UNIVERSIDAD AUTÓNOMA METROPOLITANA IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

SISTEMA PARA PROCESAMIENTO DIGITAL DE SEÑALES BIOMÉDICAS

TESIS PARA OBTENER EL GRADO DE
MAESTRÍA EN INGENIERÍA BIOMÉDICA

VERÓNICA MEDINA BAÑUELOS

ASESOR: M. EN C. MIGUEL CADENA MÉNDEZ

México, D. F., noviembre 1987

SINODALES

M. en C. Miguel Cadena Méndez

Dr. Fernando Prieto Hernández

M. en C. Caupolicán Muñoz Gamboa

AGRADECIMIENTOS

A Joaquín, por su apoyo y ayuda durante todo el proyecto.

A Miguel y Caupolicán, por sus valiosos consejos y sugerencias.

A Javier, por ayudarme con la impresión del trabajo.

A todos los amigos que me ofrecieron su apoyo incondicional y que muchas veces resolvieron mis dudas.

PROLOGO

El procesamiento de señales es un campo de las matemáticas que ha tenido un enorme crecimiento en los últimos 20 años. Su aplicación en diversas áreas ha proliferado, principalmente por el rápido avance de la computación, que ha permitido utilizar las técnicas del procesamiento digital de señales en un tiempo corto, a costos accesibles. Este trabajo tratará de la aplicacion del procesamiento digital a las señales biomédicas, que ha sido fuente de cambios importantes en los métodos tradicionales de análisis y obtención de parámetros y que ha proporcionado nuevos elementos para la evaluación del funcionamiento de diversos sistemas del cuerpo humano.

A pesar de las ventajas que proporciona el procesamiento de señales biomédicas, el empleo de equipos dedicados a este fín está aún muy restringido, principalmente por su elevado costo, que los convierte en instrumentos de lujo. El presente trabajo fue iniciado como un intento por facilitar la creación de sistemas de procesamiento digital de señales biomédicas a bajo costo; concretamente, se trata de un proyecto de infraestructura: un conjunto de programas instalado en el marco básico de una microcomputadora de bajo costo y con grandes capacidades de expansión, cuyo principal objetivo es la adquisición, manejo y procesamiento de señales biomédicas.

Las características más importantes con que cuenta este paquete son las siguientes:

1. Está diseñado siguiendo una estructura de manejo fácil de los datos, que permite adquirir, desplegar y procesar hasta cuatro señales diferentes dentro del mismo programa.

2. Cuenta con la posibilidad de mostrar gráficamente las señales, antes y después del procesamiento, con varias opciones de despliegue y expansión de las escalas.

3. Es totalmente compatible con una tarjeta de adquisición de datos, diseñada especialmente para señales biomédicas.

4. Ofrece varias funciones de procesamiento, seleccionables a través de menú: convolución, correlación, transformada de Fourier, filtrado y densidad espectral.

5. El lenguaje de programación empleado facilita el manejo del paquete en forma de módulos, que pueden ser transferidos fácilmente para la creación de programas de propósitos más específicos.

Como complemento, este informe contiene la documentación necesaria para facilitar el manejo y expansión del paquete, así como las pruebas realizadas y los resultados obtenidos hasta el momento. Está estructurado de la siguiente forma:

- En el capítulo I se presentan los motivos por los que fue desarrollado este paquete, incluyendo las ventajas que proporciona el procesamiento digital de señales, las diversas opciones para la realización de éste y algunos paquetes que han sido desarrollados con este fín. En la última sección se comenta sobre el contexto en el que fue desarrollado este sistema.

- En el capítulo II se detalla los objetivos que se persiguieron en el desarrollo de este proyecto.

- El capítulo III contiene una descripción de los criterios seguidos en la definición general del paquete. Posteriormente se incluye toda la información del desarrollo de cada una de las rutinas empleadas, divididas en dos bloques: manejo de datos y procesamiento de señales. Para cada una de las rutinas de procesamiento se utilizó el siguiente formato: una introducción sobre la importancia del algoritmo y las diferentes formas de llevarlo a cabo; la teoría necesaria para el método elegido en este paquete, siguiendo los criterios generales; la estructura interna de cada la rutina; un ejemplo del uso de la misma.

- En el capítulo de resultados (IV) se incluyen pruebas de evauación de todos los algoritmos, sometidos a señales conocidas; esta etapa es especialmente importante para asegurar la validez de los resultados que entrega el paquete. Finalmente, se describe la experiencia del uso de este programa en el desarrollo de un sistema de propósito específico para el análisis espectral de señales electroencefalográficas.

INDICE

I. INTRODUCCION

A. PROCESAMIENTO DIGITAL VS. PROCESAMIENTO ANALOGICO

El campo de aplicación del procesamiento digital de señales se ha extendido enormemente en los últimos años. Aunque las bases teóricas de este tema fueron establecidas en el siglo XVII, no fue sino hasta los últimos 20 años, con el advenimiento de la última generación de computadoras, que fue posible diseñar eficientemente los algoritmos para el procesamiento de alguna señal y construir equipos rápidos dedicados a este fín. Actualmente, las aplicaciones del procesamiento digital de señales incluyen campos tan diversos como imágenes, señales biomédicas y señales sismográficas (Oppenheim, 1978).

Las ventajas que ofrece el procesamiento digital de señales comparado con el procesamiento analógico son múltiples (Chen, 1979). La transmision y procesamiento de una señal representada en código binario es casi inmune al ruido. Por lo tanto, el procesamiento de señales digitales es más confiable que el de señales analógicas; esta CONFIABILIDAD puede además incrementarse empleando códigos de detección y corrección de errores. Una aplicación en la que puede apreciarse esta ventaja es en la transmisión de imágenes a grandes distancias.

La exactitud con la que puede ser diseñado un sistema digital es mayor, comparada con la de un sistema analógico, que se vé limitada por factores externos. Este problema se observa claramente en el diseño de filtros analógicos óptimos, en donde se requiere de resistencias cuyos valores no están disponibles comercialmente. Al ajustarse a valores comerciales, el comportamiento real del filtro se aleja del diseño original. Por otro lado, la EXACTITUD en un sistema digital está limitada principalmente por el número de bits empleados en la codificación de una señal analógica y por la frecuencia de muestreo de la misma.

Otra ventaja del procesamiento digital es que permite emplear una misma unidad digital para procesar varias señales. Por ejemplo, si en una línea telefónica la voz se muestrea a una velocidad de 8000 muestras/seg, codificada en 8 bits, el canal de voz se transmitirá a 64,000 bits/seg. Sin embargo, los circuitos integrados actuales permiten transmitir a una velocidad de hasta 1,544,000 bits/seg, o sea, 24 canales de voz por una misma línea telefónica. Este proceso se denomina MULTICANALIZACION EN TIEMPO y puede realizarse solamente con señales digitales.

Las técnicas de procesamiento digital actuales han facilitado grandemente el análisis de una señal o de un sistema en el DOMINIO DE LA FRECUENCIA; para este fín existe una gran variedad de equipos de análisis espectral para múltiples aplicaciones.

Además de las ventajas mencionadas, un sistema digital puede ser repetido tantas veces como se requiera y su funcionamiento será idéntico; las señales digitales pueden ser almacenadas por tiempo indefinido sin pérdida de exactitud; los datos pueden almacenarse y recuperarse fácil y rápidamente. Finalmente, las técnicas digitales son flexibles en su diseño, de tal manera que, por ejemplo, un filtro puede ser modificado rápidamente empleando un conjunto diferente de coeficientes en su respuesta a impulso.

Estas ventajas han hecho que el procesamiento digital de señales se prefiera definitivamente sobre el analógico.

B. OPCIONES PARA LA REALIZACION DEL PROCESAMIENTO DIGITAL

Una vez que se ha elegido trabajar con señales digitales, se requiere contar con un sistema que acepte una señal analógica en la entrada, la procese digitalmente y entregue un resultado confiable.

Existen principalmente dos opciones para lograr este objetivo:

1. La adquisición de un sistema completo de procesamiento, que incluya la estructura y la programación necesarias para la ejecución de funciones definidas. Actualmente existen en el mercado varios sistemas de este tipo: Hewlett Packard, PDP, Tektronix, entre otros. Sin embargo, esta opción presenta principalmente tres desventajas:

   a) Los algoritmos están desarrollados específicamente para la ferretería propia del sistema y no se pueden aplicar fácilmente a otro sistema que pudiera ofrecer más ventajas. Es decir, es una herramienta totalmente "cerrada" para el usuario.

   b) Generalmente, la documentación que acompaña al sistema es muy limitada. Esto constituye un fuerte problema desde el punto de vista de "confiabilidad" de los algoritmos, puesto que el usuario recibe un resultado final, sin posibilidad de seguir las etapas intermedias.

   c) El costo de estos sistemas es demasiado alto, lo que los convierte en "instrumentos de lujo".

2. La adquisición de un sistema que ofrezca una infraestructura básica, en lo que a ferretería y programación se refiere, que permita un fácil acceso y adaptación para múltiples aplicaciones. Específicamente, se puede mencionar la construcción de sistemas de medición de diversas variables analógicas, a partir de un marco básico de circuitos. Actualmente, los sistemas que ofrecen las cualidades de flexibilidad y accesibilidad son las microcomputadoras llamadas personales, que pueden ser adquiridas a un bajo costo (1,500 a 3,000 dólares) (Byte, 1984). En cuanto a la programación de dichos sistemas se genera, a su vez, la siguiente alternativa:

   a) El empleo de programación comercial, desarrollada específicamente para el tipo de microcomputadora. La principal desventaja que se presenta al elegir esta opción es, nuevamente, la información limitada con la que cuenta el usuario.

   b) El desarrollo de la programación propia para las necesidades específicas de cada aplicación, mediante la instalación de algoritmos conocidos, debidamente probados y documentados. Esta opción, además de proporcionar la solución adecuada a cada aplicación, genera confianza en la interpretación y análisis de los resultados.

De los caminos mencionados, resulta obvio que la última opción proporciona confianza en los resultados obtenidos y flexibilidad para diversas aplicaciones, a la vez que permite tener experiencia y facilidad para su aplicación en el procesamiento de diversos tipos de señales.

C. ANTECEDENTES

Se han realizado varios paquetes de programación para el procesamiento digital de señales en diversas instituciones, tanto en el extranjero, como nacionales. Algunos de éstos han sido adecuados a las necesidades específicas de cada grupo de trabajo, dependiendo de la infraestructura que se tiene y del tipo de procesamiento que se quiere realizar. En la mayoría de los casos, estos paquetes no han sido desarrollados con el fín de proporcionar una herramienta flexible y de fácil empleo, para que sea utilizada por un mayor número de personas. Asimismo, existen varios paquetes disponibles comercialmente, para usos más generales, pero que presentan algunas de las desventajas ya mencionadas. De todos estos paquetes, conviene mencionar los que han sido desarrollados para una infraestructura común y de bajo costo, puesto que esta característica los hace más accesibles. Particularmente, me refiero a las microcomputadoras compatibles con IBM-PC, que aparentemente estarán vigentes durante varios años en el ámbito nacional, por sus capacidades, costo y fácil adquisición.

Actualmente, existen varios paquetes disponibles comercialmente, que realizan diversas funciones básicas y especiales de procesamiento. Por ejemplo, la compañía Data Physics Corporation ha lanzado el paquete SIGNALCALC (DPC, 1986), para emplearse en microcomputadoras de características similares a la IBM-PC; entre las funciones que realiza se tiene: operaciones aritméticas, análisis espectral, correlación, convolución y diferentes opciones para el despliegue de las señales. El costo del paquete es superior a los 2,000 dólares.

La compañía Signal Technology ha creado varios programas de diferente complejidad, para emplearse en microcomputadoras compatibles con IBM-PC (XT o AT) (Signal Technology, 1986); entre las funciones más importantes que realiza el paquete mas completo (ILS-PC II) se encuentran: análisis espectral, operaciones aritméticas, filtrado digital, calculos estadísticos, convolución y correlación. El costo aproximado de este paquete es de 3,000 dólares.

A nivel institucional, se han desarrollado varios paquetes para satisfacer necesidades particulares en el procesamiento de señales específicas. El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), en un intento por normalizar y reunir la gran diversidad de paquetes desarrollados, publicó en 1975 un conjunto de algoritmos y programas, desarrollados en lenguaje Fortran por varios autores. Este documento constituye una excelente referencia, y contiene una revisión muy completa de los algoritmos desarrollados hasta el momento de su publicación (IEEE, 1975). En la Universidad de Purdue (Indiana) se ha desarrollado un conjunto de programas para el procesamiento de potenciales evocados, que puede ser empleado con fínes más generales (McGillem, Auñón, 1981). Este trabajo constituyó la base para el desarrollo del paquete que se presenta, en cuanto a las rutinas que se instalaron y la elección de los algoritmos. El lenguaje de programación del trabajo de Auñón (Fortran) fue sustituido por un lenguaje más estructurado, y se mejoró la configuración propia del paquete, con el fín de tener una mayor flexibilidad en el manejo y presentación de los resultados.

Finalmente, se han diseñado varios programas para la realización de funciones especiales, como filtrado digital de una señal. En la Universidad de Washington (Missouri) se ha desarrollado un paquete para el diseño de filtros digitales, de varios tipos, cuya estructura permite probar diferentes métodos de diseño, en una forma fácil y amable para el usuario (Yeung, Kim, 1986). En el área de Ingeniería Electrónica de la UAM-I se ha desarrollado un programa similar para diseño de filtros de respuesta a impulso infinita, instalado también en una microcomputadora compatible con PC (Villaseñor, 1986).

D. PLANTEAMIENTO DEL PROBLEMA

El Laboratorio de Investigación en Computación y Procesamiento de Señales (LICPOS) del área de Ingeniería Biomédica de la UAM-Iztapalapa fue creado en el año de 1983, con el objetivo de diseñar y desarrollar equipos de adquisición y procesamiento de señales biomédicas, basados principalmente en tecnología digital. En primer lugar, se definieron dos grandes tipos de proyectos que debían ser realizados, para conseguir el objetivo final:

a) Proyectos de infraestructura, cuya principal finalidad es dotar de las herramientas necesarias para facilitar el objetivo planteado inicialmente.

b) Proyectos de aplicación, que son propiamente equipos construídos empleando las herramientas del punto anterior, y que son creados para la adquisición o análisis de una señal de interés particular.

Dentro de los proyectos de infraestructura se presentó la necesidad de contar con un sistema de procesamiento digital de señales biomédicas, que constituyera un apoyo para el desarrollo de instrumentos de aplicación específica, que involucran el análisis espectral y filtrado de señales biomédicas, principalmente (Pronaes, 1983).

Inicialmente, se contaba con el apoyo de un sistema marca Tektronix-Digital (PDP-11) dedicado al procesamiento de señales; este sistema maneja varios arreglos de datos simultáneamente y contiene diversas funciones de procesamiento de señales (integración, derivación, transformada de Fourier, convolución y correlación entre las más importantes). La principal desventaja que presenta este sistema es su incompatibilidad con otros equipos y la documentación insuficiente que, además de limitar sus aplicaciones, impide extender su uso para la elaboración de otros sistemas.

Una solución alternativa para resolver el problema consistía en la adquisición de paquetes comerciales de programas para su instalación en un sistema más compatible. Estos paquetes presentan nuevamente la desventaja de impedir su expansión y aplicación a otros programas, como ya se mencionó en la sección anterior.

Estos problemas condujeron a la necesidad de elaborar un paquete de programas propio, instalado en un sistema de fácil adquisición, que diera la experiencia para permitir su inmediata aplicación y extensión, y que además fuera suficientemente flexible, para los fines del laboratorio. De acuerdo a estas necesidades, y considerando las ventajas y desventajas de los paquetes descritos en la sección anterior, se definieron los objetivos para la realización del sistema.

II. OBJETIVOS

Diseño e instalación de un paquete de programación para el procesamiento digital de señales biomédicas, con las siguientes características:

A. Diseño interactivo con el usuario, capaz de procesar una o varias señales almacenadas en disco, con los siguientes algoritmos, seleccionables a través de menú:

   1) Convolución
   2) Transformada rápida de Fourier
   3) Filtrado digital
   4) Densidad espectral
   5) Correlación

B. Presentación gráfica de resultados, con facilidad de autoescalamiento en forma lineal, tanto para las señales de entrada como para las señales ya procesadas.

C. Utilización de un lenguaje de programación que facilite la documentación y el acceso a cualquier usuario. Asimismo, que proporcione la capacidad de expansión para la instalación de otros algoritmos de procesamiento.

D. Instalación en una microcomputadora de bajo costo, con gran capacidad de expansión y compatibilidad con otros sistemas.

E. Compatibilidad con una tarjeta de conversión analógica-digital, diseñada para la adquisición de señales biomédicas, siguiendo un criterio similar al de este conjunto de programas.

III. DESARROLLO

A. CRITERIOS DE DISEÑO

Antes de iniciar el proyecto, fue necesario definir algunas características que fueran las más adecuadas para el cumplimiento de los objetivos planteados; por ejemplo, el tipo de computadora en el que se debía instalar; el lenguaje de programación que facilitara la extensión o creación de otros programas y la estructura del paquete que permitiera una fácil interacción con el usuario.

Como se mencionó, uno de los objetivos que se persiguió en la realización de este paquete fue contar con un instrumento de bajo costo, altamente compatible con varios tipos de computadoras. En estos momentos, las que brindan ambas ventajas son las microcomputadoras compatibles con IBM-PC. El costo actual de estas microcomputadoras es de alrededor de 1,500 dólares y, por lo tanto, se han hecho accesibles a un número cada vez mayor de personas. Estas cuentan con interfases suficientes para tener compatibilidad con muchos otros sistemas. Por esto, y por contar con varias máquinas de este tipo en el laboratorio, se eligió instalar el paquete desarrollado en una computadora personal marca Printaform.

En cuanto al lenguaje de programación, el paquete fue desarrollado totalmente en lenguaje Pascal, excepto las rutinas de manejo del convertidor analógico/digital para la adquisición de datos, que fueron escritas en lenguaje ensamblador. El uso de Pascal resolvió el problema de contar con un lenguaje estructurado, fácilmente entendible y que pudiera permitir el manejo de las diferentes rutinas independientemente, para su posterior inclusión en otros programas.

Una de las características más importantes de este conjunto de programas es la estructura para el manejo de la información que presenta. El usuario puede elegir diversas opciones de adquisición, presentación y procesamiento de datos y manejar separadamente varias señales, dentro del mismo programa. Esta estructura fue definida tomando como base al sistema Digital/Tektronix (PDP-11), que es bastante cómodo de manejar en este aspecto (Digital, 1975); en este sistema se manejan hasta cuatro señales diferentes, con una longitud de 512 datos cada una. El manejo de señales de esta forma es bastante flexible, y la longitud de los arreglos es muy conveniente, particularmente para la presentación de la información y para el cálculo de transformadas de Fourier, como se verá posteriormente.

El resultado de algunas funciones incluídas en este paquete como convolución y correlación, generalmente excede los 512 puntos; por ésto, se incluyó otro arreglo de datos de la longitud máxima para estas funciones (1024 puntos), que además es empleado para depositar el resultado de todas las operaciones de procesamiento de una señal. Por otro lado, algunas de las funciones de procesamiento (transformada de Fourier y densidad espectral) son inherentemente complejas; por ésto, fue necesario definir todos los datos de cada arreglo como variables complejas, en donde las partes real e imaginaria se manejan como números reales, en forma de una mantisa y un exponente; de esta manera todas las operaciones pueden ser realizadas en el formato real, y el resultado puede tener o no parte imaginaria. En algunos casos es conveniente analizar la magnitud de una señal, por lo que fue necesario incluir una opción que efectuara esta operación. Como el lenguaje de programación usado no maneja aritmética de números complejos, hubo que definir las operaciones de suma, resta y multiplicación para este tipo de variables.

Finalmente, las funciones de procesamiento que se incluyeron fueron elegidas considerando los paquetes existentes comercialmente, y los algoritmos empleados más comúnmente para el análisis de una señal. Para esta elección, se tomó como base el trabajo publicado por McGillem (McGillem, Auñón, 1981).

En resumen, la estructura del programa permite manejar cuatro arreglos de datos para el usuario (512 datos complejos cada uno) y un arreglo de resultados (1024 complejos). Cada arreglo de datos puede manejarse independientemente, lo que permite realizar diferentes funciones en las señales contenidas en cada arreglo. Los resultados de todas las funciones son depositados siempre en el arreglo de resultados, de tal manera que la señal original no se pierde.

El paquete fue estructurado en forma de menús, para facilitar el seguimiento de cada una de las rutinas. El menú principal está constituido basicamente por dos bloques de rutinas: manejo de datos y análisis de una señal.

1. MANEJO DE DATOS. Este conjunto constituye la infraestructura necesaria para el movimiento y despliegue de los arreglos de datos. Se tiene las siguientes opciones:
   a. Adquisición
   b. Intercambio
   c. Normalización
   d. Cálculo de magnitud
   e. Lectura y almacenamiento en disco
   f. Despliegue en pantalla
   g. Graficación en papel

2. PROCESAMIENTO Y ANALISIS DE UNA SENAL. Contiene las rutinas para la realización de varias funciones de procesamiento:

   a. Convolución
   b. Transformada de Fourier
   c. Filtrado Digital
   d. Densidad espectral
   e. Correlación

La figura 3.1 muestra la estructura general del programa, y las rutinas y área de manejo del usuario. En las siguientes secciones se describe detalladamente el funcionamiento de cada una de las rutinas. El listado del programa completo se encuentra en el Apéndice B. La definición de variables y la estructura del lenguaje empleado permiten que el programa se explique por sí solo.

B. RUTINAS DE MANEJO DE DATOS

1. RUTINA DE ADQUISICION

a. Introducción

Existen dos formas de introducir datos al programa para su procesamiento: lectura del disco o adquisición directa del exterior a través de un cable que se conecta a la tarjeta de conversión analógica/digital. Esta segunda opción permite obtener hasta cuatro señales, que se almacenan en cada uno de los arreglos de datos, a una frecuencia de muestreo variable.

Existen varios puntos que deben considerarse al digitalizar una señal; principalmente, se requiere conservar una resolución mínima, tanto en la amplitud de la señal como en el intervalo entre muestras. La primera está definida por la longitud de la palabra digital que representará cada punto de la señal; es decir, una palabra de menor longitud (8 bits, por ejemplo) nos dará un menor número N de combinaciones digitales para representar cada muestra (256 combinaciones), por lo que el error introducido puede ser hasta de 1/N (1/256). A medida que aumenta el número de dígitos binarios, este error, conocido como de cuantización, disminuirá proporcionalmente.

Considerando que las aplicaciones de este paquete incluyen principalmente a las señales biomédicas, y tomando en cuenta el error máximo que se podría permitir, dada la amplitud de estas señales y su relación señal-ruido (generalmente menor a 60 dB), una longitud de 10 dígitos binarios (bits) es adecuada para la conversión de las muestras (Usi, 1979). La

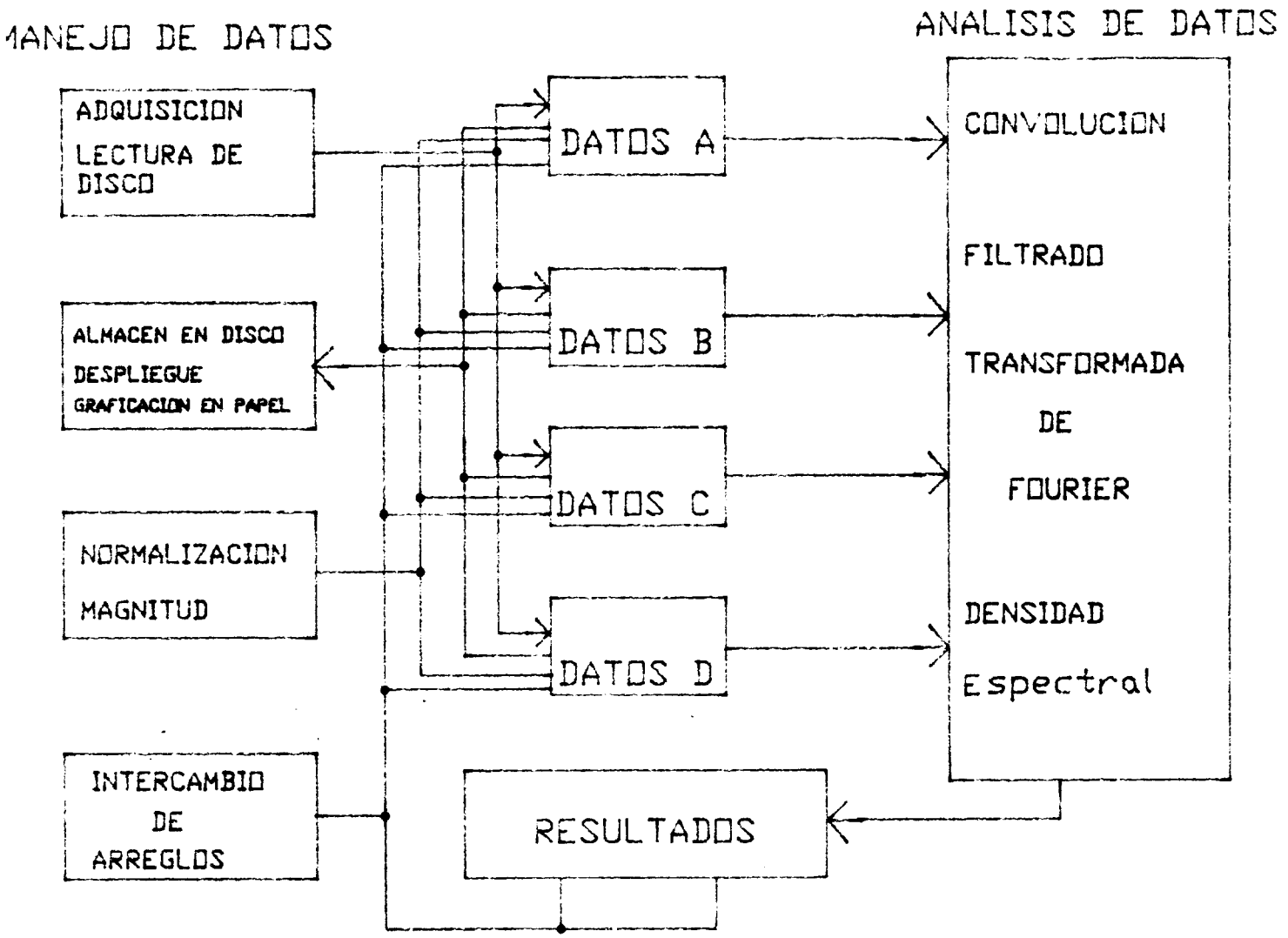


Figura 3.1 Estructura general del manejo de arreglos dentro del programa

tarjeta de conversión analógica/digital que se emplea para la adquisición de datos en este paquete maneja palabras de 12 bits, lo que proporciona una mayor calidad en la representación de la señal principalmente porque permite conservar la resolución de la señal cuando ésta tiene variaciones en la línea de base (por ejemplo, cuando existe polarización en los electrodos con los que se registra la señal).

Respecto a la frecuencia de muestreo, existe una relación perfectamente establecida, considerando el contenido en frecuencias de la señal que se desea adquirir. Esta relación, conocida como el teorema de Nyquist o del muestreo, establece que la frecuencia mínima para muestrear una señal debe ser el doble de la máxima frecuencia contenida en la señal (suponiendo que es una señal limitada en banda). Respetar esta relación evita el efecto conocido como "superposición espectral", en el que aparecen componentes de frecuencia debidos al traslape del espectro con su repetición periódica, inherente al proceso de muestreo (figura 3.2).

b. Circuito de Conversión Analógica/Digital

La tarjeta de conversión analógica/digital, construída en el Laboratorio (Azpiroz, 1986) se basa en un convertidor de aproximaciones sucesivas discreto, que emplea un registro de aproximaciones sucesivas de 12 bits (DM2504) y un convertidor digital/analógico de la misma resolución (AD565). Esta tarjeta fue diseñada para varias aplicaciones y maneja hasta 12 canales de entrada analógicos y 12 de salida digitales. Los canales están multiplexados por medio de los circuitos CD4051 y CD4052. La frecuencia de muestreo se controla con un reloj programable, 8253 de Intel, que manda un pulso a la línea de inicio de conversión del registro de aproximaciones sucesivas, a la frecuencia fijada por el usuario. La comunicación entre el convertidor y la computadora se realiza a través de algunos registros para las líneas de control y una interfase programable para periféricos (PPI), 8255 de Intel, por la que se envían los datos adquiridos y algunas de las líneas de control. La línea de fín de conversión, que llega por uno de los puertos del PPI desde el registro de aproximaciones sucesivas, se revisa en el programa de adquisición para tomar los datos al finalizar la conversión. Aunque el convertidor puede manejar frecuencias de muestreo hasta de 8KHz por canal, sensar la línea de fín de conversión desde el programa lo hace más lento (máximo 10 KHz/número de canales).

Esta configuración discreta permite tener tanto la conversión analógica/digital como la digital/analógica con los mismos componentes. Esta tarjeta fue desarrollada en el área de Ingeniería Biomédica de la UAM-Iztapalapa y se emplea en diversos sistemas (Véase la información detallada, el diagrama eléctrico y las especificaciones en el apéndice A).

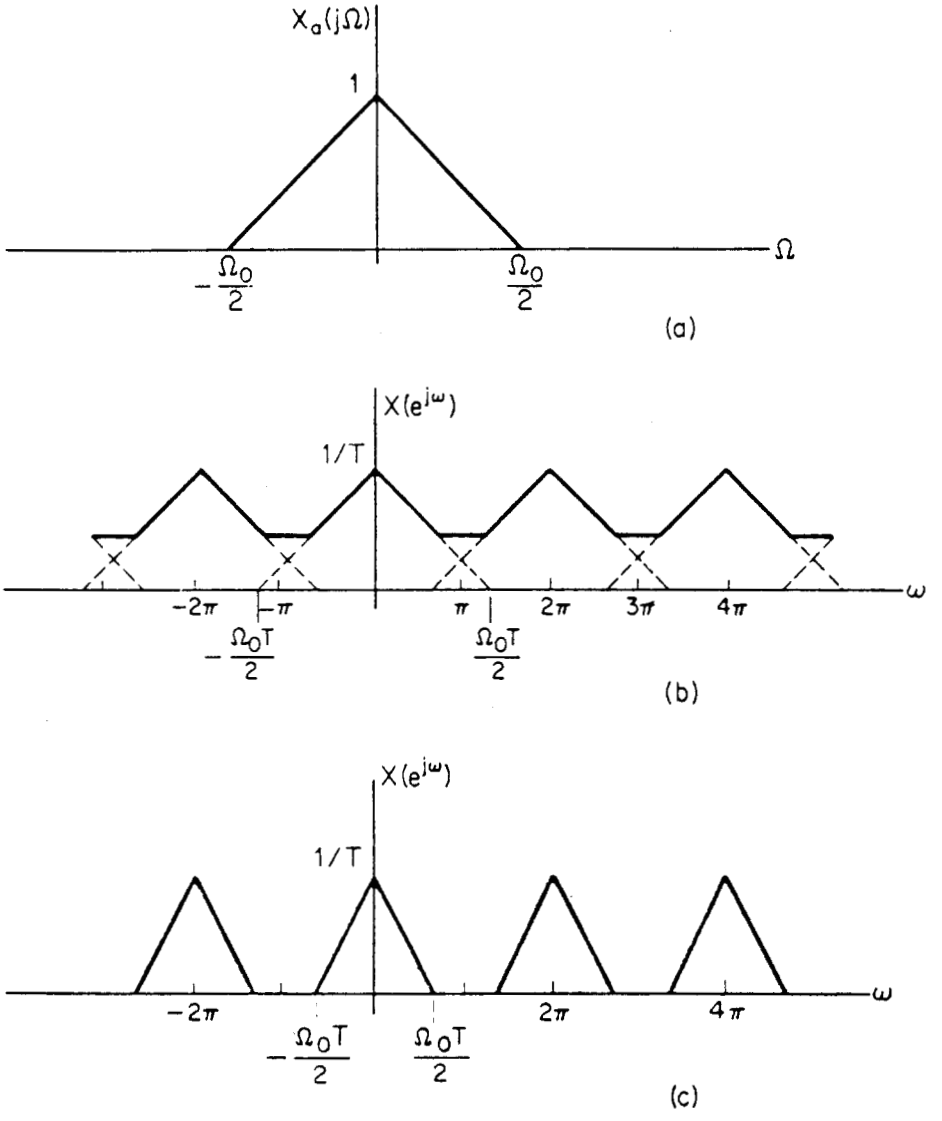


Figura 3.2 (a) Espectro de una señal analógica
(b) Espectro correspondiente a la misma señal tomada a una frecuencia de muestreo menor a Ω_0
(c) Espectro correspondiente a la misma señal con frecuencia de muestreo mayor a Ω_0

c. Programación para la adquisición

El manejo de las líneas de control y la programación necesaria para la adquisición, así como la adquisición misma, fueron realizadas en lenguaje ensamblador, puesto que permite un manejo más directo a una mayor velocidad.

