





Universidad Autónoma de San Luis Potosí  
Facultad de Ingeniería

# ANÁLISIS EN TIEMPO-FRECUENCIA DE SEÑALES NO ESTACIONARIAS

TESIS

Que para obtener el grado de  
Maestro en Ingeniería Eléctrica

P R E S E N T A :

Ing. Javier Salvador González Salas

San Luis Potosí, S.L.P., Noviembre 1998





MAYO 28, 1998.

Al Ing. Javier Salvador González Salas  
P r e s e n t e.-

En atención a su solicitud de autorización de Temario, presentada - por el Dr. Jesús Urias Hermosillo, Asesor de la Tesis de Maestría que deberá desarrollar en su Examen de Grado en la Maestría en Ingeniería Eléctrica. Me es grato comunicarle que en la Sesión de Consejo Técnico Consultivo celebrada el día 28 de Mayo del presente año, fué - aprobado el Temario propuesto:

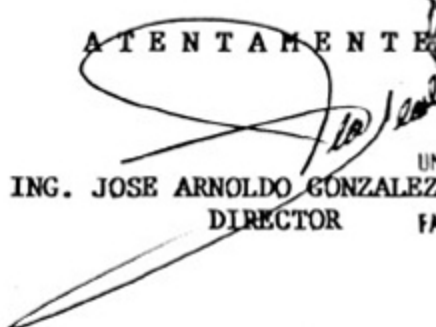
" ANALISIS EN TIEMPO-FRECUENCIA DE SEÑALES NO ESTACIONARIAS "

TEMARIO:

- I.- INTRODUCCION
- II.- ANALISIS EN TIEMPO-FRECUENCIA
- III.- SISTEMA DE HERRAMIENTAS COMPUTACIONALES
- IV.- RESULTADOS
- V.- CONCLUSIONES
- VI.- APENDICES
- BIBLIOGRAFIA

" MODOS ET CUNCTARUM RERUM MENSURAS AUDEBO "

ATENTAMENTE

  
 ING. JOSE ARNOLDO GONZALEZ ORTEGA  
 DIRECTOR

UNIVERSIDAD AUTONOMA  
 DE SAN LUIS POTOSI  
 FACULTAD DE INGENIERIA  
 DIRECCION

75



ANIVERSARIO  
SIEMPRE AUTONOMA

"1998, 75. Aniversario de la Autonomía Universitaria"

'real.



Universidad Autónoma de San Luis Potosí  
Facultad de Ingeniería

**CENTRO DE INVESTIGACIÓN Y ESTUDIOS DE  
POSGRADO**

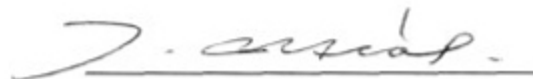
**MAESTRÍA EN INGENIERÍA ELÉCTRICA  
OPCIÓN CONTROL AUTOMÁTICO**

**ANÁLISIS EN TIEMPO-FRECUENCIA DE SEÑALES  
NO ESTACIONARIAS**

Presenta:

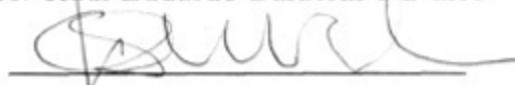
\_\_\_\_\_  
Ing. Javier Salvador González Salas

Sinodales:



Dr. Jesús Urías Hermosillo  
(Asesor de tesis)

\_\_\_\_\_  
Dr. Raúl Eduardo Balderas Navarro



Dr. Salvador Guel Sandoval

San Luis Potosí, S.L.P., Noviembre 1998

## Agradecimientos

Deseo expresar los siguientes reconocimientos.

A mi asesor el Dr. Jesús Urías por haber propuesto y guiado este trabajo; además de sus enseñanzas para forjarme como investigador.

Al Dr. Raúl Balderas Navarro y al Dr. Salvador Guel Sandoval por haber sido miembros del comité de sinodales y por sus recomendaciones propuestas para la escritura de la tesis.

Al director, Dr. Alfonso Lastras, y al secretario académico Dr. Hugo Navaro, del IICO-UASLP por todas las facilidades y ayuda otorgada.

Al laboratorio de Comunicaciones del IICO-UASLP donde se realizó el trabajo para esta tesis, con financiamiento parcial del FAI-UASLP.

A mis profesores del CIEP-UASLP.

Al CoNaCyT por la beca otorgada para realizar los estudios de Maestría.

A todos mis compañeros de la tercera versión en la Maestría en Ingeniería Eléctrica opción Control Automático.

*A mi madre,  
a mis hermanas,  
y a Michelle.*

## Resumen

Se desarrolla un sistema de herramientas de software en ambiente gráfico para análisis de señales no estacionarias mediante representaciones en tiempo-frecuencia conocidas como espectrogramas. Las herramientas son programadas en lenguaje C con diversas opciones y son manejadas desde un ambiente gráfico elaborado en lenguaje TCL. Para probar el sistema y sus funciones se generan espectrogramas para la señal  $\sin(\frac{1}{t})$ , así como para señales de electrocardiogramas y para señales de voz.

# Contenido

<b>1</b>	<b>Introducción</b>	<b>1</b>
<b>2</b>	<b>Análisis en Tiempo-Frecuencia</b>	<b>5</b>
2.1	Introducción . . . . .	5
2.2	Transformada discreta de Fourier . . . . .	6
2.2.1	Transformada de Fourier con ventana . . . . .	10
2.3	Interpolación . . . . .	14
2.3.1	Interpolación polinomial . . . . .	14
2.3.2	Fórmula de Reconstrucción de Shannon . . . . .	19
2.4	Espectrogramas . . . . .	23
2.5	Algoritmos para generación de espectrogramas . . . . .	26
2.5.1	Cálculo de los espectros . . . . .	30
2.5.2	Escalas . . . . .	36
<b>3</b>	<b>Sistema de Herramientas Computacionales</b>	<b>41</b>
3.1	Selección de Archivos . . . . .	41
3.2	Manejo de archivo de datos . . . . .	43
3.2.1	Formato de Archivo . . . . .	47
3.3	Características para generar un espectrograma. . . . .	48
<b>4</b>	<b>Resultados</b>	<b>57</b>
4.1	Análisis de la señal $\sin(\frac{1}{t})$ . . . . .	59
4.1.1	Variación en la representación de la TF y el modo derivado . . . . .	59
4.1.2	Variación en la resolución . . . . .	61
4.1.3	Modo de ajuste al máximo . . . . .	62
4.1.4	Variación en la interpolación . . . . .	62
4.2	Espectrogramas de electrocardiogramas . . . . .	64



4.2.1	Variación del tipo de ventana . . . . .	64
4.2.2	Variación del tamaño de ventana y el paso . . . . .	66
4.2.3	Dos electrocardiogramas diferentes . . . . .	67
4.3	Análisis de una señal de voz . . . . .	69
<b>5</b>	<b>Conclusiones</b>	<b>73</b>
	<b>Apéndices</b>	<b>75</b>
<b>A</b>	<b>Funciones en lenguaje Tcl</b>	<b>75</b>
A.1	Manual de procedimientos en Tcl . . . . .	77
A.2	Manual de comandos . . . . .	88
<b>B</b>	<b>Manual de funciones en lenguaje C</b>	<b>91</b>
	<b>Bibliografía</b>	<b>107</b>

# Lista de Figuras

1.1	Señal en tiempo (intensidad vs tiempo) de un chillido de murciélago [9].	2
1.2	Transformada de Fourier (magnitud vs frecuencia) del chillido de murciélago de la señal en la figura 1.1 [9].	2
1.3	Espectrograma (frecuencia vs tiempo) de un chillido de murciélago [9].	3
2.1	Señal $x(t) = 2 \cdot \text{sen}(2 \cdot \pi \cdot 100)t + 2 \cdot \text{sen}(2 \cdot \pi \cdot 200)t + 2 \cdot \text{sen}(2 \cdot \pi \cdot 500)t$ , para $t = 0, T, \dots, nT, 63T$ , con $T = 0.0005$ .	9
2.2	Transformada de Fourier $X(k)$ de la señal que se encuentra en la figura 2.1. El rango de frecuencias es de 0 a $2\pi$ , y de $k = 0, \dots, 63$ .	10
2.3	Muestra el proceso para realizar la transformada de Fourier con ventana. El contenido de frecuencia del segmento $x_p[m]$ de la señal no estacionaria $x[n]$ , puede calcularse si se multiplica por la ventana $w[m]$ , donde $x_d[m] = x_p[m] \cdot w[m]$ , y obteniendo la transformada de Fourier de la secuencia $x_d[n]$ , llamada $X_d[k]$ .	11
2.4	Gráfica de las 4 ventanas que utiliza el SATF para el cálculo de la transformada de Fourier con Ventana. Estas son: ventana Rectangular, ventana de Hanning, ventana de Hamming y ventana de Blackman. Cada ventana ha sido generada con 32 datos.	12
2.5	Ventana de tiempo ( <i>Hanning</i> ) que se mueve a lo largo del eje t.	24
2.6	Transformadas de Fourier correspondientes a los segmentos que se obtienen con el traslado de la ventana a través del tiempo de la figura 2.5.	25
2.7	Correspondencia entre la transformada de Fourier con magnitud máxima, y la escala de colores que se utilizará para pintar el espectrograma.	26
2.8	Construcción del espectrograma utilizando los $X_{d_j}(k)$ . En el diagrama, el tiempo corre a través del eje horizontal, y la frecuencia se mide en el eje vertical de 0 a $2\pi$ .	26
2.9	Orden de colocación de las frecuencias $X_{d_j}(k)$ ( para $j = 0, \dots, m - 1$ , $k = 0, \dots, M/2 - 1$ ) representadas por colores, utilizadas para armar el espectrograma.	35

