

# INTRODUCCIÓN

## 1.1 INTRODUCCIÓN

Una de las principales aportaciones de la presente tesis es la propuesta de una nueva familia de ventanas para estimación espectral y para el diseño de filtros digitales FIR. Estas ventanas han tenido muy buenos resultados en diversas aplicaciones de procesamiento de señales periódicas [2,3,4,5,46,47]; en particular en el diseño de filtros más precisos para medición fasorial.

Generalmente los diversos dispositivos tanto de medición fasorial como los detectores de fallas y disturbios utilizan el filtrado convencional de Fourier para calcular los fasores que alimentan los algoritmos que rigen las diversas funciones de protección, control, medición y registro dinámico de variables [18,36,37]. Esta técnica de filtrado utiliza los filtros Coseno y Seno para obtener las proyecciones ortogonales del fasor. Sus respuestas impulsionales se forman restringiendo mediante una ventana rectangular las funciones seno y coseno. Para eliminar las armónicas, se requiere que la duración de la ventana rectangular sea múltiplo de un ciclo, y que el tiempo de muestreo sea una fracción de ciclo. La desventaja de estos filtros es que son sensibles a las pequeñas desviaciones frecuenciales y permeables a la energía frecuencial interarmónica, lo que conduce a mediciones erróneas. Estas son limitaciones debidas a los importantes lóbulos laterales que la ventana rectangular tiene en su espectro de frecuencia.

Los nuevos filtros digitales obtenidos a partir de la nueva familia de ventanas se liberan de las condiciones impuestas por la ventana rectangular en duración y tiempo de muestreo. Eliminan la energía interarmónica, obteniendo mediciones más exactas, y son adecuados para la detección de fallas, localización y medición.

Actualmente ningún método de segmentación por ventana, diferente al de Fourier, se ha implementado en los equipos comerciales de medición. Por lo que la principal contribución del presente trabajo, al sugerir la implementación de las nuevas ventanas en los algoritmos de medición, adquiere mayor relevancia. Con el fin de detallar la elaboración de este trabajo a continuación se explican los antecedentes del proyecto, la operación general del sistema y los objetivos y estructura de la tesis.

## **1.2 ANTECEDENTES DEL PROYECTO DE TESIS**

El monitoreo de los sistemas eléctricos de potencia y el análisis de las señales registradas antes, durante y después de una disturbio son problemas de particular interés para la industria eléctrica, y su importancia se ha incrementado a un ritmo constante a través de los años [28].

En el pasado las líneas de transmisión tenían capacidad disponible, y en general, transmitían su potencia de un punto a otro con pocas interconexiones paralelas; la evaluación de las perturbaciones en este tipo de líneas era relativamente directa. Cuando los sistemas crecieron, se hicieron cada vez más complejos, en gran parte debido al aumento del número de interconexiones con redes de mayor carga. Los sistemas comenzaron a ser operados cerca de los límites de estabilidad debido a presiones ambientales y económicas; aumentando la dependencia en los sistemas de control para mantener la estabilidad del sistema. Debido a estas razones el análisis de las fallas se hizo más complejo y el monitoreo del sistema llegó a ser imprescindible para la confiabilidad y el mantenimiento del mismo [29]. A partir de los años 70, diversas compañías desarrollaron unidades de adquisición de datos para el registro de datos de prefalla, falla y postfalla con la finalidad de obtener información sobre la estabilidad transitoria del sistema [32,33,34,35].

Un resumen donde se estudia la instrumentación necesaria para monitorear el comportamiento dinámico de los SEP durante diferentes perturbaciones se presenta en [39]. En [29] se presenta otro resumen en donde se discute la experiencia adquirida por varias compañías eléctricas con respecto a la operación de varios equipos registradores de fallas y disturbios. En [30] se presenta la aplicación de los registradores de fallas y disturbios en análisis de protecciones. Por otra parte, en [31] se presenta la experiencia adquirida con varios registradores de fallas instalados en diferentes subestaciones, ahí se presenta el diseño y características de los dispositivos, los datos recolectados y la aplicación de éstos para identificar problemas en equipos de protección. En [40] se presenta el diseño, instalación y operación de un sistema de monitores dinámicos, cuyo objetivo es validar los modelos computacionales de un sistema de control; además se describe el empleo de las técnicas de Fourier en el análisis de disturbios del sistema.

Muchas razones justifican la importancia del registro de datos generados durante un disturbio [30,31], entre ellas se pueden considerar:

- La necesidad de un conocimiento real de los fenómenos dinámicos para la validación de las simulaciones generadas por computadora.
- La creación de bases de datos, por una parte, para probar y mejorar nuevos algoritmos de protección, y por otra, para elaborar estadísticas que son importantes en la validación de los modelos eléctricos.
- La necesidad de identificar fallas en los diferentes equipos, verificando la operación adecuada de los sistemas de protección y control.
- Con los datos registrados es posible obtener información relacionada con la estabilidad y la dinámica del sistema y muchos otros datos de interés.

Actualmente los equipos registradores permiten almacenar la información de los transitorios provocados por una falla. Estos equipos, basados en microprocesadores, poseen una alta capacidad de procesamiento, almacenamiento y comunicación de datos. Algunos de

ellos incluyen su propio *software* de análisis, se comunican con otros dispositivos mediante canales de fibra óptica y cuentan con canales especiales para la sincronización [42,43]. Estos equipos, orientados a las compañías eléctricas y a las grandes industrias, ofrecen un medio de detección y registro de las señales eléctricas durante la producción de una falla. También brindan la oportunidad de analizar posteriormente los registros almacenados. Los registros de datos se almacenan en archivos de longitud variable, e incluyen datos de prefalla, falla y postfalla [41,44]. La principal desventaja es su costo, y muchas veces usan protocolos cerrados por lo que crean una dependencia hacia el fabricante cuando requieren mantenimiento o cuando se desea modificar sus algoritmos internos. En particular, ofrecen datos fasoriales aplicando sus propios algoritmos sobre señales digitales, lo cual impide el acceso directo a éstas.

Otra fuerte limitación de dichos equipos es que su *software* está programado en una memoria fija (*EPROM*), por lo que es difícil de modificar. Esta falta de flexibilidad impide variar inteligentemente el procesamiento de datos y los modos de comunicación adaptándolos a las circunstancias propias del evento o del lugar.

Una de las contribuciones de la presente investigación es el diseño de un sistema para monitorear y registrar fallas en sistemas eléctricos de potencia [2]. El desarrollo y prueba del *software* en todas sus etapas constituye una parte importante del presente trabajo, la cual se describe ampliamente en [45]. La finalidad del sistema es la de recolectar información para el estudio de oscilaciones electromecánicas. Las partes que integran dicho sistema son (ver Fig. 1.1): los módulos de acondicionamiento de señales (tarjeta adecuadora), una computadora personal (computadora anfitriona), una tarjeta procesadora de alta velocidad y un módem. Cada registrador se encargará de detectar, registrar y comunicar las señales desde diferentes sitios estratégicos del Área Noreste del Sistema Nacional de Energía Eléctrica de CFE.

El proyecto piloto pretende conectar cinco registradores a un concentrador ubicado en el Centro Nacional de Control de Energía (CENACE) en Monterrey. Los datos se enviarán a este concentrador, desde los diferentes puntos del sistema, inmediatamente después de que se haya registrado una falla. Para transportar los datos, se emplea un programa de

telecomunicaciones, dicho programa forma parte de una red de transmisión de datos, la cual se encarga de transferirlos al concentrador, vía módem. La red lleva a cabo la sincronización de cada uno de los registradores y es capaz de establecer sesiones remotas para configurar el sistema. El problema de diseño de la red de datos constituye el tema de otro trabajo paralelo de investigación, por lo que aquí se supone que ya se dispone de dicha red.

Los registradores estarían ubicados en las subestaciones y plantas generadoras claves del Área Noreste, tales como: Río Escondido en Coahuila, Río Bravo y Altamira en Tamaulipas, San Jerónimo y Monterrey en Nuevo León.

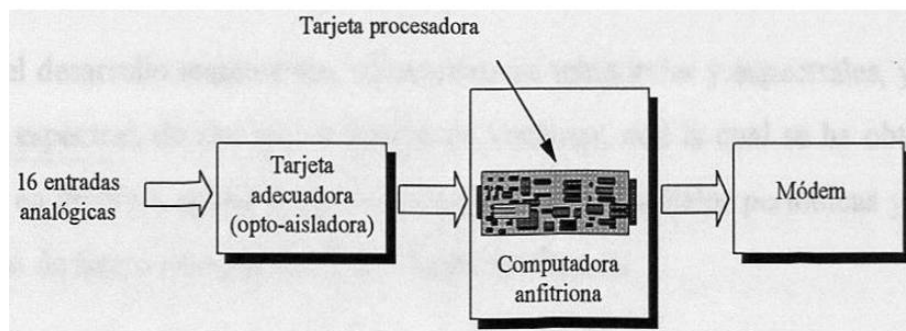


Fig. 1.1 Diagrama a bloques de los componentes físicos que integran el sistema.

### 1.3 OPERACIÓN GENERAL DEL SISTEMA

La función principal de la tarjeta procesadora es la detección de la falla. Para ello adquiere y procesa las señales de voltaje y corriente. Tiene capacidad para adquirir 16 entradas analógicas empleando una frecuencia de muestreo de 960 Hz (16 muestras por ciclo) en cada entrada. La detección del instante de la falla se realiza aplicando un esquema de filtrado digital a las señales de voltaje y corriente. Dada la frecuencia de muestreo utilizada, el registrador operará en la banda de frecuencias de 0 a 480 Hz. El detector se activará ante fenómenos de baja frecuencia como la componente aperiódica exponencial o la energía de la discontinuidad que produce una falla. Al momento de detectar la falla, el algoritmo memoriza un intervalo de prefalla, de hasta 1.6 segundos, y uno de postfalla, de hasta varias horas de duración. El algoritmo de detección es configurado por la computadora anfitriona. El traspaso de la función de detección al procesador de la tarjeta, permite que el procesador de la

computadora anfitriona atiende otras tareas, y sólo cuando ocurre una falla se encarga de registrarla. Cuando el algoritmo de detección advierte la existencia de una falla, interrumpe el programa de la computadora anfitriona, con el objeto de transferirle los datos. El programa de la computadora anfitriona registra el intervalo de prefalla y controla el intervalo de grabación de postfalla. Una vez que se ha registrado una falla, la computadora anfitriona se enlaza con el concentrador de datos a través de la red telefónica nacional.

#### 1.4 OBJETIVOS DE LA TESIS

Los objetivos principales de la tesis son:

- Presentar el desarrollo matemático, características temporales y espectrales, y aplicaciones en análisis espectral, de una nueva familia de ventanas, con la cual se ha obtenido buenos resultados en diversas aplicaciones de procesamiento de señales periódicas y en particular en el diseño de filtros más precisos para medición fasorial.
- Presentar el análisis matemático, ecuaciones de diferencia y características espectrales de una nueva familia de filtros peine denominados filtros pasapares y pasaimpares.
- Evaluar los filtros pasapares ante diversas señales de falla y comparar los resultados con los obtenidos con otros esquemas de filtrado.
- Diseñar un sistema para registrar las señales trifásicas de voltaje y corriente en las subestaciones de un sistema eléctrico de potencia. El sistema deberá activarse automáticamente y deberá registrar dichas señales al momento de presentarse una falla. Para ello emplea un algoritmo de detección, el cual se ejecuta en una tarjeta procesadora de alta velocidad.
- Describir el funcionamiento y características esenciales de la tarjeta procesadora.
- Elaborar un *software* que interactúe con la tarjeta procesadora. Así mismo, elaborar el *software* del algoritmo de detección. En ambos casos, explotando las primitivas proporcionadas por el fabricante de la tarjeta.

- Elaborar el *software* del sistema acompañado de documentación detallada a fin de poder realizar futuras modificaciones que otras aplicaciones requieran.
- Crear una base de datos a partir de los registros de falla, para explotar su contenido realizando los siguientes estudios: medición fasorial, variación dinámica de la frecuencia, medición de potencias activas y reactivas, análisis espectral (espectrogramas), identificación de los modelos dinámicos de oscilaciones electromecánicas y estudios de perturbaciones en generadores.

## 1.5 ESTRUCTURA DE LA TESIS

En la tesis se contemplan seis Capítulos y referencias bibliográficas. A continuación se describe brevemente el contenido de cada Capítulo:

En el Capítulo I se introduce y describe la importancia de la tesis, se presenta el funcionamiento general del sistema, se establecen los objetivos fundamentales del trabajo y su estructura general.

En el Capítulo 2 se describen las características de la tarjeta procesadora (organización de su *hardware* y *software*) y el funcionamiento de su sistema operativo. Además, se presenta la forma en que se lleva a cabo la comunicación con la computadora anfitriona y la manera de programar tareas de procesamiento.

En el Capítulo 3, como parte fundamental de la presente investigación, se presenta la deducción matemática de una nueva ventana para estimación espectral. Se presenta su desarrollo matemático, características temporales y espectrales, se realiza una comparación con otras ventanas clásicas, se presentan aplicaciones en análisis armónico y en espectrogramas. Finalmente se ilustra su aplicación en el diseño de filtros digitales FIR. De esta última aplicación se obtiene una nueva familia de filtros digitales muy útiles para medición fasorial cuya característica principal es la de poseer una excelente banda de rechazo.

En el Capítulo 4 se presenta el desarrollo matemático, ecuaciones de diferencia y características espectrales de una nueva familia de filtros tipo peine, derivados de los filtros

anteriores. Por sus características espectrales se les denomina filtros pasapares y pasaimpares. Estos filtros se proponen para la detección de fallas en SEP. También se presenta el diseño de un detector de fallas, el cual consiste de un filtro acoplado que permite el paso de la energía en la banda base y de un comparador de umbral que señala cuando dicha energía sobrepasa un cierto nivel. Se describen sus características espectrales y temporales. Finalmente, se evalúa el comportamiento tanto de los filtros pasapares así como el del detector ante diversas señales de falla.

En el capítulo 5, como una aplicación práctica de la presente investigación, se presenta una descripción exhaustiva de la operación del sistema, mostrando para ello los diagramas de transición de estado tanto del programa principal que se ejecuta en la computadora anfitriona como del algoritmo de detección ejecutado en la tarjeta procesadora. Se describe la forma en que se lleva a cabo la transferencia y registro de datos entre estos dos dispositivos y la forma en que el sistema interactúa con el programa de telecomunicaciones.

Por último, en el Capítulo 6 se presentan las conclusiones generales de la tesis y se mencionan sus aportaciones principales. Se dan, asimismo, recomendaciones para trabajos futuros en esta área.



# CARACTERÍSTICAS DEL PROCESADOR DE SEÑALES

## 2.1 INTRODUCCIÓN

En la actualidad los sistemas de procesamiento de señales son ejecutados en procesadores digitales de señales (DSP, *Digital Signal Processor*), el uso de estos dispositivos se ha incrementado en forma exponencial en la última década, principalmente debido al gran desarrollo de la tecnología de los circuitos integrados [13]. Estos han aumentado considerablemente su velocidad de procesamiento y su densidad de integración, permitiendo la implementación en tiempo real de algoritmos cada vez más complejos tanto en número de operaciones como en volumen de datos procesados. Esta tecnología ha ofrecido ya soluciones eficientes y abierto las puertas a una amplia variedad de nuevas aplicaciones en telefonía, navegación, comunicación de datos, sonar, radar, sismología, etc.

Dada la importancia de estos dispositivos, en el presente Capítulo y como una aplicación práctica de la presente investigación, se describen las características de la tarjeta procesadora, en la cual se ejecutan los filtros digitales desarrollados en esta tesis. Cabe aclarar que dichos filtros forman parte del algoritmo encargado de la detección de fallas.

La tarjeta procesadora es un sistema integral de adquisición y procesamiento de datos que ocupa una ranura de expansión en una computadora personal (computadora anfitriona). Combina un *hardware* de adquisición de datos con un microprocesador de 16 *bits*, una memoria intermedia y un sistema operativo multitarea de tiempo real. La tarjeta procesadora maneja todos los detalles de bajo nivel de la adquisición de datos mientras ejecuta cálculos en tiempo real. Esto permite que la computadora anfitriona se encargue solamente de la

administración de archivos y de la interacción con el usuario y con otros periféricos. En este capítulo se presentan sus características y funcionamiento. Se describe su sistema operativo y las herramientas de *software*, las cuales apoyan y facilitan la realización de aplicaciones. Se presenta la forma de establecer comunicación mediante programas escritos en lenguaje de programación C. Para ello, se muestran varios ejemplos de aplicación y diversas alternativas para la transferencia de datos. Finalmente, se describen los pasos necesarios para diseñar y programar tareas de procesamiento según las especificaciones de una aplicación.