Como se mencionó, la frecuencia de muestreo es variable y se controla a través de un reloj programable; por lo tanto, el primer paso dentro del procedimiento de adquisición es programar este reloj. La rutina pregunta desde Pascal la frecuencia de muestreo y el número de canales que se desea adquirir y calcula la palabra correspondiente que se debe enviar al 8253; posteriormente, ejecuta una rutina en ensamblador que realiza la programación. Los datos son adquiridos también a través de una rutina escrita en lenguaje ensamblador y se colocan en un arreglo de datos en el que cada localidad sucesiva contiene un dato de diferente canal; este arreglo se ordena posteriormente dentro de un procedimiento escrito en lenguaje Pascal. Finalmente, el conjunto de datos correspondiente a cada canal se coloca ordenadamente en el arreglo de datos adecuado: canal 1 - arreglo A, canal 2 - arreglo B, canal 3 - arreglo C y canal 4 - arreglo D.

La estructura de la rutina de adquisición se muestra en la figura 3.4. El listado completo se encuentra en el apéndice B.

d. Ejemplo

Como muestra, considérese una señal senoidal de 10 Hz, proveniente de un generador de funciones, que se desea adquirir a una frecuencia de muestreo de 500 Hz. Al elegir la opción de adquisición (A) dentro del programa principal aparecerá la secuencia mostrada en la figura 3.3(a). La señal adquirida, que en este caso se deposita en el arreglo A de datos, se muestra en la figura 3.3(b).

```
        ESTA RUTINA PERMITE ADQUIRIR HASTA CUATRO
          CANALES ANALOGICOS A TRAVES DEL CA/D

¿Desea continuar (S/N)?                          S

¿Frecuencia de muestreo (Hz)?                  500
¿Número de canales (1-4)?                        1

¿OPCION?
```

Figura 3.3(a) Pantalla de la rutina de adquisición

NOTA. Las respuestas del usuario aparecen en letra oscura.

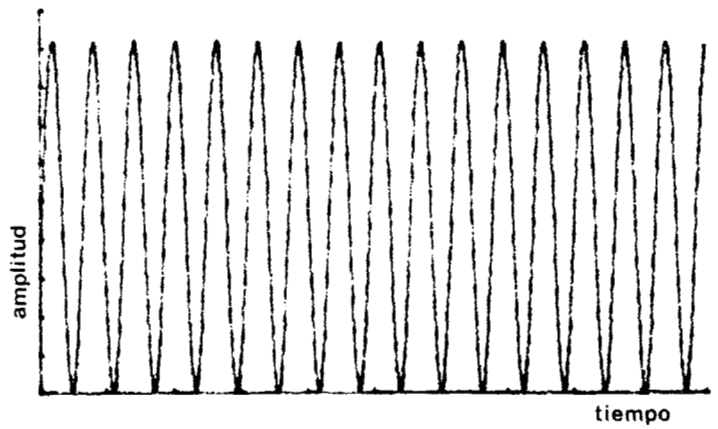


Figura 3.3 (cont.) (b) Señal depositada en el arreglo A

```
PROCEDURE Adquisición (VAR da,db,dc,dd:datos);

BEGIN
   Pregunta_parámetros;          Pregunta número de canales y
                                 frecuencia de muestreo.

   Calcula_numtimer;             Calcula palabra para progra-
                                 mar el reloj, correspondiente
                                 a la frecuencia de muestreo.

   Programa_reloj;               Esta es una rutina externa
                                 (en ensamblador) que envía la
                                 palabra al reloj programable.

   Adquiere_datos;               También es una rutina externa
                                 que regresa los datos de cada
                                 canal en la forma adquirida.

   Ordena_datos;                 Separa los datos en los
                                 arreglos correspondientes a
                                 cada canal y los deposita en
                                 el arreglo de datos pedido
                                 (da,db,dc,dd).

END;
```

Figura 3.4 Estructura de la rutina de Adquisición

2. RUTINA DE LECTURA Y ALMACENAMIENTO EN DISCO

Por razones obvias, es necesario contar dentro del paquete con rutinas que permitan almacenar o leer datos en disco, desde o hacia cualquier arreglo de datos. Todos los datos dentro del programa son manejados con aritmética de punto flotante, y por lo tanto, como datos reales. Esto representó un problema al querer guardar la información en disco, puesto que un dato real ocupa varias palabras de 8 bits (bytes); por ésto, fue necesario almacenar y leer los datos en un formato entero, que solamente ocupa dos palabras de 8 bits, lo cual da un total de 1024 bytes para cada arreglo de 512 datos.

La figura 3.5 muestra la secuencia que aparece en la pantalla al elegir esta opción. El listado de la rutina aparece en el apéndice B.

```
        ESTA RUTINA PERMITE LEER O ALMACENAR
            UN ARCHIVO DE DATOS EN DISCO

¿Desea continuar (S/N)?                          S

(L) Lectura de disco
(A) Almacenamiento en disco

¿OPCION?                                         L

¿Nombre de archivo?                              PRUEBA.DAT

¿En qué arreglo desea depositar los datos?       A
```

Figura 3.5 Pantalla de la rutina de lectura y almacenamiento.

3. RUTINA DE DESPLIEGUE DE ARREGLOS EN PANTALLA

Esta rutina permite desplegar en pantalla la parte real de cualquiera de los arreglos de datos o resultados. Existen varias opciones de despliegue:

a. Despliegue del arreglo completo (512 datos)
b. Despliegue extendido de una parte del arreglo (el número de puntos deseado)
c. Escalamiento vertical automático de los datos
d. Despliegue del arreglo sin escalamiento

La rutina fija las escalas y las unidades correspondientes, dependiendo de si es un arreglo en el tiempo o en la frecuencia, y de la frecuencia de muestreo. La figura 3.6 muestra gráficamente las diferentes opciones de despliegue. El listado de la rutina aparece en el apéndice B.

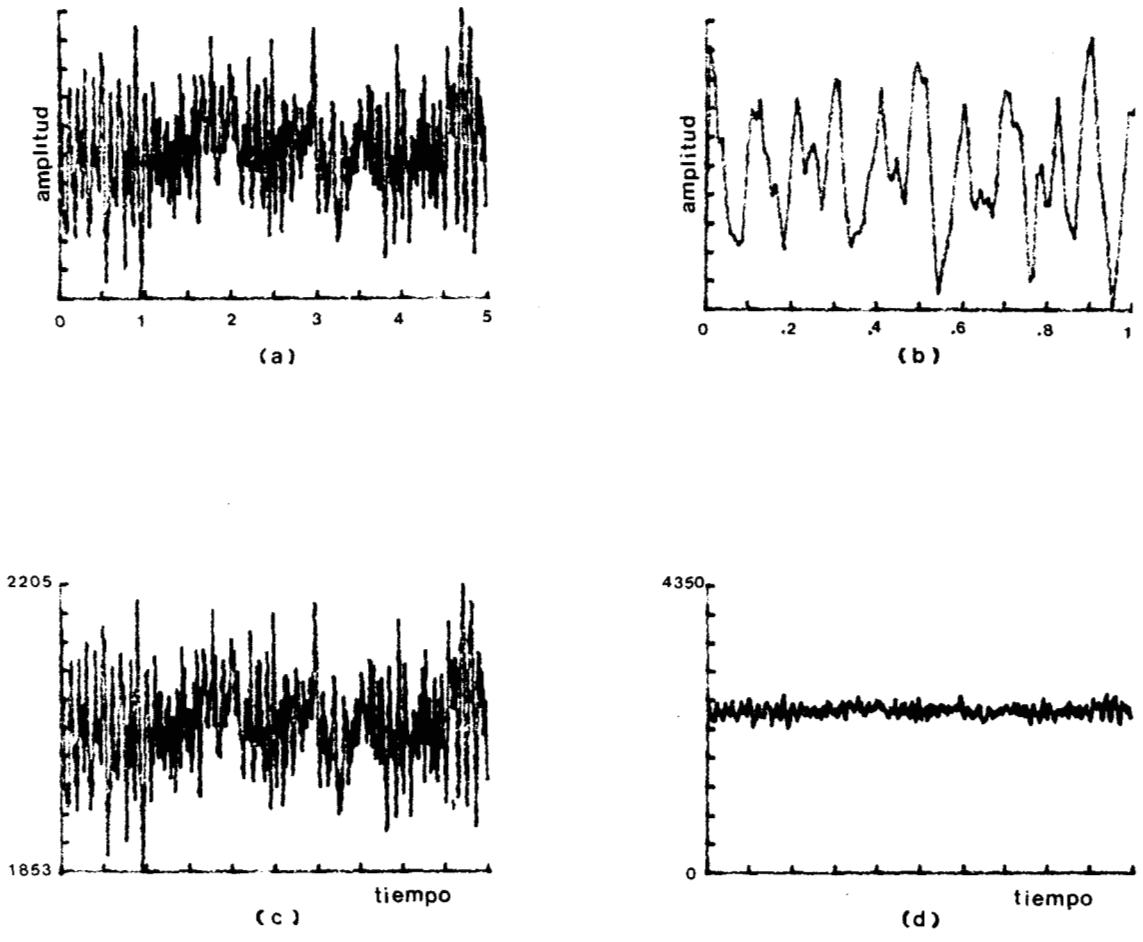


Figura 3.6 (a) Despliegue de un arreglo de datos completo
(b) Despliegue extendido de 100 puntos de la señal
(c) Despliegue con escalamiento vertical de un arreglo
(d) Despliegue del mismo arreglo sin escalamiento

4. RUTINA DE GRAFICACION

Dentro del menú principal se tiene la opción de obtener una impresión en papel de la señal contenida en cualquiera de los arreglos de datos A,B,C,D. Esta rutina está diseñada para funcionar con un graficador marca Denki, modelo Sweet-P, que se tenía disponible para este fín. La rutina permite variar las escalas tanto horizontal como vertical de la gráfica en papel. La secuencia que aparece en la pantalla al elegir la opción de graficación en el menú principal se muestra en la figura 3.7. El listado del programa se encuentra en el apéndice B.

```
        ESTA RUTINA GRAFICA CUALQUIERA DE LOS ARREGLOS
        DE DATOS EN UN GRAFICADOR MARCA DENKI (SWEET-P)

¿Desea continuar (S/N)?                                  S

¿Qué arreglo desea graficar (A,B,C,D)?                   A

¿Escala vertical (Página completa = 1)?                  .5

¿Escala horizontal (Página completa =1)?                 .5

¿OPCION?
```

Figura 3.7 Pantalla de la rutina de graficación

5. RUTINA DE NORMALIZACION

Esta rutina elimina el valor promedio de cualquiera de las secuencias contenidas en los arreglos de datos. Esta operación es de gran utilidad, especialmente si se desea realizar la convolución entre dos señales con diferente nivel promedio o si se desea obtener el contenido en frecuencias de un arreglo. La rutina calcula el promedio de los 512 puntos de la secuencia y, posteriormente, lo resta de cada valor. El resultado se deposita en el mismo arreglo. El listado de la rutina se encuentra en el apéndice B.

6. RUTINA DE CALCULO DE MAGNITUD

Para aquellas funciones cuyo resultado es inherentemente complejo (transformada de Fourier y densidad espectral), es más conveniente analizar la magnitud del resultado que las partes real e imaginaria. Para esto, se incluyó la opción (N) en el menú principal que permite realizar la siguiente operación:

$$\text{magnitud}^2 = (\text{parte real del arreglo})^2 + (\text{parte imaginaria del arreglo})^2$$

La figura 3.8 muestra la secuencia de la pantalla para la rutina de cálculo de la magnitud. El listado del programa aparece en el apéndice B.

```
                ESTA RUTINA CALCULA LA MAGNITUD DE
                      UN ARREGLO DE DATOS

¿Desea continuar (S/N)?                                    S
¿De qué arreglo desea calcular la magnitud (A,B,C,D)?      C
LISTO. El resultado se encuentra en el arreglo R.

¿OPCION?
```

Figura 3.8 Pantalla de la rutina de cálculo de magnitud.

7. RUTINA DE INTERCAMBIO DE ARREGLOS

Dada la estructura del programa y considerando que los resultados de todas las operaciones se depositan en el arreglo (R) de resultados, es necesario contar con la posibilidad de transferir las muestras de un arreglo de datos o resultados a cualquier otro. En el caso de intercambio de arreglos de datos, la transferencia es directa, puesto que son de la misma longitud; en el caso del arreglo de resultados, se maneja independientemente la parte inferior (primeros 512 puntos) y la superior (siguientes 512 puntos) y cada una de estas puede ser intercambiada con cualquier arreglo de datos.

Supóngase que deseamos intercambiar los arreglos A y C. Al elegir la opción (I) en el menú principal aparecerá la secuencia de la figura 3.9. Después de la ejecución de la rutina, los datos contenidos inicialmente en el arreglo A se encontrarán en C y viceversa; es decir, no se pierde la información de ninguno de los arreglos. El listado de la rutina aparece en el apéndice B.

```
        ESTA RUTINA PERMITE INTERCAMBIAR CUALQUIER PAR DE ARREGLOS

¿Desea continuar (S/N)?                                    S
¿Qué arreglos desea intercambiar?                          A,B

¿OPCION?
```

Figura 3.9 Pantalla de la rutina de intercambio de arreglos.

C. RUTINA DE FILTRADO

1. INTRODUCCION

En el registro de señales fisiológicas, es frecuente encontrar señales no correlacionadas con el fenómeno que se desea observar. Estas señales no deseadas son eliminadas generalmente mediante el uso de filtros, que atenúan las ondas que se encuentran dentro de cierta(s) banda(s) de frecuencia. Los filtros pueden ser de dos tipos: analógicos y digitales. Los primeros reciben como entrada una señal contínua y la entregan a la salida, con las correspondientes frecuencias atenuadas; los segundos requieren en la entrada una secuencia de números correspondiente a la señal que se desea filtrar y entregan la secuencia que representa a la señal filtrada.

Los filtros digitales presentan varias ventajas sobre los analógicos. Tienen alta inmunidad al ruido, característica que es inherente a todos los sistemas digitales. La exactitud de un filtro digital depende únicamente del error de redondeo introducido en la aritmética de la computadora, que puede reducirse mediante una programación adecuada; mientras que en los filtros analógicos la exactitud depende de los valores y tolerancia de los componentes y del ruido del circuito. Es más fácil y barato cambiar las características de un filtro digital que las de un filtro analógico, modificando un programa o una sección, o incluso manejando los coeficientes del filtro como datos de entrada. Las variaciones del voltaje de alimentación y temperatura o el envejecimiento de los componentes no afectan un programa almacenado en una computadora; por esto, los filtros digitales presentan características constantes. Esto es especialmente importante en aplicaciones médicas, donde la mayor parte de las señales contienen bajas frecuencias que pueden verse distorsionadas debido a variaciones de los componentes de un circuito analógico (Tompkins, Webster, 1981).

Los filtros digitales a su vez se dividen en dos tipos: filtros de respuesta a impulso infinita (RII) y filtros de respuesta a impulso finita (RIF). Los primeros se identifican por una ecuación de diferencias de la siguiente forma:

$$y[n] = \sum_{i=0}^{M} a_i \, x[n-i] - \sum_{i=1}^{N} b_i \, y[n-i]$$

donde x[n] es la secuencia de entrada, y[n] es la secuencia de salida y las constantes a_i y b_i son los coeficientes de la respuesta a impulso. Esta ecuación define un sistema cuya salida depende no sólo de los valores anteriores de la entrada, sino de los valores anteriores de la salida misma. La realización física de este tipo de filtros es una estructura con retroalimentación, por lo que también se conocen como filtros recursivos.

Los filtros de respuesta a impulso finita son un caso particular de los anteriores y están definidos por la siguiente ecuación de diferencias:

$$y[n] = \sum_{i=0}^{N} a_i \, x[n-i]$$

donde x[n] y y[n] son las secuencias de entrada y salida respectivamente y los coeficientes de la ecuación a_i son los coeficientes de la respuesta a impulso h[n] del sistema.

Los filtros de RIF presentan varias ventajas con respecto a los de RII (Aquino, 1984):

- Son totalmente estables, puesto que su estructura no tiene retroalimentación y, por lo tanto, no tiene polos en la función de transferencia.

- Pueden ser diseñados de tal manera que tengan una respuesta de fase lineal, si la respuesta a impulso h[n] cumple con la propiedad de simetría h[n] = h[N-1-n].

- Se puede emplear la transformada discreta de Fourier para realizar filtrados en el dominio de la frecuencia, puesto que se trata de una secuencia finita.

Por otra parte, con un filtro RII se pueden conseguir pendientes de corte más pronunciadas que con un filtro RIF del mismo orden , lo que significa un menor tiempo de cálculo para el filtrado. Además, pueden ser diseñados empleando fórmulas compactas, que en algunos casos evita el uso de una computadora.

Las aplicaciones del paquete que se presenta no incluyen el procesamiento en tiempo real, por lo que el tiempo de procesamiento no es importante. En cambio, resultan más convenientes las ventajas que presentan los filtros RIF de estabilidad y fase lineal. Por esto, se optó por el diseño de filtros de tipo RIF.

Existen varios métodos para el diseño de un filtro RIF (Rabiner, Gold, 1975):

a. Método de ventana. Se define la función de transferencia del filtro deseado, como una serie de Fourier. La respuesta a impulso relacionada con esta serie se trunca al número de puntos deseado, obteniendo los coeficientes de la respuesta a impulso.

b. Muestreo en frecuencia. Se define la función de transferencia, empleando los coeficientes de la transformada Z. De esta expresión se toman muestras equidistantes sobre el círculo unitario, que corresponden a los elementos de la transformada discreta de Fourier del filtro. Con esto se obtiene un filtro que satisface completamente la respuesta deseada en los puntos de muestra sobre el circulo unitario. El problema consiste después en ajustar los puntos en las regiones intermedias, de acuerdo a algún criterio de minimización establecido.

c. Método de diseño óptimo. En este método se definen las bandas de paso, rechazo y transición; los puntos de la función de transferencia que caen en la región de transición se consideran como variables y se ajustan para optimizar la respuesta en las otras bandas. Existen varios métodos para realizar el ajuste, como la aproximación de Chebyshev, programación lineal y otros algoritmos.

Cada uno de estos métodos presenta varias ventajas, dependiendo de su aplicación. Para la elección en este caso, se consideró la sencillez en el diseño del filtro, por lo que se eligió el método de ventana.

2. DESCRIPCION DEL METODO DE VENTANA

Un filtro FIR causal puede caracterizarse por la siguiente función de transferencia:

$$H(z) = \sum_{n=0}^{N-1} h(nT)\, z^{-n}$$

donde T es el período de muestreo y N es el orden del filtro y el número de puntos de la respuesta a impulso. Su respuesta en frecuencia está dada por:

$$H(e^{jwT}) = M(w)\, e^{j\theta(w)} = \sum_{n=0}^{N-1} h(nT)\, e^{-jwnT}$$

donde $M(w) = |H(e^{jwT})|$

y $\theta(w) = \arg H(e^{jwT})$

El retraso de fase y el retraso de grupo de un filtro están dados, respectivamente, por:

$$\tau_f = -\frac{\theta(w)}{w} \qquad \text{y} \qquad \tau_g = -\frac{d\theta(w)}{dw}$$

Para obtener un retraso de fase y grupo constantes, la respuesta de fase debe ser lineal, es decir,

$$\theta(w) = -\tau w$$

Esta restricción se satisface bajo la condición (Antoniou, 1979):

$$h(nT) = h[(N-1-n)T] \quad \text{para} \quad 0 < n < N-1$$

es decir, la respuesta a impulso del filtro debe ser una función par, simétrica con respecto al punto (N-1)/2 para N impar y con respecto a la mitad entre el punto (N-2)/2 y N/2 para N par. El retraso, tanto de fase como de grupo, esta dado por:

$$\tau = \frac{(N-1)\,T}{2}$$

Estas restricciones originan además una distribución particular en los ceros de la función de transferencia H(z), en donde cada cero aparece en forma simétrica con respecto al circulo unitario.

El primer paso en el diseño del filtro es definir la respuesta en frecuencia que se desea. La respuesta en frecuencia de un filtro no recursivo es una función periódica de w, con período w_s, que puede ser representada como una serie de Fourier.

Considérese la respuesta en frecuencia del filtro, expresada por:

$$H(e^{jwT}) = \sum_{n=-\infty}^{\infty} h(nT)\, e^{-jwnT}$$

donde

$$h(nT) = \frac{1}{w_s} \int_{-w_s/2}^{w_s/2} H(e^{jwT}) \, e^{jwnT} \, dw$$

Si $e^{jwT} = z$, tenemos:

$$H(z) = \sum_{n=-\infty}^{\infty} h(nT) \, z^{-n}$$

Por lo tanto, teniendo una expresión analítica de la respuesta deseada, puede definirse fácilmente la función de transferencia correspondiente. Sin embargo, para poder realizar en la práctica esta función, debe truncarse a un número de puntos finito, asignando la respuesta a impulso de la siguiente manera:

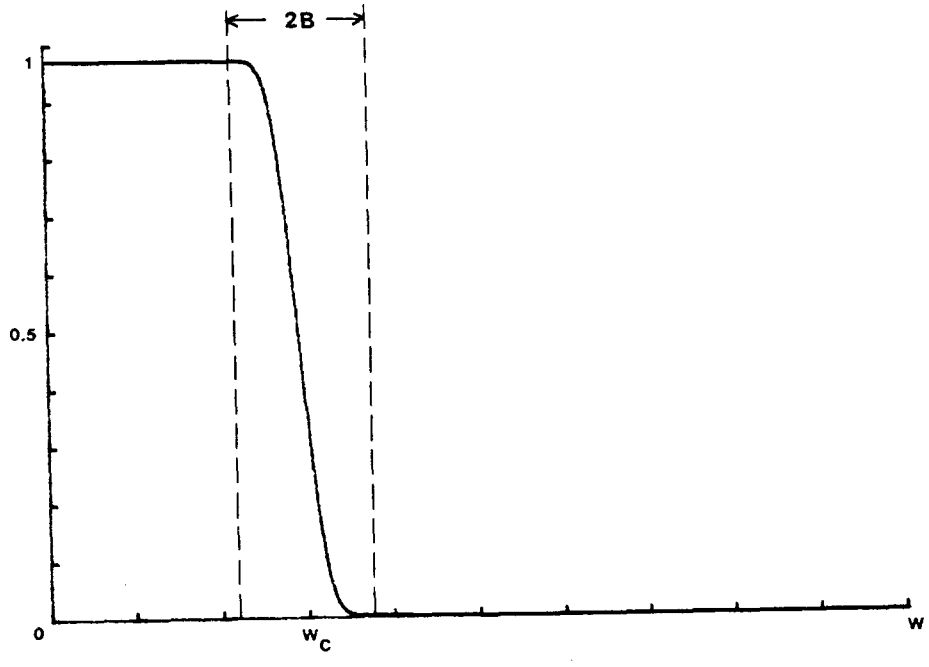
$$h(nT) = \emptyset \quad \text{para} \quad |n| > (N-1)/2$$

de donde:

$$H(z) = h(\emptyset) + \sum_{n=1}^{(N-1)/2} [h(-nT) \, z^{n} + h(nT) \, z^{-n}]$$

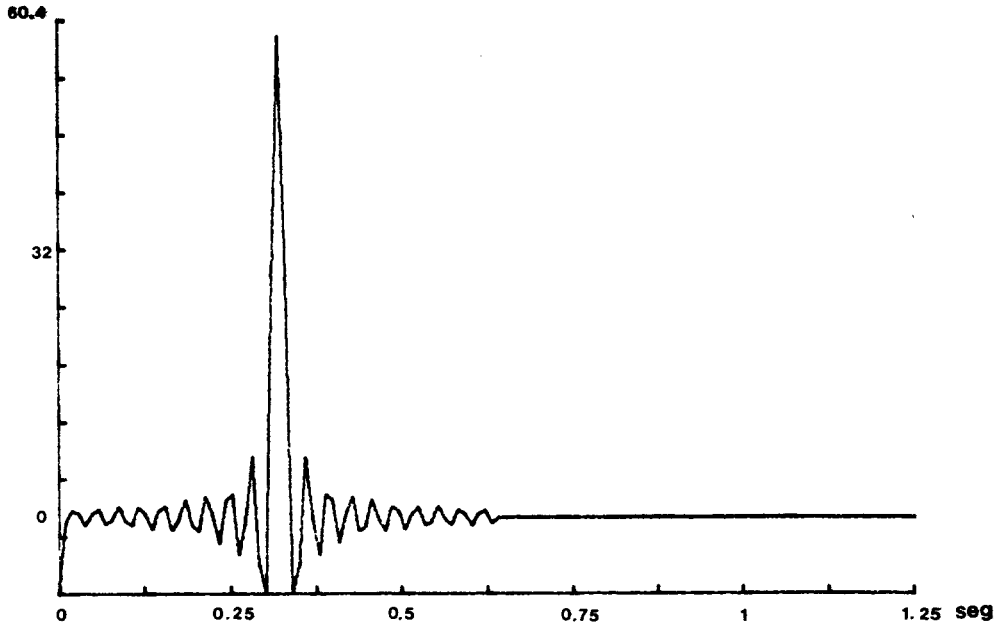
La amplitud de la respuesta en frecuencia del filtro diseñado de esta manera presenta fuertes oscilaciones en las transiciones de las diferentes bandas, debido a la lenta convergencia de la serie de Fourier en estos puntos. Este efecto se conoce como fenómeno de Gibbs y su frecuencia aumenta en función del número de puntos del filtro (N). Un método rudimentario para evitar estas discontinuidades es "suavizar" la o las pendientes de corte, mediante una función sencilla (Antoniou, 1979). En este caso se eligió una función cosenoidal de la siguiente forma:

$$a(t) = \frac{1}{2} \left(1 + \cos \left[\frac{\pi \, (w - w_c + B)}{2 \, B} \right] \right)$$

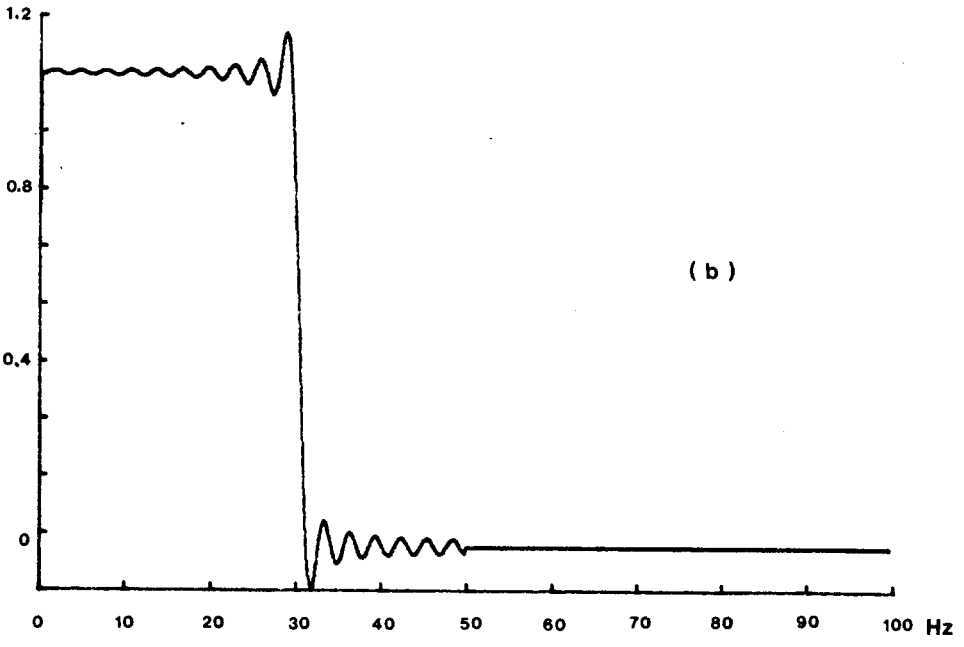


El porcentaje de atenuación se determina por la relación $2B/w_c$. Los valores típicos de atenuación están entre 10% y 20%. La respuesta a impulso resultante, posterior a la atenuación, presenta una cantidad considerablemente menor de oscilaciones. Sin embargo, para evitar aún más el fenómeno de Gibbs en el truncamiento de la respuesta a impulso y, por lo tanto, de la función de transferencia, se emplea una función de ventana, que no es más que otra función "suavizadora", de la longitud del filtro. Gráficamente puede analizarse el efecto de una ventana en la función de transferencia y en la respuesta a impulso del filtro. La figura 3.10 muestra la respuesta en frecuencia y la respuesta a impulso de un filtro pasa bajas ideal, con frecuencia de corte de 30 Hz y porcentaje de atenuación cosenoidal de 0 %. El filtro es de orden 64, lo que significa que la respuesta a impulso ha sido truncada a 64 puntos, multiplicando por una ventana rectangular. Por otro lado, la figura 3.11 muestra el efecto equivalente empleando una ventana de tipo Hamming para el mismo filtro. Se observa la disminución sustancial en el número y amplitud de las oscilaciones laterales.

Finalmente, tanto la función de transferencia como la respuesta a impulso del filtro pueden usarse para realizar el filtrado de una señal, mediante la multiplicación de los espectros, si se usa la primera, o mediante la función de convolución en el dominio del tiempo, en el caso de emplear la segunda. Los resultados obtenidos en ambos casos son idénticos.