2.10	Barra de colores utilizada para representar los espectrogramas. . . . .	36
3.1	Ventana principal del sistema de análisis en tiempo-frecuencia. . . . .	42
3.2	Ventana de selección de archivos. . . . .	43
3.3	Señal no estacionaria, interpolada con la fórmula de interpolación polinomial y con la fórmula de reconstrucción Shannon. La líneas verticales (impulsos) representan a la señal no interpolada. La línea consecutiva que une cruces (+) es la señal interpolada con la fórmula de Shannon. Los rombos representan la señal interpolada con la fórmula polinomial. . . . .	44
3.4	Señal de una derivación de un electrocardiograma. Los ejes del diagrama representan amplitud contra tiempo en unidades arbitrarias. . . . .	45
3.5	Transformada Discreta de Fourier de la señal del electrocardiograma que se encuentra en la figura 3.4. . . . .	46
3.6	Ventana de Caracterización de Espectrograma. . . . .	50
3.7	Figura que ilustra la manera de como son las ventanas gráficas donde los espectrogramas son presentados. . . . .	56
4.1	Señal $\sin(1/t)$ muestreada a tiempos de $t = 0.001, 0.001 + T, \dots, 0.001 + nT, \dots, 0.1032$ , donde $T = 0.002$ segundos. . . . .	59
4.2	Espectrogramas de las señal $\sin(1/t)$ correspondientes a la magnitud, a la magnitud en decibels y a la fase (diagramas a,b,d respectivamente) de la transformada de Fourier. Además el espectrograma de las señal $\sin(1/t)$ derivada generado con la magnitud de la TF (diagrama d). Las características comunes de los espectrogramas son: ventana de Blackman, tamaño de ventana = 20% resolución baja, paso = 2, e interpolación polinomial. . . . .	60
4.3	Espectrograma de las función $\sin(1/t)$ utilizando cuatro resoluciones diferentes: 64, 128, 256, y 512 (imágenes a,b,c y d respectivamente). Las características comunes de los espectrogramas son: representación de la magnitud de la TF, ventana de Blackman, tamaño de ventana = 20%, paso = 2, e interpolación polinomial. . . . .	61
4.4	Espectrograma generado con el valor máximo de la <i>magnitud</i> ajustado a un 6% del valor máximo obtenido en el espectrograma sin ajustar, mostrado en el diagrama (a) de la figura 4.2. El espectrograma fue generado con las mismas características que el espectrograma de la figura 4.2 (a). . . . .	62

4.5	Espectrograma de la señal $\text{sen}(1/t)$ generado con la opción de la interpolación con la fórmula de reconstrucción de Shannon. Las características del espectrograma son las mismas que tiene el espectrograma de la figura 4.2 (b), excepto la interpolación. . . . .	63
4.6	Electrocardiograma 1 (amplitud vs tiempo en unidades arbitrarias) . . .	64
4.7	Espectrogramas que analizan la señal del electrocardiograma 1 utilizando la 4 ventanas de tiempo del SATF: rectangular, Hanning, Hamming y Blackman. Las características comunes de los espectrogramas son: representación de la magnitud de la TF, resolución = 128, tamaño de ventana = 15 %, paso = 10 e interpolación polinomial. . . . .	65
4.8	Espectrogramas que analizan la señal del electrocardiograma 1. Muestran la manera de como el cambio del <i>paso</i> afecta el bosquejo del mismo. Las características comunes de los espectrogramas son: representación de la magnitud de la TF, ventana de Blackman, resolución = 128, tamaño de ventana = 15% e interpolación polinomial . . . . .	66
4.9	Espectrogramas de la señal del electrocardiograma 1 que muestran la manera de como afecta el cambio del tamaño de ventana al bosquejo del mismo. Las características comunes de los espectrogramas son: representación de la magnitud de la TF, ventana de Blackman, resolución = 128, paso = 10 e interpolación polinomial. . . . .	67
4.10	Señales del electrocardiograma 2 y el electrocardiograma 3 [12] (diagramas (a) y (b) respectivamente) y sus los espectrogramas que las analizan (diagramas (c) y (d) respectivamente). Los 2 espectrogramas están generados con las mismas características del diagrama (d) de la figura 4.7.	68
4.11	Señal de voz de Homero Simpson de 39954 datos (amplitud vs tiempo).	69
4.12	Señal de voz de Homero Simpson de 2048 datos (amplitud vs tiempo).	69
4.13	Espectrograma de señal de voz de Homero simpson. Las características con las que se generó el espectrograma se encuentran listadas en la página 70 de esta sección. . . . .	70
4.14	Señal completa de la voz de H. Simpson (amplitud vs tiempo) mostrando la sección de la señal que será analizada. . . . .	71
4.15	Sección extraída de la señal de voz completa de H. Simpson (amplitud vs tiempo), ya interpolada , y reducida a 2048 datos. En la señal se indican la amplitudes que tienen mayor magnitud, distinguidas como amplitudes A y amplitudes B. La sección de la señal entera de donde fue extraído este segmento de voz se muestra en la figura 4.14. . . . .	71

- 4.16 Espectrograma que analiza la sección de señal extraída de la voz de Homero Simpson. Las características del espectrograma son: representación de la magnitud TF, ventana de Blackman, resolución = 64, *paso* = 10 e interpolación polinomial. . . . . 72

# Lista de Tablas

2.1	Tabla de los polinomios de Lagrange que describen el algoritmo de Neville para la interpolación utilizando un polinomio de tercer orden [7]. . . . .	15
2.2	Modos de lectura definidos para el sistema de análisis en tiempo-frecuencia. Donde: $n = 0, 1, \dots, NP - 1$ , y $T$ es el periodo de muestreo. $NP$ es el número de datos que contiene archivo a ser leído. . . . .	17
2.3	Lista de parámetros de la sintaxis de los comandos de Tcl <code>fftspec</code> y <code>fpbspec</code> . . . . .	27
3.1	Contenido de las columnas de datos en los archivos de datos con transformada de Fourier. . . . .	48
3.2	Opciones de los parámetros requeridos en la ventana de caracterización. . . . .	49
3.3	Parámetros, que se muestran en las ventanas gráficas donde los espectrogramas son presentados . . . . .	55
4.1	Lista de parámetros para generar un espectrograma. . . . .	57

# Capítulo 1

## Introducción

La gran mayoría de los fenómenos naturales generan señales no estacionarias (sus parámetros varían con respecto al tiempo), y su estudio ha tenido una gran importancia (por ejemplo las señales biomédicas de electrocardiogramas [10]-[11] y encefalogramas [11]). Generalmente, la representación de estas señales en tiempo-intensidad, tiene la mayoría de las veces formas muy complicadas [9] y pueden ser difíciles de analizar ó estudiar. La representación en tiempo-frecuencia de la señal proporciona información ó patrones más fáciles de diferenciar que la representación tiempo-intensidad [9].

Las representaciones en tiempo-frecuencia despliegan el cambio de contenido de frecuencia en la señal a través del tiempo [9]. Un chillido de murciélago, por ejemplo, proporciona una excelente motivación para el procesamiento de señales basado en tiempo-frecuencia [9]. Como se muestra en las figuras 1.1 y 1.2, ni la señal en tiempo ni su espectro de Fourier, revelan una estructura sencilla de la señal como lo hace su espectrograma mostrado en la figura 1.3.

El objetivo de esta tesis es desarrollar una herramienta de software para el análisis en tiempo-frecuencia de señales no estacionarias. La herramienta de software es llamada Sistema de Análisis en Tiempo-Frecuencia (SATF) de señales no estacionarias . Utiliza la técnica matemática de la transformada de Fourier con ventana para señales discretas (digitalizadas). El SATF es un ambiente gráfico que consta de operaciones para interpolar, obtener transformada de Fourier y generar espectrogramas. Estas operaciones se aplican sobre archivo de datos con formatos determinados.

