilizamos otros valores de μ podemos observar en la siguiente figura cómo al aumentar μ converge más rápido aunque el desajuste final es mayor (rizado en la representación de los coeficientes del filtro) y, cómo al disminuir μ converge más lento siendo el desajuste final menor.



Si se repite lo anterior con la función de coste valor absoluto: $C(e[n])=|e[n]|$

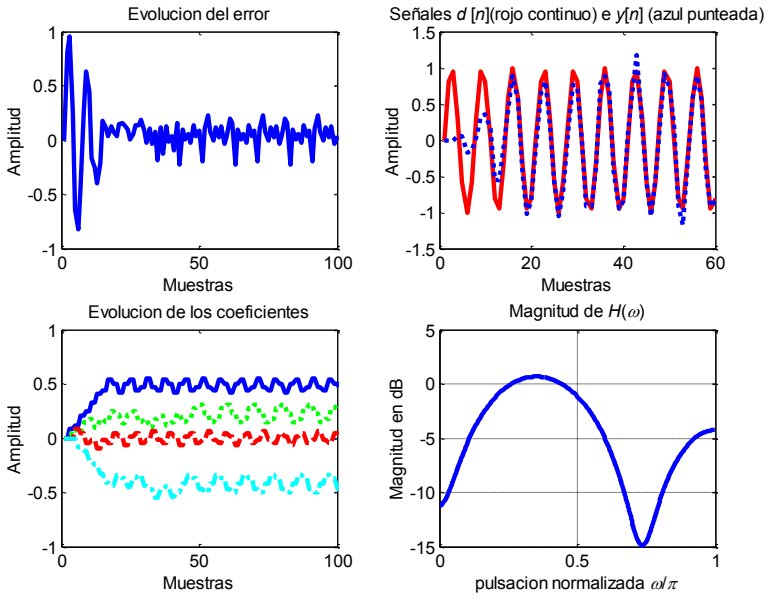
En este caso:

$$\nabla_{\bar{w}} C(e[n]) = \frac{\partial |e[n]|}{\partial \bar{w}} = -signo(e[n])\bar{x}[n]$$

de donde el algoritmo adaptativo queda

$$\bar{w}[n+1] = \bar{w}[n] + \mu\, signo(e[n])\bar{x}[n]$$

Algoritmo denominado LMA, que puede considerarse una versión cuantificada del LMS. Aplicando el LMA al ejemplo anterior obtenemos:



4.5 Problemas de Aula

-Problema 4.1

Un transformador ideal de Hilber discreto en el tiempo se define como:

$$\begin{cases} -90\ grados & 0 < w < \pi \\ +90\ grados & -\pi < w < 0 \\ con\ magnitud\ constabte = 1 \end{cases}$$

También se le conoce como desfasador de fase de 90 grados.

a) Obtener la ecuación $H_d(e^{jw})$ y dibujar su fase

b) ¿Qué clase de filtro FIR de fase lineal se puede utilizar para aproximar el transformador de Hilbert?

c) Para diseñar este filtro por el método de enventanamiento calcular su respuesta al impulso.

d) ¿Cuál es l retardo si M=21?

e) ¿Cuál es el retardo si M=20?

- Problema 4.2

Los modelos de propagación de onda están basados en el muestro espacio-temporal de la onda propagada en un tubo rectilíneo (tubos de instrumentos sin curvas). Así, es bien conocido que la oscilación de la presión en un conducto cilíndrico sin pérdidas es descrita por la ecuación:

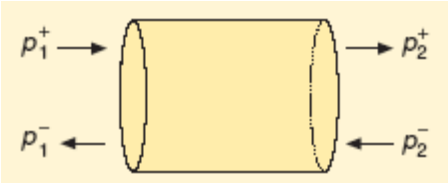
$$\frac{\partial^2 p}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}$$

donde p es la presión en función de la posición x en el tiempo t. La solución a esta ecuación puede escribirse como una suma de dos ondas de presión viajando en direcciones opuestas a velocidad c. Así en un punto x=xi de un tubo cilíndrico de

sección $S = \pi\ a^2$, estas dos ondas p_i^+ y p_i^- se relacionan con la presión local por medio de la ecuación $p_i = p_i^+ + p_i^-$.

Si dicho cilindro es de longitud *L*, la relación entre la transformada de Fourier de la presión de onda en cada extremo del cilindro puede ser expresada por la siguiente función de transferencia

$$H(\omega) = \frac{P_2^+}{P_1^+} = \frac{P_1^-}{P_2^-} = e^{-jkL}$$



donde $k = \omega / c$. Así, en el caso de no considerar las pérdidas en el tubo, $H(\omega)$ es simplemente la representación en el dominio de la frecuencia de un retardo de L/c segundos.

Esto puede ser simulado en tiempo discreto como sigue: dado un periodo de muestreo T=1/fs la propagación de la onda de un extremo al otro del tubo puede ser modelada como un retardo entero de N muestras (z^{-N}) en serie con un filtro de retardo fracional $H_{FD}(z)$ que implementa un retardo no entero de D muestras; el retardo total (N+D)T debe ser igual a L/c. El filtro de retardo fraccional puede ser diseñado por el método del enventanado, aunque los métodos mas comunes son el de filtros paso todo de bajo orden de Thiran o interpoladores FIR de Lagrange.

Un modelado preciso del cilindro debe considerar los efectos de las pérdidas en el tubo, lo cual requiere introducir en $H(\omega)$ los efectos de las pérdidas. Estos efectos puedes ser tenidos en cuenta reescribiendo la función de transferencia del tubo como

$$H_L(\omega) = e^{-\Gamma L}$$

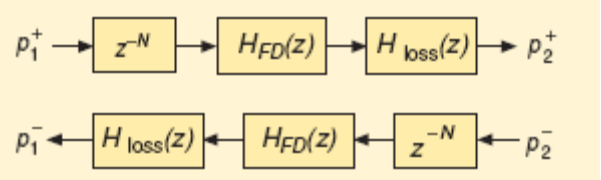
donde Γ es una constante compleja (parte real e imaginaria) que representa las pérdidas de propagación. La función de transferencia $H_L(\omega)$ contiene el retardo de la función de transferencia sin pérdidas. Por tanto, se puede descomponer $H_L(\omega)$ como dos filtros en

serie, el primero $H(\omega)$ que representa el retardo sin pérdidas y el segundo $H_{loss}(\omega)$ que representa sólo las pérdidas definido como

$$H_{loss}(\omega) = \frac{H_L(\omega)}{H(\omega)} = \frac{e^{-j\Gamma L}}{e^{-j\omega L/c}}$$

el cual puede ser aproximado por un filtro digital y completar el modelo de propagación.

Concluyendo, el modelo de propagación del tubo de un instrumento de aire viene definido por tres filtros: el primero implementa un retardo entero, el segundo un retardo no entero, y el tercero las pérdidas de las paredes del tubo. En este examen hay que realizar un programa Matlab que implemente dicho modelo.



Suponiendo que conoce las variables T, L, Γ, y c

a) Realice un programa que calcule el retardo entero N del modelo de propagación de la onda por el tubo y lo implemente como un filtro.

b) Calcule analíticamente la expresión de la respuesta al impulso de un filtro FIR de fase lineal que realice un retardo fraccional D, siendo $0 < D < 1$. Hágalo con el método del enventanado.

c) A partir de la expresión obtenida en 2, realice una función Matlab

h=FiltroRetardoRacional(T,L,c,Lhn)

cuyas entradas sean las variables conocidas y Lhn la longitud de la respuesta al impulso (suponga Lhn impar). La salida debe ser la respuesta al impulso del filtro que implemente un retardo no entero de D muestras.

d) Realice un programa que, a partir de las variables conocidas y la respuesta al impulso h[n] calculada en el apartado anterior, implemente el retardo fraccional D del modelo de propagación de la onda por el tubo.

e) Realice un programa en Matlab que implemente $H_{loss}(\omega)$.

- Problema 4.3

Este problema, trata sobre canceladores de eco en circuitos de telefonía de larga distancia. Dicho problema aparece en los puntos de conversión de 2 a 4 hilos, denominados "híbridas". La figura 1 muestra una híbrida.

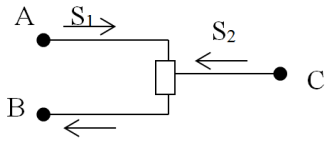


Figura 1: Circuito híbrido.

El funcionamiento normal de la misma consiste en que la señal S_1 procedente del punto A, pasa al punto C pero no pasa al punto B, mientras que la señal S_2 , procedente del punto C, pasa al punto B. En la práctica es difícil lograr que no pase nada de la señal S_1 al punto B por lo que un modelo de la señal que aparece en el punto B sería el mostrado en la figura 2, donde la rama multiplicada por *a* indica la fuga de señal hacia el punto B procedente del A. Este fenómeno es el que origina los ecos.

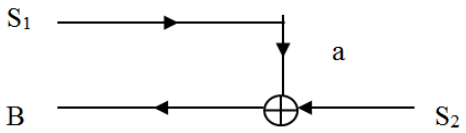


Figura 2: Modelo de circuito híbrido.

Para solucionar el problema, una posibilidad es usar canceladores de ecos. La figura 3 muestra como se inserta el cancelador de ecos en el circuito de la figura 1. El cancelador de eco es un sistema adaptativo que pretende minimizar la potencia en el punto B.

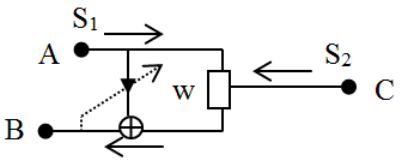


Figura 3: Cancelador de eco en circuito híbrido.

Un modelo completo con la híbrida y el cancelador de eco puede verse en la figura 4

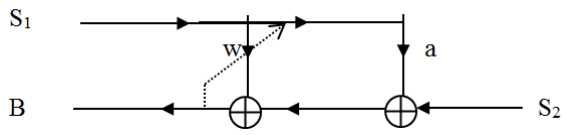


Figura 4: Modelo de cancelador de eco en circuito híbrido

Suponga que S_1 y S_2 están incorreladas. En relación a la figura 4, se pide:

a) Determine la expresión de la potencia en B en función de w, la potencia P_{S1} de S1 y la potencia P_{S2} de S2.

b) Para el caso de utilizar un algoritmo adaptativo de gradiente, indique cuánto vale μ_{MAX}, de acuerdo al criterio que conoce.

c) Determine el valor óptimo del peso w (el que minimiza la potencia en B).

d) Dibuje la superficie de error para los casos $P_{S2}=0$ y $P_{S2}\neq 0$ ($P_{S1}\neq 0$) ¿Varía el valor óptimo del peso? ¿En qué se diferencian las superficies de error de ambos casos?

e) Indique las diferencias prácticas en el comportamiento de un algoritmo LMS en los casos $P_{S2}=0$, y $P_{S2}\neq 0$. ($P_{S1}\neq 0$) en función de la velocidad de convergencia, y desajuste final de la curva de aprendizaje.

f) Suponga que la potencia de la señal S1 disminuye a la décima parte, es decir, $P'_{S1}=P_{S1}/10$. Dibuje las superficies de error en ambos casos. ¿Varía el valor óptimo del peso? ¿En qué se diferencian?

g) Caso de que la adaptación se llevara a cabo con un algoritmo LMS, suponiendo que μ deba escogerse el mismo para ambas situaciones (apartado f) indique cómo elegiría el valor de μ_{MAX}. Suponiendo el mismo valor de μ para ambos casos, ¿varía la velocidad de convergencia de un caso al otro?

- Problema 4.4

En este problema se estudia un sistema de cancelación de ruido. El sistema (Figura 1) consta de dos micrófonos que captan la presión sonora en dos puntos de una sala ($x_1(t)$ y $x_2(t)$). En dicha sala existen dos señales, la señal $s(t)$, que es la señal de interés, y la señal $r(t)$, que es una señal indeseada. El primer micrófono ($x_1(t)$) capta tanto la señal de

interés *s(t)* como una versión filtrada de *r(t)*, que denominaremos $r_1(t)$, mientras que el segundo micrófono únicamente capta *r(t)*.

Las señales captadas por los micrófonos se muestrean a través de dos conversores C/D que trabajan a la misma frecuencia de muestreo, sin aliasing obteniendo:

$$x_1[n] = x_1(nT_s) = s(nT_s) + r_1(nT_s) = s[n] + r_1[n]$$

$$x_2[n] = x_2(nT_s) = r(nT_s) = r[n]$$

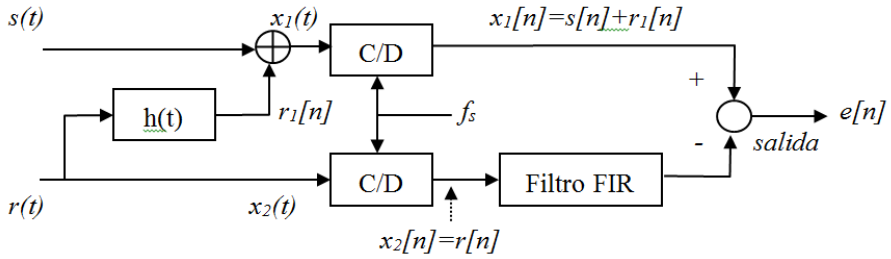


Figura 1: Cancelación de ruido.

Suponga además que las señales *s(t)* y *r(t)* están incorreladas entre sí y que la autocorrelación de *r(t)* es:

$$R_r(\tau) = \sigma^2 \operatorname{sinc} \frac{\tau}{T_s}$$

Suponga que el filtro FIR tiene 5 coeficientes y que el sistema que modela la sala tiene una respuesta impulsiva:

$$h(t) = 0.4\delta(t - 3T_s)$$

Se pide:

a) Determine los pesos del filtro FIR que minimizarían la potencia de la señal e[n].

b) Escriba la expresión de la señal de salida e[n] cuando los pesos del filtro FIR son los calculados en el apartado anterior. Indique si la minimización de la potencia de dicha señal implica la cancelación total de las componentes indeseadas en la salida.

- Problema 4.5

El eco se puede modelar como el resultado de un filtrado. Si llamamos *s[n]* a la señal original (sin eco), la señal con eco *x[n]* se puede modelar como:

$$x[n] = s[n] * h_{eco}[n]$$

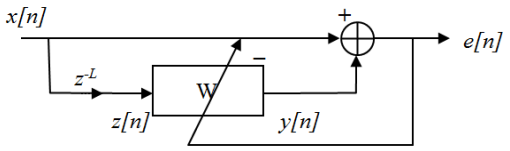
donde

$$h_{eco}[n] = \delta[n] + a\delta[n-M]$$

siendo *0<a<1*. Consideremos el caso de *M = 5* para el resto del problema. Suponga que *s[n]* es un proceso aleatorio estacionario cuya autocorrelación viene dada por:

$$R_s[m] = \begin{cases} 1 & m = 0 \\ 1/2 & |m| = 1 \\ 0 & resto \end{cases}$$

Este problema trata sobre canceladores de eco adaptativos, es decir, tratar mediante filtros adaptativos de eliminar (o al menos reducir) el eco presente en la señal *x[n]*. Para ello se propone el siguiente esquema de filtrado adaptativo:



a) Determine la autocorrelación de la señal con eco *x[n]*. Calcule la potencia de *x[n]*.

b) Suponiendo que el filtro adaptativo sea un filtro FIR de 3 coeficientes, y que el retardo de la figura vale L = 4, determine las ecuaciones normales para calcular los coeficientes del filtro minimizando la potencia de *e[n]* en función de *x[n]*.

c) Determine el valor en función de a de los coeficientes del filtro hacia los que convergería el filtro mediante un algoritmo de gradiente.

d) Suponiendo que los coeficientes del filtro, una vez alcanzada la adaptación permancen en su valor óptimo, determine la relación entre *s[n]* y *e[n]*. ¿Se ha eliminado totalmente el eco?

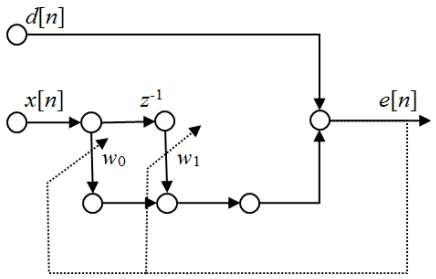
e) Suponga ahora que L = 0. Determine nuevamente R y P. Determine de nuevo los pesos óptimos y la relación entre *s[n]* y *e[n]*. ¿Se ha eliminado el eco?

f) Suponga finalmente que L = 7. Determine nuevamente R y P. Determine de nuevo los pesos óptimos y la relación entre s[n] y e[n]. ¿Se ha eliminado el eco?

g) A la vista de los resultados anteriores, ¿Puede indicar qué margen de valores del retardo L se puede aplicar y qué sucede si dicho retardo se toma demasiado pequeño o demasiado grande?

- Problema 4.6

Este problema trata sobre filtros adaptativos cuando las señales a la entrada son de banda estrecha (tonos). Para ello considere inicialmente el esquema habitual de un filtro adaptativo de dos coeficientes trabajando como cancelador tal y como se muestra en la figura:



Suponga que las señales x[n] y d[n] son las siguientes:

$$x[n] = \cos(\omega_0 n + \theta)$$
$$d[n] = sen(\omega_0 n + \theta)$$

siendo θ una variable aleatoria uniforme en el intervalo $[-\pi, \pi]$. Se pide:

a) Determine la función de autocorrelación de la señal x[n] y la correlación cruzada $R_{dx}[m] = E\{d[n]x[n-m]\} = E\{d[n+m]x[n]\}$.

b) Escriba la ecuación de la superficie de error $E\{e^2[n]\}$ en función de los coeficiente $\overline{w} = [w_0, w_1]$ del filtro. Obtenga dicha relación de forma matricial. Indique los valores de los elementos de las matrices que utilice.

c) Determine, en función de ω_0, los pesos óptimos. Calcule la expresión del error cometido tras la adaptación del filtro ($E\{e^2[n]\}$ mínimo) y calcule su valor numérico en función de ω_0.

d) Para paliar el problema de que los pesos óptimos se hacen muy grandes si el valor de ω_0 es muy pequeño se modifica el esquema de la figura del filtro adaptativo anterior cambiando el retardo de una muestra z-1 por un retardo de M muestras z-M.

Indique la nueva expresión de la superficie de error en forma matricial indicando los nuevos valores de los elementos de las matrices. Calcule de nuevo el vector de pesos óptimos así como la potencia de la señal error tras la adaptación.

NOTA: Relaciones útiles

$$\cos A + \cos B = 2\cos\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right),$$

$$\text{sen}A + \text{sen}B = 2\text{sen}\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right), \text{ y } 1-\cos A = 2sen^2\left(\frac{A}{2}\right).$$

- Problema 4.7

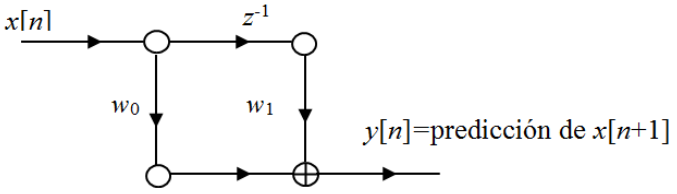
La secuencia de números es un fragmento de una serie de datos naturales

n	0	1	2
x[n]	2,4	1,2	-1,9

Mediante análisis de un fragmento muy largo de dichos datos se ha estimado la autocorrelación de dicha señal, obteniéndose

m	0	1	2	3
R[m]	21,5	0,1	-19,8	-0,4

Resultaría interesante ser capaz de predecir muestras futuras de la señal. Para ello se piensa utilizar un pedictor lineal FIR de dos coeficiente como el de la figura.



de modo que el error cuadrático medio sea mínimo.

a) Determine el valor óptimo de los coeficientes w_0 y w_1.

b) Utilizando los valores del apartado anterior determine cuánto vale la predicción de *x[2]* (valor predicho para el instante 2 en función de los valores del instante *n=1* y *n=0*). Compara el resultado con el valor real de *x[2]*.

c) Suponiendo que la señal *x[n]* sea un proceso aleatorio AR de orden 2, es decir, que la señal se ajusta a dicho modelo, determine la expresión de su densidad espectral de potencia.

d) Determine y dibuje la densidad espectral de potencia de la señal error de predicción.

e) Suponiendo que los coeficientes hallados en el apartado 1 fuesen los correspondientes a la iteración de un algoritmo LMS para predecir la muestra *n*=2 en función de *n=0* y *n=1*, indicar cuáles serían los pesos que se utilizarían para predecir la muestra 3 en función de la 2 y la 1. (suponga $\mu = \frac{\mu_{max}}{10}$ y determine μ_{max}).

- Problema 4.8

La estimación del retardo entre dos señales recibidas es un problema relevante en muchas aplicaciones, por ejemplo, en la resolución del sincronismo en sistemas de comunicaciones, en localización y seguimiento de objetivo en radar y sonar, etc. En este problema se debe desarrollar un algoritmo que estima el retardo entre dos señales:

$$x[n] = s[n] + \phi[n]$$
$$y[n] = s[n - D[n]] + \theta[n]$$

Donde $s[n] = A[n]\cos(\omega_0 n)$ es la señal original de banda estrecha con frecuencia central ω_0, *D[n]* es el retardo normalizado por el periodo de muestreo *T*, y $\phi[n]$ y $\theta[n]$ es ruido aditivo supuesto blanco.

El objetivo es estimar y seguir el valor de *D[n]* de la forma más precisa y rápida posible. Para ello vamos a seguir las ideas del algoritmo ETDE que utiliza un LMS para estimar el retardo entre *x[n]* e *y[n]*.

a) En primer lugar trabajaremos con filtros como retardadores. Supuesta la señal *x[n],* diseñe un filtro que obtenga la señal *y[n]* con un retardo entero *D[n]=3*. Escriba la expresión de su *H(z)* y realice un programa que obtenga *y[n]* a partir de *s[n]*.

b) En este apartado se trata de trabajar con retardos no enteros. Realice un programa en Matlab que diseñe un filtro de *L* muestras que obtenga la señal *y[n]* con un retardo no entero *D[n]=3.2* (tres coma dos). Añada la líneas necesarias al programa para obtener *y[n]* a partir de *s[n]*.

c) El algoritmo ETDE estima el retardo desconocido entre dos señales mediante el algoritmo LMS. Para ello parte de las señales *x[n]* e *y[n]*. Una de las dos se utiliza como entrada al filtro y la otra como señal deseada. El filtro adaptativo minimiza el error cometido compensando el retardo entre las dos señales. El valor de *D[n]* se estima como el retardo de grupo del filtro. Dibuje el esquema del algoritmo ETDE.

d) Realice un programa que, partiendo de las señales *x[n]* e *y[n]* represente el retardo *D[n]* entre las dos señales. Pude suponer que $\omega_0 = \pi / 2$. Utilice las ideas del apartado anterior.

4.6 Ejercicios Prácticos

Señal para filtros diseñados por especificaciones

Mediante la correspondiente utilidad de Matlab, grabe aproximadamente 3 segundos un sonido que conste de voz con un tono de fondo de 1600Hz. Para ello genere el tono con su altavoz y durante la audición grabe el mismo tono mientras habla. Realice la grabación con frecuencia de muestreo 8000 muestra por segundo, con 16 bits por muestra.

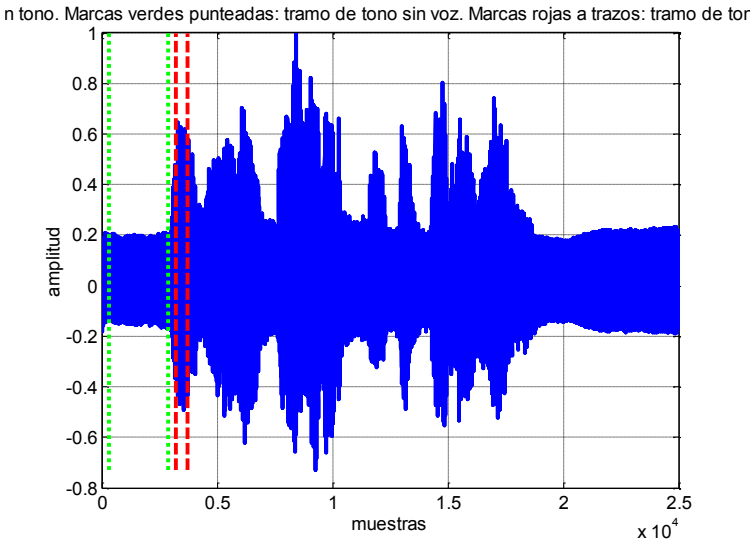
Ejemplo de código para realizar la grabación (en script ejecutable)

*tono=cos(2*pi/5*[0:5*fs]);soundsc(tono);*

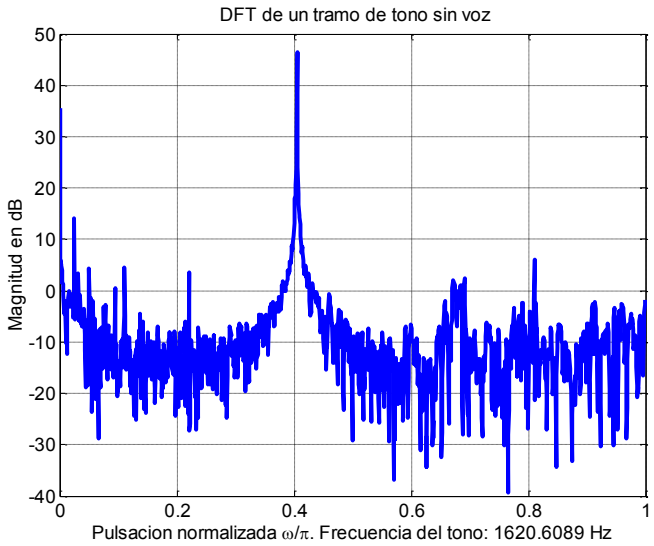
*voztono=wavrecord(3*fs,fs,'double');*

Elimine la media de la voz grabada y normalice su amplitud máxima a 1, asígnele la variable *voztono*. Verifique con el comando soundsc la grabación realizada.

Encuentre el principio y final de una zona de tono sin voz y de otra zona de voz con tono. Dibuje *voztono* con los principios y finales seleccionados.



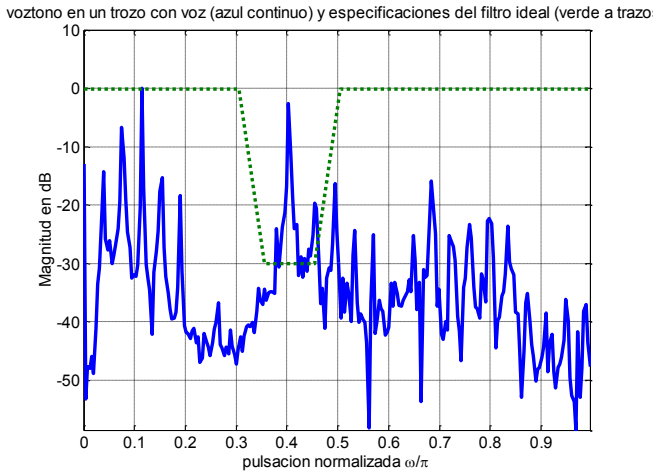
Represente la transformada de Fourier del tramo seleccionado de tono sin voz y estime la frecuencia del tono superpuesto a la señal. Tenga en cuenta que el tono puede tener armónicos que aparezcan en la gráfica



Especificaciones del filtro para los métodos clásicos.

Para eliminar el tono interferente de la voz se utilizará un filtro banda eliminada centrado en la frecuencia del tono interferente.

Del tramo seleccionado de voz con tono interferente, calcule la magnitud de su DFT. Con ayuda de dicha gráfica defina los vectores ω_i=[0 ω_{p1} ω_{s1} ω_{s2} ω_{p2} 1] y H_i=[1-δ_1 1-δ_1 δ_2 δ_2 1-δ_1 1-δ_1] (donde δ_1 es el rizado de la banda de paso y δ_2 el rizado en la banda atenuada) de tal manera que *plot*(ω_i,H_i) dibuje las especificaciones deseadas para el filtro que elimine el tono de la voz. Represente en una sola gráfica el espectro de la voz con tono interferente y las especificaciones del filtro normalizadas a amplitud uno.



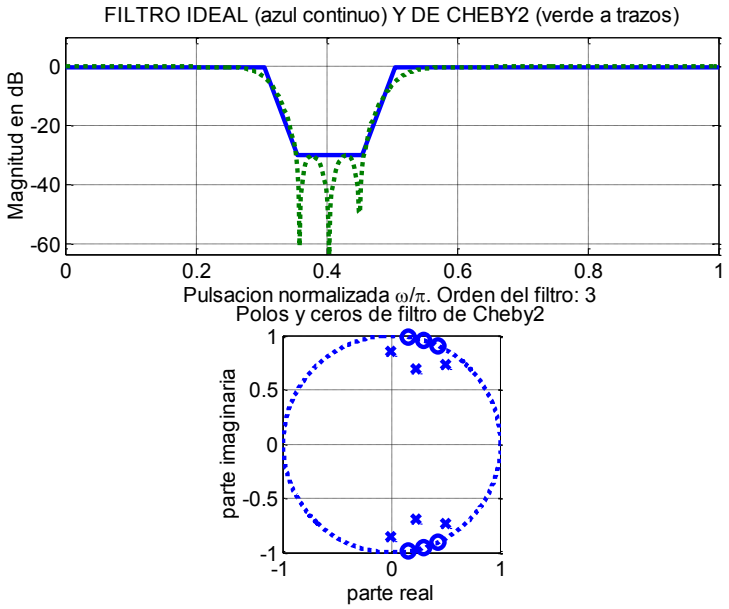
Recuerde que el objetivo de esta práctica es aprender las características que diferencian un filtro de otro; así, no utilice especificaciones demasiado estrictas como lo son rizados muy pequeños o anchos de banda muy estrechos.

Diseño del filtros IIR

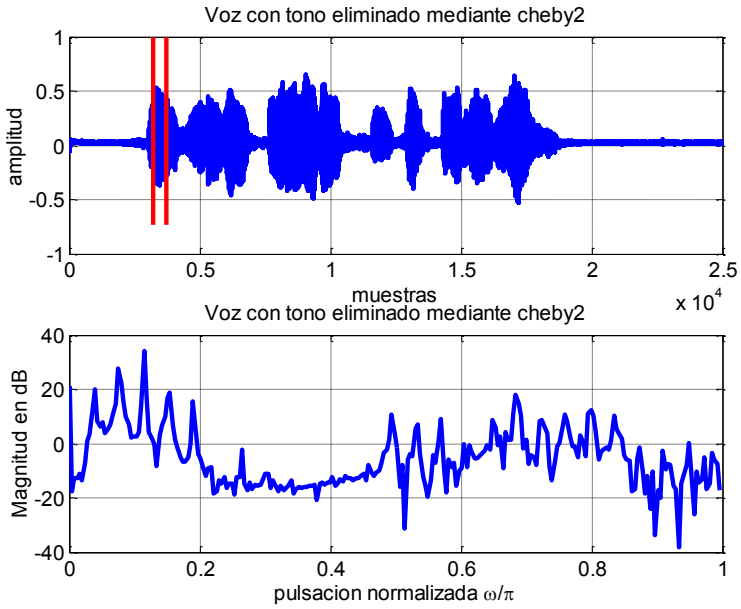
Se trata en este apartado de diseñar un filtro IIR que cumplan las especificaciones (ω_i, H_i) mediante el método de la transformación bilineal.

Calcule e imprima el orden N y los valores ω_n de un filtro de Chebyshev tipo II que cumpla las especificaciones anteriores mediante la función *cheb2ord* de Matlab (los valores *Rp* y *Rs* corresponden a: $Rp=-20log_{10}(1-\delta_1)$ y $Rs=-20log_{10}\delta_2$). Calcule los coeficientes de dicho filtro mediante las función *cheby2* de Matlab. Obtenga la salida a la voz con tono interferente del filtro diseñado utilizando la función *filter* de Matlab.

Represente el espectro de la respuesta al impulso del filtro solapada con las especificaciones deseadas y el diagrama de polos y ceros. ¿Cumple el filtro las especificaciones?



Calcule la salida del filtro de Chebyshev tipo II diseñado en el anterior apartado cuando la entrada es *voztono*. Dibuje la salida del filtro seleccionando un tramo con voz. Calcule y dibuje la magnitud de la DFT del tramo de voz seleccionado. Escuche la salida del filtro. ¿Se ha eliminado el tono interferente?

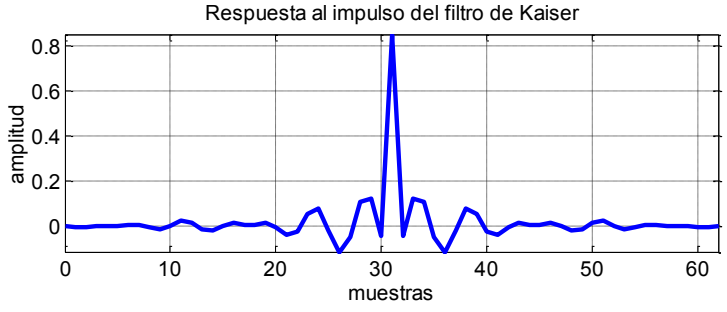


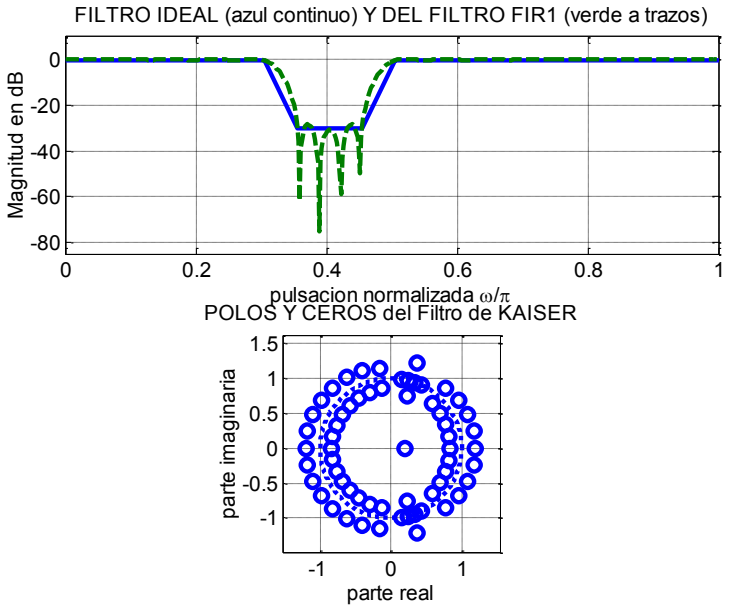
Puede repetir el apartado 3 con un filtro de butterworth (funciones *buttord* y *butter*), chebyshev tipo I (funciones *cheb1ord* y *cheby1*) y elíptico (funciones *ellipord* y *ellip*).

Diseño de filtros FIR mediante enventanado.