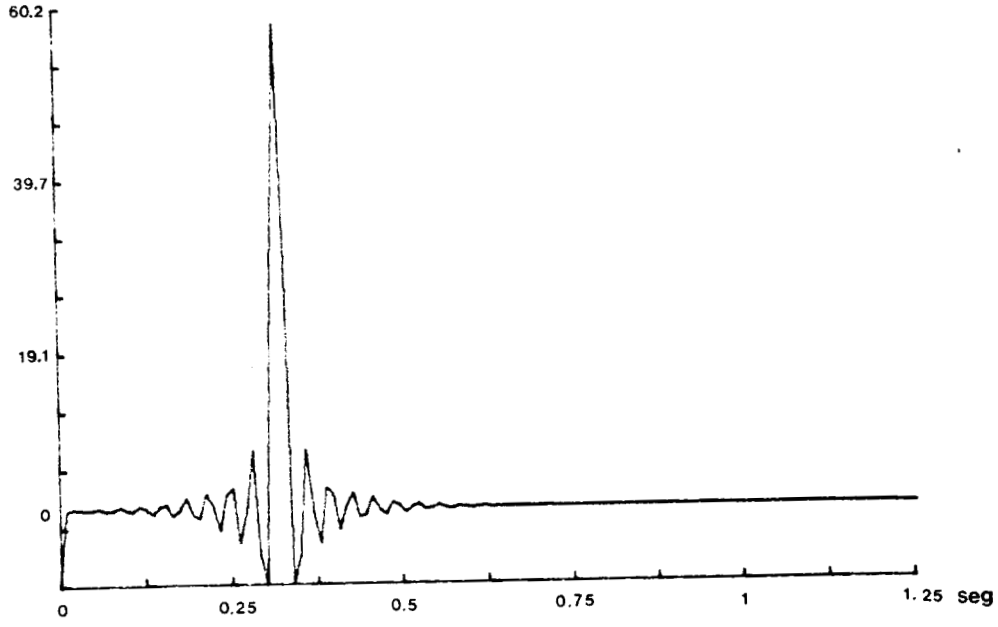


(a)

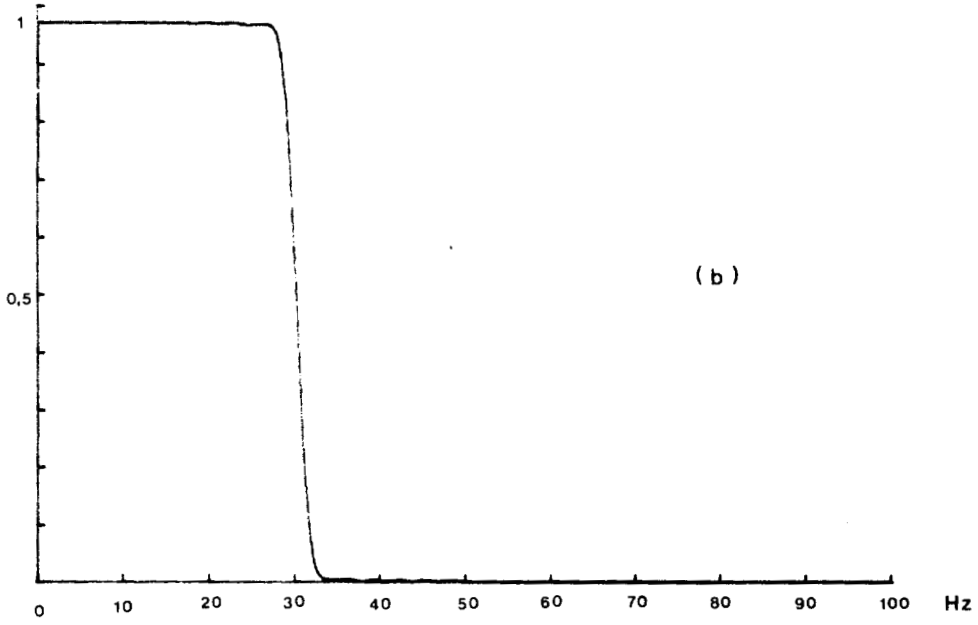


(b)

Figura 3.10 (a) Respuesta a impulso de un filtro pasa bajas, con frecuencia de corte 30 Hz, truncada a 64 puntos con una ventana rectangular
(b) Respuesta en frecuencia de (a)



(a)



(b)

Figura 3.11 (a) Respuesta a impulso de un filtro pasa bajas, con frecuencia de corte 30 Hz, truncada a 64 puntos con una ventana de tipo Hamming
(b) Respuesta en frecuencia de (a)

3. DESCRIPCION DEL ALGORITMO

La opción (F) Filtrado del programa principal permite diseñar la respuesta a impulso y la función de transferencia de un filtro digital de respuesta a impulso finita. El tipo de filtro y el método de diseño empleados presentan ciertas ventajas con respecto a otros filtros. Se puede diseñar filtros de todos tipos: pasa bajas, pasa altas o pasa bandas. Las frecuencias de corte están limitadas solamente por la frecuencia de muestreo usada y por un porcentaje de atenuación, como se describirá posteriormente. Se ofrece la posibilidad de emplear una ventana de Hamming, para disminuir en lo posible el fenómeno de Gibbs.

La rutina diseña la respuesta a impulso y la función de transferencia del filtro mediante la ejecución ordenada de los siguientes pasos:

a. Definición de la función de transferencia ideal del filtro deseado. Cuando el usuario determina la o las frecuencias de corte del filtro, se define la función de transferencia con pendiente de corte infinita en los puntos deseados. Esta función se traduce en el dominio del tiempo en una respuesta a impulso con demasiados lóbulos laterales y de amplitud muy grande con respecto al lóbulo principal. Al filtrar la señal, esta respuesta a impulso produce el fenómeno de Gibbs, que distorsiona el resultado.

b. Multiplicación de la función de transferencia por un porcentaje de atenuación en las pendientes de corte. El fenómeno de Gibbs puede evitarse en cierto grado "suavizando" las pendientes de corte, en este caso, con una función cosenoidal. La atenuación cosenoidal es un factor proporcionado por el usuario y está dada como un porcentaje de la frecuencia de corte. Para el caso de filtros pasa banda, la atenuación será mayor en las frecuencias de corte más altas.

c. Cálculo de la transformada inversa de Fourier para obtener la respuesta a impulso correspondiente. Cabe recordar que la función que se desea transformar debe tener una configuración par. Esto significa que en los 512 puntos que ha definido el usuario debe caber un período completo de la función de transferencia del filtro. Esto limita la frecuencia máxima de corte que se puede definir a únicamente la mitad de la frecuencia de muestreo, considerando además el porcentaje de atenuación.

d. Multiplicación por una ventana de Hamming, cuando se requiera. La función de transferencia "suavizada" corresponde a una respuesta a impulso con un número menor, pero no adecuado de lóbulos laterales. Para evitar aún más el fenómeno de Gibbs, conviene multiplicar por una ventana, que no es más que otra función "suavizadora". Para este fín, se ha elegido la ventana de Hamming, que proporciona una atenuación de 40 dB en el primer lóbulo lateral, con respecto al principal. En caso de no multiplicar por la ventana de Hamming, en realidad se opta por multiplicar por una ventana rectangular (truncando a 512 puntos), que atenúa solamente 10 dB.

e. Cálculo de la transformada de Fourier de la respuesta a impulso, para obtener la función de transferencia final del filtro.

La rutina deposita la respuesta a impulso final del filtro en el arreglo A de datos y la función de transferencia en el arreglo C. Así, el filtrado de una señal puede realizarse mediante la convolución de la respuesta a impulso del filtro con la señal en el tiempo, o multiplicando la función de transferencia del filtro con el espectro en frecuencias de la señal y posteriormente regresando al tiempo, mediante la transformada inversa de Fourier. Ambos métodos conducen a resultados idénticos.

La estructura del programa principal se detalla en la figura 3.12. El listado completo de todos los procedimientos se encuentra en el apéndice B.

4. EJEMPLO

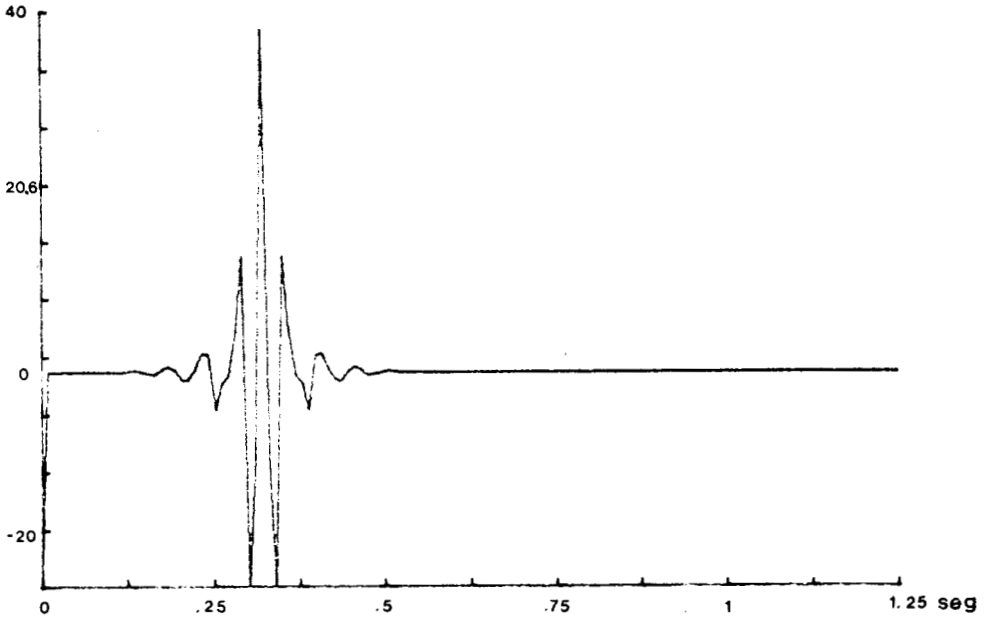
Se quiere diseñar un filtro pasa banda, de frecuencia de corte de 20 a 40 Hz, para filtrar una señal electroencefalográfica adquirida a una frecuencia de muestreo de 102.4 Hz. Después de seleccionar la opción F en el menú principal, aparecerá la secuencia de la figura 3.13. La respuesta a impulso y la función de transferencia del filtro se muestran en la figura 3.14. La señal sin filtrar se convoluciona con la respuesta a impulso del filtro, para obtener la función filtrada, como se muestra en la figura 3.15.

```
PROCEDURE filtro(da,dc)                ;da-arreglo de datos A
                                       ;dc-arreglo de datos C
                                       ;A- respuesta a impulso
                                       ;C-función de transfer.
BEGIN

   Parámetros;                        ;Pregunta parámetros a
                                       ;usuario:
                                            tipo de filtro
                                            frec. de corte
                                            atenuación cos.
                                            tipo de ventana
                                            frec. de muestreo

   Correspondiente;                    ;Calcula  función  de
                                       ;transferencia atenuada

   IFFT;                               ;Calcula resp. a imp.

   IF ventana='SI' THEN

      Hamming;                         ;Calcula la ventana y
                                       ;multiplica por resp imp

   FFT;                                ;Calcula función de
                                       ;transferencia final

END;
```

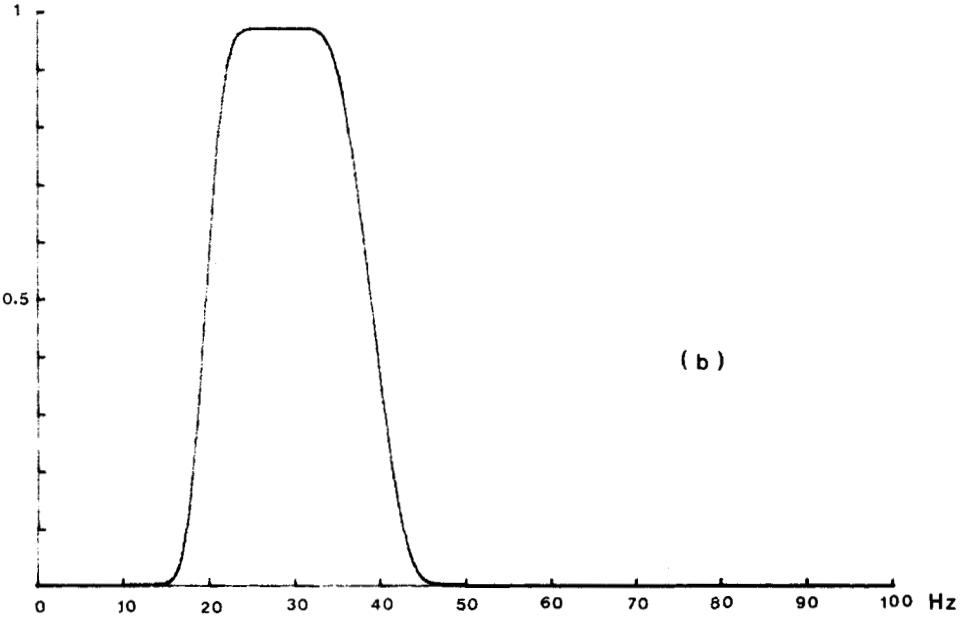
Figura 3.12 Estructura de la rutina de filtrado

```
        ESTA RUTINA CALCULA AL RESPUESTA A IMPULSO Y LA
          FUNCION DE TRANSFERENCIA DE UN FILTRO Y  LAS
        DEPOSITA EN LOS ARREGLOS A  Y C RESPECTIVAMENTE

¿Desea continuar (S/N)?                                          S

¿Cuántas bandas de paso desea?                                   1

¿En qué frecuencia comienza la banda 1 (Hz)?                     20

¿En qué frecuencia termina la banda 1 (Hz)?                      40

¿Qué porcentaje de atenuación cosenoidal desea (0 - 1)?          0.3

¿Intervalo de muestreo (seg)?                                    .01

¿Número de puntos del filtro (0-255, impar)?                     65

¿Desea aplicar una ventana de Hamming a sus datos (S/N)?         S

LISTO. La respuesta a impulso del filtro se encuentra en el
       arreglo A y la función de transferencia en el C.

¿OPCION?
```

Figura 3.13 Pantalla de la rutina de filtrado.

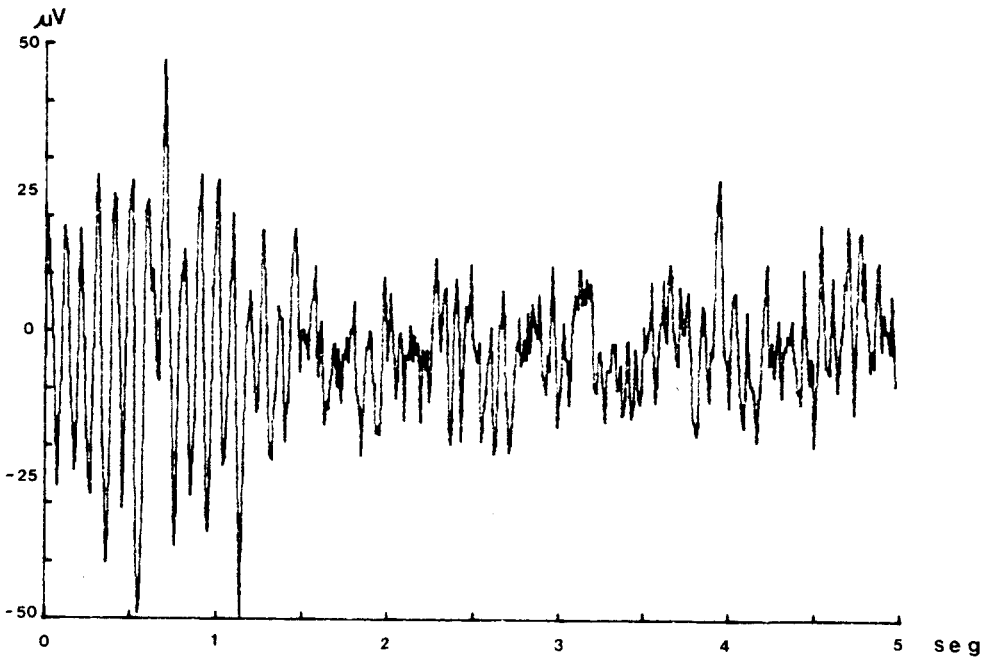


(a)

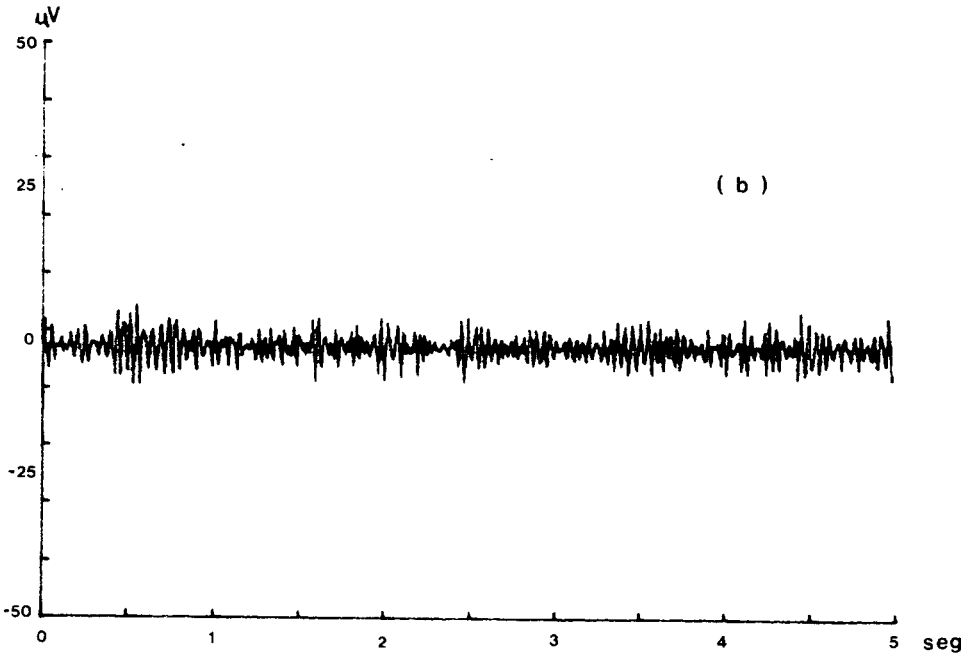


(b)

Figura 3.14 (a) Respuesta a impulso de un filtro pasa banda de 20 a 40 Hz, frecuencia de muestreo 102.4 Hz, truncada a 65 puntos con una ventana Hamming
(b) Respuesta en frecuencia de (a)



(a)



(b)

Figura 3.15 (a) Señal electroencefalográfica, frecuencia de muestreo 102.4 Hz
(b) Señal filtrada, resultado de la convolución de (a) con la respuesta a impulso del filtro mostrado en la figura 3.14

D. RUTINA PARA EL CALCULO DE LA TRANSFORMADA DE FOURIER

1. INTRODUCCION

Una transformación es cualquier función que se aplique a un conjunto de datos generalmente para facilitar su análisis. El ejemplo más común de una transformación útil es el logaritmo de una función, que nos permite transformar las operaciones multiplicación y división, un tanto complejas, en simples sumas y restas. La transformada de Fourier es la representación de una señal por medio de la suma de funciones senoidales de diferentes frecuencias; es decir, es la representación en el dominio de la frecuencia de una señal. Aunque inicialmente esta transformación fue definida por Fourier para la solución de un problema de transferencia de calor, sus aplicaciones se han extendido a prácticamente todos los campos de la ciencia y la tecnología: sistemas lineales, óptica, física cuántica, procesos aleatorios, probabilidad, problemas de valor en la frontera, etc.

La transformada de Fourier es una transformación lineal de una función y se define de la siguiente manera:

$$F(w) = \int_{-\infty}^{+\infty} f(t)\, e^{-jwt}\, dt$$

Para aquellas funciones en que la expresión analítica de la transformada es muy complicada, se utiliza la transformada discreta de Fourier (TDF), la cual representa muestras de la función F(w) y es una función periódica de w, de la siguiente manera:

$$F(k) = \sum_{n=0}^{N-1} f(n)\, e^{-jkn/N} \qquad k=0,1,..,N-1$$

donde f(n) son también muestras tomadas de la señal original f(t) a un cierto intervalo de muestreo. La naturaleza discreta de ambas señales les permite ser analizadas mediante una computadora. Sin embargo, el número de operaciones, y por lo tanto el tiempo necesario para calcular una transformada discreta de Fourier de la forma definida anteriormente, es muy grande (aproximadamente N^2 multiplicaciones complejas y N^2 sumas complejas, donde N es el número de datos); esto limitó durante muchos años el uso de la transformada de Fourier. En 1965 John W. Tukey y James W. Cooley publicaron un algoritmo eficiente para el cálculo de la transformada discreta de Fourier, que redujo el número de operaciones considerablemente (Cooley, Tukey, 1965). Posteriormente, se dieron a conocer los trabajos de diversos autores sobre algoritmos y técnicas similares para reducir el tiempo de cálculo de una transformación (Rudnick, 1966; Good, 1968). El conjunto de todas estas técnicas se conoce como transformada rápida de Fourier (TRF) y ha permitido extender las aplicaciones del análisis en el dominio de la frecuencia a varios tipos de señales.

El principio fundamental de reducción de estos algoritmos se basa en la periodicidad y simetría del factor e(jwn), cuando el número de datos N es una potencia de 2. Estas características pueden explotarse arreglando los coeficientes de la secuencia, ya sea en el tiempo o en la frecuencia, de tal manera de descomponerla en secuencias más pequeñas; es decir, en el cálculo de transformadas discretas de menor longitud. La manera en que se efectúa esta reducción conduce a diferentes algoritmos, que disminuyen el tiempo de cálculo más o menos en la misma proporción. Los algoritmos principales pueden dividirse en dos grupos:

- Decimación en tiempo. Consiste en arreglar la secuencia de entrada x(n) en secuencias más pequeñas, considerando los datos pares e impares.

- Decimación en frecuencia. Consiste en partir la secuencia de salida o los elementos de la transformada discreta de Fourier X(k) en secuencias de longitud menor.

Ambos métodos conducen a resultados similares en cuanto al número de operaciones (aproximadamente $N \log_2 N$). Aunque estos algoritmos pueden aplicarse a cualquier número de datos, su eficiencia aumenta si se consideran secuencias cuya longitud sea una potencia de 2.

Para el desarrollo de este paquete, se empleó el método de decimación en tiempo, en el que se entrega la secuencia de entrada, que ha sido previamente ordenada de acuerdo al algoritmo, y se obtiene la secuencia ordenada de los coeficientes de la transformada discreta. En la siguiente sección se describe el algoritmo de decimación en tiempo, tal como aparece en la literatura (Oppenheim, Schafer 1975).

2. DESCRIPCION DEL METODO DE DECIMACION EN TIEMPO

La disminución del número de operaciones en este método se logra partiendo la longitud de la secuencia inicial en secuencias más pequeñas, agrupando los términos en función de las características de periodicidad y simetría de la exponencial compleja:

$$e^{-j\,(2\pi/N)\,k\,(N-n)} = e^{j\,(2\pi/N)\,kn}$$

$$e^{-j\,(2\pi/N)\,kn} = e^{-j\,(2\pi/N)\,k\,(n+N)} = e^{-j\,(2\pi/N)\,(k+N)\,n}$$

Considérese el caso particular cuando N es una potencia entera de base 2, es decir:

$$N = 2^{v}\ ,\quad \text{donde } v \text{ es un número entero.}$$

La transformada discreta de Fourier (X(k)) de la secuencia x(n) se define con la siguiente relación:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\,(2\pi/N)\,nk} \qquad k=0,1,\ldots,N-1$$

La relación inversa, es decir, la antitransformada de la función X(k) se define de la siguiente manera:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j\,(2\pi/N)\,nk} \qquad n=0,1,\ldots,N-1$$

Esta permite recuperar la secuencia x(n) a partir de su transformada. Ambas secuencias x(n) y X(k) son periódicas, con período N, por las características del par de transformación discreta de Fourier (Brigham, 1979).

El cálculo de las ecuaciones anteriores es idéntico, exceptuando el signo de la exponencial y el factor 1/N. Por esto, basta con analizar el caso de la transformada directa. Partiendo la secuencia de entrada x(n) en dos secuencias, construidas con los puntos pares e impares de x(n), se tiene:

$$X(k) = \sum_{n\ par} x(n)\, e^{-j\,(2\pi/N)\,nk} + \sum_{n\ impar} x(n)\, e^{-j\,(2\pi/N)\,nk}$$

Sustituyendo las variables n = 2r para n par y n = 2r+1 para n impar, se tiene:

$$X(k) = \sum_{r=0}^{N/2-1} x(2r)\, e^{-j\,(2\pi/N)\,2rk} + \sum_{r=0}^{N/2-1} x(2r+1)\, e^{-j(2\pi/N)(2r+1)k}$$

$$= \underbrace{\sum_{r=0}^{N/2-1} x(2r)\, e^{-j(4\pi/N)rk}}_{G(k)} + e^{-j(2\pi/N)k} \underbrace{\sum_{r=0}^{N/2-1} x(2r+1)\, e^{-j(4\pi/N)rk}}_{H(k)}$$

G(k) y H(k) son dos transformaciones de N/2 puntos cada una, correspondientes a las secuencias de puntos pares e impares, respectivamente, de x(n). Nótese que cada una de ellas requiere de (N/2) multiplicaciones y sumas, más las N operaciones coorrespondientes a la combinación de ambas transformadas; es decir, un total de N + (N /2) multiplicaciones y sumas complejas. Esta cantidad es menor que el número de operaciones iniciales N^2 , para cualquier N mayor que 2.

Llámese ahora g(r) a la secuencia formada por los datos pares de x(n) y h(r) a la formada por las muestras impares. El mismo procedimiento se sigue para obtener las transformadas G(k) y H(k), de tal forma que se tiene:

$$G(k) = \sum_{r=0}^{N/2-1} g(r)\, e^{-j\,(4\pi/N)\,rk} \qquad k=0,..,N/2-1$$

$$= \sum_{l=0}^{N/4-1} g(2l)\, e^{-j(8\pi/N)lk} + e^{-j(4\pi/N)k} \sum_{l=0}^{N/4-1} g(2l+1) e^{-j(8\pi/N)lk}$$

Las divisiones se continuan hasta que se llega a tener la transformada más pequeña, de dos puntos, definida así:

$$M(k) = m(0) + e^{-j\pi k} m(1) \qquad k = 0,1$$

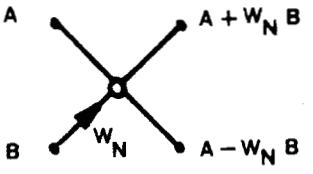
o sea:

$$M(0) = m(0) + m(1)$$

$$M(1) = m(0) - m(1)$$

Cada etapa de división se denomina decimación y el número total de decimaciones que se realiza para dividir toda la secuencia corresponde a la potencia (v) a la que se eleva la base (2) para obtener el número de datos (N).

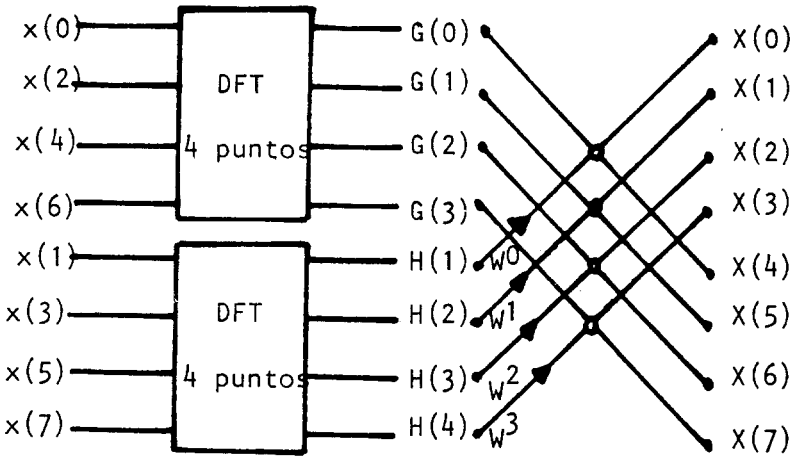
Gráficamente, este proceso se representa definiendo una operación básica de dos números; conocida como "mariposa", de la siguiente manera:

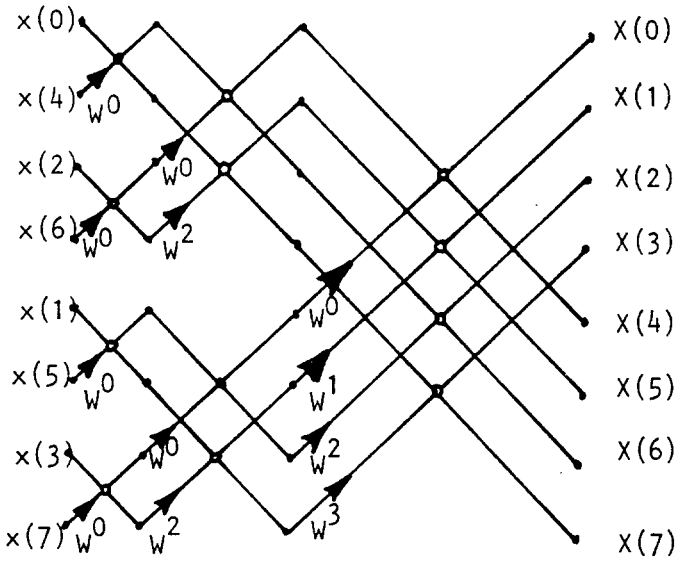


donde $W_N = e^{-j(2\pi/N)}$

La figura 3.16 muestra la representación total del cálculo de una transformada discreta de Fourier de 8 puntos, siguiendo el algoritmo descrito. Es importante notar el uso de la propiedad de periodicidad de la exponencial:

$$e^{-j(2\pi/N)(k+N/2)} = - e^{-j(2\pi/N)k}$$





$$W = e^{-j\,2\pi/N} \qquad N = 8$$

Figura 3.16 Estructura total para el cálculo de una transformada discreta de Fourier de 8 puntos, empleando el método de decimación en tiempo

que permite ahorrar muchas multiplicaciones complejas en cada decimación. En la gráfica se puede observar que se requiere un número total de N sumas complejas y N multiplicaciones complejas por cada decimación, lo que significa una cantidad final de $2N\log_2 N$ operaciones para el cálculo de todos los puntos de la transformada.

Nótese que en cada proceso de decimación, la secuencia de entrada se arregla en sus elementos pares e impares. Esto ocasiona que al finalizar las decimaciones los valores de x(n) aparezcan desordenados en la entrada, mientras que los coeficientes de la transformada discreta se obtienen en el orden adecuado. El orden de la secuencia de entrada se puede determinar mediante el algoritmo de inversión de bits (Rabiner, 1975). Supóngase que se tiene una secuencia de entrada x(n), que debe ser ordenada para iniciar el algoritmo de decimación en tiempo. Representando el indice de cada dato en codigo binario, para el ejemplo N=8 se necesitan tres bits. Si se invierte el orden de los bits, se tendrá la secuencia "ordenada" para el algoritmo de decimación en tiempo:

Secuencia inicial		Secuencia ordenada	
dec	bin	bin	dec
x(0)	x(000)	x(000)	x(0)
x(1)	x(001)	x(100)	x(4)
x(2)	x(010)	x(010)	x(2)
x(3)	x(011)	x(110)	x(6)
x(4)	x(100)	x(001)	x(1)
x(5)	x(101)	x(101)	x(5)
x(6)	x(110)	x(011)	x(3)
x(7)	x(111)	x(111)	x(7)

El método de decimación en frecuencia es muy similar al descrito y presenta las mismas ventajas en cuanto a la reducción del número de operaciones, por lo que resulta prácticamente lo mismo el empleo de cualquiera de los dos algoritmos.

3. DESCRIPCION DE LA RUTINA

La opción (T) del programa principal permite obtener los coeficientes de la Transformada de Fourier de cualquiera de las secuencias contenidas en los arreglos de datos (A,B,C,D), y la deposita en el arreglo de resultados. El método empleado es el de decimación en tiempo descrito en la sección anterior.

La rutina realiza los siguientes pasos:

a. Intercambia los datos de la secuencia de entrada, de acuerdo al algoritmo de inversión de bits.

b. Para cada decimación:

   - Calcula los coeficientes exponenciales que se necesitan en esa etapa

   - Calcula las operaciones "mariposa" de esa decimación

c. Si es una transformación inversa, multiplica cada coeficiente del resultado por el factor 1/N.