## 2.2 ORGANIZACIÓN DEL HARDWARE

La Fig. 2.1 muestra la organización del *hardware* de la tarjeta procesadora. Se puede observar que posee los componentes principales de una microcomputadora: microprocesador, memoria RAM y ROM, entradas y salidas tanto digitales como analógicas, circuitos de sincronización y control. Además de controladores de acceso directo a memoria DMA (*Direct Memory Access*) y *buffers* FIFO (*first in-first out*), cuenta con un microprocesador Intel 486 para emular el procesador de señales digitales. Esto permite una mejor flexibilidad en la gestión del sistema, especialmente en el manejo de entradas de datos. Además, las tareas de procesamiento de señales digitales se programan con un compilador de lenguaje C genérico, normalmente el mismo que se utiliza para programar la computadora anfitriona.

Los circuitos de tiempo controlan las entradas y salidas, descargando al procesador de dichas tareas. Los controladores FIFO y DMA son responsables de la transferencia rápida de datos hacia la memoria de la computadora anfitriona con intervención mínima del procesador. Los voltajes de entrada son multiplexados y luego amplificados. El proceso de cuantificación utiliza un cuantificador con 12 *bits* de resolución. Dentro de la tarjeta procesadora estos números son representados mediante enteros de 16 *bits*.

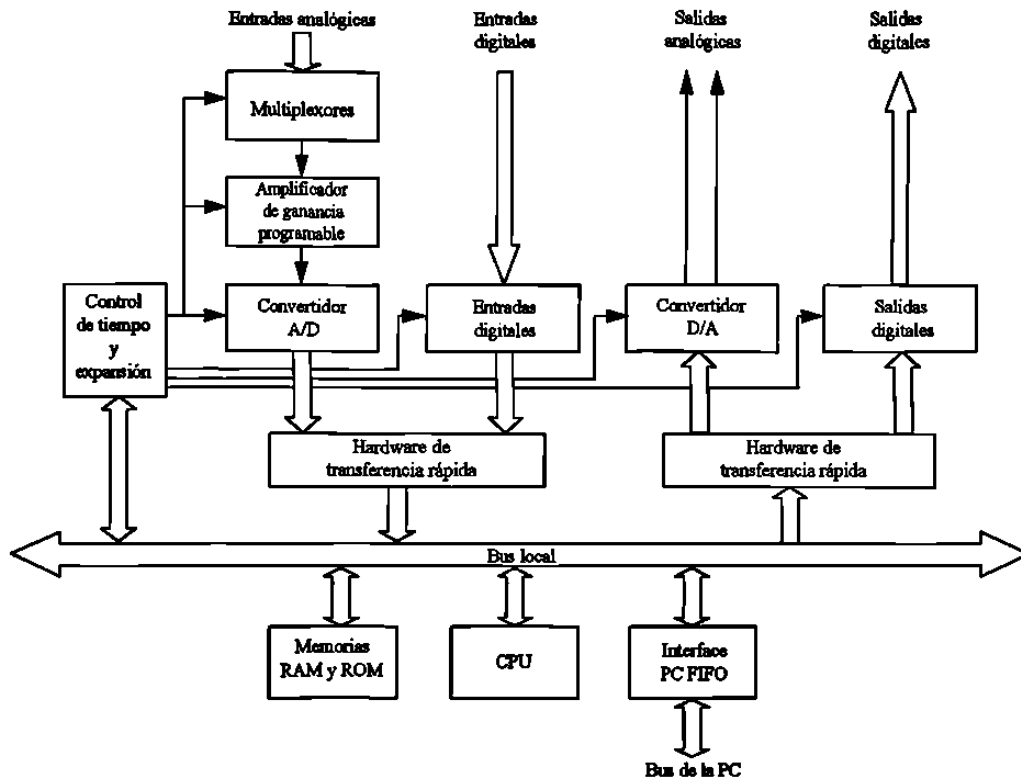


Fig. 2.1 Organización del *hardware* de la tarjeta procesadora.

La Tabla 2.1 presenta un resumen de las principales especificaciones técnicas de la tarjeta procesadora. Una información más completa y detalles para la instalación de su *hardware* y *software* se presenta en [21].

Tabla 2.1 Especificaciones técnicas de la tarjeta procesadora

Entradas analógicas (asíncronas):	16
Niveles de entrada:	0 a 5 volts, -2.5 a 2.5 volts, -5 a 5 volts y -10 a 10 volts
Frecuencia de muestreo máxima:	769 KHz
Ganancias por entrada analógica:	1, 10, 100 y 500.
Entradas discretas (sincronas):	16
Salidas analógicas:	2
Niveles de salida:	Idénticos a los niveles de entrada.
Salidas discretas:	16
Microprocesador:	Intel 486SX2
Velocidad del procesador:	48 MHz
Memoria RAM:	4 Mb
Resolución del convertidor análogo digital (A/D):	12 bits
Procesador digital de señales:	Emulación en el procesador principal

En la siguiente sección se explica el proceso de conversión A/D para las señales analógicas de entrada.

### 2.3 TRATAMIENTO DE LAS SEÑALES ANALÓGICAS DE ENTRADA

Las entradas y salidas analógicas de la tarjeta procesadora son señales de voltaje. La tarjeta procesadora convierte los niveles de voltaje de cada entrada a enteros, mediante un convertidor A/D.

Los voltajes en cada entrada analógica son amplificados y alimentados a un convertidor A/D. La ganancia del amplificador varía de 1 a 500 y se especifica independientemente para cada señal de entrada. Cada vez que se muestrea una señal analógica de entrada, el convertidor A/D devuelve un entero de 16 *bits*. El intervalo válido de los voltajes de entrada depende de la ganancia del amplificador y del rango dinámico del convertidor A/D. La Tabla 2.1 muestra los intervalos de las señales de entrada que pueden ser configurados. Esta configuración es llevada a cabo mediante interruptores implementados en *hardware*; detalles de esta configuración se describe ampliamente en [21].

Para un intervalo de entrada bipolar, las posibles salidas del convertidor A/D variarán de -32768 a +32752. Esto corresponde a una variación de la señal de entrada desde -X hasta ligeramente abajo de +X, donde X representa el valor de saturación del convertidor A/D (2.5, 5 o 10 volts).

La siguiente ecuación relaciona el voltaje de entrada del convertidor A/D en función de su salida y del valor de saturación del mismo:

$$V = \frac{R}{32768} X, \quad (2.1)$$

donde,  $V$  representa el voltaje de entrada al convertidor A/D,  $R$  su salida y  $X$  el valor de saturación.

Con el objeto de comprender el funcionamiento del *software*, a continuación se describe su organización y se presenta una introducción a su sistema operativo.

## 2.4 ORGANIZACIÓN DEL SOFTWARE

La tarjeta procesadora cuenta con un sistema operativo (SO) multitarea de tiempo real [22] y con una amplia variedad de programas y herramientas de *software* que simplifican la realización de aplicaciones [23,24,25].

En el SO, un procedimiento es definido como un grupo de instrucciones que ejecutan alguna función. El SO permite que el usuario defina procedimientos de entrada, de procesamiento y de salida. En los del segundo tipo se encuentran las instrucciones que definen tareas de procesamiento. Cuando un procedimiento está activo, todas las tareas que integran el procedimiento se ejecutan simultáneamente. Una misma instrucción puede usarse varias veces para definir distintas tareas y cada tarea puede procesar datos provenientes de diferentes entradas. Por último, una aplicación comúnmente está integrada por un procedimiento de entrada, uno de procesamiento y posiblemente por uno de salida.

## 2.5 INTRODUCCIÓN AL SISTEMA OPERATIVO

El SO es cargado en la memoria RAM de la tarjeta. Cuando se ejecuta alguna aplicación, el SO controla el proceso de muestreo de las señales analógicas y organiza las diversas tareas que en ese momento se están ejecutando. Una tarea definida en el SO se encargará de procesar datos y transferirlos hacia la computadora anfitriona para su ulterior procesamiento. Las tareas se comunican a través de memorias intermedias (*buffers*) dinámicas, a las cuales se puede agregar o sustraer datos, reteniéndolos hasta que sean procesados. El SO realiza automáticamente la asignación de memoria, permitiendo su expansión y contracción; conserva los datos de las memorias intermedias en el orden correcto y realiza tareas de sincronización. Si alguna tarea excede el tiempo asignado o no puede proceder por cualquier otra razón, el SO continúa con la siguiente tarea.

El SO también proporciona memorias especiales llamadas “disparadores”, las cuales se usan principalmente en aplicaciones que requieren la sincronización de diferentes tareas. Una tarea se define mediante una instrucción y especificando uno o más de los siguientes parámetros: constantes, variables, memorias intermedias y “disparadores”.

La Fig. 2.2 presenta un diagrama que muestra la organización del *software* para una aplicación típica.

### 2.5.1 Intérprete de instrucciones

El intérprete de instrucciones del sistema operativo es un programa que siempre se encuentra activo. Su función principal es analizar y ejecutar las instrucciones que son enviadas de la computadora anfitriona hacia la tarjeta procesadora.

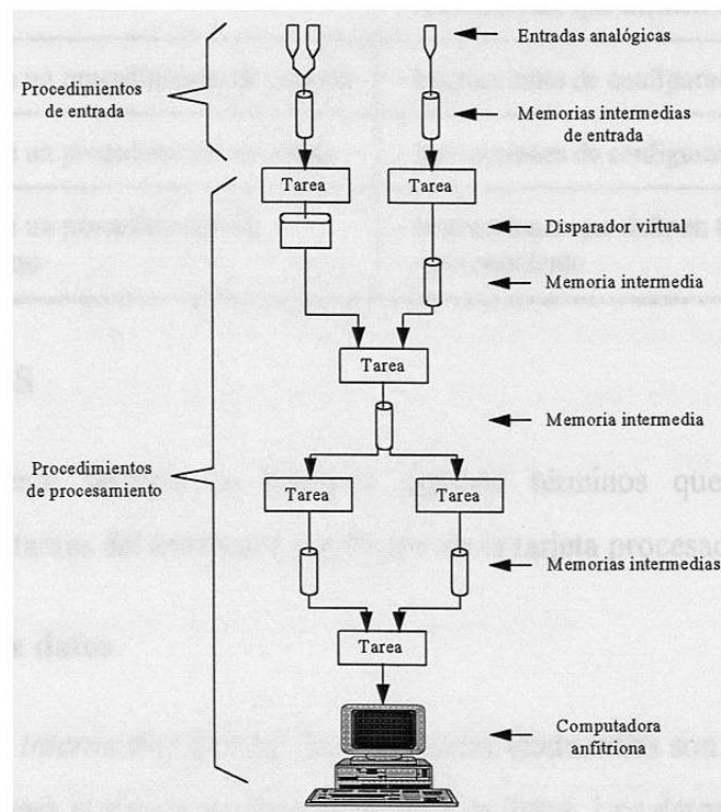


Fig. 2.2 Organización del *software* para una aplicación típica.

Las instrucciones del sistema operativo se dividen en las siguientes categorías: instrucciones del sistema, instrucciones que definen estructuras de datos, instrucciones de configuración e instrucciones que definen tareas de procesamiento. Las de configuración se subdividen a su vez en: instrucciones de configuración de entrada y salida.

## 2.5.2 Estados del intérprete de instrucciones

El intérprete de instrucciones tiene diferentes estados de operación. Cada clase de instrucciones debe introducirse cuando el intérprete se encuentra en un estado apropiado. La Tabla 2.2 muestra las instrucciones que pueden introducirse en cada estado.

Tabla 2.2 Estados del intérprete de instrucciones del SO

Estado	Tipos de instrucciones
1.- Sistema	- Instrucciones del sistema - Instrucciones que definen estructuras de datos
2.- Definición de un procedimiento de entrada	- Instrucciones de configuración de entrada
3.- Definición de un procedimiento de salida	- Instrucciones de configuración de salida
4.- Definición de un procedimiento de procesamiento	- Instrucciones que definen tareas de procesamiento

## 2.6 DEFINICIONES

En la presente sección se explican algunos términos que se relacionan con características importantes del *hardware* y *software* de la tarjeta procesadora.

### 2.6.1 Estructuras de datos

1) *Memorias intermedias (pipes)*: Las memorias intermedias son *buffer* FIFO (*first in-first out*) utilizadas para el almacenamiento temporal de datos. Los datos son agregados en un extremo de la memoria y son extraídos del extremo opuesto, de esta manera, los datos extraídos se encuentran en el mismo orden en que fueron introducidos. Por su parte, si los datos son agregados a una velocidad mayor que a la que son extraídos, el tamaño de la memoria se incrementa. El espacio de almacenamiento es asignado y liberado automáticamente por el SO, permitiendo que la memoria se expanda o contraiga tanto como sea requerido.

2) “Disparadores” (*Triggers*): Los “disparadores” son memorias intermedias especiales que permiten una comunicación sincronizada entre diferentes tareas. Existen dos clases de tareas que utilizan este tipo de memorias. Las primeras, se encargan de insertar un disparo sólo cuando ocurre un evento específico, en este momento, dicha tarea almacena el número de muestra del dato causante del disparo en esta memoria especial. Las segundas, esperan a que la tarea anterior inserte un disparo antes de emprender una acción específica; en el momento en que detecta un disparo, procesa datos relativos al mismo, teniendo capacidad para procesar datos generados antes y después del evento.

### **2.6.2 Memoria intermedia de entrada (*Input channel pipe*)**

Las memorias intermedias de entrada son memorias especiales en las que el *hardware* de la tarjeta procesadora coloca los valores de salida del convertidor A/D y los datos provenientes de las entradas discretas. Cada memoria intermedia de entrada se asocia a una entrada analógica, o en su caso, al puerto de comunicación binario. Las entradas analógicas son digitalizadas secuencialmente por el convertidor A/D. Algunas de ellas pueden ignorarse y también pueden tener ganancias diferentes. Varias tareas pueden leer datos de una misma memoria intermedia de entrada.

### **2.6.3 Memoria intermedia de salida (*Output channel pipe*)**

Una memoria intermedia de salida es una memoria especial, a partir de la cual el *hardware* de la tarjeta procesadora dispone de datos para alimentar al convertidor D/A o para alimentar las salidas discretas. Cada memoria intermedia de salida se asocia a una salida analógica o al puerto de comunicación binario.

### **2.6.4 Memorias intermedias de comunicación (*Communication pipes*)**

Las memorias intermedias de comunicación son usadas para coordinar la transferencia de datos entre la tarjeta procesadora y la computadora anfitriona. La comunicación se establece a través del bus principal de la computadora anfitriona. El SO coloca los datos que son enviados de la computadora anfitriona hacia la tarjeta procesadora en una memoria



intermedia de comunicación de entrada. Así mismo, los datos que serán transferidos de la tarjeta procesadora hacia la computadora anfitriona, se colocan en una memoria intermedia de comunicación de salida. La comunicación puede realizarse de dos formas: en formato texto o binario. La tarjeta procesadora define automáticamente cuatro memorias intermedias las cuales llevan a cabo ambos tipos de comunicación: \$SYSIN, \$SYSOUT, \$BININ y \$BINOUT. Las primeras dos se usan para establecer una comunicación en formato texto y las restantes para comunicación binaria; su descripción se presenta a continuación:

1. \$SYSIN. Los datos que son transferidos de la computadora anfitriona hacia la tarjeta procesadora se colocan en esta memoria. El intérprete de instrucciones lee los datos presentes en dicha memoria para su ulterior procesamiento.
2. \$SYSOUT. Los datos e información del sistema (mensajes de error e información de estado) que son transferidos de la tarjeta procesadora hacia la computadora anfitriona se colocan en esta memoria.
3. \$BININ y \$BINOUT son memorias intermedias de comunicación de entrada y salida, respectivamente. Se usan para transferir datos binarios a alta velocidad mediante DMA y evitan la conversión de formato texto a binario y viceversa, lo cual las hace más eficientes que \$SYSIN y \$SYSOUT.

A continuación se describen los diferentes tipos de instrucciones que utiliza el sistema operativo.

## 2.7 INSTRUCCIONES DEL SISTEMA OPERATIVO

El SO cuenta con una gran variedad de instrucciones, las cuales realizan diferentes funciones dentro de un estado de operación determinado. En esta sección se presenta una descripción de cada una de estas instrucciones, mostrando para ello sus funciones específicas. Una información más detallada de éstas se encuentra en [22,25].

*1) Instrucciones del sistema:* Estas instrucciones inician y detienen el proceso de muestreo, establecen las opciones del SO y solicitan información de estado.

2)*Instrucciones que definen estructuras de datos:* Estas instrucciones ejecutan el siguiente conjunto de tareas relacionadas:

- definen símbolos del sistema (nombres de variables, constantes, memorias intermedias, disparadores, etc.);
- asignan memoria dinámicamente,
- inicializan el área de memoria apropiada.