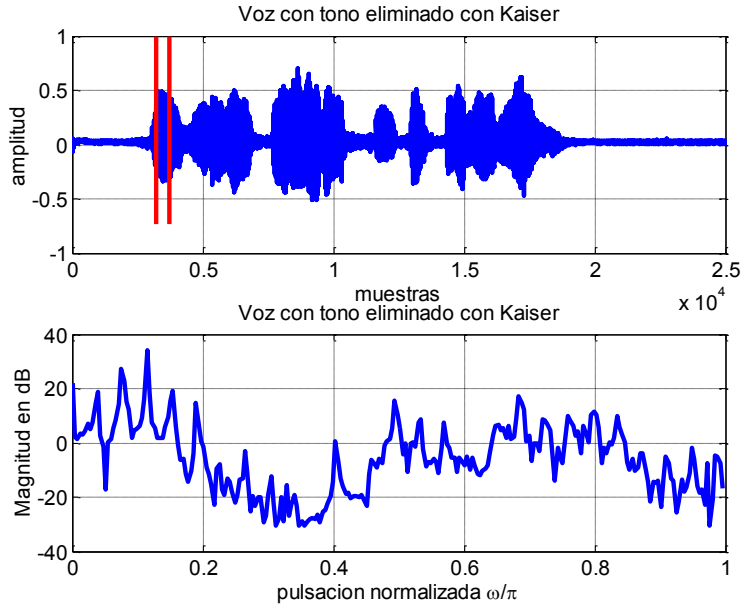
En este apartado se trata de diseñar un filtro FIR que aproxime las especificaciones (ωi,Hi) mediante el método del enventanado. Para utilizar este método, primero hay que elegir la ventana a utilizar, calcular sus parámetros para que el filtro cumpla las especificaciones indicadas y, finalmente, enventanar la respuesta al impulso ideal.

Suponiendo que se utiliza una ventana de Kaiser, estime los parámetros necesarios (ω_c, longitud M de la ventana, y la β) para que el filtro cumpla las especificaciones deseadas (puede utilizar la función *kaiserord* de Matlab). Obtenga, con estos parámetros, la ventana de Kaiser mediante la función *kaiser* de Matlab. Calcule los coeficientes del filtro mediante la función *fir*1 de Matlab. Represente la respuesta al impulso del filtro diseñado, su magnitud y diagrama de polos y ceros.





Calcule la salida del filtro de Kaiser diseñado en el anterior apartado cuando la entrada es *voztono*. Dibuje la salida del filtro seleccionando un tramo con voz. Calcule y dibuje la magnitud de la DFT del tramo de voz seleccionado. Escuche la salida del filtro. ¿Se ha eliminado el tono interferente?

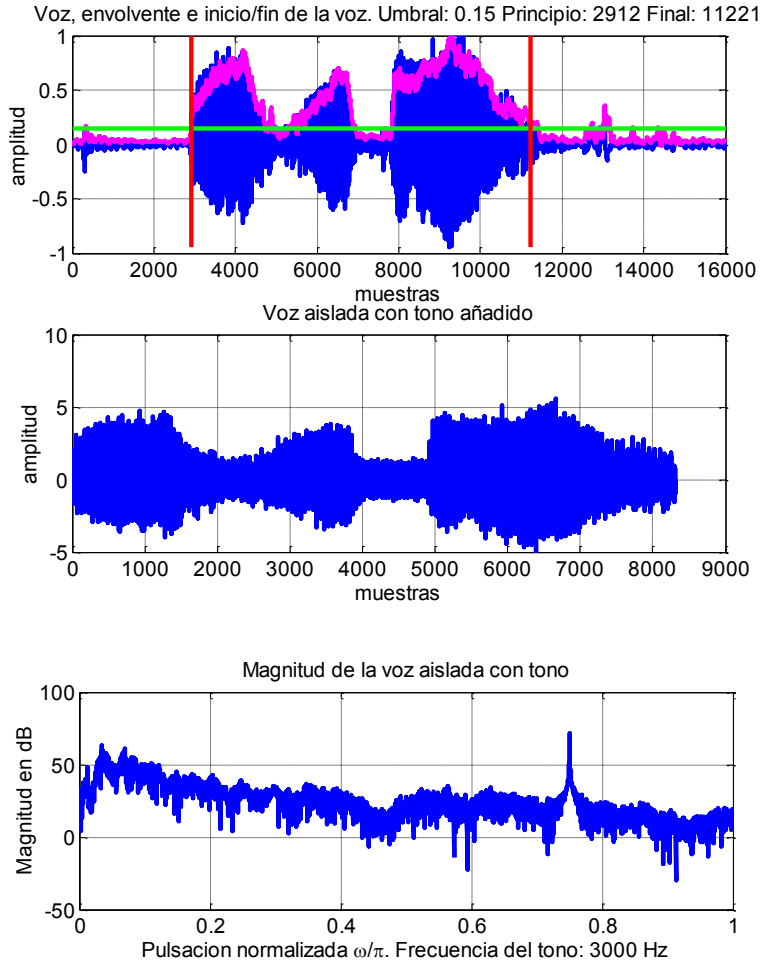


Señal para filtros diseñados por mínimos cuadrados

Para programar y estudiar las propiedades del algoritmo de mínimos cuadrados se utilizara primero un caso sintético.

Mediante la correspondiente utilidad de Matlab, grabe un fichero de aproximadamente 2 segundos de voz. Realice la grabación con frecuencia de muestreo

8000 muestra por segundo, con 16 bits por muestra. Verifique su correcta grabación escuchándola con el comando *soundsc*. Normalice la voz a amplitud máxima igual a 1 y asígnele la variable *voz*. Aísle la voz eliminando los silencios del principio y final de la grabación. Añádale un tono en frecuencia discreta 3*π/4 con una relación entre la potencia de la voz aislada y del tono de 5 dB. Escuche el resultado de añadir el tono interferente a la voz e indique la frecuencia continua del tono. Dibuje la grabación de voz con detección de principio y final, la voz con el tono añadido, y la Magnitud de la transformada de Fourier de la voz con el tono.

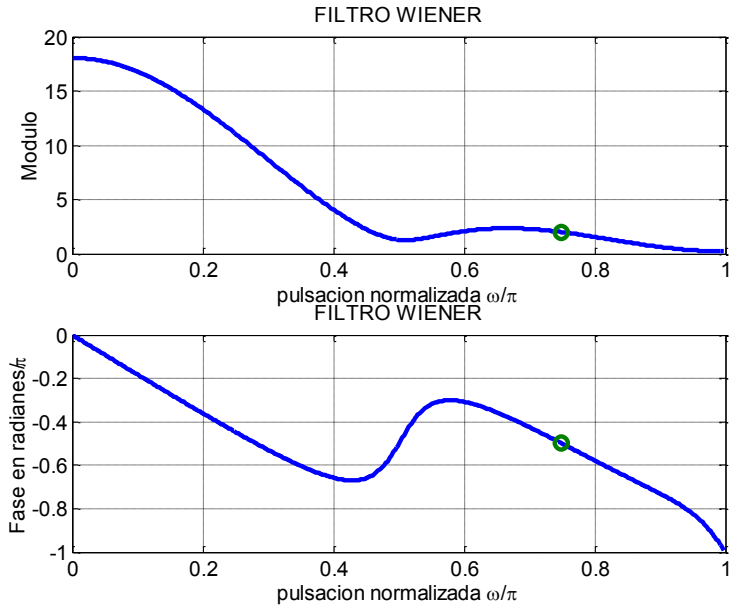


Filtro de mínimos cuadrados

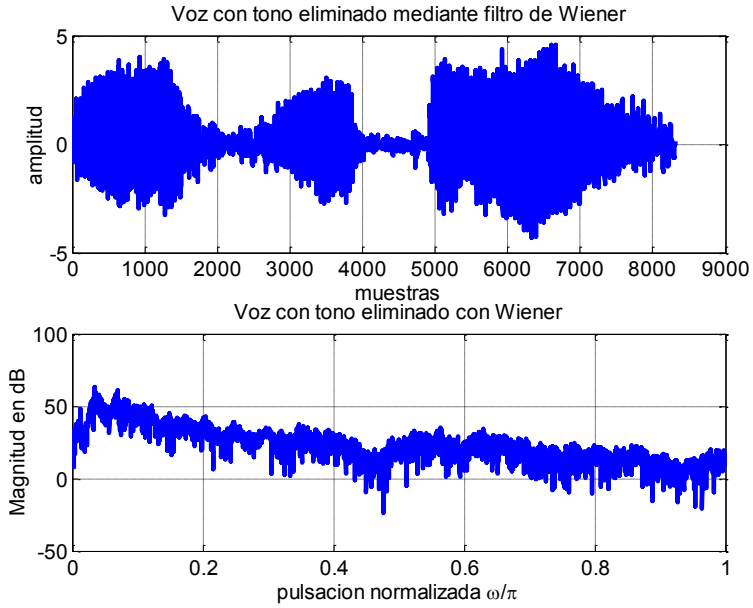
En este apartado se trata de eliminar el tono de la voz diseñando un filtro FIR cuyos coeficientes sean calculados mediante el algoritmo de Wiener. Debe suponer que sólo dispone de la voz con el tono y de la frecuencia del tono.

Dibuje el diagrama de bloques del procedimiento a seguir definiendo la señal de entrada al filtro, la señal deseada y el orden *M* del filtro adaptativo.

Dibuje la magnitud del filtro obtenido así como su fase para verificar que el filtro es capaz de cambiar la amplitud y fase de la sinusoide de entrada igualándola a la amplitud y fase de la sinusoide de la señal deseada.



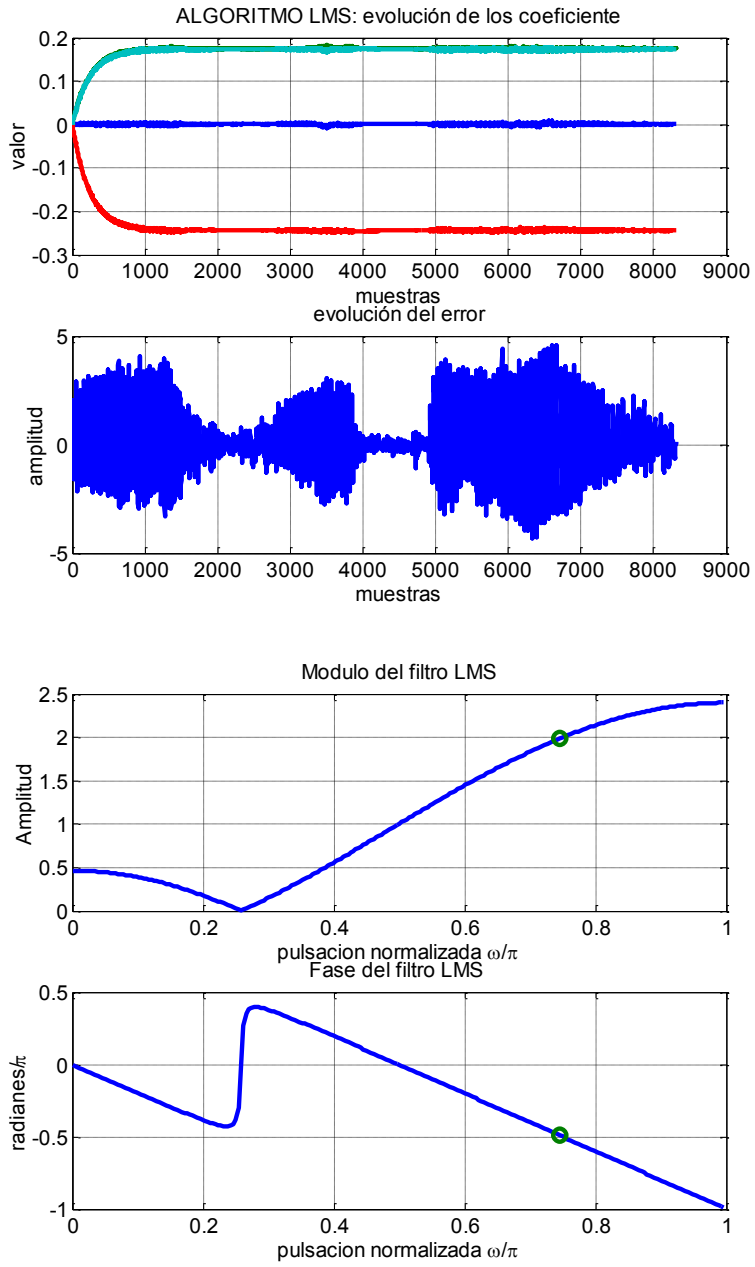
Dibuje la señal con el tono eliminado junto con la magnitud de su transformada de Fourier. Escuche dicha señal. ¿Se ha eliminado el tono interferente?



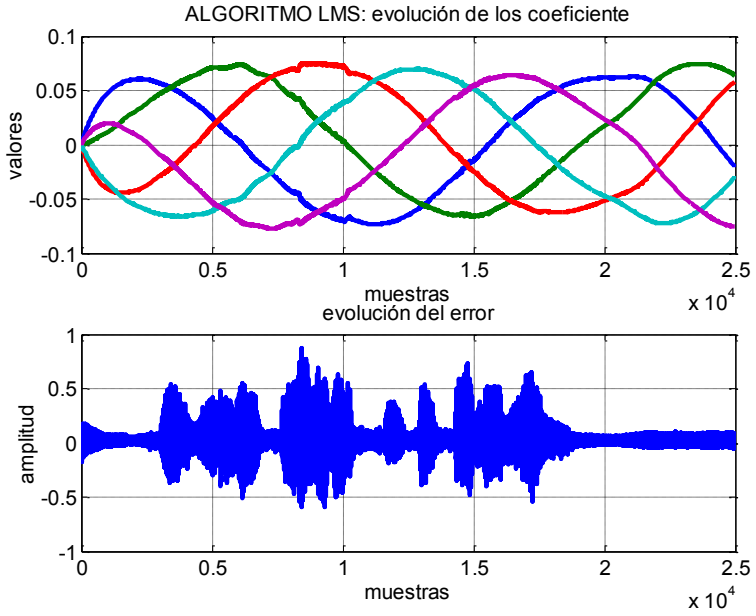
Algoritmo adaptativo de mínimos cuadrados LMS

En este apartado se trata de eliminar el tono de la voz mediante técnicas adaptativas. Dibuje el diagrama de bloques del procedimiento a seguir. Defina el orden M y el paso de adaptación μ para el entrenamiento del filtro adaptativo. Obtenga mediante el algoritmo LMS los coeficientes del filtro adaptativo que elimine el tono.

Para realizar un estudio de la convergencia del algoritmo, represente la evolución del error cometido en cada iteración (curva de aprendizaje) y la evolución de todos los coeficientes del filtro adaptativo en una sola gráfica. Escuche la salida. Dibuje la magnitud y fase del filtro final. Comente los resultados.



Repita este apartado pero en vez de utilizar el tono generado, utilice el tono grabado. Defina de nuevo el orden *M*, la señal de entrada al filtro y la señal deseada, y el paso de adaptación μ para el entrenamiento del filtro adaptativo.



Repita este apartado con el algoritmo LMA

Capítulo V. Muestreo de Señales Continuas

En este capítulo se presentan los métodos de muestreo de señales continuas. Se comenzará con el muestreo temporal para terminar con el muestreo espacial.

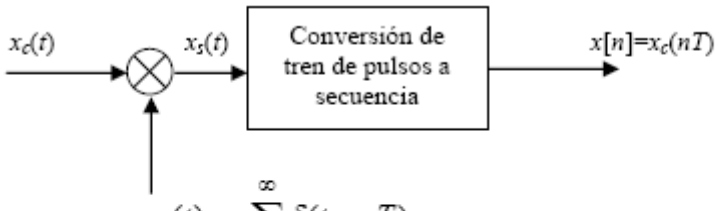
El objetivo de este capítulo es procurar que el alumno compruebe las relaciones existentes entre las señales continuas y discretas, así como los efectos en el dominio de la frecuencia al diezmar o interpolar una señal.

5.1 Muestreo de señales continuas en el tiempo

En muestreo es importante conocer:

a. Bajo qué condiciones una señal continua en el tiempo $x_c(t)$ puedes ser adecuadamente representada con sus muestras $x[n]=x_c(nT)$, siendo T el periodo de muestreo y su inversa $f_s=1/T$ la frecuencia de muestreo.
b. Cuál es la relación entre $X_c(\Omega)$ y $X(\omega)$ siendo $X_c(\Omega)$ la transformada de Fourier de $x_c(t)$ y $X(\omega)$ la transformada de Fourier de $x[n]=x_c(nT)$.

Para responder las anteriores preguntas sobre la relación entre $x_c(t)$ y $x[n]=x_c(nT)$ se recurre al siguiente modelo matemático



$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

$$S(\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_s) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - \frac{2\pi k}{T})$$

Este modelo de muestreo modula la señal continua mediante un tren de deltas seguido por la conversión del tren de deltas a una señal discreta. Mediante el primer paso obtenemos la señal continua $x_s(t)$ de valor cero excepto en los múltiplos enteros de T (instantes de muestreo) en donde vale $x_c(nT)$. La secuencia $x[n]$ se obtiene creando una secuencia ordenada por un índice n con los valores que toma la señal $x_s(t)$ en los instantes nT, perdiendo así la información temporal entre muestra y muestro, esto es, normalizando en el tiempo.

Se analiza dicho modelo en dos pasos,

Primer paso: $x_c(t) \rightarrow x_s(t)$

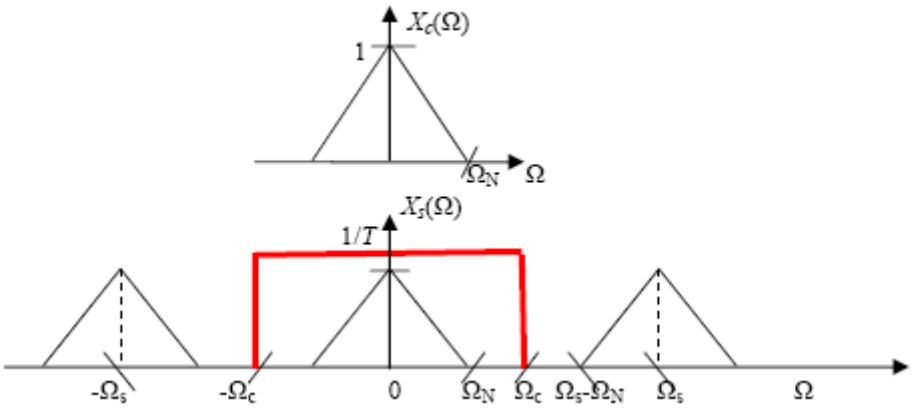
Del modelo anterior se tiene

$$x_s(t) = x_c(t) \cdot s(t) = x_c(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT) \qquad \text{[Ec.57]}$$

aplicando transformada de Fourier

$$X_s(\Omega) = \frac{1}{2\pi} X_c(\Omega) * S(\Omega) = \frac{1}{2\pi} X_c(\Omega) * \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - \frac{2\pi}{T} k) =$$
$$X_s(\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(\Omega - \frac{2\pi}{T} k) \qquad \text{[Ec.58]}$$

Ésta relación muestra que la transformada de Fourier de $x_s(t)$ consiste en copias o réplicas cada $\Omega_s = 2\pi/T$ de la transformada de Fourier de $x_c(t)$.



Por tanto, se puede recuperar $x_c(t)$ a partir de $x_s(t)$ mediante un filtrado paso bajo con frecuencia de corte $\Omega_N < \Omega_c < (\Omega_s - \Omega_N)$ y ganancia T si se cumple:

a. que la señal $x_c(t)$ sea limitada en banda, esto es $X_c(\Omega)=0$ $\Omega \geq \Omega_N$

b. que no se solapen las diferente réplicas del espectro: $\Omega_s - \Omega_N \rangle \Omega_N \Rightarrow \Omega_s \rangle 2\Omega_N$

La condición *b* se conoce con el nombre de teorema de Nyquist.

Segundo paso: $x_s(t) \rightarrow x[n]$

La relación entre $x_s(t)$ y $x[n]$ se obtiene comparando ambas transformadas de Fourier

La transformada de Fourier de $x_s(t)$ *es*

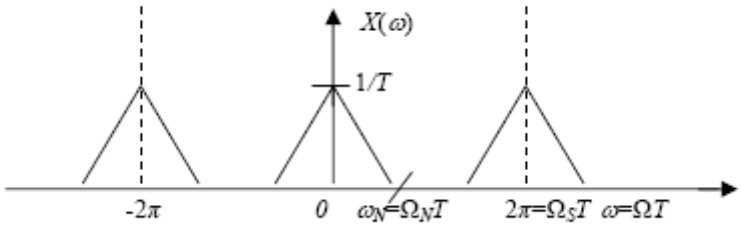
$$X_s(\Omega) = \int_{-\infty}^{+\infty} x_s(t) \cdot e^{-j\Omega t} dt = \sum_{n=-\infty}^{\infty} x_c(nT) \cdot e^{-j\Omega nT} = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-j\Omega nT} \qquad \text{[Ec.59]}$$

y la de $x[n]$

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n} \qquad \text{[Ec.60]}$$

identificando términos:

$$X(\omega) = X_s(\Omega)\Big|_{\Omega=\frac{\omega}{T}} = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(\Omega - \frac{2\pi}{T} k\right)\Big|_{\Omega=\frac{\omega}{T}} \qquad \text{[Ec.61]}$$



Consideraciones prácticas

Actualmente la tecnología es capaz de obtener una señal $x[n]$ suficientemente aproximada a la obtenida por el modelo matemático. Tan sólo habría que tener en cuenta el filtro antialiasing (filtro paso bajo con frecuencia de corte π/T) previo al proceso de muestreo con el fin de evitar solapamiento espectral en el proceso de muestreo, y el ruido de cuantificación.

Conversión discreto-continuo

Se estudia ahora la recuperación de $x_c(t)$ a partir de $x[n]$. Al igual que en el caso anterior, se realiza en dos pasos con el mismo modelo matemático:

Primer paso: $x[n] \rightarrow x_s(t)$

Se introduce la información temporal del periodo de muestreo en $x[n]$

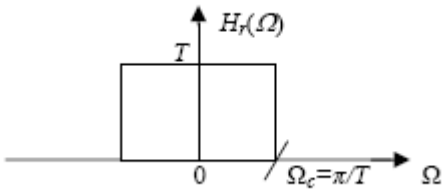
$$x_s(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot \delta(t - nT) \qquad \text{[Ec.62]}$$

cuya expresión equivalente en frecuencia es $X_s(\Omega) = X(\omega)\big|_{\omega=\Omega T}$

Segundo paso: $x_s(t) \rightarrow x_r(t)$ (señal $x_c(t)$ reconstruida)

Se realiza mediante un filtro paso bajo $H_r(\Omega)$ de frecuencia de corte $\Omega_N<\Omega_c<(\Omega_s-\Omega_N)$ y ganancia T. Generalmente se usa $\Omega_c=\pi/T$, esto es:

$$H_r(\Omega)=\begin{cases} T & |\Omega|<\pi/T \\ 0 & resto \end{cases}$$



su respuesta al impulso viene dada por:

$$h_r(t)=\frac{sen(\pi t/T)}{\pi t/T}=sinc(t/T) \qquad \text{señal que se anula cuando } t=nT \qquad \text{[Ec.63]}$$

de donde la señal reconstruida es:

$$x_r(t)=x_s(t)*h_r(t)=\int_{-\infty}^{\infty}x_s(\tau)h_r(t-\tau)d\tau=\sum_{n=-\infty}^{\infty}x_s(nT)h_r(t-nT) \qquad \text{[Ec.64]}$$

esto es, la señal reconstruida es una combinación lineal de funciones *sinc* desplazadas a cada muestra nT y de amplitud igual a su muestra. En los instantes nT la señal vale $x_s(nT)$ y en el resto de instantes toma un valor igual a la contribución (combinación lineal) de todas las *sinc*s. Este filtro puede verse como una fórmula de interpolación entre muestra y muestra.

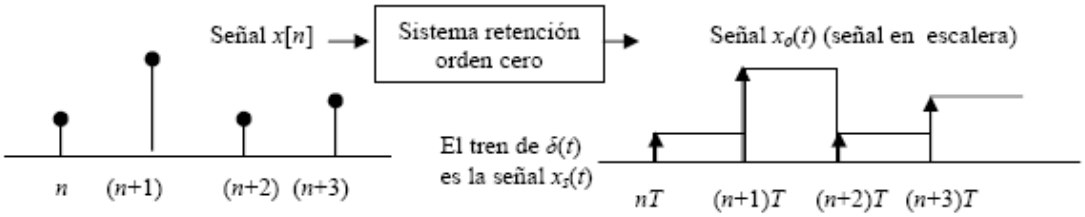
El espectro de la señal reconstruida es:

$$X_r(\Omega)=H_r(\Omega)X_s(\Omega)=H_r(\Omega)X(\Omega T)=\begin{cases} TX(\Omega T) & |\Omega|<\pi/T \\ 0 & resto \end{cases} \qquad \text{[Ec.65]}$$

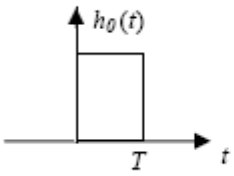
Consideraciones prácticas

El proceso de reconstrucción de la señal discreta descrito no es viable tecnológicamente pues el filtro interpolador $H_r(\Omega)$ es ideal. Por esta razón se recurre a aproximaciones. Dos de las más utilizadas son la interpolación de orden cero y la interpolación lineal.

La interpolación de orden cero consiste en reconstruir la señal como una escalera mediante un sistema de retención de orden cero obteniendo la señal $x_o(t)$, tal y como se ilustra en la figura (existen efectos adicionales como el *decay* y los *glitches* que no se tienen en cuenta en este capítulo).



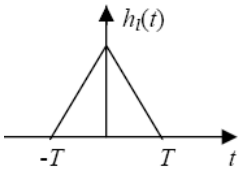
La señal $x_o(t)$ equivale a filtrar la señal $x_s(t)$ por un filtro con respuesta al impulso $h_0(t)$



$$h_0(t)=\begin{cases}1 & 0<t<T\\ 0 & resto\end{cases} \qquad H_0(\Omega)=\frac{2\sin(\Omega T/2)}{\Omega}e^{-j\Omega T/2}$$

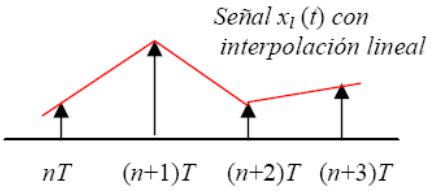
de donde $X_o(\Omega)=X(\Omega T)H_0(\Omega)$

La segunda aproximación consiste en utilizar una interpolación lineal, esto es, unir las deltas de $x_s(t)$ mediante una línea recta para obtener la señal $x_l(t)$ tal y como se muestra en la figura. Este caso equivale a filtrar la señal $x_s(t)$ por un filtro de respuesta al impulso $h_l(t)$



$$h_l(t)=\begin{cases}1-|t|/T & |t|<T\\ 0 & resto\end{cases}$$

$$H_l(\Omega)=\left(\frac{2\sin(\Omega T/2)}{\Omega}\right)^2$$



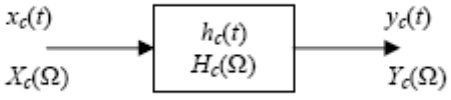
En este caso, la señal reconstruida vendría dada por:

$$X_l(\Omega)=X(\Omega T)H_l(\Omega) \qquad \text{[Ec.66]}$$

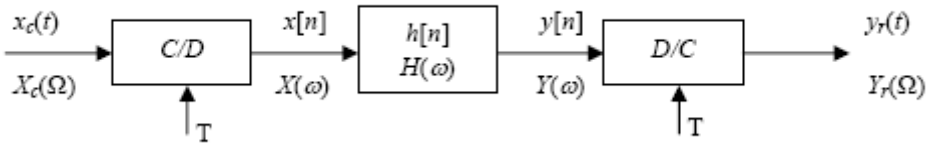
Para evitar que el filtro reconstructor deje pasar parte de las réplicas espectrales de $x_s(t)$, se disminuye T para que las réplicas espectrales estén más separadas y el filtro reconstructor tenga espacio para la banda de transición. En el resto del capítulo se obvia éste hecho y se sigue trabajando con el $H_r(\Omega)$ ideal.

Aplicación: simulación discreta de sistemas continuos

El objetivo de este apartado es, partiendo del sistema LTI continuo



diseñar un sistema discreto $h[n]$



tal que $y_r(t)= y_c(t)$ o $Y_r(\Omega)= Y_c(\Omega)=H_c(\Omega)X_c(\Omega)$, suponiendo que el valor de T es tal que no hay aliasing en el muestreo (Esto supone $X_c(\Omega)=0\ \Omega\geq\Omega_N$ y $\Omega_s=2\pi/T>2\ \Omega_N$)

Solución: $h[n]$ se obtienen igualando $Y_r(\Omega)= Y_c(\Omega)$. Para ello, como:

$$Y_r(\Omega)=H_r(\Omega)Y(\Omega T)=H_r(\Omega)H(\Omega T)X(\Omega T)=H_r(\Omega)H(\Omega T)\frac{1}{T}\sum_{k=-\infty}^{\infty}X_c(\Omega-k\Omega_s)$$

se debe cumplir

$$H_r(\Omega)H(\Omega T)\frac{1}{T}\sum_{k=-\infty}^{\infty}X_c(\Omega-k\Omega_s)=H_c(\Omega)X_c(\Omega) \qquad \text{[Ec.67]}$$

esto es , suponiendo que no hay solapamiento espectral en el proceso de muestreo y que $H_r(\Omega)$ es ideal de ganancia T:

$$H_c(\Omega)=H(\Omega T) \qquad |\Omega|<\pi/T \qquad \text{[Ec.68]}$$

que equivale a $H(\omega)=H_c(\omega/T) \qquad |\omega|<\pi$

esto es $h[n]= h_c(nT)$

Este método se denomina de invarianza al impulso y tiene el problema de tener que truncar la $h[n]$ en el caso de sistemas IIR.

5.2 Métodos de Interpolación y diezmado

A menudo es necesario disponer de la misma señal $x_c(t)$ muestreada a diferentes frecuencias de muestreo.

- ✓ Sea $x_1[n]$ la señal discreta obtenida al muestrear $x_c(t)$ con un periodo T_1.
- ✓ Sea $x_2[n]$ la señal discreta obtenida al muestrear $x_c(t)$ con un periodo T_2.

El objetivo de este apartado es obtener $x_2[n]$ a partir de $x_1[n]$ en el dominio discreto sin necesidad de volver a muestrear $x_c(t)$.

Para ello se consideran dos casos:

1. Diezmado, donde $T_2=M\,T_1$, siendo M un número entero.
2. Interpolación, donde $T_2= T_1/L$, siendo L un número entero.

Diezmado

Tomando como punto de partida $x_1[n]=x_c(nT_1)$ con $X_c(\Omega)=0 \quad |\Omega|>\Omega_N$

Y queriendo obtener $x_2[n]=x_c(nT_2)$ con $T_2=MT_1$

El paso en discreto se realiza:

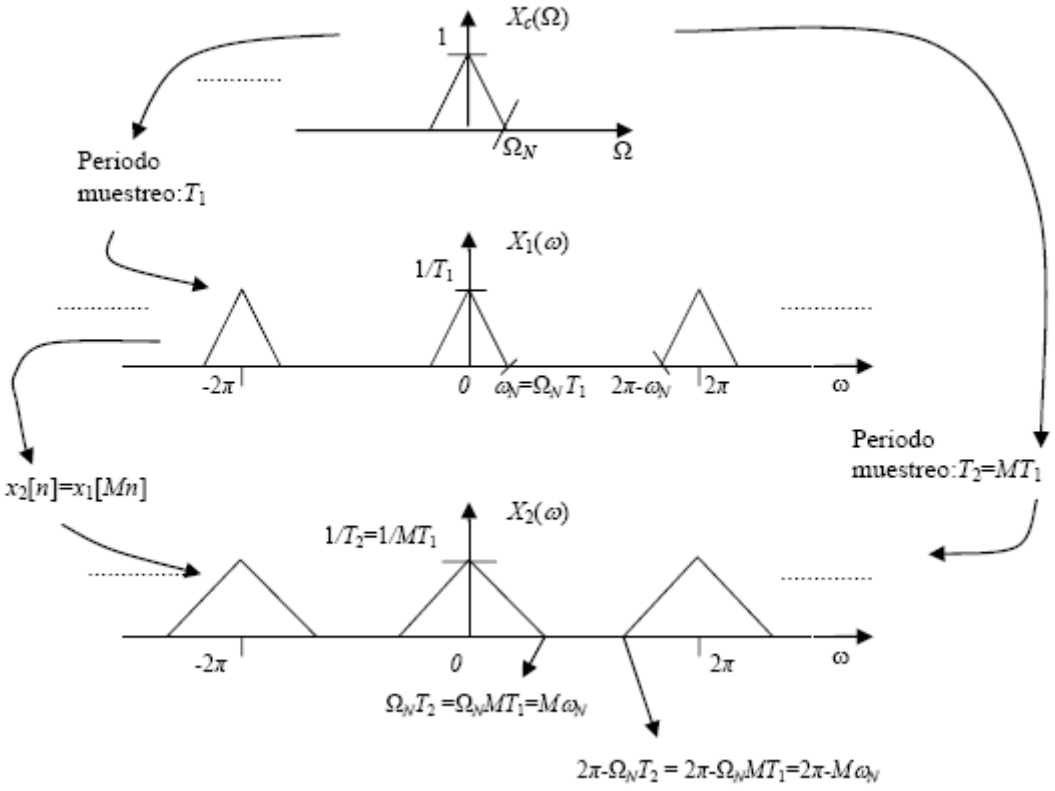
$x_2[n]=x_c(nT_2)= x_c(nMT_1)=x_1[nM]$

esto es, $x_2[n]$ se obtiene seleccionando una muestra cada M de $x_1[n]$. Por tanto equivale a "muestrear" la señal muestreada.

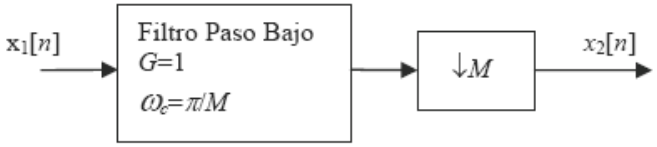
El efecto en frecuencia del paso de $x_1[n]$ a $x_2[n]$ se obtiene calculando $X_2(\omega)$ en función de $X_1(\omega)$ como sigue:

$$X_2(\omega)=\frac{1}{M}\sum_{r=0}^{M-1}X_1\left(\frac{1}{M}(\omega-2\pi r)\right)$$

así vemos que al diezmar $x_1[n]$ el espectro $X_1(\omega)$ reduce su amplitud un factor M a la vez que se ensancha un factor M pudiendo provocar solapamiento espectral. Un ejemplo gráfico de este proceso es:



Pare evitar dicho solapamiento debe garantizarse que la máxima frecuencia $\omega_N=\Omega_N T_1$ de $X_1(\omega)$ cumpla $\omega_N \le \pi/M$. Esto puede garantizarse filtrando la señal $x_1[n]$ por un filtro paso bajo de frecuencia de corte π/M y ganancia unidad antes de diezmar la señal.



Interpolación

Tomando como punto de partida $x_1[n]=x_c(nT_1)$ con $X_c(\Omega)=0 \quad |\Omega|>\Omega_N$

Y queriendo obtener $x_2[n]=x_c(nT_2)$ con $T_2=T_1/L$

Lo cual equivale a una señal $x_2[n]$ obtenida al muestrear con una frecuencia de muestreo mayor. Así hay que añadir muestras entre las muestras de $x_1[n]$ ya existentes.