Debe recordarse que la transformada discreta de Fourier es por naturaleza una función periódica y que supone que la secuencia de entrada es también una función periódica, con período N, centrada en el orígen. Por lo tanto, la secuencia de entrada debe entregarse en una configuración par, de tal forma que el último punto del arreglo se continúe con el primero, completando un período de la señal. Esto evita un fenómeno conocido como "de fuga", en el que aparecen varios componentes de alta frecuencia debido a la discontinuidad de la señal.

El resultado se deposita en el arreglo R, en donde los primeros 512 puntos corresponden a la parte real de la transformada y los últimos 512 puntos a la parte imaginaria. El resultado se puede convertir a su representación en magnitud y fase, ejecutando la opción magnitud del menú principal.

La estructura de la rutina principal se detalla en la figura 3.17. El listado de los procedimientos se encuentra en el apéndice B.

```
PROCEDURE FFT(d,r,dec,N,tt)          ;d - arreglo de datos
                                     ;r - arreglo de resultados
                                     ;dec-número de decimaciones
                                     ;N - número de datos
                                     ;tt- tipo de transformación
BEGIN
   Intercambio;                      ;Ordena coeficientes de
                                     ;entrada
   FOR i:=1 to dec DO
   BEGIN
      Calcula_exponentes;            ;Para cada decimación
      Realiza_mariposas;             ;calcula los coeficientes
   END;                              ;y realiza las mariposas

   IF  tt=1 THEN                     ;Si la transformación es
                                     ;indirecta multiplica por
     Resultado:= Resultado/N              ;1/N
END;
```

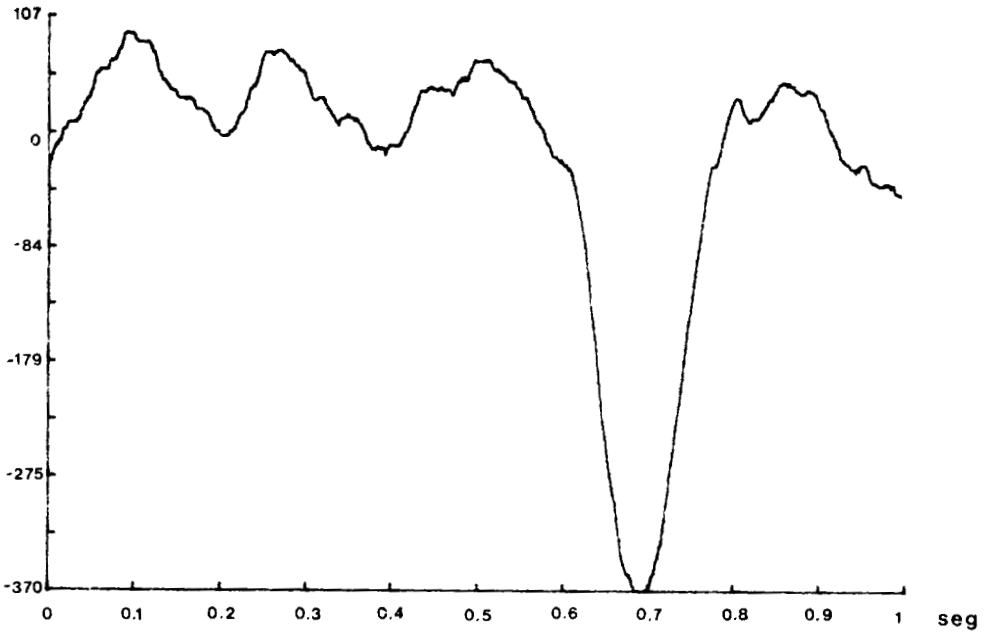
Figura 3.17 Estructura de la rutina de transformación de Fourier

4. EJEMPLO

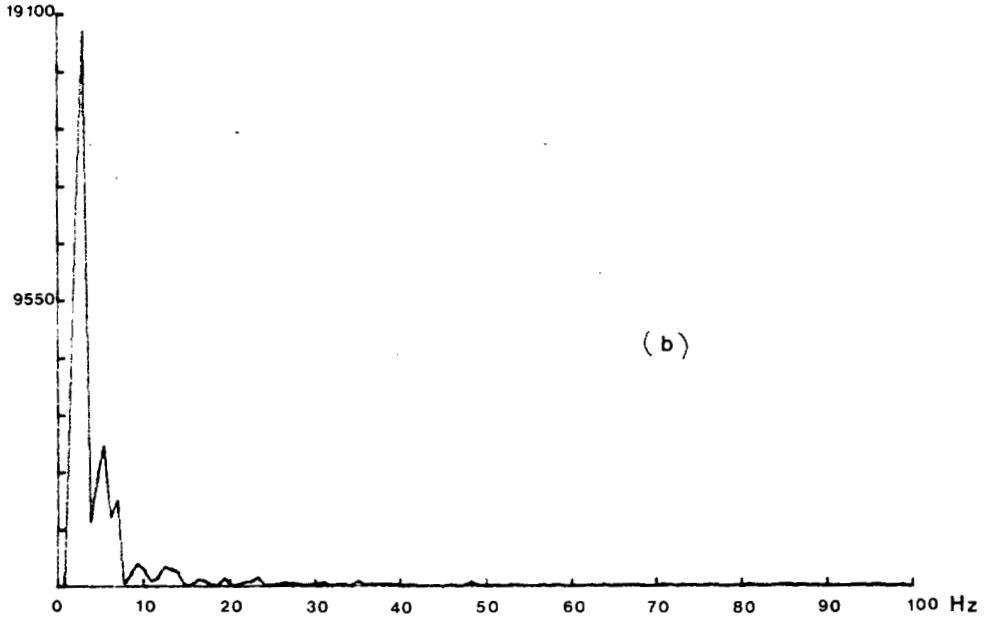
Considérese la señal mostrada en la figura 3.19(a).Se trata de una señal impedancimétrica del tórax, adquirida a una frecuencia de muestreo de 500 Hz y depositada en el arreglo A. Al elegir la opción (T) en el menú principal aparecerá la secuencia de la figura 3.18. El resultado se deposita en el arreglo R y se muestra en la figura 3.19(b).

```
        ESTA RUTINA OBTIENE LA TRANSFORMADA DE FOURIER
      DE UN ARREGLO DE DATOS Y LA DEPOSITA EN EL ARREGLO R

¿Desea continuar (S/N)?                                  S
¿Transformada (D)irecta o (I)nversa?                     D
¿Qué arreglo desea transformar (A,B,C,D)?                A

LISTO. El resultado se encuentra en el arreglo R
¿OPCION?
```

Figura 3.18 Pantalla de la rutina de Transfomación de Fourier



(a)



(b)

Figura 3.19 (a) Señal impedancimétrica del tórax, frecuencia de muestreo 500 Hz
(b) Transformada de Fourier de (a)

E. RUTINA PARA EL CALCULO DE LA DENSIDAD ESPECTRAL

1. INTRODUCCION

Una señal determinística es aquella que puede ser definida para todos los valores del tiempo, y que puede ser reproducida repetidamente con exactitud; este tipo de señales puede ser analizado fácilmente. Sin embargo, las señales encontradas en la práctica, por lo general no son determinísticas, no pueden ser descritas con certeza, ni pueden ser reproducidas. Estas señales se conocen como estocásticas o aleatorias y son producto de un fenómeno aleatorio por naturaleza.

El contenido de frecuencias de una señal determinística puede ser determinado mediante el uso de la transformada de Fourier; para una señal estocástica se emplea una función denominada espectro de potencia, densidad espectral de potencia o, simplemente, densidad espectral. Esta función representa la distribución probabilística de la potencia de la señal, con respecto a la frecuencia (Lathi, 1979).

Existen varios métodos para evaluar la densidad espectral de una función (Chen, 1979):

a. Estimación de la densidad espectral a partir del periodograma. Este método consiste en partir la secuencia del tiempo que se quiere analizar en segmentos más cortos, calcular la transformada discreta de Fourier (TDF) de estos segmentos y promediar los resultados. La magnitud cuadrada de la TDF de cada segmento se conoce como periodograma. La estimación de la densidad espectral es más exacta mientras mayor sea el número de periodogramas que se promedia, y se puede mejorar aún más multiplicando los segmentos obtenidos por una función ventana, antes de calcular la TDF correspondiente. El procedimiento se realiza de la siguiente manera:

   - Se divide la secuencia en intervalos de longitud L. Generalmente estos segmentos se traslapan una cantidad L/2.

   - Se multiplica cada segmento por una función ventana, centrada en el valor central del segmento.

   - Se evalúa la TDF empleando algún algoritmo de transformación rápida, es decir, se calcula el periodograma de cada segmento.

   - Se promedian los periodogramas para obtener la estimación final de la densidad espectral.

Este método requiere de un gran número de operaciones si se quiere obtener una buena aproximación de la densidad espectral, lo que lo hace poco práctico.

b. Estimación de la densidad espectral mediante el cálculo de la transformada de Fourier de la función de autocovarianza. La función de autocovarianza de una señal es una estimación de la variación de la misma con respecto al tiempo. Esta función siempre es igual, aun cuando se tomen diferentes muestras del conjunto de señales que genera un proceso aleatorio. Por esto, su transformada de Fourier es también independiente de la señal que se considere, siempre que provenga del mismo fenómeno. Este método permite obtener la densidad espectral directamente en el dominio de la frecuencia, como se analizará en la siguiente sección, con un gran ahorro en el número de operaciones y, por lo tanto, en el tiempo de cálculo.

c. Estimación de la densidad espectral a partir del modelo del proceso. En los casos anteriores, la estimación de la densidad espectral se realiza tomando una o varias secuencias de la señal que se quiere analizar. Un procedimiento alternativo consiste en emplear un modelo general que represente a todo el proceso que genera la señal aleatoria. A partir de los parámetros que definen al modelo se calcula el espectro del proceso, empleando el criterio de minimización de alguna función de error previamente definida.

Los métodos descritos conducen a resultados semejantes, en cuanto a la estimación de la densidad espectral. Sin embargo, el segundo método es más sencillo y requiere de un número menor de operaciones (McGillem, 1979). Por esto, es el más adecuado para los propósitos de este paquete.

2. METODO DE ESTIMACION ESPECTRAL A PARTIR DE LA AUTOCOVARIANZA

La función de autocovarianza $c(\tau)$ de una señal $x(t)$ se define como la función de autocorrelación $r(\tau)$ menos el valor promedio de la señal:

$$c(\tau) = r(\tau) - \frac{1}{N} \sum_{n=0}^{N-1} x(n)$$

La función de autocorrelación $r(\tau)$ se define de la siguiente manera:

$$r(\tau) = \int_{-\infty}^{\infty} x(\tau)\, x(t+\tau)\, d\tau$$

Calculando la transformada de Fourier de dicha función se obtiene:

$$F\{r(\tau)\} = S(w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)\, x(t+\tau)\, d\tau\, e^{-jwt}\, dt$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)\, x(t+\tau)\, e^{-jwt}\, d\tau\, dt$$

Sea $t' = t+\tau$ $\quad t = t'-\tau$ $\quad dt = dt'$

$$S(w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)\, x(t')\, e^{-jw(t'-\tau)}\, d\tau\, dt'$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)\, x(t')\, e^{-jwt'}\, e^{jw\tau}\, d\tau\, dt'$$

$$= \int_{-\infty}^{\infty} x(\tau)\, e^{jw\tau}\, d\tau \int_{-\infty}^{\infty} x(t')\, e^{-jwt'}\, dt'$$

$$X^{*}(w) \qquad\qquad X(w)$$

$$S(w) = X^{*}(w)\, X(w) = |X(w)|^{2}$$

Prácticamente, se obtiene una mejor estimación espectral de la densidad espectral $S(w)$ si se aplica una ventana a $r(\tau)$ antes del cálculo de la transformada de Fourier.

El cálculo de la función de autocorrelación en el dominio del tiempo es una operación muy lenta. De la relación obtenida se puede observar que es posible calcular la densidad espectral directamente en el dominio de la frecuencia, ahorrando una gran cantidad de operaciones, mediante el cálculo directo de la magnitud cuadrática de la transformada de Fourier de la señal original. Los pasos para la realización del procedimiento completo son:

- Se adquiere la señal que se desea analizar, respetando el criterio de Nyquist para la toma de muestras; se calcula el valor promedio y se resta de cada muestra.

- Se evalúa la TDF de la secuencia obtenida y se obtiene la magnitud cuadrática (S(w)).

- Se calcula la TDF inversa de S(w), obteniendo así las muestras de la función de autocorrelación, que se multiplican por una ventana de la longitud de la secuencia.

- Finalmente, se calcula nuevamente la TDF de las muestras corregidas, para obtener la mejor estimación de la densidad espectral.

Este procedimiento requiere del cálculo de dos transformaciones de Fourier de la longitud de la secuencia, más las operaciones necesarias para la multiplicación por la ventana. El tiempo necesario para realizar todo el procedimiento es aproximadamente 8 veces menor al requerido para calcular la función de autocorrelación en el tiempo (Chen, 1979).

3. DESCRIPCION DE LA RUTINA

Al elegir la opción (Q) del menú principal, se calcula la densidad espectral de cualquiera de los arreglos de datos (A,B,C,D), mediante el cálculo de la transformada de Fourier de la función de autocovarianza. El resultado se deposita en el arreglo R. La rutina realiza la siguiente secuencia:

- Calcula el valor promedio de la señal contenida en el arreglo de datos que se desea analizar y lo resta de todos los puntos.

- Obtiene la transformada rápida de Fourier de la secuencia contenida en el arreglo de datos. La rutina empleada es la misma descrita en la rutina TRF (Ver sección III-D).

- Calcula la magnitud cuadrada de la transformada de Fourier.

- Calcula la transformada inversa de Fourier para obtener las muestras correspondientes a la función de autocorrelación.

- Multiplica la función de autocorrelación por una función ventana, que suaviza el truncamiento de los 512 puntos con los que se trabaja. En este caso se eligió una ventana de Hamming, que proporciona una atenuación de 40 dB en el primer lóbulo secundario, con respecto al principal (Ver sección III-C).

- Calcula la TRF directa, para obtener la densidad espectral corregida.

Es importante notar que al ejecutar la rutina para el cálculo de la TRF, el resultado será una secuencia periódica, con período N=512, por las características propias de la transformada discreta de Fourier. En la pantalla aparecerá en los primeros 256 puntos la densidad obtenida y en los siguientes 256 un espejo de éstos.

La estructura de la rutina principal se describe en la figura 3.20. El listado completo de todos los programas se encuentra en el apéndice B.

```
PROCEDURE Densidad_Espectral;

BEGIN
   Normaliza;                  ;Elimina el valor promedio de
                               ;cada punto
   FFT;                        ;Calcula TDF de la secuencia

   Magnitud;                   ;Calcula magnitud cuadrada de
                               ;la TDF
   IFFT;                       ;Regresa a dominio del tiempo,
                               ;obtiene muestras de la función
                               ;de autocorrelación

   Hamming;                    ;Calcula la ventana de Hamming y
                               ;multiplica por la secuencia ant.
   FFT;                        ;Obtiene estimación corregida de
                               ;la densidad espectral
END;
```

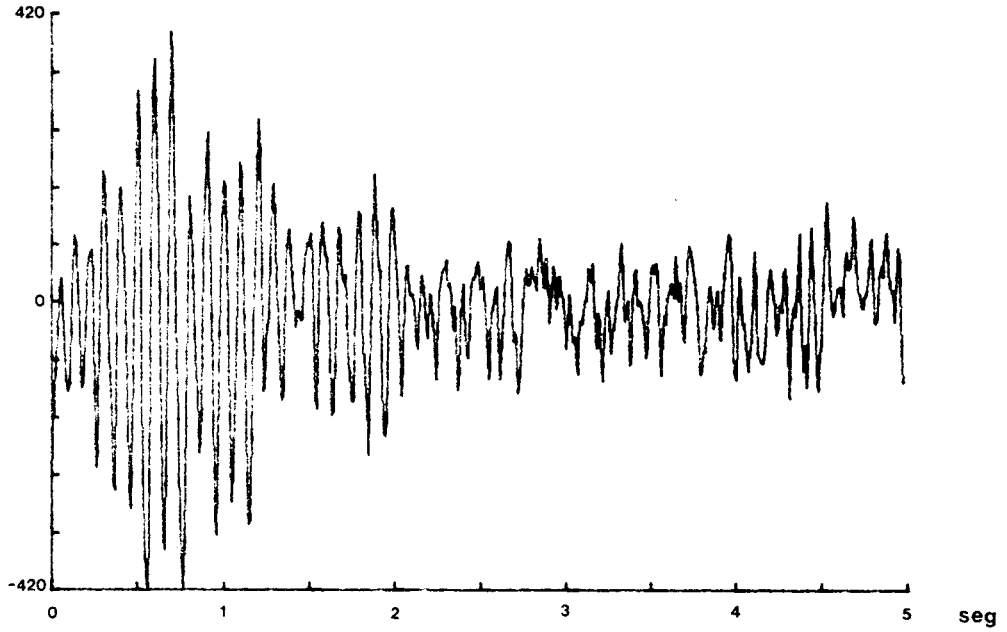
Figura 3.20 Estructura de la rutina de densidad espectral

4. EJEMPLO

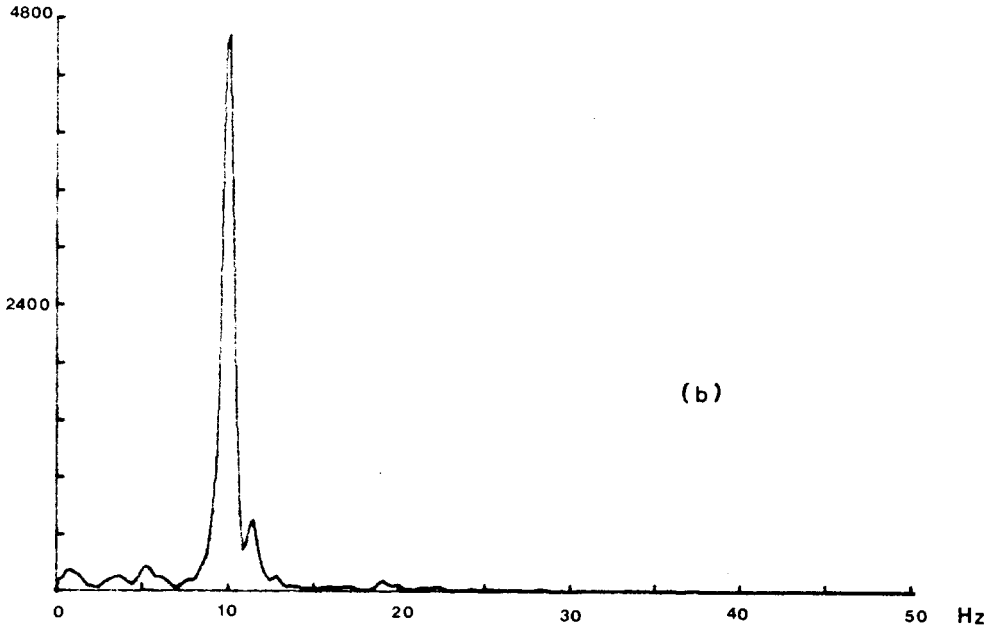
Considérese la función mostrada en la figura 3.22(a), que corresponde a una señal electroencefalográfica adquirida a 102.4 Hz y depositada en el arreglo A. Al oprimir la tecla (Q) en el menú principal aparecerá la secuencia de la figura 3.21. La densidad espectral de la señal se deposita en el arreglo R y es la que se muestra en la figura 3.22(b).

```
        ESTA RUTINA CALCULA LA DENSIDAD ESPECTRAL DE UN
         ARREGLO DE DATOS Y LA DEPOSITA EN EL ARREGLO R

¿Desea continuar (S/N)?                                  S

¿De qué arreglo desea obtener la densidad espectral?     A

¿Intervalo de muestreo (seg)?                            .01

LISTO. Su resultado se encuentra en el arreglo R

¿OPCION?
```

Figura 3.21 Pantalla de la rutina de densidad espectral



(a)



(b)

Figura 3.22 (a) Señal electroencefalográfica, frecuencia de muestreo 102.4 Hz
(b) Densidad espectral de (a)

F. RUTINA PARA EL CALCULO DE LA FUNCION DE CONVOLUCION

1. INTRODUCCION

Los principios que rigen el procesamiento digital de señales están íntimamente relacionados con lo que se denomina teoría de los sistemas lineales. Esta teoría relaciona la salida de un procesador con su entrada, y puede ser desarrollada independientemente para señales medidas en forma contínua, como para señales discretas, que ocurren sólo en ciertos instantes de tiempo.

En general, una señal discreta es el resultado del muestreo periódico de una señal contínua y se espera que del procesamiento de esta señal discreta resulte una nueva señal discreta, cuyos valores también se habrían obtenido si primeramente se procesa la señal y luego se muestrea. Por lo tanto, el objetivo de procesar una señal discreta es obtener resultados equivalentes al procesamiento de la señal contínua.

Dentro del conjunto de todos los sistemas que realizan algún tipo de procesamiento a una señal, son particularmente interesantes los sistemas lineales, invariantes en el tiempo (SLIT). Un sistema de este tipo satisface ciertas relaciones entre la entrada y la salida:

- Si la entrada se multiplica por una constante, la salida estará multiplicada por la misma constante (propiedad de escalamiento).

- Si dos o más entradas se aplican simultáneamente, la salida será la suma de las respuestas obtenidas si las entradas se presentan individualmente (propiedad de superposición).

- La respuesta a una entrada dada no varía con respecto al tiempo.

El comportamiento de un SLIT puede ser descrito en forma precisa y la salida puede ser representada en forma única, en función de la entrada y la respuesta del sistema a un impulso. La respuesta a impulso del sistema es la característica que lo define, puesto que cualquier señal que se aplique en la entrada puede ser representada como una suma de impulsos. Una vez definida la respuesta a uno de estos impulsos y por las propiedades mencionadas anteriormente, es posible determinar la respuesta del sistema a cualquier tipo de señal.

2. LA FUNCION DE CONVOLUCION

Una de las funciones más empleadas en el análisis de señales y sistemas es la función impulso, representada por $\delta(t)$, que puede verse como un pulso muy angosto, con área unitaria. Algunas de sus propiedades son:

$$\int_{-\infty}^{\infty} \delta(t)\, dt = 1$$

$$\int_{-\infty}^{\infty} \delta(t)\, x(t)\, dt = x(\emptyset)$$

$$\int_{-\infty}^{\infty} \delta(t - t_0)\, x(t)\, dt = x(t_0)$$

Esta última propiedad de corrimiento nos permite representar cualquier señal $x(t)$ mediante la suma (integral) de una serie de impulsos. Esto puede verse haciendo un simple cambio de variable en la expresión anterior:

$$\int_{-\infty}^{\infty} x(\tau)\, \delta(t-\tau)\, d\tau = x(t)$$

Considérese un SLIT, cuya respuesta a impulso, $h(t)$, es conocida. Si se aplica a la entrada la señal $x(t)$, representada en forma de suma de impulsos, se tendrá a la salida la suma de las respuestas a cada uno de ellos, es decir:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau$$

Esta expresión se conoce como convolución contínua y muestra una relación única entre la entrada y la salida de un sistema lineal invariante en el tiempo. La convolución es una operación conmutativa:

$$y(t) = \int_{-\infty}^{\infty} x(t-\tau)\, h(\tau)\, d\tau$$

Convencionalmente, la operación de convolución se representa con un asterisco:

$$y(t) = x(t) * h(t) = h(t) * x(t)$$

Análogamente, se puede realizar la operación de convolución tomando señales discretas:

$$y(n) = T \sum_{k=-\infty}^{\infty} x(k)\, h(n-k) = T\, x(n) * h(n)$$

donde y(n), x(n) y h(n) representan muestras de las señales analógicas y(t), x(t) y h(t) respectivamente, tomadas a un intervalo de muestreo T.

La función de convolución puede ser representada también en el dominio de la frecuencia, aplicando la transformación de Fourier:

$$y(n) = T \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$$

$$Y(e^{jw}) = T \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x(k)\, h(n-k) \right) e^{-jwn}$$

Sea $m = n-k$; $n = m+k$

$$Y(e^{jw}) = T \sum_{m=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x(k)\, h(m)\, e^{-jw(m+k)}$$

Arreglando términos se tiene:

$$Y(e^{jw}) = T \underbrace{\sum_{m=-\infty}^{\infty} h(m)\, e^{-jwm}}_{H(e^{jw})} \underbrace{\sum_{k=-\infty}^{\infty} x(k)\, e^{-jwk}}_{X(e^{jw})}$$

$$Y(e^{jw}) = T\ H(e^{jw})\ X(e^{jw})$$

Esto es, la operación convolución entre dos funciones en el dominio del tiempo corresponde, en el dominio de la frecuencia, al producto de las transformaciones de las funciones. Este interesante resultado, conocido como teorema de la convolución, puede simplificar en algunos casos la evaluación de la salida de un sistema.

3. DESCRIPCION DEL ALGORITMO

La opción (C) del programa principal permite realizar la convolución de las secuencias contenidas en los arreglos A y B. El resultado de la convolución, cuya longitud es igual a la suma de las longitudes de las secuencias A y B, menos 1 (para nuestro caso 1023), se deposita en el arreglo R de resultados. La función de convolución discreta, definida anteriormente, es de evaluación directa con simples sumas y productos. El algoritmo es muy sencillo y solamente requiere del parámetro correspondiente al intervalo de muestreo (T) de las señales que se desean convolucionar. El programa realiza la siguiente secuencia:

a. Pregunta intervalo de muestreo

b. Calcula la sumatoria correspondiente para cada punto de la convolución

c. Multiplica el resultado por el intervalo de muestreo

d. Deposita la secuencia en el arreglo de resultados

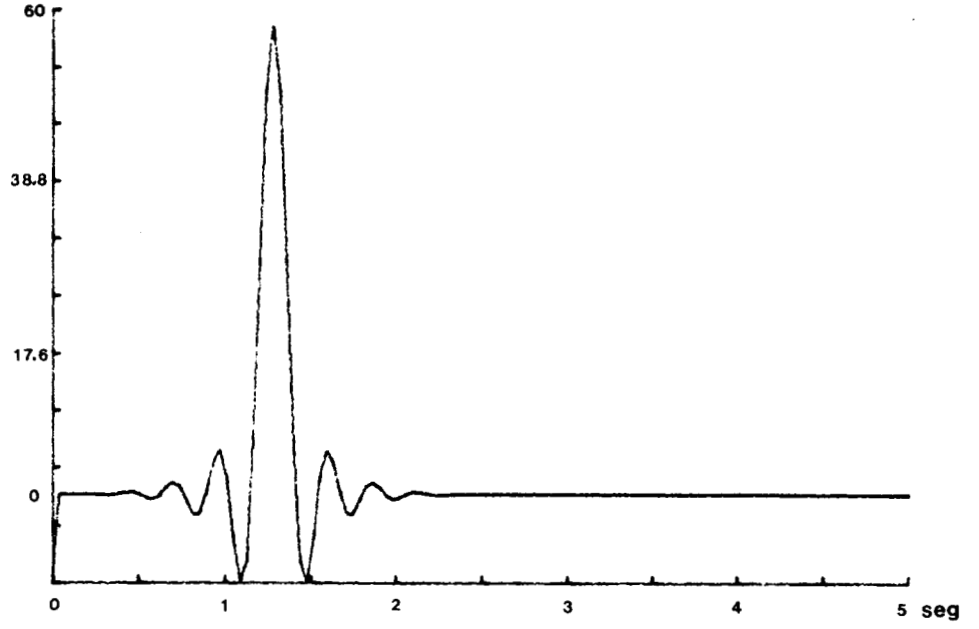
Por el número de operaciones que se deben realizar, el cálculo de la convolución es muy lento, por lo que en algunos casos resulta más conveniente realizar la operación en el dominio de la frecuencia.