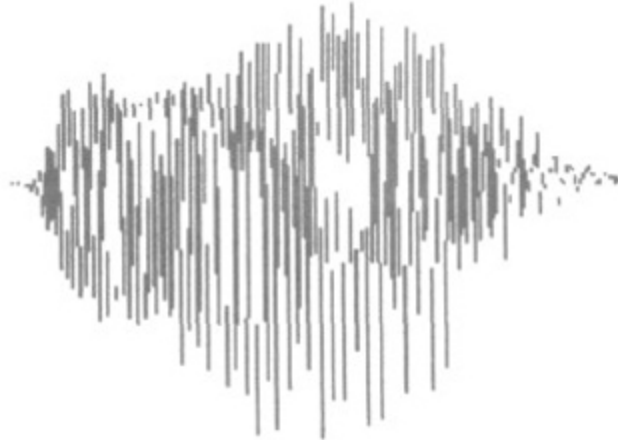


Figura 1.1: Señal en tiempo (intensidad vs tiempo) de un chillido de murciélago [9].

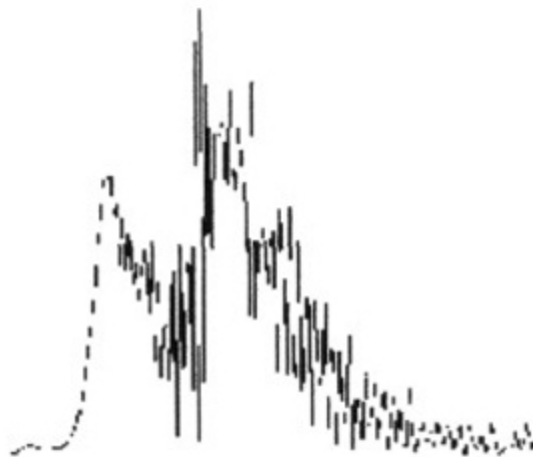


Figura 1.2: Transformada de Fourier (magnitud vs frecuencia) del chillido de murciélago de la señal en la figura 1.1 [9].



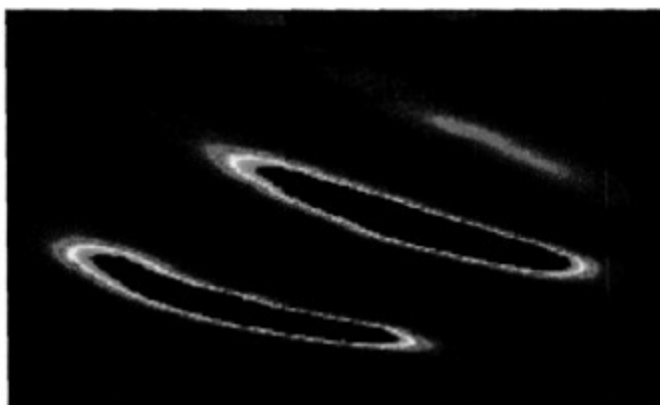


Figura 1.3: Espectrograma (frecuencia vs tiempo) de un chillido de murciélago [9].

El SATF contiene procedimientos, manejados en un ambiente gráfico, para el tratamiento de archivo de datos. Los archivos de datos son de cuatro tipos: datos crudos (extensión `.raw`), datos interpolados (extensión `.intp`), datos con transformada de Fourier (extensión `.fft`) y archivo de datos utilizados para crear espectrogramas (extensión `.spc`).

Cada tipo de archivo tiene sus características. Los archivos de datos crudos contienen la señal que se desea analizar. Todavía no se les ha efectuado alguna operación del SATF. Los archivos de datos interpolados, son archivos de datos crudos que se les ha ajustado el número de datos sin perder la forma de la señal. Los archivos de datos con transformada de Fourier, son archivos de datos interpolados que se le ha obtenido su respuesta en frecuencia. Los archivos con extensión `.spc` contienen los parámetros para construir un espectrograma que ha sido creado anteriormente.

En el Capítulo (II) se hace una breve revisión del algoritmo matemático de la transformada de Fourier para señales discretas llamado transformada discreta de Fourier, así como la técnica que obtiene el contenido de frecuencia en un segmento de una señal llamada transformada de Fourier con ventana. Se describen las herramientas matemáticas utilizadas para la interpolación de datos, que son la interpolación polinomial y la fórmula de reconstrucción de Shannon, así como los programas de cómputo que desar-

rollan estas técnicas. Después, se explica la manera de como se construyen los diagramas en tiempo-frecuencia, y las funciones correspondientes en lenguaje C que la realizan.

El Capítulo (III) describe el Sistema de Análisis en tiempo-frecuencia y se explica la forma en que el SATF trata los archivos de datos y los programas ejecutables. Las principales ventanas del ambiente gráfico del sistema son la ventana principal, la ventana de selección de archivos y la ventana de caracterización de espectrograma. Se describen todos los parámetros para generar espectrogramas, y las opciones que tiene cada uno de estos parámetros.

En el cuarto Capítulo se ponen a prueba las funciones del SATF. Se generan espectrogramas con diversas opciones para la señal  $\sin(1/t)$  y para una señal de electrocardiograma. Los espectrogramas muestran la forma en que afecta la variación de los parámetros de generación del espectrograma en la apariencia de los mismos. Se generan los espectrogramas de otras 2 señales de electrocardiogramas, donde se puede apreciar los detalles particulares que diferencian a cada uno de los espectrogramas correspondientes a los 3 electrocardiogramas mencionados. Se genera el espectrograma de una señal de voz, y después se genera el espectrograma de un segmento de la misma, con el propósito de obtener una mejor representación en tiempo-frecuencia en este tipo de señal. Al final se presentan las conclusiones de acuerdo a los resultados obtenidos.

## Capítulo 2

# Análisis en Tiempo-Frecuencia

### 2.1 Introducción

La técnica de análisis en tiempo-frecuencia de una señal proporciona una imagen que representa el cambio de los contenidos de frecuencia de una señal a través del tiempo [9]. Existen algunos métodos que la desarrollan. El método utilizado en este trabajo es el espectrograma, que se define como el espectro de la señal analizada vista a través de una ventana de tiempo que se mueve a lo largo del eje  $x$  [9].

Por convención, los espectrogramas serán generados a partir de los archivos de datos interpolados. Debido a que se analizan señales discretas, la ventana de tiempo estará representada por una secuencia de datos. Su tamaño será descrito como una fracción del archivo de datos interpolados del cual se va a generar un espectrograma. El SATF contiene 4 tipos de ventana: Rectangular, Hamming, Hanning y Blackman [1].

Como se analizan señales discretas (digitalizadas) se utiliza la transformada discreta de Fourier (DFT) para cada ventana de tiempo. Para incrementar la rapidez en el cálculo, la DFT se efectúa por un algoritmo de cómputo más rápido, la transformada rápida de Fourier (FFT).

Las características de la FFT requiere el ajuste de datos en las ventanas de tiempo. La FFT trabaja con cantidades de datos que son potencias de dos [8]. Si la ventana de tiempo no cumple con esta condición, su número de

datos tiene que ser aumentado a una cantidad que sea potencia de dos. El método utilizado para aumentar los datos es la interpolación.

Resumiendo, elaborar espectrogramas requiere principalmente:

- Ajuste de datos (interpolación).
- Ventanas de tiempo.
- Transformada discreta de Fourier.

A continuación se explicará con detalle todos los algoritmos necesarios en la construcción de espectrogramas.

## 2.2 Transformada discreta de Fourier

Debido a que las señales no estacionarias que se analizan son discretas (ya que se encuentran almacenadas en archivos de datos), en esta sección se describe el algoritmo de la transformada de Fourier para señales discretas.

Una secuencia  $x[n]$  se representa por una integral de Fourier de la forma:

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n}, \quad (2.1)$$

donde  $X(e^{j\omega})$  está dado por

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}. \quad (2.2)$$

Las ecuaciones 2.1 y 2.2 conforman la transformada de Fourier. La ecuación (2.2) es la transformada de Fourier para tiempo discreto de la secuencia  $x[n]$ . [2]

Para una secuencia de duración finita es posible desarrollar una representación de Fourier alternativa, llamada transformada discreta de Fourier (DFT). La DFT es una secuencia que corresponde a muestras igualmente espaciadas en frecuencia [2]. Esta juega un papel muy importante en la implementación de diversos algoritmos para el proceso digital de señales.