La obtención de $x_2[n]$ a partir de $x_1[n]$ en discreto se realiza en dos pasos:

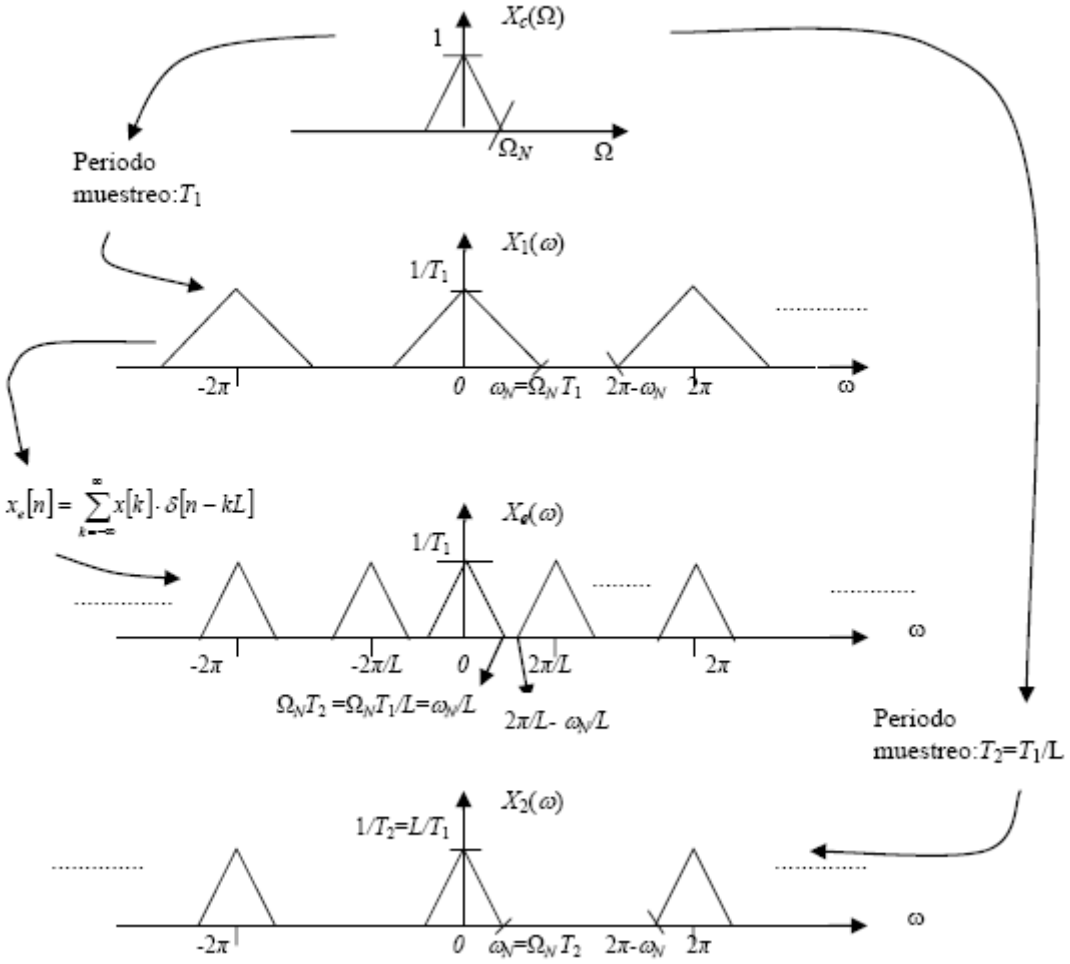
Paso 1. Se añaden L-1 ceros entre muestra y muestra de $x_1[n]$ obteniendo $x_e[n]$

$$x_e[n]=\begin{cases} x_1[n/L] & n\ múltiplo\ \ de\ L \\ 0 & resto \end{cases} = \sum_{k=-\infty}^{\infty} x_1[k]\delta[n-kL]$$

cuyo espectro es:

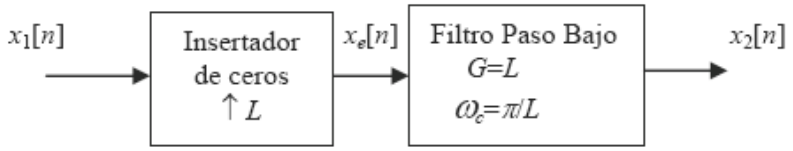
$$X_e(\omega)=\sum_{k=-\infty}^{\infty} x_1[k]e^{-j\omega kL} = X_1(\omega L)$$

esto es, el espectro se comprime un factor L al insertar L ceros entre muestra y muestras. Se puede mostrar este efecto gráficamente de la siguiente manera donde se muestra claramente la necesidad de filtrar paso bajo el espectro $X_e(\omega)$ para obtener $X_2(\omega)$.



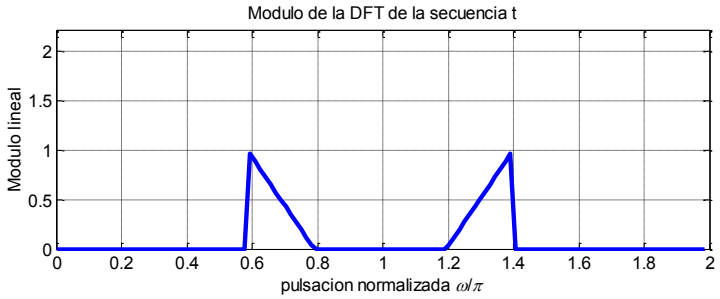
Paso 2. Para eliminar las réplicas del espectro que aparecen al insertar ceros, se filtra la señal $x_e[n]$ por un filtro paso bajo de ganancia L y frecuencia de corte $\frac{\omega_N}{L} \le \omega_c \le \frac{2\pi}{L} - \frac{\omega_N}{L}$ (generalmente $\omega_c=\pi/L$).

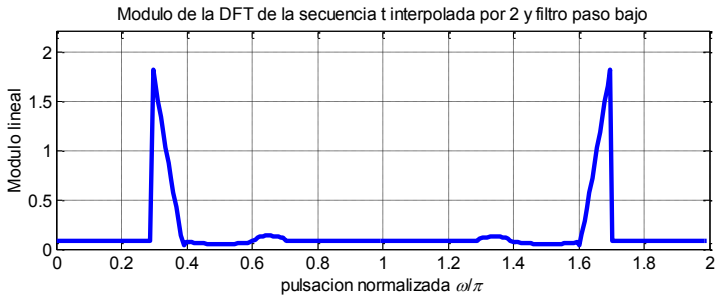
Así, el procedimiento general para interpolar es:



Ejemplo práctico en Matlab

```
% se interpola por L
L=2;
% Se genera la secuencia con espectro triangular
t=triang(26);
t=[zeros(38,1); t(14:26); zeros(13,1)];
t=[t ; t(length(t):-1:1)];
t=ifft(t);
% Se interpola la secuencia t sin utilizar filtro.
ti1=zeros(L*length(t),1);
ti1(1:L:length(ti1))=t;
% Se repite lo anterior con el filtro paso bajo
[ti2,h]=interp(t,2);
```

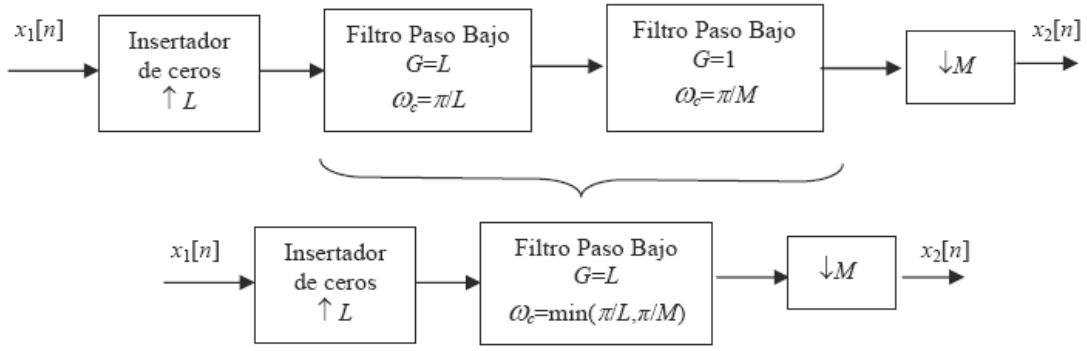




Para cambiar la frecuencia de muestreo por un factor racional: $T_2=MT_1/L$ se realiza primero la interpolación y luego el diezmado. La razón es minimizar los problemas de solapamiento espectral, puesto que al estrechar primero el espectro con la interpolación será menos probable que el posterior ensanchamiento debido al diezmado produzca solapamiento espectral.

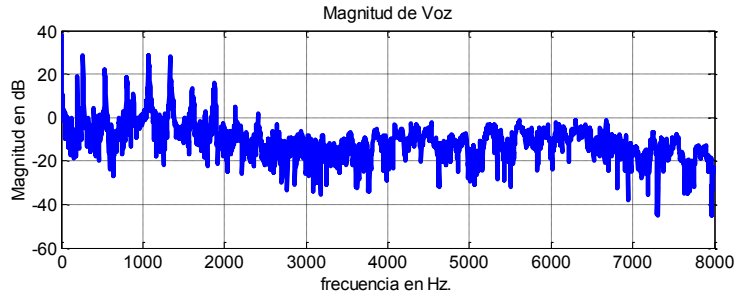
El diagrama de bloques utilizado para cambiar por un factor racional la frecuencia de muestreo es:

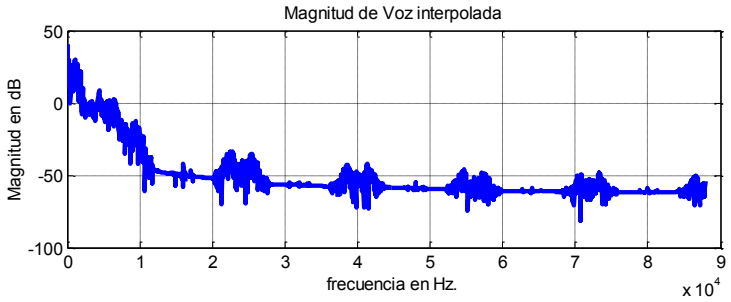
$x_1[n]=x_c(nT_1) \rightarrow x_2[n]=x_c(nT_2)= x_c(nMT_1/L)$

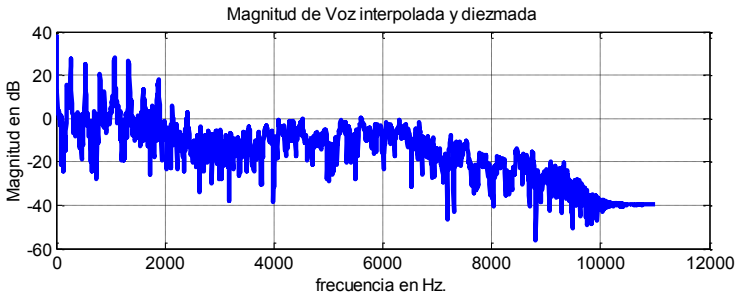


Ejemplo en Matlab

```
[voz,fs]=wavread('vocal');
% Se pasa la voz de 8000 a 11000 muestras/seg en vozM.
vozI=interp(voz,11);
vozM=decimate(vozI,8);
% equivale a
%vozM=resample(voz,11,8);
```





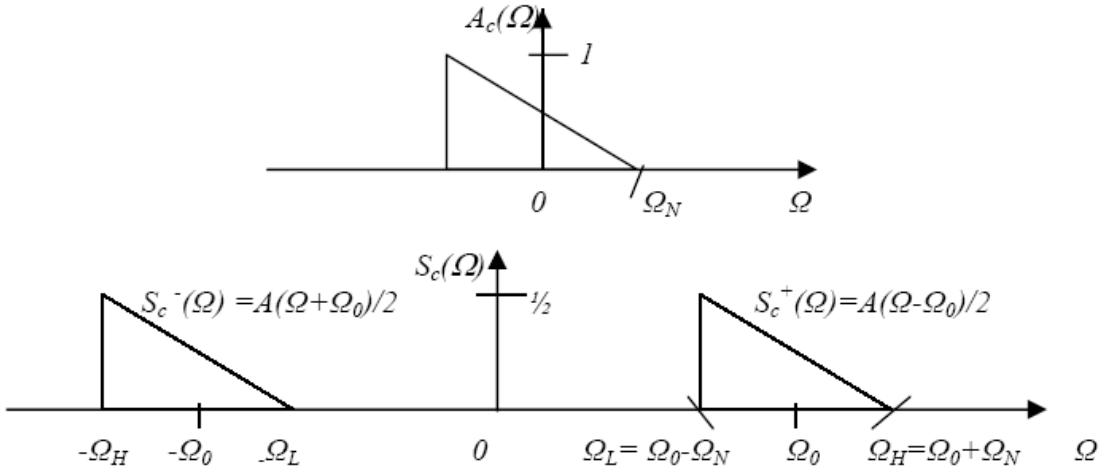


Aplicación: submuestreo

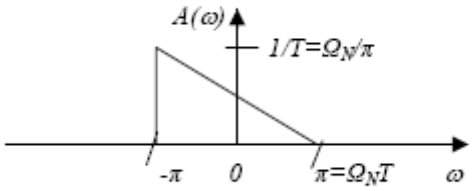
Una señal paso banda es de la forma:

$$s_c(t) = a_c(t)\cos(\Omega_0 t)$$

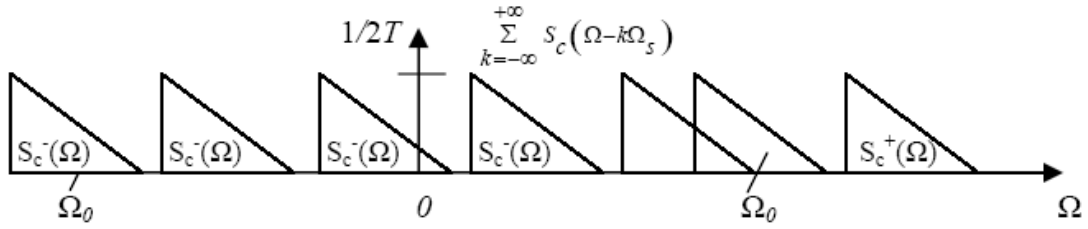
donde $a_c(t) = r(t)\cos(\phi(t)) + jr(t)\sin(\phi(t))$ es la señal compleja de transformada de Fourier $A_c(\Omega)=0 \ \ |\Omega| > \Omega_N$



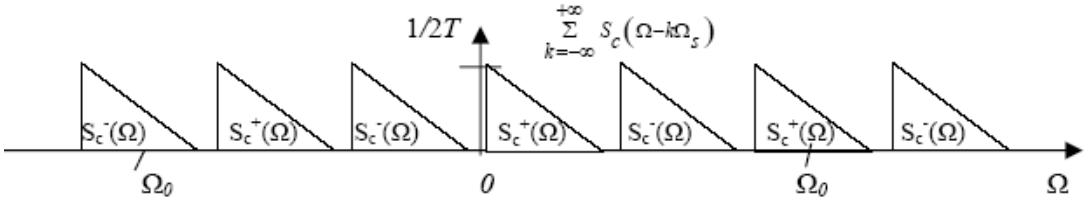
El objetivo de este apartado es obtener $a[n]=a_c(nT)$, cuya transformada de Fourier $A(\omega)$ será la de la figura, muestreando $s_c(t)$ con la menor frecuencia de muestreo $1/T$ posible.



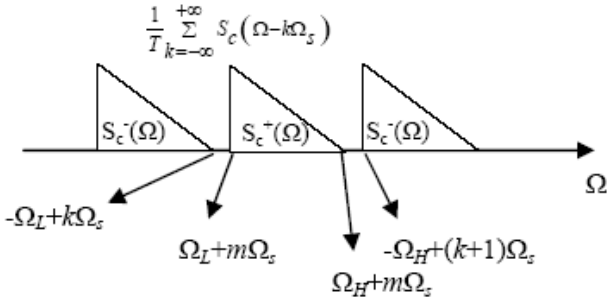
Muestrear $s_c(t)$ según el principio de Nyquist para posteriormente obtener $a_c(t)$ de forma discreta implica $\Omega_s=2\pi/T > 2(\Omega_0 + \Omega_N)$, lo cual puede provocar problemas tecnológicos sin contar con el derroche de ancho de banda que este proceder supone. El problema al muestrear directamente $s_c(t)$ con $\Omega_s > 2\Omega_N$ es que las réplicas del espectro positivo $S_c^+(\Omega)$ se solapen con las réplicas del espectro negativo $S_c^-(\Omega)$.



El concepto del submuestreo es buscar una frecuencia de muestreo Ω_s tal que no provoque solape entre las réplicas de $S_c^+(\Omega)$ y $S_c^-(\Omega)$.



para buscar las relaciones que deben cumplir Ω_0, Ω_s, y Ω_N para que no se solapen, se recurre al siguiente gráfico donde se relacionan la réplica k y k+1 del espectro negativo y la réplica m del espectro positivo sin solaparse.



Para que no exista solape se debe cumplir, para todo k y m

$-\Omega_L + k\cdot\Omega_s \leq \Omega_L + m\Omega_s$
$\Omega_H + m\Omega_s \leq -\Omega_H + (k+1)\Omega_s$
de donde

$$\Omega_s \leq \frac{2\Omega_L}{(k-m)} \qquad \text{y} \qquad \Omega_s \geq \frac{2\Omega_H}{(k-m+1)}$$

por tanto:

$$\frac{2\Omega_H}{k-m+1} \leq \Omega_s \leq \frac{2\Omega_L}{k-m}$$

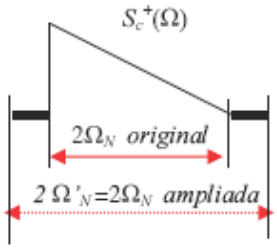
en el caso de igualdad y realizando el cambio de variable k-m+1=q

$$\Omega_s = \frac{2\Omega_L}{q-1} = \frac{2\Omega_H}{q}$$

como $\Omega_L = \Omega_H - 2\Omega_N$

$$\frac{2(\Omega_H - 2\Omega_N)}{q-1} = \frac{2\Omega_H}{q}$$

se obtiene la relación: $q = \frac{\Omega_H}{2\Omega_N}$ (número entero)

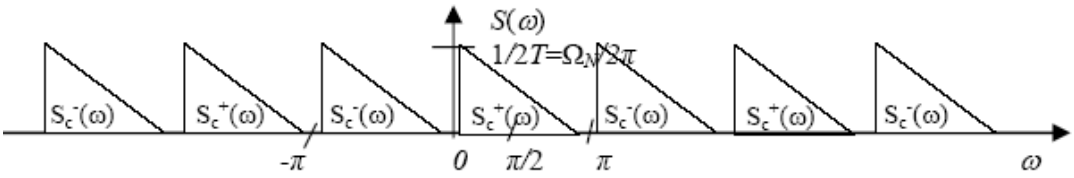


por tanto, para que no exista solape, Ω_H tiene que ser múltiplo de $2\Omega_N$. En el caso de que no sea entero se ensancha tal y como indica la figura.

Una vez ajustado Ω_N, la frecuencia de muestreo para que no exista solape será:

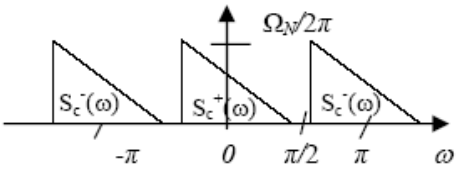
$$\Omega_s = \frac{2\Omega_H}{q} = \frac{2\Omega_H}{\Omega_H / 2\Omega_N} = 4\Omega_N \qquad (4\Omega'_N \text{ en caso de ensanche})$$

El resultado al muestrear $s_c(t)$, con q impar y $\Omega_s = 4\Omega_N$, es:

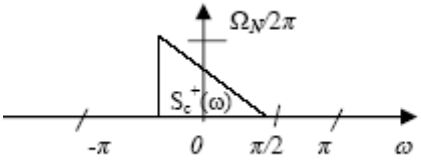


de donde para obtener la señal buscada $a[n] = a_c(nT)$ con espectro $A(\omega)$ hay que:

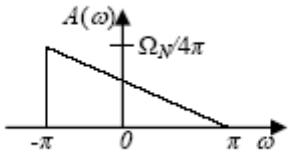
- Desplazar $S(\omega)$ hacia la izquierda $\pi/2$ multiplicando $s[n]$ por $e^{-j\pi n/2}$.



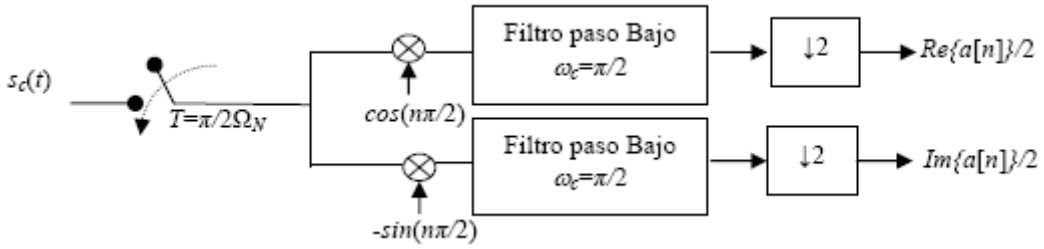
- Se filtra paso bajo con frecuencia de corte $\pi/2$



- Se diezma por dos obteniendo $a[n]$.

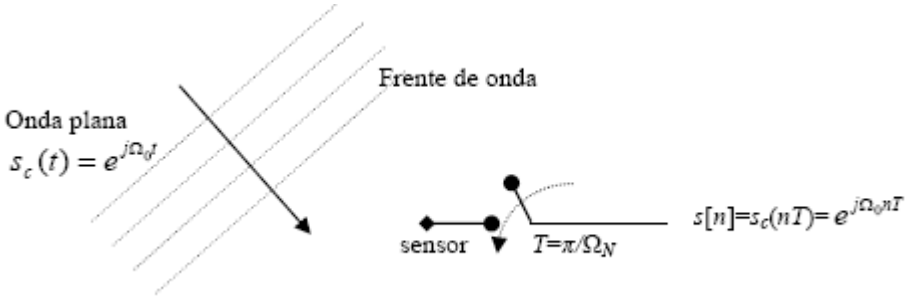


Así, el esquema completo de este procedimiento es:



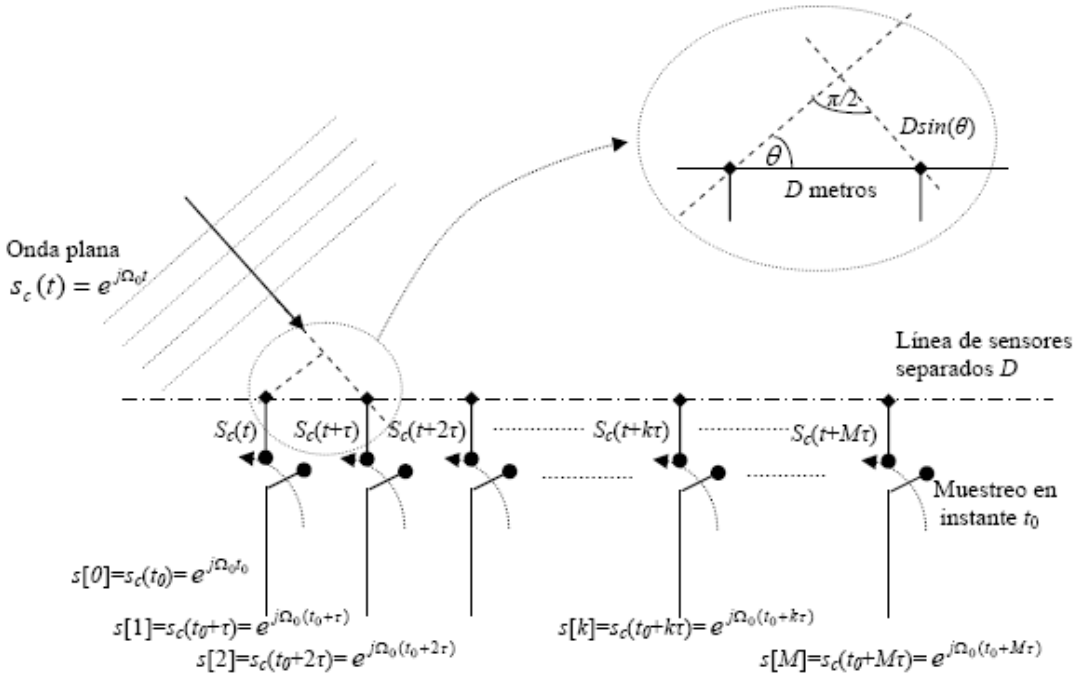
5.3 Muestreo espacial de señales continuas

Hasta el momento hemos considerado el caso de muestrear una señal con un sensor en un punto determinado a intervalos de tiempo regulares *T*. En el caso que dicha señal sea una onda plana $s_c(t) = e^{j\Omega_0 t}$ propagándose en el espacio a velocidad *c,* el esquema propuesto es:



donde la transformada de Fourier de *s*[*n*] es una *sinc* con el máximo $\omega_0=\Omega_0 T$ indicando la frecuencia de dicha onda (conocido *T*). Así con un filtro puedo separar señales que no estén solapadas en frecuencia.

En este apartado se trata de, en vez de muestrear en un punto, muestrear en múltiples puntos equiespaciados *D* metros como en la siguiente figura



obteniendo la señal $s[k]=s_c(t_0+k\tau)=e^{j\Omega_0(t_0+k\tau)}=e^{j\Omega_0 t_0}e^{j\Omega_0 k\tau}=cte\cdot e^{j\Omega_0 k\tau}$

si se realiza la transformada de Fourier de $s[k]$ se obtendrá una *sinc* con el pico en $\omega_0=\Omega_0\tau$ indicando el retardo de la onda entre sensor y sensor (conocido Ω_0).

Conocer el retardo equivale a conocer la dirección de llegada de la onda (definida como el ángulo entre la línea de sensores y el frente de onda) pues el tiempo τ que tarda la señal en llegar de un sensor al siguiente que viene dado por

$$\text{tiempo}=\frac{\text{espacio}}{\text{velocidad}}=\tau=\frac{D\sin(\theta)}{c}$$

entonces el ángulo de llegada viene dado por:

$$\omega_0=\Omega_0\tau=\Omega_0\frac{D\sin(\theta)}{c}\rightarrow\theta=\text{asin}(\frac{\omega_0 c}{\Omega_0 D})$$

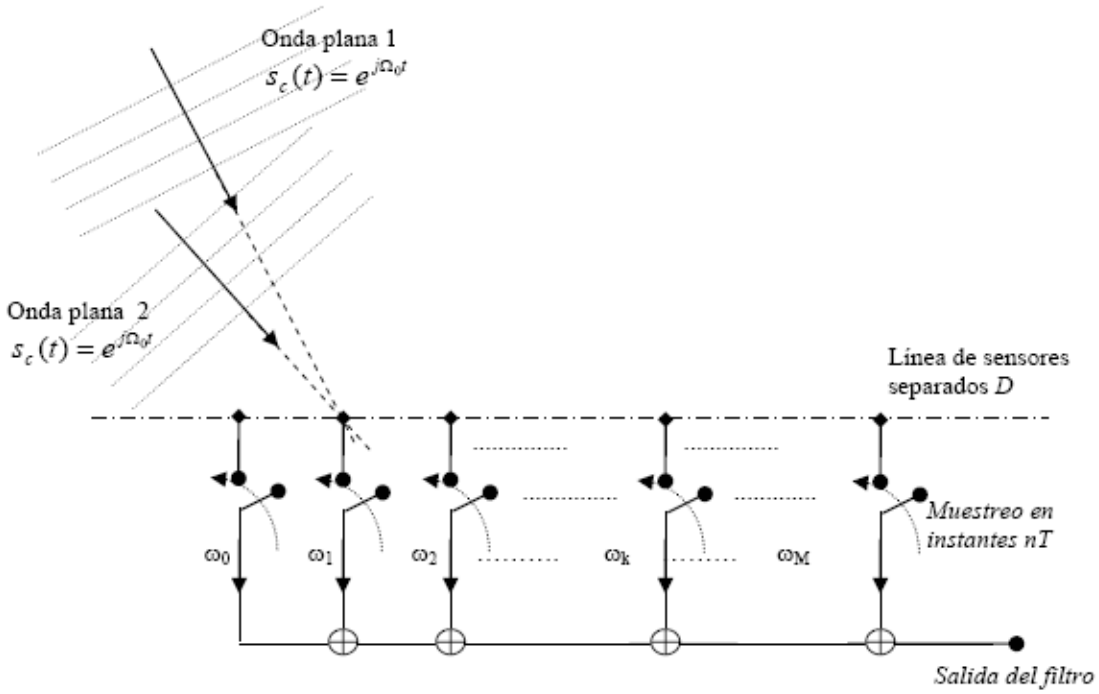
Téngase en cuenta que el pico de la *sinc* estará comprendido entre $0\le\omega_0\le\frac{\Omega_0 D}{c}$ (para que la función *asin* no tenga un argumento mayor que uno. Si la onda llega con ángulo *π/2* y la D es la máxima para que no halla aliasing, el cociente es igual a 1).

Y al igual que en el muestreo temporal, debe existir una distancia D_{max} entre sensores para que no exista aliasing. Puesto que el periodo de muestreo espacial es *D* (implica frecuencia de muestreo espacial 1/*D*), y la frecuencia espacial de la onda es el

inverso de la longitud de onda $1/\lambda=f_0/c$ ($\Omega_0=2\pi f_0$), la $D_{máxima}$ se calcula aplicando el criterio de Nyquist, esto es:

$$f_{muestreo} \geq 2\, f_{máxima\ de\ la\ señal} \quad \rightarrow \quad \frac{1}{D} \geq \frac{2}{\lambda} \quad \rightarrow \quad D_{máxima} \leq \frac{\lambda}{2}$$

Como conclusión de lo anterior, al muestrear espacialmentre la señal se puede conocer la dirección de llegada de las ondas planas al array de sensores y, con un filtro, separar las ondas incidentes según su dirección de llagad aunque tengan la misma Ω_0. Un ejemplo de diseño de dicho filtro es el siguiente

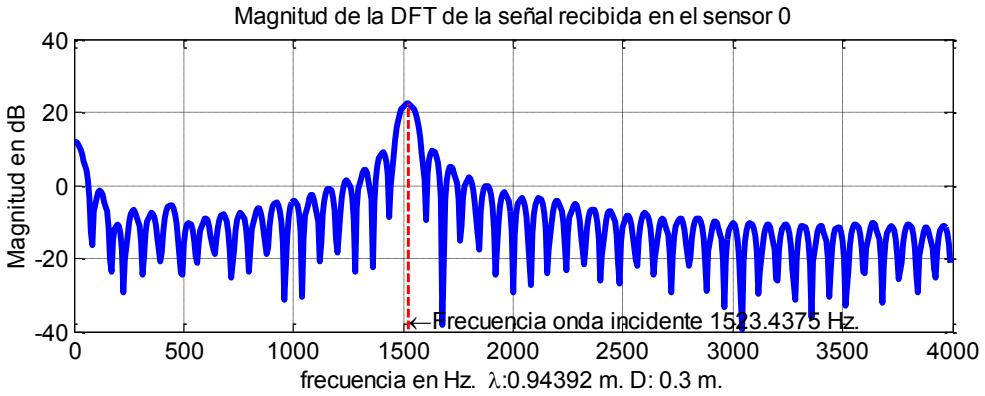


Un ejemplo de diseño de este filtro sería el siguiente. Suponga que se tienen M=16 sensores, que la distancia entre sensores es D=30*cm*., que la velocidad en el medio es c=1438 *metros/segundo* que T=1/8000 *segundos* y que la frecuencia de las dos ondas incidentes es f_0=1600*Hz*.

Suponga que tenemos en las columnas de una matriz denominada *array* 100 muestras recogidas de cada sensor (matriz de 100 filas y M=16 columnas).

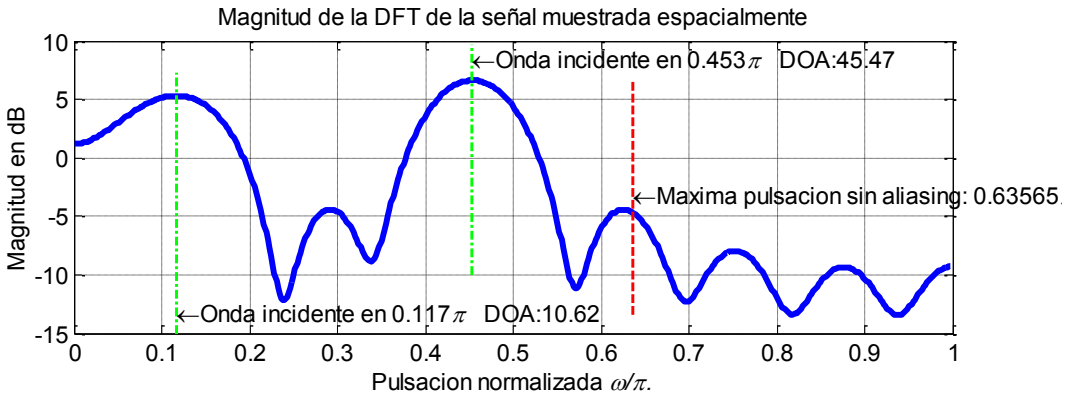
Para detectar la frecuencia temporal de las ondas incidentes se realizaría una transformada de Fourier de una columna

```
% datos del array
load array
% velocidad de la onda en el medio
c=1438;
% numero de sensores
nsensor=16;
% frecuencia de muestreo
fs=8000;
% distancia entre sensores (en metros)
dsensor=0.3;
% frecuencia de la onda en Hz
[S,w]=freqz(array(:,1),1,512,8000);
S=abs(S).^2;
[Y,I]=max(S);
fo=w(I);
% se calcula la landa
landa=c/fo;
```



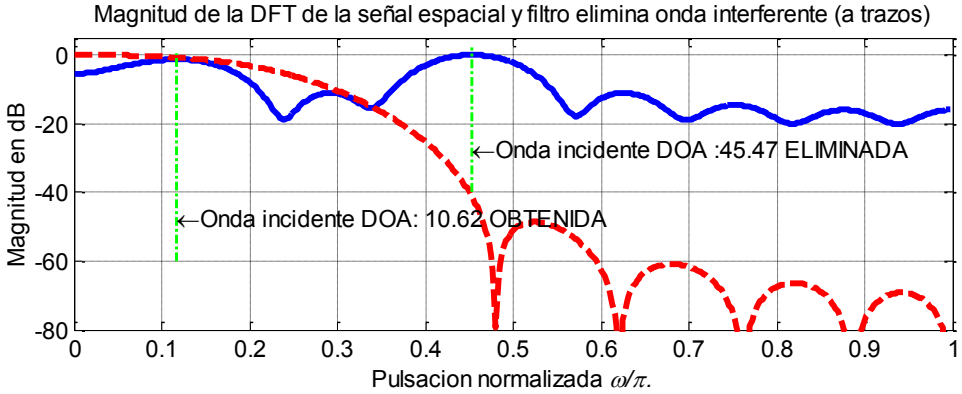
como se cumple el criterio de Nyquist tanto en el muestreo temporal como en el muestreo espacial, se pasa a calcular las direcciones de llegada de las ondas incidentes

```
% máximo angulo
maxang=2*pi*fo*dsensor/c;
% número de ondas planas
nondasplanas=2;
% se estima la magnitud de la DFT espacial de la onda de llegada.
[ARRAY,w]=freqz(array(1,:),1,512);
ARRAY=abs(ARRAY).^2;
% estimación del angulo de llegada DOA
% primero se calcula donde están los picos de la magnitud
ind=find(w<maxang);
tend=sign(ARRAY(ind(2:end))-ARRAY(ind(1:end-1)));
picos=find(eq(-diff(tend),2))+1;
% Y a partir de ellos el angulo de llegada en grados
DOA=asin(w(picos)*c/dsensor/2/pi/fo)*180/pi
```



Si la onda proveniente de θ=10.62 (ω=0.117π) se considera la onda de interés y la onda proveniente de θ=45.47 (ω=0.453π) se considera la onda interferente, para eliminar esta última hay que diseñar un filtro que deje pasar la pulsación discreta 0.117π y elimine la pulsación discreta 0.453π.