4. EJEMPLO

Considérese el caso de un filtro diseñado con la opción de filtrado (pasa bajas, con frecuencia de corte 30 Hz), cuya respuesta a impulso ha sido colocada previamente en el arreglo A (figura 3.24). En el arreglo B se ha depositado una señal electrocardiográfica, obtenida a 200 Hz (figura 3.25). Al realizar la convolución, se eliminarán las frecuencias por encima de la frecuencia de corte, de acuerdo a la función de transferencia del filtro diseñado. Al elegir la opción (C) aparece en la pantalla la secuencia de la figura 3.23. El resultado de la convolución, depositado en el arreglo R, que corresponde a la señal filtrada se muestra en la figura 3.26.

```
ESTA RUTINA REALIZA LA CONVOLUCION ENTRE LOS DATOS
CONTENIDOS EN LOS ARREGLOS A Y B

¿Están listos sus datos (S/N)?                    S

¿Intervalo de muestreo (seg)?                     .005

¿Número de puntos del arreglo A?                  65

¿Número de puntos del arreglo B?                  512

LISTO. Su resultado se encuentra en el arreglo R.

¿OPCION?
```

Figura 3.23 Pantalla de la rutina de convolución



(a)

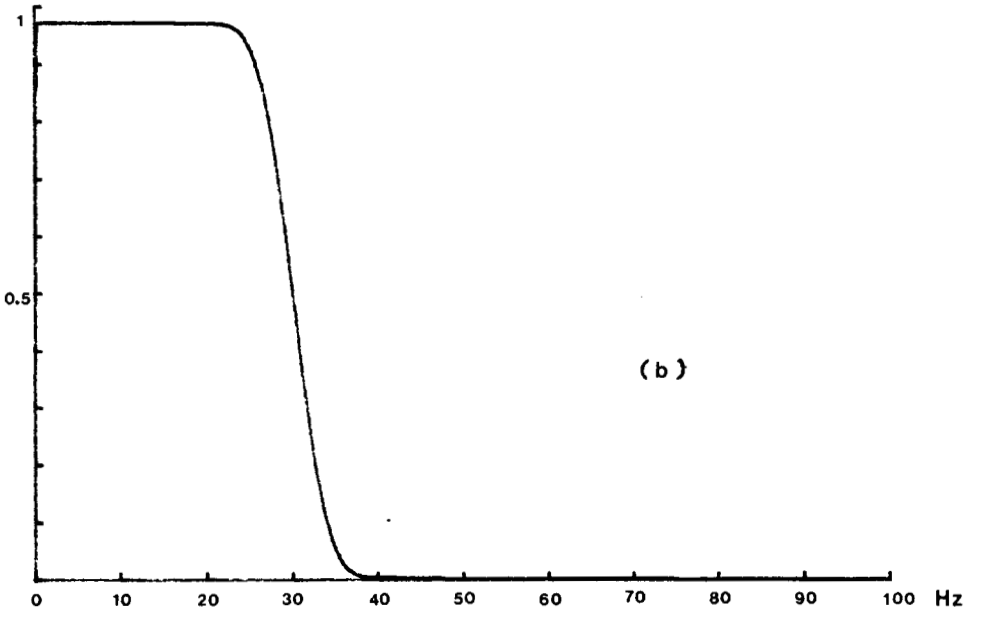
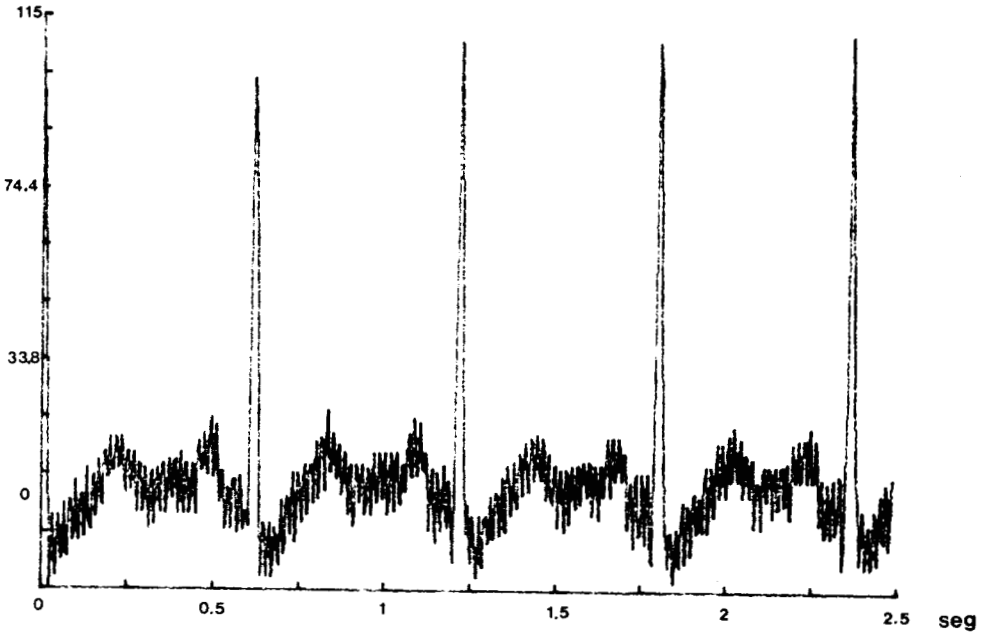


Figura 3.24 (a) Respuesta a impulso de un filtro pasa bajas, con frecuencia de corte 30 Hz, truncado con una ventana de Hamming y con una atenuación cosenoidal de 30 %
(b) Respuesta en frecuencia de (a)



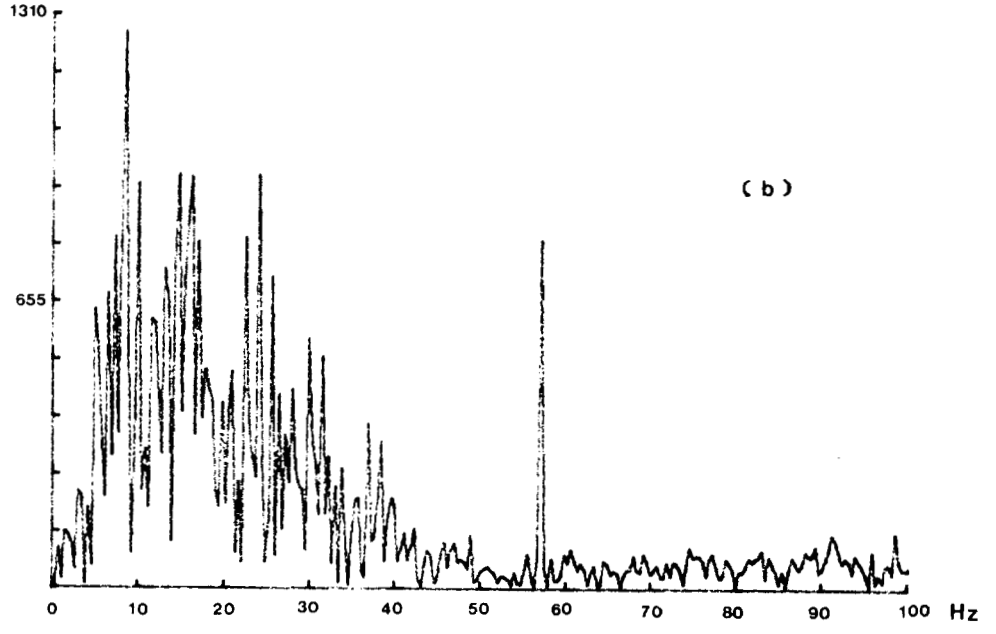
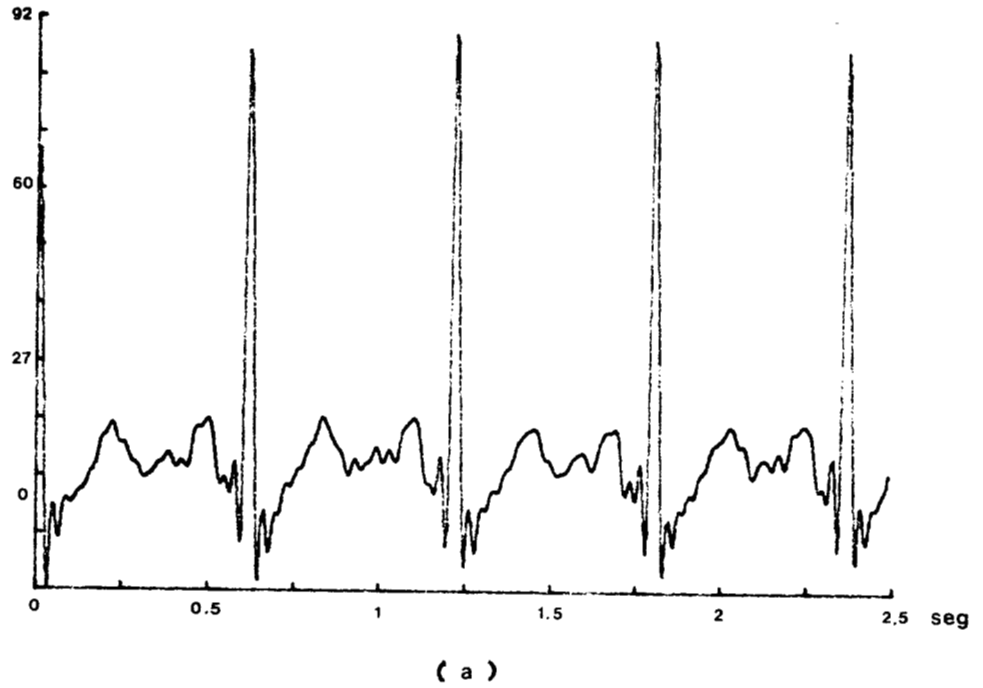


Figura 3.25 (a) Señal electrocardiográfica adquirida a una frecuencia de muestreo de 200 Hz
(b) Transformada de Fourier de (a)



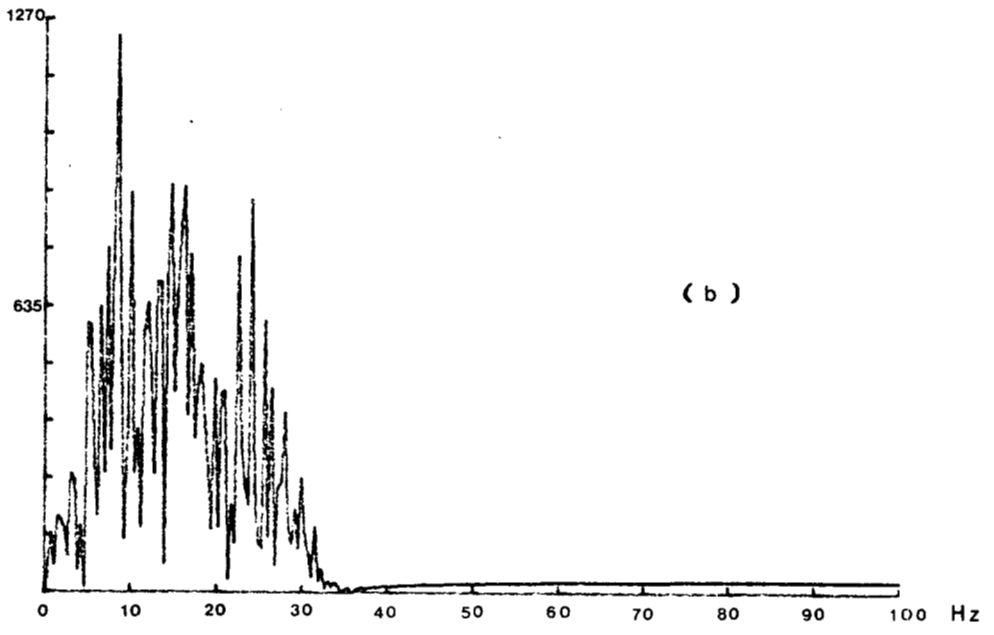


Figura 3.26 (a) Señal electocardiográfica de la figura 3.25(a) pasada por el filtro de la figura 3.24
(b) Transformada de Fourier de (a)

IV. RESULTADOS

Las aplicaciones de un paquete de procesamiento digital de señales biomédicas, como el presentado en este trabajo, son múltiples. Baste mencionar el uso que puede tener en señales electrocardiográficas para la detección de diferentes patologías cardíacas (Uijen, 1979), en señales impedancimétricas para la determinación del gasto cardíaco (Muzi, 1986), en señales electromiográficas para el cálculo del movimiento en diferentes articulaciones (LeFever, 1982), etc.

Del empleo de este paquete se puede mencionar la experiencia en dos diferentes aspectos:

- Comprobación del funcionamiento de las rutinas

- Aplicación en el procesamiento de señales adquiridas

Respecto al primer punto, el funcionamiento del programa ha sido comprobado por comparación con un sistema de adquisición y procesamiento de señales, marca Digital (PDP-11), cuya arquitectura está diseñada para realizar diversas funciones de procesamiento en forma rápida. Se compararon las siguientes rutinas:

1. Transformada rápida de Fourier
2. Convolución

Para la comprobación de las otras rutinas, se empleó un paquete desarrollado en la Universidad de Purdue (McGillem, 1980), escrito en lenguaje Fortran:

3. Filtros
4. Densidad Espectral
5. Correlación y Autocorrelación

Asimismo, algunas rutinas fueron probadas con señales típicas, cuyos resultados podían predecirse. Este proceso genera confianza en el funcionamiento del paquete.

1. Densidad Espectral (Figura 4.1)
2. Transformada rápida de Fourier (Figuras 4.2 y 4.3)
3. Filtrado (Ver sección de filtrado)
4. Convolución (Figura 4.4)

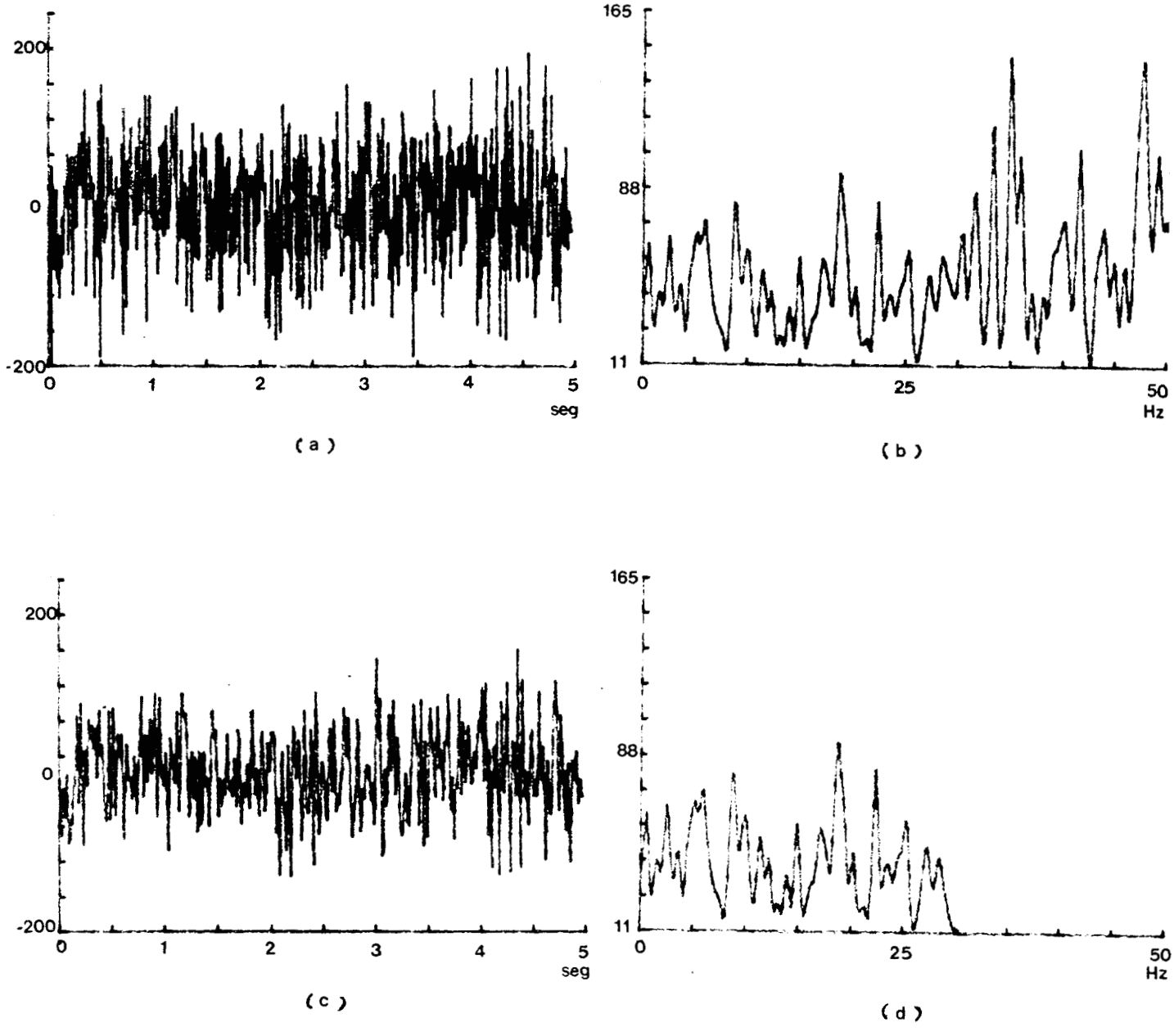
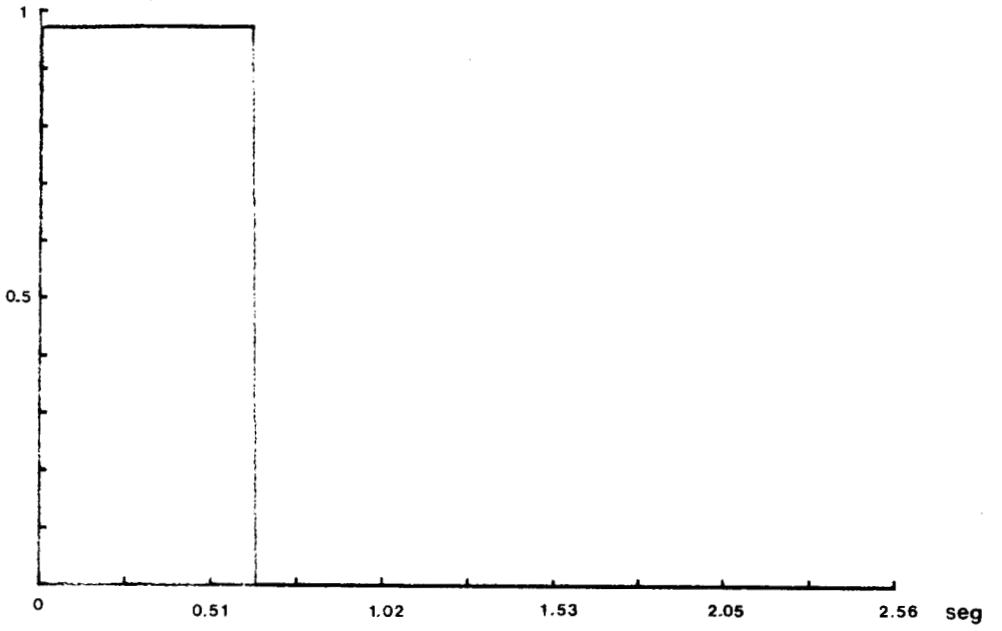
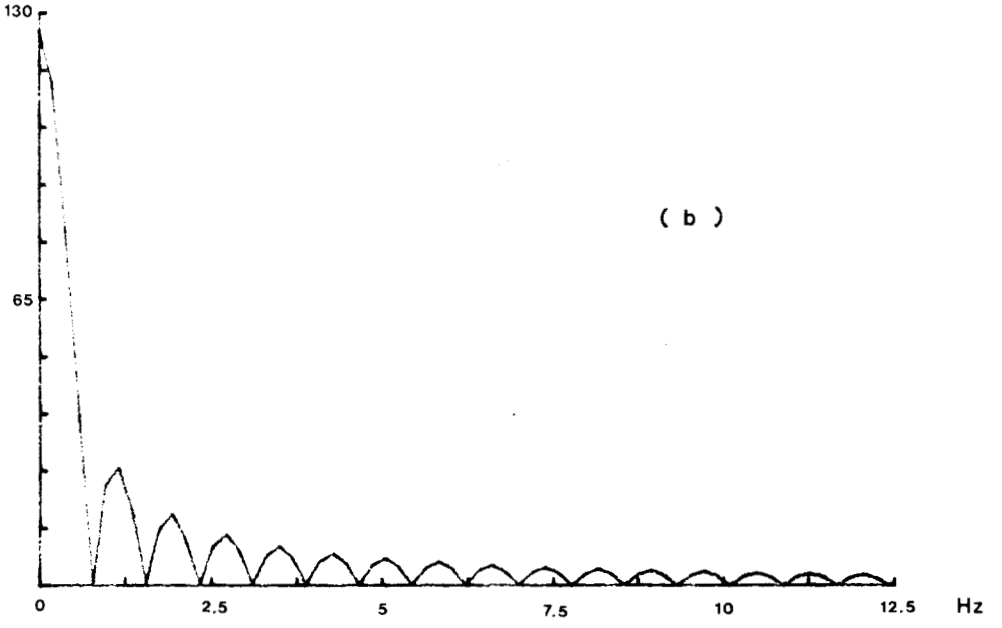


Figura 4.1 (a) Señal de ruido aleatorio
(b) Densidad espectral de (a)
(c) Señal de ruido aleatorio filtrada a 30 Hz
(d) Densidad espectral de (c)



(a)



(b)

Figura 4.2 (a) Función cuadrada depositada en un arreglo de datos
(b) Magnitud de la transformada de Fourier de (a)

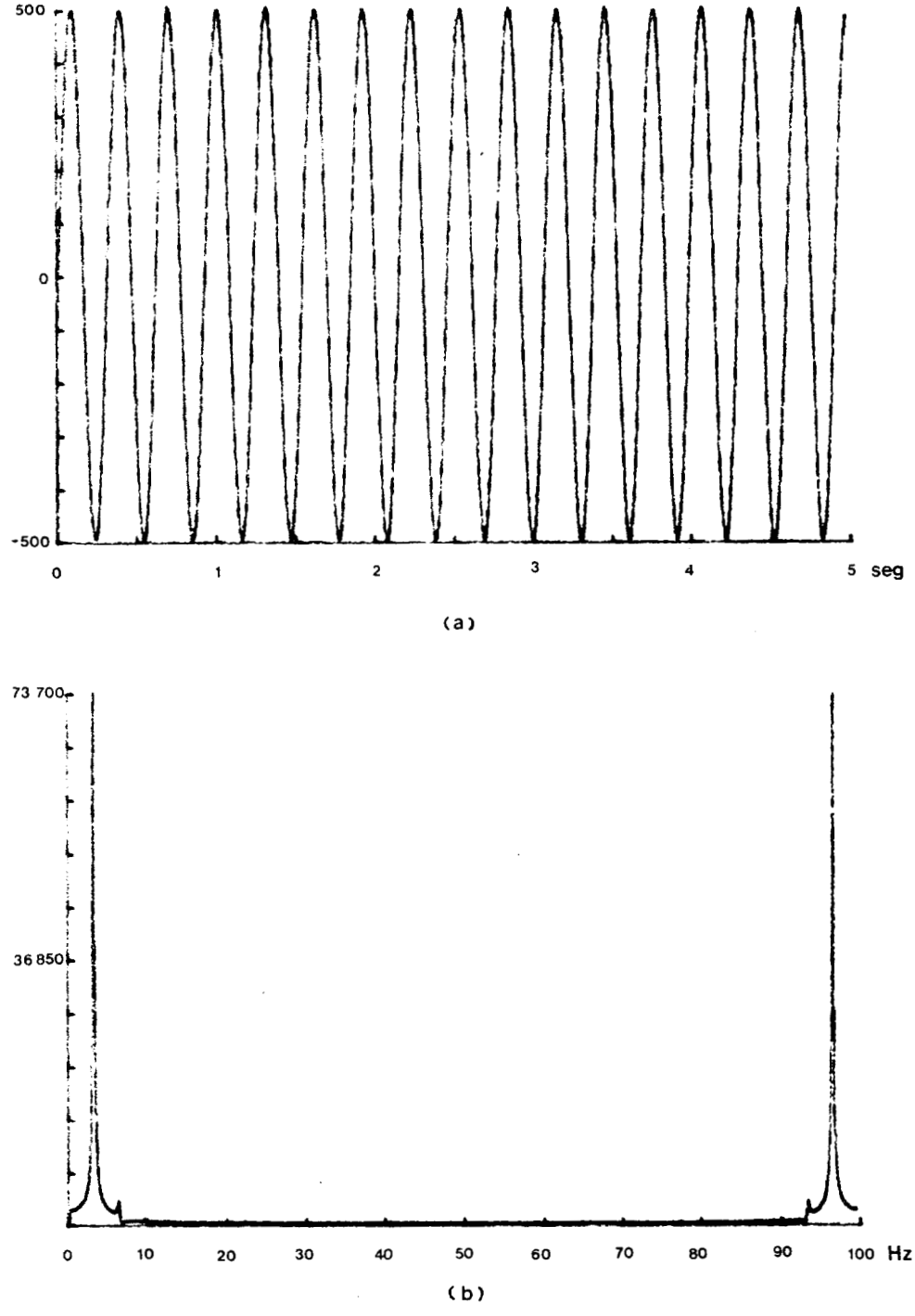
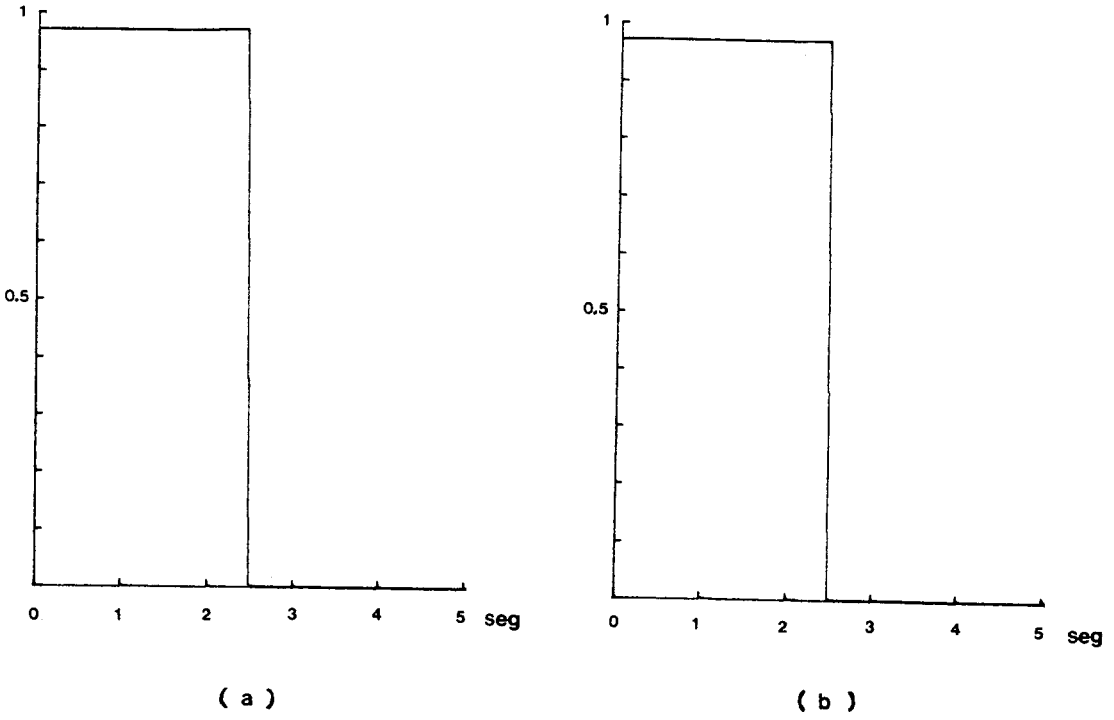


Figura 4.3 (a) Señal senoidal, frecuencia de muestreo 200 Hz
(b) Magnitud de la transformada de Fourier de (a)



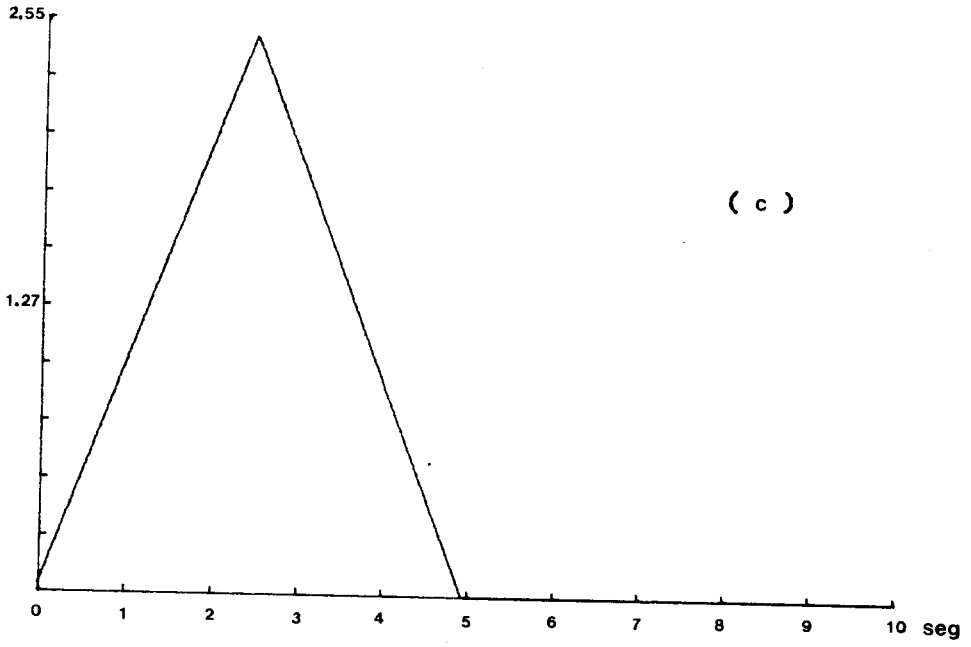


Figura 4.4 (a),(b) Función cuadrada depositada en los arreglos A y B
(c) Resultado de la convolución entre (a) y (b)

Respecto al procesamiento de señales adquiridas con el sistema, se trabajó con el electroencefalograma (EEG), para el desarrollo de un sistema de mapeo cerebral. Clínicamente, esta señal representa las variaciones de la actividad eléctrica cerebral en diferentes condiciones del sujeto. La figura 4.5 presenta una señal típica de EEG en una persona normal en estado de reposo; cada canal muestra una región diferente sobre la corteza cerebral, de acuerdo a una configuración establecida para la colocación de los electrodos. La interpretación de este tipo de señales es difícil puesto que no existe un patrón o patrones establecidos para diferentes conductas o estados funcionales del cerebro. Sin embargo, se ha establecido una clasificación gruesa del contenido en frecuencias de la señal, centrado en cuatro bandas principalmente, que facilita el análisis de la señal y que permite establecer situaciones funcionalmente anormales (Most, 1980). Hasta ahora, el método más empleado para obtener una estimación del contenido en frecuencias del EEG en estas cuatro bandas de interés ha sido la determinación de los intervalos de tiempo más visibles en el trazo original. Este método, además de introducir errores en la estimación de las frecuencias, se ve afectado por un gran componente subjetivo, que depende de la experiencia del interpretador. Se han propuesto diversos tipos de representación del EEG para disminuir estos problemas y actualmente existen varios sistenas que permiten realizar un análisis automático de la energía de la señal contenida en las bandas de interés (Itil, 1985).

Con estos antecedentes, en el área de Ingeniería Biomédica de la UAM-Iztapalapa, se inició el diseño y construcción de un sistema para la adquisición y el análisis automáticos del EEG, para estados hipofuncionales, con las siguientes características:

- Conversión de 12 canales de entrada, a frecuencia de muestreo fija (102.4 Hz, para un intervalo de 5 seg en 512 muestras).

- Cálculo de la energía contenida en las siguientes bandas de frecuencia:

  Delta: 0.5 - 3.9 Hz
  Theta: 4 - 7.9 Hz
  Alfa: 8 - 12.9 Hz
  Beta: 13 - 30 Hz

- Presentación adecuada de los resultados, para facilitar su interpretación.

Para el desarrollo de dicho sistema, se requirió de una etapa de prueba en cuanto al número de datos que se deseaba adquirir, el tipo de procesamiento que resultaba más adecuado y la forma idónea para la presentación de resultados. Particularmente, en los dos primeros aspectos, fue de gran utilidad contar con el paquete de procesamiento (PROCESA), que nos permitió realizar todas las pruebas con unos cuantos canales y, posteriormente, transferir las rutinas para su adecuación al número de canales deseados. En base a las pruebas realizadas, se fijó la frecuencia de muestreo en 102.4 Hz, para tener un intervalo de 5 seg exactos en 512 puntos; dadas las características de la señal, 5 seg de información son suficientes para el cálculo de la energía. En la parte de procesamiento, se encontró que la densidad espectral entrega una mejor aproximación del contenido en frecuencias del EEG, y además es una función más "suave", en comparación con la transformada de Fourier. Esta última característica facilita la integración de la función para el cálculo de la energía en las bandas que se quieren analizar (Figura 4.5). Finalmente, las cuatro energías calculadas para cada canal se muestran de acuerdo a su localización anatómica, con una asignación de tono proporcional y con un color diferente para cada banda (Muñoz, 1986).