3)*Instrucciones de configuración de entrada:* Definen la configuración del *hardware* de muestreo de la tarjeta procesadora.

4)*Instrucciones de configuración de salida:* Definen la configuración del *hardware* de salida (convertidor D/A) de la tarjeta procesadora.

5)*Instrucciones que definen tareas de procesamiento:* Al declarar una instrucción que define una tarea de procesamiento, la tarjeta procesadora establece una estructura adecuada para ejecutarla. Una tarea de procesamiento usualmente lee datos de una o más memorias intermedias, los procesa en alguna forma, y posteriormente registra los resultados en otra u otras memorias. Existen dos clases de tareas: las predefinidas por el SO y las que son programadas por el usuario, estas últimas posteriormente son cargadas al SO mediante programas especializados.

Cuando se inicia un procedimiento, se inicia a su vez la ejecución de las tareas que lo integran. Éstas son ejecutadas por programas altamente optimizados escritos en lenguaje ensamblador o lenguaje C.

A continuación se describen las instrucciones utilizadas para transferir datos hacia la computadora anfitriona.

## **2.8 TRANSFERENCIA DE DATOS HACIA LA COMPUTADORA ANFITRIONA**

El objetivo de esta sección es presentar las características y funcionamiento de las instrucciones que se utilizan para transferir datos hacia la computadora anfitriona. Existen instrucciones que realizan esta transferencia en formato texto y otras en formato binario. Las instrucciones PRINT y FORMAT se usan para transferir datos en formato texto. Ambas instrucciones formatean los datos en caracteres ASCII antes de que sean enviados hacia la computadora anfitriona.

Para transferir datos binarios a alta velocidad se utiliza la instrucción BPRINT. Esta instrucción trabaja en forma similar a la instrucción PRINT, excepto que los datos son transferidos directamente en modo binario. Otra forma de transferir datos binarios hacia la computadora anfitriona, se realiza al definir la memoria intermedia \$BINOUT como la memoria intermedia de salida de alguna tarea. En la siguiente sección se explican las recomendaciones para aumentar la eficiencia de una aplicación.

## **2.9 CONSIDERACIONES PARA AUMENTAR LA EFICIENCIA DE UNA APLICACIÓN**

La eficiencia de una aplicación dependerá del volumen de datos de entrada a procesar, de la cantidad y tipo de tareas y del número de operaciones, de la forma en que se transferirán los datos procesados hacia la computadora anfitriona, de la velocidad del reloj de la tarjeta procesadora y de la computadora anfitriona, y de otros factores. La tarjeta procesadora invierte una pequeña cantidad de tiempo cada vez que deposita o toma un dato de una memoria intermedia. Por lo que, cada operación innecesaria limitará la máxima velocidad de procesamiento que pueda alcanzar.

En la mayoría de las aplicaciones, se recomienda procesar los datos en la tarjeta procesadora, en vez de transferirlos hacia la computadora anfitriona para su procesamiento. Este paralelismo permite que la carga computacional recaiga en la tarjeta procesadora, mientras que la computadora anfitriona se dedica a otras tareas (atención del *mouse*, teclado, impresora, interfaz gráfica, etc.). Esta recomendación fue tomada en cuenta durante el

desarrollo del programa del registrador: el algoritmo de detección lo ejecuta la tarjeta procesadora, mientras que la computadora anfitriona atiende al teclado o al módem.

Para aumentar la eficiencia durante la transferencia de datos hacia la computadora anfitriona, es recomendable transferir bloques de datos utilizando el formato binario, en vez del formato ASCII. Hay que destacar que la transferencia de bloques binarios se lleva a cabo mediante los controladores DMA. Esto permite que los datos pasen de la memoria interna de la tarjeta procesadora hacia la memoria RAM de la computadora anfitriona con intervención nula del procesador. Este procedimiento fue el utilizado en el presente trabajo, principalmente durante la transferencia de los datos de falla hacia la computadora anfitriona.

## **2.10 DESBORDAMIENTO DE DATOS DE UNA MEMORIA INTERMEDIA**

Cuando se realiza un muestreo de alta frecuencia, las memorias intermedias de entrada pueden llenarse rápidamente, si los datos presentes en dichas memorias no son procesados a una velocidad mayor que la de su llegada, ocurrirá un desbordamiento o incapacidad para almacenar más datos de entrada. En este momento, la tarjeta procesadora suspenderá el muestreo, sin perder los datos almacenados, reanudándolo una vez que éstos han sido procesados.

Un desbordamiento generalmente ocurre en las memorias intermedias de entrada, aunque también se puede presentar en las memorias intermedias definidas por el usuario.

El desbordamiento puede prevenirse reduciendo la frecuencia de muestreo, cambiando la configuración de la tarjeta a una forma más eficiente; ejecutando tareas de procesamiento más eficientes o programando tareas de procesamiento personalizadas utilizando el lenguaje C. Esta última opción, es la más adecuada para llevar a cabo tareas de procesamiento complejas, ya que toma ventaja de la potencialidad del lenguaje C, al permitir programar tareas más eficientes y con las características adecuadas para la aplicación. Tomando en cuenta estas ventajas, en el presente trabajo se seleccionó esta opción para programar el algoritmo de detección. En la siguiente sección se mencionan las principales herramientas de *software* con las que cuenta la tarjeta procesadora.

## **2.11 HERRAMIENTAS DE SOFTWARE PARA EL DESARROLLO DE APLICACIONES**

La tarjeta procesadora cuenta con un programa interactivo para el desarrollo de aplicaciones, integrado principalmente por un editor de texto, herramientas gráficas, y facilidades para el procesamiento y registro de datos. Además, proporciona una serie de programas que permiten realizar pruebas de evaluación, registro de datos, implementación de filtros digitales e integración de nuevas tareas de procesamiento en el SO de la tarjeta. Cada uno de estos programas, se describe ampliamente en [22].

A continuación se ilustra la forma en que interactúa la tarjeta procesadora con los lenguajes de alto nivel que residen en la computadora anfitriona.

## **2.12 PROGRAMACIÓN CON LENGUAJES DE ALTO NIVEL**

Los programas residentes en la computadora anfitriona se comunican con la tarjeta procesadora a través del controlador ACCEL. El ACCEL es un dispositivo para el sistema operativo DOS proporcionado por el fabricante de la tarjeta y su función es implementar el protocolo de comunicación entre la computadora anfitriona y la tarjeta procesadora.

Cualquier lenguaje de programación que soporte dispositivos de entrada y salida puede establecer comunicación con la tarjeta procesadora a través del controlador ACCEL. El fabricante proporciona herramientas de *software* para establecer esta comunicación empleando los lenguajes de programación BASIC, Pascal, C y FORTRAN [23]. En el presente trabajo se seleccionó el lenguaje de programación C para establecer comunicación con la tarjeta procesadora, dado que brinda el acceso a ambos niveles de programación, y además permite una comunicación muy eficiente, principalmente durante la transferencia de datos hacia la computadora anfitriona. En las siguientes secciones se describirá la forma en que se lleva a cabo esta comunicación.

### **2.12.1 Memorias intermedias de comunicación en la computadora anfitriona**

El controlador ACCEL es instalado automáticamente por un programa de configuración inicial al momento de inicializar la computadora anfitriona. Dicho programa también define dos memorias intermedias de comunicación en la memoria de la computadora anfitriona, y le asigna un número a cada una. La primera es la memoria intermedia de comunicación 0, se utiliza para establecer comunicación en formato texto y para transferir instrucciones a la tarjeta procesadora. La segunda es la memoria intermedia de comunicación 1, se utiliza para establecer comunicación en formato binario y principalmente para transferir datos a alta velocidad. La Fig. 2.3 representa el papel desempeñado por el controlador ACCEL al coordinar la comunicación entre la tarjeta procesadora y la computadora anfitriona. Una descripción detallada de este proceso se describe ampliamente en [21,23].

### **2.12.2 Dispositivos ACCEL**

El controlador ACCEL define dos tipos de dispositivos: el dispositivo ACCEL numerado y el dispositivo ACCEL dirigido. A continuación se presentan las características del primero, debido a que fue el que se utilizó en el presente trabajo; información del segundo se presenta en [23].

El usuario puede definir varios dispositivos ACCEL numerados y cada uno puede comunicarse con una memoria intermedia de comunicación específica de la computadora anfitriona. Para lograr esto es necesario que un programa abra uno de estos dispositivos y lo relacione con la memoria intermedia de comunicación deseada. El nombre de un dispositivo ACCEL numerado está integrado por el nombre ACCEL seguido por el número de la memoria intermedia de comunicación (generalmente cero o uno).

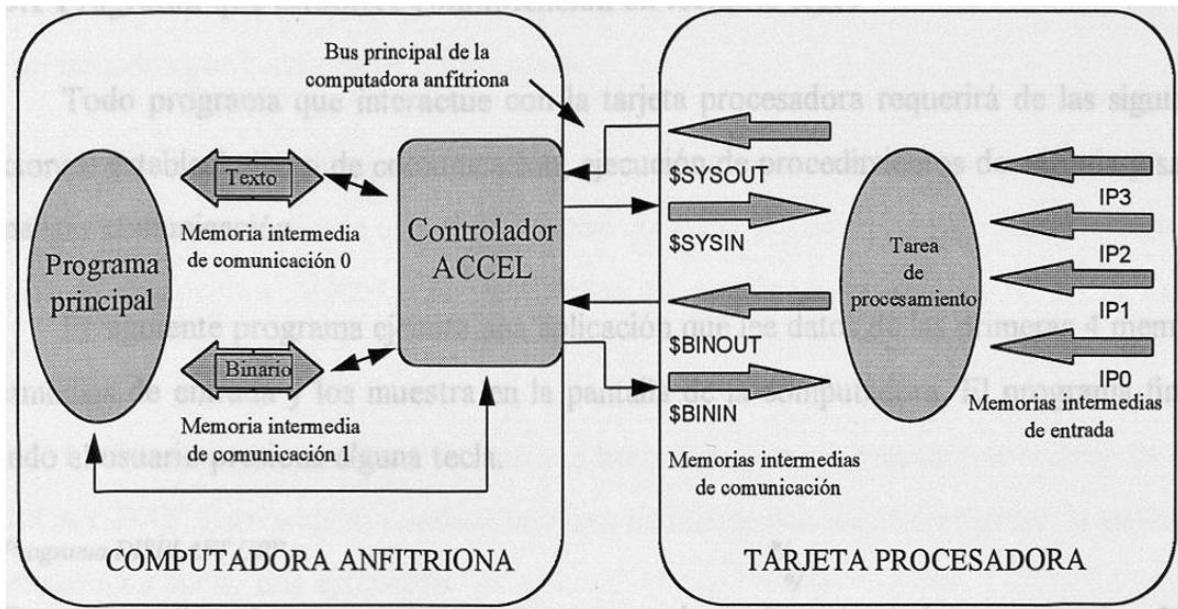


Fig. 2.3 Diagrama que representa el papel desempeñado por el controlador ACCEL al coordinar la comunicación entre la computadora anfitriona y la tarjeta procesadora.

A continuación se presenta y describe la operación de las funciones y subrutinas de programación más importantes para el control y configuración de la tarjeta procesadora. Dichas funciones son proporcionadas por el fabricante de la tarjeta y corresponden al lenguaje de programación "C y C++".

### 2.13 INTERACCIÓN CON LA TARJETA PROCESADORA MEDIANTE EL LENGUAJE DE PROGRAMACIÓN C

La tarjeta procesadora puede controlarse totalmente mediante el lenguaje de programación C. En esta sección se describe la forma de crear programas utilizando este lenguaje, explotando las primitivas proporcionadas por el fabricante de la tarjeta. Éstas están integradas por funciones que simplifican la escritura de los programas y permiten principalmente: establecer y finalizar la comunicación con la tarjeta, enviar instrucciones de configuración, cargar archivos con aplicaciones, iniciar y detener dichas aplicaciones, transferir datos a alta velocidad (mediante DMA) y determinar el estado del sistema.

Los programas que se presentarán a continuación fueron compilados en el compilador Borland C++ 3.1 [26].

### 2.13.1 Programa que establece comunicación en formato texto

Todo programa que interactúe con la tarjeta procesadora requerirá de las siguientes secciones: establecimiento de comunicación, ejecución de procedimientos de entrada y salida, y finalizar comunicación.

El siguiente programa ejecuta una aplicación que lee datos de las primeras 4 memorias intermedias de entrada y los muestra en la pantalla de la computadora. El programa finaliza cuando el usuario presiona alguna tecla.

```

/* Programa DISPLAYT.CPP                                     */
/*                                                         */
/* Este programa ilustra la comunicación con la tarjeta procesadora mediante el envío de instrucciones de */
/* entrada y la lectura de dato en formato texto.          */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

extern "C" {
    #include "dapio.h"      /* Archivo de cabecera de la tarjeta procesadora */
    #include "dapio.c"     /* Archivos con el código fuente                */
    #include "cfgdap.c"
    #include "dapiocd.c"
    #include "c_lib.c"
}

void main ()
{
    int y1, y2, y3, y4;

    clrscr();
    NInitDap("ACCEL0");          /* Establece comunicación con la tarjeta */

    /* Configura la tarjeta procesadora mediante el envío del archivo de instrucciones DISPLAYT.DAP */
    printf("Configurando tarjeta procesadora\n");
    if (fConfigDap(DapOut,"DISPLAYT.DAP") >= 200)          /* Transfiere archivo de instrucciones */
    {
        printf("Error al configurar tarjeta\n");
        exit(2);
    }
    while (fGetDapAvail(DapIn) == 0);          /* Espera a que existan datos por transferir */
    printf("Transfiriendo datos hacia la computadora anfitriona\n");
    while(1) {
        fscanf (DapIn,"%d %d %d %d",&y1,&y2,&y3,&y4);          /* Lee datos de la tarjeta */
        printf ("%5d %5d %5d %5d\n", y1, y2,y3,y4);
        if(kbhit()) break;
    }
    LeaveDap();          /* Finaliza comunicación con la tarjeta procesadora */
}

```



Las directivas *include* al inicio del programa incluyen los archivos de cabecera proporcionados por el fabricante de la tarjeta.

La función *"NInitDap()"* establece la comunicación, abre dos archivos llamados *DapIn* y *DapOut* y los asocia con el dispositivo ACCEL. En este ejemplo, ambos archivos se asignan al dispositivo ACCEL0, cuyo nombre es pasado como parámetro en la función *"NInitDap()"*.

La función *"fConfigDap()"* transfiere hacia la tarjeta procesadora el archivo de texto DISPLAY.DAP. Este archivo contiene una lista de instrucciones que configuran la tarjeta y la instruyen para iniciar una aplicación.

La función *"fGetDapAvail()"* retorna la cantidad de *bytes* disponibles para ser transferidos por la tarjeta procesadora hacia la computadora anfitriona. El lazo *while* que contiene esta función espera a que ésta devuelva un valor distinto de cero. Esto asegura que la tarjeta procesadora se encuentra enviando datos cuando el programa intenta leerlos, en caso contrario, el programa permanecerá en un estado de espera hasta que los datos sean transferidos.

El segundo lazo *while* se usa para leer continuamente datos de la tarjeta procesadora mediante la función *"fscanf()"*. Esta función extrae 4 datos del flujo apuntado por *DapIn*, cada dato es depositado en su respectiva variable y corresponden a los primeros cuatro valores de cada una de las memorias intermedias de entrada. La función *"printf()"* despliega estos datos en pantalla. El lazo *while* finaliza cuando el usuario presiona alguna tecla.

Finalmente, la función *"LeaveDap()"* cierra los archivos *DapIn* y *DapOut* y da por concluida la comunicación con la tarjeta procesadora.