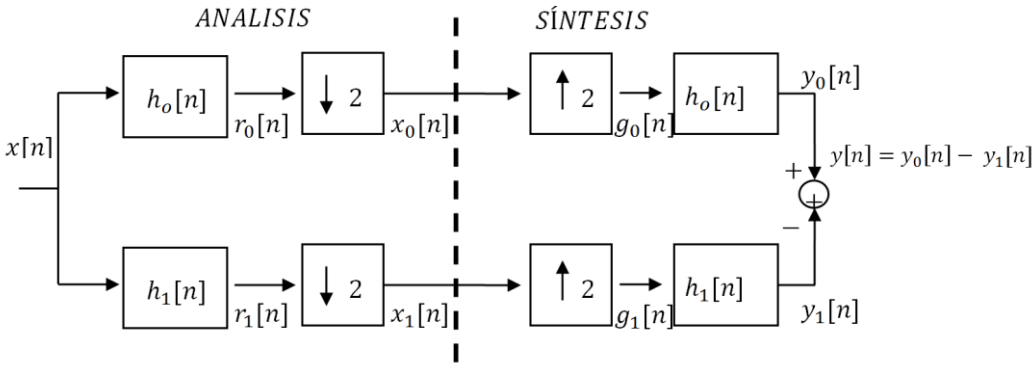
```
% diseño del filtro
M=16;
b=fir1(M-1,0.25);
% se filtra la señal espacial
salida30grados=zeros(size(array,1),1);
for i=1:size(array,1);
    salida30grados(i)=sum(array(i,:).*b);
end
```

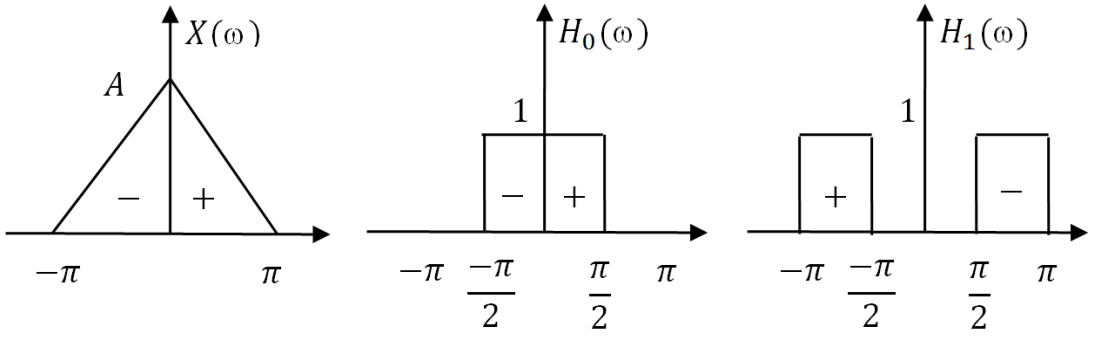


5.4 Problemas de Aula

-Problema 5.1

SISTEMA MULTIRATE:





a) Dibujar $X_0(\omega)$, $G_0(\omega)$y $Y_0(\omega)$

b) Calcular la expresión de $G_0(\omega)$ en función de $X(\omega)$ y $H_0(\omega)$.

c) Dibujar $X_1(\omega)$, $G_1(\omega)$y $Y_1(\omega)$

- Problema 5.2

Sea $x_c(t)$ una señal analógica cuya transformada de Fourier es

$$X_c(\Omega) = \begin{cases} 1 & |\Omega_c| < 2\pi W \\ 0 & |\Omega_c| > 2\pi W \end{cases}$$

Se quiere obtener a partir de $x_c(t)$ dos secuencias $y_i[n]$ y $z_i[n]$ definidas a través de

$$y_i[n] = x_c(\frac{n}{2fs}) \qquad z_i[n] = x_c(\frac{n}{4fs})$$

Para ello se propone utilizar el sistema de la figura 1, donde $S_1(\omega)$ y $S_2(\omega)$ son dos sistemas lineales e invariantes en el tiempo y $fs=2W$.

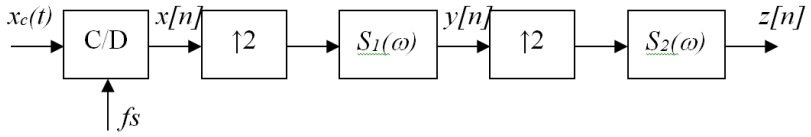


Figura 1: Sistema del problema 1.

a) Especifique las respuestas frecuenciales de los filtros $S_1(\omega)$ y $S_2(\omega)$ con mayores bandas de transición que permiten obtener las secuencias $y[n]=y_i[n]$ y $z[n]=z_i[n]$.

b) ¿Existe algún problema a la hora de implementar el sistema de la figura 1 con los filtros propuestos en el apartado anterior en un procesador discreto?. En caso afirmativo, comente cuáles son los problemas que encuentra.

c) Se quiere elegir $S_1(\omega)$ y $S_2(\omega)$ de manera que el sistema de la figura 1 proporcione dos secuencias que sean buenas aproximaciones de $y_i[n]$ y $z_i[n]$ ($y[n] \approx y_i[n]$ y $z[n] \approx z_i[n]$) con el menor número de operaciones/muestra posible. Para ello se propone utilizar dos filtros cuyas funciones de transferencia son

$$H_1(z) = \frac{1}{32}(-5z^3 + 20z^1 + 32 + 20z^{-1} - 5z^{-3})$$

y

$$H_2(z) = \frac{1}{2}(z^1 + 2 + z^{-1})$$

y cuyas respuestas frecuenciales $H_1(\omega)$ y $H_2(\omega)$ respectivas se muestran en la Figura 2.

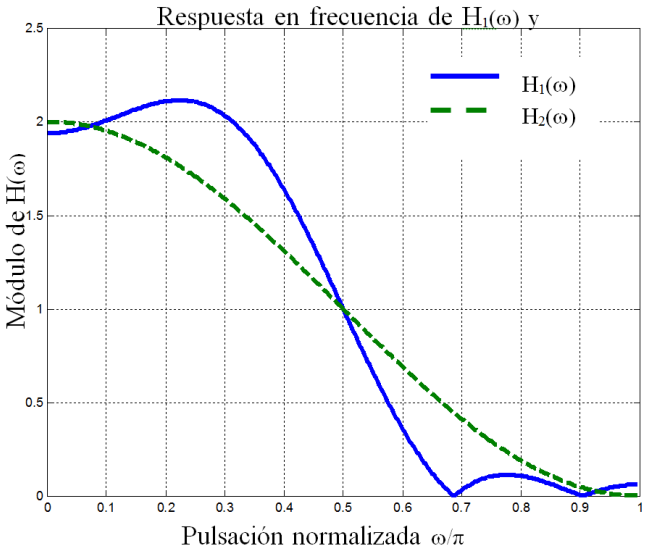


Figura 2: Respuestas frecuenciales de los filtros $H_1(z)$ y $H_2(z)$

A la hora de elegir los filtros $S_1(\omega)$ y $S_2(\omega)$ se consideran dos opciones:

- ✓ Opción a: $S_1(\omega) = S_2(\omega) = H_1(\omega)$
- ✓ Opción a: $S_1(\omega) = S_2(\omega) = H_2(\omega)$

¿Qué ventajas e inconvenientes presenta una opción respecto a la otra?

d) En el estándar de compresión de vídeo H.26L se utiliza un esquema como el de la figura 1 con $S_1(\omega)=H_1(\omega)$ y $S_2(\omega)=H_2(\omega)$. Comente qué sentido tiene utilizar dos filtros diferentes y por qué se utilizan con este orden.

e) Si la frecuencia de muestreo del sistema de la figura 1 fuese fs=20W, ¿qué filtros de entre $H_1(z)$ y $H_2(z)$ elegiría para $S_1(\omega)$ y $S_2(\omega)$?

-Problema 5.3

Este problema, trata sobre los convertidores C/D denominados Sigma-Delta (ΣΔ). La ventaja de este tipo de convertidores es que permite reducir muchísimo la complejidad del filtro antialiasing, que en la mayoría de los casos queda limitado a una simple red RC. Este problema trata de justificar por qué.

Este tipo de convertidores puede modelarse como se muestra en la figura 1, donde x100 representa un multiplicador de frecuencia de reloj, es decir a su salida presenta una frecuencia de reloj 100 veces superior a la de su entrada.

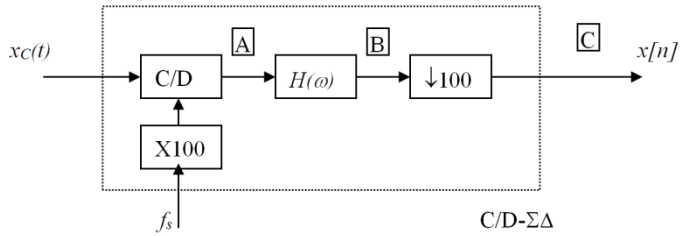


Figura 1: Modelo de convertidor C/D-ΣΔ

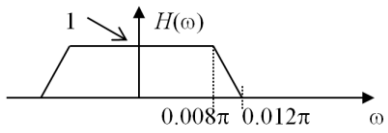


Figura 2: Respuesta en frecuencia del filtro de convertidor C/D-ΣΔ

Suponga que la señal (real) que se desea muestrear, $x_C(t)$, tiene una transformada de Fourier en la que se distinguen dos bandas

- De *0* a W_1 hercios: banda de interés
- De W_1 a W_2 hercios: banda sin interés, pero con contenido espectral no nulo.

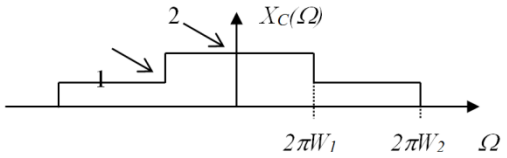


Figura 3: Transformada de Fourier de $x_C(t)$

Para centrar las ideas suponga que $x_C(t)$ es una señal de audio con $W_1=20$ *khz* y que el margen W_1 a W_2 corresponde a ruido indeseado.

a) Para el caso $W_2=W_1$, y $f_s=40$ *khz* dibuje los espectros en A , B y C . ¿Existe aliasing en algún punto del sistema?

b) Suponga que las muestras obtenidas del convertidor C/D-ΣΔ, se hacen pasar a través de un convertidor D/C ideal, tal como se muestra en la figura 4. Dibuje el espectro de $y_c(t)$. ¿Coincide con la zona de interés ($0\leq\Omega\leq 2\pi W_1$) de $x_C(t)$? Si no es así indique cuáles son las diferencias.

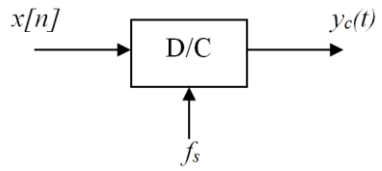


Figura 4: Conversión a analógico de las muestras obtenidas del convertidor C/D-ΣΔ

c) Suponga ahora que W_2=*400 khz*, W_1 =*20khz* y f_s =*40 khz*, dibuje los espectros en A, B y C. ¿Existe aliasing en C? ¿En qué margen de frecuencias? ¿Es posible eliminar dicho aliasing mediante algún sistema discreto? Si la respuesta es sí indique cómo y si la respuesta es no indique por qué no.

d) Para entender mejor la ventaja de usar un C/D-*ΣΔ*, frente a un C/D convencional ideal, dibuje el espectro de salida *s[n]* cuando la señal del apartado anterior se muestrea con f_s=*40 khz* (figura 5). ¿Existe aliasing? ¿En qué margen de frecuencias? ¿Es posible eliminarlo con algún sistema discreto? Si la respuesta es sí indique cómo y si la respuesta es no indique por qué no.

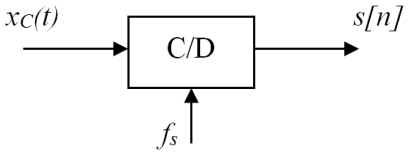
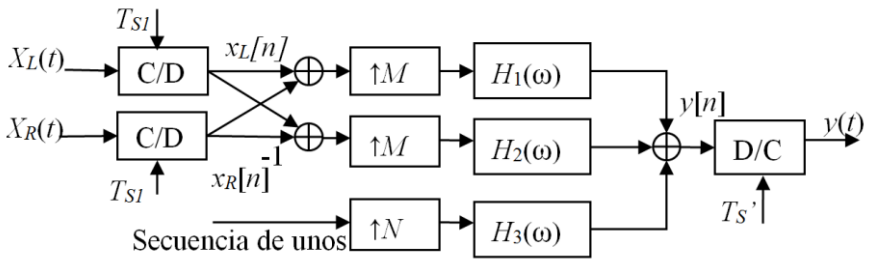


Figura 5: Conversión utilizando un convertidor C/D ideal convencional.

e) Para el caso del conversor C/D-*ΣΔ*, determine cuanto puede valer W_2 como máximo para que el margen libre de aliasing sea $\omega<0.8\pi$.

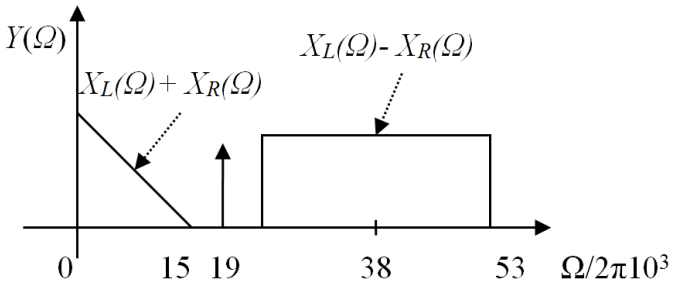
- Problema 5.4

Este problema trata sobre la implementación de un codificador FM-estéreo utilizando procesado discreto. Para ello se implementa el esquema de la figura. Suponga que las señales de audio a multiplexar $X_L(t)$ y $X_R(t)$ son de 15 kHz. de ancho de banda.



a) Determinar la relación entre T_{S1} y T_S' para que el sistema pueda funcionar en tiempo real.

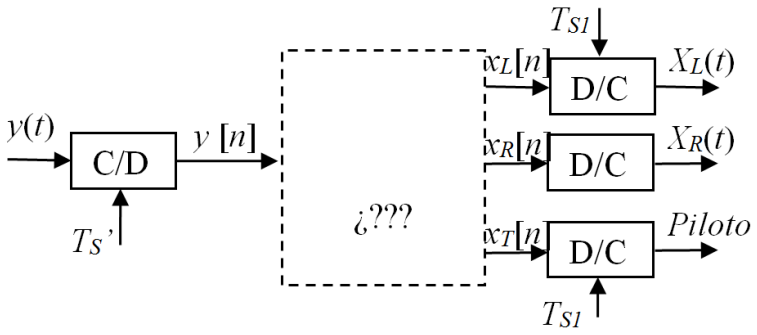
b) Determinar los factores M y N así como las respuestas en frecuencia de los filtros $H_1(\omega)$, $H_2(\omega)$ y $H_3(\omega)$, y los periodos de muestreo T_{S1} y T_S' de modo que la señal de salida $y(t)$ sea el múltiplex FM cuyo espectro se muestra en la figura. Dibuje los espectros de las señales en todos los puntos de las tres ramas. Para la respuesta en frecuencia de los filtros, considere filtros ideales.



c) Realice un programa en matlab, que a partir de las secuencias $x_R[n]$ y $x_L[n]$ obtenga la secuencia $y[n]$. Para ello diseñe los filtros que sean necesarios.

d) Indique justificadamente el diagrama de bloques de un decodificador del múltiplex totalmente discreto (ver siguiente figura). Dicho decodificador deber tener una entrada y tres salidas, una para cada canal de audio y una tercera para encender el piloto de estéreo. Debe especifique la respuesta en frecuencia de los filtros que necesite emplear, suponiendo que son filtros ideales

(AYUDA: Para recuperar las distintas señales, básicamente hay que realizar lo siguiente: 1) Separar mediante filtros cada una de las 3 señales que integran el múltiplex, 2) Obtener un tono discreto que se corresponda con la portadora para demodular la señal L-R, y 3) demodular dicha señal)



e) Realice un programa en matlab, que a partir de la secuencia $y[n]$, obtenga las secuencias $x_R[n]$, $x_L[n]$ y $x_T[n]$.

- Problema 5.5:

Considere el diagrama de la figura 1. Considere que f_{s1} = 10Khz. H_1 es un filtro paso-bajo ideal de frecuencia de corte 1/4 (pulsación de corte π/2) y ganancia en la banda de paso G = 2, y el filtro H_2 tiene la respuesta impulsional que se muestra en la figura 2.

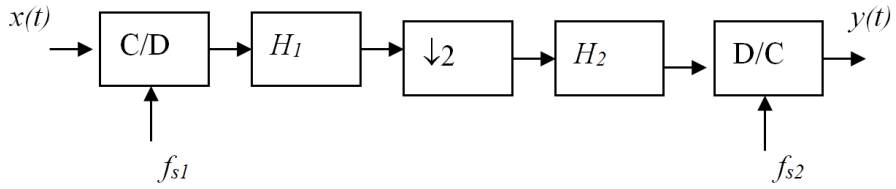


Figura 1: Diagrama del sistema.

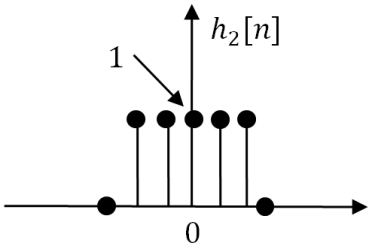


Figura 2: Respuesta impulsional de H2

Se pide:

a) Determinar la relación entre f_{s1} y f_{s2} para que el sistema pueda funcionar en tiempo real (no salgan más muestras de las que entren).

b) Ancho de banda máximo W de la señal de entrada $x(t)$ para que la señal de salida no tenga aliasing.

c) Suponiendo que no existe aliasing, y que la transformada de Fourier de *x(t)* e *y(t)* son *X(Ω)* e *Y(Ω)*, determinar $H_{eff}(\Omega) = \frac{Y(\Omega)}{X(\Omega)}$

d) Calcular la salida y(t) cuando x(t) = cosΩ_0t con $\Omega_0 = 2\pi 9000$

- Problema 5.6:

Sea $x_c(t)$ la temperatura en grados centígrados en un determinado punto del espacio a lo largo del tiempo. Al muestrear $x_c(t)$ a las horas en punto, por ejemplo a las 0.00, 1.00, 2.00 ..., se obtiene la secuencia *x[n]*.

a) Si $x_c(t)$ es una señal pasobajo limitada en banda ($X(\omega) = 0$; $|\omega| > 2\pi W$), ¿Cuál es el máximo ancho de banda *W* de $x_c(t)$ para que $x_c(t)$ se pueda recuperar a partir de *x[n]*?

Suponga a partir de ahora que la señal tiene el ancho de banda *W* del apartado anterior.

b) Dibuje el diagrama de bloques de un sistema que permita obtener una secuencia $x_1[n]$ en la que cada muestra represente la temperatura cada cuarto de hora (por ejemplo a las 0.00, 0.15, 0.30, 0.45, 1.00, 1.15, 1.30, ...). Especifique la tarea de cada bloque.

c) Proponga el diagrama de bloques de un sistema que permita obtener una secuencia $x_2[n]$ en la que cada muestra represente la temperatura a las horas en punto y diez minutos (por ejemplo a las 0.10, 1.10, 2.10, 3.10, 4.10, 5.10, ...).

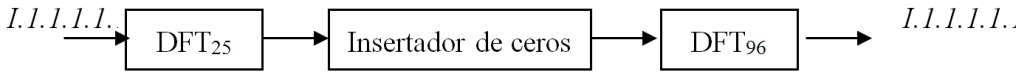
d) Suponga ahora que se han obtenido sólo 24 muestras de temperatura en las horas en punto de un determinado día. Estos valores se muestran en la siguiente tabla.

Hora	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Temperatura	18	17	16	14	11	11	12	15	17	19	20	22	23	24	25	26	26	25	24	23	22	21	20	19

Como no se tienen datos del resto de instantes de tiempo, se supone que los valores de temperatura se repiten cada 24 horas, lo cual se aproxima bastante a la realidad. Calcule la temperatura correspondiente a la 8.45 y a las 23.45 del mismo día en el que se han tomado las temperaturas, utilizando un interpolador lineal que en este caso es:

n	-4	-3	-2	-1	0	1	2	3	4
h(n)	0	0.25	0.5	0.75	1	0.75	0.5	0.25	0

e) Otra solución para interpolar la señal y obtener la temperatura en otros instantes de tiempo es realizar la operación del siguiente diagrama de bloques:



en el que:

- ✓ el primer bloque realiza la DFT de 24 puntos de la secuencia temperatura
- ✓ *el* insertador de ceros *inserta ceros en* $X_{24}[k]$ *dando lugar a la secuencia* $X_{96}[k]$

5.5 Ejercicios Prácticos

Efectos del aliasing sobre la voz

Se ha muestreado una frase de voz con las frecuencias de muestreo y filtros *antialiasing* especificados en la tabla. Rellenar las cuadrículas especificando en que operaciones de muestreo hay *aliasing* (escribir SI o NO).

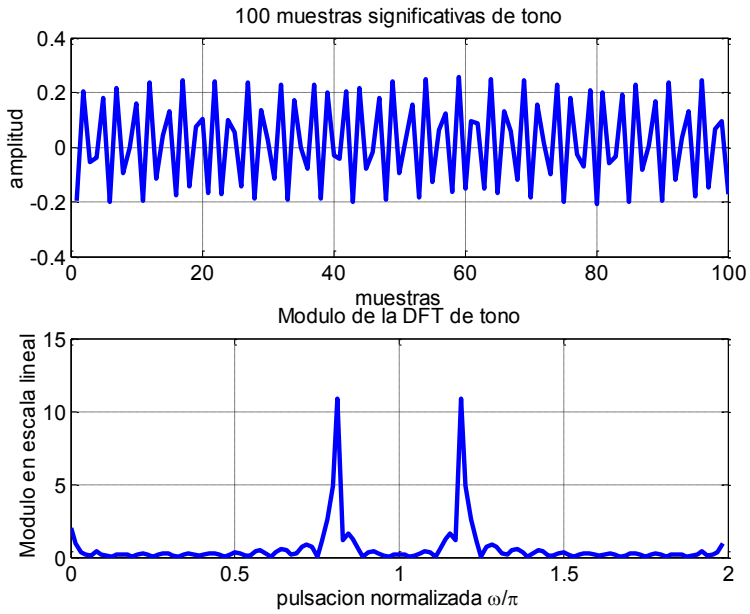
En el caso 10 se tienen diez segundos de voz convertidos de discreto a continuo con las condiciones especificadas en la tabla cambiando, durante la reproducción, la frecuencia de corte del filtro reconstructor de 8000 a 4000Hz y de nuevo a 8000 Hz: comentar el efecto obtenido.

Grabación	Frecuencia de muestreo en muestras/seg	Frecuencia de corte del filtro *antialiasing* en *Hz*	Frecuencia de corte del filtro Reconstructor en *Hz*	¿Hay solapamiento espectral?
1	Grabación original			
2	10000	2800	2800	
3	8000		2800	
4	6000		2800	
5	3000		2800	
6	3000	3800	3800	
7	6000		3800	
8	8000		3800	
9	10000		3800	
10	8000	3800	8000→4000→8000	

Cambio de la frecuencia de corte del filtro reconstructor

Genere un tono de 3200Hz por el altavoz de su tarjeta de sonido y grabe 10000 muestras de dicho sonido a 8000 muestras por segundo con su tarjeta de sonido. Denomine a la señal grabada *tono*. Guarde dicha secuencia en el fichero *tono.mat*.

Representar en pantalla 100 muestras significativas de *tono*. A partir de las anteriores 100 muestras, calcule y represente el módulo de su DFT de 128 muestras.



Si convierto *tono* de discreto a continuo con frecuencia de reconstrucción 10000 muestras/seg.:

1. ¿Cuál debe ser la frecuencia de corte del filtro reconstructor?. ¿De qué frecuencia es el tono continuo resultante?. ¿Cuánto tiempo durará?

$f_{corte}=$ $f_{tono}=$ $Duración\_tono=$

2. ¿Entre que valores debe estar la frecuencia de corte del filtro reconstructor para que se oigan cuatro tonos?, ¿Cuáles son las frecuencias de los tonos?.

$<f_{corte}<$

$f_{tono1}=$ $f_{tono2}=$ $f_{tono3}=$ $f_{tono4}=$

3. ¿Cuál ha de ser la frecuencia de reconstrucción y la frecuencia de corte del filtro reconstructor para oír un tono de 2000 Hz. a partir de la secuencia discreta *tono*?

$f_{muestreo}=$ $f_{corte}=$

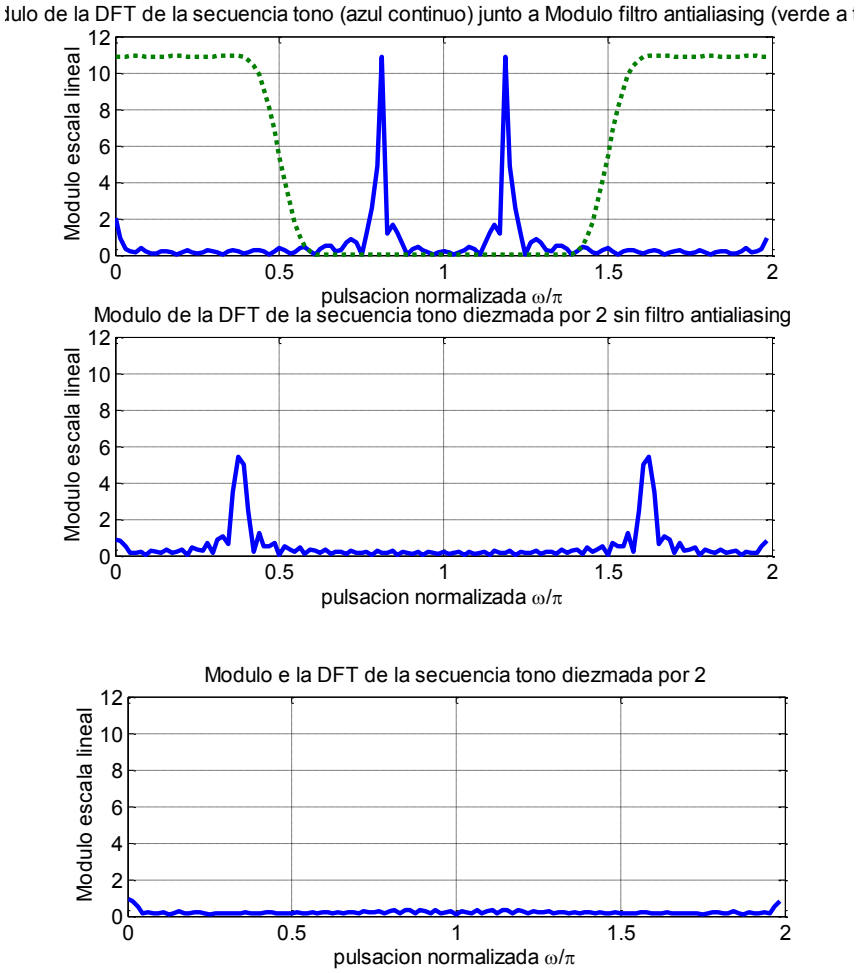
¿Cuál será la duración del tono obtenido en este apartado?

$Duración\_tono=$

Con el comando *soundsc* escuche la secuencia *tono* a diferentes frecuencias de reproducción.

Diezmado

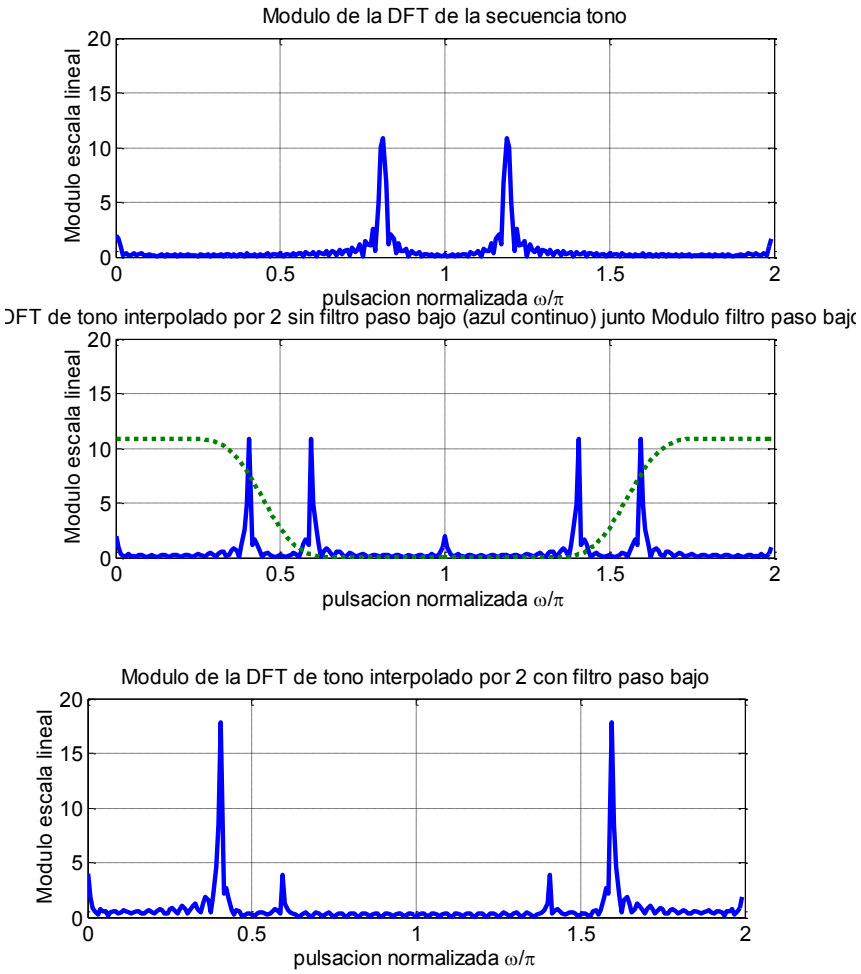
A partir de las 100 muestras seleccionadas de *tono* en el apartado 2.A, cree la señal *tonod1* obtenida al diezmar la secuencia *tono* por dos sin utilizar el clásico filtro *antialiasing*. Repita el proceso anterior de diezmado por dos de las 100 muestras seleccionadas de la secuencia *tono* utilizando el filtro *antialiasing* (puede ser útil utilizar la función *decimate* de Matlab) obteniendo la secuencia *tonod2*. Compare *tono*, *tonod1* y *tonod2* representando, en escala lineal y en gráficas separadas, el módulo de la DFT de128 muestras de *tono, tonod1* y *tonod2*. Dibuje, junto al módulo de la DFT de *tono,* el módulo del filtro antialiasing.



Interpolación

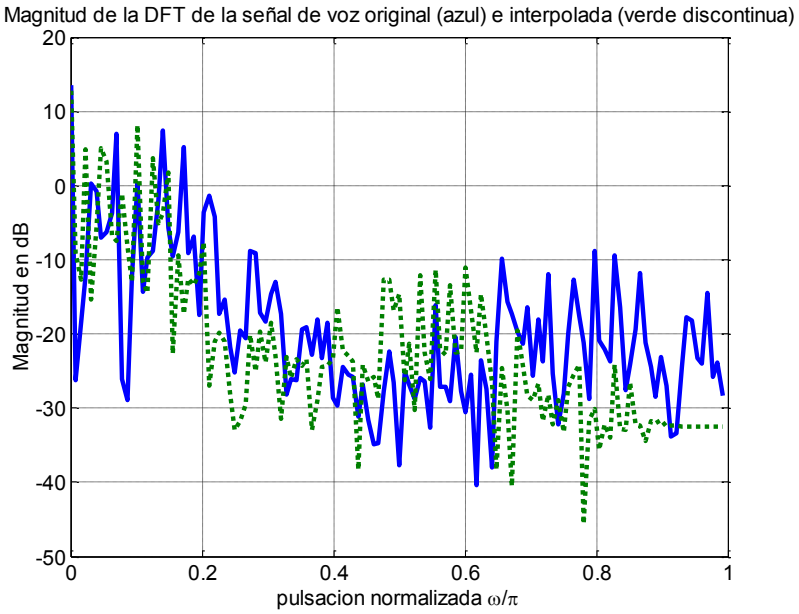
Partiendo de las 100 muestras de la anterior secuencia *tono*, genere la secuencia *tonoi1* interpolando la secuencia *tono* por un factor 2 sin el filtro paso bajo utilizado en interpolación, esto es, *tonoi1* será igual a las muestras de *tono* con un cero intercalado

entre muestra y muestra. Repita el proceso anterior de interpolación de la secuencia *tono* por 2 utilizando el filtro paso bajo (puede ser útil la función *interp* de Matlab) obteniendo la secuencia *tonoi2*. Compare *tono*, *tonoi1* y *tonoi2* representando en escala lineal 256 muestras del módulo de la DFT de *tono*, *tonoi1*, y *tonoi2*



Cambio frecuencia de muestreo por un factor racional

Grabe una vocal sostenida de voz de un segundo de duración con frecuencia de muestreo 8000 muestras/segundo. En caso necesario, recorte los silencio del principio y fin. Suponga que dicha secuencia se encuentra en la variable *voz*. Modificar de forma discreta la secuencia *voz* para que al reproducir la nueva secuencia obtenida *vozM* a 11000 muestras/seg suene exactamente igual que al reproducir *voz* a 8000 muestras/seg. Representar, solapadas en una sola gráfica, en dBs, la magnitud de la DFT de 256 muestras de *voz* y *vozM*. Comentar los resultados. Con el comando *soundsc* de Matlab reproducir, *voz* a 8000 y 11000 muestras/seg., y *vozM* a 8000 y 11000 muestras/seg.