Considerando una secuencia de longitud finita  $x[n]$  que contiene  $N$  muestras, entonces el par de fórmulas para la DFT de esta secuencia es:

$$x[n] = \begin{cases} \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn}, & 0 \leq n \leq N-1 \\ 0 & \text{otro valor,} \end{cases} \quad (2.3)$$

$$X[k] = \begin{cases} \sum_{n=0}^{N-1} x[n] W_N^{kn}, & 0 \leq k \leq N-1 \\ 0, & \text{otro valor,} \end{cases} \quad (2.4)$$

en donde

$$W_N = e^{-j(w\pi/N)}. \quad (2.5)$$

El cálculo de una DFT de  $N$  puntos corresponde al cálculo de  $N$  muestras de la transformada de Fourier en  $N$  frecuencias igualmente espaciadas,  $\omega_k = w\pi k/N$ ; i.e., en  $N$  puntos sobre el círculo unitario [2].

El conjunto de algoritmos que son más eficientes en la obtención de la DFT son llamados transformada rápida de Fourier [3]. El algoritmo desarrollado para el cálculo de los espectros es el diezmado en tiempo. Resulta de la descomposición de la DFT en cálculos de DFT's más pequeñas [8]. El objetivo es efectuar menos operaciones aritméticas en la obtención de una DFT de  $N$  puntos. Para reducir la cantidad de operaciones al máximo es necesario que  $N$  sea potencia de dos. Como  $N$  es un entero par se puede calcular  $X[k]$  separando  $x[n]$  en dos secuencias de  $(N/2)$  puntos: en puntos con numeración par ( $x[2n]$ ) y puntos con numeración impar en ( $x[2n+1]$ ). Con  $X[k]$  dado por

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, \dots, N-1, \quad (2.6)$$

y separando  $x[n]$  en puntos pares e impares, obtenemos  $X[k]$  dado por [3]

$$X[k] = \sum_{n \text{ par}} x[n] W_N^{nk} + \sum_{n \text{ impar}} x[n] W_N^{nk}. \quad (2.7)$$

Sustituyendo  $n = 2r$  en muestras pares, y  $n = 2r + 1$  en muestras impares, tenemos que [3]:

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r]W_{N/2}^{rk} + \sum_{r=0}^{(N/2)-1} x[2r+1]W_{N/2}^{nk}. \quad (2.8)$$

Las sumatorias en (2.8) son a su vez DFT's. Sustituyendo las sumatorias por las transformadas  $G[k]$  y  $H[k]$ , respectivamente, tenemos [3]:

$$X[k] = \begin{cases} G[k] + W_N^k H[k] & \text{para } 0 \leq k \leq \left(\frac{N}{2} - 1\right) \\ G\left[k - \frac{N}{2}\right] - W_N^{k-\frac{N}{2}} H\left[k - \frac{N}{2}\right] & \text{para } \frac{N}{2} \leq k \leq (N - 1). \end{cases} \quad (2.9)$$

Continuando este proceso de separación en muestras pares y muestras impares, finalmente se obtendrán DFTs de longitud dos puntos para ser calculadas. El tiempo de la obtención de la FFT es menor que el de la obtención de la DFT [8]:

$$\text{tiempo de ejecución de la FFT} = \frac{\text{tiempo de ejecución de la DFT}}{(\log_2 N)}. \quad (2.10)$$

El algoritmo que obtiene la FFT [8] de una secuencia con N datos es:

```
void Fft(float *Re, float *Im, int Pwr, int Dir),
```

donde **Re**, **Im** son punteros a arreglos de números de punto flotantes. Estos arreglos corresponden a la parte real e imaginaria de la señal de entrada  $x[n]$ . **Pwr** es la dimensión de los arreglos flotantes representada en una potencia de dos. Por último **Dir** determina si se quiere ejecutar la FFT ( $Dir \geq 0$ ) ó la FFT inversa ( $Dir \leq 0$ ).  $X[k]$  es regresado en los mismos punteros donde se le pasa a la función la secuencia  $x[n]$ . La parte real de  $X[k]$  es ubicada en el arreglo que inicia en la posición que apunta **Re**, y la parte imaginaria de  $X[k]$  es almacenada en el arreglo al que apunta **Im**.

Se elaboró una función que obtiene la transformada de Fourier de un archivo de  $N$  datos, donde  $N$  es potencia de 2. La función es:

```
int Tfourier(char *intp, char *fft),
```

donde `intp` es el puntero al arreglo de caracteres que contiene el nombre del archivo de datos donde se almacena la secuencia  $x[n]$ , que será representado en frecuencia. Y el argumento `fft` es el puntero al arreglo de caracteres que contiene el nombre del archivo donde se almacenarán los datos de la representación en frecuencia  $X[k]$ .

La función `Tfourier` efectúa los siguientes pasos:

- Lee, y almacena en un arreglo global, los datos del archivo que contiene la secuencia a representar en frecuencia. Utiliza la función `Lec_Datos`
- Obtiene la transformada de Fourier de la secuencia almacenada, llamando a la función `Fft` (explicada anteriormente).
- Escribe los datos de la transformada de Fourier en el archivo de datos especificado por la cadena de caracteres donde apunta el parámetro de entrada `fft`.

Como ejemplo, se calcula la transformada de Fourier la señal sinusoidal:

$$2 * \text{sen}(2 * \pi * 100)t + 2 * \text{sen}(2 * \pi * 200)t + 2 * \text{sen}(2 * \pi * 500)t,$$

que se muestra en la figura 2.1. Su transformada de Fourier se muestra en la figura 2.2.

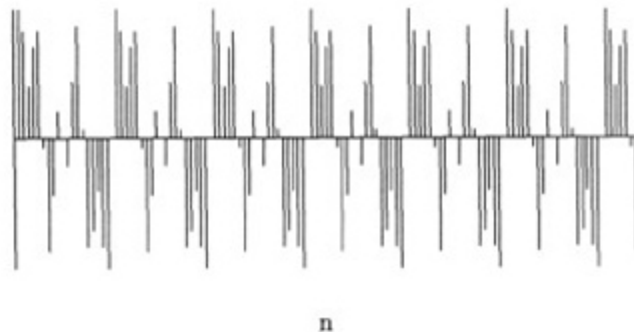


Figura 2.1: Señal  $x(t) = 2 * \text{sen}(2 * \pi * 100)t + 2 * \text{sen}(2 * \pi * 200)t + 2 * \text{sen}(2 * \pi * 500)t$ , para  $t = 0, T, \dots, nT, 63T$ , con  $T = 0.0005$ .

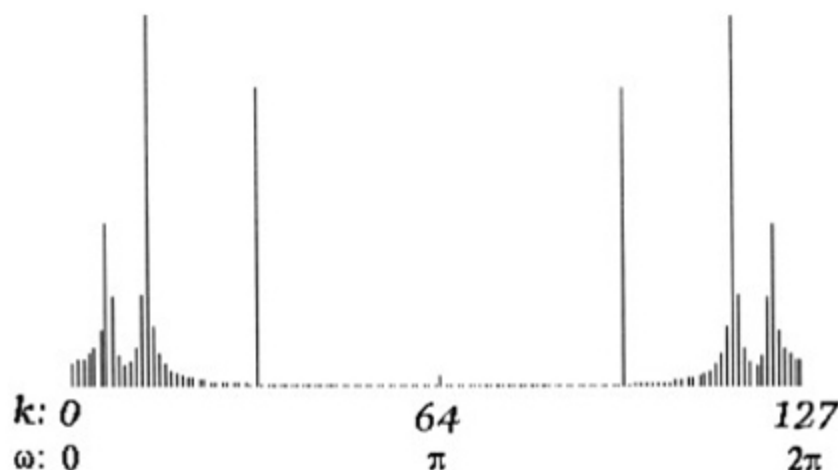


Figura 2.2: Transformada de Fourier  $X(k)$  de la señal que se encuentra en la figura 2.1. El rango de frecuencias es de 0 a  $2\pi$ , y de  $k = 0, \dots, 63$ .

### 2.2.1 Transformada de Fourier con ventana

La transformada de Fourier con ventana es una técnica matemática que nos sirve para analizar el contenido de frecuencia en un segmento de la señal no estacionaria. Se explicará brevemente esta técnica.

Si deseamos conocer el contenido de frecuencia de una señal deseada  $x_p[m]$  (para  $m = 0, \dots, M - 1$ ), que es un segmento de la señal discreta no estacionaria  $x[n]$  mostrada en la figura 2.3, es necesario multiplicarla por una ventana  $w[m]$  ( $m = 0, \dots, M - 1$ ), tal que:

$$x_d[m] = x_p[m] * w[m] \quad n = 0, \dots, M - 1, \quad (2.11)$$

y después obtener la transformada de Fourier de la secuencia  $x_d[m]$ :

$$X_d[k] = \sum_{m=0}^{M-1} x[m] e^{-j(\frac{wmk\pi}{N})}, \quad k = 0, 1, \dots, M - 1. \quad (2.12)$$

El proceso de obtener la transformada de Fourier del producto de la ventana  $w[w]$  por el segmento de una señal, es llamada transformada de Fourier con ventana.