### 2.13.2 Transferencia de bloques binarios para su registro en disco

El siguiente programa aumenta las capacidades del programa DISPLAYT.CPP. Por una parte, el programa realiza una comunicación más eficiente con la tarjeta procesadora al

realizar operaciones de transferencia de datos en bloques binarios empleando los controladores DMA; y por otra, muestra cómo ejecutar una operación de registro de datos a alta velocidad. Además, se muestra cómo variar parámetros en las instrucciones de configuración de la tarjeta procesadora. El programa solicita, por parte del usuario, el nombre del archivo donde se almacenarán los datos que serán registrados, posteriormente abre dicho archivo para escritura binaria y registra los datos enviados por la tarjeta procesadora. En este programa, la comunicación con la tarjeta procesadora inicia y finaliza sin el uso de las funciones “NinitDap()” y “LeaveDap()”; y además, los datos enviados por la tarjeta procesadora son leídos por la computadora anfitriona en formato binario mediante el dispositivo ACCEL1 y utilizando la memoria intermedia de comunicación 1. El código C para este ejemplo se presenta a continuación:

```

/* Programa REGDISCO.CPP                                     */

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

extern "C" {
    #include "dapio.h"
    #include "dapio.c"
    #include "dapiocd.c"
    #include "cfgdap.c"
}
#define BlockSize 128 /* Tamaño del bloque de datos que será transferido */

void main ()
{
    int Data[BlockSize], i;
    FILE *DapTextOut,
        *DapTextIn,
        *DapBinIn,
        *Arch_Reg;
    char Nombre_de_Archivo[20];

    clrscr();
    printf ("Introduce el nombre del archivo : ");
    scanf ("%19s",Nombre_de_Archivo);
    if((Arch_Reg = fopen (Nombre_de_Archivo,"wb"))==NULL) { /* Apertura del archivo binario */
        clrscr();
        printf("Error al abrir el archivo %s",Nombre_de_Archivo);
        exit(1);
    }
    /* Abre el dispositivo ACCEL1 para recibir datos binarios de la tarjeta procesadora, y ACCEL0 para */
    /* establecer comunicación con la tarjeta procesadora en formato texto */
    if (((DapBinIn = fopen("ACCEL1","rb")) == NULL) ||
        ((DapTextOut = fopen("ACCEL0","wt")) == NULL) ||
        ((DapTextIn = fopen("ACCEL0","rb")) == NULL)) {

```

```

        clrscr();
        printf("Error al abrir el dispositivo ACCEL\n");
        exit(1);
    }
    /* Establece, para todos los archivos, el estado binario del DOS */
    fBinaryMode(DapBinIn);
    fBinaryMode(DapTextOut);
    fBinaryMode(DapTextIn);
    setvbuf(DapTextOut, NULL, _IONBF, 0); /* Esta función permite que las instrucciones sean enviadas */
                                        /* directamente a la tarjeta procesadora. */
    fWriteln(DapBinIn, "S,M00"); /* Establece el estado correcto del controlador ACCEL */
    fprintf(DapTextOut, "RESET\n"); /* Envía una instrucción RESET a la tarjeta procesadora */
    /* Elimina datos basura de las memorias intermedias de la tarjeta procesadora */
    fflush(DapBinIn);
    fflush(DapTextIn);
    /* Esta función le asigna al parámetro %1 en el archivo de instrucciones APLICFFT.DAP el valor de 520.8 */
    SetParam(1, "520.8");
    /* Configura la tarjeta procesadora mediante el envío del archivo de instrucciones APLICFFT.DAP */
    if (fConfigDap(DapTextOut, "APLICFFT.DAP") >= 200) {
        clrscr();
        printf("Error al configurar tarjeta procesadora\n");
        exit(2);
    }
    while (fGetDapAvail(DapBinIn) == 0) /* Espera a que existan datos por transferir */
        printf("Registrando datos");
    for (i=1; i<=100; i++) { /* Lee y registra 100 bloques de datos */
        gotoxy(20,2); printf("%d",i);
        fGetDapBuf(Data, sizeof(int), BlockSize, DapBinIn);
        if((fwrite(Data, sizeof Data, 1, Arch_Reg)) != 1) {
            clrscr();
            printf("Error de escritura en el archivo %s", Nombre_de_Archivo);
            exit(2);
        }
    }
    printf("\n");
    fprintf(DapTextOut, "STOP\n"); /* Envía una instrucción STOP a la tarjeta */
    fWriteln(DapBinIn, "R"); /* Restaura el estado del controlador ACCEL */
    /* Cierra todos los archivos */
    fclose(Arch_Reg);
    fclose(DapTextOut);
    fclose(DapTextIn);
    fclose(DapBinIn);
}

```

El dispositivo ACCEL0 se abre como un archivo de entrada y se asigna al flujo *DapTextIn*. A su vez se abre como un archivo de salida y se asigna al flujo *DapTextOut*. El dispositivo ACCEL1 se abre como un archivo de entrada y se asigna al flujo *DapBinIn*, este dispositivo se usa para transferir datos binarios a la computadora anfitriona.

Todos los archivos se establecen al estado binario del DOS con la función "*fBinaryMode()*". Esto permite que los datos manejados por los flujos *DapTextIn*,

*DapTextOut* y *DapBinIn*, sean almacenados temporalmente (por el sistema operativo DOS) en formato binario, evitando la conversión al código ASCII [23].

La llamada a la función “*setvbuf()*” permite que las instrucciones sean enviadas directamente a la tarjeta, evitando el almacenamiento temporal en la memoria intermedia manejada por el flujo *DapTextOut*.

La primer llamada a la función “*fWriteIoCtlStr()*” establece el estado adecuado del controlador ACCEL [23].

Enseguida, la función “*fprintf()*” envía una instrucción RESET. La llamada a la función “*fFlushDap()*” elimina los datos presentes en las memorias intermedias de entrada de la computadora anfitriona. Esto descarta el uso de los datos depositados en estas memorias por aplicaciones anteriores.

La función “*fConfigDap()*” envía a la tarjeta procesadora el archivo de instrucciones APLICFFT.DAP. Por otra parte, la función “*fGetDapBuf()*” se utiliza para leer bloques de datos de la tarjeta procesadora. Estos bloques son transferidos hacia la computadora anfitriona mediante los controladores DMA, con lo cual aumenta la eficiencia de la transferencia. El primer parámetro especifica el *buffer* donde se colocarán temporalmente los datos que son transferidos de la tarjeta. El segundo parámetro especifica el tamaño en *bytes* de cada dato que será transferido. El tercer parámetro indica la cantidad de datos a transferir. El último parámetro es un puntero a archivo que se asocia con el flujo de entrada proveniente de la tarjeta.

Este ejemplo emplea la capacidad de la función “*fConfigDap()*” para reemplazar parámetros en las instrucciones que se envían a la tarjeta procesadora. La función “*SetParam()*” relaciona el parámetro 1 con la cadena “520.8”. De acuerdo a esta relación, la función “*fConfigDap()*”, reemplaza el símbolo %1 en el archivo de instrucciones APLICFFT.DAP con la cadena asociada al parámetro 1. El contenido del archivo APLICFFT.DAP es el siguiente:

```

;%DEFAULT %1=1041.6
RESET
IDEFINE A 1
    SET IPIPE0 S0
    TIME %1
END
PDEFINE B
    FFT(5,8,0,IPO,$BINOUT)
END
START A,B

```

La primer línea especifica el valor implícito que tomará el símbolo %1 en el caso de que no se especifique ningún parámetro en la función *“SetParam()”*. Esta instrucción siempre debe aparecer en la primer línea. La capacidad de variar parámetros, permite en este ejemplo modificar el tiempo de muestreo en forma automática a partir del programa en C.

El ejemplo anterior presentó la forma de registrar datos binarios a alta velocidad, una vez que han sido registrados, es posible que otros programas tengan acceso a ellos ya sea localmente mediante otras rutinas o remotamente transfiriendo el archivo a otra computadora para su análisis. En el presente trabajo, este algoritmo se seleccionó como base para el desarrollo del programa principal del registrador, debido a su eficiencia para transferir datos de falla a la computadora anfitriona. En la siguiente sección se explica cómo programar tareas de procesamiento mediante el lenguaje de programación C apoyándose en las primitivas proporcionadas por el fabricante de la tarjeta [24].

## 2.14 PROGRAMACIÓN DE TAREAS DE PROCESAMIENTO

El SO de la tarjeta procesadora cuenta con una amplia variedad de tareas predefinidas. Estas tareas son programadas en lenguaje C o lenguaje ensamblador, aceptan una cierta cantidad de parámetros (variables, constantes o memorias intermedias), ejecutan un procesamiento y depositan los resultados en otras memorias intermedias o variables. Cuando alguna tarea requiere de un procedimiento más especializado o más eficiente, el SO permite que el usuario programe y agregue sus propias tareas de procesamiento. Las tareas son programadas en un compilador de lenguaje C genérico, una vez compilada puede cargarse en el SO de la tarjeta procesadora mediante programas especializados. Posteriormente pueden ser utilizadas como cualquier otra tarea predefinida.

Las tareas de procesamiento programadas por el usuario tienen las siguientes ventajas con respecto a las tareas predefinidas en el SO:

- Pueden implementar procesamiento en tiempo real.
- Se pueden combinar en una sola tarea operaciones de varias tareas de procesamiento predefinidas, mejorando su eficiencia.
- Una tarea puede leer cualquier cantidad de variables, constantes o memorias intermedias. De igual forma puede escribir sus resultados en cualquier cantidad de memorias intermedias o variables.
- Es posible realizar operaciones matemáticas en punto flotante empleando los operadores del lenguaje C.

Un programa que implementa una tarea de procesamiento usualmente consiste en una sección de inicialización integrada por una lista de parámetros que se pasarán a la tarea seguida de un lazo de procesamiento. Cuando la tarjeta procesadora recibe una instrucción START primeramente ejecuta el código de inicialización. Este código verifica que la instrucción que llamó a la tarea contenga parámetros válidos. Si este es el caso, la tarea iniciará el lazo de procesamiento, en caso contrario, el SO desplegará un mensaje de error. El lazo de procesamiento leerá datos procedentes de las distintas variables o memorias intermedias, procesará estos datos y escribirá los resultados en otras variables o memorias. Después de que la tarjeta procesadora ejecute el código de inicialización exitosamente, procesará datos indefinidamente hasta que se envíe una instrucción STOP, RESET o RESTART.

En el presente trabajo se emplearon las bibliotecas de funciones proporcionadas por el fabricante de la tarjeta. Estas fueron compiladas con el compilador Microsoft C/C++. Debido a esta razón, los programas que implementan las tareas de procesamiento necesitan ser escritos en cualquiera de las versiones 4, 5, 6 o 7 de dicho compilador. Si se usa otro compilador, se generarán errores al momento de compilar y enlazar el código del programa

con el código binario de las funciones de biblioteca del fabricante. La desventaja de no contar con el código fuente de las funciones que integran las bibliotecas, evita que se pueda utilizar cualquier otro compilador C. El programa que se presentará a continuación fue compilado utilizando el compilador Microsoft C/C++ versión 7.0 [27].

El siguiente ejemplo muestra los principales componentes de los que consta un programa que implementa una tarea de procesamiento. El programa lee datos de una memoria intermedia, rectifica los datos y los deposita en otra memoria intermedia.

```

/* Nombre del archivo: RECTIFIC.C                               */
/* Nombre de la tarea de procesamiento: RECTIFIC (p1, p2)      */
#include <cdapcc.h>

void rectific (PIPE *in_pipe, PIPE *out_pipe)
{
    long int val;
    open_pipe (in_pipe, P_READ);          /* Prepara una memoria intermedia para lectura */
    open_pipe (out_pipe, P_WRITE);        /* Prepara una memoria intermedia para escritura */
    while (1) {
        val = (int) get_pipe (in_pipe);   /* Lee un dato de la memoria intermedia asociada a in_pipe */
        if (val < 0) val = -val;          /* Procesamiento */
        put_pipe (out_pipe, val);         /* Escribe el resultado del procesamiento en la memoria intermedia */
    }                                     /* asociada a out_pipe */
}

void main (PIB **plib)
{
    void **argv;
    int argc;
    /* Verifica los parámetros que se pasan a la tarea */
    argv = param_process (plib, &argc, 2, 2; T_PIPE_W, T_PIPE_W);
    rectific ((PIPE *) argv[1], (PIPE *) argv[2]); /* Llama a la función que ejecutará el procesamiento */
}

```

En este programa, el archivo de cabecera CDAPCC.H define todas las constantes, macros, tipos de datos y prototipos requeridos para acceder a cada una de las funciones del sistema. Los identificadores PIB, T\_PIPE\_W, P\_READ, P\_WRITE, “*param\_process()*”, “*open\_pipe()*”, “*get\_pipe()*” y “*put\_pipe()*”, también se definen en el archivo CDAPCC.H.

La tarea de procesamiento inicia su ejecución en la función “*main()*”. El SO de la tarjeta procesadora le pasa a dicha función un puntero que contiene la dirección de inicio de cada uno de los parámetros de la tarea. La función “*param\_process()*” verifica que cada uno

de los parámetros sean del tipo correcto y devuelve un arreglo de punteros con la dirección de cada uno. El primer parámetro es *argv[1]* y el segundo *argv[2]*.

La tarea de procesamiento RECTIFIC acepta como parámetros dos memorias intermedias apuntadas por *argv[1]* y *argv[2]*. El identificador PIPE define una estructura de datos, la cual no es manipulada directamente, sino pasada a las funciones del sistema. La función "*main()*" convierte los dos argumentos de RECTIFIC en datos de tipo PIPE y los pasa a una función auxiliar llamada "*rectific()*". Esta función prepara las memorias intermedias *in\_pipe* y *out\_pipe* para lectura y escritura, respectivamente. El lazo *while* ejecuta el procesamiento de datos. La función "*get\_pipe()*" elimina el siguiente valor de la memoria intermedia de entrada apuntada por *in\_pipe* y lo coloca en la variable *val*. La instrucción *if* convierte cada valor negativo a positivo. La función "*put\_pipe()*" envía el dato procesado a la memoria intermedia de salida apuntada por *out\_pipe*.

Para usar la instrucción RECTIFIC, es necesario compilarla, enlazarla, convertirla a un código binario y cargarla en el SO de la tarjeta procesadora. Este proceso se explica exhaustivamente en [24].

El código de esta tarea se puede modificar fácilmente de tal forma que ejecute diferentes funciones de procesamiento. Simplemente hay que reemplazar la instrucción *if* con el código C requerido.

## 2.15 CONCLUSIONES

- Se presentó las características y funcionamiento del *hardware* y *software* de la tarjeta procesadora. El primero posee los componentes principales de una microcomputadora mientras que el segundo está integrado por un sistema operativo (SO) multitarea de tiempo real, el cual cuenta con una amplia variedad de programas y herramientas que simplifican la realización de aplicaciones.
- Se describen las diferentes clases de memorias intermedias e instrucciones del SO. Las primeras permiten la comunicación entre tareas de procesamiento mientras que las



segundas controlan el *hardware* de muestreo, definen estructuras de datos, implementan y sincronizan diferentes tareas y proporcionan información de estado.

- Se describen las consideraciones para aumentar la eficiencia de la tarjeta procesadora y la manera de evitar el desbordamiento de las memorias intermedias.
- Se presentan las diferentes formas de controlar y configurar a la tarjeta procesadora mediante programas escritos en el lenguaje de programación C. Mostrando para ello el código fuente documentado de diferentes aplicaciones. Además, se ilustra la forma de programar tareas de procesamiento empleando este mismo lenguaje.
- Se mostraron diferentes alternativas para la comunicación y transferencia de datos entre la computadora anfitriona y la tarjeta procesadora.
- Las memorias intermedias de comunicación \$BININ y \$BINOUT son las más eficientes para establecer comunicación y transferir datos binarios hacia la computadora anfitriona o viceversa, ya que evitan que los datos sean convertidos a formato texto, y además, toman ventaja de los controladores DMA.
- Para aumentar la eficiencia de una tarea de procesamiento, se recomienda procesar datos en la tarjeta procesadora, en vez de transferirlos hacia la computadora anfitriona para su procesamiento; así mismo para aumentar la eficiencia durante la transferencia de datos hacia la computadora anfitriona, es recomendable transferir bloques de datos utilizando el formato binario en vez del formato ASCII.
- Para evitar el desbordamiento de las memorias intermedias es necesario, reducir la frecuencia de muestreo, cambiar la configuración de la tarjeta procesadora a una forma más eficiente o programar tareas de procesamiento personalizadas utilizando el lenguaje C.
- Una de las ventajas del paralelismo es la facilidad para la depuración de los programas. La sección de instrucciones que configuran la tarjeta puede depurarse con programas

especializados proporcionados por el fabricante. Mientras que el código C puede depurarse con un compilador de C genérico.

- Las tareas de procesamiento programadas por el usuario permiten implementar procesamiento en tiempo real, son más eficientes que las tareas predefinidas, pueden ejecutar operaciones en punto flotante, son flexibles y se adaptan a las características particulares de una aplicación.

## **NUEVA FAMILIA DE VENTANAS PARA ESTIMACIÓN ESPECTRAL Y DISEÑO DE FILTROS DIGITALES**

### **3.1 INTRODUCCIÓN**

El objetivo principal de este capítulo es presentar el desarrollo matemático de una nueva familia de ventanas, junto con sus características temporales y espectrales, así como sus aplicaciones en estimación espectral y filtrado digital.