Muestreo espacial de señales

En este apartado se trata de estimar las direcciones de llegada de dos sonido de banda estrecha emitido en el agua que llegan a un array de sensores. El array consta de 16 sensores colocados en línea recta a una distancia de 30 cm entre sí. Todos los sensores muestrean la señal de llegada al mismo tiempo a 8000 muestras por segundo. 50 de las muestras recogidas por cada sensor se han almacenado en las columnas de la matriz denominada *array* definida como:

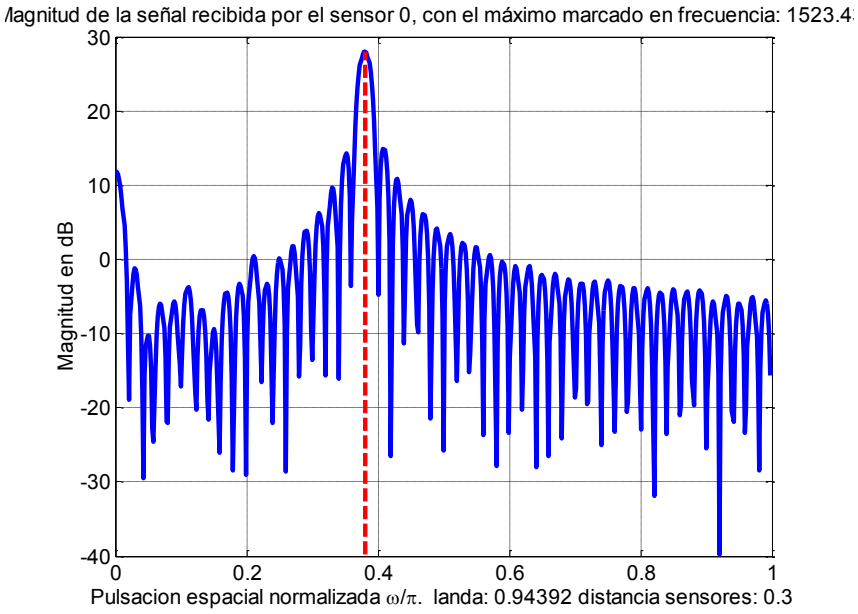
array=[

-0.4157 0.2052 0.3154 0.0219 0.2092 0.0443 -0.4528 -0.0726 0.4172 0.1157 0.0502 0.1898 -0.2826 -0.3329 0.3221 0.3271
0.0880 0.4895 0.0659 -0.0262 0.0823 -0.3092 -0.1939 0.4502 0.2878 -0.1184 0.0606 -0.1056 -0.3457 0.2235 0.5027 -0.0060
0.5314 0.2022 -0.2261 -0.0047 -0.0993 -0.2172 0.3475 0.4502 -0.1549 -0.1606 0.0247 -0.2181 0.0813 0.5585 0.1010 -0.2820
0.3412 -0.2968 -0.1828 0.0747 -0.1055 0.1839 0.5048 -0.0726 -0.3701 0.0377 0.0014 -0.0047 0.4663 0.2404 -0.3743 -0.1546
-0.2258 -0.3736 0.1353 0.1067 0.0583 0.4077 0.0652 -0.4634 -0.0705 0.2222 0.0308 0.2714 0.3139 -0.3242 -0.3362 0.2122
-0.4744 0.0746 0.3300 0.0439 0.1998 0.1629 -0.4088 -0.2313 0.3506 0.1802 0.0695 0.2573 -0.1765 -0.4398 0.1781 0.3547
-0.0556 0.4641 0.1499 -0.0257 0.1383 -0.2455 -0.3332 0.3386 0.3740 -0.0437 0.0749 -0.0249 -0.4031 0.0514 0.5076 0.0900
0.4584 0.3175 -0.1714 -0.0160 -0.0545 -0.2989 0.2121 0.5165 -0.0278 -0.1608 0.0391 -0.2319 -0.0687 0.5205 0.2331 -0.2346
0.4481 -0.1848 -0.2320 0.0636 -0.1327 0.0635 0.5291 0.0835 -0.3466 -0.0243 0.0012 -0.0919 0.3925 0.3672 -0.2769 -0.2074
-0.0845 -0.4092 0.0499 0.1043 0.0002 0.3951 0.2083 -0.4037 -0.1816 0.1934 0.0161 0.1988 0.3999 -0.1921 -0.3842 0.1384
-0.4644 -0.0710 0.3166 0.0667 0.1738 0.2548 -0.3235 -0.3438 0.2664 0.2195 0.0459 0.2886 -0.0456 -0.4529 0.0536 0.3677
-0.2144 0.4101 0.2320 -0.0238 0.1675 -0.1656 -0.4129 0.2074 0.4276 0.0072 0.0680 0.0642 -0.3770 -0.0832 0.4848 0.1878
0.3586 0.4205 -0.1053 -0.0369 -0.0218 -0.3381 0.0746 0.5422 0.0913 -0.1637 0.0601 -0.1889 -0.1708 0.4568 0.3611 -0.1711
0.5319 -0.0577 -0.2721 0.0236 -0.1450 -0.0350 0.5106 0.2385 -0.3056 -0.0710 0.0242 -0.1373 0.3147 0.4786 -0.1642 -0.2580
0.0696 -0.4240 -0.0588 0.0860 -0.0374 0.3531 0.3512 -0.3080 -0.2646 0.1632 0.0231 0.1488 0.4658 -0.0482 -0.4245 0.0298
-0.4304 -0.2236 0.2571 0.0863 0.1522 0.3473 -0.1911 -0.4144 0.1704 0.2508 0.0571 0.3127 0.0855 -0.4614 -0.1002 0.3416
-0.3672 0.2990 0.2845 0.0098 0.2051 -0.0361 -0.4385 0.0627 0.4464 0.0882 0.0815 0.1402 -0.3518 -0.2387 0.4065 0.2787
0.2106 0.4659 -0.0100 -0.0238 0.0589 -0.3177 -0.0734 0.5221 0.2215 -0.1235 0.0657 -0.1570 -0.2924 0.3377 0.4588 -0.0837
0.5399 0.0832 -0.2411 0.0317 -0.1031 -0.1366 0.4492 0.3791 -0.2172 -0.1171 0.0194 -0.2067 0.1917 0.5407 -0.0171 -0.2776
0.2200 -0.3526 -0.1096 0.1077 -0.0702 0.2781 0.4622 -0.1774 -0.3231 0.0927 0.0001 0.0625 0.4789 0.1194 -0.4061 -0.0645
-0.3224 -0.2909 0.2222 0.1142 0.1115 0.4058 -0.0493 -0.4495 0.0378 0.2397 0.0332 0.2963 0.2224 -0.3929 -0.2264 0.2845
-0.4106 0.2026 0.3408 0.0394 0.2163 0.0755 -0.4360 -0.0978 0.4077 0.1326 0.0732 0.2124 -0.2538 -0.3484 0.2974 0.3277
0.0843 0.5072 0.0942 -0.0250 0.1064 -0.2913 -0.2167 0.4394 0.3118 -0.0919 0.0733 -0.0743 -0.3518 0.1931 0.4967 0.0050
0.5392 0.2380 -0.2104 0.0048 -0.0861 -0.2301 0.3393 0.4701 -0.1309 -0.1448 0.0424 -0.2133 0.0558 0.5449 0.1182 -0.2795
0.3806 -0.2683 -0.1901 0.0788 -0.1079 0.1804 0.5170 -0.0438 -0.3478 0.0386 0.0174 -0.0184 0.4457 0.2546 -0.3670 -0.1542
-0.1939 -0.3792 0.1265 0.1114 0.0652 0.4172 0.0948 -0.4447 -0.0757 0.2360 0.0290 0.2510 0.3205 -0.3110 -0.3364 0.2161

```
-0.4651  0.0465  0.3371  0.0636  0.2090  0.1829 -0.3929 -0.2306  0.3502  0.1925  0.0603  0.2543 -0.1641 -0.4360  0.1746  0.3609
-0.0948  0.4706  0.1786 -0.0125  0.1502 -0.2271 -0.3297  0.3278  0.3954 -0.0388  0.0636 -0.0196 -0.3915  0.0420  0.5100  0.1089
 0.4582  0.3492 -0.1485 -0.0082 -0.0428 -0.2997  0.2041  0.5309 -0.0088 -0.1711  0.0361 -0.2162 -0.0798  0.5205  0.2640 -0.2289
 0.4777 -0.1527 -0.2261  0.0631 -0.1311  0.0648  0.5307  0.1171 -0.3476 -0.0355  0.0114 -0.0984  0.3916  0.3977 -0.2650 -0.2327
-0.0479 -0.3994  0.0420  0.1085  0.0023  0.3934  0.2441 -0.3949 -0.1960  0.1927  0.0176  0.1998  0.4258 -0.1709 -0.4052  0.1017
-0.4471 -0.0828  0.3130  0.0680  0.1818  0.2770 -0.3029 -0.3506  0.2539  0.2239  0.0535  0.3095 -0.0249 -0.4704  0.0056  0.3504
-0.2260  0.3988  0.2398 -0.0067  0.1762 -0.1349 -0.4082  0.1843  0.4277  0.0261  0.0837  0.0760 -0.3854 -0.1281  0.4552  0.1970
 0.3448  0.4282 -0.0888 -0.0299  0.0039 -0.3233  0.0500  0.5365  0.1198 -0.1510  0.0631 -0.1902 -0.2104  0.4140  0.3656 -0.1575
 0.5319 -0.0378 -0.2577  0.0354 -0.1190 -0.0503  0.4992  0.2674 -0.2914 -0.0759  0.0183 -0.1690  0.2723  0.4751 -0.1399 -0.2664
 0.0948 -0.4089 -0.0542  0.1137 -0.0412  0.3403  0.3735 -0.2894 -0.2752  0.1482  0.0052  0.1163  0.4483 -0.0265 -0.4239  0.0132
-0.4191 -0.2133  0.2707  0.0962  0.1488  0.3601 -0.1732 -0.4265  0.1478  0.2398  0.0333  0.2862  0.0989 -0.4487 -0.1222  0.3386
-0.3489  0.2964  0.3080  0.0217  0.2077 -0.0268 -0.4469  0.0338  0.4326  0.0778  0.0607  0.1400 -0.3374 -0.2561  0.3963  0.2914
 0.2035  0.4896  0.0164 -0.0252  0.0562 -0.3222 -0.1047  0.5049  0.2255 -0.1355  0.0477 -0.1509 -0.2958  0.3208  0.4682 -0.0666
 0.5529  0.1253 -0.2369  0.0165 -0.1113 -0.1645  0.4298  0.3859 -0.2209 -0.1419  0.0144 -0.1984  0.1750  0.5448  0.0062 -0.2858
 0.2701 -0.3407 -0.1347  0.0892 -0.0895  0.2579  0.4653 -0.1675 -0.3432  0.0710  0.0076  0.0564  0.4758  0.1405 -0.4066 -0.0928
-0.3026 -0.3157  0.1922  0.1010  0.0943  0.4058 -0.0316 -0.4658  0.0018  0.2401  0.0434  0.2892  0.2371 -0.3876 -0.2577  0.2629
-0.4272  0.1615  0.3255  0.0307  0.2140  0.0888 -0.4447 -0.1326  0.3975  0.1505  0.0709  0.2173 -0.2501 -0.3727  0.2622  0.3265
 0.0386  0.4857  0.0956 -0.0277  0.1123 -0.2892 -0.2443  0.4137  0.3257 -0.0799  0.0669 -0.0774 -0.3661  0.1499  0.4927  0.0284
 0.5111  0.2430 -0.2092  0.0040 -0.0829 -0.2507  0.3071  0.4766 -0.1051 -0.1573  0.0306 -0.2231  0.0161  0.5339  0.1453 -0.2610
 0.3811 -0.2591 -0.1969  0.0752 -0.1156  0.1552  0.5106 -0.0175 -0.3564  0.0164  0.0043 -0.0453  0.4283  0.2802 -0.3340 -0.1697
-0.1788 -0.3886  0.1167  0.1142  0.0478  0.3977  0.1129 -0.4421 -0.1064  0.2157  0.0154  0.2342  0.3387 -0.2730 -0.3411  0.1853
-0.4718  0.0318  0.3390  0.0561  0.1899  0.1882 -0.3851 -0.2616  0.3211  0.1863  0.0543  0.2625 -0.1281 -0.4279  0.1346  0.3606
-0.1121  0.4676  0.1858 -0.0229  0.1361 -0.2258 -0.3505  0.2893  0.3901 -0.0301  0.0651  0.0080 -0.3772  0.0098  0.4946  0.1362
 0.4515  0.3628 -0.1521 -0.0362 -0.0520 -0.3087  0.1620  0.5242  0.0105 -0.1670  0.0494 -0.2033 -0.0945  0.4921  0.2864 -0.2071
];
```

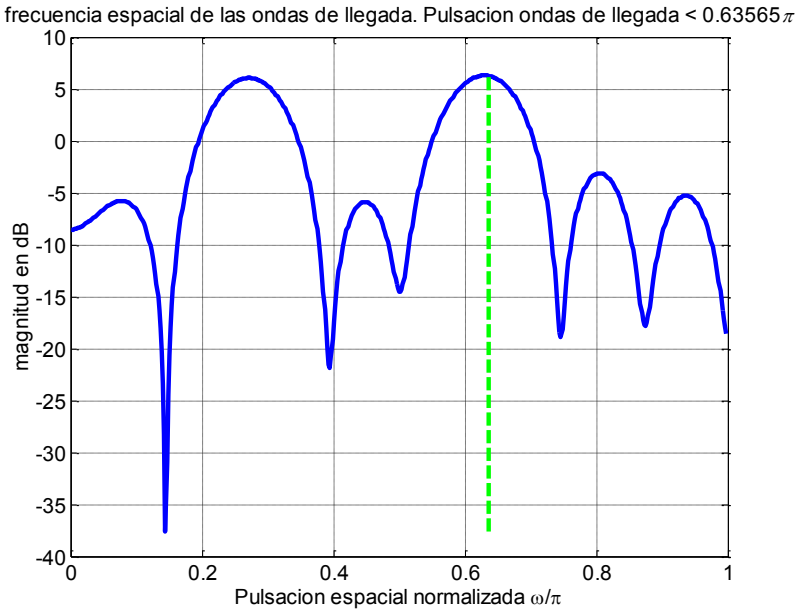
La primera columna corresponde a la señal del sensor 0, la segunda columna a la señal del sensor 1, así sucesivamente hasta el último sensor. Suponiendo que la velocidad del sonido en el agua es 1438 metros / segundo se pide:

Representar la magnitud en dBs de la transformada de Fourier de la señal recogida por el primer sensor. A partir de dicha representación calcular la frecuencia del sonido incidente, calcule su landa y compruebe que la distancia entre sensores es apropiada para calcular la dirección de llegada de dicho sonido.

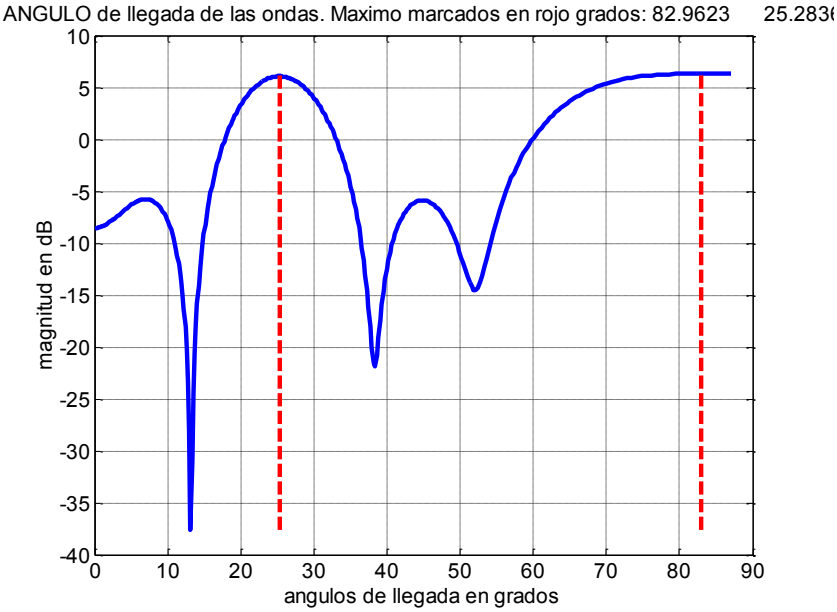


Represente una gráfica con la magnitud en dB de la transformada de Fourier de la onda espacial para, a partir del máximo de dicha magnitud, poder calcular la

dirección de llegada. ¿Entre qué pulsaciones debe estar el pico para que no haya aliasing? ¿Cuántas ondas componen la señal de llegada al array?, y ¿cuáles son los ángulos de llegada de cada onda plana al array?



Dibuje de nuevo la gráfica anterior representando en el eje x las direcciones de llegada en grados que corresponden a cada pulsación. Marque la dirección de llegada del sonido recogido por el array en grados



Capítulo VI. Estimación Espectral

En este capítulo dejamos atrás el procesado de señal y proseguimos con lo que se denomina análisis espectral de la señal. El objetivo del presente capítulo es que el alumno aprenda a calcular los diferentes estimadores espectrales y compruebe sus propiedades.

6.1 Introducción

El análisis de señal trata de extraer la información contenida en la señal sin preocuparse por preservar su forma de onda. Por ejemplo, cuando se recibe una señal de voz, el objetivo del análisis es extraer su información: qué dice el locutor, quién es el locutor, en qué idioma habla, etc. sin preocuparnos que el proceso sea reversible o en preservar la forma de onda de la señal recibida. Evidentemente tenemos que tratar el tema de la variabilidad, esto es, dos locutores nunca pronuncian igual un mismo fonema, por ejemplo la /a/, pero la información es común a todas las formas posibles de pronunciarla. Por lo tanto la misma información nos llega de formas muy distintas y hay que ser capaces de extraer la información de forma independiente a la forma de onda concreta de la señal que nos llega.

Ya que la forma de onda en sí no es una buena característica de la información, ésta suele buscarse en el espectro que es más robusto o estable, esto es, el espectro de diferentes señales que transportan una misma información cambia menos que la forma de onda de las señales. La razón es que el espectro en vez de fijarse en la forma de onda caracteriza la relación que existe entre las diferentes muestras, esto es, la relación entre $x[n]$ y $x[n-1]$, entre $x[n]$ y $x[n-2]$, entre $x[n]$ y $x[n-3]$,…, hasta $x[n]$ y $x[n-M]$. Dicha relación se puede caracterizar de dos formas, mediante la autocorrelación dando lugar a los métodos espectrales no paramétricos o clásicos, o mediante una función dando lugar a los métodos espectrales paramétricos.

En el caso de la estimación espectral no paramétrica, el espectro $S_x(\omega)$ de la señal $x[n]$ se define como la transformada de Fourier de la autocorrelación

$$S_x(\omega) = \sum_{m=-\infty}^{+\infty} R_x(m) e^{-j\omega m}$$

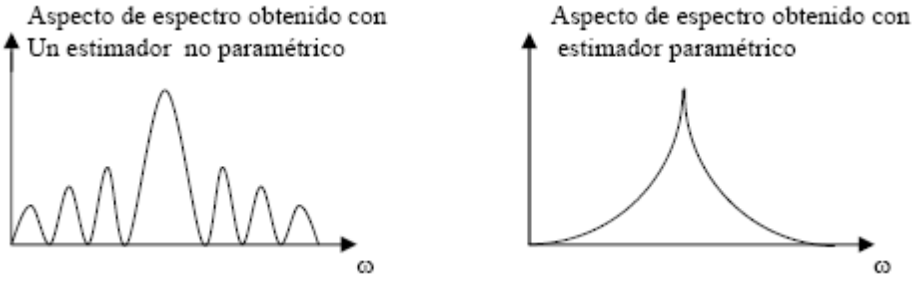
En cambio, en los métodos paramétricos se supone que el espectro puede modelarse mediante una función a estimar, por ejemplo:

$$x[n] = -\sum_{k=1}^{P} a_k x[n-k]$$

Así, el modelado espectral paramétrico será un proceso de 3 pasos:

1. Seleccionar el tipo de función adecuada que relacione las muestras entre sí. Esta función tendrá unos parámetros a estimar que den la relación concreta entre muestras de $x[n]$.
2. Se estiman los parámetros de la función que modela la relación entre muestras, en el ejemplo los a_k .
3. A partir de la función que relaciona las muestras de $x[n]$ se obtiene el espectro del proceso estocástico.

Como ventajas de este proceso mencionar que al incorporar al algoritmo de estimación del espectro algún conocimiento sobre el proceso estocástico, esto es la fórmula de relación entre muestras, la estimación tendrá mayor resolución y fidelidad sin tener la apariencia de *sinc* propia de los estimadores no paramétricos.



Como desventaja comentar que si la relación entre muestras del proceso estocástico $x[n]$ no es descrita correctamente por el tipo de función elegida, la estimación espectral no será correcta. Ésta no-correspondencia puede ser causada por la aparición de un ruido aditivo a la señal. En otras palabras, son menos robustos que los no paramétricos.

6.2 Estimación Espectral no paramétrica: Periodograma y Welch

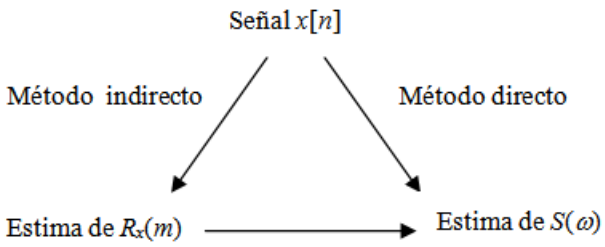
En el caso de la estimación espectral no paramétrica, el espectro $S_x(\omega)$ de la señal $x[n]$ se define como la transformada de Fourier de la autocorrelación:

$$S_x(\omega) = \sum_{m=-\infty}^{+\infty} R_x(m) e^{-j\omega m} \qquad \text{[Ec.69]}$$

Existen dos métodos de estimación espectral no paramétrica: los métodos directos y los métodos indirectos. Los métodos indirectos estiman en primer lugar la autocorrelación $R_x(m)$ a partir de $x[n]$ para obtener la estima del espectro $S_{indirecto}(\omega)$ mediante la transformada de Fourier de dicha autocorrelación.

Los métodos directos estiman el espectro $S_{directo}(\omega)$ de $x[n]$ como la magnitud (módulo al cuadrado de su transformada de Fourier) de $x[n]$. Esta estimación es una aplicación de la relación de Wiener-Khintchine:

$$S_x(\omega) = \sum_{m=-\infty}^{+\infty} R_x(m) e^{-j\omega m} = \left| \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n} \right|^2 \qquad \text{[Ec.70]}$$



En éste capítulo veremos dos ejemplos de los directos, el periodograma propuesto es 1898 y el método de Welch propuesto en 1967.

El periodograma estima el espectro como

$$S_{PER}(\omega) = \frac{1}{N} \cdot |X(\omega)|^2 = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \right|^2 \qquad \text{[Ec.71]}$$

El problema de este estimador es que la señal $x[n]$ está limitada en el tiempo por una ventana rectangular (multiplicada) lo cual conlleva que el periodograma resultante sea la convolución del espectro con la transformada de Fourier de una ventana triangular de longitud N-1 centrada en el origen. Lo cual limita la precisión en frecuencia del periodograma.

Una forma de aumentar la precisión espectral es modificando la ventana de tal manera que la nueva ventana se aproxime más a una delta. Así se propone un nuevo estimador: el periodograma modificado, definido como:

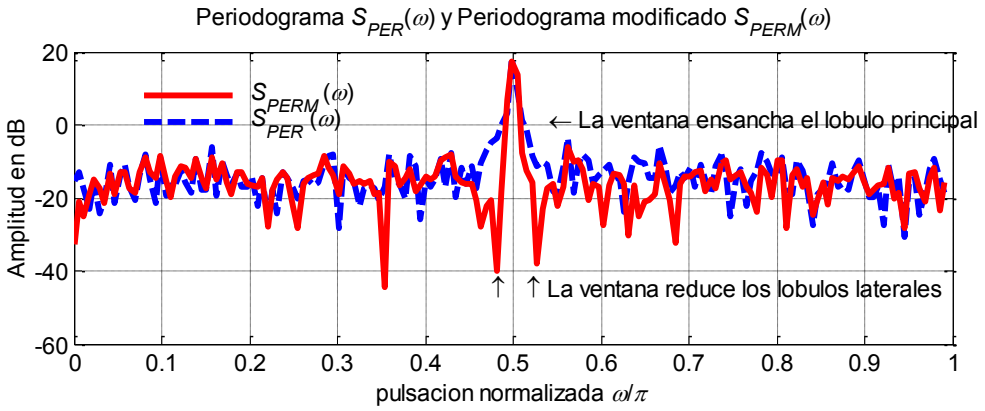
$$S_{PERM}(\omega) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]w[n]e^{-j\omega n}\right|^2 \qquad \text{[Ec.72]}$$

donde $w[n]$ es la ventana que, para no modificar la energía del periodograma, debe cumplir:

$$\frac{1}{N}\sum_{n=0}^{N-1} w^2[n] = 1 \qquad \text{[Ec.73]}$$

La ventana $w[n]$ aumenta la anchura de su lóbulo principal y reduciendo la altura de sus lóbulos laterales, tal y como puede observase en el siguiente ejemplo

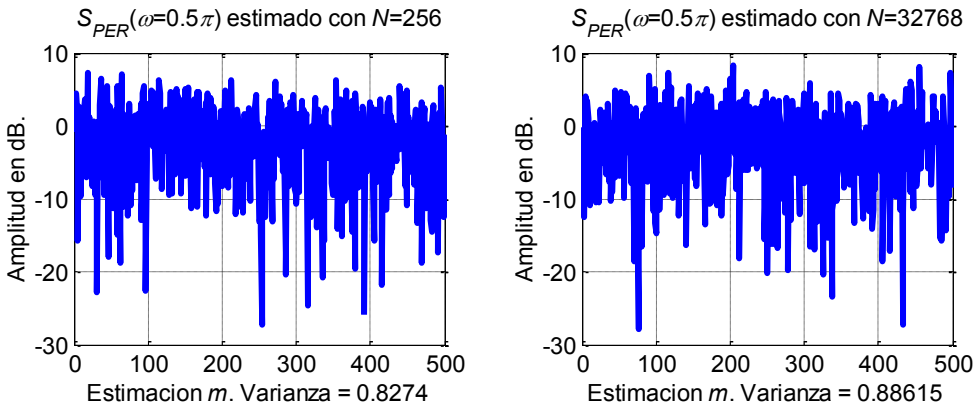
```
N=345;
x=sin(pi/2*[0:N-1]');
x=x+randn(N,1)*0.2;
Sper=(abs(fft(x)).^2)/N;
Sper=Sper(1:N/2);
w=hamming(N);
w=w.*sqrt(N/sum(w.^2));
Sperm=(abs(fft(x.*w)).^2)/N;
Sperm=Sperm(1:N/2);
```



Otro problema del periodograma es su variabilidad. Por ejemplo, el periodograma modificado de ruido blanco dista mucho de ser una recta debido a la gran variabilidad del periodograma. Y como puede verse en el siguiente ejemplo, dicha variabilidad no disminuye con la longitud temporal de la ventana.

```
% Se estima el periodograma M=500 veces de N=256 muestras
% de ruido blanco gaussiano de media cero y varianza unidad
M=500;N=256;
Sper1medios=zeros(M,1);
```

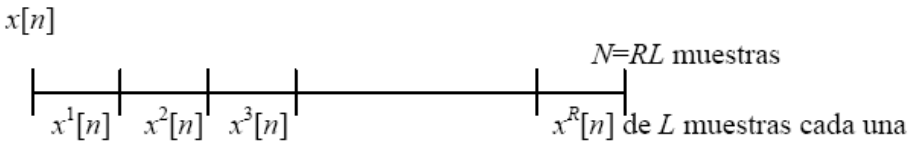
```
for m=1:M
    x=randn(N,1);
    w=hamming(N);
    w=w.*sqrt(N/sum(w.^2));
    Sper=(abs(fft(x.*w)).^2)./N;
    %me quedo con la estima de la frecuencia w=0.5pi
    Sper1medios(m)=Sper(N/4+1);
end
% Se estima el periodograma M=500 veces de N=32768 muestras
M=500;N=32768;
Sper2medios=zeros(M,1);
for m=1:M
    x=randn(N,1);
    w=hamming(N);
    w=w.*sqrt(N/sum(w.^2));
    Sper=(abs(fft(x.*w)).^2)./N;
    Sper2medios(m)=Sper(N/4+1);
end
```



En un intento de reducir más dicha variabilidad del periodograma modificado se recurre al principio bien conocido de que al promediar un número R de variables aleatoria independientes con función densidad de probabilidad gaussiana y varianza σ^2 la varianza resultante de la variable aleatoria promedia es σ^2/R.

Así en 1967 se propuso el estimador espectral de Welch, cuyo procedimiento es el siguiente:

1. Se descompone la realización $x[n]$ de N muestras en R tramas de longitud L. A cada trama se le denomina $x^i[n]$ como en la siguiente figura



2. Se calcula el periodograma modificado de cada trama.

$$S^{i}_{PERM}(\omega) = \frac{1}{L} \cdot \left| \sum_{n=0}^{L-1} x^{i}[n] \cdot w[n] \cdot e^{-j\omega n} \right|^{2} \qquad i=1,2,3,.....,R \qquad \text{[Ec.74]}$$

3. Se promedian todos estos periodogramas modificados:

$$S_{WELCH}(\omega) = \frac{1}{R} \sum_{i=1}^{R} S^{i}{}_{SPERM}(\omega) \qquad \text{[Ec.75]}$$

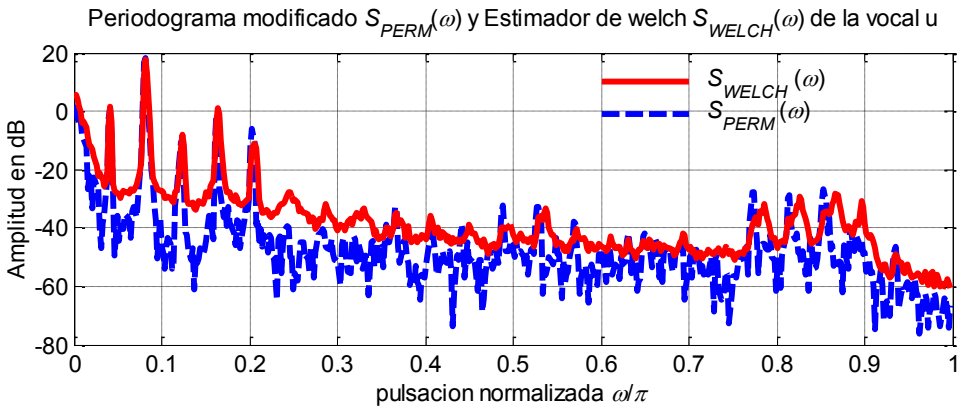
Si los periodogramas modificados de cada trama son independientes entre sí, la varianza o variabilidad del periodograma de Welch se reduce por *R*, esto es:

$$V\{S_{WELCH}(\omega)\} = \frac{1}{R} \cdot V\{S_{SPERM}(\omega)\} \qquad \text{[Ec.76]}$$

El problema del estimador de Welch es que al reducir la longitud de la ventana de *N* a *L* muestras, el ancho del lóbulo principal de la ventana aumenta perdiendo resolución espectral. Así, el estimador de Welch reduce la variabilidad a costa de perder resolución espectral. Para reducir este efecto negativo hay que aumentar el valor de *L* todo lo que sea posible. Una forma de aumentar *L* sin reducir *R*, ya que *N* es un número fijo, es solapar las tramas. El problema de solapar excesivamente las tramas es que los periodogramas de cada trama dejan de ser independientes entre sí, condición de reducción de la varianza. Así se debe llegar a un compromiso entre el aumento del lóbulo principal o pérdida de resolución y la disminución de la variabilidad. Como solución de compromiso, se permite un solape de hasta *L*/2 muestras.

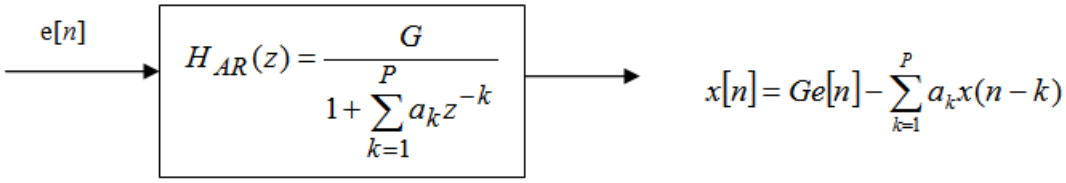
Ejemplo de estimador de Welch de una vocal *u* sostenida. Obsérvese como se reduce la variabilidad (el trazo aparece más suave) y como aumenta el ancho de los picos espectrales.

```
% Método de Welch
u=wavrecord(16000,8000,'double');
N=length(u);
L=1024;
S=L/4;        %solapamiento
R=fix(N/(L-S))
w=hanning(L);
w=w.*sqrt(L/sum(w.^2));
Uwelch=zeros(L,1);
for i=0:R-1,
    Uwelch=Uwelch+(abs(fft(w.*u(i*(L-S)+1:i*(L-S)+L))).^2)./L;
end
Uwelch=Uwelch./R;
Uwelch=Uwelch(1:L/2);
```



6.3 Estimación espectral paramétrica: Modelo de predicción lineal (LP) o autorecurrente (AR)

En el caso de la estimación espectral paramétrica, nos centramos en el modelo AR que supone la señal obtenida como la salida de un filtro de solo polos cuya entrada es ruido blanco.



Así, el espectro de $x[n]$ será:

$$S_{AR}(\omega)=\frac{G^2}{\left|1+\sum_{k=1}^{P}a_k e^{-j\omega k}\right|^2}$$

Por lo tanto hay que calcular los coeficientes $\{a_k\}_{k=1}^{P}$, el orden P y la ganancia del filtro para obtener el espectro.

Analizando el modelo impuesto a la señal, la relación que expresa esta función indica que la muestra n de la señal puede obtenerse como combinación lineal de las P muestras anteriores más un error de predicción. Los procesos que cumplen esta relación o propiedad se denominan linealmente predecibles (procesos LP), autorecurrente o autorregresivos (AR). Sus realizaciones serán señales LP o AR.

Para estimar los parámetros LP o AR de $x[n]$ se despeja el error de predicción (entrada al filtro)

$$e[n] = x[n] - \left(-\sum_{k=1}^{P} a_k x[n-k] \right) = \sum_{k=0}^{P} a_k x[n-k], \text{ siendo } a_0=1 \qquad \text{[Ec.77]}$$

Y se minimiza su energía

$$\frac{\partial E\{e[n]^2\}}{\partial a_m} = E\left\{ \frac{\partial e[n]^2}{\partial a_m} \right\} = E\left\{ 2e[n]\frac{\partial e[n]}{\partial a_m} \right\} = 0 \qquad m=1,2,....,P \qquad \text{[Ec.78]}$$

de la anterior expresión de $e[n]$, se tiene que:

$$\frac{\partial e[n]}{\partial a_m} = \frac{\partial}{\partial a_m}\left[\sum_{k=0}^{P} a_k x[n-k] \right] = x[n-m] \qquad m=1,2,...,P$$

Por lo tanto

$$E\left\{ 2e[n]\frac{\partial e[n]}{\partial a_m} \right\} = E\{2e[n]x[n-m]\} = 0 \qquad m=1,2,...,P \qquad \text{[Ec.79]}$$

a esta expresión $E\{e[n]x[n\text{-}m]\}=0$ se le denomina principio de ortogonalidad. De él se deduce que el error cometido $e[n]$ en la predicción de $x[n]$ no tiene nada en común con las muestras $x[n\text{-}m]$ $m=1,2,...,P$ utilizadas para predecir $x[n]$, esto es, el error de predicción y las muestras utilizadas para predecir son ortogonales.

Sustituyendo en el principio de ortogonalidad el valor de $e[n]$ se tiene:

$$E\{e[n]x[n-m]\} = E\left\{ \left[\sum_{k=0}^{P} a_k x[n-k] \right] x[n-m] \right\} = E\left\{ \sum_{k=0}^{P} a_k x[n-k]x[n-m] \right\} = \sum_{k=0}^{P} a_k E\{x[n-k]x[n-m]\} = 0$$

donde $E\{x[n\text{-}k]x[n\text{-}m]\}$ es la autocorrelación $R_x[|m\text{-}k|]$ $x[n]$.

Sustituyendo la autocorrelación en la anterior expresión queda

$$\sum_{k=0}^{P} a_k R_x(|m-k|) = 0, \qquad m=1,2,...,P \qquad \text{[Ec.80]}$$

como $a_0=1$, se despeja dicho término obteniendo

$$\sum_{k=1}^{P} a_k R_x(|m-k|) = -R_x(m), \qquad m=1,2,...,P \qquad \text{[Ec.81]}$$

que son las denominadas ecuaciones normales.