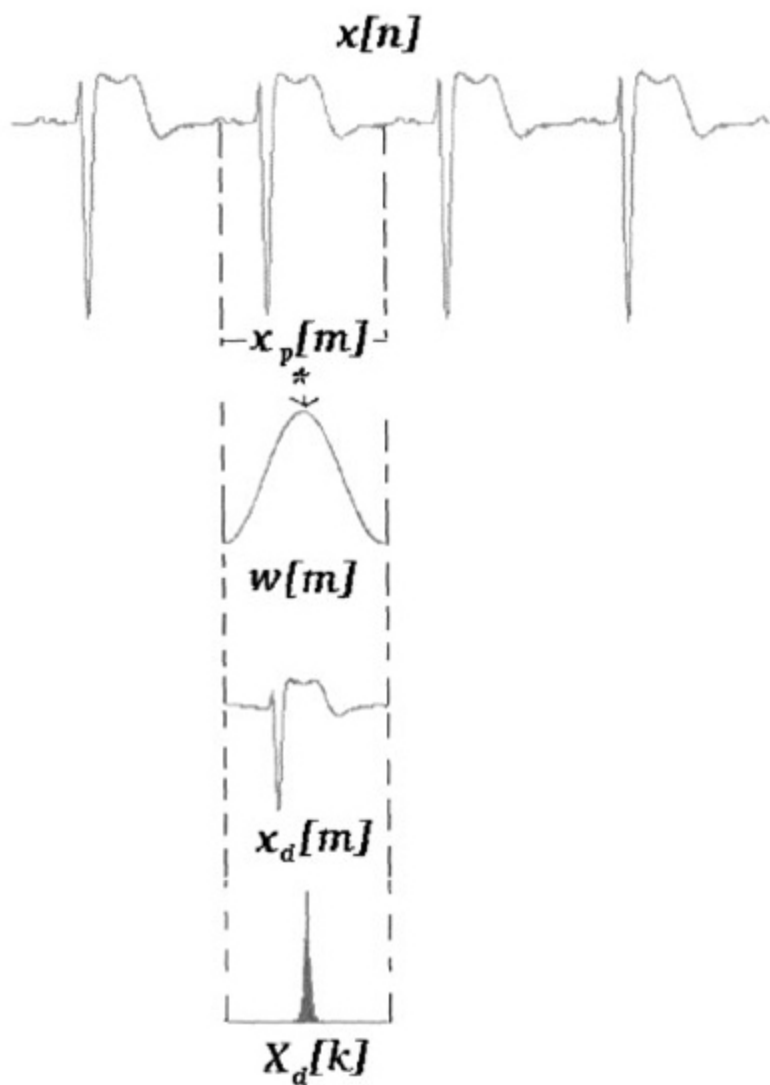


Figura 2.3: Muestra el proceso para realizar la transformada de Fourier con ventana. El contenido de frecuencia del segmento  $x_p[m]$  de la señal no estacionaria  $x[n]$ , puede calcularse si se multiplica por la ventana  $w[m]$ , donde  $x_d[m] = x_p[m] * w[m]$ , y obteniendo la transformada de Fourier de la secuencia  $x_d[n]$ , llamada  $X_d[k]$ .

Las ventanas utilizadas en el sistema de análisis en tiempo-frecuencia para obtener la transformada de Fourier con ventana son la ventana rectangular, la ventana de Hanning, la ventana de Hamming y la ventana de Blackman. La gráfica de las cuatro ventanas se muestra en la figura 2.4. Los algoritmos de cálculo de las ventanas [1] se describen a continuación:

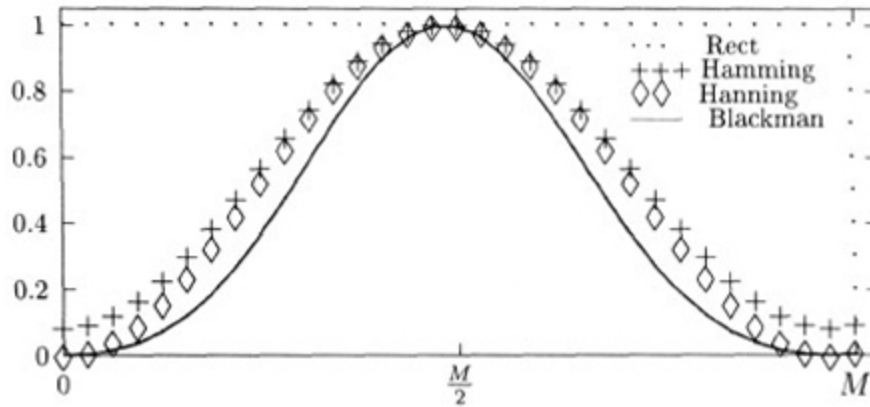


Figura 2.4: Gráfica de las 4 ventanas que utiliza el SATF para el cálculo de la transformada de Fourier con Ventana. Estas son: ventana Rectangular, ventana de Hanning, ventana de Hamming y ventana de Blackman. Cada ventana ha sido generada con 32 datos.

*Rectangular*

$$w[n] = \begin{cases} 1, & 0 \leq n \leq M, \\ 0, & \text{otro valor;} \end{cases} \quad (2.13)$$

*Hanning*

$$w[n] = \begin{cases} 0.5 - 0.5 \cos \frac{2\pi n}{M}, & 0 \leq n \leq \frac{M}{2}, \\ 0, & \text{otro valor;} \end{cases} \quad (2.14)$$

*Hamming*

$$w[n] = \begin{cases} 0.54 - 0.46 \cos \frac{2\pi n}{M}, & 0 \leq n \leq \frac{M}{2}, \\ 0, & \text{otro valor;} \end{cases} \quad (2.15)$$

*Blackman*

$$w[n] = \begin{cases} 0.42 - 0.5 \cos \frac{2\pi n}{M} + 0.08 \cos \frac{4\pi n}{M} & 0 \leq n \leq \frac{M}{2}, \\ 0, & \text{otro valor.} \end{cases} \quad (2.16)$$

Las funciones que calculan las secuencias de datos de las ventanas Rectangular, Hanning, Hamming y Blackman respectivamente son:

```
void Rect(float *f, int size)
void Hanning(float *f, int size)
void Hamming(float *f, int size)
void Blackman(float *f, int size)
```

El argumento *f* representa el puntero al arreglo flotante donde la función regresará los valores calculados de la secuencia de datos de la ventana. *size* es el tamaño de la ventana en número de datos requerido.

Los algoritmos de cómputo construidos para las funciones que calculan la secuencia de datos de las ventanas son muy sencillos. Dentro de un ciclo de *size* iteraciones, se van calculando cada uno de los valores de la secuencia de la ventana. Las 4 funciones fueron programadas de la misma forma:

```
void FuncionVentana(float *f, int size){
int n;
float soporte, soporte2;

for( n=0; n< size; n++)
{
operacion que calcula el dato n
*(f + n) = soporte2;
}
}
```

Las funciones que calculan las ventanas difieren en la operación que contienen para obtener el dato *n*. A continuación se muestran las instrucciones en lenguaje *C*, para la obtención del dato *n*, en las tres funciones.

- Función Hanning  
`soporte2 = 0.5*( 1 - cos( 2*PI*n/(size -1) ) );`

- Función Hamming  
`soporte2 = 0.54 - 0.46*cos( 2*PI*n/(size -1) ) ;`
- Función Blackman  
`soporte2 = 0.42 - 0.5*cos( 2*PI*n/(size -1) ) ;`  
`soporte2 = soporte2+ 0.08*cos( 4*PI*n/(size-1) );`

La función que calcula la ventana rectangular no realiza ningún cálculo, solamente iguala a uno todos los valores de la secuencia que almacena la ventana rectangular.

## 2.3 Interpolación

El algoritmo para calcular la transformada de Fourier (TF) de una secuencia de  $N$  datos requiere que  $N$  sea una potencia de dos. Por lo tanto es necesario ajustar el número de datos de la secuencia si ésta no cumple con el requerimiento. Pero con la condición de que los datos de la secuencia ajustada deben tener la misma forma de la señal contenida en los datos de la secuencia no ajustada. Por lo tanto antes de obtener la transformada de Fourier de una señal que se encuentra almacenada en un archivo que contiene una cantidad de datos distinta a una potencia de dos, la cantidad de datos del archivo es aumentada a la próxima potencia de dos. El método utilizado para el incremento de datos es la interpolación. En esta sección se explican los dos tipos de interpolación que utiliza el SATF.