La nueva familia de ventanas se obtuvo como resultado de una investigación efectuada con el objeto de mejorar las propiedades del filtrado convencional de Fourier en la función de localización y medición de fallas en relevadores digitales [6,7]. El criterio de diseño de dicho estudio consistió en reducir las fugas que los filtros convencionales tienen en las bandas interarmónicas, definiendo con ese propósito una hermética banda de paro. De lo anterior se obtuvo una nueva familia de ventanas con características muy atractivas para las aplicaciones de estimación espectral ya que poseen lóbulos laterales inferiores a los de las ventanas clásicas.

La selección de una función de ventana es un factor muy importante para el análisis espectral. Al aplicar una función ventana a una señal en el tiempo se convoluciona en frecuencia su espectro ideal con la transformada de Fourier de la función de ventana [1,14]. Dicha convolución distorsiona el espectro ideal de la señal analizada. Por una parte, el lóbulo principal de la ventana reduce la resolución espectral, ensanchando las componentes armónicas; y por otra, los lóbulos laterales interfieren en la estimación de la amplitud espectral en frecuencias vecinas.

La nueva familia de ventanas se obtiene al aplicar la transformada de Fourier inversa a la siguiente secuencia de funciones:

$$O_N(f) = \begin{cases} \cos^N\left(\frac{\pi}{2f_0}f\right), & |f| \leq f_0 \\ 0, & |f| > f_0 \end{cases} \quad (3.1)$$

donde  $N$  es un entero no negativo, llamado factor leptógeno ya que adelgaza el ancho del lóbulo principal, acotado por  $2f_0$ . Dicha secuencia carece totalmente de lóbulos laterales. Sin embargo, las funciones temporales correspondientes son de duración infinita. El criterio de diseño de una ventana consiste pues en escoger la ventana más breve de tal manera que los lóbulos laterales (fenómeno de Gibbs) que aparecen al truncar la función sean despreciables.

La presentación de las nuevas ventanas se encuentra en [5], en [4] se ilustran aplicaciones de análisis espectral y estimación de señales aleatorias estacionarias, mientras que otras características y propiedades interesantes se estudian en [46].

En la primera parte de este capítulo se presenta la deducción matemática de la nueva familia de ventanas a partir de la transformada de Fourier inversa de la Ec. (3.1). Una diversidad de ventanas puede ser obtenida a partir de esta familia simplemente variando el factor leptógeno y su duración. En seguida, se considera su aproximación a una curva gaussiana cuando  $N$  tiende a un valor infinito. Posteriormente, se realizan aplicaciones de procesamiento de señales periódicas, esto con la finalidad de mostrar la potencialidad de la nueva familia de ventanas como una alternativa para el posterior procesamiento de los datos obtenidos por el registrador (análisis armónico, análisis espectral (espectrogramas), medición fasorial, variación dinámica de la frecuencia, etc.). Para ello se selecciona una ventana de entre todo el conjunto de funciones, aquí se presentan sus características temporales y frecuenciales, comparándolas con algunas ventanas clásicas: rectangular, Hamming, Hanning y Blackman. Dicho ejemplo ilustra la interferencia de ruidos con contenido espectral que aparece con las ventanas clásicas. Un estudio comparativo muestra los resultados obtenidos al analizar dos tonos puros con una pequeña separación frecuencial. En seguida se considera su aplicación en

espectrogramas. Finalmente se presentan las características de dichas ventanas considerándolas ahora como respuestas impulsionales en el diseño de filtros digitales.

### 3.2 OBTENCIÓN DE LA NUEVA VENTANA

El propósito de esta sección es obtener la familia de ventanas temporales correspondientes a la secuencia de funciones frecuenciales definidas en la Ec. (3.1). Esta ecuación se puede escribir brevemente como:

$$O_N(f) = V(f) \cos^N\left(\frac{\pi}{2f_0} f\right), \quad (3.2)$$

donde  $N$  es un entero no negativo y  $V(f)$  es la respuesta a la frecuencia del filtro pasa bajos ideal de ancho de banda  $f_0$  (ver Fig. 3.1). Esto es:

$$V(f) = \begin{cases} 1, & |f| \leq f_0 \\ 0, & |f| > f_0 \end{cases}. \quad (3.3)$$

La transformada inversa de Fourier de la Ec. (3.2) es:

$$o_N(t) = \int_{-\infty}^{\infty} O_N(f) \exp(2\pi f t) df. \quad (3.4)$$

Para  $N=0$ ,  $O_M(f) = V(f)$  y  $o_M(t)$  es la función seno cardinal de amplitud  $2f_0$  (ver Fig. 3.2):

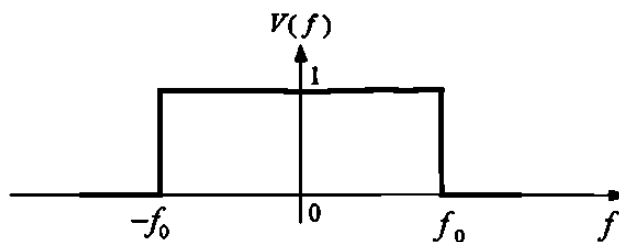


Fig. 3.1 Respuesta a la frecuencia del filtro pasa bajos ideal de ancho de banda  $f_0$ .

$$o_0(t) = 2f_0 \operatorname{sen} c(2\pi f_0 t) = v(t), \quad (3.5)$$

el ancho del lóbulo principal de esta función es  $T=1/f_0$  y sus cruces por cero están localizados en  $t_k=k(T/2)$ , donde  $k$  es un entero no nulo.

Usando la formula de Euler, y el teorema Binomial en la Ec. (3.2), se obtiene:

$$O_N(f) = \frac{1}{2^N} \sum_{k=0}^N \binom{N}{k} \exp\left[j \frac{\pi f}{2f_0} (2k - N)\right] V(f) \quad (3.6)$$

La transformada inversa de Fourier del  $k$ -ésimo termino de dicha expansión binomial es:

$$\int_{-\infty}^{\infty} \exp\left[j\pi f \left(\frac{(2k - N)}{2f_0} + 2t\right)\right] df = \delta\left(t - \frac{[2k - N]}{4f_0}\right), \quad (3.7)$$

donde  $\delta(t)$  es la Distribución de Dirac en  $t=0$ . Aplicando la linealidad de la transformada de Fourier y el teorema de convolución a la Ec. (3.6), se obtiene:

$$o_N(t) = \left[ \frac{1}{2^N} \sum_{k=0}^N \binom{N}{k} \delta\left(t - \frac{(2k - N)}{4f_0}\right) \right] * v(t) \quad (3.8)$$

donde  $*$  es el operador de convolución.

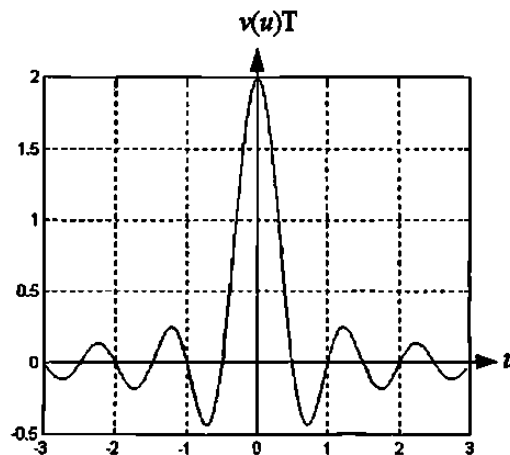


Fig. 3.2 Función seno cardinal de amplitud  $2f_0$ ,  $To_0(u) = 2\operatorname{sen}c(2\pi u) = Tv(u)$  y  $u=t/T$ .

El miembro izquierdo de la Ec. (3.8) corresponde a una distribución de Bernoulli con  $N+1$  pulsos instantáneos separados  $T/2$  s, y es la transformada inversa de Fourier del factor cosenoidal de la Ec. (3.2). Dada la Ec. (3.8), la función  $o_M(t)$  corresponde a la salida de un convertidor digital-análogo ideal cuando la distribución de Bernoulli de orden  $N$  pasa a través de él, ya que dicho convertidor ideal es un filtro pasa bajos ideal de ancho de banda  $f_0$ . La salida continua es entonces una función formada por la interpolación perfecta de la distribución binomial. El resultado es el siguiente:

$$o_N(t) = \frac{1}{2^N} \sum_{k=0}^N \binom{N}{k} v(t - t_k). \quad (3.9)$$

La función  $o_M(t)$  se forma por la superposición de las funciones seno cardinal moduladas por el  $k$ -ésimo coeficiente de Bernoulli, y centradas en los instantes

$$t_k = \frac{(2k - N)}{4} T, \quad k = 0, 1, 2, \dots, N. \quad (3.10)$$

Las Figs. 3.3 y 3.4 muestran las funciones  $O_M(f)$  y  $o_M(t)$  respectivamente para  $N=0,2,4,8$ . La Fig. 3.3 muestra la evolución del lóbulo de las funciones  $O_M(f)$  cuando  $N$  crece. Cuando  $N=0$ , se tiene el filtro pasa bajos ideal; para  $N=1$ , el primer lóbulo de una cosenoidal; y cuando  $N$  va aumentando, el lóbulo se va adelgazando hasta formar curvas de Gauss centradas en la frecuencia cero. La Fig. 3.4 muestra la evolución de las funciones  $o_M(t)$  en tiempo normalizado. Se observa que para  $N=0$ ,  $o_M(t)$  es la función seno cardinal. Esta función exhibe oscilaciones que se distribuyen sobre un largo intervalo de tiempo. Las oscilaciones persisten hasta  $N=6$ . Para  $N > 6$ , las funciones toman la forma de una curva gaussiana cada vez más aplanada.

En la siguiente sección se demuestra que las funciones  $o_M(t)$  tienden a una curva gaussiana cuando  $N$  aumenta.

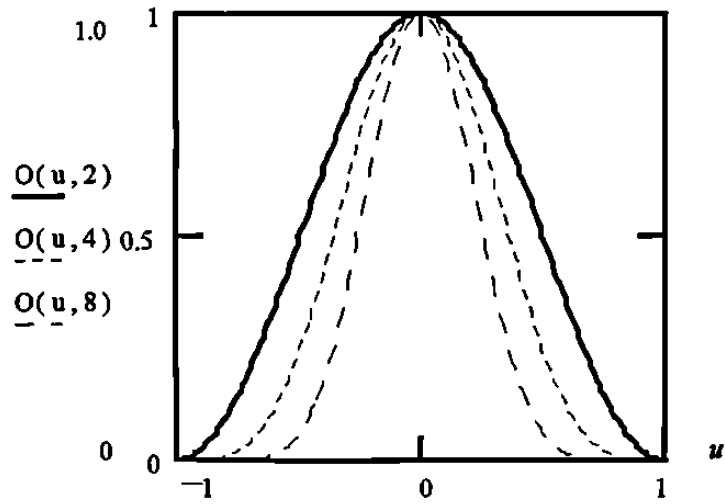


Fig. 3.3. Características frecuenciales de las funciones  $O_N(u)$ ,  $u=ff_0$ .

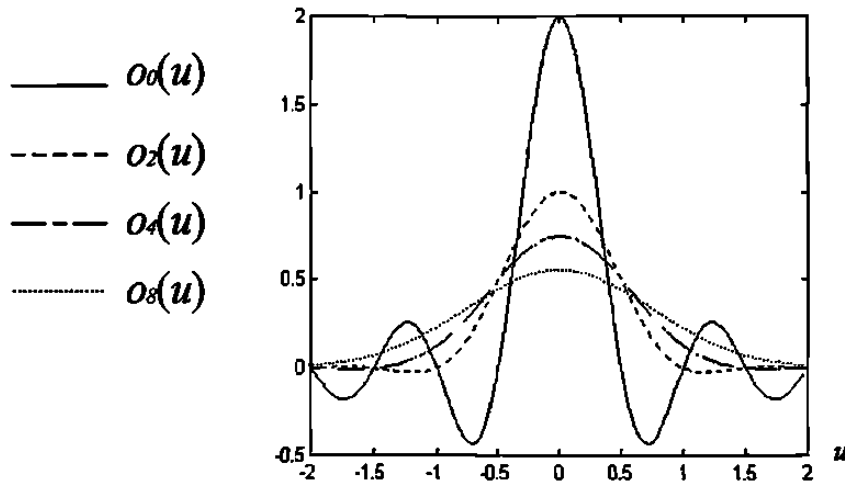


Fig. 3.4. Características temporales de las funciones  $o_N(u)$ , para una duración de cuatro ciclos,  $u=t/T$  y  $N=0,2,4,8$ .

### 3.3 APROXIMACIÓN A UNA CURVA GAUSSIANA

Aplicando el teorema de De-Moivre Laplace a la Ec. (3.8), es posible aproximar  $o_N(t)$  a una función gaussiana para grandes valores de  $N$ . Dicho teorema usa la formula de Stirling [8] para aproximar mediante una función exponencial el  $k$ -ésimo coeficiente de la expansión binomial. Si los términos del binomio son  $p$  y  $q$ , se tiene:



$$\binom{N}{k} p^k q^{N-k} \approx \frac{1}{\sqrt{2\pi Npq}} \exp\left[-\frac{(k-Np)^2}{2Npq}\right], \quad (3.11)$$

para  $\sigma = (Npq)^{1/2} \gg 1$ , y  $k=Np \pm \sigma$ . Para  $p=q=1/2$ , la aproximación a la función en la Ec. (3.8) es:

$$g_N(t) = v(t) * \sqrt{\frac{2}{N\pi}} \sum_{k=0}^N \exp\left[-\frac{(k-N/2)^2}{N/2}\right] \delta\left(t - \frac{(2k-N)}{4f_0}\right). \quad (3.12)$$

La Ec. (3.12) es la convolución de  $v(t)$  con una distribución gaussiana discreta. Expresando esta ecuación en términos de los instantes  $t_k$  de la Ec. (3.10), se tiene

$$g_N(t) = v(t) * \sqrt{\frac{2}{N\pi}} \sum_{k=0}^N \exp\left[-\frac{(2f_0 t_k)^2}{N/2}\right] \delta(t - t_k). \quad (3.13)$$

La ecuación anterior es la interpolación perfecta de una distribución normal. Resolviendo dicha convolución, se obtiene fácilmente la función continua de  $g_N(t)$  como:

$$g_N(t) = 2f_0 \sqrt{\frac{2}{N\pi}} \exp\left[-\frac{(2f_0 t)^2}{N/2}\right] \quad (3.14)$$

Esta es una curva gaussiana con desviación estándar  $\sigma=(N/4)^{1/2}$ . Asumiendo que la curva se atenúa completamente en  $t_0 \geq 3\sigma/2f_0$ , el tiempo de atenuación en términos de  $N$  será

$$t_0 = \frac{3\sqrt{N}}{4} T, \quad (3.15)$$

donde  $T$  es la duración de un ciclo. Note que  $t_a$  es proporcional a la desviación estándar  $\sigma$  y ésta, a su vez, es proporcional a  $\sqrt{N}$ . Dado que  $\sigma$  interviene en la curva gaussiana como un factor de escalamiento lineal, cuando el orden de la función  $N$  aumenta, el ancho del lóbulo en frecuencia se reduce en proporción a  $\sqrt{N}$ .

Con la ayuda de las funciones  $g_N(t)$  se demuestra que la familia de funciones  $o_N(t)$  evoluciona desde una función seno cardinal, cuando  $N=0$ , hasta la curva gaussiana  $g_N(t)$  de la Ec. (3.14), cuando  $N$  es grande. En la Fig. 3.4 se observa que, a partir de  $N=4$ , la aproximación gaussiana es ya bastante buena.

### 3.4 DEFINICIÓN DE UNA VENTANA

Para definir una ventana a partir de la Ec. (3.9) es necesario especificar el factor leptógeno y una duración finita en el tiempo. Esta última puede obtenerse definiendo un intervalo de duración en función del periodo  $T$  como:

$$D = \delta T, \quad (3.16)$$

donde  $\delta$  es un escalar.

Al confinar la función de ventana en el dominio del tiempo se presenta el fenómeno de Gibbs en el dominio de la frecuencia, con la consecuente aparición de lóbulos laterales. Para reducir este efecto, la nueva ventana ( $\bar{o}_N(t)$ ) en su definición debe incluir la mayor parte de los valores significativos de la función ideal  $o_N(t)$ . Esto significa que sólo se deben de trincar valores no significativos de la función ideal. Sin embargo, debido a que algunas aplicaciones pueden ser más restrictivas en lo que respecta a su duración en el tiempo que en las características frecuenciales, la nueva ventana puede definirse de cualquier duración. En este sentido, su definición es similar a la definición de las ventanas de Káiser [1,9,10]. Las ventanas de Káiser se definen en función de dos parámetros: el parámetro de longitud (especificado por la cantidad de muestras) y el parámetro de forma. El primero, establece principalmente la anchura del lóbulo principal mientras que el segundo fija la amplitud relativa de los lóbulos laterales. Por su parte, en la definición de las nuevas ventanas, la forma de éstas no sólo depende del factor leptógeno sino también de su duración. La anchura del lóbulo principal se puede modificar variando el factor leptógeno  $N$ , sin perturbar dramáticamente la amplitud relativa de los lóbulos laterales (ver Fig. 3.5). Mientras que la duración  $D$  establece básicamente el nivel relativo de los lóbulos laterales (ver Fig. 3.6). Para duraciones cercanas al

limite de atenuación dada por la Ec. (3.15), la amplitud relativa del primer lóbulo lateral es pequeña. Sin embargo, si se consideran duraciones mayores, este nivel puede reducirse aún más.