Estas ecuaciones pueden escribirse para cada *m:*

$m=1;$ $a_1R_x(0)+ a_2R_x(1)+...+ a_PR_x(P\text{-}1)= -R_x(1)$
$m=2;$ $a_1R_x(1)+ a_2R_x(0)+...+ a_PR_x(P\text{-}2)= -R_x(2)$
$m=3;$ $a_1R_x(2)+ a_2R_x(1)+...+ a_PR_x(P\text{-}3)= -R_x(3)$

……
$m=P;\ \ a_1R_x(P\text{-}1)+ a_2R_x(P\text{-}2)+...+ a_PR_x(0)= -R_x(P)$
que en notación matricial se escribe

$$\begin{pmatrix} R_x(0) & R_x(1) & \cdots & R_x(P-1) \\ R_x(1) & R_x(0) & \cdots & R_x(P-2) \\ \cdots & \cdots & \cdots & \cdots \\ R_x(P-1) & R_x(P-2) & \cdots & R_x(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_P \end{pmatrix} = - \begin{pmatrix} R_x(1) \\ R_x(2) \\ \cdots \\ R_x(P) \end{pmatrix}$$

A la primera matriz la denominaremos matriz de autocorrelación $\overline{\overline{R_x}}$ de dimensión P y a los dos vectores nos referiremos como $\bar{a}=(a_1,\ a_{2,\ \ldots\ ,}\ a_P)^T$ y $\overline{R_x}=(R_x(1),\ R_x(2),\ \ldots\ ,\ R_x(P))^T$. Con esta notación, las ecuaciones normales quedan:

$$\overline{\overline{R_x}}\,\bar{a} = -\overline{R_x} \qquad \text{[Ec.82]}$$

de donde los coeficientes $\{a_k\}_{k=1}^{P}$ del filtro se pueden obtener

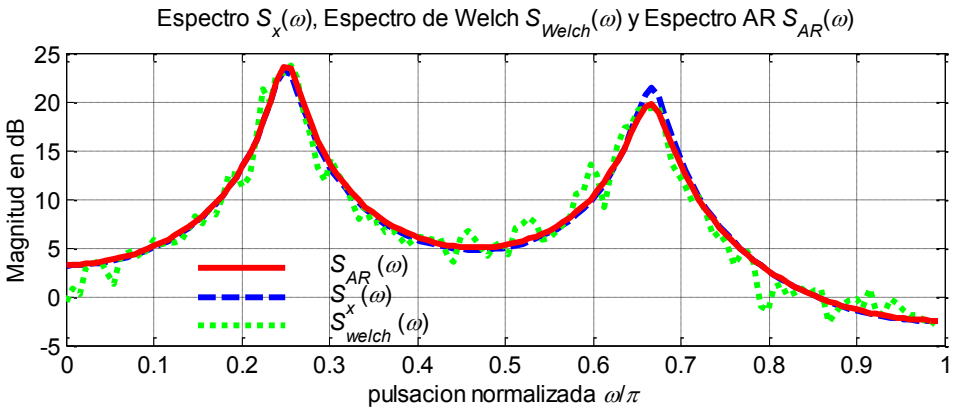
$$\bar{a} = -inv\left(\overline{\overline{R_x}}\right)\overline{R_x} \qquad \text{[Ec.83]}$$

La ganancia del filtro $E\{e[n]^2\}$ se obtiene:

$$G^2 = E\left\{e[n]^2\right\} = R_x(0) + \sum_{k=1}^{P} a_k R_x(k)$$

Ejemplo de estimación de espectro AR.

```
% se genera una realización de un proceso estocástico AR
% Modelo del proceso
a=[ 1.0000   -0.3935    0.5287   -0.3551    0.8145];
G=2.3;
Sx=abs(freqz(G,a,129)).^2;
% generación realización del proceso
x=filter(G,a,randn(2056,1));
% Espectro de Welch: Swelch
H=spectrum.welch('hamming',256,50);
Swelch=psd(H,x);
% se estima el espectro AR de orden 4
P=4;
% se estima la autocorrelación sesgada
R=xcorr(x,P,'biased');R=R(P+1:end);
% se forma la matriz de toeplitz
Rx=toeplitz(R(1:end-1));
% se forma el vector r
r=R(2:end);
% se resuelven las ecuaciones normales
ae=-inv(Rx)*r; %ae=[-0.3783    0.5218   -0.3575    0.8176]
% se estima G
Ge=sqrt(R(1)+sum(ae.*R(2:end)));   %Ge=2.366
% se estima su espectro
[Har,w]=freqz(Ge,[1; ae],129);
```



6.4 Estimación espectral de señales estacionarias a tramos: Espectrograma

Los estimadores espectrales anteriores suponen que la relación entre muestras no varía en las *N* muestras de $x[n]$, pero en la práctica señal varían con el tiempo, esto es, no son estacionarias. En este caso resulta interesante la hipótesis de estacionariedad a tramos para conocer como varía la información de la señal en el tiempo.

Una señal estacionaria a tramos es aquella en la que su espectro se mantienen constante durante un intervalo de tiempo para, a continuación cambiar a otro valor en el cual se mantendrán constantes otro intervalo de tiempo, y así sucesivamente. Por ejemplo, la voz se puede considerar una señal estacionaria a tramos con intervalos de aproximadamente 20mseg.

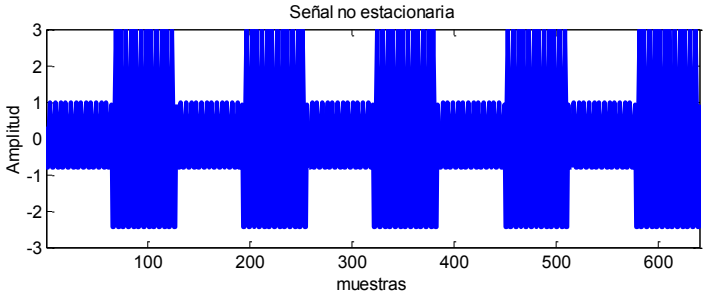
El procedimiento para analizar estas señales es estimar el espectro de cada tramo estacionario y representar la secuencia de momentos en una sola gráfica. Concretamente, la táctica habitual es:

1. Se divide la señal en tramos, generalmente solapados (no más de la mitad), multiplicando cada tramo por una ventana. La longitud del tramo no debe ser mayor a la del intervalo en que se asume la propiedad de estacionariedad.
2. Se calcula el espectro en cada tramo
3. Se concatenan los espectros de cada tramo, representando la evolución temporal del momento estimado en cada tramo.

La representación tendrá dos ejes: el eje del tiempo y el eje de la frecuencia. Así se puede representar como una imagen en dos dimensiones (eje temporal y eje de frecuencia) dibujando con colores más oscuros las zonas de mayor potencia del espectro y con colores más claros las zonas donde el valor de la potencia del espectro es menor. A esta representación se le denomina espectrograma.

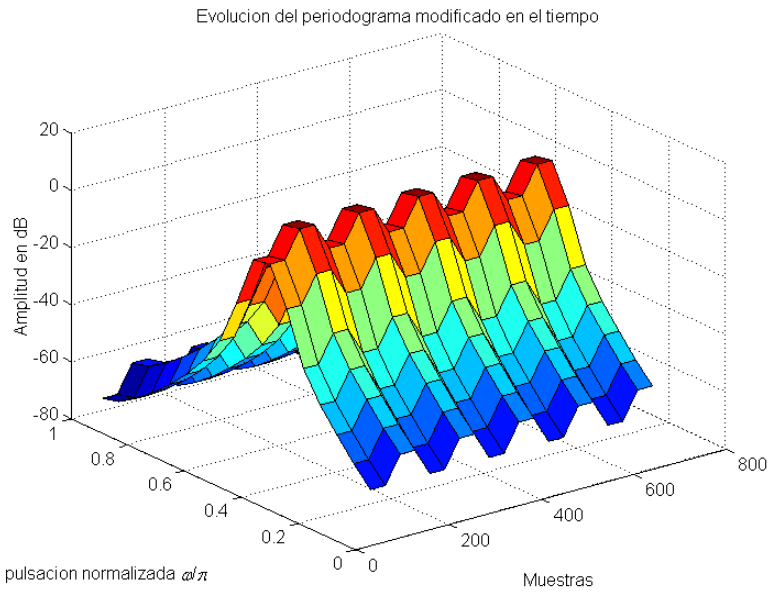
Ejemplo: sea la señal

```
n=[1:64]';
y1=cos(0.4*pi.*n);
y2=3*cos(0.4*pi.*n);
y=[y1;y2;y1;y2;y1;y2;y1;y2;y1;y2];
```



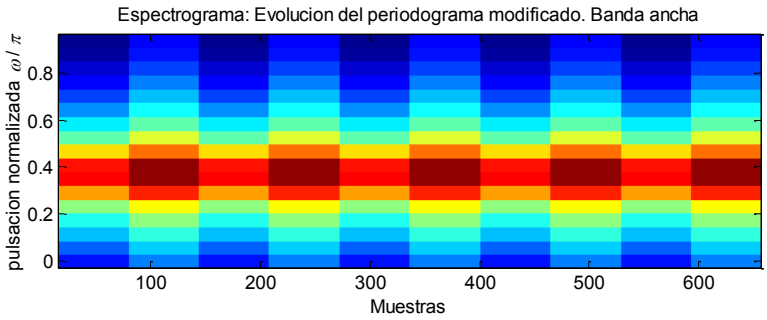
Para calcular su espectrograma se divide la señal en tramos de L muestras, y se estima el espectro de cada trama. Evidentemente el espectro de cada tramo será una *sinc* al cuadrado centrada en 0.4π. El espectro de los tramos con el tono de mayor amplitud tendrán una *sinc* al cuadrado de mayor amplitud que la *sinc* de los tramos con el tono de menor energía. Concatenando las *sinc* del espectro de cada tramo y representándolas en tres dimensiones se tiene, en el caso de L=32:

```
B=spectrogram(y,hanning(32),0,32);
EB=(abs(B).^2)/32;      % valor del periodograma modificado
[F,N]=size(B);
surf([1:N]*32,[0:F-1]./F,10*log10(EB))
```



Si en vez de representar la concatenación de espectros en escala logarítmica en tres dimensiones, se representa como una imagen en dos dimensiones conservando el eje de frecuencia y tiempo e indicando la potencia espectral (en este caso la amplitud de la sinc al cuadrado con colores) con colores indicando los colores más vivos (rojos) mayor potencia espectral y los colores más fríos (azules) menor potencia espectral), se obtiene el denominado espectrograma, que en el caso del ejemplo es:

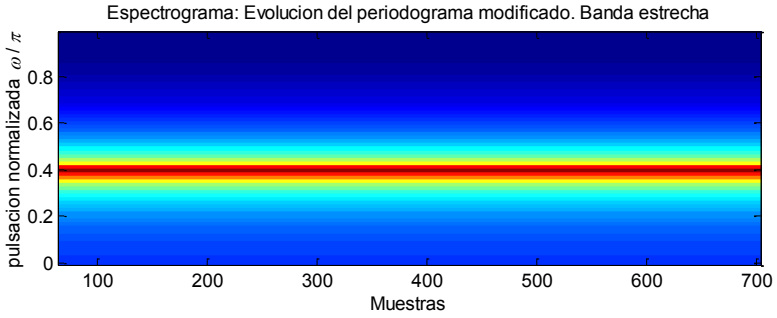
```
B=spectrogram(y,hanning(32),0,32);
EB=(abs(B).^2)/32;
[F,N]=size(B);
imagesc([1:N]*32,[0:F-1]./F,10*log10(EB));axis xy
```



En este espectrograma puede observarse claramente la situación de los tonos de mayor amplitud (tonos rojos más intensos en 0.4π) a los tonos de menos amplitud. También pueden apreciarse claramente las transiciones como líneas verticales: una transición (discontinuidad) genera potencia espectral en todas las frecuencias originando dichas líneas. El problema de esta representación, con mucha resolución temporal, es que no resalta bien la frecuencia del tono, esto es, tiene alta resolución temporal, pero baja resolución en frecuencia. A este espectrograma se le denomina de banda ancha.

Para aumentar la resolución en frecuencia, se aumenta la longitud de la ventana de 32 a 128 muestras. El espectrograma resultante es el siguiente:

```
B=spectrogram(y,hanning(128),0,128);
EB=(abs(B).^2)/128;[F,N]=size(B);
imagesc([1:N]*128,[0:F-1]./F,10*log10(EB));axis xy
```



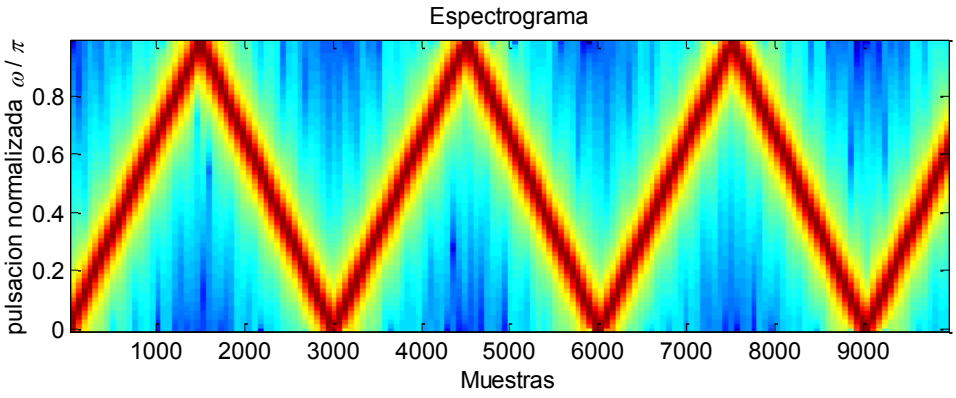
puede observarse como se marca bien el tono en 0.4π a costa de perder resolución temporal pues la estimación espectral de cada tramo largo promedia la energía del tramo. A esta representación se le denomina espectrograma de banda estrecha.

Anotar que el espectrograma representa el valor de la frecuencia instantánea, esto es, si tenemos la señal $x[n]=\cos(\vartheta(n))$, cuya fase instantánea es $\vartheta(n)$, la curva representada en su espectrograma será la de su frecuencia instantánea: $f_i(n)=\dfrac{d\vartheta(n)}{dn}$, esto es, representa la variación de la fase en el tiempo.

Por ejemplo, el espectrograma de la señal $x[n]=\cos(\dfrac{\pi}{3000}n^2)$ representará la recta $2\pi m/3000$, esto es, vale π en n=1500,4500...

Comprobación en Matlab:

```
n=[0:9999];
x=sin(pi*(n.^2)/3000);
B=spectrogram(x,hanning(128),64,128);
[F,N]=size(B);
imagesc([1:N]*128/2,[0:F-1]./F,10*log(abs(B)));axis xy
```

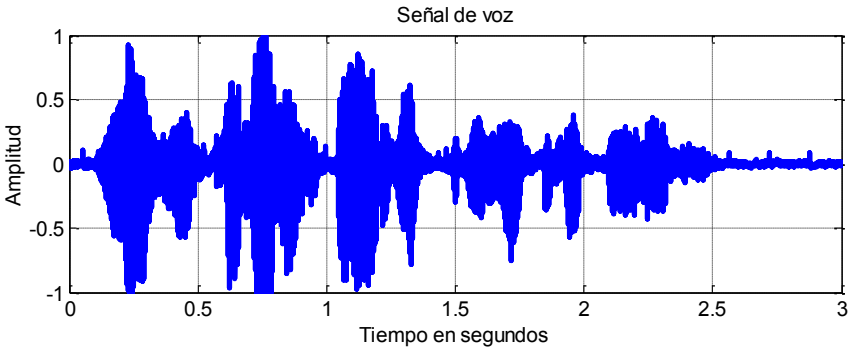


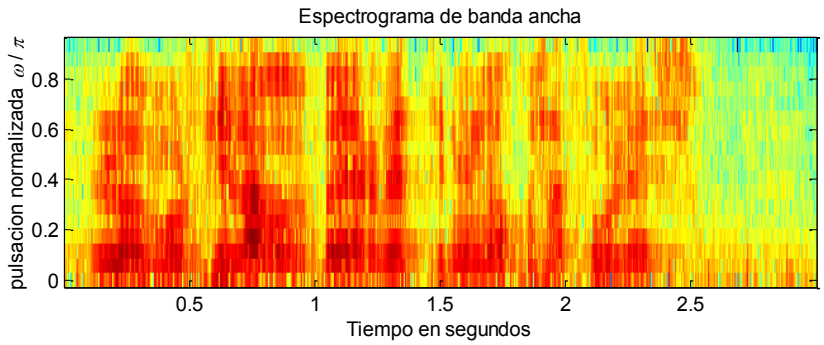
Así podríamos decir que el espectrograma es una representación que nos permite seguir los picos (formantes) del espectro de una señal. Así para señales más complejas como la

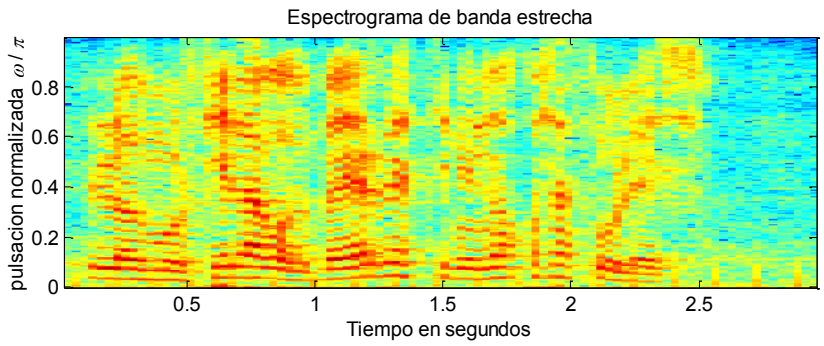
voz, podemos seguir la evolución de los formantes mediante un espectrograma de banda estrecha. Por esta razón, los espectrogramas de banda estrecha tienen una aspecto de líneas horizontales, mientras que los de banda ancha tienen aspecto de líneas verticales.

Un ejemplo de espectrograma de banda ancha y banda estrecha de voz:

```
voz=wavrecord(24000,8000,'double')
% Banda ancha
B=spectrogram(voz,hanning(32),16,32);
EB=(abs(B).^2)/32;
[F,N]=size(B);
imagesc([1:N]*16,[0:F-1]./F,10*log10(EB));axis xy
% Banda estrecha
B=spectrogram(voz,hanning(512),256,512);
EB=(abs(B).^2)/512;
[F,N]=size(B);
imagesc([1:N]*256,[0:F-1]./F,10*log10(EB));axis xy
```







6.5 Problemas de Aula

-Problema 6.1

La tabla siguiente contiene las muestras de un proceso estocástico del que se desea estimar su espectro. Para ello se decide utilizar diferentes métodos que van a ir obteniéndose en este problema.

n	0	1	2	3	4	5	6	7	8	9
x(n)	1	-1	3	5	4	0	-2	-3	6	0

d) Calcule el estimador de Blackman Tukey utilizando una ventana triangular de duración 3 y el estimador sesgado de la autocorrelación

e) Obtenga el estimador de Welch con segmentos de 4 muestras y un solape del 50% para $\omega=\pi$.

f) Suponga que las muestras de la tabla corresponden a un proceso que sigue un modelo AR de orden 1. Obtenga la estima de su Densidad Espectral de Potencia mediante métodos paramétricos.

-Problema 6.2:

Se han estimado las dos primeras muestras de la autocorrelación de un proceso *x[n]* AR de orden 1. Estos valores son *R(0)=1* y *R(1)=0.8*. Se pide

a) Calcular los parámetros o coeficientes del modelo AR(1) que estima la densidad espectral del proceso $S_x(w)$ y escribid la expresión de $S_{AR}(w)$

b) Estimar el valor de la autocorrelación *R(2)* a partir del modelo AR anterior, suponiendo que el proceso *x[n]* se ajusta exactamente al modelo AR calculado.

-Problema 6.3:

Calcule la *x[n]* del siguiente segmento de espectrograma.

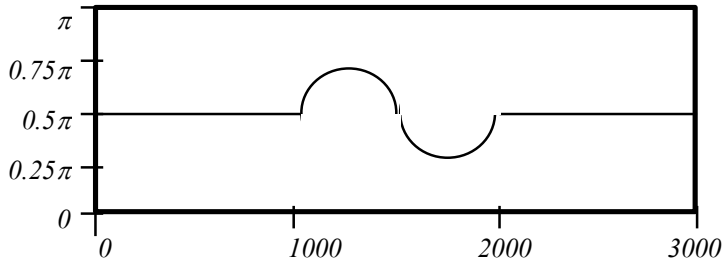


Figura 1. Espectrograma del problema 6.3

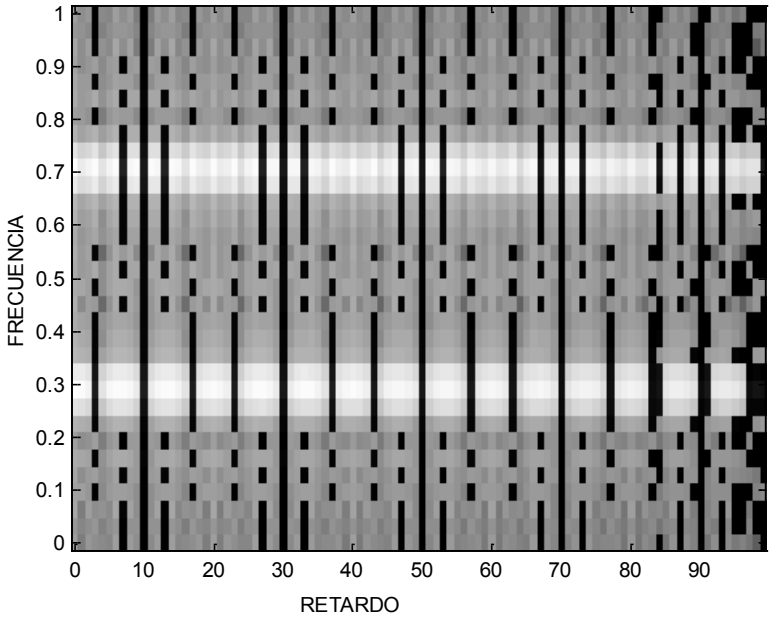
-Problema 6.4:

Dibujar el espectrograma de:

$$x[n] = cos\left(\frac{\pi}{2}n + 8\,cos\frac{\pi}{32}n\right)$$

-Problema 6.5:

Al calcular el correlograma de una señal *x[n]* se obtiene que uno de los planos frecuencia retardo nos da la siguiente imagen (más claro indica mayor valor del correlograma)



a) Se pide escribir, de forma razonada, la fórmula de la señal de entrada *x[n]* (suponga que la señal x[n] es estacionaria y ergódica).

b) Si tenemos el correlograma de una señal *x[n]* en una matriz tridimensional $C_x(i,j,k)$, donde *i* es el índice del eje de tiempo, *j* el índice del eje de frecuencia, y *k* el índice del eje de retardos, se pide realizar un programa en Matlab que represente el espectrograma de *x[n]* a partir de la matriz $C_x(i,j,k)$.

c) Si tenemos el correlograma de una señal *x[n]* en una matriz tridimensional $C_x(i,j,k)$, donde *i* es el índice del eje de tiempo, *j* el índice del eje de frecuencia, y *k* el índice del eje de retardos, se pide realizar un programa en Matlab que calcule la autocorrelación de la señal *x[n]* a partir de la matriz *Cx(i,j,k)*.

d) Hay algunos correlogramas en donde los retardos del eje de retardos no son enteros. Para obtenerlos, en vez de calcular la autocorrelación como

$$R_x[i] = \sum_{n=-\infty}^{n=+\infty} x[n]x[n+i]$$

se calcula como:

$$R_x^{NE}[i] = \sum_{n=-\infty}^{n=+\infty} x[n]x[n+m(i)]$$

donde *m(i)* son los retardos no enteros.

Se pide en este apartado realizar un programa que dado un vector m que contiene los retardos no enteros, obtenga el valor de la autocorrelación R_x^{NE} con esos retardos no enteros. Tenga en cuenta que si dicho programa recibe un vector m de números enteros, los resultados de R_x^{NE} coincidirán con los de R_x.

6.6 Ejercicios Prácticos

Generación de tonos DTMF

Tradicionalmente la manera de señalizar en telefonía ha sido mediante interrupciones controladas (de 40 msg. a 60 msg.) de la línea telefónica y se le denominaba señalización por pulsos. Sin embargo desde la década de los 70´s, se empezó a concebir nuevos métodos que fueran dentro de la banda telefónica de 300 a 3400 Hz. y que la marcación se enviara por tonos, es decir señales audibles y que sin que agregaran ruido a la línea o transitorios indeseables, se pudieran enviar y detectar en forma inconfundible, por esto se ideo el concepto DTMF.

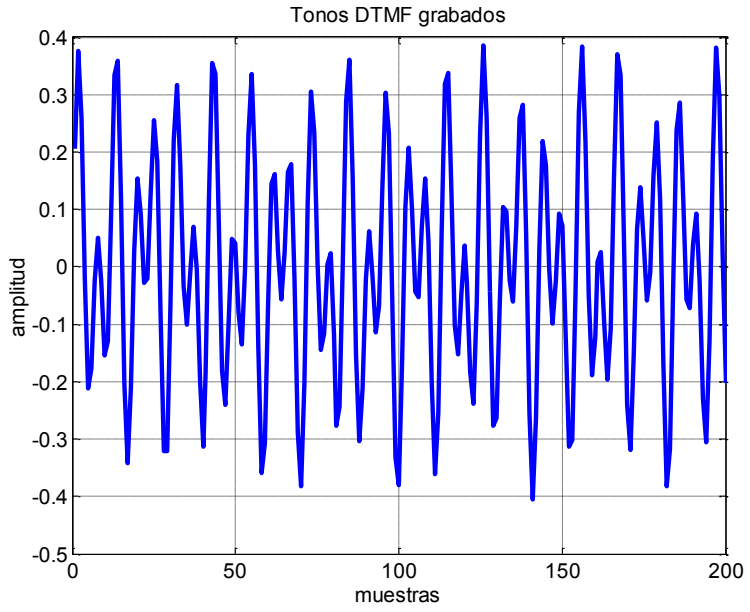
DTMF proviene de las palabras en inglés Dual Tone Multi Frecuency, que significa dos tonos de múltiples frecuencias, y que en español más común denominamos señalización DTMF o marcación por Tonos. Se eligió un conjunto de frecuencias bajas y un conjunto de frecuencias altas o tonos bajos y tonos altos, y para cada dígito del 1 al 0, se enviará la suma algebraica de dos señales senoidales una del conjunto de tonos bajos y otra del conjunto de tonos altos, de acuerdo a la tabla siguiente:

Tecla	Frecuencia	Tecla	Frecuencia
1	**697+1209 Hz.**	**7**	**852+1209 Hz.**
2	**697+1336 Hz.**	**8**	**852+1336 Hz.**
3	**697+1477 Hz.**	**9**	**852+1477 Hz.**
A	**697+1633 Hz.**	**C**	**852+1633 Hz.**
4	**770+1209 Hz.**	*****	**941+1209 Hz.**
5	**770+1336 Hz.**	**0**	**941+1336 Hz.**
6	**770+1477 Hz.**	**#**	**941+1477 Hz.**
B	**770+1633 Hz.**	**D**	**941+1633 Hz.**

Así por ejemplo cuando la tecla 4 se pulsa se envía la señal que es la suma de dos senoidales una de frecuencia 770 Hz. y la otra de 1209 Hz, y la central telefónica podrá decodificar esta señal como el dígito 4 y obrará en consecuencia.

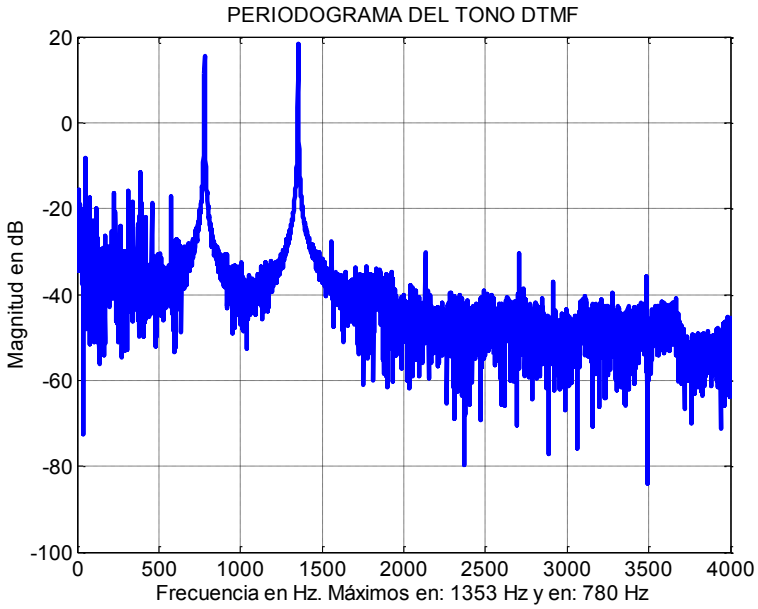
Los tonos solo pueden tener variaciones de ± 1.5 % de su fundamental, y normalmente la señal de tono alto es 3 a 4 dB más fuerte que la de tono bajo. Actualmente existen una gran variedad de circuitos integrados, tanto generadores, como detectores DTMF, así mismo ya empiezan a aparecer en el mercado circuitos microcontroladores que incluyen el detector y generador de DTMF como parte interna de los mismos y con capacidad de control del programa.

En este apartado se pretende realizar un detector DTMF mediante estimadores espectrales directos, puesto que en esta aplicación sólo interesa detectar la frecuencia de los tonos y no es necesario que el proceso sea reversible. Para ello genere el sonido obtenido al pulsar la tecla que estime oportuna. Grabe 1 segundo de este sonido con la correspondiente utilidad de Matlab, a 8000 muestras por segundo y 16 bits por muestra. Denomine a la secuencia obtenida *tonoDTMF*. Represente 200 muestras de dicha grabación y escúchela



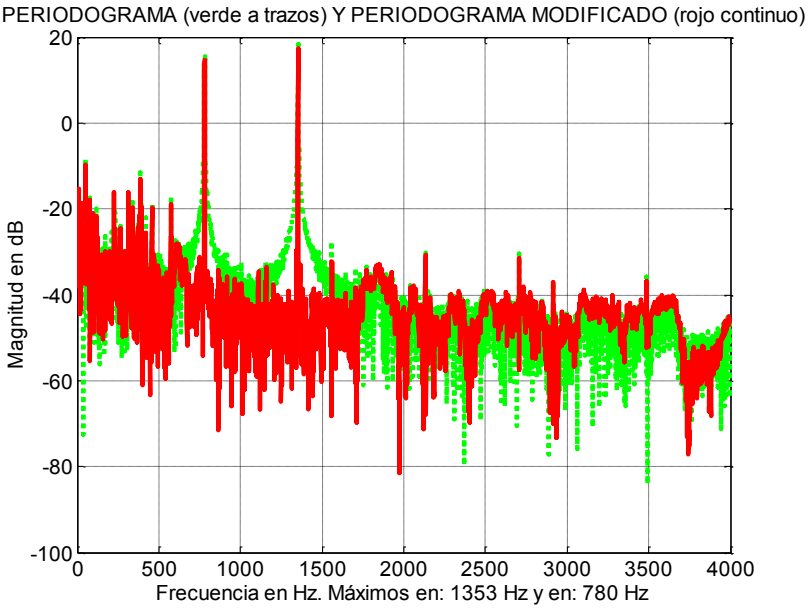
Estimación del periodograma de un tono DTMF

Estime a que tecla pulsada corresponde el sonido *tonoDTMF* mediante el periodograma de *tonoDTMF*. Para ello calcule su periodograma y represéntelo en el rango [0 *fs/2*] Hz. Estime la frecuencia de los picos y la tecla pulsada.



Estimación del periodograma modificado de un tono DTMF

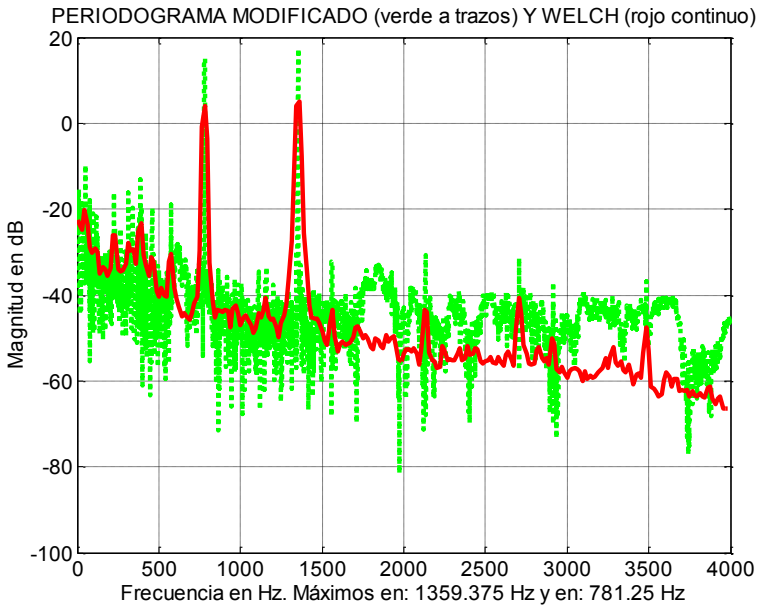
Repita la anterior estimación con el Periodograma modificado con una ventana de Hanning. Represente, solapado el periodograma y el periodograma modificado de *tonoDTMF* en el rango [0 *fs/2*] Hz. Estime la frecuencia de los picos y la tecla pulsada. Note como se aprecia la reducción de la amplitud del lóbulo lateral al usar la ventana en ausencia de ruido.



Estimación del espectro de Welch de un tono DTMF

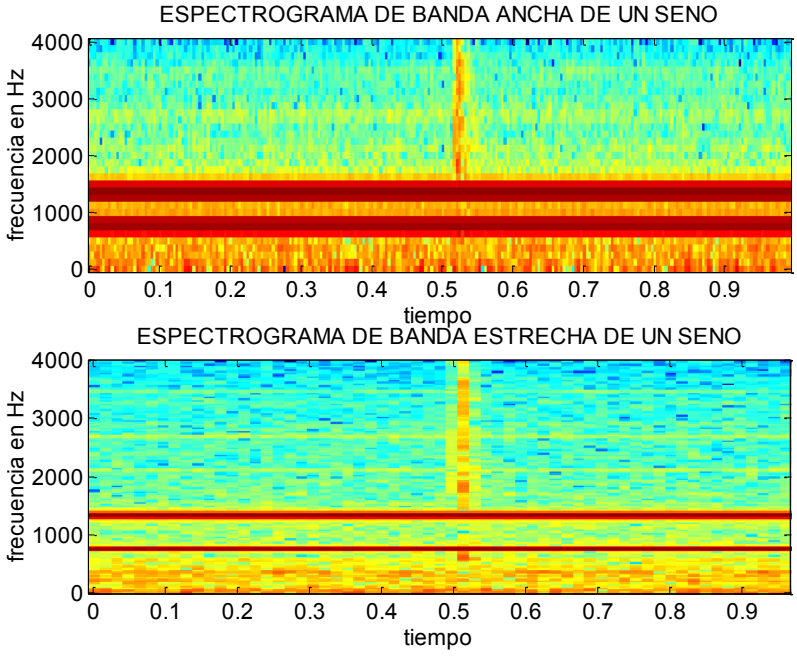
Repita la anterior estimación con el método de Welch. Represente, solapado el periodograma modificado y el periodograma de Welch de *tonoDTMF* en el rango [0

fs/2] Hz.. Estime la frecuencia de los picos y la tecla pulsada. Muestre cómo el método de Welch reduce la consistencia a costa de aumentar el sesgo.