### 2.3.1 Interpolación polinomial

A veces se conoce el valor de una función  $f(x)$  en un conjunto de puntos  $x_1, x_2, \dots, x_N$  (donde  $x_1 < \dots < x_N$ ), pero no tenemos una expresión analítica para  $f(x)$  que nos permita calcular un punto arbitrario. Por ejemplo, que las  $f(x_i)$ 's se obtuvieran de una señal biomédica (por ejemplo un electrocardiograma ó un encefalograma) que no pudiera ser convertida en una función simple. La tarea de estimar  $f(x)$  para un  $x$  arbitrario ubicado entre el punto mas pequeño y el mas grande de los  $x_i$  es llamada interpolación [7].

La interpolación polinomial de grado  $N - 1$ , a través de  $N$  puntos  $y_1 = f(x_1), y_2 = f(x_2), \dots, y_N = f(x_N)$ , está dada explícitamente por la fórmula

$x_1 :$	$y_1 = P_1$			
		$P_{12}$		
$x_2 :$	$y_2 = P_2$		$P_{123}$	
		$P_{23}$		$P_{1234}$
$x_3 :$	$y_3 = P_3$		$P_{234}$	
		$P_{34}$		
$x_4 :$	$y_4 = P_4$			

Tabla 2.1: Tabla de los polinomios de Lagrange que describen el algoritmo de Neville para la interpolación utilizando un polinomio de tercer orden [7].

clásica de Lagrange [7]:

$$\begin{aligned}
 P(x) = & \frac{(x - x_2)(x - x_3) \cdots (x - x_N)}{(x_1 - x_2)(x_1 - x_3) \cdots (x_1 - x_N)} y_1 + \frac{(x - x_1)(x - x_3) \cdots (x - x_N)}{(x_2 - x_1)(x_2 - x_3) \cdots (x_2 - x_N)} y_2 \\
 & + \cdots + \frac{(x - x_2)(x - x_3) \cdots (x - x_{N-1})}{(x_N - x_1)(x_N - x_2) \cdots (x_N - x_{N-1})} y_N \quad (2.17)
 \end{aligned}$$

La fórmula de Lagrange (ecuación 2.17) no proporciona la estimación del error. Por lo tanto se utiliza el algoritmo de Neville para la construcción de la interpolación polinomial.

Con referencia a la tabla 2.1, sea  $P_1$  el valor de  $x$  del único polinomio de grado cero (i.e., una constante), pasando a través del punto  $(x_1, y_1)$ , por lo tanto  $P_1 = y_1$ . Así mismo se define  $P_2, P_3, \dots, P_N$ . Ahora sea  $P_{12}$  el valor de  $x$  del único polinomio de grado uno que pasa a través de los puntos  $(x_1, y_1)$  y  $(x_2, y_2)$ . También para  $P_{23}, P_{34}, P_{(N-1)N}$ . Similarmente para polinomios de orden mas alto, hasta  $P_{123\dots N}$ , que es el valor del único polinomio interpolado a través de  $N$  puntos. Los distintos  $P$ 's forman una *tabla* con sus *antecesores* en la izquierda, permitiendo a un solo descendiente en el extremo derecho. La tabla 2.1 presenta un ejemplo para  $N = 4$ .

El algoritmo de Neville, que se muestra en la ecuación 2.18, es una forma recursiva de colocar los valores dentro de la tabla formada por las  $P$ 's, una columna por tiempo de izquierda a derecha. Es basada en la relación entre un *hijo* y sus 2 *padres* [7].

$$P_{i(i+1)\dots(i+m)} = \frac{(x - x_{i+m})P_{i(i+1)\dots(i+m-1)} + (x_i - x)P_{(i+1)(i+2)\dots(i+m)}}{x_i - x_{i+m}} \quad (2.18)$$

Un mejoramiento de la recurrencia anterior es mantener una mínima diferencia entre los padres e hijos. Entonces se define ( para  $m = 1, 2, \dots, N-1$ ),

$$C_{m,i} \equiv P_{i\dots(i+m)} - P_{i\dots(i+m-1)},$$

$$D_{m,i} \equiv P_{i\dots(i+m)} - P_{(i+1)\dots(i+m)}.$$

Utilizando la ecuación 2.18 en las definiciones anteriores resulta:

$$D_{m+1,i} = \frac{(x_{i+m+1} - x)(C_{m,i+1} - D_{m,i})}{x_i - x_{i+m+1}},$$

$$C_{m+1,i} = \frac{(x_i - x)(C_{m,i+1} - D_{m,i})}{x_i - x_{i+m+1}}.$$

En cada nivel  $m$ , las  $C$ 's y  $D$ 's son las correcciones que calculan la interpolación un orden mas alto. La respuesta final  $P_{1\dots N}$  es igual a la suma de todos los  $y_i$ , mas un conjunto de  $C$ 's y/o  $D$ 's que forman una trayectoria a través las ramas del árbol familiar que llegan hasta el hijo que se encuentra en el extremo derecho [7].

La función que realiza todas las operaciones para la interpolación polinomial [7], utilizando un polinomio de orden  $N-1$ , es:

```
void polint(float xa[], float ya[], int n, float x,
           float *y, float *dy),
```

donde dados los arreglos  $xa[1..n]$  y  $ya[1..n]$ , y dado un valor  $x$ , `polint` regresa el valor  $y$  correspondiente a la abscisa  $x$ .

El comando de Tcl construido para interpolar un archivo de  $N$  datos a una cantidad de  $M$  datos, donde  $M$  es la siguiente potencia de dos mayor que  $N$ , es `poli1`. La función que maneja todo las operaciones que se realizan cuando se usa el comando `poli1` es:

```
void Principal(char *raw, char *intp).
```

modo	descripción	tipo de lectura
0	$f(nT)$	entero
1	$nT f(nT)$	flotante
2	$f(nT)$	flotante
3	$nT f(nT)$	entero

Tabla 2.2: Modos de lectura definidos para el sistema de análisis en tiempo-frecuencia. Donde:  $n = 0, 1, \dots, NP - 1$ , y  $T$  es el periodo de muestreo.  $NP$  es el número de datos que contiene archivo a ser leído.

Realiza los siguientes pasos:

- Lee y coloca en un arreglo global, los datos del archivo que van a ser interpolados. Utiliza la función `Datos` para la lectura del archivo, cuyo nombre se encuentra en la posición de memoria que apunta `raw`.
- Llama a la función `Interpolacion` que se encarga de interpolar los datos almacenados en el arreglo global, y almacenarlos en el archivo cuyo nombre se encuentra en la posición de memoria que apunta `intp`.

### Función Datos

En la función:

```
float Datos( int *res, int *no_datos, char *raw).
```

`raw` representa lo mismo que el argumento `raw` de la función `Principal`, los otros argumentos representan:

`res` Apuntador a la posición donde se colocará la cantidad de datos que tendrá el nuevo archivo de datos interpolados.

`no_datos` Apuntador a la posición donde se colocará la cantidad de datos que contiene el archivo que almacena los datos sin interpolar.

Después de abrir el archivo para lectura, la función `Datos` lee el número de datos  $NP$  que contiene el archivo no interpolado y el modo de lectura (formato de lectura de datos). Se definieron 4 modos de lectura para el sistema de análisis que se describen en la tabla 2.2. Después la función `Datos` calcula el número de datos  $NNP$  que contendrá el archivo de datos interpolados, cumpliendo con la condición de que  $NNP$  es la próxima potencia de

2 mayor que  $NP$ . En el siguiente ciclo `for` se aprecia como se calcula esta cantidad.

```
for( NNP = 1 ; NNP < NP ; NNP = NNP*2)
```

El siguiente paso de la función `Datos` es leer los  $NP$  datos del archivo conforme al modo de lectura del archivo. Los datos se almacenan en dos arreglos o un arreglo global según el modo. Al final de la lectura calcula el incremento en tiempo  $dx$  que tendrá la nueva secuencia de datos interpolados. Se describe a continuación como la función `Datos` calcula estos incrementos.

**modo 0**             $dx = x[NP - 1] / ( NNP - 1.0 )$ .

**modo 1**             $dx = ( x[NP - 1] - x[ 0 ] ) / ( NNP - 1.0 )$ .

**modo 2**             $dx = x[NP - 1] / ( NNP - 1.0 )$ .

**modo 3**             $dx = ( x[NP - 1] - x[ 0 ] ) / ( NNP - 1.0 )$ .

Donde  $x[n]$  es el arreglo que contiene la secuencia de datos en tiempo para los modos de lectura 1 y 3.

### **Función Interpolación**

La definición completa de la función `Interpolación` es:

```
void Interpolacion( int NNP, int NP, float dx, char *intp)
```

Los argumentos de esta función son los mismos que han sido explicados anteriormente en esta sección.