Los coeficientes para la nueva ventana son obtenidos al evaluar la Ec. (3.9) en los siguientes instantes de tiempo discreto:

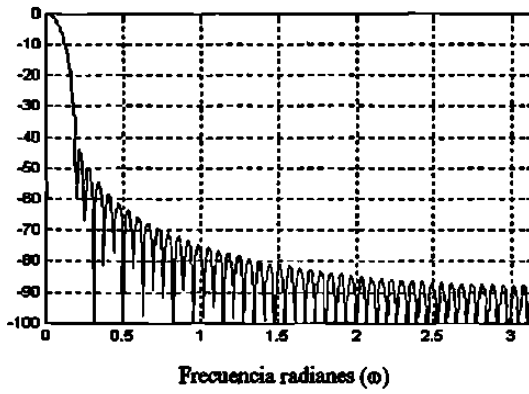
$$t_k = \left( \frac{k}{M} - \frac{\delta}{2} \right) T, \quad k = 0, 1, \dots, M-1, \quad (3.17)$$

donde  $M$  es la cantidad de muestras por ciclo.

En las Figs. 3.5, 3.6 y 3.7 se muestra el espectro de amplitud descrito por las funciones  $20\log_{10}|\bar{O}_N(e^{j\omega})|$  para las nuevas ventanas, donde  $\bar{O}_N(e^{j\omega})|_{\omega=2\pi k/M}$  es la transformada de Fourier discreta de la Ec. (3.9), al restringir su duración en el tiempo ( $D$ ) a un valor finito, para  $k=0, 1, 2, \dots, M-1$ . Para una mejor visualización se grafica sólo la mitad del espectro en forma continua en vez de puntos discretos. En la Fig. 3.5 se presentan los espectros para diferentes valores del factor leptógeno  $N$ , en estas gráficas se consideró  $M=32$  y  $D=3T$ . Observe que al incrementar el factor leptógeno disminuye la anchura del lóbulo principal y se modifica ligeramente la amplitud relativa de los lóbulos laterales. Para  $N=2, 4$  y  $6$  la amplitud relativa del primer lóbulo lateral permanece por debajo de los  $-40$  dB, elevándose a  $-35$  dB para  $N=8$ . En la Fig. 3.6 se muestran los espectros obtenidos, pero ahora variando el intervalo de duración  $D$ , para estas gráficas se tomó  $N=4$  y  $M=32$ . Observe que al ir incrementando el intervalo de duración  $D$  disminuyen notoriamente la amplitud relativa de los lóbulos laterales.

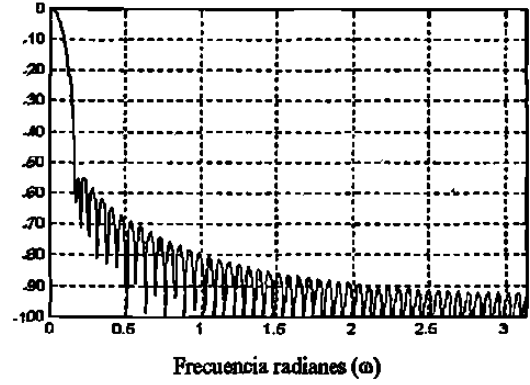
Después de que una nueva ventana ha sido especificada, se puede aplicar un escalamiento temporal para mejorar su resolución frecuencial. Esto se puede observar en la Fig. 3.7, la cual muestra las características espectrales de una ventana, al variar la cantidad de muestras  $M$ . Para dichas gráficas se mantuvo constante su duración  $D=3T$  y se tomó  $N=4$ . Observe cómo se mejora la resolución espectral sin modificar la amplitud relativa de los lóbulos laterales.

$$20\log_{10}|\overline{O}_2(e^{j\omega})|$$



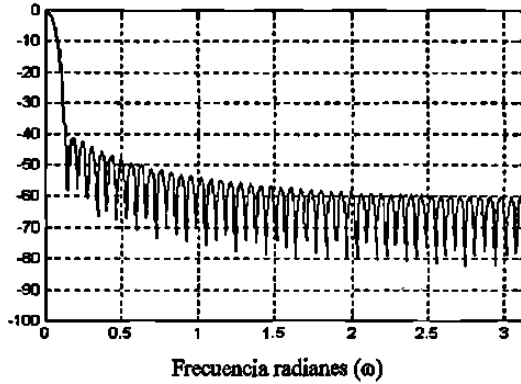
(a)

$$20\log_{10}|\overline{O}_4(e^{j\omega})|$$



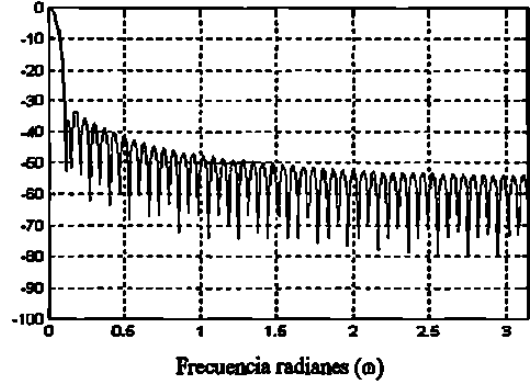
(b)

$$20\log_{10}|\overline{O}_6(e^{j\omega})|$$



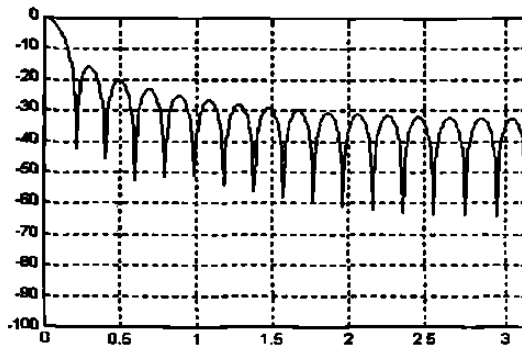
(c)

$$20\log_{10}|\overline{O}_8(e^{j\omega})|$$

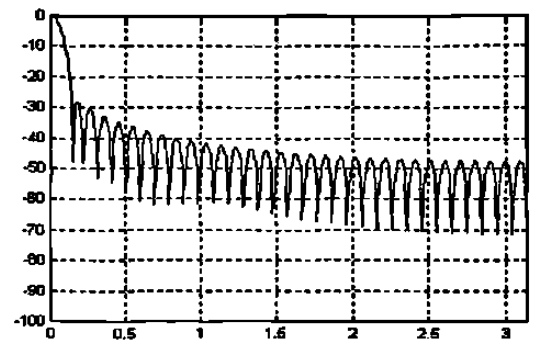


(d)

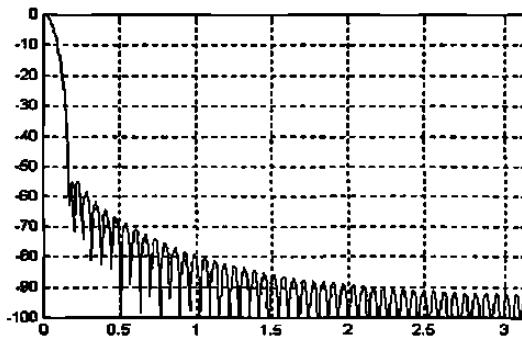
Fig. 3.5 Espectro de amplitud (en dB) de la nueva ventana para diferentes valores del factor leptógeno, manteniendo constante la duración  $D=3T$  con  $M=32$ . a)  $N=2$ . b)  $N=4$ . c)  $N=6$ . d)  $N=8$ .

$20\log_{10}|\overline{O}_4(e^{j\omega})|$ 
Frecuencia radianes ( $\omega$ )

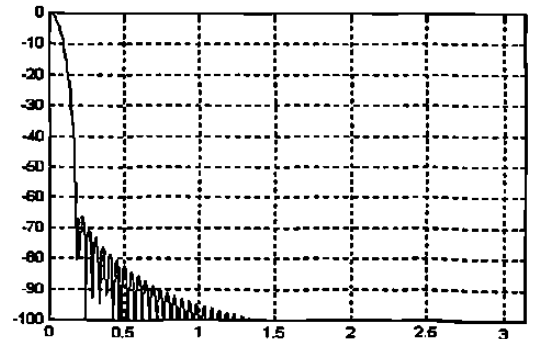
(a)

 $20\log_{10}|\overline{O}_4(e^{j\omega})|$ 
Frecuencia radianes ( $\omega$ )

(b)

 $20\log_{10}|\overline{O}_4(e^{j\omega})|$ 
Frecuencia radianes ( $\omega$ )

(c)

 $20\log_{10}|\overline{O}_4(e^{j\omega})|$ 
Frecuencia radianes ( $\omega$ )

(d)

Fig. 3.6 Espectro de amplitud (en dB) de la nueva ventana (considerando  $N=4$  en todos los casos) para diferentes valores del intervalo de duración  $D$  con  $M=32$ . a)  $D=T$ . b)  $D=2T$ . c)  $D=3T$ . d)  $D=4T$ .

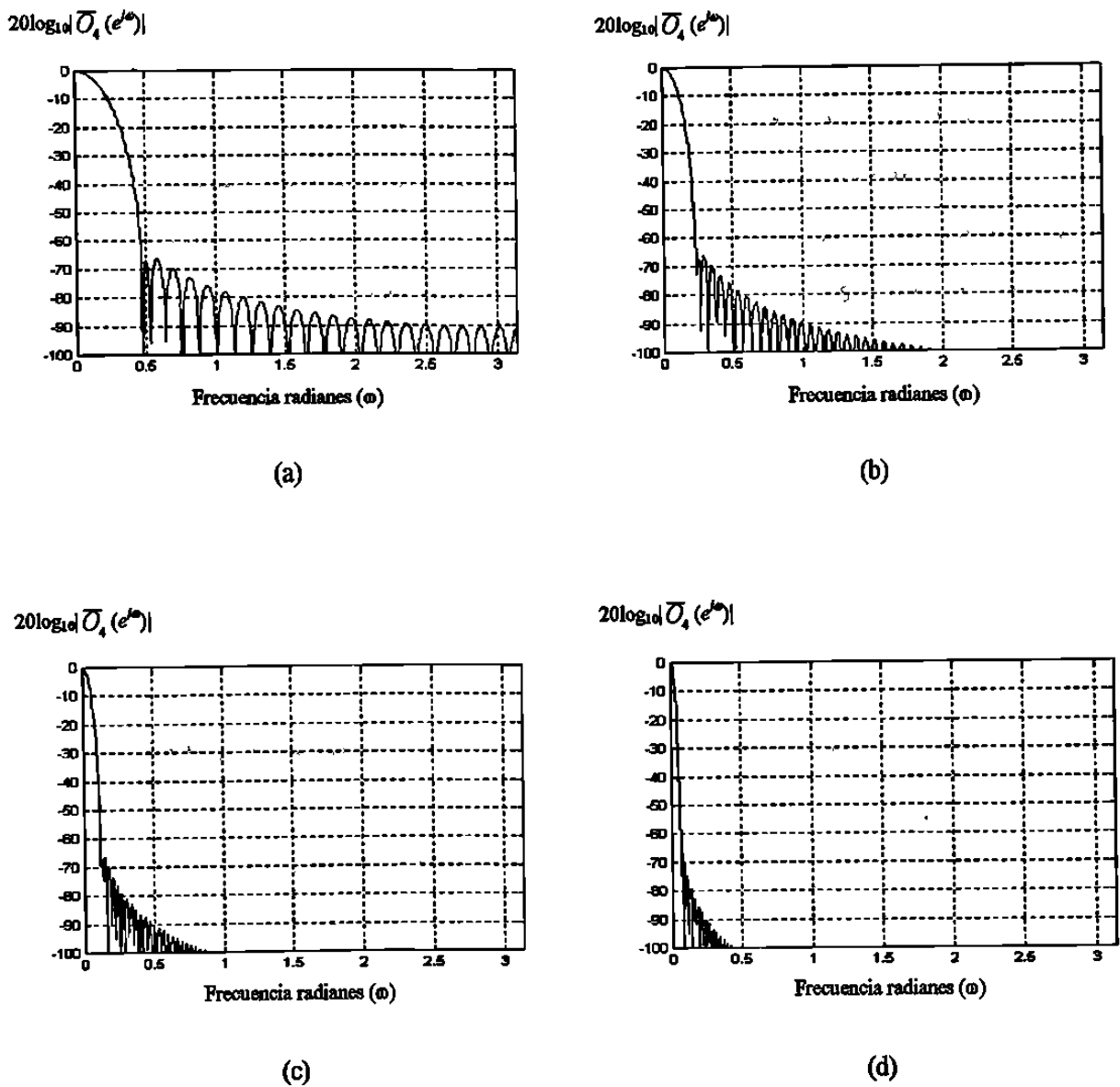


Fig. 3.7 Espectro de amplitud (en dB) de la nueva ventana para diferentes cantidades de muestras  $M$ , manteniendo constante la duración  $D=3T$ , con  $N=4$ . a)  $M=16$ . b)  $M=32$ . c)  $M=64$ . d)  $M=128$ .

Para el diseño de una ventana se puede emplear el intervalo de duración  $D$  para fijar la amplitud relativa de los lóbulos laterales y el factor leptógeno  $N$  junto con la cantidad de muestras  $M$  pueden definir la anchura deseada del lóbulo principal. De esta forma, se puede obtener una gran variedad de ventanas variando cualquiera de estos parámetros.

Como se mencionó en el Capítulo I, una de las principales finalidades del registrador, es la de recolectar datos para posteriormente explotar su contenido realizando los siguientes

estudios: medición fasorial, variación dinámica de la frecuencia, medición de potencias activas y reactivas, análisis espectral (espectrogramas), análisis armónico, identificación de los modelos dinámicos de oscilaciones electromecánicas y estudios de perturbaciones en generadores. En las siguientes secciones, se propone la ventana definida con  $N=4$  y  $D=3T$  como una herramienta de procesamiento digital para llevar a cabo dichos estudios. Para ello se comparan sus características espectrales con las ventanas clásicas y se considera su aplicación en análisis armónico y en espectrogramas. Como se observa en la Fig. 3.4, las oscilaciones de esta ventana son muy pequeñas (inferior al 1% de la amplitud máxima) para  $t \geq 1.5T$ , y por ser la más corta se seleccionó como la mejor ventana de dicha familia. Con una duración de 3 ciclos, es la que se afecta menos al truncarla, ya que presenta un nivel casi despreciable en sus lóbulos laterales, por lo que se puede considerar carente de ellos. Dicha ventana, será referida como ventana  $\bar{\sigma}_4$ .

### 3.5 COMPARACIÓN DE LA VENTANA PROPUESTA CON LAS VENTANAS CLÁSICAS

El truncamiento de las funciones es equivalente a aplicarles una ventana rectangular, lo cual se traduce en una convolución con un seno cardinal en el dominio de la frecuencia [1,9,10], generando pequeños lóbulos laterales. Una de las ventanas menos afectada por el truncamiento, para una duración de tres ciclos, es la ventana  $\bar{\sigma}_4$ . El nivel relativo de sus lóbulos laterales es inferior a los de las ventanas clásicas, como se podrá observar más adelante.

Algunas de las ventanas comúnmente usadas en los análisis espectrales y en el diseño de filtros digitales son la rectangular, Hamming, Hanning, Bartlett, Blackman y Káiser. Sus ecuaciones se encuentran en [1,9,10].

La Fig. 3.8 muestra el espectro de amplitud descrito por la función  $20\log_{10}|W(e^{j\omega})|$  para la ventana rectangular, Hanning, Hamming y para la ventana  $\bar{\sigma}_4$ , respectivamente.  $W(e^{j\omega})|_{\omega=2\pi n/L}$  es la transformada de Fourier discreta de cada una de estas ventanas para  $L=50$ ,

donde  $L$  es la cantidad de muestras, para  $k=0, 1, \dots, L-1$ . Para una mejor visualización se grafica solo la mitad del espectro en forma continua, en vez de puntos discretos.

La Tabla 3.1 muestra un resumen de las características de dichas ventanas. Tanto el ancho del lóbulo principal, como la amplitud relativa del primer lóbulo lateral se muestran en dicha tabla. Se han ordenado del lóbulo principal más delgado al más ancho; y en ese orden las amplitudes relativas de los lóbulos laterales descenden.