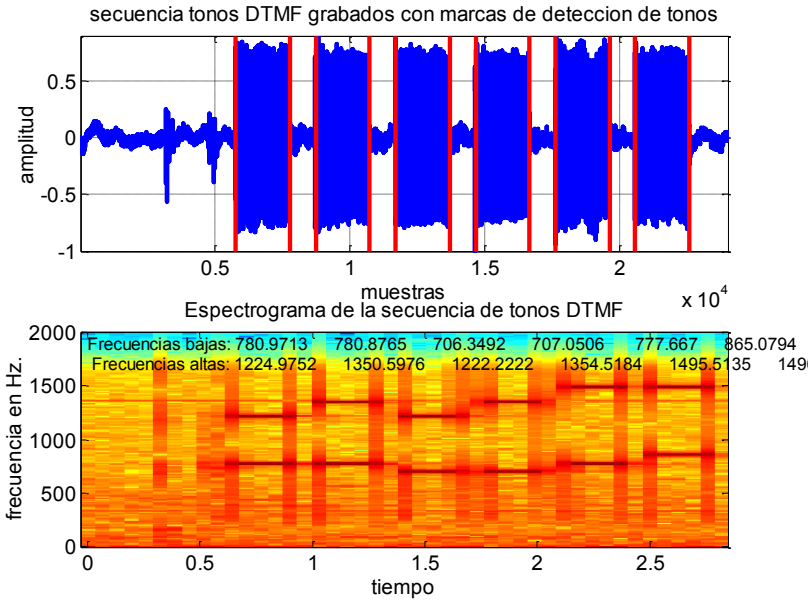
Espectrograma de un tono DTMF

Para visualizar la diferencia entre un espectrograma de banda ancha y estrecha, represente simultáneamente en pantalla el espectrograma de banda ancha (FFT de longitud 64 muestras) y el espectrograma de banda estrecha (FFT de longitud 256) de *tonoDTMF*. Para calcular el espectrograma puede utilizar la función *specgram(s,Lfft,fs)*.



Espectrograma de una secuencia de tonos DTMF

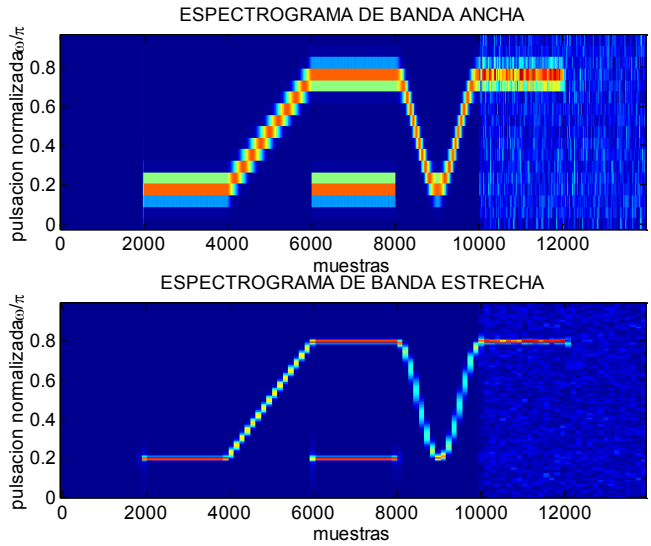
Genere una secuencia de tonoDTMF conteniendo un número de 6 dígitos. Detecte los principios y finales de cada tonoDTMF. Estime el número que representa dicha secuencia. Dibuje el espectrograma de la secuencia.



Espectrograma de una señal de frecuencia variable (*chirp*)

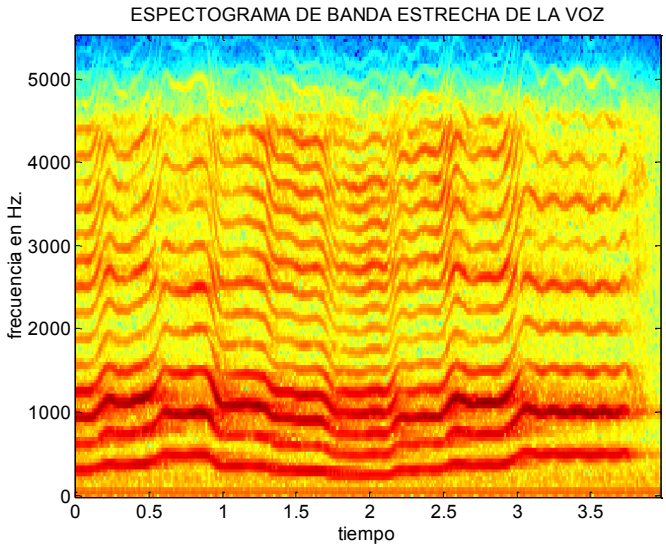
Forme la siguiente secuencia: 2000 muestras de silencio, seguidas de 2000 muestras de un seno de pulsación 0.2π, seguidas de 2000 muestras de una secuencia sinusoidal cuya pulsación comience en 0.2π y varíe linealmente para terminar con una pulsación de 0.8π, seguidas de 2000 muestras de la secuencia suma de dos senos de pulsación 0.2π y 0.8π, seguidas de 2000 muestras de una secuencia sinusoidal cuya pulsación comience en 0.8π y varíe de forma sinusoidal con periodo 4000 y amplitud 0.6π para terminar con una pulsación de 0.8π, seguidas de 2000 muestras de un seno con pulsación 0.8π con ruido aditivo de potencia tal que SNR = 5dB, seguidas de 2000 muestras del ruido.

Represente en pantalla a la vez el espectrograma de banda ancha y estrecha de la secuencia. Comente los resultados. Escuche con el comando *soundsc* a 2000 muestras por segundo la secuencia generada.



Espectrograma de un arpegio de voz

Grabe un fichero de cuatro segundos de voz a 22050 muestras por segundo en el que pronuncie una vocal sostenida en diferentes tonos. Represente el espectrograma de banda estrecha de la señal de voz.



Capítulo VII. Procesado y Análisis de Imagen

El objetivo del procesado de imagen es realizar transformaciones sobre una imagen en el dominio discreto con el fin de mejorarla o extraer alguna información útil de ella. Habitualmente incluye tres pasos:

1. Importar la imagen desde un escáner óptico o cámara digital en cualquiera de los formatos standard existentes en el mercado: bmp, tiff, jpeg (o jpg), gif, png, avi, mpg etc.

2. Analizar y transformar la imagen. Incluye operaciones como compresión de imagen para almacenamiento o transmisión, mejora de imagen, identificación de patrones, etc.

3. Presentación de la imagen, tanto en formato de imagen modificada o informe sobre la imagen.

Las 5 operaciones habituales realizadas en procesado de imagen son:

1. Visualización, generalmente de objetos no visible, por ejemplo en procesado de imágenes multibanda.

2. Realzado de imagen para crear una imagen mejor

3. Análisis de imágenes de interés: segmentación.

4. Caracterización de los objetos de una imagen

5. Reconocimiento de imagen distinguiendo objetos en la imagen

En éste capítulo se introducen someramente ejemplos de las cinco operaciones mencionadas

7.1 Visualización de imágenes

El primer paso en procesado de imagen es comprender que una imagen está compuesta por una matriz de dos dimensiones. Cada pixel o componente de la matriz contiene un

número que representan el nivel de gris de dicho píxel (imagen en escala de grises). Si el nivel de gris es blanco o negro estamos en el caso de una imagen binaria.

Izquierda: Imagen en escala de grises representada por su matriz de pixeles, derecha: ejemplo de imagen a color, escala de grises y binaria

En el caso de imágenes a color, cada pixel está representado por tres números que estarán relacionados con su color, intensidad, luminancia o saturación entre otros dependiendo del formato con el que se trabaje.

1. Si la imagen está en formato RGB los tres números representan la intensidad de rojo, verde y azul, ajustándose a los aquí falta algo

2. Si la imagen esta en formato CMY (se utiliza para codificar colores en dispositivos de impresión) los tres números representan C=G+B, M=R+B y Y=R+G, correspondientes a Cian (C), Magenta (M) y Amarillo (Y).

3. Si la imagen esta en formato YUV (utilizado en radiodifusión de TV del sistema PAL) las tres componentes (una componente de luminancia y dos componentes de crominancia) se obtienen:

$$Y = 0.30R + 0.59G + 0.11B$$
$$U = 0.493(B - Y)$$
$$V = 0.897(R - Y)$$

4. En formato YIQ (utilizado en radiodifusión de TV del sistema NTSC: USA y Japón) la relación es:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.513 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

5. En formato HSI donde cada componente tiene un significado perceptual (H: matriz, S: saturación, e I: intensidad):

$$\begin{bmatrix} I \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/\sqrt{6} \\ 1/\sqrt{6} & -2/\sqrt{6} & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$H = \tan^{-1}(V_2 / V_1)$$

$$S = [V_1^2 + V_2^2]^{1/2}$$

Si cada pixel se representa por más de tres números estamos en el caso de imágenes multiespectrales o hiperespectrales en las que cada número representa la reflectancia en una banda.

También existe el Estándar DICOM: (Digital Imaging and Communications in Medicine). Más que un formato es un estándar para la gestión integral de series de imágenes médicas, desde su almacenamiento hasta su distribución y anotación. Las estaciones de visualización permiten hacer rendering 3D y "viajar" por el interior del cuerpo humano a partir de series DICOM.

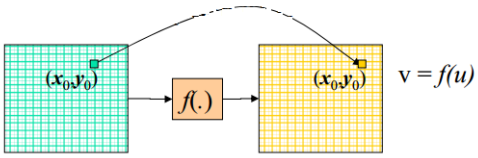
En la calidad de la imagen afectan aspectos como:

1. El muestreo espacial o el número de pixeles por pulgada.

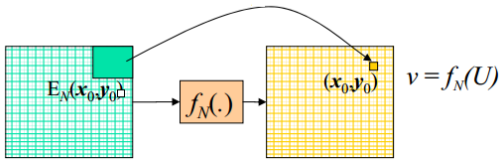
2. La cuantificación o el número de niveles de gris o paleta de colores

7.2 Operaciones básicas de realzado de imagen.

Hay dos tipos de operaciones, las puntuales que son independientes de la posición de los pixeles y las de bloque o vecindad cuyo resultado depende de la vecindad del pixel

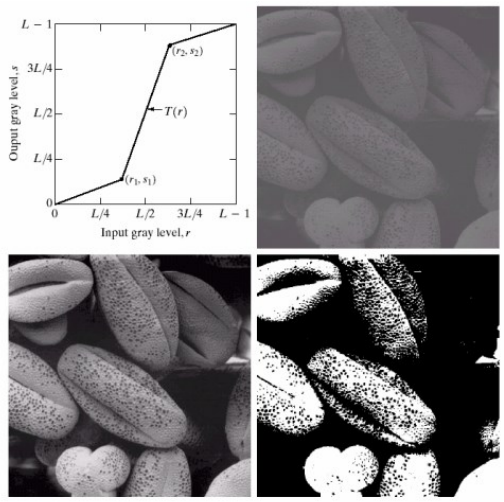


Operación puntual



Operación de bloque

Ejemplos de operaciones puntuales pueden ser la modificación de contrate, recorte o umbralización. Un ejemplo se tiene en la siguiente figura, donde se muestra la gráfica de la función $v = f(u)$ a partir de la cual mejorar el contraste de una imagen, la figura original con bajo contraste (arriba derecha), la figura restaurada (abajo izquierda) y binarizada (abajo derecha).

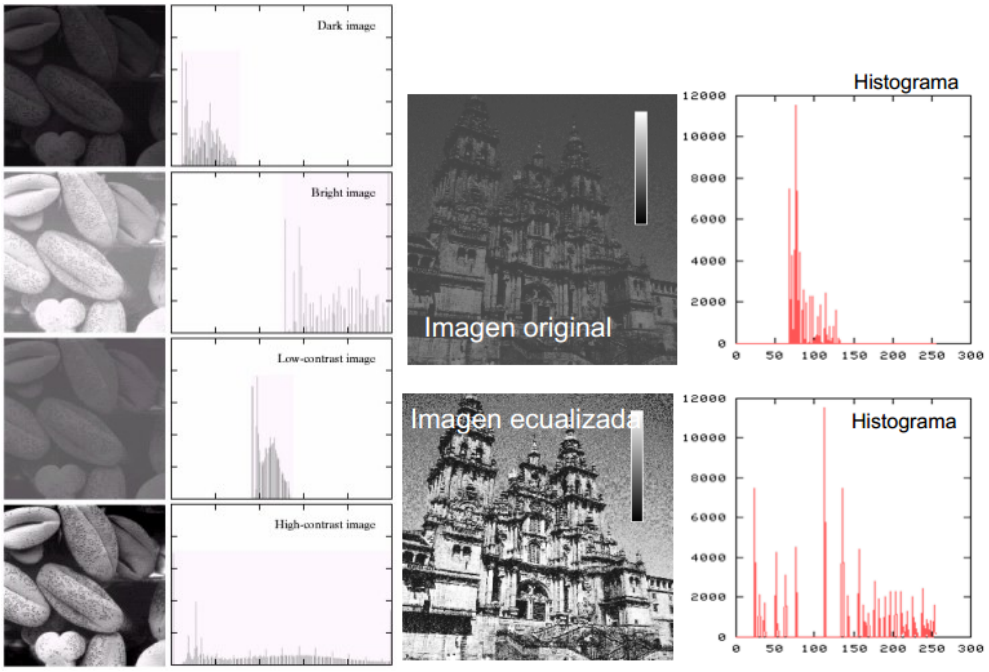


Una aplicación para visualizar bajos niveles de intensidad con mayor margen dinámico es la transformación logarítmica $v = f(u) = 10\log(1+u)$

Otro tipo de operaciones trabajan sobre el histograma de una imagen. El histograma es la representación de la frecuencia relativa de cada color en una imagen. Mide la frecuencia relativa de apariciones de los niveles de gris de una imagen:

$$h(n) = numero\ de\ veces\ que\ aparece\ el\ nivel\ n$$

Modificando el histograma se cambia la imagen como puede verse en el siguiente ejemplo. Si el histograma es transformado a plano se dice que se ha ecualizado la imagen (todos los niveles de gris aparecen por igual).



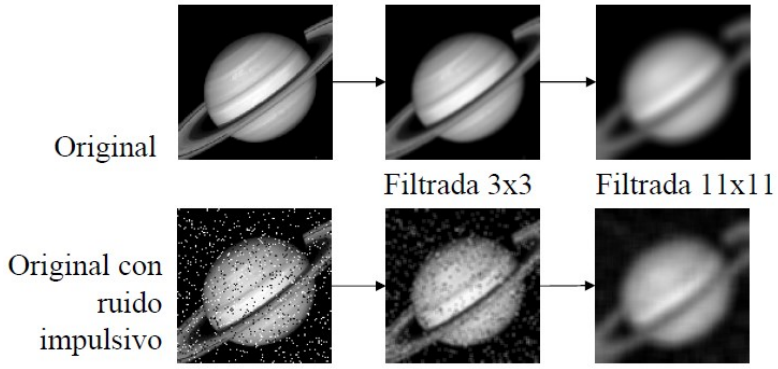
Las operaciones puntuales también pueden utilizarse para detectar movimiento en secuencias de imágenes mediante resta. En el siguiente ejemplo se puede observar como los objetos en movimiento se representan por zonas más densas de píxeles con alto valor de intensidad, mientras que las zonas en reposo se caracterizan por zonas con baja densidad de estos píxeles.

Las operaciones espaciales o de vecindad se definen en un entorno E_N (vecindad) del punto a transformar (m_0, n_0). La herramienta habitual son las operaciones basadas en máscaras espaciales (plantillas, ventanas, kernels o filtros FIR). La máscara es un array pequeño en relación a la imagen (3x3, 5x5, 7x7,...) y los valores de los coeficientes determinan el proceso de transformación.

Los filtros lineales e invariantes en el espacio (LSI) están caracterizados por una función (imagen) respuesta al impulso de modo que

$$O(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k,l) I(x-k, y-l) = h(x,y) * I(x,y)$$

El filtrado paso-bajo suele utilizarse para desenfocar, suavizar, eliminar ruido. Por ejemplo:



En cambio, el filtrado paso alto se utiliza para resaltar bordes, enfocar, detección de piezas, objetivos. Los coeficientes de la máscara deben sumar 0 y sirven para enfatizar regiones con transiciones abruptas de intensidad. En general, se reduce mucho el contraste y aparecen valores negativos, lo cual obliga a escalar o recortar.

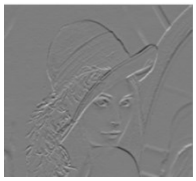
Los filtros paso alto más sencillos son las máscaras de derivación direccional:

1. la máscara de Roberts (gradiente cruzado):

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

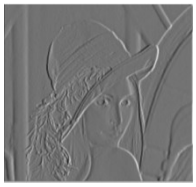
2. Máscara Prewit:

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$



3. Máscara de sobel:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

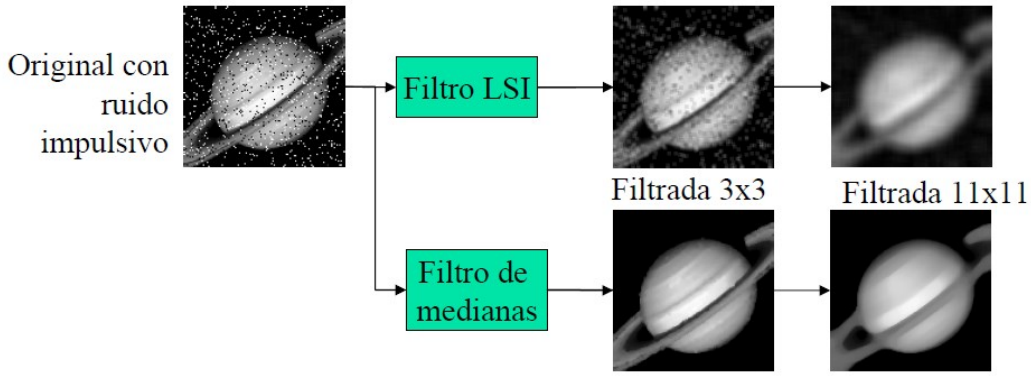


La máscara de Roberts también puede considerarse un filtro derivativo.

También son habituales los filtros de estadísticos ordenados: son filtros en los que la operación a realizar es no lineal. Funcionan ordenando los valores en la vecindad de cada punto de menor a mayor, y obteniendo algún valor a partir de la lista ordenada.

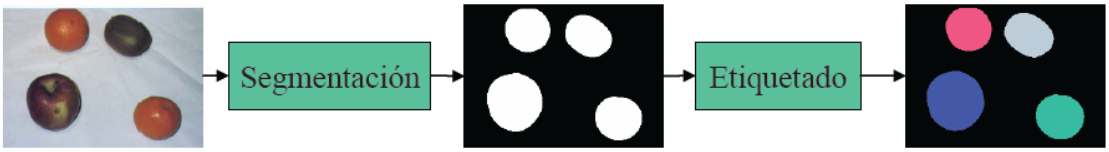
Ejemplos: Mínimo: selecciona el valor más pequeño dentro de la vecindad y asigna ese valor a todos los píxeles dentro de ella, Máximo: selecciona el valor más alto. Mediana:

selecciona el valor en la posición intermedia, el cual suele utilizarse para eliminar ruido impulsivo preservando los bordes de la imagen original.



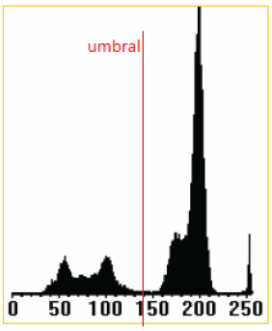
7.3 Análisis de Imagen

Una vez realzada la imagen el siguiente paso es la segmentación: segmentación que divide la imagen en regiones con características similares. Cada una de las regiones de interés (que comparten ciertas propiedades) se denomina objeto. El resultado de la segmentación: separación de objetos. Para diferenciar los objetos, éstos tendrán asignadas unas etiquetas.



La segmentación puede estar basada en características (niveles de gris, color, texturas), transiciones (bordes), modelos (Hough), homogeneidad (fusión de regiones, zonas planas, Propagación de Marcadores) o en Morfológica Matemática.

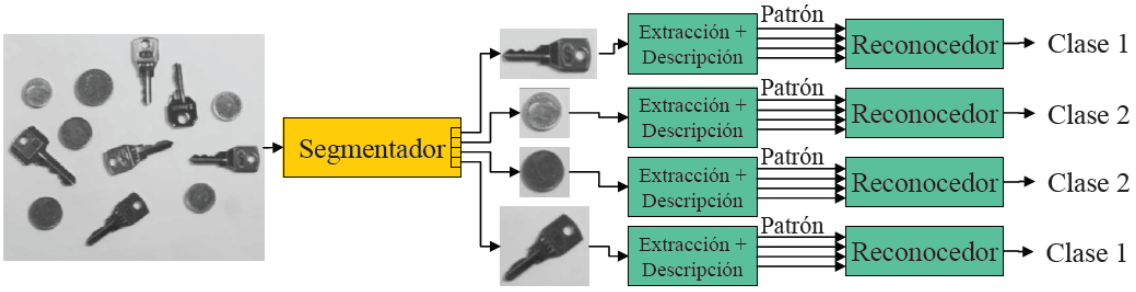
En el caso ideal en que el objeto posea un rango estrecho de niveles de gris frente a un fondo uniforme, podemos establecer un nivel de gris intermedio (umbral) para separar objeto y fondo. Para separar por umbral, es útil recurrir al histograma tal y como lo hace el método de Otsu, cuyo umbral u divide el histograma en dos partes.



$$L(x,y)=\begin{Bmatrix}1, I(x,y)\le u\\ 0, I(x,y)>u\end{Bmatrix}$$

7.4 Caracterización de objetos

Una vez preprocesada la imagen, el siguiente paso para su análisis es la caracterización y reconocimiento de sus objetos. La etapa de extracción de características se encarga de extraer la información discriminatoria. Su principal propósito es reducir la dimensionalidad del problema de reconocimiento de patrones. Un ejemplo del sistema completo puede verse en la siguiente figura donde se quieren separar las llaves y monedas. Mediante las operaciones de procesado previamente estudiadas en éste libro, se segmentan los diferentes objetos presentes en la imagen. De cada uno de los objetos se extraen sus características, por ejemplo su excentricidad, y el reconocedor o clasificador lo asigna a la clase 1 (llave) o clase 2 (moneda).



Uno de los descriptores del contorno de objetos más utilizados son los descriptores de Fourier. Para ellos se define el contorno como una sucesión de N puntos es un sentido determinadop, por ejemplo el de las agujas del reloj: $\{(x_o, y_o), (x_1, y_1), \dots, (x_N, y_N)\}$. Cada par de coordenadas se prepresenta por un complejo $s(k) = x(k) + jy(k)$ y se calcula la transformada de Fourier de dicha secuencia obteniendo

$$a(u) = \frac{1}{N}\sum_{k=0}^{N-1} s(k)e^{-j2\pi uk/N}$$

Estos descriptores, en módulo, son invariantes a la rotación y desplazamiento, Si se normaliza su amplitud también son invariantes al escalado.

Otros descriptores tratan de caracterizar la forma a partir de sus momentos estadísticos de orden $p+q$ pues al igual que con los Descriptores de Fourier para contornos, es posible recuperar la función $f(x,y)$ a partir del conjunto infinito de sus momentos. Para una imagen binaria de dimensiones LxM definida por $f(x,y) = 1$ para el objeto y $f(x,y) = 0$ para el fondo, sus momentos se definen como:

$$m_{p,q} = \sum_{x=0}^{L}\sum_{y=0}^{M} f(x,y)x^p y^q$$

Los momentos invariantes a la traslación se definen como: $\mu_{p,q} = \sum_{x=0}^{L}\sum_{y=0}^{M} f(x,y)(x-\bar{x})^p(y-\bar{y})^q$ siendo $\bar{x} = m_{1,0}/m_{0,0}$ y $\bar{y} = m_{0,1}/m_{0,0}$

Los momentos invariantes al escalado por un factor s *(*$x' = sx$ *y* $y' = sy$*)* se obtienen como $\eta_{p,q} = \mu'_{p,q}/\left(\mu'_{0,0}\right)^{\gamma}$ siendo $\gamma = (p+q+2)/2$ y $\mu'_{p,q} = s^{2\gamma}\mu_{p,q}$.

También existen momentos invariantes a la rotación y reflexión.

Otros descriptores de regiones tratan de caracterizar los atributos geométricos de la forma. Así se describe con su perímetro (P), área (A), centro de masas ($m_{1,0}$ y $m_{0,1}$), compacidad ($\gamma = 4\pi A/P^2$), la elongación o excentricidad del objeto que puede medirse como la relación radio máximo dividido por radio mínimo desde el centro de masas al contorno, la orientación definida como el ángulo del eje correspondiente al mínimo momento de inercia.

Más descriptores de los atributos geométricos son el rectángulo circunscrito que es el menor rectángulo que encierra al objeto. Se supone el rectángulo alineado con la orientación del objeto. Esta definición puede generalizarse al *boundig box* que es el polígono convexo (todos sus ángulos internos iguales o menores de 180º) más pequeño que encierra el objeto. Mencionar como último descriptor geométrico la elipse de ajuste (*best-fit ellipse*) que se define como la elipse cuyo momento de segundo orden es igual al del objeto.

Todos los descriptores de atributos geométricos pueden obtenerse fácilmente con la función *regionprops* de Matlab. El siguiente paso en el análisis de la imagen es el clasificador o etapa de toma de decisiones. Su rol es asignar a la categoría apropiada los patrones de clase desconocida a priori.

7.5 Sistema de clasificación

Clasificar es predecir sobre la naturaleza desconocida de una observación. La tarea de clasificar es un atributo básico en el comportamiento de las personas y animales. Una observación es un vector $x \in R^d$ de dimensión *d* que describe el objeto a clasificar. Las observaciones, también denominadas patrones, pueden ser espaciales (caracteres, imágenes, etc...) y/o temporales (formas de onda, series, etc...). La naturaleza desconocida de la observación se denomina clase $y \in \{1....M\}$.

Clasificar es crear una función $g(x): R^d \longrightarrow \{1...M\}$ denominada clasificador, que asigna cada objeto u observable a su clase. Para crear el clasificador se recurre a la

distribución conjunta (X, Y) que describe la frecuencia de encontrarse un particular par (x, y). La regla g debe reflejar dicha frecuencia de ocurrencia de cada par, sabiendo que diferentes "*y*" pueden dar la misma "*x*".

Una vez definida dicha función, decimos que ocurre un error cuando $g(x) \neq y$, siendo la probabilidad de error $L(g) = P\{g(x) \neq y\}$. En términos de error, el mejor clasificador posible es el que minimiza el error, esto es: $g^* = \underset{g:R^d \to \{1...M\}}{\arg\min}\ P\{g(x) \neq y\}$

Encontrar g^* es el problema de Bayes. A g^* y a $L^* = L\ (g^*)$ se les denomina clasificador de Bayes y error de Bayes respectivamente.

Teorema de Bayes: sea $\{Y_1, Y_2, \ldots, Y_i, \ldots, Y_n\}$ un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero. Sea *X* un suceso cualquiera del que se conocen las probabilidades condicionales $P(X|Y_i)$ Entonces, la probabilidad $P(Y_i|X)$viene dada por la expresión:

$$P(Y_i|X) = \frac{P(X|Y_i)P(Y_i)}{P(X)}$$

donde $P(Y_i)$ son las probabilidades a priori, $P(X|Y_i)$ es la probabilidad de *X* en la hipótesis Y_i y $P(Y_i|X)$ son las probabilidades a posteriori.

Clasificador de Bayes

Para obtener el clasificador de Bayes en un caso de dos clases, suponga que conocemos las distribuciones a priori, esto es, conocemos $P(x|Y = 1)$ y $P(x|Y = 2)$ y las probabilidades de cada clase, esto es la $P(Y = 1)$ y $P(Y = 2)$. Esto implica que conocemos todos los posibles casos de cada clase.

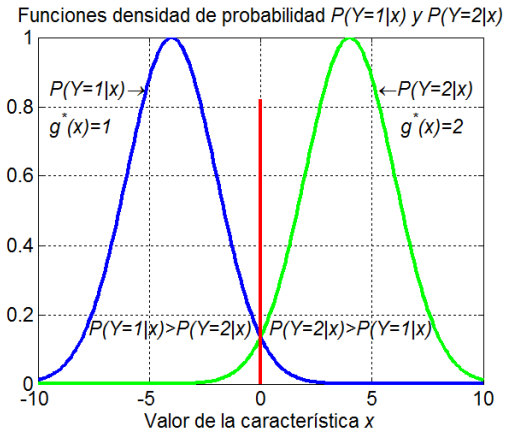
Lo que a nosotros nos interesa para clasificar es la probabilidad a posteriori, esto es $P(Y = 1|x)$ y $P(Y = 2|x)$ las cuales se pueden obtener a partir del teorema de Bayes como:

$$P(Y = 1|x) = \frac{P(x|Y = 1)P(Y = 1)}{P(x)} = \frac{P(x|Y = 1)P(Y = 1)}{\sum_{i=1}^{i=2} P(x|Y = i)P(Y = i)}$$

La función de decisión de Bayes o decisión óptima $g^* : R^d \to \{1\ 2\}$

se define como $g^*(x) = \begin{cases} 1 & si\ \mathrm{P(Y=1\,|\,x) > P(Y=2\,|\,x)} \\ 2 & resto \end{cases}$

y el error cometido será $L^* = L(g^*) = E\{\min(\mathrm{P(Y=1\,|\,x), P(Y=2\,|\,x)})\}$ que se denomina error de Bayes. Un ejemplo puede verse en la siguiente gráfica



Esta regla puede generalizarse a M clases como sigue:

$$x \to Y = k \quad si \quad k = \underset{1 \le i \le M}{\operatorname{argmax}} P(Y = i|x)$$

En este caso el error de Bayes se define $L^* = \sum_{i=1, i \neq k}^{M} P(Y = i|x)$

Aprendizaje con Bases de Datos

Como generalmente las distribución a priori no se conocen, y por tanto g^* es desconocida, para encontrar un clasificador se recurre a una base de datos, esto es, a pares $D_n=(X_i,Y_i)$ con $i=1,2,...,n$ observados en el pasado. En este caso el clasificador será una aplicación $R^d x\{R^d x\{1...M\}\}^n \to \{1...M\}$

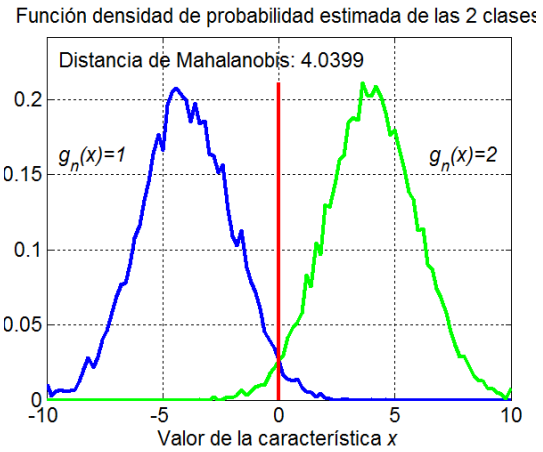
Para que dicha base de datos sea buena, dicha secuencia de datos debe estar idénticamente e independientemente distribuida, esto es, debe tener las mismas distribuciones que $P(x|Y = i), i = 1, \dots, M$. Dicha suposición es muy restrictiva y en la práctica es difícil de cumplir.

El clasificador construido a partir de dicha base de datos D_n se denomina g_n. El proceso de creación de g_n se denomina aprendizaje supervisado y la clase es predicha como $Y = g_n(X; D_n)$.

La aplicación del clasificador de Bayes a este caso sería estimar las funciones densidad de probabilidad de cada variable aleatoria, mediante el histograma, y utilizarlas como si fueran las reales. Un ejemplo puede verse en la siguiente gráfica

```
% clasificador bayes estimado
m1=-4;m2=4;sigma1=2;sigma2=2;
x1=randn(10000,1)*sigma1+m1;
x2=randn(10000,1)*sigma2+m2;
% se estiman las funciones densidad de probabilidad,
x=linspace(-10,10,101);
paso=x(2)-x(1);
fdp1=hist(x1,x);fdp1=fdp1./sum(fdp1.*paso);
```

```
fdp2=hist(x2,x);fdp2=fdp2./sum(fdp2.*paso);
% Distancia de Mahalanobis
p=0.5;
m1=mean(x1);
m2=mean(x2);
sumcov=p*cov(x1)+(1-p)*cov(x2);
DM=sqrt((m1-m2)*inv(sumcov)*(m1-m2));
```



En este caso, para calcular el error de clasificación se recurre a estimarlo sobre la base de datos como:

$$\hat{L}_n = \frac{1}{n}\sum_{i=1}^{n} I_{\{g_n(X_i)\neq Y_i\}}$$

Esta estimación del error se llama método de método de resustitución y es una estimación es muy optimista pues obtiene el error sobre los datos de entrenamiento del clasificador. Para evitar este sesgo, el procedimiento habitual es utilizar una secuencia de test $T_m = ((X_{n+1}, Y_{n+1}), \ldots, (X_{n+m}, Y_{n+m}))$ y calcular:

$$\hat{L}_{n,m} = \frac{1}{m}\sum_{j=1}^{m} I_{\{g_n(X_{n+j})\neq Y_{n+j}\}}$$

que es más realista.

```
% se obtiene el umbral entre la clase 1 y clase 2 el que fdp2>fdp1
umbral=x(find(diff(fdp2>fdp1)==1));
% Calculo de errores método resustitución
Errores=sum(x1>umbral)+sum(x2<umbral);
% Calculo de errores realista
% se generan de nuevo las distribuciones
x1=randn(10000,1)*sigma1+m1;
x2=randn(10000,1)*sigma2+m2;
Errores=sum(x1>umbral)+sum(x2<umbral);

      Errores resustitución: 2.125%
      Errores realista: 2.42%
```

Otra forma de mostrar los errores cometidos es la matriz de confusión, que indica de forma matricial los errores y aciertos cometidos en el proceso de clasificación. En la gráfica se muestra un ejemplo para el caso biclase.

		Clases estimadas	
		Clase A	Clase B
Clases reales	Clase A	94.63 %	5.37 %
	Clase B	13.95 %	86.05 %

Evaluación de la dificultad del problema de clasificación

Teniendo claro que cada aplicación tiene su solución, ante un problema de clasificación es interesante estimar su error de Bayes L^*, pues si es grande cualquier clasificador funcionará mal. La estimación de L^* se puede realizar a partir de la estimación de medidas de distancia entre los datos. Por ejemplo:

Distancia de Mahalanobis

La distancia de Mahalanobis entre dos variables aleatorias X_0 y X_1 nos da una idea de la distancia entre dos distribuciones y se define como:

$$\Delta = \sqrt{(m_1 - m_0)^T \Sigma^{-1} (m_1 - m_0)}$$

Dónde: $m_0 = E\{X_0\}$, $m_1 = E\{X_1\}$ son las medias. Si se definen las matrices de covarianza $\Sigma_0 = E\{(X_0 - m_0)(X_0 - m_0)^T\}$ y $\Sigma_1 = E\{(X_1 - m_1)(X_1 - m_1)^T\}$, se define $\Sigma = p\Sigma_0 + (1-p)\Sigma_1$ siendo p la probabilidad de cada variable aleatoria.

Se pude demostrar que si Δ es grande, L^* es pequeña, en concreto:

$$L^* \leq \frac{2p(1-p)}{1 + p(1-p)\Delta^2}$$

Así tenemos una cota superior para L^*

```
p=0.5;
m1=mean(x1);
m2=mean(x2);
sumcov=p*cov(x1)+(1-p)*cov(x2);
DM=sqrt((m1-m2)*inv(sumcov)*(m1-m2));
CotaSuperiorL=2*p*(1-p)/(1+p*(1-p)*DM*DM)*100;

      Cota de L*: 10.0968 %
```

Si L^* es muy grande, para aumentar la distancia entre las clases se recurre a buscar otras características o transformar las ya existente en otras más eficientes.