Después de abrir el archivo donde se almacenarán los datos interpolados, la función `Interpolación` imprime en el archivo el número de datos que almacenará ( $NNP$ ). También imprime el primer dato interpolado ya que es idéntico al primer dato de la secuencia no interpolada. Después inicia el proceso de interpolación de datos. En un ciclo de  $NNP - 1$  iteraciones, se va calculando e imprimiendo los datos interpolados. El ciclo es mostrado a continuación:



```

for( j = 1, valx = dx + x[0]; j < NNP ; j++, valx += dx )
  { if( inic < ( NP - ORD - 1 ) )
    { inic = (int)( valx / dx1 );
      if(inic > MPAQ) inic -= MPAQ; }
    polint( x+inic, y+inic, ORD, valx, &salida, &diferencia);
    fprintf( fp, "%e %e\n", valx, salida, inic );
      }
  }

```

En cada ciclo del algoritmo en lenguaje C presentado anteriormente, se llama a la función `polint` para calcular el valor de la función  $f(t)$  en  $t = j \cdot dx$ , donde  $j$  identifica el número de dato estimado de la secuencia interpolada.  $f(t)$  es la función mencionada en la página 14 de esta sección (pero ahora con la variable  $t$ ), cuyos puntos  $(nT, f(nT))$  para  $n = 0, \dots, NP - 1$  se encuentran almacenados en los arreglos que apuntan las variables  $x$  (almacena las abscisas) y  $y$  (almacena las ordenadas). El valor  $j \cdot dx$  es pasado a la función `polint` en la variable `valx`, y el valor estimado  $f(j \cdot dx)$  es regresado en la variable `salida`. El parámetro `ORD` que le es pasado a la función `polint` se refiere al número de datos que se utilizan para interpolar a  $j \cdot dx$ , se utilizó `ORD = 4`. `x+inic` y `y+inic` apuntan al dato inicial de la secuencia de 4 datos no interpolados que se utilizará para estimar a  $f(j \cdot dx)$ . Entonces los datos de la secuencia no interpolada utilizados para estimar a  $f(j \cdot dx)$  son: `x[inic], y[inic], ..., x[inic+3], y[inic+3]`. El ciclo que calcula la secuencia interpolada, ubica al valor a interpolar  $j \cdot dx$  con respecto a `x[inic]` de la siguiente manera:

$$x[inic] \leq j \cdot dx \leq x[inic + 1]. \quad (2.19)$$

Si  $j \cdot dx > x[inic + 1]$  entonces `inic = inic + 1`, y de nuevo se cumple la relación 2.19.

### 2.3.2 Fórmula de Reconstrucción de Shannon

Sea  $x_r(t)$  una señal continua, la fórmula de reconstrucción de Shannon se realiza interpolando los datos de una secuencia  $x[n]$  tal que:

$$x[n] = x_r(nT) \quad \text{para } n = 0, \dots, N - 1, \quad (2.20)$$

a una secuencia llamada  $x_i[m]$  que se representa como:

$$x_i[m] = x_r(mI) \quad \text{para } m = 0, \dots, M - 1, \quad (2.21)$$

donde  $T$  e  $I$  están definidos como:

$$T = \frac{1.0}{N - 1}, \quad (2.22)$$

$$I = \frac{1.0}{M - 1}. \quad (2.23)$$

$N$  corresponde al número de datos de la secuencia no interpolada (secuencia de datos crudos), y  $M$  a la cantidad de datos de la secuencia interpolada ( $M > N$ ).

La ecuación 2.24 muestra el cálculo del dato  $m$  de la secuencia interpolada, por medio de la fórmula de reconstrucción de Shannon, utilizando la secuencia  $x[n]$ .

$$x_i[m] = \sum_{n=-n_d}^{n_d} x[n] \frac{\text{sen}(\pi(mI - nT)/T)}{\pi(mI - nT)} \quad (2.24)$$

La ecuación 2.24 muestra que  $2*n_d+1$  será el número de datos utilizados de la secuencia de datos crudos, para obtener el dato  $m$  de la secuencia interpolada.

La función que obtiene el dato  $m$  de la secuencia interpolada utilizando la fórmula de reconstrucción de Shannon es:

```
void pasabaja( float periodo[ ], float *ym, int m, int n0)
```

Los argumentos de la función representan:

**periodo** Arreglo que contiene el valor del periodo  $T$  de la secuencia de datos no interpolada y el periodo  $I$  de la secuencia interpolada.

**ym** Puntero a la posición donde la función regresará el dato  $m$  estimado.

**m** Número de dato a estimar perteneciente a la secuencia no interpolada.

**n0** El número de dato, perteneciente a la secuencia no interpolada  $x[n]$ , que será el dato central ( $n_d = 0$ ) en la fórmula de Shannon que se muestra en la ecuación 2.24

El valor  $n_d$  perteneciente a la sumatoria de la ecuación 2.24 es obtenido por la función **pasabaja** a través de una variable global.

El comando de Tcl creado para interpolar un archivo de datos utilizando la fórmula reconstrucción de Shannon es **fpb**. Su sintaxis es la siguiente:

```
fpb 'archivo1 archivo2'
```

donde **archivo1** es el nombre del archivo que contiene la secuencia de datos no interpolada, y **archivo2** es el nombre del archivo donde se almacenará la secuencia de datos interpolados que será calculada.

### **Función InterFPB**

La función que maneja el proceso que realiza el comando **fpb** es:

```
void InterFPB( char *raw, char *intp )
```

Sus argumentos representan lo mismo que los argumentos de la función **Principal** explicada en la página 16.

La función **InterFPB** primero da lectura a los datos que van a ser interpolados llamando a la función **Datos** explicada en la sección 2.3.1. El archivo cuyo nombre se encuentra en la posición que apunta **intp** (argumento de la función **InterFPB**), es abierto para almacenar los datos interpolados. Enseguida, la función **InterFPB** imprime en el archivo abierto el número de datos  $M$  que tendrá la secuencia interpolada y almacena el primer dato de la secuencia interpolada, ya que es idéntico al primer dato de la secuencia no interpolada. Después efectúa el proceso de interpolación de datos en tres ciclos, que suman un total de  $M - 1$  iteraciones. En cada iteración de los tres ciclos se llama a la función **pasabaja** (explicada anteriormente) para estimar un dato de la secuencia interpolada. Los ciclos se diferencian en la forma de cálculo del argumento **n0** de la función **pasabaja**. A continuación se explican los tres ciclos.

#### *Primer Ciclo*

En el primer ciclo el argumento **n0** de la función **pasabaja** es constante e igual a 4. Se calculan todos los datos de la secuencia interpolada que están en el intervalo  $0 < t < n_d * T$ , donde  $n_d = 4$  (**nodatos**). Por lo tanto los datos estimados en este ciclo son:

$$\mathbf{x}_i[1], \dots, \mathbf{x}_i[\text{inicio}],$$

donde

$$\text{inicio} = (\text{int})n_d * I/T.$$

### Segundo Ciclo

En el segundo ciclo el argumento  $n_0$  de la función `InterFPB` va cambiando con respecto al dato  $m$  que va a ser estimado, de la siguiente manera:

$$n_0 = (\text{int})(m * I/T).$$

Los datos de la secuencia interpolada estimados se encuentran dentro del intervalo  $n_d * T \leq t < (N - n_d) * T$ , éstos son:

$$x_i[\text{inicio} + 1], \dots, x_i[\text{fin}],$$

donde

$$\text{fin} = M - \text{inicio} - 1$$

### Tercer Ciclo

En este ciclo, el valor de  $n_0$  es constante e igual a  $N - n_d - 1$ . Se estiman los datos de la secuencia interpolada que se encuentran en el intervalo  $(N - n_d) * T \leq t < (N - 1) * T$ , que son:

$$x_i[\text{fin}], \dots, x_i[M - 1]$$

Ahora se muestra el algoritmo de cómputo de la función `InterFPB` para calcular la secuencia interpolada de datos:

```
for( m = 1; m < inicio; m++)
  { pasabaja( periodo, &salida, m, NoDatos);
    valx = valx + dx;
    fprintf( fp, "%e %e\n", valx, salida);
  }

for( m = inicio; m <= fin; m++)
  { n0 = (int)( m * I / T);
    pasabaja( periodo, &salida, m, n0);
    valx = valx + dx;
    fprintf( fp, "%e %e \n", valx, salida );
  }

for( m = fin; m < M; m++)
```

```

{ pasabaja( periodo, &salida, m, N - NoDatos -1);
  valx = valx + dx;
  fprintf( fp, "%e %e\n", valx, salida );
}

```

## 2.4 Espectrogramas

Espectrograma es el espectro de la señal vista a través de una ventana de tiempo que se mueve a lo largo del eje  $t$  [9]. La técnica matemática utilizada para generarlo es la transformada de Fourier con ventana, descrita en la página 10 de este Capítulo. La figura 2.5 muestra como la ventana de Hamming se mueve a lo largo del eje horizontal  $t$ .