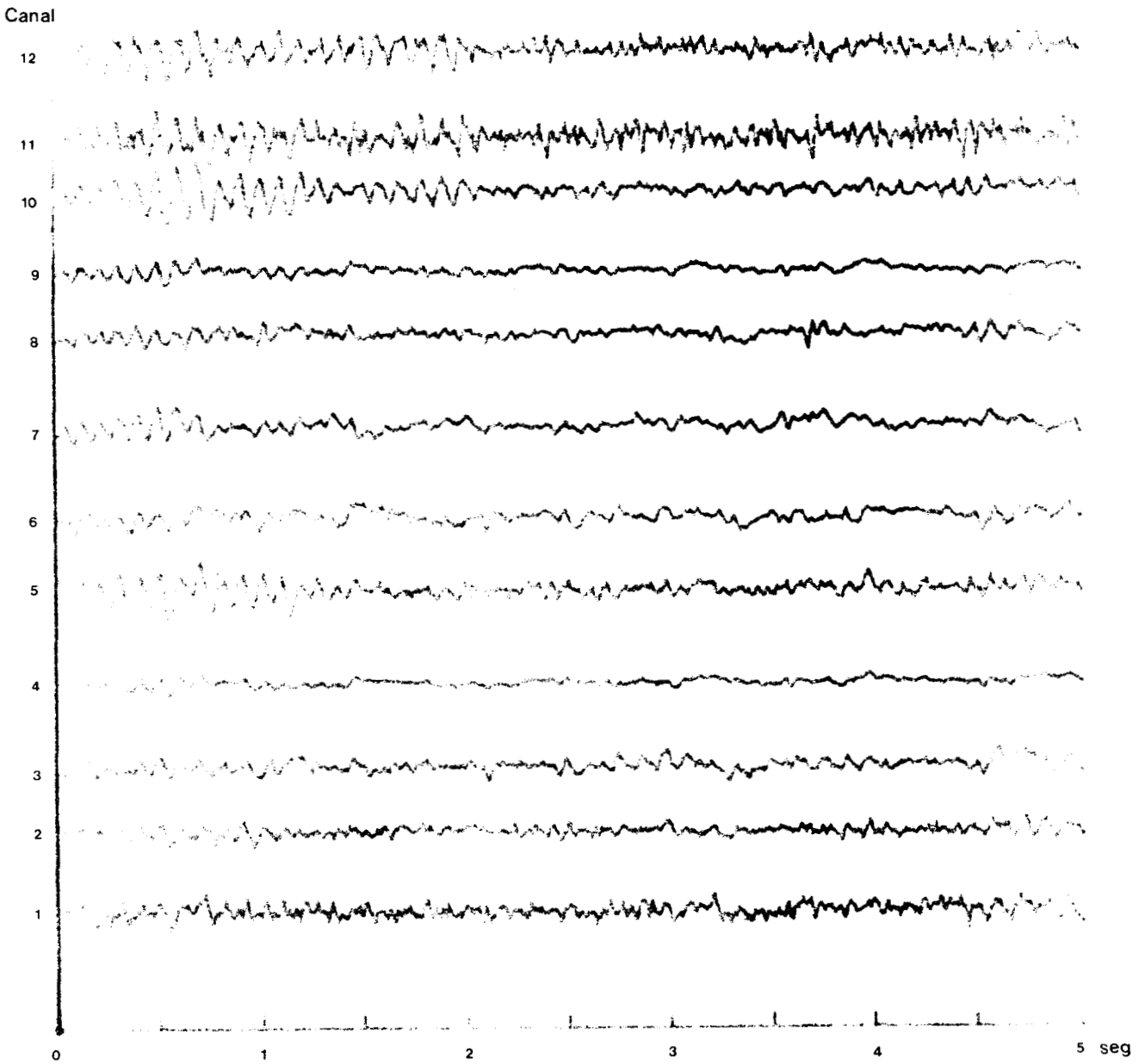


Figura 4.5 (a) Señal electroencefalográfica, frecuencia de muestreo 102.4 Hz

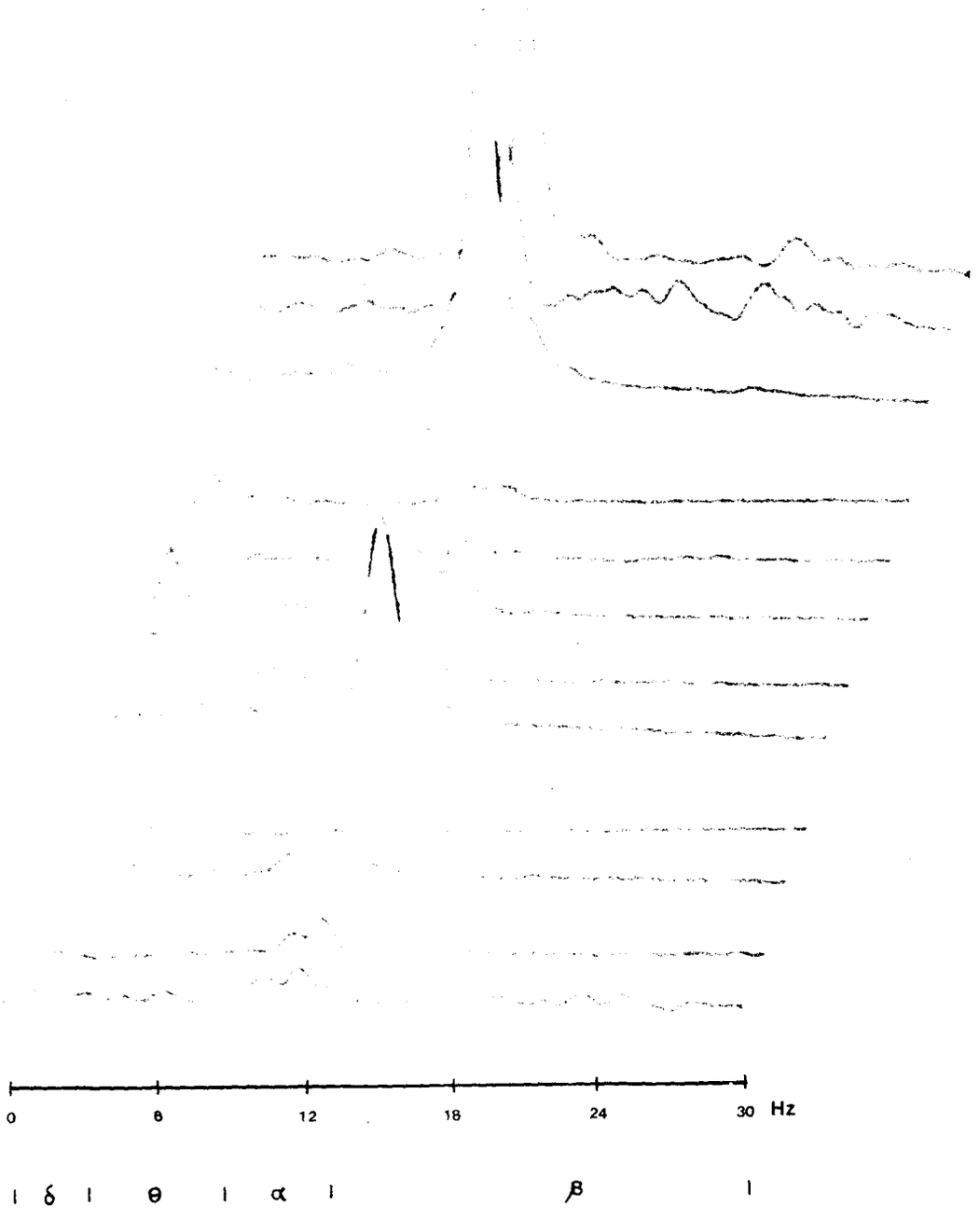


Figura 4.5 (cont.) (b) Densidades espectrales de (a)

V. CONCLUSIONES

El paquete de programas presentado constituye una herramienta basica para la adquisición y el análisis de diversos tipos de señales biomédicas. En este documento se ha incluido los motivos y criterios para su desarrollo, la información necesaria para su manejo y reproducción, la teoría básica para el entendimiento de cada una de las rutinas y la evaluación de los algoritmos propuestos. Los resultados obtenidos permiten apreciar el potencial de aplicaciones que tiene este paquete, principalmente en la creación de programación más específica para otros sistemas de procesamiento de señales biomédicas.

Como todo trabajo, y aunque los objetivos planteados han sido cubiertos en su totalidad, el paquete es susceptible de ser mejorado en muchos aspectos. La inclusión de un número mayor de algoritmos permitiría aumentar sus aplicaciones. Particularmente, se pueden sugerir los siguientes bloques:

- Rutinas de filtrado digital por otros métodos, que permitirían elegir la técnica más adecuada, dependiendo de la aplicación.

- Rutinas de despliegue con empleo de cursores y capacidad de extensión en ambos ejes. Esto es particularmente importante cuando se desea analizar la señal punto a punto y comparar diferentes funciones.

- Rutinas de operaciones aritméticas y de derivación e integración de arreglos. Como ejemplo de su importancia, basta considerar el caso de filtrado en el dominio de la frecuencia, a través del producto de dos espectros, o el cálculo de áreas para la determinación de un parámetro de la señal.

- Rutinas de aproximación de una señal a través de modelos matemáticos. El caso más empleado actualmente son los modelos autorregresivos, que han ganado gran importancia en el reconocimiento de patrones, aplicado a diversos tipos de señales biomédicas.

Por otro lado, la presentación del paquete puede ser mejorada, en cuanto al manejo de menús, presentación de información y entrada de variables. En este último caso, el contar con una rutina de validación de formato para el ingreso de las variables, permitiría al usuario sentirse seguro y cómodo con el manejo del paquete.

Respecto a la experiencia que se ha tenido con el paquete en el área de docencia, es necesaria la elaboración de un conjunto de prácticas que permitan al alumno "experimentar" con cada una de las funciones que se ofrece, aunque esto implicaría cambiar un poco la presentación de los datos.

Finalmente, para aplicaciones futuras se puede pensar en incrementar la velocidad de ejecución de algunas rutinas. Algunas de las rutinas instaladas en este momento pueden ser aceleradas fácilmente, con algún cambio en el manejo de los datos. Por ejemplo, es posible aplicar la transformación de Fourier a dos señales reales simultáneamente, considerando una de ellas como la parte real y otra como la parte imaginaria de un arreglo. Del resultado obtenido después de la transformación, es posible separar el espectro correspondiente a cada una de las funciones de entrada, ahorrando así el tiempo de ejecución para el cálculo de una transformada. Otra opción para aumentar la velocidad consiste en instalar algunas rutinas en lenguaje ensamblador; particularmente, aquellas que incluyen operaciones lentas para la microcomputadora (por ejemplo, la convolución). Por último , conviene considerar el uso de circuitos que permitan acelerar el funcionamiento de toda la microcomputadora, como es el caso de los coprocesadores comerciales o de diseño específico para procesamiento digital de señales.

BIBLIOGRAFIA

Antoniou, A.
Digital Filters: Analysis and Design
McGraw Hill Book Company, New York, 1979

Azpiroz, J., Gómez, S.
"Sistema de Adquisición de Señales Biomédicas"
Memorias del XII Congreso de la Academia Nacional de Ingeniería, Sept. 1986

Brigham, E.O.
The Fast Fourier Transform
Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1974

Byte, The Small Systems Journal
Guide to the IBM Personal Computers
McGraw Hill, Vol. , No. 9, 1984

Chen, Chi-Tsong
One-Dimensional Digital Signal Processing
Marcel Dekker, Inc., New York, 1979

Cooley, J.W., Tukey, J.W.
"An Algorithm for the Machine Calculation of Complex Fourier Series"
Mathematics of Computation, Vol. 19, No. 90, 1965.

Data Physics Corporation (DPC)
Signalcalc: Signal Processing Software
Technical Information

DPO TEK Basic Tektronix
CP65511, CP65521, CP65681 Users Instruction Manual
Tektronix, Inc., 1975

Good, I.J.
"The Interaction Algorithm and Practical Fourier Analysis"
J. Royal Statistical Society, Ser. B, Vol. 20, 1968

IEEE Acoustics, Speech and Signal Processing Society
Programs for Digital Signal Processing
Digital Signal Processing Society, IEEE Press, 1979

Itil, T., Shapiro, D., Eralp, E., Akman, A., Itil, K., Garbizu, C
"A new brain function diagnostic unti, including the dinamic brain mapping of computer analyzed EEG, evoked potential and sleep" New trends in experimental and clinical psychiatry. Vol.1 No. 2, 1985

LeFever, R.S., DeLuca, C.J.
"A procedure for decomposing the myoelectric signal into its constituent action potentials"
IEEE Transactions on Biomedical Engineering, Vol. 29, 1982

McGillem, C.D., Auñón, J.I.
Signal Processing of Event Related Brain Potentials
Purdue University, West Lafayette, Indiana, 1981

Most, M.P., Craib, A.R.
EEG Handbook, 2nd. Edition
Beckman Instruments Inc., 1975

Muñoz, C.
"Procesamiento y presentación de resultados en el análisis automático del EEG"
Revista Mexicana de Ingeniería Biomédica, Vol.7 No.2, 1986

Muzi, M.
"Computer-automated impedance derived cardiac indexes"
IEEE Transactions on Biomedical Engineering, Vol. 33, No.1, 1986

Oppenheim, A.V. (Ed.)
Applications of Digital Signal Processing
Prentice Hall, Inc. Englewood Cliffs, New Jersey, 1978

Oppenheim, A.V., Schafer, R.W.
Digital Signal Processing
Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1975

Programa Nacional de Educación Superior (PRONAES)
Propuesta para la creación de un Laboratorio de Investigación en Computación y Procesamiento de Señales (LICPOS), 1983

Rabiner, L.R., Gold, B.
Theory and Application of Digital Signal Processing
Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1975

Rudnick, P.
"Notes on the Calculation of Fourier Series"
Mathematics of Computation, Vol. 20, Junio, 1966

Signal Technology, Inc.
ILS-PC: Interactive Laboratory System
Technical Information

Tompkins, W.D., Webster, J.G.
Design of microcomputers-based medical instrumentation
McGraw Hill, 1981

Uijen, G.J., de Weerd, J.P., Vendrik, A.J.
"Accuracy of QRS detection in relation to the analysis of high frecuency components in the electrocardiogram"
Medical and Biological Engineering and Computing, Vol.17, 1979

Usi, S., Amridror, I.
"Digital Low Pass Diferentiation for Biological Signal Processing"
IEEE Transactions on Biomedical Engineering, Vol. 29, No. 19, Oct. 1982

Yeung, P., Kim, Y.
"Computer-aided Design of Digital Filters and their Simulation"
Proceedings of the Eigth Annual Conference of the IEEE-EMBS, 1986

APENDICE A

CIRCUITO DE CONVERSION ANALOGICA DIGITAL

SISTEMA DE ADQUISICION DE SEÑALES BIOMEDICAS

Joaquín Azpiroz Leehan y Sonia Gómez Díaz Ceballos
Area de Ingeniería Biomédica
Departamento de Ingeniería Eléctrica
Universidad Autónoma Metropolitana-Iztapalapa
Av. Purísima y Michoacán 09340 México D.F.

ABSTRACT

An acquisition system for biomedical signals is presented. It is designed to eliminate the need for continous recording on paper and the traditional analog processing techniques. The system is based on a PC compatible computer and is complemented by a signal processing package which was designed to implement an acquisition and processing unit that incorporates all the advantages of digital signal processing. This system has 12 imput and 12 output channels with 12 bit resolution. In addition, data acquisition and storage are handled by interrupts, in order to perform signal analysis in real time.

RESUMEN

Se presenta un sistema de adquisición para señales biomédicas. Su diseño está basado en la necesidad de eliminar el registro contínuo en papel y las técnicas convencionales de procesamiento analógico en aplicaciones de docencia e investigación. Este sistema utiliza un marco básico consistente en una computadora tipo PC y se complementa con un sistema de procesamiento de señales biomédicas. El sistema cuenta con 12 canales de entrada, 12 canales de salida y una resolución de 12 bits. Además, utiliza manejo de interrupciones para adquirir los datos mientras parte del procesamiento se lleva a cabo simultáneamente.

INTRODUCCION

Desde fines del siglo pasado, el registro y análisis de señales bioeléctricas ha sido una herramienta poderosa para la clínica médica. Conforme ha avanzado la instrumentación electrónica, se ha hecho posible la detección y el registro de un mayor número de fenómenos fisiológicos que aportan información importante acerca de los procesos vitales de los organismos a quienes se registran.

Tradicionalmente, estos registros se han hecho de manera analógica (inicialmente con quimógrafos, luego con polígrafos) y el análisis de la señal ha sido primordialmente en el dominio del tiempo. Los filtros que se han utilizado tradicionalmente para el filtrado de estas señales son redes RC y amplificadores operacionales, pero la poca exactitud en los valores de los componentes y su variabilidad en el tiempo limitan el uso de estos filtros a órdenes inferiores a 6. Otros tipos de procesamiento, como el reconocimiento de patrones, se ha hecho de manera empírica, a partir de la experiencia adquirida por los médicos a lo largo del tiempo (como en el caso de electrocardiografía y electroencefalografía).

Con el advenimiento de las últimas generaciones de microcomputadoras personales, se ha hecho posible el contar con un sistema de adquisición y procesamiento de señales de bajo costo y con un poder computacional comparable al de una minicomputadora de hace algunos años. Un sistema de este tipo cuenta con varias ventajas sobre un sistema de registro analógico de señales biomédicas:

- Es posible eliminar el trazo contínuo sobre papel para experimentos largos donde sólo en ciertos momentos se logran registrar los fenómenos deseados. Este sistema se puede sustituir por la presentación de la señal sobre la pantalla de la computadora y el almacenamiento automático de la señal en disco. Esta puede analizarse y graficarse en caso de que se desee un registro permanente.

- El filtrado digital de las señales tiene ventajas sobre el analógico. Algunas de estas son el poder diseñar filtros ideales, filtros de grandes pendientes de corte y con fase lineal o filtros adaptativos. Los filtros digitales siempre conservan sus características, ya que no dependen de componentes que puedan variar con el tiempo.

- Simplifica el análisis convencional de las señales. Es decir, permite el cálculo de amplitudes, fases, integrales, etc. (en el método tradicional de análisis, se recurría a calcular el área bajo la curva, haciendo un recorte del registro en papel de ésta, pesándolo y comparando este peso con el de un recorte de papel que sirve de referencia). Además, permite que el análisis en el dominio de la frecuencia, lo que hace posible la visualización de información no disponible de otra manera.

El sistema que se ha diseñado, cuenta con todas las herramientas necesarias para llevar a cabo la adquisición, análisis y despliegue de información. Es capaz de transformar a un sistema de registro en una unidad de adquisición y análisis de señales idóneo para aplicaciones biomédicas.

DESCRIPCION DEL SISTEMA

I. Características del sistema de computación:

Se utilizó una computadora compatible con la computadora personal IBM (IBM-PC) debido a su poder computacional, su costo relativamente bajo y su arquitectura abierta. Esta computadora utiliza el microprocesador 8088 de INTEL, el cual tiene un canal de datos de 8 bits y una arquitectura de 16 bits orientada al uso de registros y con facilidad para el manejo de dispositivos periféricos. Conjuntamente con el 8088 se puede utilizar un procesador numérico (8087) que permite aumentar la velocidad de ejecucuón de algunos programas de manera significativa. La documentación tanto del microprocesador 8088 y sus periféricos, como de la computadora es lo suficientemente extensa como para permitir el diseño de subsistemas que se pueden conectar a las ranuras de expansión de la máquina y que se comportan como puertos de entrada/salida.

II. Características del sistema de conversión:

El sistema de adquisición de señales se diseñó de tal manera que fuera compatible con los sistemas mas comunes de registro de señales biomédicas (fisiógrafos y polígrafos). La amplitud de la señal de entrada puede ser de ± 0.5V a ± 5.0V.

Se utilizó un solo convertidor Digital/Analógico (AD565) para llevar a cabo las funciones de conversión A/D y D/A, ya que sus características de interconexión permitían adicionarle un registro de aproximaciones sucesivas (DM2504) para utilizarlo como convertidor Analógico/Digital, o un convertidor de corriente a voltaje para utilizarlo como Digital/Analógico y conmutar entre ambas configuraciones fácilmente.

Los 12 bits de resolución de este convertidor son más que suficientes para representar adecuadamente a una señal fisiológica. Es raro que se pueda obtener una señal de este tipo con una relación señal/ruido mejor a los 60 dB, lo que significa que dentro de una señal de 1 volt pico a pico de amplitud, se tiene por lo menos 1 milivolt de ruido. Aún cuando serían suficientes 10 bits de resolución (1024 niveles), se optó por utilizar un convertidor de 12 bits (4096 niveles) para poder aceptar señales con variaciones en su línea de base sin que se sature el convertidor (1).

Junto con el convertidor de 12 bits, se diseñó un sistema de multicanalización para permitir la conversión de 12 canales de entrada y la graficación de 12 canales de salida. Como el tiempo de conversión es corto (10 microsegundos), es posible adquirir 12 canales simultáneamente con una frecuencia de muestreo máxima de 8KHz por canal. Esta frecuencia de muestreo es superior a la requerida por la mayoría de las señales fisiológicas.

La programación del sistema permite que la frecuencia de muestreo varíe entre 0.1Hz y 50 KHz (disminuyendo el número de canales cuando se quiere una frecuencia de muestreo mayor a 8 KHz). El manejo de este frecuencia de muestreo se hace a través de un contador programable -- (8253), el cual da la orden de inicio de conversión a tiempos preprogramados.

La adquisición se hace por medio de interrupciones, lo que permite que el sistema de computación realice parte de las tareas de análisis de señales mientras se está llevando cabo la adquisición. Los 12 canales tanto de entrada como de salida permiten que el sistem de adquisición pueda utilizarse con la gran mayoría de sistemas de registro que existen actualmente en el mercado (Fig 1).

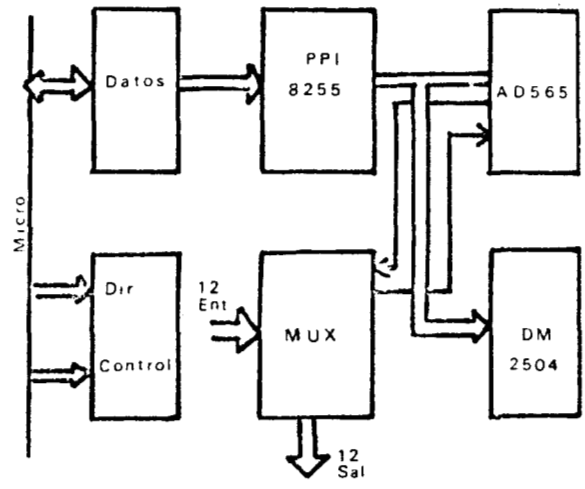


Figura 1. Diagrama a bloques del Sistema.

ANALISIS DEL CIRCUITO

I. Circuitos de acoplamiento:

Este subsistema permite el acoplamiento del convertidor con los canales de la computadora (direcciones, datos y control) a través de los receptáculos de las tarjetas de expansión. Las especificaciones de la PC indican que cada línea de la tarjeta de expansión debe presentar sólamente 1 carga TTL al canal de expansión. Debido a esto, se utilizan 2 acopladores (Buffers) tipo 74LS244 que están permanentemente habilitado y de los cuales se obtiener las líneas de direccionamiento AØ-A9 y las líneas de control de lectura y escritura en memoria y en periféricos (puertos). Las mismas especificaciones indican que los puertos 300_H al $31F_H$ están disponibles para cualquier uso, así que se utiliza una decodificación tal, que se seleccionan los puertos 310_H al $31F_H$ para el convertidor (línea DIR de control interno)(2,3) La salida de esta decodificación no absoluta se acopla a las líneas de lectura y escritura de puertos (IOR e IOW) y ésto habilita a un acoplador bidireccional (74LS245) conectado al canal de datos ((IOR+IOW)·DIR). El control de la dirección de transferencia de datos está dado por la línea IOR que se conecta directamente a la entrada DIR del 74LS245. La decodificación específica del contador programable, el registro de control (74LS174) y la interfase periférica programable (PPI 8255 que se encarg de capturar los datos al término de la conver-

sión), se hace a través de un decodificador integrado (74LS139), el cual recibe como entradas a la línea DIR (decodificación general de los puertos) y las líneas de direccionamiento A_4, A_3 y A_2. Las salidas de este circuito corresponden a los siguientes puertos:

ØA - 310_H - 313_H - 8255
1A - 314_H - 317_H - 8253
2A - 318_H - $31B_H$ - 74LS174

La salida ØA se conecta a la selección del PPI, la 1A a la selección del contador y la 2A al reloj del registro. Como el PPI y el contador se comportan como varios puertos, las líneas AØ y A1 se conectan a ambos circuitos.(Fig. 2).

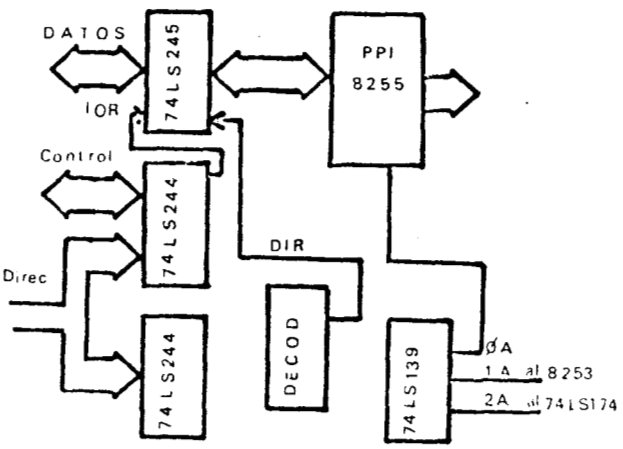


Figura 2. Circuitos de interfase.

II.Interfase periférica programable 8255

El 8255 es un circuito integrado que permite la comunicación uni o bidireccional entre la microcomputadora y los dispositivos periféricos. Consta de 24 líneas programables, organizadas en 3 puertos de 8 bits denominados A, B y C que permiten la transferencia de información de manera incondicional (modo Ø), o utilizando señales de reconocimiento (Handshaking). En este caso, se conectan 12 líneas (puerto B y parte alta del puerto A) a las líneas de datos del convertidor. Por estas líneas se mandan o se reciben los datos, dependiendo de la modalidad de la conversión (A/D o D/A).

Para la operación como convertidor A/D, se utiliza la entrada de datos con señal de reconocimiento (modo 1). Las líneas de conversión completa del sistema de conversión se conectan a un bit del puerto C. Cuando esta última línea es activada, los datos son "amarrados" por los puertos A y B y se genera una interrupción para que la microcomputadora entre a una rutina de servicio, reciba los datos y continúe con la ejecución del programa original(4).

Cuando la conversión es D/A, los datos se mandan por los puertos A y B en el modo incondicional (modo Ø) y estos se transforman en un nivel de voltaje a la salida del AD565.

III. Contador programable:

Este circuito tiene 3 contadores programables integrados, de 16 bits cada uno. Su función es dar la frecuencia de muestreo adecuada. Para esto, divide la frecuencia del reloj de la microcomputadora entre el número adecuado para dar la señal de inicio de conversión al registro de aproximaciones sucesivas. Una vez programado, da la frecuencia de muestreo de manera automática.

IV. Registro de Control:

El registro 74LS174 se utiliza para almacenar la palabra de control que indica al sistema de conmutación y a los multiplexores qué tipo de conversión se va a efectuar (A/D o D/A y cual va a ser el canal se salida. Este circuito reconoce la palabra de control cuando la microcomputadora manda la instrucción de salida al puerto 317_H. La información queda a la salida del registro hasta que se manda una orden de cambio en la palabra.

V. Canales de entrada y salida:

Los canales de entrada del convertidor utilizan seguidores de voltaje a partir de amplificadores operacionales TL074, que permiten un desacoplamiento de impedancia y una buena adquisición sin pérdida de la señal analógica proveniente del fisiógrafo o del amplificador utilizado. Estos 12 canales son multiplexados por unos circuitos CMOS 4051 y 4052, controlados por una palabra digital proveniente del registro 74LS174, el cual activa a un multiplexor, inhibe al otro y selecciona al canal que será conectado al amplificador; las salidas de los 2 multiplexores están conectadas entre sí, ya que al inhibir a uno de ellos, éste pone su salida en estado de alta impedancia y no interfiere con la operación del multiplexor activo. La señal de salida de los multiplexores se conecta a un amplificador con ganancia variable que permite llevar la señal de entrada a un nivel de voltaje adecuado para que la resolución de la señal muestreada por el convertidor sea óptima.

En el caso de la conversión Digital/Analógica, la señal del convertidor de corriente a voltaje se conecta al multiplexor 4052 que funciona de manera bidireccional o a un 4051, permitiendo la salida de la señal analógica por cualquiera de los 12 canales de salida (4 por el 4052 y 8 por el 4051). El control de estos multiplexores está dado por el registro 74LS174

VI. Convertidor:

Para construir la parte de conversión, se utilizo el convertidor AD565, que por medio del manejo de unos interruptores, permite la operación en modo A/D o D/A (5).

- Conversión Digital/Analógica:
  La conversión D/A se llevó a cabo por medio de la conexión de un circuito convertidor de corriente a voltaje, que da a su salida una señal analógica de voltaje que puede variar entre -5 y +5 volts.

- Conversión Analógica/Digital:
  Este tipo de conversión se lleva a cabo mediante la conexión de un registro de aproximaciones sucesivas (DM2504) con el AD565. ESte circuito obliga al AD565 a producir una señal analógica de salida que se compara con la entrada analógica que se quiere convertir, a través de un comparador integrado (LM311). Este registro da una señal de conversión completa al PPI, de tal manera, que éste genera una interrupción que obliga a la computadora a adquirir el dato digitalizado.

La selección del modo de operación es sencilla. Está controlada por un multiplexor 4052 que conmuta de una configuración a otra a partir de 2 líneas que en un caso se conectan del convertidor al comparador, y en otro del AD565 al convertidor de corriente a voltaje. El registro de control maneja la conmutación(Fig.4).

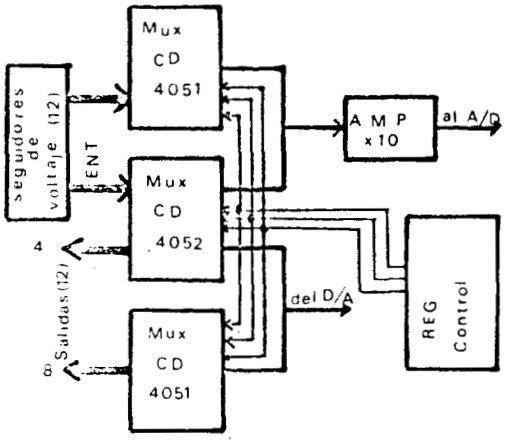


Figura 3. Multiplexores de entrada y salida.

PROGRAMACION

El sistema de conversión de datos es reconocido por la computadora como una serie de puertos. La información que se escribe en ellos controla al convertidor y los resultados se leen de estos. Su mapeo se indica a continuación:

PUERTO	DISPOSITIVO
310	Puerto A del PPI
311	Puerto B del PPI
312	Puerto C del PPI
313	Puerto de control del PPI
317	Registro de control 74LS174
318	Contador Ø del 8253
319	Contador 1 del 8253
31A	Contador 2 del 8253
31B	Registro de control del 8253

La programación de la conversión de datos consta de 3 módulos básicos:

I. Programación de los periféricos
   A. Operación del PPI

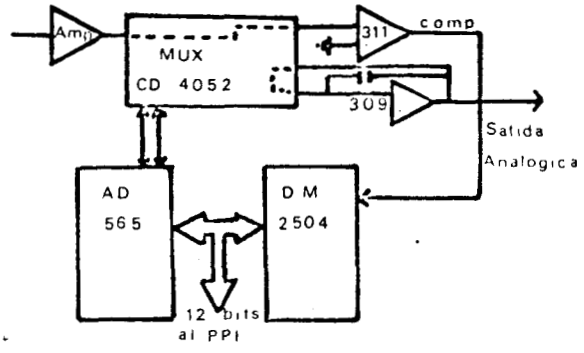


Figura 4. Multiplexor selector de tipo de operación.

   B. Programación de la frecuencia de muestreo a través del 8253

II. Programación del canal y del tipo de conversión

III. Programación de la rutina de interrupción de la microcomputadora

Dentro de la sección de programación de los periféricos, se programa al PPI para que adquiera los datos automáticamente en cuanto reciba la señal de conversión completa. Para esto, se programa a los puertos A y B en modo 1. También se hace la programación para que a través del control de una de las líneas del puerto C se habilite una interrupción en cuanto se reciban los datos (modo bit set-reset).

El canal y el tipo de conversión se selecciona por medio de la escritura de un dato en el puerto 317_H (correspondiente al registro 74LS174). La parte baja de ese dato (DØ-D3) indica el canal seleccionado, mientras que el bit D4 indica el tipo de conversión (A/D para el bit=Ø y D/A para el bit=1).

Al adquirirse el dato, se genera una interrupción, que es reconocida por la microcomputadora. El contador de programa de ésta brinca a una dirección especificada con anterioridad donde se tiene el programa de almacenamiento de datos. El dato recientemente convertido se lee de los puertos A y B, se ajusta a un formato de 16 bits y se almacena. Al finalizar la rutina, se continúa con el programa en ejecución antes de que se generara la interrupción (6,7).

APLICACIONES

Dentro de las aplicaciones que se contemplan para esta unidad, está la sustitución de los fisiógrafos o polígrafos que se utilizan para registrar varias señales fisiológicas simultáneamente. Estos equipos, cuya aplicación principal es en docencia, pueden ser sus

tituidos por el equipo anteriormente descrito al construir los preamplificadores adecuados. El costo de la utilización de esta nueva tecnología es menor al de la adquisición de un fisiógrafo comercial, además de que la capacidad de procesamiento de señales que posee el sistema de computación es infinitamente mayor.