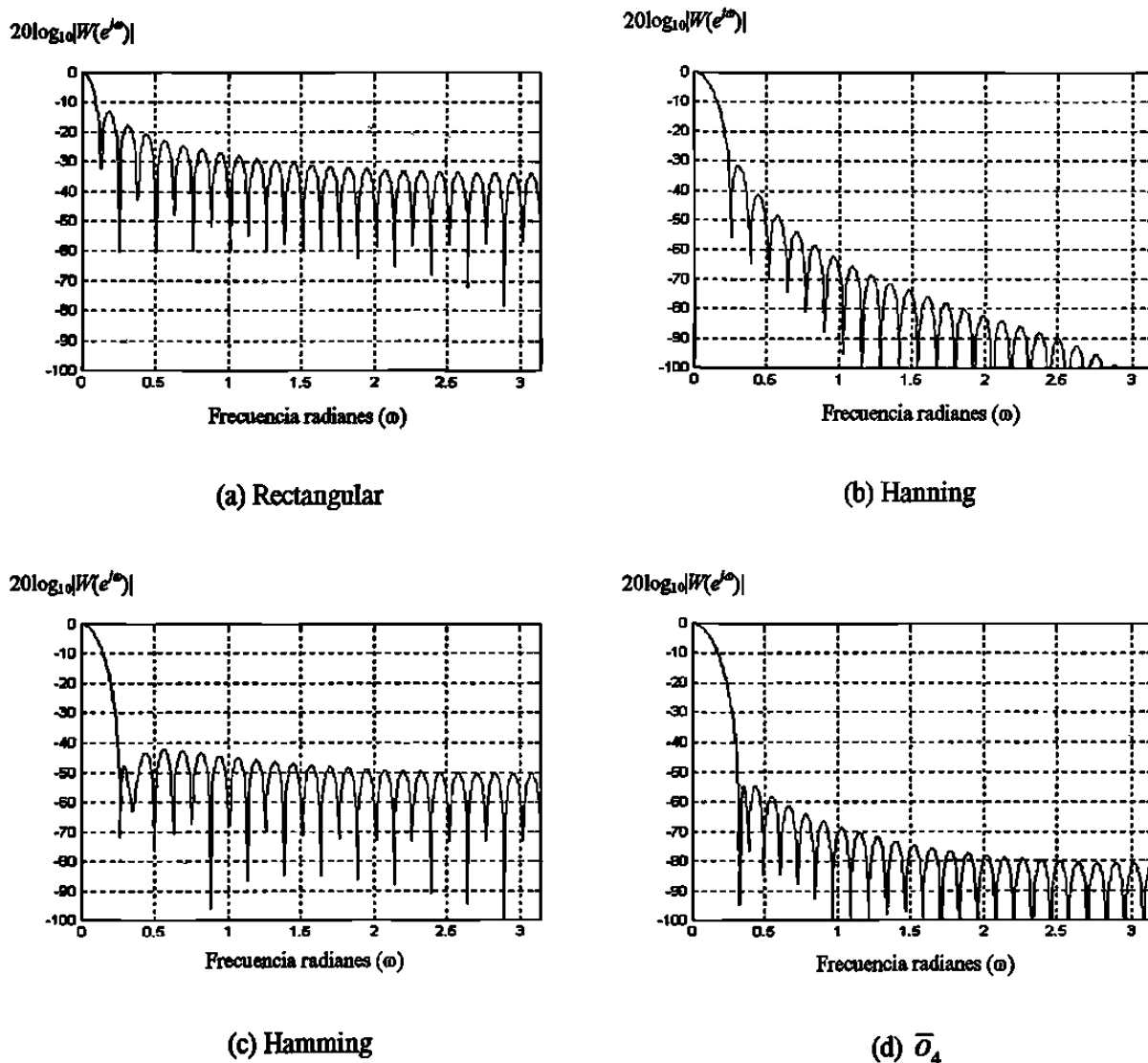


Fig. 3.8 Espectro de amplitud para diferentes ventanas.



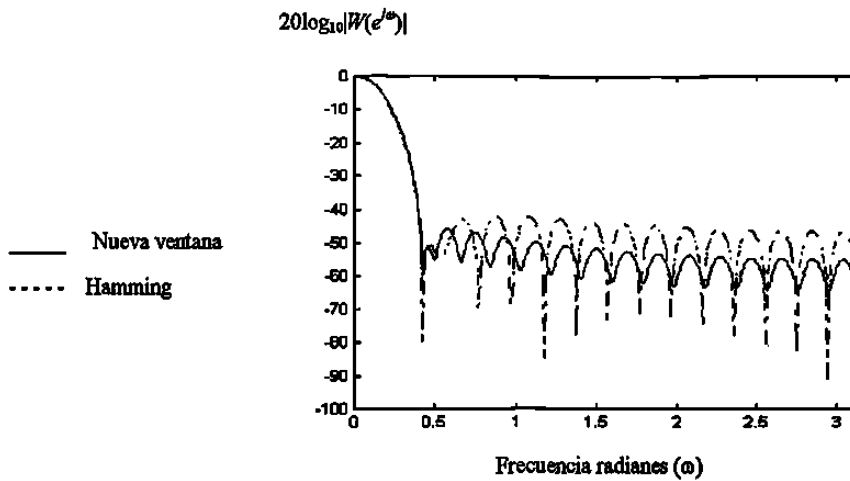
Tabla 3.1 Comparación de las ventanas

Tipo de Ventana	Anchura aproximada del lóbulo principal	Amplitud relativa del lóbulo lateral (dB)
Rectangular	$4\pi/(L+1)$	-13
Bartlett	$8\pi/L$	-25
Hanning	$8\pi/L$	-31
Hamming	$8\pi/L$	-41
$\bar{o}_4$	$10\pi/L$	-55
Blackman	$12\pi/L$	-57

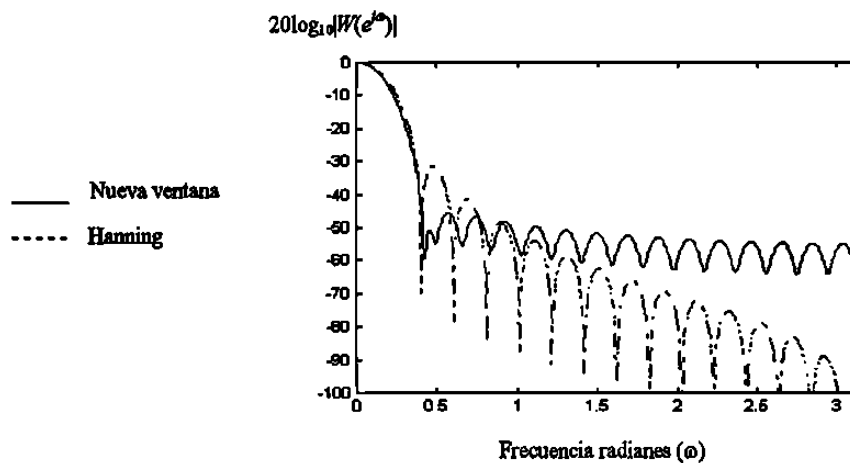
Un descripción detallada de las características de las ventanas clásicas empleadas en análisis espectrales se puede encontrar en [11]. Claramente se puede observar que la ventana rectangular es la que tiene el lóbulo principal más estrecho. Sin embargo, su primer lóbulo lateral se encuentra 13 dB por debajo del lóbulo principal.

La ventana  $\bar{o}_4$  reduce el nivel del primer lóbulo lateral hasta -55 dB, casi una milésima parte de la amplitud del lóbulo principal, superando las otras ventanas. Los lóbulos laterales pueden reducirse aún más si se definen ventanas de mayor duración. Por ejemplo, en la Fig. 3.6(d) para una duración de cuatro ciclos la amplitud relativa del lóbulo lateral disminuye hasta cerca de -70 dB y hasta -96 dB para una ventana con duración de ocho ciclos. Al comparar la ventana  $\bar{o}_4$  con la ventana de Hamming de la Fig. 3.8(c), se observa que los lóbulos de la primera van disminuyendo a lo largo de todo el espectro hasta -80 dB, mientras que los de la Hamming se mantienen en un nivel casi constante. Los lóbulos laterales de la ventana de Hanning en la Fig. 3.8(b) también disminuyen a lo largo del espectro de frecuencia, sin embargo, el nivel de los primeros lóbulos laterales está muy por arriba de los de la ventana  $\bar{o}_4$ .

Al igual que en la ventana de Káiser, es posible obtener una nueva ventana equivalente para cada una de las ventanas clásicas. Por ejemplo, para  $N=0$  se obtiene una ventana equivalente a la rectangular; para  $N=4$  y  $D=2.6T$  se obtiene una ventana equivalente a las de Hamming y Hanning, incluso mejorando la amplitud de los lóbulos laterales de dichas ventanas, esto se observa en la Fig. 3.9.



(a)



(b)

Fig. 3.9 Espectro de amplitud que muestra la nueva ventana equivalente para algunas ventanas clásicas tomando  $L=32$  muestras. La nueva ventana requirió  $N=4$  y una duración  $D=2.6T$  en ambas gráficas. a) Ventana equivalente a la de Hamming. b) Ventana equivalente a la de Hanning.

Al tener lóbulos laterales muy bajos, la nueva ventana elimina prácticamente la interferencia entre contenidos espectrales adyacentes. Dado que su lóbulo principal es amplio, se recomienda para el análisis espectral de señales periódicas o cuasi periódicas, donde la separación armónica es conocida, tal es el caso de las señales de voltaje y corriente eléctrica. En aplicaciones de filtrado es muy útil para el diseño de filtros digitales donde se desea el paso de una cierta banda de frecuencias y un fuerte bloqueo de las frecuencias adyacentes en segmentos cortos de señal de uno o dos ciclos. Por ejemplo en detección de portadoras de

sistemas de comunicación digital [12], y en general, en análisis armónico en la extracción de tonos puros [6,7] donde la banda de separación frecuencial es conocida *a priori*.

### 3.6 ANÁLISIS DE SEÑALES SENOIDALES

A continuación se analiza el efecto de la ventana  $\bar{o}_4$  sobre dos señales senoidales, con el objeto de ilustrar la interferencia entre contenidos espectrales adyacentes.

La transformada de Fourier de una señal cosenoidal  $A\cos(\Omega_0 t + \theta)$  es un par de impulsos en  $+\Omega_0$  y  $-\Omega_0$ . En el análisis de señales senoidales, una función de ventana, ensancha los impulsos del espectro ideal los cuales adquieren la forma del espectro de la ventana utilizada. Esto hace que las frecuencias exactas sean menos definidas, afectando por tanto la capacidad para identificar señales que tienen una separación frecuencial pequeña [1,9,10] debido al traslape frecuencial introducido por la ventana. Además, dicha interferencia entre contenidos espectrales vecinos perturba la estimación de amplitud en una cierta frecuencia.

Considere la suma de dos componentes cosenoidales continuas en el tiempo

$$c(t) = A_0 \cos(\Omega_0 t + \theta_0) + A_1 \cos(\Omega_1 t + \theta_1) \quad -\infty < t < \infty, \quad (3.18)$$

donde  $\Omega_0 = 2\pi f_0$  y  $\Omega_1 = 2\pi f_1$ .

Respetando el teorema de muestreo de Nyquist [1,9,10] y suponiendo que no existen errores de cuantificación [13], se obtiene la señal discreta en el tiempo

$$x[n] = A_0 \cos(\omega_0 n + \theta_0) + A_1 \cos(\omega_1 n + \theta_1) \quad -\infty < n < \infty, \quad (3.19)$$

donde  $\omega_0 = \Omega_0 \tau$ ,  $\omega_1 = \Omega_1 \tau$  y  $\tau$  es el periodo de muestreo.

Al aplicar una secuencia de ventana  $w[n]$  de longitud finita a la Ec. (3.19), se tiene:

$$v[n] = A_0 w[n] \cos(\omega_0 n + \theta_0) + A_1 w[n] \cos(\omega_1 n + \theta_1). \quad (3.20)$$

Usando la formula de Euler y aplicando la propiedad de escalamiento en la frecuencia, se obtiene la transformada discreta de Fourier de  $v[n]$  como

$$\begin{aligned}
 V(e^{j\omega}) = & \frac{A_0}{2} e^{j\theta_0} W(e^{j(\omega-\omega_0)}) + \frac{A_0}{2} e^{-j\theta_0} W(e^{j(\omega+\omega_0)}) \\
 & + \frac{A_1}{2} e^{j\theta_1} W(e^{j(\omega-\omega_1)}) + \frac{A_1}{2} e^{-j\theta_1} W(e^{j(\omega+\omega_1)})
 \end{aligned} \tag{3.21}$$

La ecuación anterior indica que la transformada de Fourier de la señal que fue ponderada por la función ventana superpone réplicas de la transformada de Fourier de la ventana sintonizadas en las frecuencias  $\pm\omega_0$  y  $\pm\omega_1$ .

En la Fig. 3.10(a) se muestra el uso de una ventana rectangular  $w[n]$  de longitud 64 para dos tonos definidos por  $f_0=45$  Hz y  $f_1=60$  Hz,  $A_0=0.7$  y  $A_1=1$  en la Ec. (3.21). Y muestreados a  $1/\tau=180$  Hz considerando  $\theta_0=\theta_1=0$ . Para ilustrar las características esenciales, se muestra sólo la magnitud de la transformada de Fourier discreta. En dicha figura se observa el traslape de contenidos adyacentes. Para los parámetros propuestos, la separación de las componentes individuales es clara, sin embargo sus amplitudes son modificadas por los lóbulos laterales de la armónica vecina. A medida que los dos tonos de frecuencia se aproximan en frecuencia, aumenta la interferencia de los lóbulos laterales de la componente vecina. La ventana rectangular tiene la ventaja de contar con el lóbulo principal más estrecho, sin embargo cuenta con el nivel más alto del primer lóbulo lateral. Estos indican la existencia de energía frecuencial en lugares donde no existen, mostrando un espectro engañoso. La Fig. 3.10(b) muestra el espectro de frecuencia del mismo caso utilizando la ventana de Hanning. Se observa que los lóbulos laterales disminuyen considerablemente aunque la interferencia entre contenidos espectrales vecinos persiste. La Fig. 3.10(c) muestra la ventana de Hamming, la cual presenta características muy semejantes comparada con la de Hanning, reduciendo aún más la interferencia, aunque todavía se alcanzan a percibir lóbulos laterales a lo largo de todo el espectro como un pequeño nivel de ruido. Finalmente, la Fig. 3.10(d) muestra el espectro obtenido con la ventana  $\bar{\sigma}_4$ . Ésta mejora las características espectrales de las otras ventanas en cuanto carece prácticamente de lóbulos laterales.