7.6 Ejemplos de clasificadores

Reglas de vecino más cercano (KNN)

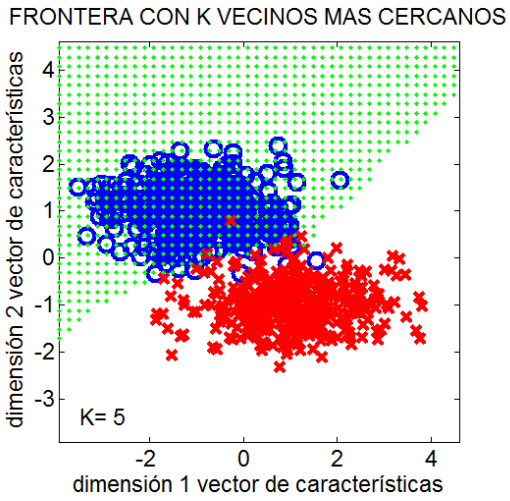
Son clasificadores que asignan la entrada a la clase más cercana, así su regla de clasificación es del tipo:

$$g_n(x) = \begin{cases} 1 & si \sum_{i=1}^{n} \omega_{ni} I_{\{Y_i=1\}} > \sum_{i=1}^{n} \omega_{ni} I_{\{Y_i=0\}} \\ 0 & resto \end{cases}$$

Siendo $\omega_{ni} = 1/k$ si X_i está entre los k vecinos más cercanos y $\omega_{ni} = 0$ en los demás.

Las fronteras entre clases de este tipo de clasificadores son como las de la figura

```
% Clasificador K vecinos mas cercanos
K=5;
% se generan caracteristicas de dos dimensiones
L=200;
sigma=[1 0.5];
x0=randn(2*L,2).*(ones(2*L,1)*sigma)+ones(2*L,1)*[-1 1];
x1=randn(2*L,2).*(ones(2*L,1)*sigma)+ones(2*L,1)*[1 -1];
X=[x0; x1];
d=[ones(2*L,1);-ones(2*L,1)]; % etiquetas muestras
% límites de la distribución
desde=-4;hasta=4;
% se dibuja la frontera
plot(x0(:,1),x0(:,2),'o',x1(:,1),x1(:,2),'rx');hold on
for ix=desde:0.2:hasta,
    for iy=desde:0.2:hasta,
        punto=[ix,iy];
        % distancia a todos los puntos
        dist=sum((X-ones(4*L,1)*punto).^2,2);
        % se buscan los mas cercanos
        [aux,ind]=sort(dist);
        clase=sign(sum(d(ind(1:K))));
        if gt(clase,0)
            plot(punto(1),punto(2),'g.')
        end
    end
end
```



Discriminante lineal para distribuciones univariantes

Son aquellos que dividen el espacio en dos semiplanos. En el caso univariante estos clasificadores son del tipo: $y = g(x) = \begin{cases} 1 & si\ x \le x' \\ 0 & resto \end{cases}$

El problema es encontrar $x^* = \arg\min_{x'} P\{g(x) \neq y\}$

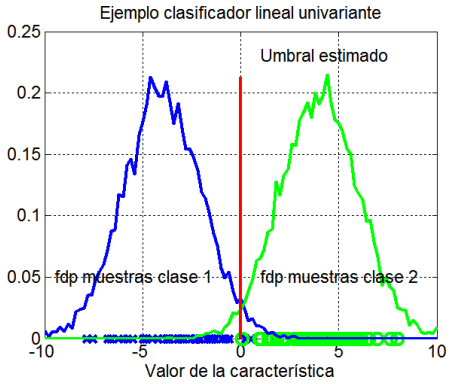
Si se tienen dos distribuciones X_0 y X_1 de medias $m_0 = E\{X_0\}$ y $m_1 = E\{X_1\}$ y varianzas $\sigma_0^2 = E\{(X_0 - m_0)^2\}, \sigma_1^2 = E\{(X_1 - m1)^2\}$, se tiene que:

$x^* = m_0 + \Delta_0 = m_1 - \Delta_1$

Siendo: $\Delta_0 = \dfrac{|m_1 - m_0|\sigma_0}{(\sigma_0 + \sigma_1)}$ y $\Delta_1 = \dfrac{\sigma_1}{\sigma_0}\Delta_0$

Siempre se puede estimar $\hat{m}_0, \hat{m}_1,\ \hat{\sigma}_0,\ y\ \hat{\sigma}_1$ y a partir de ellos obtener una estimación de la cota y $\hat{\Delta}_1,\ y\ \hat{\Delta}_0$ lo cual significa diseñar un clasificador. Ejemplo

```
% clasificadores lineales
% se generan las variables aleatorias
m1=-4;m2=4;sigma1=2;sigma2=2;L=10000;
x1=randn(L,1)*sigma1+m1;x2=randn(L,1)*sigma2+m2;
% se estiman las funciones densidad de probabilidad
x=linspace(-10,10,101);paso=x(2)-x(1);
fdp1=hist(x1,x);fdp1=fdp1./sum(fdp1.*paso);
fdp2=hist(x2,x);fdp2=fdp2./sum(fdp2.*paso);
% se estima el umbral a partir de las medias y varianzas estimadas
m1=mean(x1);m2=mean(x2);
s1=std(x1);s2=std(x2);
a=m1+abs(m2-m1)*s1/(s1+s2);
% se calcula el error
Error=sum([gt(x1,a); lt(x2,a)])/(2*L)*100;
```



Discriminante lineal

Cuando el hiperplano que divide ambos espacios se define como:

$$y = g(x) = \begin{cases} 1 & si \sum_{i=1}^{d} a_i x^{(i)} + a_0 > 0 \\ 0 & resto \end{cases}$$

Una forma de obtener el vector a de forma supervisada es calcular aquel vector que minimiza el error de clasificación definido como $E = (y - \hat{y})^2$ (MSE), el cual se calcula como:

$$\frac{\partial E}{\partial a_i} = 2(y - \hat{y}) \frac{\partial (y - \hat{y})}{\partial a_i} = 2(y - \hat{y}) x^{(i)} = 2(\sum_{j=0}^{d} a_i x^{(j)} - \hat{y}) x^{(i)} = 0 \text{ con } x^{(i)} = 1,\ i = 1,2,...,d\,.$$

Por lo tanto, el vector a se puede calcular como la solución a la ecuación:

$$\sum_{j=0}^{d} a_i x^{(j)} x^{(i)} = \hat{y} x^{(i)}$$

Que es similar a la obtenida en el filtro de Wienner.

Otro procedimiento para obtener un discriminante lineal es suponer que las funciones de distribución de las clases se pueden aproximar por distribuciones normales.

La distribución normal se define como $f(x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} e^{-\frac{1}{2}(x-m)^T \Sigma (x-m)}$

El clasificador lineal óptimo se define como:

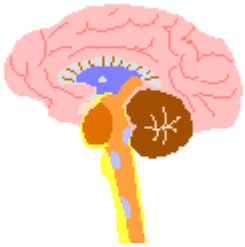
$$y = g^*(x) = \begin{cases} 1 & si\ pf_1(x) > (1-p) f_0(x) \\ 0 & resto \end{cases} = \begin{cases} 1 & si\ a^T x + a_0 > 0 \\ 0 & resto \end{cases}$$

lo cual lleva, con $\Sigma = \Sigma_0 = \Sigma_1$ a:

$$a = (m_1 - m_0)\Sigma^{-1} \text{ y } a_0 = 2\log(\frac{p}{1-p}) + m_0^T \Sigma^{-1} m_0 - m_1^T \Sigma^{-1} m_1$$

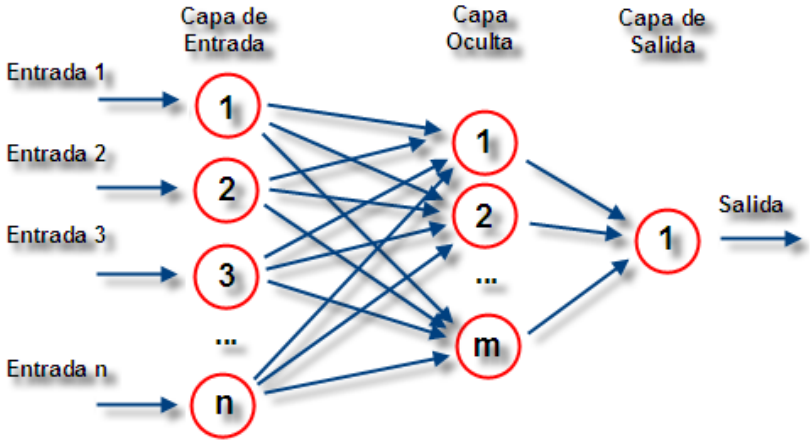
Redes Neuronales

Las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales.

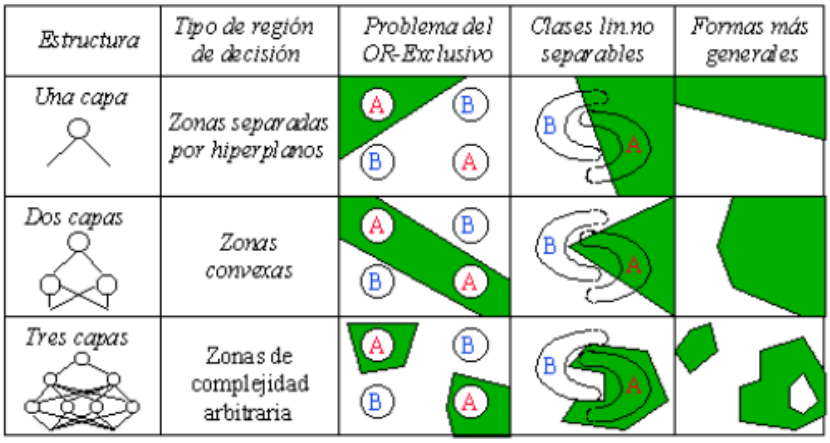


El cerebro tiene una estructura muy apropiada para realizar las tareas de reconocimiento de patrones y asociación con tolerancia al ruido y variabilidad. El cerebro consta de un sistema de interconexión de más de 10 billones de unidades de proceso llamadas neuronas en una red con varios miles de interconexiones por neurona que colaboran para producir un estímulo de salida.

En inteligencia artificial se trata de emular dicho sistema con estructuras denominadas redes de neuronas o redes neuronales.



Cada neurona se modela como una suma ponderada de las entradas a la cual se le aplica una función no lineal de tipo sigmoidal. Ejemplos de las fronteras de decisión obtenidas se muestran en la siguiente gráfica:



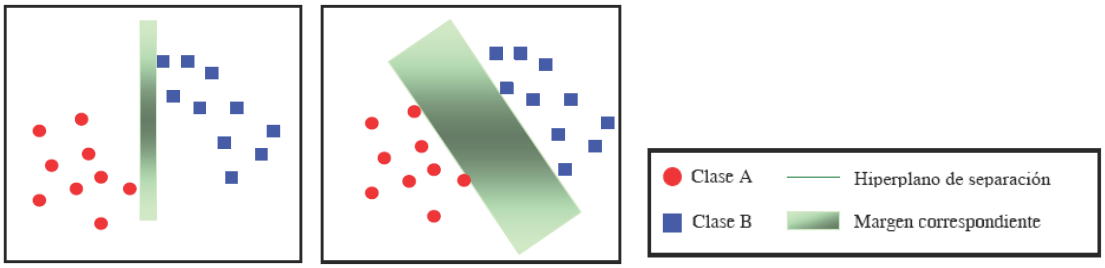
Aunque la corteza cerebral tiene entre 50 y 100 capas, en la práctica no se suelen utilizar más de tres capas.

Máquinas de Vectores Soporte

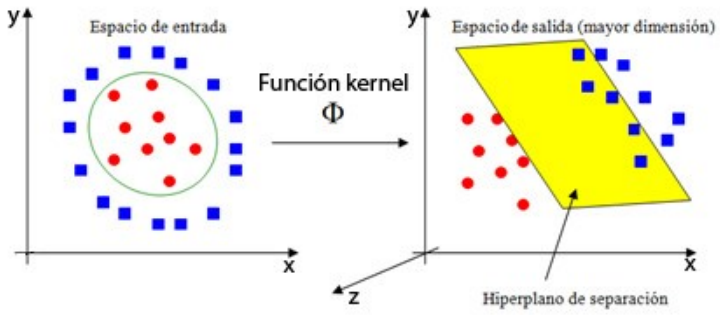
Las máquinas de soporte vectorial o máquinas de vectores de soporte (Support Vector Machines, SVMs) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T.

Un SVM busca un hiperplano que separe de forma óptima los puntos de una clase de la de otra, que eventualmente han podido ser previamente proyectados a un espacio de dimensionalidad superior, incluso infinita.

En ese concepto de "separación óptima" es donde reside la característica fundamental de las SVM: este tipo de algoritmos buscan el hiperplano que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. Por eso también a veces se les conoce a las SVM como clasificadores de margen máximo. De esta forma, los puntos del vector que sean etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.



En el caso de muestras linealmente separables, las dos clases son separables mediante una función lineal que maximice el margen. En el caso de clase no linealmente separables se emplea el truco del kernel, que consiste en mapear los datos a otro espacio Euclideo H (que puede ser de dimensión infinita)



Un kernel muy utilizado es el gaussiano definido: $k(x,y)=e^{-\|x-y\|^2/2\sigma^2}$,

Algoritmos de agrupamiento u autoorganización

Son redes no supervisadas que se entrenan sólo con patrones de entrada buscando características comunes entre los datos de entrada y los agrupa en función de esas características. El algoritmo más característico es el de las K-medias que se implementa como sigue:

Sea un conjunto de datos $x_i, i = 1, \ldots, L$ que se quiere dividir en K agrupaciones.

1. Se define el número de clases o agrupaciones K y se seleccionan como centros de cada agrupación μ_k , k=1,...,K datos de entrenamiento seleccionados aleatoriamente o equiespaciados.
2. Se clasifican los datos de entrenamiento según la regla del vecino más próximo:

   $$x_i \in clase\ k \quad si \quad d(x_i, \mu_k) < d(x_i, \mu_l) \quad \forall l$$

   La medida de distancia da las características comunes con las cuales agrupo.
3. Se recalculan los centros. En el caso de la distancia euclídea, el nuevo centro k es la media aritmética de los datos x_i asignados a la clase k.
4. Se calcula el error: si la disminución del error relativo es menor al umbral se finaliza la iteración; si no, se vuelve al paso 2.

7.7 Problemas de Aula

-Problema 7.1

Las técnicas de mejora de imágenes permiten aumentar la proporción de señal a ruido y acentuar las características de las imágenes modificando los colores o las intensidades de una imagen.

Dadas las siguientes imágenes en escala de grises:

```
pout = imread('pout.tif');
tire = imread('tire.tif');
```

Comprobar la eficiencia de los siguientes métodos:

- **imadjust:** incrementa el contraste en la imagen mapeando los valores de intensidad de entrada a unos nuevos valores. Por ejemplo, un 1% de los datos son saturados a niveles bajos y altos de intensidad.
- **histeq:** ecualiza el histograma y da un peso similar (uniforme) a todos los niveles de gris.
- **adapthisteq:** se ecualiza el histograma pero sobre regiones en lugar de sobre toda la imagen.

- Con imadjust:

```
pout_imadjust = imadjust(pout);
subplot(211);imagesc(pout)
subplot(212);imhist(pout)
subplot(211); imagesc(pout_imadjust);
subplot(212);imhist(pout_imadjust)
```

- con histeq

```
pout_histeq = histeq(pout);
subplot(211); imagesc(pout_histeq);
subplot(212);imhist(pout_histeq)
```

- con adapthisteq

```
pout_adapthisteq = adapthisteq(pout);
subplot(211); imagesc(pout_adapthisteq);
subplot(212);imhist(pout_adapthisteq)
```

Repetir con la otra imagen.

- Problema 7.2

El contraste de las imágenes en color también puede ser mejorado o adaptado según unas necesidades. Lo que se suele hacer es transformar una imagen de espacios de color como el RGB a un nuevo espacio de color como por ejemplo el Lab, donde la componente L contiene la información de luminosidad. Si mejoramos el contraste L, mejoraremos el contraste sin que ello afecte al color original.

Según la imagen:

```
[X map] = imread('shadow.tif'); %Las imágenes .tif codifican el color
aparte por lo que es necesario usar map
shadow = ind2rgb(X,map); % convertimos a espacio color rgb
```

Mejore su contraste de acuerdo a las técnicas vistas en el apartado anterior:

```
% mapas de conversion
srgb2lab = makecform('srgb2lab');
lab2srgb = makecform('lab2srgb');
% convertimos a L*a*b*
shadow_lab = applycform(shadow, srgb2lab);
% el valor de luminosidad varía entre 0 y 100.
max_luminosity = 100;
L = shadow_lab(:,:,1)/max_luminosity;

% sustituimos la nueva luminosidad y convertir al espacio RGB
shadow_imadjust = shadow_lab;
shadow_imadjust(:,:,1) = imadjust(L)*max_luminosity;
shadow_imadjust = applycform(shadow_imadjust, lab2srgb);

shadow_histeq = shadow_lab;
shadow_histeq(:,:,1) = histeq(L)*max_luminosity;
shadow_histeq = applycform(shadow_histeq, lab2srgb);

shadow_adapthisteq = shadow_lab;
shadow_adapthisteq(:,:,1) = adapthisteq(L)*max_luminosity;
shadow_adapthisteq = applycform(shadow_adapthisteq, lab2srgb);
```

-Problema 7.3

Para el procesado de imágenes interesa tener una iluminación uniforme pero esto no es siempre posible. Dada la imagen:

```
I = imread('rice.png');
imagesc(I)
```

Realizar un pro-procesado que elimine en la medida de lo posible el efecto indeseado producido por la iluminación no uniforme.

```
%Estimamos el fondo a través de la operación morfológica open, también se
puede hacer con un filtrado paso bajo
background = imopen(I,strel('disk',15));
imagesc(background)
%Ahora solo tenemos que restar el fondo estimado a la imagen de interés
I2 = I - background;
imagesc(I2)
```

- Problema 7.4

La detección de bordes es una herramienta comúnmente utilizada en el pre-procesado de imágenes. Sus aplicaciones son múltiples pero cobran especial interés en imágenes donde se quieran detectar objetos situados sobre un fondo uniforme.

Dada la imagen:

```
I = imread('cell.tif');
imagesc(I)
```

Detectar la célula presente en la imagen:

En un primer paso detectaremos los bordes de la imagen, en este caso optamos por el operador de sobel (probar con otros):

```
[~, threshold] = edge(I, 'sobel');
%fudgeFactor es un factor nos permite ajustar el umbral a la imagen en
cuestión, probar con varios para ver su efecto
for fudgeFactor=0.1:0.1:0.9
   BWs = edge(I,'sobel', threshold * fudgeFactor);
   imagesc(BWs)
   pause(2);
end
```

El fondo uniforme hace fácilmente distinguible objetos no uniformes como la célula. El siguiente paso consiste en dilatar la imagen para enfatizar aún más esas regiones no uniformes.

```
BWsdil = imdilate(BWs,ones(5));
imagesc(BWsdil);
```

Y se rellenan los huecos o espacios para así conseguir un objeto cerrado.

```
BWdfill = imfill(BWsdil, 'holes');
imagesc(BWdfill);
```

Se eliminan objetos situados en el borde de la imagen y suavizamos los bordes para obtener un contorno más limpio.

```
BWnobord = imclearborder(BWdfill, 4);
imagesc(BWnobord)
BWfinal = imerode(BWnobord,ones(3));
imagesc(BWfinal)
```

Ahora solo tenemos que calcular el perímetro del objeto binario y superponerlo a la imagen original.

```
BWoutline = bwperim(BWfinal);
Segout = I;
Segout(BWoutline) = 255;
imagesc(Segout)
```

- Problema 7.5

Los diferentes objetos en una imagen binaria se pueden etiquetar de acuerdo a sus características. Algunas de las más habituales son: centroide, área, orientación, excentricidad, perímetro,…

Dada la siguiente imagen:

```
BW = imread('text.png');
imagesc(BW)
```

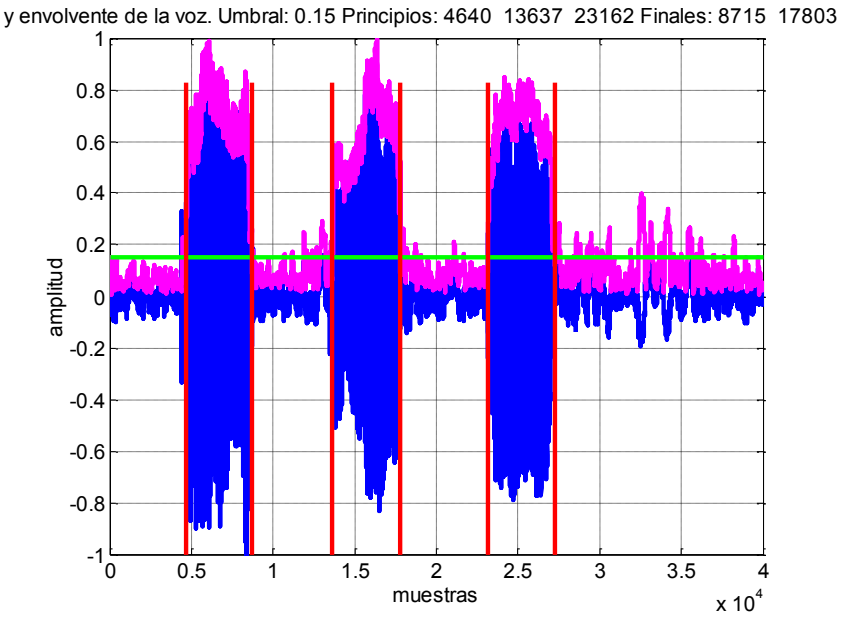
Etiquete los components conectados, compute sus centroides y superponga la localización de éste en la imagen.

```
s =regionprops(BW, 'centroid');
centroids = cat(1, s.Centroid);
imshow(BW)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
```

- Problema 7.6

En este problema se trata de utilizar los cepstrum para realizar un reconocedor de voz sencillo. Para ello realice las siguientes tareas:

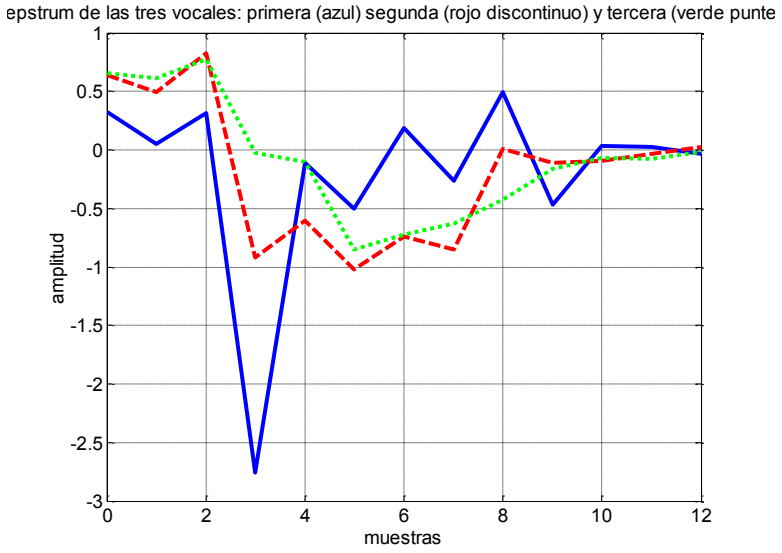
a) Grabe una frase con tres vocales diferentes separadas. Calcule el principio y fin de cada vocal. Represente gráficamente la grabación con los principios y fin de cada vocal. Escuche la grabación.



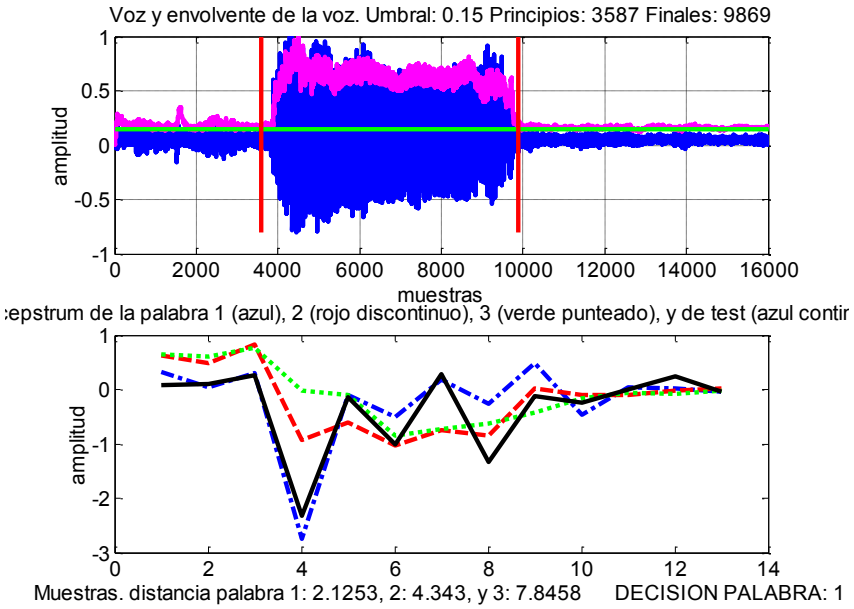
b) Para cada vocal aislada calcule sus parámetros mediante el siguiente proceso:

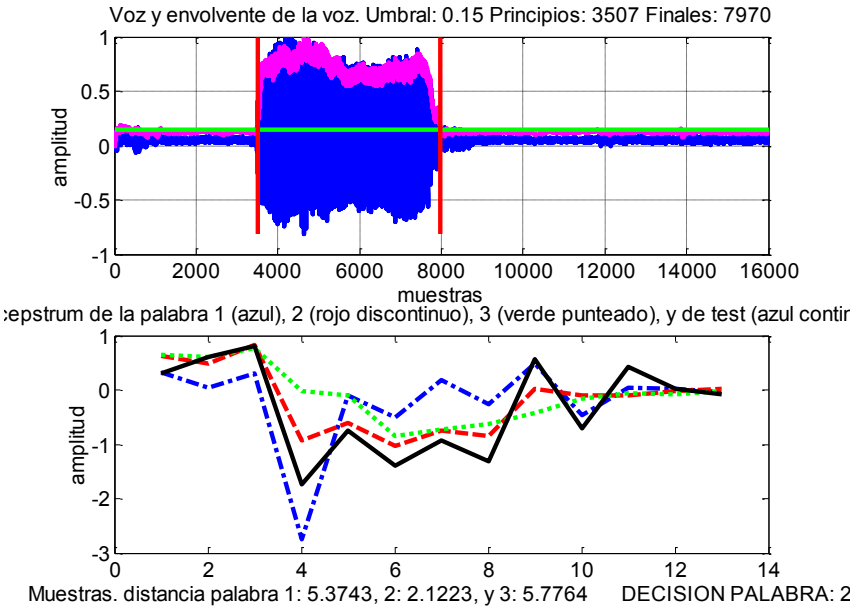
1. Elimine la media y filtre la voz por un filtro de paso alto de un solo coeficiente para allanar el espectro.
2. Divida la voz en tramas de unos 20 mseg de duración y solapadas unos 10 mseg aproximadamente.
3. De cada trama, calcule los coeficientes cepstrum con la función *rceps* de Matlab. Tenga en cuenta que habitualmente el primer término suele desecharse pues corresponde a la energía, siendo de gran variabilidad incluso entre fonaciones de un mismo locutor. Así mismo, los últimos cepstrum no tienen información del tracto vocal, principal característica diferenciadora entre palabras. Por lo tanto, escoja como vector de parámetros cepstrum de cada trama unos 13 cepstrum a partir del segundo. Habitualmente, estos parámetros cepstrum escogidos se enventanan con una ventana de hamming.

4. Como parámetros de vocal, calcule el vector medio de los vectores de parámetros cepstrum de cada trama
5. Dibuje los parámetros cepstrum promedio de cada vocal



c) Grabe de nuevo una de las vocales pronunciadas. Aísle la vocal, calcule sus parámetros cepstrum. Calcule la distancia euclídea de los parámetros de esta palabra a los parámetros de cada una de las palabras. A partir de las medidas de distancia obtenidas, decida por proximidad (mínima distancia) cual ha sido la palabra pronunciada. Verifique el correcto funcionamiento con las tres palabras. Represente la palabra grabada con su principio y fin, sus parámetros, distancias y decisión.

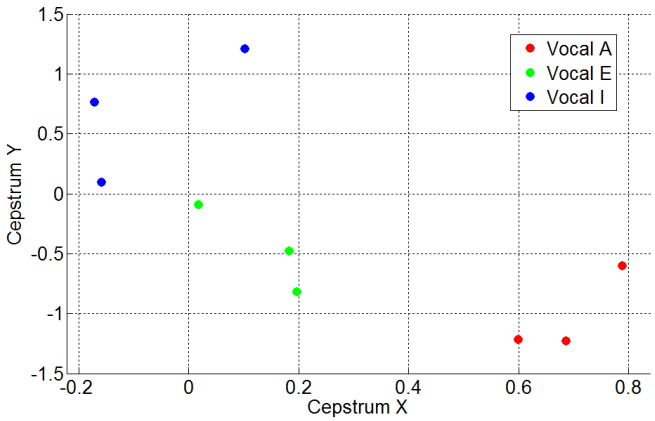




d) Los algoritmos de clasificació permiten implementar metodologías de aprendizaje. Estos algoritmos se basan en una información de entrenamiento a partir de la cual modelar los patrones deseados. En apartados anteriores hemos visto una clasificación basada en distancia. En el siguiente apartado implementaremos el algoritmo de aprendizaje basado en los k-vecinos más cercanos o KNN.

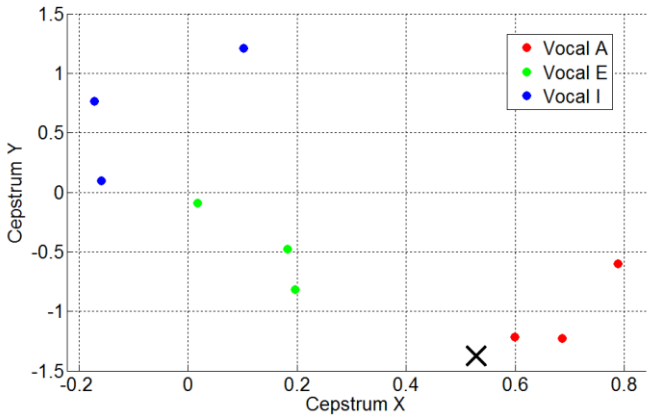
En primer lugar se cargan los ceptrums de entrenamiento correspondientes a las vocales a, e, i. Se representan las distintas clases (3 clases, 3 vocales) con sus respectivas muestras (3 muestras por vocal) en base a dos ceptrums. Esto permite visualizar a modo de ejemplo las clases en una gráfica bidimensional, aunque posteriormente se utilizarán los 13 ceptrums disponibles para realizar la clasificación.

```
%Cargo los ceptrum correspondientes a las vocales a, e, i tanto de
entrenamiento como de test
load mceps_matrix
%Utilizo 2 cepstrum para poder representarlo en 2 dimensiones
ceps=[1 2];
gscatter(mceps(:,ceps(1)),mceps(:,ceps(2)),etiquetas')
set(legend,'location','best')
grid on
```



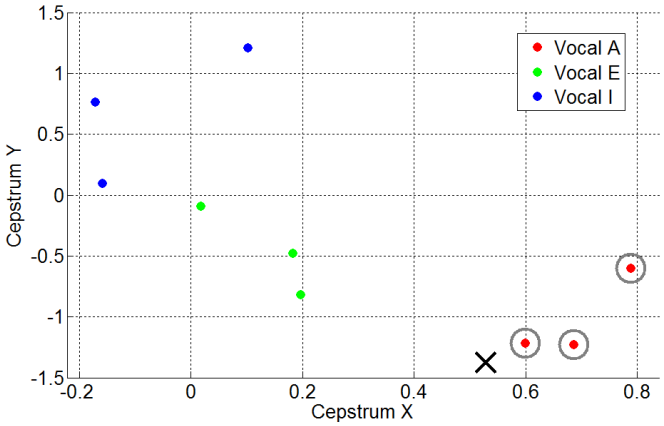
El siguiente paso es cargar el patrón a clasificar. Este patrón no se sabe a qué vocal pertenece y se pretende que el sistema lo reconozca automáticamente. Nuevamente se representamos dos ceptrums para ilustrar el problema de clasificación.

```
newpoint = mcepst(ceps(1):ceps(2));
line(newpoint(1),newpoint(2),'marker','x','color','k',...
   'markersize',10,'linewidth',2)
```



Llegados a este punto se clasifica el nuevo patrón y se busca la clase a la que pertenece en función de k=3 vecinos más cercanos. Ahora si se emplean los 13 ceptrums para clasificar las diferentes muestras (aunque a odo ilustrativo se siguen representando en pantalla los resultados en base a solo dos de los ceptrums).

```
%%% Utilizo los 13 cepstrum para clasifiar, no 2
[n,d] = knnsearch(mceps(:,1:13),mcepst(1:13),'k',3)
line(mceps(n,ceps(1)),mceps(n,ceps(2)),'color',[.5                .5
.5],'marker','o',...
    'linestyle','none','markersize',10)
tabulate(etiquetas(n))
```



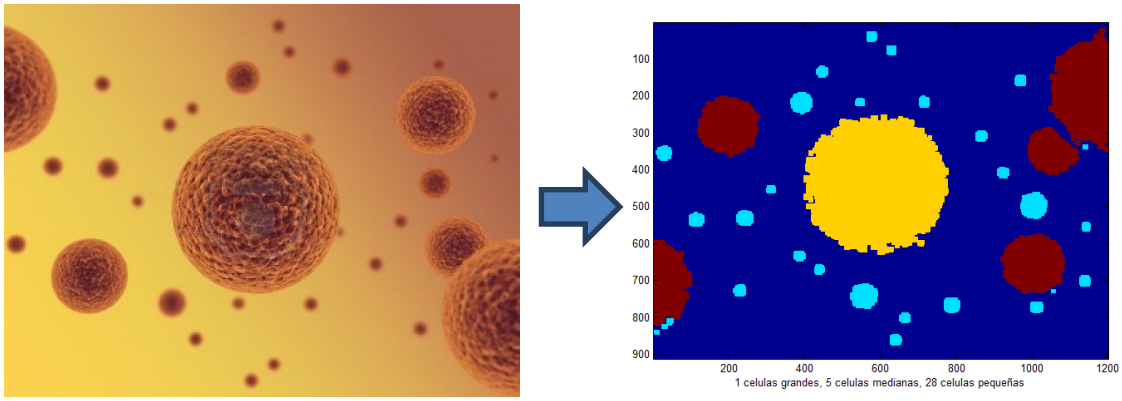
7.8 Ejercicios Prácticos

Diseño de clasificador de células de una imagen biológica

En esta práctica se van a desarrollar técnicas automáticas para contar el número de células de una imagen biológica. Existen una gran cantidad de técnicas diferentes, pero se van a recoger las más básicas con la finalidad de entender en entorno de trabajo en procesado de imágenes. Hay que tener presente para el desarrollo de esta práctica, que el reconocimiento de imágenes no es tan solo una la aplicación de técnicas, sino un arte, en cuanto a la propia limitación de la inteligencia artificial. No hay una técnica que resuelva todas las dificultades, sino que hay que saber cuándo aplicar dicha técnica u otra, en función del caso que se plantea.

El método que se propone para contar las células es sencillo: leer la imagen en color propuesta, pasarla a niveles de gris y realizar la segmentación de la imagen en gris para luego clasificarlas en tres tamaños: pequeñas, medianas y grandes. Para ello se aconseja caracterizar cada célula con descriptores de tamaño y utilizar un clasificador.

La salida debe ser una imagen en la que cada célula tenga un color según su clasificación. Cuenta cuantas células hay de cada tamaño.



Otras imágenes con la cuales también puede trabajar para clasificar y contar.