Otra aplicación contemplada a corto plazo es la evaluación de las curvas de respuesta en frecuencia de los auxiliares auditivos y la adquisición y análisis de potenciales evocados auditivos.

En este momento se utiliza este sistema para la adquisición y el procesamiento de señales impedancimétricas del corazón. En este proyecto se desea correlacionar las características de esta señal con el gasto cardiaco, especialmente durante el ejercicio. Además, el sistema de adquisición se utiliza dentro de un sistema de análisis computarizado del electroencefalograma. Este sistema lleva a cabo la adquisición de 12 canales de EEG simultáneamente, para que posteriormente se lleve a cabo el análisis en frecuencia, el análisis estadístico de la ocurrencia de estos eventos y finalmente el mapeo de esta información para presentar al médico una imagen que indica dónde se generan los eventos de interés (por ejemplo, ondas alfa, beta, etc.), hacia adonde se propagan y qué características adicionales presenta. Esta información se utilizará para facilitar el diagnóstico clínico y para la evaluación de fparmacos que tienen acción sobre el sistema nervioso (Fig.5).

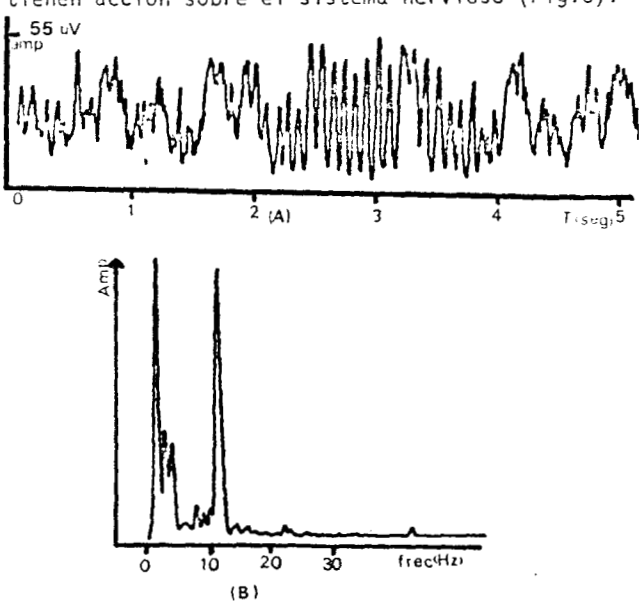


Figura 5. Ejemplos de aplicaciones:
A: Señal electroencefalográfica original. Registro de 5 segundos.

B: Función de densidad espectral calculada a partir de la señal en (A). Se observa que la actividad alfa de la señal produce un pico en 11 Hz.

CONCLUSIONES

La unidad de adquisición de señales anteriormente descrita está diseñada para interactuar con un sistema completo de programación para el análisis de señales biomédicas. Ambos sistemas ofrecen una herramienta de bajo costo y de gran poder computacional con el que hasta ahora no era posible contar. El procesamiento digital de señales biomédicas y particularmente este sistema ofrece la capacidad de presentar nueva información sobre señales analógicas que podrá resultar en nuevas aportaciones en el campo de la ingeniería biomédica.

REFERENCIAS

1. Usi,S, Itzak Amridror "Digital Low-Pass Differentiation for Biological Signal Processing". IEEE Transactions on Biomedical Engineering Vol. 29 No.19. Oct. 1982.

2. IBM Corp. IBM PC Technical Reference Manual 1983

3. Corona Data Systems Corona PC Technical Manual J1,1983.

4. INTEL Corp. IAPx86/88 System Manual 1982

5. Analog Devices Data Acquisition Handbook 1982.

6. Bradley, D.J. Assembly Language Programming for the IBM Personal Computer Prentice-Hall, 1984.

7. Scanlon, L.J. IBM PC Assembly Language: A Guide for Programmers R.J.Brady Co. 1983.

APENDICE B

LISTADO COMPLETO DEL PROGRAMA

```
PROGRAM ProcesamientoDeSenales;

TYPE
    complejo=array[1..2]of real;
    datos=array[1..512]of complejo;
    resultado=array[1..1024]of complejo;
    magni=array[1..1024]of real;

VAR
    da,db,dc,dd:datos;
    ra:resultado;
    d,e,f,g,h,s:char;
    delta:real;
    numarr,i,j,lc:integer;
    rb:magni;

        {Los procedimientos siguientes (Magnitud, FFT  y Normaliza) son
        comunes  a  algunas  de  las  opciones  contenidas  en  el  programa
        principal.  Aparecen corridos hacia la derecha para marcar la
        diferencia}

        PROCEDURE magnitud(VAR a:datos;VAR av:magni;n:integer);

          {Procedimiento magnitud
              entrada: Variable a, tipo datos (arreglo de 512 complejos)
              salida:  Variable av, tipo magni (arreglo de 512 reales)
              n:       Numero de datos}

          BEGIN
             FOR i:=1 to n do
                 av[i]:=SQRT(sqr(a[i,1])+sqr(a[i,2]));
          END;

        PROCEDURE FFT(VAR x,a:datos;m,isign:integer);

          {Procedimiento FFT Transformada rapida de Fourier
              entrada:   Variable x, tipo datos (arreglo de 512 complejos)
              salida:    Variable a, tipo datos (idem)
              m:         Potencia de dos correspondiente al numero de datos
              isign:     Determina si es FFT directa (isign=1)
                         o inversa (isign=-1)}
```

```
VAR
   u,w,t,tneg,ga:complejo;
   n,le,le1,l,ip,j,i:integer;

PROCEDURE intercambio(VAR a:datos;n:integer);

     {Pertenece al procedimiento FFT.  Realiza el intercambio de
     los datos de entrada, para el algoritmo de decimacion en
     tiempo, segun el metodo de inversion de bits

     Entrada:  Variable a, tipo datos
     Salida:   La misma variable, con los datos ordenados
     n:        numero de datos}

    VAR
      nv2,nm1,i,j,k,h:integer;
      t:complejo;

    BEGIN
      nv2:=n div 2;
      nm1:=n-1;
      j:=1;
      FOR i:=1 to nm1 do
         BEGIN
            IF i<j THEN
            FOR h:=1 to 2 do
               BEGIN
                  t[h]:=a[j,h];
                  a[j,h]:=a[i,h];
                  a[i,h]:=t[h];
               END;
            k:=nv2;
            WHILE k<j do
               BEGIN
                 j:=j-k;
                 k:=k div 2;
               END;
            j:=j+k;

         END;       {FOR i}

    END;            {Procedure intercambio}
```

```
FUNCTION pot(v,g:integer):integer;

    {Hace g elevado a la v}

    BEGIN
       IF v>=1 THEN pot:=g*pot(v-1,g)
       ELSE
          pot:=1;

    END;

PROCEDURE multiplicacionmatvect(VAR a:complejo;VAR b:datos;
                                VAR c:complejo;VAR i:integer);

    {Realiza la multiplicacion de dos complejos
    Entrada: El elemento b[i], del arreglo b de datos
             y el numero c
    Salida:  Variable a, tipo complejo}

    BEGIN
       a[1]:=b[i,1]*c[1]-b[i,2]*c[2];
       a[2]:=b[i,1]*c[2]+b[i,2]*c[1];
    END;

PROCEDURE suma(VAR d,e:datos;f:complejo;i,j:integer);

    {Suma dos elementos complejos
    Entrada: El elemento e[j], del arreglo e de datos
             y el numero f
    Salida:  El elemento d[i], del arreglo d de datos}


    BEGIN
       d[i,1]:=e[j,1]+f[1];
       d[i,2]:=e[j,2]+f[2];
    END;

PROCEDURE multiplicacionvect(VAR g,l,m:complejo);

    {Reliza la multiplicacion de dos numeros complejos
    Entrada: Variables l y m, tipo complejo
    Salida:  Variable g, tipo complejo}
```

```
    BEGIN
       g[1]:=(l[1]*m[1])-(l[2]*m[2]);
       g[2]:=(l[1]*m[2])-(-l[2]*m[1]);
    END;


BEGIN

  FOR i:=1 to 512 do
     FOR j:=1 to 2 do
        a[i,j]:=x[i,j];                    {transfiere datos de entrada
                                            x a la variable a}

  n:=pot(m,2);                             {calcula numero de datos}

  intercambio(a,n);                        {ordena datos de entrada}

  FOR l:=1 to m do
    BEGIN
      le:=pot(l,2);
      le1:=le div 2;
      u[1]:=1.0;
      u[2]:=0.0;
      w[1]:=cos(pi/le1);
      w[2]:=sin(pi*isign/le1);
      FOR j:=1 to le1 do
        BEGIN
           i:=j;
           WHILE i<=n do
             BEGIN
                ip:=i+le1;
                multiplicacionmatvect(t,a,u,ip);
                tneg[1]:=-t[1];
                tneg[2]:=-t[2];
                suma(a,a,tneg,ip,i);
                suma(a,a,t,i,i);
                i:=i+le;
             END;          {while}
           ga[1]:=u[1];
           ga[2]:=u[2];
           multiplicacionvect(ga,u,w);
           u[1]:=ga[1];
           u[2]:=ga[2];
        END;               {FOR j}
    END;                   {FOR i}
```

```
        IF isign>0 then                    {Si es transformada inversa
                                           divide entre el numero de datos}
          BEGIN
            FOR i:=1 to n do
                FOR j:=1 to 2 do
                    a[i,j]:=a[i,j]/n;
          END;

      END;      {Procedure FFT}

    PROCEDURE Normaliza(VAR x:datos);

       {Calcula el promedio de un arreglo de datos y lo resta a cada
       elemento; es decir, elimina el valor de D.C.}

       VAR
          media:real;

       BEGIN
          media:=0.0;
          FOR i:=1 to 512 do
              media:=media+x[i,1];

          media:=media/512;

          FOR i:=1 to 512 do
              x[i,1]:=x[i,1]-media;

       END;      {Procedure Normaliza}

{Los procedimientos siguientes realizan las diferentes opciones que contiene
el programa principal}

PROCEDURE Adquisicion(VAR da,db,dc,dd:datos);

     {Adquiere 512 datos a traves de la tarjeta de C A/D
     Entrada:  Ninguna
     Salida:   Arreglos da,db,dc,dd, tipo datos}


     TYPE
        arreglo_largo=array[1..2048] of integer;
```

```
VAR
   numcan:integer;
   datos_conv:arreglo_largo;

PROCEDURE Adquiere(VAR datos_conv:arreglo_largo;numcan:integer);
          external 'ADQUI.COM';

PROCEDURE Separa(datos_conv:arreglo_largo;VAR da,db,dc,dd:datos);

   BEGIN
      FOR i:=1 to 512 do
         BEGIN
            da[i,1]:=datos_conv[i];
            db[i,1]:=datos_conv[i+512];
            dc[i,1]:=datos_conv[i+1024];
            dd[i,1]:=datos_conv[i+1536];
         END;
   END;

PROCEDURE lee_parametros_programa_conv;

   {Lee parametros de frecuencia de muestreo y numero de canales.
   Calcula el dato correspondiente a la frec de muestreo que debe
   mandarse al reloj programable}

   VAR
      frecmue:real;
      numtimer:integer;

   PROCEDURE Programa_convertidor(numtimer:integer);
             external 'CONVERTI.COM';

   BEGIN
      writeln;
      write('(Numero de canales (1-4)? ');
      readln(numcan);
      writeln;
      write('(Frecuencia de muestreo (30 - 10000 Hz)? ');
      readln(frecmue);
      numtimer:=round(1.195E6/(frecmue*numcan));

      Programa_convertidor(numtimer);

   END;
```

```
BEGIN
   clrscr;
   writeln('          ESTA RUTINA ADQUIERE HASTA 4 CANALES ANALOGICOS');
   writeln('          (512 datos c/u) Y LOS DEPOSITA EN LOS ARREGLOS');
   writeln('                   DE DATOS CORRESPONDIENTES:');
   writeln('                    CANAL 1 - Arreglo A ');
   writeln('                    CANAL 2 - Arreglo B ');
   writeln('                    CANAL 3 - Arreglo C ');
   writeln('                    CANAL 4 - Arreglo D ');

   REPEAT
      writeln;
      write('(Desea continuar (S/N)?  ');
      read(kbd,e);
      writeln(e);
   UNTIL Upcase(e) in ['S','N'];

   CASE Upcase(e) of

      'S': BEGIN

              Lee_parametros_programa_conv;

              Adquiere(datos_conv);
              Separa(datos_conv,da,db,dc,dd);

              write('LISTO. Sus datos se encuentran en los ');
              writeln('arreglos correspondientes');

              END;  {Case f}
           END;     {Begin}

   END;             {Case e}
END;                {Begin}


PROCEDURE Convolucion(VAR da,db:datos;VAR ra:resultado);

   {Calcula la funcion de convolucion entre los arreglos de datos da y db
   Entrada: Arreglos da y db, tipo datos
   Salida:  Arreglo ra, tipo resultados}

   VAR
      na,nb:integer;
```

```
PROCEDURE Convol(VAR a,b:datos;VAR c:resultado;la,lb:integer;
                 delta:real);

  {Evalua los puntos de la convolucion
  Entrada: Arreglos a, b, tipo datos;
           variables la,lb, longitud del arreglo a y b respectivamente
           Variable delta, intervalo de muestreo, tipo real
  Salida:  Arreglo r, tipo resultado}


  VAR
     i,j,k:integer;

  BEGIN
     lc:=la+lb-1;
     FOR i:=1 to 512 do                  {Borra el arreglo donde depositara
                                          el resultado}
        c[i,1]:=0.0;

     FOR i:=1 to la do                   {Evalua la convolucion}

        FOR j:=1 to lb do
           BEGIN
              k:=i+j-1;
              c[k,1]:=c[k,1]+a[i,1]*b[j,1];
           END;

      FOR i:=1 to lc do                  {multiplica resultado por el
                                          intervalo de muestreo}
         c[i,1]:=c[i,1]*delta;

   END;    {Procedure convol}


BEGIN

   clrscr;
   writeln;
   writeln('         ESTA RUTINA REALIZA LA CONVOLUCION DE LOS DATOS');
   writeln('              CONTENIDOS EN LOS ARREGLOS A Y B');
   writeln;
```

```
        REPEAT
            write('Están listos sus datos? (S/N)  ');
            read(kbd,e);
            writeln(e);

        UNTIL e in ['S','s','N','n'];

        CASE e OF

            'S','s':BEGIN
                        writeln;
                        write('Intervalo de muestreo (seg)?  ');
                        readln(delta);
                        writeln;
                        write('Número de puntos del arreglo A?   ');
                        readln(na);
                        writeln;
                        write('Número de puntos del arreglo B?   ');
                        readln(nb);
                        Convol(da,db,ra,na,nb,delta);
                        writeln;
                        writeln('LISTO.  Su resultado se encuentra en el');
                        writeln('        arreglo R');

                    END;

             END;    {Case}

    END;            {Procedure  Convolucion}

PROCEDURE Despliegue(VAR da,db,dc,dd:datos;VAR ra:resultado);

    {Despliega cualquiera de los arreglos de datos o resultados
    Entrada: Arreglos da,db,dc,dd,tipo datos
             Arreglo ra, tipo resultado}

    VAR
       ca:datos;
       miny,maxy:real;
```

```
PROCEDURE despl(VAR d:datos);

  {Realiza el despliegue del arreglo d
  Entrada: Arreglo d, tipo datos}

   VAR
     escver,numhor,eschor,rango,escala,inter:real;
     numpun:integer;
     tipo:char;
     unidades:string[3];

   PROCEDURE minmax(d:datos;VAR miny,maxy:real);

      {Calcula el maximo y el minimo del arreglo d, para fijar los
      factores de expansion si se desea escalamiento automatico
      Entrada: Arreglo d, tipo datos que se desea desplegar
      Salidas: Variables miny,maxy minimo y maximo respectivamente del
               arreglo d}


      BEGIN
         miny:=1e10+36;
         maxy:=-miny;
         FOR i:=1 to numpun do
             BEGIN
               IF d[i,1]<miny THEN miny:=d[i,1];
               IF d[i,1]>maxy THEN maxy:=d[i,1];
             END;
      END;        {Procedure minmax}


   PROCEDURE parametros;

      {Lee parametros: tipo de arreglo (Tiempo o frecuencia) y frecuencia
      de muestreo y fija escala horizontal y unidades}

      BEGIN
          writeln;
          write('Archivo de tiempo (T) o frecuencia (F)  ');
          read(kbd,tipo);
          writeln(tipo);
          write('Intervalo de muestreo? ');
          readln(inter);
```

```
        CASE tipo OF
           't','T': BEGIN numhor:=512*inter/10; unidades:='seg';  END;
           'f','F': BEGIN numhor:=1/(inter*10); unidades:='Hz';   END;
        END;    {case}

    END;     {Procedure parametros}


PROCEDURE ejes;

VAR
    rayita:integer;

    BEGIN

      {dibuja las rayitas de los ejes}

      FOR i:=1 to 20 do
         BEGIN
            IF i<15 THEN
            draw(52,round(174-i*12.5),56,round(174-i*12.5),1);
            draw(54+25*i,172,54+25*i,176,1);
         END;

      FOR i:=1 to 9 do
          draw(48,round(174-2*i*12.5),52,round(174-2*i*12.5),1);

      FOR i:=1 to 10 do
          draw(54+25*2*i,176,54+25*2*i,180,1);

      {dibuja el marco}

      draw(54,0,54,174,1);
      draw(54,174,566,174,1);
      draw(566,0,566,174,1);
      draw(54,0,566,0,1);

      {dibuja escala horizontal}

       FOR i:=1 to 10 do
          BEGIN
             gotoxy(round(i*6.3+6),24);
             write(numhor*i/eschor:2:1);
          END;
```

```
    gotoxy(76,24);
    write(unidades);

    {dibuja escala vertical}

    FOR i:=0 to 7 do
        BEGIN
            gotoxy(2,(22-i*3));
            write(miny+escver*i:2:2);
        END;

END;    {Procedure ejes}


BEGIN
   writeln;

   REPEAT
      writeln;
      write('Desea graficar todo el arreglo (S/N)?  ');
      read(kbd,h);
      writeln(h);
   UNTIL h in ['S','s','N','n'];

   CASE h of

   'S','s': BEGIN
               numpun:=511;          {numero de puntos a graficar}
               eschor:=1.0;          {con escala horizontal unitaria}
            END;

   'N','n': BEGIN
               writeln;
               write('Cuántos puntos desea graficar (1-511)?   ');
               readln(numpun);
               eschor:=512/numpun; {calcula esc hor correspondiente}
            END;
   END;    {Case h}

   REPEAT
     write('Desea escalamiento vertical automático (S/N)?  ');
     read(kbd,g);
     writeln(g);
   UNTIL g in ['S','s','N','n'];
```

```
        parametros;
        clrscr;
        hires;
        hirescolor(13);

        CASE g of

                {dibuja arreglo con escalamiento automatico}

        'S','s': BEGIN
                    minmax(d,miny,maxy);
                    rango:=maxy-miny;

                    IF rango<>0 THEN escala:=174/rango
                    ELSE escala:=1;

                    {escala es el factor para el despliegue}
                    {escver es el paso entre unidades verticales}

                    escver:=rango/7;
                    ejes;

                    FOR i:=1 to numpun do
                             draw(round((i*eschor)+53),
                                  round(174-(d[i,1]-miny)*escala),
                                  round(((i+1)*eschor)+53),
                                  round(174-(d[i+1,1]-miny)*escala),1);
                 END;

                {dibuja arreglo sin escalamiento automatico}

        'N','n': BEGIN
                    escver:=4350/7;
                    miny:=0;
                    ejes;

                    FOR i:=1 to numpun do
                             draw(round((i*eschor)+53),
                                  round(174-d[i,1]*0.04),
                                  round(((i+1)*eschor)+53),
                                  round(174-d[i+1,1]*0.04),1);
                 END;

        END;    {Case g}

        REPEAT
        UNTIL keypressed;
        textmode;

END;        {Procedure despl}
```

```
BEGIN
   clrscr;
   writeln;
   writeln('          ESTA RUTINA PERMITE DESPLEGAR EN PANTALLA');
   writeln('      CUALQUIERA DE LOS ARREGLOS DE DATOS O RESULTADO');
   writeln;
   REPEAT
       write('Desea continuar (S/N)?   ');
       read(kbd,e);
       writeln(e);
   UNTIL e in ['S','s','N','n'];

   CASE e of

       'S','s': BEGIN
                  REPEAT
                     writeln;
                     write('(Qué arreglo desea desplegar');
                     writeln(' (A,B,C,D,R)? ');
                     read(kbd,f);
                     writeln(f);
                  UNTIL f in ['A','a','B','b','C','c','D','d','R','r'];

                  CASE f of
                      'A','a': despl(da);
                      'B','b': despl(db);
                      'C','c': despl(dc);
                      'D','d': despl(dd);
                      'R','r': BEGIN
                                  FOR i:=1 to 512 do
                                      FOR j:=1 to 2 do
                                           ca[i,j]:=ra[i,j];

                                  despl(ca);
                                  write('Presione una tecla para des');
                                  write('plegar la segunda parte del');
                                  writeln(' arreglo');

                                  REPEAT
                                  UNTIL keypressed;
                                  read(kbd,g);
```

```
                                    FOR i:=1 to 512 do
                                        FOR j:=1 to 2 do
                                            ca[i,j]:=ra[i+512,j];

                                    despl(ca);
                                END;

                        END;    {case f}

                    END;        {begin}

        END;    {Case e}

    END;        {Procedure despliegue}

PROCEDURE Filtrado(VAR da,dc:datos);

    {Calcula respuesta a impulso y funcion de transferencia de un filtro
    Salidas: Arreglo da, tipo datos, contiene la respuesta a impulso
             Arreglo dc, tipo datos, contiene la funcion de transferencia}

    VAR
        atencos,inte:real;
        frecor:array[1..20]of real;
        nupunt,ventana,bandas:integer;
        vent:magni;


    PROCEDURE filt(VAR da,dc:datos);

    {Calcula respuesta a impulso y funcion de transferencia de un filtro
    Salidas: Arreglo da, tipo datos, contiene la respuesta a impulso
             Arreglo dc, tipo datos, contiene la funcion de transferencia}


    VAR
        com,nupunt2:integer;

        PROCEDURE parametros;

            {Lee parametros del usuario: numero de bandas, frecuencias de
            corte de cada banda, porcentaje de atenuacion cosenoidal, numero
            de puntos del filtro, intervalo de muestreo, tipo de ventana
            (rectangular o Hamming)}
```

```
BEGIN
  writeln;
  write('(Cuántas bandas de paso desea?  ');
  readln(bandas);

  FOR i:=1 to bandas do
       BEGIN
          writeln;
          write('(En qué frecuencia comienza la banda  ',i);
          write(' (Hz)  ');
          readln(frecor[2*i-1]);
          writeln;
          write('(En que frecuencia termina la banda  ',i);
          write(' (Hz)  ');
          readln(frecor[2*i]);
       END;

  writeln;
  write('(Qué porcentaje de atenuación cosenoidal desea ');
  write('(0.0-1.0)?  ');
  readln(atencos);

  writeln;
  write('(Intervalo de muestreo (seg)?  ');
  readln(inte);

  writeln;
  write('(Número de puntos del filtro (impar)?  ');
  readln(nupunt);

  REPEAT
    writeln;
    write('(Desea aplicar una ventana de Hamming a sus datos ');
    write('(S/N)?  ');
    read(kbd,f);
    writeln(f);
  UNTIL f in ['S','s','N','n'];

  CASE f of
    'S','s':  ventana:=1;
    'N','n':  ventana:=0;
  END;

END;      {Procedure parametros}
```

```
PROCEDURE correspondiente(VAR dc:datos);

{Calcula el numero de puntos correspondiente a la frecuencia de
corte de cada banda y va definiendo la funcion de transferencia

Salida: Arreglo dc, tipo datos, que contiene la funcion de
        transferencia definida}


TYPE
   arreglore=array[1..20]of real;
   arregloen=array[1..20]of integer;

VAR
   atenuacion,puntosatenuados,puntosfrecor:arreglore;
   indinferior,indsuperior:arregloen;

   PROCEDURE funtran(VAR d:datos;indinferior,indsuperior:arregloen;
                     puntosatenuados:arreglore);

      BEGIN
         FOR i:=1 to bandas do

           BEGIN

             FOR j:=indsuperior[2*i-1] to indinferior[2*i] do
               d[j,1]:=1.0;

             IF puntosatenuados[2*i-1]<>0 then

             FOR j:=indinferior[2*i-1]+1 to indsuperior[2*i-1]-1 do
                d[j,1]:=0.5*(1-cos((j-indinferior[2*i-1])*
                             3.14159/puntosatenuados[2*i-1]));

             IF puntosatenuados[2*i]<>0 then

             FOR j:=indinferior[2*i]+1 to indsuperior[2*i]-1 do
                d[j,1]:=0.5*(1+cos((j-indinferior[2*i])*
                             3.14159/puntosatenuados[2*i]));

           END;
         FOR i:=2 to 256 do
              d[514-i,1]:=d[i,1];

      END;  (Procedure funtran)
```

```
BEGIN    {Procedure correspondiente}

  FOR i:=1 to bandas do

    BEGIN

      {porcentaje de atenuacion en cada banda}

      atenuacion[2*i-1]:=atencos*frecor[2*i-1];
      atenuacion[2*i]:=atencos*frecor[2*i];

      {numero de puntos atenuados en c/banda}

      puntosatenuados[2*i-1]:=512*atenuacion[2*i-1]*inte;
      puntosatenuados[2*i]:=512*atenuacion[2*i]*inte;

      {numero de puntos de cada frecuencia de corte}

      puntosfrecor[2*i-1]:=512*frecor[2*i-1]*inte+1;
      puntosfrecor[2*i]:=512*frecor[2*i]*inte+1;

      {indice inferior de cada banda}

      indinferior[2*i-1]:=round(puntosfrecor[2*i-1]-
                                puntosatenuados[2*i-1]/2);
      indinferior[2*i]:=round(puntosfrecor[2*i]-
                              puntosatenuados[2*i]/2);

      {indice superior de cada banda}

      indsuperior[2*i-1]:=round(puntosfrecor[2*i-1]+
                                puntosatenuados[2*i-1]/2);
      indsuperior[2*i]:=round(puntosfrecor[2*i]+
                              puntosatenuados[2*i]/2);


    END;

    funtran(dc,indinferior,indsuperior,puntosatenuados);

END;    {Procedure correspondiente}
```

```
PROCEDURE hamming(VAR vent:magni;nupunt:integer);

    {Calcula la ventana de Hamming
    Entrada: Variable nupunt, numero de puntos de la ventana
    Salida:  Arreglo vent, tipo magni}

    VAR
        npts2:integer;

    BEGIN
        npts2:=(nupunt div 2)+1;
        j:=1;
        FOR i:=npts2 to nupunt do
           BEGIN
              vent[i]:=0.54+0.46*cos((2*pi*j)/nupunt);
              j:=j+1;
           END;

        FOR i:=1 to npts2-1 do
           vent[i]:=vent[nupunt-i+1];

    END;       {Procedure Hamming}


PROCEDURE prepara(VAR a:datos;la:integer;VAR b:magni);

    VAR
       k,kf,kff:integer;

    BEGIN
       k:=(la-1)div 2;
       j:=k+1;
       FOR i:=1 to k+1 do
          BEGIN
             b[i]:=a[j,1];
             j:=j+1;
          END;

       kf:=512-la;
       kff:=k+2+kf-1;
       FOR i:=k+2 to kff do
          b[i]:=0.0;

       j:=1;
```

```
        FOR i:=kff+1 to 512 do
           BEGIN
              b[i]:=a[j,1];
              j:=j+1;
           END;

     END;


BEGIN       {Procedure filt}

     FOR i:=1 to 1024 do rb[i]:=0.0;

     parametros;                    {pregunta parametros del usuario}
     correspondiente(dc);           {calcula la aten. correspondiente
                                     para cada frecuencia y la funcion
                                     de transferencia del filtro}
     FOR i:=1 to 512 do
        dc[i,1]:=dc[i,1]/inte;

     FFT(dc,dc,9,1);                {Calcula resp a impulso}
     nupunt2:=(nupunt-1) div 2;
     com:=512-nupunt2;
     FOR i:=1 to nupunt2 do
        BEGIN
           rb[i]:=dc[com+i,1];
           rb[nupunt2+i]:=dc[i,1];
        END;
     rb[nupunt]:=dc[nupunt2+1,1];
     IF ventana=1 THEN              {Si debe multiplicar por ventana
                                     de Hamming, la calcula y
                                     multiplica la resp a impulso}
        BEGIN
           hamming(vent,nupunt);
           FOR i:=1 to nupunt do
              rb[i]:=rb[i]*vent[i];
        END;

     FOR i:=1 to 512 do             {Deposita la resp a impulso
                                     corregida en el arreglo da}
       BEGIN
         da[i,1]:=rb[i];
         da[i,2]:=0.0;
       END;
```

```
        prepara(da,nupunt,rb);          {Prepara el arreglo para transf}

        FOR i:=1 to 512 do
            BEGIN
               dc[i,1]:=rb[i]*inte;
               dc[i,2]:=0.0;
            END;

        FFT(dc,dc,9,-1);                {Calcula func de transf final}

        FOR i:=257 to 512 do dc[i,1]:=0.0;

    END;        {Procedure filt}


BEGIN           {Procedure filtrado}

    clrscr;
    writeln('         ESTA RUTINA CALCULA LA RESPUESTA A IMPULSO Y LA');
    writeln('              FUNCION DE TRANSFERENCIA DE UN FILTRO ');
    writeln('      Y LAS DEPOSITA EN LOS ARREGLOS A Y C RESPECTIVAMENTE');
    writeln;

    REPEAT
        write('Desea continuar (S/N)? ');
        read(kbd,e);
        writeln(e);
    UNTIL e in ['S','s','N','n'];

    FOR i:=1 to 512 do
      FOR j:=1 to 2 do
       BEGIN
         da[i,j]:=0.0;                  {Borra arreglos da y dc}
         dc[i,j]:=0.0;
       END;

    CASE e OF
       'S','s': BEGIN

                  filt(da,dc);

                  writeln;
                  write('LISTO. La respuesta a impulso del filtro se');
                  writeln(' encuentra en el arreglo A');
                  writeln;
```

```
                    write('       La función de transferencia se');
                    writeln(' encuentra en el arreglo C');

                  END;

        END;      {Case e}

    END;          {Procedure filtrado}

PROCEDURE Graficacion(VAR da,db,dc,dd:datos);

    {Manda a graficar en papel cualquiera de los arreglos de datos El
    graficador que maneja es el Sweet-P}

     PROCEDURE grafica(VAR d:datos);

          {Pregunta factores de escalamiento horizontal y vertical y calcula
          las escalas correspondientes}

          VAR
             comp,eschor,miny,maxy,rango,escala:real;

          PROCEDURE minmax(d:datos;VAR miny,maxy:real);

          BEGIN
             miny:=1e10+36;
             maxy:=-miny;
             FOR i:=1 to 512 do
                 BEGIN
                   IF d[i,1]<miny THEN miny:=d[i,1];
                   IF d[i,1]>maxy THEN maxy:=d[i,1];
                 END;
          END;     {Procedure minmax}

          BEGIN
            eschor:=1.0;
            minmax(d,miny,maxy);
            rango:=maxy-miny;
            IF rango<>0 THEN escala:=1838/rango
            ELSE escala:=1.0;
            write('(Cuanto desea comprimir la grafica?  ');
            readln(comp);
            escala:=escala/comp;
            writeln('escala ',escala);
            writeln('miny ',miny);
```

```
        readln(escala);
        readln(miny);
        writeln('Que escala horizontal desea?');
        readln(eschor);

        writeln(LST,'AY 1000,100,20;');
        writeln(LST,'MA 0,0;');
        writeln(LST,'AX 1536,153,20;');
        writeln(LST,'MA 0,0;');

        FOR i:=1 to 511 do
            writeln(LST,'DA',round(i*4*eschor),',',
                            round((d[i,1]-miny)*escala),',',
                            round((i+1)*4*eschor),',',
                            round((d[i+1,1]-miny)*escala),';');

      END;    {Procedure grafica}

BEGIN
   clrscr;
   writeln('               ESTA RUTINA PERMITE GRAFICAR UN ARREGLO DE');
   writeln('                  DATOS EN UN GRAFICADOR MARCA SWEET-P');
   REPEAT
      writeln;
      writeln('(Desea continuar (S/N)?  ');
      read(kbd,e);
      writeln(e);
   UNTIL e in ['S','s','N','n'];

   CASE e of
   'S','s': BEGIN
               writeln(lst,'ho;');
               writeln(lst,'pd;');
               write('(Qué arreglo desea graficar?  ');
               read(kbd,f);
               writeln(f);
               CASE f of
               'A','a': grafica(da);
               'B','b': grafica(db);
               'C','c': grafica(dc);
               'D','d': grafica(dd);
               END;  {Case f}
            END;  {Begin}
   END;    {Case e}

END;       {Procedure graficacion}
```

```
PROCEDURE LecturaYAlmacenEnDisco(VAR da,db,dc,dd:datos);

     VAR
        archivo:file of integer;
        nombre:string[20];
        Existe:boolean;

     PROCEDURE LeeArchivo(VAR d:datos);

     VAR
         lectura:integer;

         BEGIN
            FOR i:=1 to 512 do
              BEGIN
                 read(archivo,lectura);
                 d[i,1]:=lectura;
                 d[i,2]:=0.0;
              END;

         END;

     PROCEDURE EscribeArchivo(VAR d:datos);

     VAR
         escritura:integer;

         BEGIN
           FOR i:=1 to 512 do
              BEGIN
              escritura:=round(d[i,1]);
              write(archivo,escritura);
              END;
         END;

     BEGIN
        clrscr;
        writeln('          ESTA RUTINA PERMITE LEER O ALMACENAR UN ARCHIVO');
        writeln('                          DE DATOS EN DISCO');
        writeln;
        REPEAT
           write('Desea continuar (S/N)?    ');
           read(kbd,e);
           writeln(e);
        UNTIL e in ['S','s','N','n'];
```

```
CASE e of
   'S','s': BEGIN
                writeln;
                writeln('(L) Lectura de disco');
                writeln('(A) Almacen en disco');
                writeln;
                REPEAT
                  write('OPCION? ');
                  read(kbd,f);
                  writeln(f);
                UNTIL f in ['L','l','A','a'];
                writeln;
                write('Nombre del archivo?  ');
                readln(nombre);

                CASE f of
                    'L','l': BEGIN
                               REPEAT
                                  Assign(archivo,nombre);
                                  {$I-} Reset(archivo) {$I+};
                                  Existe:=(IOresult=0);
                                  IF not Existe then BEGIN
                                    writeln;
                                    write('EL ARCHIVO ',nombre);
                                    writeln(' NO EXISTE');
                                    writeln;
                                    write('Nombre del archivo?  ');
                                    readln(nombre);
                                    END;
                               UNTIL Existe;
                               writeln;
                               REPEAT
                                write('En qué arreglo desea depo');
                                write('sitar los datos (A,B,C,D)?');
                                read(kbd,g);
                                writeln(g);
                               UNTIL Upcase(g) in ['A','B','C','D'];

                               CASE g of
                                  'A','a': LeeArchivo(da);
                                  'B','b': LeeArchivo(db);
                                  'C','c': LeeArchivo(dc);
                                  'D','d': LeeArchivo(dd);
                               END;
                               Close(archivo);
                            END;
```

```
                        'A','a': BEGIN
                                   Assign(archivo,nombre);
                                   Rewrite(archivo);
                                   writeln;
                                   REPEAT
                                   write('Qué arreglo desea almacenar');
                                   write(' (A,B,C,D)?  ');
                                   read(kbd,g);
                                   writeln(g);
                                   UNTIL Upcase(g) in ['A','B','C','D'];

                                   CASE g of
                                        'A','a': EscribeArchivo(da);
                                        'B','b': EscribeArchivo(db);
                                        'C','c': EscribeArchivo(dc);
                                        'D','d': EscribeArchivo(dd);

                                   END;

                                 Close(archivo);
                                 END;

                      END; {Case f}

              END;

      END; {Case e}

    END; {Procedure Lectura_y_almacen_en_disco}

PROCEDURE IntercambioDeArreglos(VAR da,db,dc,dd:datos;VAR ra:resultado);

  VAR
      tempo1,tempo2:datos;

  PROCEDURE intcambio(VAR inta,intb:datos);

    VAR
       intc:datos;

    BEGIN
        FOR i:=1 to 512 do
          BEGIN
            intc[i,1]:=intb[i,1];
            intb[i,1]:=inta[i,1];
```

```
          inta[i,1]:=intc[i,1];
        END;

  END;  (procedure intcambio)

PROCEDURE intercambioR(VAR inta:datos;VAR ra:resultado);

   VAR
     temp:datos;

   BEGIN

        writeln;
        write('El arreglo de resultados tiene una longitud ');
        writeln('de 1024 datos.');
        REPEAT
          write('(Desea intercambiar los primeros 512  puntos?');
          read(kbd,g);
          writeln(g);
        UNTIL g in ['S','s','N','n'];

        CASE g of
        'S','s': FOR i:=1 to 512 do
                    temp[i,1]:=ra[i,1];

        'N','n': FOR i:=513 to 1024 do
                    temp[i-512,1]:=ra[i,1];

         END;
         intcambio(inta,temp);

   END;  (Procedure intercambio)


BEGIN
   clrscr;
   writeln('      ESTA RUTINA LE PERMITE INTERCAMBIAR CUALQUIER ARREGLO');
   writeln('                  DE DATOS O RESULTADO ');
   writeln;
   REPEAT
     write('Desea continuar (S/N)?  ');
     read(kbd,e);
     writeln(e);
   UNTIL e in ['S','s','N','n'];
```

```
CASE e of
  'S','s': BEGIN
             REPEAT
               writeln;
               write('( Qué arreglos desea intercambiar ?  ');
               REPEAT
                 read(kbd,f);
                 write(f,',');
               UNTIL f in ['A','a','B','b','C','c','D','d','R','r'];
               read(kbd,s);
               writeln(s);
             UNTIL s in ['A','a','B','b','C','c','D','d','R','r'];

             CASE f of
               'A','a': CASE Upcase(s) of
                          'A': write('No tiene caso intercambiar');
                               writeln(' el mismo arreglo');
                          'B': intcambio(da,db);
                          'C': intcambio(da,dc);
                          'D': intcambio(da,dd);
                          'R': intercambioR(da,ra);
                        END;

               'B','b': CASE Upcase(s) of
                          'A': intcambio(db,da);
                          'B': write('No tiene caso intercambiar');
                               writeln(' el mismo arreglo');
                          'C': intcambio(db,dc);
                          'D': intcambio(db,dd);
                          'R': intercambioR(db,ra);
                        END;

               'C','c': CASE Upcase(s) of
                          'A': intcambio(dc,da);
                          'B': intcambio(dc,db);
                          'C': write('No tiene caso intercambiar');
                               writeln(' el mismo arreglo');
                          'D': intcambio(dc,dd);
                          'R': intercambioR(dc,ra);
                        END;

               'D','d': CASE Upcase(s) of
                          'A': intcambio(dd,da);
                          'B': intcambio(dd,db);
                          'C': intcambio(dd,dc);
```

```
                              'D': write('No tiene caso intercambiar');
                                   writeln(' el mismo arreglo');
                              'R': intercambioR(dd,ra);
                            END;

                    'R','r': CASE s of
                              'A','a': intercambioR(da,ra);
                              'B','b': intercambioR(db,ra);
                              'C','c': intercambioR(dc,ra);
                              'D','d': intercambioR(dd,ra);
                              'R','r': BEGIN
                                       write('Esta opción intercambia');
                                       write(' los primeros 512 puntos ');
                                       write('del arreglo de resultados');
                                       write(' con los 512 últimos');
                                       writeln(' puntos');
                                       FOR i:=1 to 512 do
                                         BEGIN
                                           tempo1[i,1]:=ra[i,1];
                                           tempo2[i,1]:=ra[i+512,1];
                                         END;
                                         intcambio(tempo1,tempo2);
                                       END;

                            END;     {Case s}

                  END;    {Case f}

                END;

      END;            {Case e}

   END;               {Procedure IntercambioDeArreglos}

PROCEDURE Normalizacion(VAR da,db,dc,dd:datos);

     {Normaliza un arreglo de datos
     Llama a la rutina Normaliza, definida al inicio del programa
     Entrada: Arreglos da,db,dc,dd, tipo datos
     Salida : Los mismos arreglos, el correspondiente normalizado}
```

```
BEGIN
   clrscr;
   writeln('          ESTA RUTINA NORMALIZA UN ARREGLO DE DATOS,');
   writeln('       ES DECIR, ELIMINA EL VALOR DE DC DE CADA ELEMENTO');
   writeln;

   REPEAT
     write('Desea continuar (S/N)?  ');
     read(kbd,e);
     writeln(e);
   UNTIL e in ['S','s','N','n'];

   CASE e of
     'S','s': BEGIN
                writeln;
                REPEAT
                    write('Qué arreglo desea normalizar (A,B,C,D)?  ');
                    read(kbd,f);
                    writeln(f);
                UNTIL f in ['A','a','B','b','C','c','D','d'];

                CASE f of
                    'A','a': Normaliza(da);
                    'B','b': Normaliza(db);
                    'C','c': Normaliza(dc);
                    'D','d': Normaliza(dd);

                 END; {Case f}

              END;

     END;   {case e}

END;   {Procedure Normalizacion}


PROCEDURE Calculamagnitud(VAR da,db,dc,dd:datos;VAR ra:resultado);

   {Llama a la rutina Magnitud, definida al inicio del programa
   Entrada: Arreglos da,db,dc,dd, tipo datos
   Salida : Arreglo ra, contiene la magnitud del arreglo pedido}
```

```
BEGIN
   clrscr;
   writeln('            ESTA RUTINA CALCULA LA MAGNITUD DE ');
   writeln('            CUALQUIERA DE LOS ARREGLOS DE DATOS');
   writeln;
   REPEAT
       write('Desea continuar (S/N)?  ');
       read(kbd,e);
       writeln(e);
   UNTIL e in ['S','s','N','n'];

   CASE e of
       'S','s': BEGIN
                  writeln;
                  REPEAT
                     write((De que arreglo desea calcular la magnitud');
                     write(' (A,B,C,D)? ');
                     read(kbd,f);
                     writeln(f);
                  UNTIL f in ['A','a','B','b','C','c','D','d'];

                  CASE f of
                     'A','a': magnitud(da,rb,512);
                     'B','b': magnitud(db,rb,512);
                     'C','c': magnitud(dc,rb,512);
                     'D','d': magnitud(dd,rb,512);

                  END;  {case f}

                  FOR i:=1 to 512 do
                     BEGIN
                        ra[i,1]:=rb[i];
                        ra[i+512,1]:=0.0;
                     END;

                  writeln;
                  write('LISTO. Su resultado se encuentra en el ');
                  writeln(' arreglo R');


                END;

   END;              {case e}

END;                 {Procedimiento CalculaMagnitud}
```

```
PROCEDURE DensidadEspectral(VAR da,db,dc,dd:datos;VAR ra:resultado);

    PROCEDURE Densidad(VAR d:datos;VAR ra:resultado);

      {Calcula la densidad espectral de un arreglo de datos. Llama a
      rutinas FFT, Magnitud y Normaliza, definidas al inicio del programa.
      Entrada: Arreglo d, tipo datos, que se quiere procesar
      Salida:  Arreglo ra, tipo resultados, contiene la densidad
               espectral del arreglo d}

      VAR
         media,pi,retraso,inter:real;
         ihamm:integer;
         sp,wt:magni;
         r:datos;

      PROCEDURE hammre(VAR w:magni;ihamm:integer);

          {Calcula la ventana de Hamming con la que se corregiran los datos
          del arreglo d}

          BEGIN
             FOR i:=1 to 512 do
                w[i]:=0.0;

             pi:=3.1416;
             w[1]:=1.0;
             FOR i:=2 to ihamm do
                BEGIN
                  w[i]:=0.54+(0.46*cos(pi*(i-1)/(ihamm-1)));
                  j:=512+2-i;
                  w[j]:=w[i];
                END;

           END;      {Procedure Hammre}

      BEGIN

         writeln;
         write('Intervalo de muestreo (seg)?  ');
         readln(inter);

         Normaliza(d);                    {Elimina DC del arreglo para obtener
                                           la funcion de covarianza}
```

```
      FFT(d,r,9,-1);                    {Calcula FFT del arreglo y su
                                         magnitud}

      Magnitud(r,sp,512);

      FOR i:=1 to 512 do
          BEGIN                         {Calcula magnitud cuadratica}
             sp[i]:=SQR(sp[i]);
             r[i,1]:=sp[i];
          END;

      FFT(r,r,9,1);                     {Obtiene la autocovarianza}

      FOR i:=1 to 512 do
          r[i,1]:=r[i,1]/512;

      retraso:=0.3;

      ihamm:=round(512*retraso);
      hammre(wt,ihamm);                 {Calcula ventana y multiplica la
                                         autocovarianza}

      FOR i:=1 to 512 do
          r[i,1]:=r[i,1]*wt[i];

      FFT(r,r,9,-1);                    {Regresa a frecuencia y calcula
                                         densidad espectral final}

      Magnitud(r,sp,512);

      FOR i:=1 to 512 do
         ra[i,1]:=sp[i]*inter;

   END;          {Procedure densidad}


BEGIN
   clrscr;
   writeln('          ESTA RUTINA CALCULA LA DENSIDAD ESPECTRAL');
   writeln('         DE UN ARREGLO DE DATOS, MEDIANTE EL CALCULO');
   writeln('         DE LA TRANSFORMADA DE FOURIER DE LA FUNCION');
   writeln('            DE AUTOCOVARIANZA, Y LA DEPOSITA EN EL ');
   writeln('                              ARREGLO R');
   writeln;
```

```
      REPEAT
         write('Desea continuar (S/N)?   ');
         read(kbd,e);
         writeln(e);
      UNTIL e in ['S','s','N','n'];
      writeln;
      CASE e of
         'S','s': BEGIN
                     REPEAT
                        write('De qué arreglo desea obtener la densidad');
                        write(' espectral (A,B,C,D)?  ');
                        read(kbd,f);
                        writeln(f);
                     UNTIL f in ['A','a','B','b','C','c','D','d'];

                     CASE f of
                        'A','a': Densidad(da,ra);
                        'B','b': Densidad(db,ra);
                        'C','c': Densidad(dc,ra);
                        'D','d': Densidad(dd,ra);
                     END;

                     writeln;
                     write('LISTO. Su resultado se encuentra en el ');
                     writeln(' arreglo R');

                  END;

        END;  {Case e}

   END;       {Procedure DensidadEspectral}


PROCEDURE TransformadaDeFourier(VAR da,db,dc,dd:datos;VAR ra:resultado);

   {Calcula la Transformada de Fourier de cualquier arreglo de datos.
   Llama a rutina FFT definida al inicio del programa

   Entradas: Arreglos da,db,dc,dd, tipo datos
   Salida:   Arreglo ra, tipo resultado, que contiene la transformada del
             arreglo correspondiente}

   VAR
      r:datos;
```

```
BEGIN
   clrscr;
   writeln('         ESTA RUTINA OBTIENE LA TRANSFORMADA DE FOURIER');
   writeln('      DE UN ARREGLO DE DATOS Y LA DEPOSITA EN EL ARREGLO R');
   writeln;
   REPEAT
      write('Desea continuar (S/N)?  ');
      read(kbd,e);
      writeln(e);
   UNTIL e in ['S','s','N','n'];
   writeln;

   FOR i:=1 to 1024 do
      FOR j:=1 to 2 do
         ra[i,j]:=0.0;

   CASE e of
     'S','s': BEGIN
                 REPEAT
                    write('(Transformada Directa (D) o Inversa (I) ?  ');
                    read(kbd,g);
                    writeln(g);
                 UNTIL g in ['D','d','I','i'];

                 REPEAT
                    write('(Qué arreglo desea transformar (A,B,C,D)?  ');
                    read(kbd,f);
                    writeln(f);
                 UNTIL f in ['A','a','B','b','C','c','D','d'];

                 CASE g of

                 'D','d': CASE f of
                             'A','a': FFT(da,r,9,-1);
                             'B','b': FFT(db,r,9,-1);
                             'C','c': FFT(dc,r,9,-1);
                             'D','d': FFT(dd,r,9,-1);
                          END;
                 'I','i': CASE f of
                             'A','a': FFT(da,r,9,1);
                             'B','b': FFT(db,r,9,1);
                             'C','c': FFT(dc,r,9,1);
                             'D','d': FFT(dd,r,9,1);
                          END;
                 END;
```

```
                FOR i:=1 to 512 do
                    ra[i,1]:=r[i,1];
                writeln;
                write('LISTO. El resultado se encuentra en el ');
                writeln('arreglo R');
              END;

      END;          {Case e}

   END;             {Procedure TransformadaDeFourier}

PROCEDURE EliminaRetraso(VAR ra:resultado);

   BEGIN
      writeln('Cuantos puntos desea eliminar?');
      readln(numarr);
      FOR i:=1 to 1024-numarr do
         ra[i,1]:=ra[i+numarr,1];
   END;

PROCEDURE Menu;

   BEGIN
      clrscr;
      writeln;
      writeln;
      writeln('            PROCESAMIENTO BASICO DE SE%ALES');
      writeln;
      writeln('             (A) Adquisición');
      writeln('             (C) Convolución');
      writeln('             (D) Despliegue');
      writeln('             (E) Elimina retraso del filtro');
      writeln('             (F) Filtrado');
      writeln('             (G) Graficacion');
      writeln('             (L) Lectura y almacén en disco');
      writeln('             (I) Intercambio de arreglos');
      writeln('             (M) Menú');
      writeln('             (N) Normalización de arreglos');
      writeln('             (O) Cálculo de la magnitud');
      writeln('             (Q) Densidad espectral');
      writeln('             (S) Regresa a Turbopascal');
      writeln('             (T) Transformada de Fourier');
      writeln;

   END;
```

```
BEGIN                          {Programa principal}
    clrscr;
    FOR  i:=1 to 512 do        {Inicializa arreglos}
       FOR  j:=1 to 2 do
          BEGIN
             da[i,j]:=0.0;
             db[i,j]:=0.0;
             dc[i,j]:=0.0;
             dd[i,j]:=0.0;
           END;

    Menu;                      {Presenta menu}

    REPEAT                     {Lee opcion y ejecuta}
       writeln;
       write('OPCION?  ');
       read(kbd,d);
       writeln(d);
       CASE d OF

            'A','a': Adquisicion(da,db,dc,dd);
            'C','c': Convolucion(da,db,ra);
            'D','d': Despliegue(da,db,dc,dd,ra);
            'E','e': EliminaRetraso(ra);
            'F','f': Filtrado(da,dc);
            'G','g': Graficacion(da,db,dc,dd);
            'L','l': LecturaYAlmacenEnDisco(da,db,dc,dd);
            'I','i': IntercambioDeArreglos(da,db,dc,dd,ra);
            'M','m': Menu;
            'N','n': Normalizacion(da,db,dc,dd);
            'O','o': Calculamagnitud(da,db,dc,dd,ra);
            'Q','q': DensidadEspectral(da,db,dc,dd,ra);
            'T','t': TransformadaDeFourier(da,db,dc,dd,ra);
            'S','s': BEGIN
                     END;

            ELSE  writeln('OPCION INCORRECTA');
            END;

    UNTIL d in ['S','s'];

END.        {Programa principal}
```