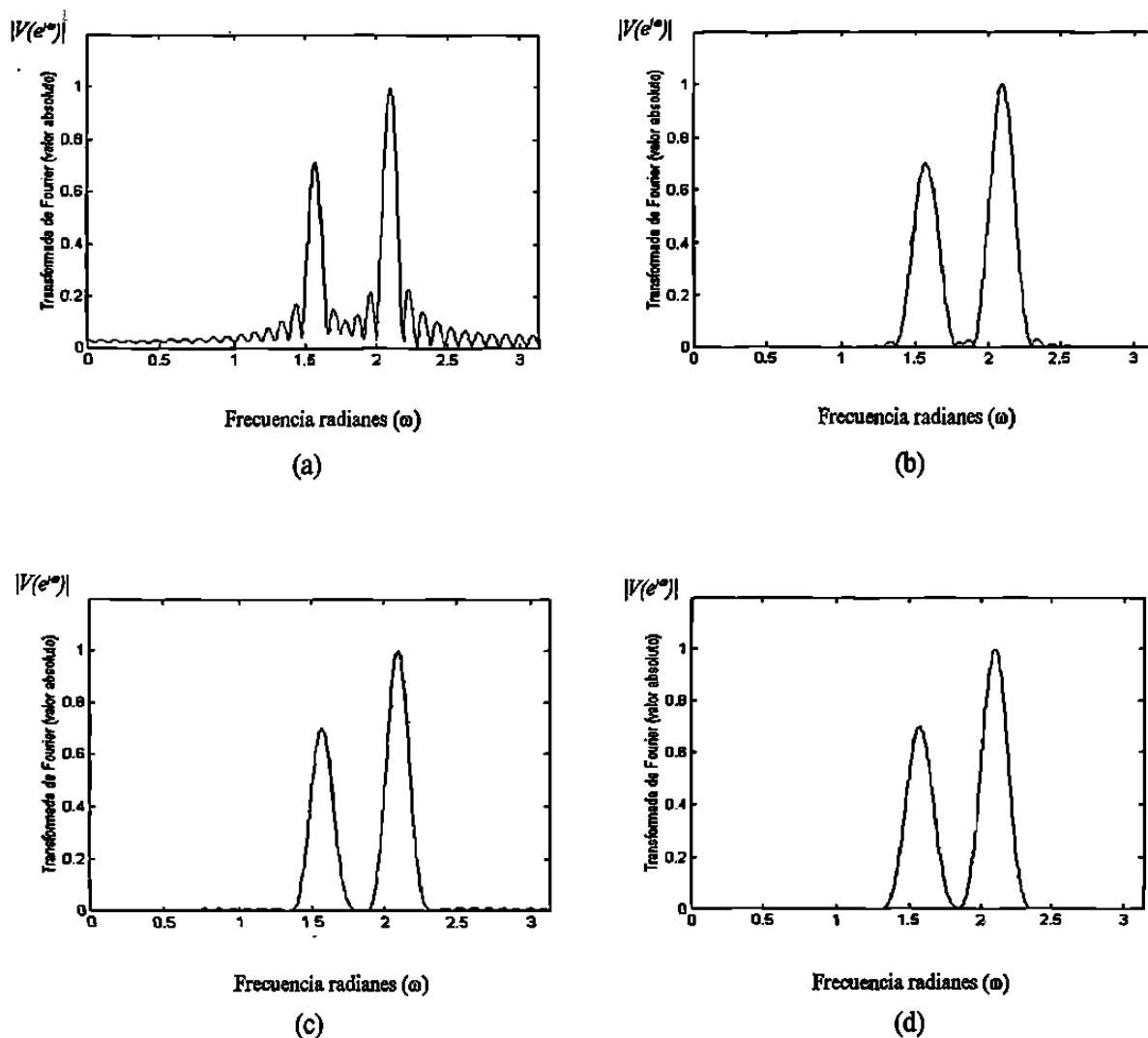


Fig. 3.10 Espectro de amplitud para dos tonos puros empleando diferentes ventanas. a) Ventana rectangular. b) Ventana de Hanning. c) Ventana de Hamming. d) Ventana  $\bar{\sigma}_4$ .

### 3.7 ESPECTROGRAMAS

En los espectrogramas, la ventana  $\bar{\sigma}_4$ , prácticamente elimina la interferencia entre contenidos frecuenciales vecinos. La Fig. 3.11 muestra dos espectrogramas para una señal en la que se introduce una segunda armónica. La primera gráfica usa la ventana  $\bar{\sigma}_4$ , mientras que la segunda emplea la ventana rectangular. En el primer caso se pueden observar claramente bandas suprimidas; mientras que en el segundo, se observa como se desparrama la energía, interfiriendo en la amplitud de la componente fundamental. Se observa además, que la ventana

rectangular es muy sensible a la fase relativa de la señal. En las siguientes secciones se presenta la aplicación de las nuevas ventanas en el diseño de filtros digitales FIR.

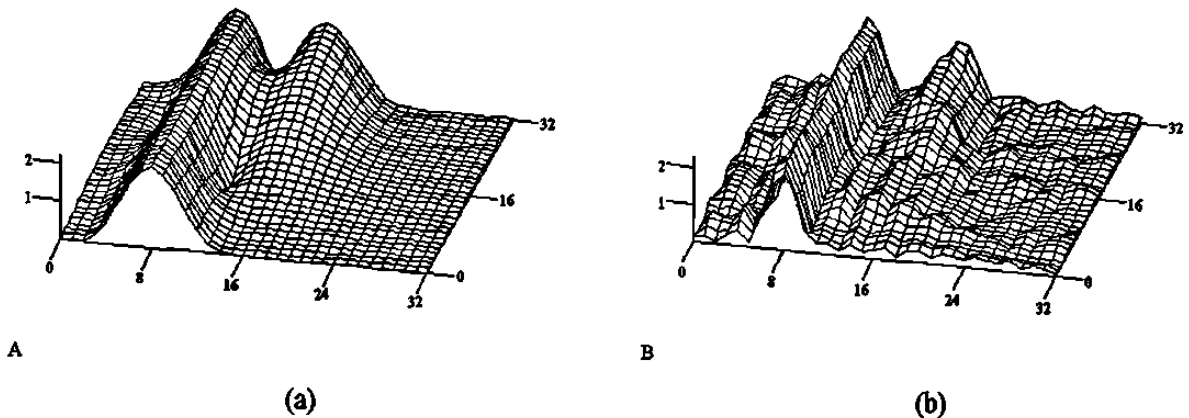


Fig. 3.11 Espectrogramas que ilustran el transitorio de una señal en la que se introduce una segunda componente armónica a) Ventana  $\bar{O}_4$ . b) Ventana rectangular.

### 3.8 APLICACIONES DE LA NUEVA FAMILIA DE VENTANAS EN EL DISEÑO DE FILTROS FIR

En las secciones anteriores se presentaron las características de la secuencia de funciones temporales  $o_M(t)$ . Considerando ahora estas funciones como respuestas impulsionales, la correspondiente secuencia de espectros  $O_M(f)$ , constituye una nueva opción para el diseño de filtros FIR; este tema es tratado ampliamente en [47]. Bajo esta nueva óptica, la Fig. 3.3 puede ser leída como la respuesta a la frecuencia de los filtros pasabajas de orden  $N=2, 4$  y  $8$ . Realmente, dichos filtros se obtienen aplicando el procedimiento de aproximación funcional de filtros selectivos en frecuencia. Los detalles de esta metodología aparecen en [15,16,17]. Este método se desarrolla en términos de aproximaciones al filtro pasabajas ideal, y básicamente consiste en definir la familia de funciones en frecuencia que determinan el tipo de bordes laterales del filtro. A cada familia de funciones corresponde un tipo de filtro. Los filtros Butterworth, Chebyshev y elípticos provienen de este método. Sin embargo, los nuevos filtros, a diferencia de los filtros Chebyshev, elípticos y Butterworth, tienen una banda supresora hermética, son no casuales y de fase nula.

A continuación se estudian las características de los filtros pasabanda obtenidos mediante una simple sintonización frecuencial de las funciones  $o_N(t)$ . Los filtros FIR se obtienen truncando las respuestas infinitas al impulso. En base a lo anterior, las respuestas impulsionales de los nuevos filtros se obtiene aplicando la ventana  $\bar{o}_N(t)$  a las ondas coseno y seno:

$$h_c(t) = \bar{o}_N(t) \cos(2\pi f_0 t), \quad (3.22)$$

$$h_s(t) = j\bar{o}_N(t) \text{sen}(2\pi f_0 t). \quad (3.23)$$

donde  $f_0$  es la frecuencia fundamental. El filtro seno se define imaginario con el objeto de ponerlo en cuadratura con el coseno.

Las respuestas frecuenciales de dichos filtros son las siguientes:

$$H_c(f) = \frac{1}{2} [\bar{O}_N(f + f_0) + \bar{O}_N(f - f_0)], \quad (3.24)$$

$$H_s(f) = \frac{1}{2} [-\bar{O}_N(f + f_0) + \bar{O}_N(f - f_0)], \quad (3.25)$$

donde la función  $\bar{O}_N(f)$  es el espectro de la ventana  $\bar{o}_N(t)$ . Nótese además, que ambos filtros, son las convoluciones en frecuencia de la transformada de Fourier de la ventana  $\bar{o}_N(t)$  con las transformadas de las funciones coseno y seno imaginario. Por lo tanto, la respuesta a la frecuencia de los filtros ortogonales combinan dos réplicas de  $\bar{O}_N(f)$  sintonizadas en las frecuencias  $\pm f_0$ .

La respuesta impulsional del nuevo filtro es la suma de la de los dos filtros anteriores:

$$h_N(t) = h_c(t) + h_s(t) \quad (3.26)$$

y su transformada de Fourier es la siguiente:

$$H_N(f) = H_c(f) + H_s(f) = \bar{O}_N(f - f_0). \quad (3.27)$$

$H_N(f)$  es una simple réplica de  $\bar{O}_N(f)$  sintonizada en  $f_0$ , es decir, real, con un único lóbulo en  $f_0$ .

### 3.8.1 Discretización de los filtros

A continuación se consideran las características frecuenciales de los filtros digitales. Las secuencias de las respuestas al impulso de los filtros digitales se obtienen muestreando las funciones (3.22) y (3.23). Si  $\tau$  es el período de muestreo, dichas secuencias son las siguientes:

$$h_c[n] = h_c(n\tau); \quad h_s[n] = h_s(n\tau). \quad (3.28)$$

La transformada de Fourier de una secuencia  $h[n]$ , obtenida por el muestreo de la función continua  $h(t)$ , es:

$$\tilde{H}(f) = f_m \sum_{k=-\infty}^{\infty} H(f - kf_m), \quad (3.29)$$

donde  $H(f)$  es la transformada de Fourier de  $h(t)$ . Se sabe que  $\tilde{H}(f)$  es una función repetitiva en frecuencia, formada por la superposición de réplicas de  $H(f)$  de magnitud  $f_m$ , sintonizadas en frecuencias múltiplos enteros de la frecuencia de muestreo  $f_m = 1/\tau$ . El ancho de banda de los filtros continuos es  $2f_0$ . Si se desea evitar los empalmes de las réplicas, se requiere una frecuencia de muestreo no menor que  $4f_0$ . Esta frecuencia de muestreo es muy baja en comparación con la requerida por los filtros convencionales de Fourier, debido a sus extendidos lóbulos laterales (ver Fig. 3.12) [6,7]. Además, no requiere ser múltiplo de la frecuencia fundamental, ya que los filtros resultantes tienen una banda supresora hermética.

Al restringir la función ideal  $o_N(t)$  a una duración finita  $D$ , se obtiene la ventana  $\bar{o}_N(t)$ . Este truncamiento introduce pequeños lóbulos laterales en el espectro ideal de  $o_N(t)$ , resultantes de la convolución frecuencial del lóbulo de dicha función con el seno cardenal de la ventana rectangular truncante. La Fig. 3.13 muestra la respuesta a la frecuencia del filtro  $H_c(f)$



utilizando la ventana  $\bar{o}_2(t)$  de duración  $D=3T$  y  $M=32$ . Cambiando el signo del lóbulo izquierdo se obtiene  $H_s(f)$ .

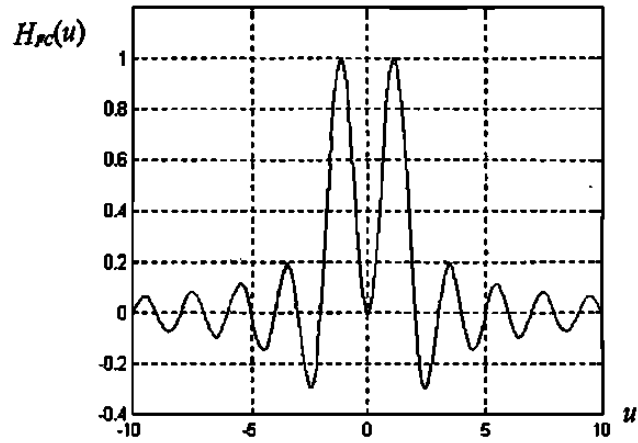


Fig. 3.12 Filtro convencional de Fourier tipo coseno, donde  $u=ff_0$ .

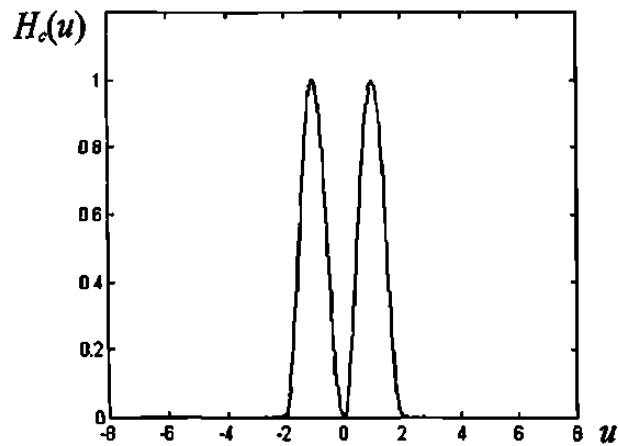


Fig. 3.13 Respuesta a la frecuencia del filtro  $H_c(f)$  utilizando la ventana  $\bar{o}_2(t)$  de duración  $D=3T$  y  $M=32$ , donde  $u=ff_0$ .

Para efectuar un buen filtrado, los nuevos filtros requieren una duración mínima de dos ciclos, esta duración es superior a la de los filtros convencionales de Fourier, lo cual puede ser inapropiado para las aplicaciones que deben tomar una decisión en un tiempo inferior a los dos ciclos.

### 3.8.2 Implementación de los filtros

A continuación se describe la implementación digital de los nuevos filtros. Sea  $s[n]$  la señal de interés. Definiendo las proyecciones ortogonales de dicha señal como  $g[n]$  y  $h[n]$ , se tiene:

$$g[n + L] = \frac{2}{Lf_0} \sum_{k=0}^{Q-1} s[n + k] \bar{o}_N[k - L] \cos\left(\frac{2\pi k}{L}\right), \quad (3.30)$$

$$h[n + L] = \frac{-2}{Lf_0} \sum_{k=0}^{Q-1} s[n + k] \bar{o}_N[k - L] \text{sen}\left(\frac{2\pi k}{L}\right), \quad (3.31)$$

$$n=0,1,\dots, Q - \delta L;$$

donde  $L$  es la cantidad de muestras por ciclo,  $n$  representa los instantes de muestreo,  $\delta$  la duración del filtro en ciclos y  $Q$  la cantidad de muestras a procesar. Las secuencias  $g[n]$  y  $h[n]$  resultan de la convolución digital de la señal con las respuestas impulsionales de los filtros  $h_c[n]$  y  $h_s[n]$ . Nótese que la reversa de la secuencia impulsional que interviene en las convoluciones afecta solamente a la función senoidal por un simple cambio de signo. Por ello, (3.30) y (3.31) constituyen las partes real e imaginaria de la transformada de Fourier discreta de la señal segmentada con una función ventana y evaluada en la frecuencia fundamental, a excepción del signo negativo de (3.31). Por tanto, ambas convoluciones se pueden calcular mediante la transformada rápida de Fourier (FFT, *Fast Fourier Transform*) y el simple cambio de signo en la proyección vertical.

### 3.9 CONCLUSIONES

- Se presentó el desarrollo matemático, características temporales y espectrales, y aplicaciones de una nueva familia de ventanas, las cuales tienen excelentes características frecuenciales y pueden emplearse tanto en aplicaciones de estimación espectral como de filtrado digital.
- La ventana  $\bar{\sigma}_4$  supera a las ventanas convencionales en cuanto que sus lóbulos laterales tienen una amplitud relativa inferior. Con un nivel relativo de lóbulos laterales tan bajo (de -55 dB), se puede considerar prácticamente carente de ellos. Sin embargo, su lóbulo principal es más ancho.
- Se demostró que la nueva familia de ventanas se aproximan a una curva gaussiana conforme se incrementa el factor leptógeno  $N$ . Además, se pueden definir ventanas, al igual que la ventana de Káiser, que sean equivalentes a cada una de las ventanas clásicas en cuanto a la anchura de su lóbulo principal, incluso mejorando la amplitud relativa de sus lóbulos laterales.
- Las características espectrales de la nueva familia de ventanas (anchura de su lóbulo principal y amplitud relativa de sus lóbulos laterales), se pueden modificar variando el factor leptógeno  $N$  y su duración  $D$  respectivamente. Una vez definida una ventana, se puede reducir su ancho espectral aplicando escalamiento temporal.
- Al tener lóbulos laterales muy bajos, la ventana  $\bar{\sigma}_4$  elimina prácticamente la interferencia entre contenidos espectrales vecinos. Por lo que se recomienda su uso en análisis armónico y en espectrogramas, donde la separación armónica es conocida. En ese caso, para evitar interferencias se puede definir el ancho del lóbulo principal igual o inferior a la separación interarmónica.
- Los filtros generados a partir de la nueva familia de ventanas mejoran los filtros convencionales de Fourier, dado que minimizan las fugas interarmónicas. Son muy útiles en el diseño de filtros digitales donde se desea el paso de una cierta banda de frecuencias y un

fuerte bloqueo de las frecuencias restantes, en estimación fasorial en sistemas eléctricos de potencia y en detección de portadoras en sistemas de comunicación digital.

- La desventaja de estos filtros es que requieren un mínimo de dos ciclos para efectuar un buen filtrado, lo cual los descarta de aplicaciones de protección de líneas de transmisión. Sin embargo, para la presente aplicación de detección y registro de señales de fallas, dicha limitación no representa ningún problema.