APENDICE C

CONSIDERACIONES DE ERROR

El uso de una computadora y un lenguaje de programación para realizar operaciones aritmeticas o efectuar ciertas funciones más complejas, introduce errores de diversos tipos. El tamaño de la palabra del procesador, el tipo de aritmética, la codificación de la función, son algunas de las causas que provocan diferencias entre el resultado obtenido por el algoritmo y la función o resultado que se espera obtener.

De estos errores se puede mencionar especialmente aquellos debidos al proceso de cuantización de una señal analógica, es decir, al proceso de conversión analógica/digital. En el capitulo III se menciona el error de superposición espectral, debido a una falla en la frecuencia de la toma de muestras. Igualmente, la amplitud de la señal debe ser considerada en la codificación de la señal. En este punto se puede mencionar el error de saturación, que sucede cuando el valor de la señal cae fuera del rango que permite el convertidor analógico/digital a la entrada. En esos puntos, el valor de la señal se considera como el código máximo o mínimo que entrega el convertidor, y puede ser muy diferente del valor real. Este error se evita variando la ganancia del amplificador a la entrada de la tarjeta de adquisición, de tal manera que a todos los valores de la señal les corresponda un código discreto.

En el caso contrario, cuando el valor de la señal a la entrada es demasiado pequeño, puede ocurrir que únicamente se empleen los valores de los bits menos significativos en la codificación. El numero de combinaciones posibles para codificar una señal está en función de la longitud de la palabra que maneja el convertidor (en el caso de un convertidor de 12 bits se tienen 4096 combinaciones). Cuando solamente se emplea un número reducido de éstas, se pierde resolución en la amplitud de la señal. Este error nuevamente se reduce ajustando la ganancia en la entrada de la tarjeta de adquisición, de tal forma que la señal ocupe casi todo el rango del convertidor.

El uso de un lenguaje de programación introduce también errores debidos a la representación que emplea para manejar los datos. Basicamente se pueden mencionar dos errores debidos a este factor:

- Error de truncamiento, en el que se tiene un numero fijo de bits para representar un dato, y la combinación correspondiente siempre será el código menor más cercano al número. El máximo error introducido en este caso depende del tamaño mínimo de cuantización y en ningun caso es mayor que éste.

- Error de redondeo, en donde el número se aproxima al código más cercano, ya sea hacia arriba o hacia abajo. El máximo error que se comete con esta representación nunca excede de la mitad del tamaño mínimo de cuantización.

Estos errores, aunque no pueden eliminarse totalmente, en algunos casos se pueden manipular de tal manera que al final de una serie de operaciones, se obtenga un valor que estadísticamente se acerque más a los valores reales.

Por otro lado, la cantidad de error total en un proceso se ve afectada por el tipo de aritmética empleado. En el modo de aritmética de punto fijo, la magnitud del error puede ser considerable, aunque se introduce error solamente en algunos tipos de operación, como la multiplicación. El empleo de aritmética de punto flotante resulta más adecuado porque el error generalmente es menor, pero se presenta tanto en multiplicaciones como en adiciones. El lenguaje de programación empleado para este paquete (Pascal) realiza operaciones con aritmética de punto flotante, por lo que se puede considerar que el error es mínimo.

Es difícil evaluar el efecto independiente de cada uno de los factores involucrados en la determinación del error. Sin embargo, se puede hacer una medición que involucre el algoritmo, el lenguaje de programación y el tipo de computadora empleados. Debido a que las rutinas que involucran un mayor número de operaciones y que intervienen en las otras funciones de procesamiento son la transformada de Fourier y la convolución, el error global fue evaluado en estas dos rutinas, de la forma en que se describe:

1. Transformada de Fourier. El algoritmo empleado para la transformada de Fourier funciona en ambos sentidos, es decir, la misma rutina es empleada para realizar una transformación directa como para una inversa, cambiando únicamente el signo de las exponenciales complejas y un factor de escalamiento a la salida. Ambos procesos involucran el mismo número de multiplicaciones y sumas complejas. Para evaluar el error de esta rutina, se eligió una señal sencilla conocida, que fue introducida como datos de entrada al algoritmo de transformación directa de Fourier; del resultado obtenido, se recuperó la señal original, a través ahora de la transformada inversa de Fourier. La señal reconstruída se comparó con la señal inicial, considerando que la diferencia corresponde al doble del error introducido por la rutina en el peor de los casos (Tabla C.1).

	(a)	(b)	(c)
1	1.000000000E+000	1.000000000E+000	0.000000000000000E+000
2	1.000000000E+000	1.000000000E+000	4.218847493575590E-015
3	1.000000000E+000	1.000000000E+000	1.776356839400250E-015
4	1.000000000E+000	1.000000000E+000	4.218847493575590E-015
5	1.000000000E+000	1.000000000E+000	6.661338147750940E-016
6	1.000000000E+000	1.000000000E+000	4.218847493575590E-015
7	1.000000000E+000	1.000000000E+000	1.776356839400250E-015
8	1.000000000E+000	1.000000000E+000	4.218847493575590E-015
9	1.000000000E+000	1.000000000E+000	6.661338147750940E-016
10	1.000000000E+000	1.000000000E+000	4.107825191113080E-015
11	1.000000000E+000	1.000000000E+000	1.998401444325280E-015
12	1.000000000E+000	1.000000000E+000	3.996802888650560E-015
13	1.000000000E+000	1.000000000E+000	6.661338147750940E-016
14	1.000000000E+000	1.000000000E+000	4.218847493575590E-015
15	1.000000000E+000	1.000000000E+000	1.776356839400250E-015
16	1.000000000E+000	1.000000000E+000	4.218847493575590E-015
17	0.000000000E+000	1.887212582E-017	1.887212582657830E-017
18	0.000000000E+000	-5.379832[illegible]E-017	5.379832111000310E-017
19	0.000000000E+000	-3.979814[illegible]E-016	3.979814448703711E-016
20	0.000000000E+000	1.065302[illegible]E-016	1.065302266553660E-016
21	0.000000000E+000	4.032486[illegible]E-017	4.032486680148640E-017
22	0.000000000E+000	1.231[illegible]E-016	1.231423662789351E-016
23	0.000000000E+000	2.827[illegible]E-016	2.827179016299820E-016
24	0.000000000E+000	4.106[illegible]E-016	4.106238616418660E-016
25	0.000000000E+000	-3.27[illegible]E-017	3.273462664426620E-017
26	0.000000000E+000	1.662[illegible]E-016	1.662819120364120E-016
27	0.000000000E+000	2.886[illegible]E-016	2.886530865080100E-016
28	0.000000000E+000	1.110[illegible]E-016	1.110765668695620E-016
29	0.000000000E+000	5.806[illegible]E-017	5.806420082647870E-017
30	0.000000000E+000	1.469[illegible]E-016	1.469189842473800E-016

Tabla C.1 (a) Señal cuadrada introducida a la rutina de transformación de Fourier
(b) Señal recuperada a partir de los coeficientes de Fourier de (a)
(c) Diferencia entre (a) y (b)

2. **Convolución.** En este caso, fue más sencillo evaluar la rutina considerando como entrada dos funciones conocidas, cuyo resultado al evaluar la convolución fuera determinado analíticamente. Se tomaron dos funciones cuadradas a la entrada, que fueron generadas por programa. La tabla C.2 muestra el resultado de la convolución obtenido empleando la rutina, las muestras de la función que se esperaba obtener y la diferencia entre ambas.

$$g(n) = x(n) = \begin{cases} 1 & 1 < n < 15 \\ 0 & \text{en otro caso} \end{cases}$$

$$y(n) = g(n) * x(n) = \sum_{k=-\infty}^{\infty} g(k)\, x(n-k)$$

$$y(n) = \begin{cases} 0 & -\infty < n < 1 \\ \sum_{k=1}^{n} 1 = n & 1 \leq n < 15 \\ \sum_{k=n-14}^{15} 1 = 30-n & 15 \leq n < 30 \\ 0 & 30 \leq n < \infty \end{cases}$$

	(a)	(b)	(c)
1	5.000000000E-002	5.000000000E-002	0.000000000000000E+000
2	1.000000000E-001	1.000000000E-001	0.000000000000000E+000
3	1.500000000E-001	1.500000000E-001	0.000000000000000E+000
4	2.000000000E-001	2.000000000E-001	0.000000000000000E+000
5	2.500000000E-001	2.500000000E-001	0.000000000000000E+000
6	3.000000000E-001	3.000000000E-001	0.000000000000000E+000
7	3.500000000E-001	3.500000000E-001	0.000000000000000E+000
8	4.000000000E-001	4.000000000E-001	0.000000000000000E+000
9	4.500000000E-001	4.500000000E-001	0.000000000000000E+000
10	5.000000000E-001	5.000000000E-001	0.000000000000000E+000
11	5.500000000E-001	5.500000000E-001	0.000000000000000E+000
12	6.000000000E-001	6.000000000E-001	0.000000000000000E+000
13	6.500000000E-001	6.500000000E-001	0.000000000000000E+000
14	7.000000000E-001	7.000000000E-001	0.000000000000000E+000
15	7.000000000E-001	7.000000000E-001	0.000000000000000E+000
16	7.000000000E-001	7.000000000E-001	0.000000000000000E+000
17	6.500000000E-001	6.500000000E-001	0.000000000000000E+000
18	6.000000000E-001	6.000000000E-001	0.000000000000000E+000
19	5.500000000E-001	5.500000000E-001	0.000000000000000E+000
20	5.000000000E-001	5.000000000E-001	0.000000000000000E+000
21	4.500000000E-001	4.500000000E-001	0.000000000000000E+000
22	4.000000000E-001	4.000000000E-001	0.000000000000000E+000
23	3.500000000E-001	3.500000000E-001	0.000000000000000E+000
24	3.000000000E-001	3.000000000E-001	0.000000000000000E+000
25	2.500000000E-001	2.500000000E-001	0.000000000000000E+000
26	2.000000000E-001	2.000000000E-001	0.000000000000000E+000
27	1.500000000E-001	1.500000000E-001	0.000000000000000E+000
28	1.000000000E-001	1.000000000E-001	0.000000000000000E+000
29	5.000000000E-002	5.000000000E-002	0.000000000000000E+000
30	0.000000000E+000	0.000000000E+000	0.000000000000000E+000

Tabla C.2 (a) Muestras obtenidas de la convolución de dos señales cuadradas, de longitud 15 y con intervalo de muestreo 0.5 seg., mediante la rutina de convolución
(b) Muestras obtenidas de la misma función, obtenidas analíticamente
(c) Diferencia entre (a) y (b)

APENDICE D

TIEMPOS DE EJECUCION

TABLA I. COMPARACION DE TIEMPOS DE EJECUCION DE DIFERENTES ALGORITMOS Y COMPUTADORAS PARA UNA TRANSFORMADA RAPIDA DE FOURIER DE 512 puntos.*

Signalcalc (DPC)	
(HP 68010)	.599
(HP 68020)	.173
Basic (HP 68010)	16.775
(HP 68020)	5.109
ILS-PC (ST)	
(PC- XT con 8087)	3.45
(PC XT sin 8087)	18.75
(PC AT con 80287)	1.75
(PC AT sin 80287)	6.52
DPO TEK Basic PDP-11	17.00
PROCESA	
(PC XT con 8087)	12.00
(PC XT sin 8087)	25.00
(PC AT sin 80287)	5.00

* Todos los tiempos están en segundos

TABLA 2. TIEMPOS DE EJECUCION PARA LAS DIFERENTES RUTINAS DEL PAQUETE DE PROCESAMIENTO (PROCESA) MEDIDOS EN UNA COMPUTADORA TIPO IMB-PC XT *

Transformada de Fourier (512 puntos)	12
Convolución (512 X 512 puntos)	182
Densidad espectral (512 puntos)	37
Magnitud (512 puntos)	1
Lectura de disco flexible	2
Almacenamiento en disco flexible	4

* Todos los tiempos se expresan en segundos