El incremento de tiempo o número de datos con el que se mueve la ventana de tiempo será definido como el *paso* de la ventana, como lo presenta la figura 2.5. Si tenemos una ventana de tiempo de  $M$  datos que se mueve a un paso de  $d$  datos, entonces el número  $m$  de ventanas con las cuales la señal no estacionaria podrá ser vista en frecuencia es:

$$m = \frac{N - M}{d}, \quad (2.25)$$

donde  $N$  es el número de datos de la secuencia que almacena a la señal no estacionaria  $x[n]$  que se desea analizar.

Cada vez que se traslada la ventana a través del tiempo, se obtiene la transformada de Fourier con ventana de la señal  $x[n]$ . Donde al segmento de la señal no estacionaria que le corresponde multiplicarse por la ventana, es el segmento que será representado en frecuencia. En total son  $m$  segmentos, donde cada segmento es representado por una secuencia de  $M$  datos llamada  $x_p[l]$  para  $p = 0, \dots, m - 1$ .

Para generar el espectrograma de la señal  $x[n]$  es necesario calcular los espectros de todos los segmentos  $x_p[l]$  que se adquieren con el traslado de la ventana de tiempo. Primero se efectúa la multiplicación de todos los segmentos  $x_p[l]$ , por la ventana de tiempo  $w[l]$  (para  $l = 0, \dots, M - 1$ ). El resultado

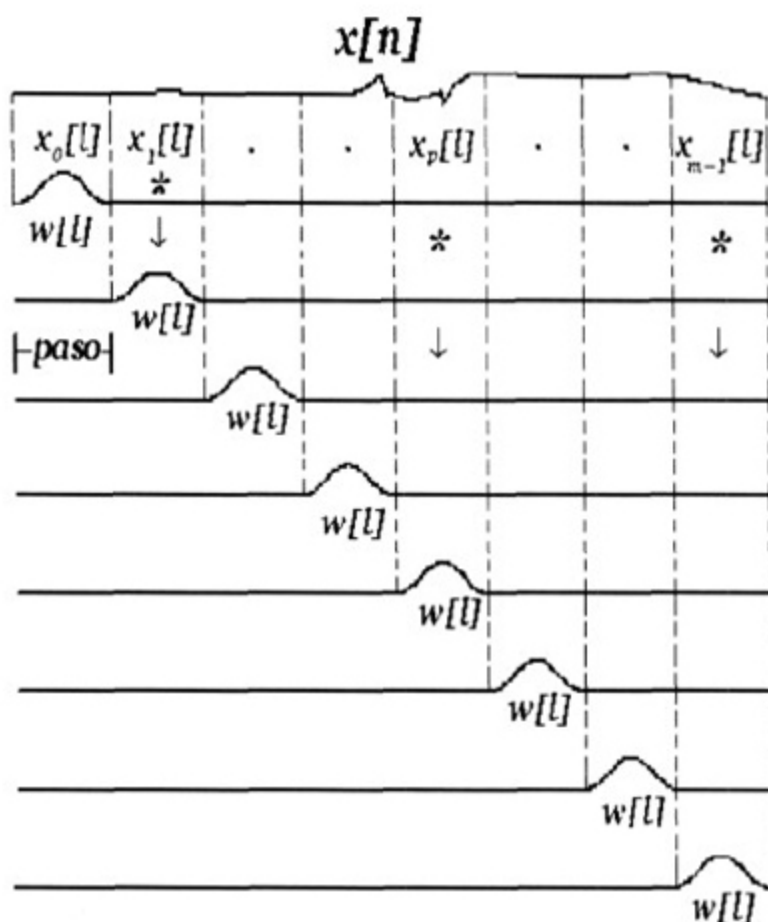


Figura 2.5: Ventana de tiempo (*Hanning*) que se mueve a lo largo del eje  $t$ .

del producto son otras  $m$  secuencias que se describen a continuación:

$$\begin{aligned}
 x_{d0}[n] &= x[n] * w[n] & 0 \leq n \leq M-1, \\
 x_{d1}[n-d] &= x[n] * w[n-d] & d \leq n \leq d+M-1, \\
 &\vdots \\
 x_{d(m-2)}[n-(m-2)d] &= x[n] * w[n-(m-2)d] & (m-2)d \leq n \leq (m-2)d+M-1, \\
 x_{d(m-1)}[n-(m-1)d] &= x[n] * w[n-(m-1)d] & (m-1)d \leq n \leq (m-1)d+M-1,
 \end{aligned}$$

Se obtiene la DFT de  $M$  datos de cada una de las funciones  $x_{dj}[l]$ , para  $j = 0, \dots, m-1$ . Donde la transformada de Fourier de la secuencia de la

señal discreta  $x_{dj}[l]$  es la secuencia compleja  $X_{dj}[k]$ . La figura 2.4 contiene las transformadas de Fourier  $X_{d0}[k], \dots, X_{d(m-1)}[k]$ , de la señal no estacionaria  $x[n]$  multiplicada por ventana de tiempo  $w[l]$  que se encuentran en la figura 2.5.

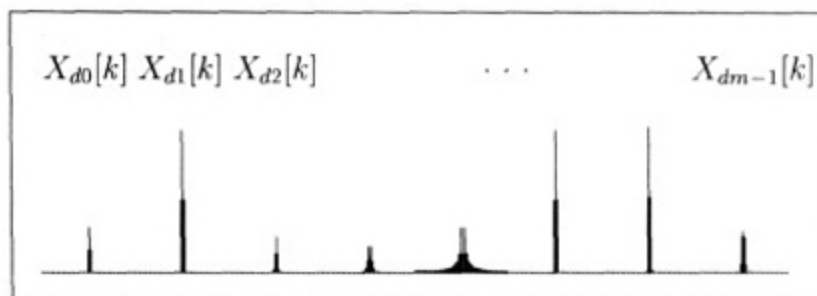


Figura 2.6: Transformadas de Fourier correspondientes a los segmentos que se obtienen con el traslado de la ventana a través del tiempo de la figura 2.5.

Podemos expresar una transformada  $X_{dj}[k]$  (para  $k = 0, \dots, M-1$ ) como una ristra de colores. Donde la magnitud de cada  $X_{dj}(0), X_{dj}(1), \dots, X_{dj}(M-1)$  es representada por un color obtenido de una escala. La elaboración de la escala se realiza con respecto a la frecuencia que tiene la magnitud mas grande  $|X_{dj}(k)|_{max}$  de todas las frecuencias  $X_{dj}(k)$  para  $k = 0, \dots, M-1$  y  $j = 0, \dots, m-1$ . En la figura 2.7 se muestra la correspondencia de la  $|X_{dj}(k)|_{max}$  con la escala de colores que se utilizará para colorear el espectrograma.

La forma de como se construye un espectrograma, utilizando los espectros  $X_{d0}(k), X_{d1}(k), \dots, X_{d(m-1)}(k)$  (ya representados en ristras de colores), es mostrada en la figura 2.8.

Al unir todas las transformadas  $X_{dj}[k]$  (para  $j = 0, \dots, m-1$ ) en la forma que se muestra en la figura 2.8, se termina de construir el espectrograma de la señal no estacionaria  $x[n]$  analizada.

En el espectrograma el tiempo avanza sobre el eje  $t$  variando según el *paso* de la ventana de tiempo. La frecuencia se lee en el eje  $y$ . El rango de frecuencias es de 0 a  $2\pi$ , debido a las propiedades de la DFT.

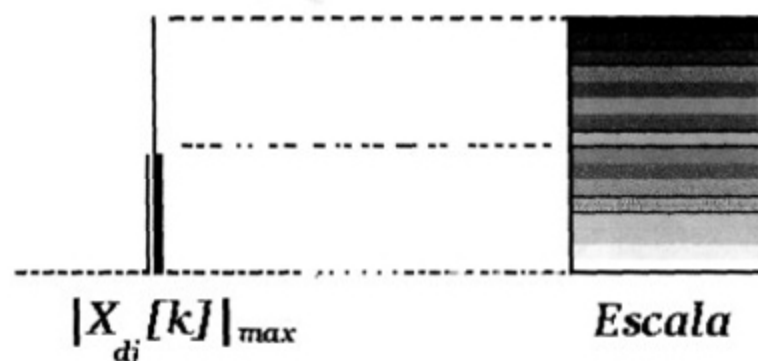


Figura 2.7: Correspondencia entre la transformada de Fourier con magnitud máxima, y la escala de colores que se utilizará para pintar el espectrograma.

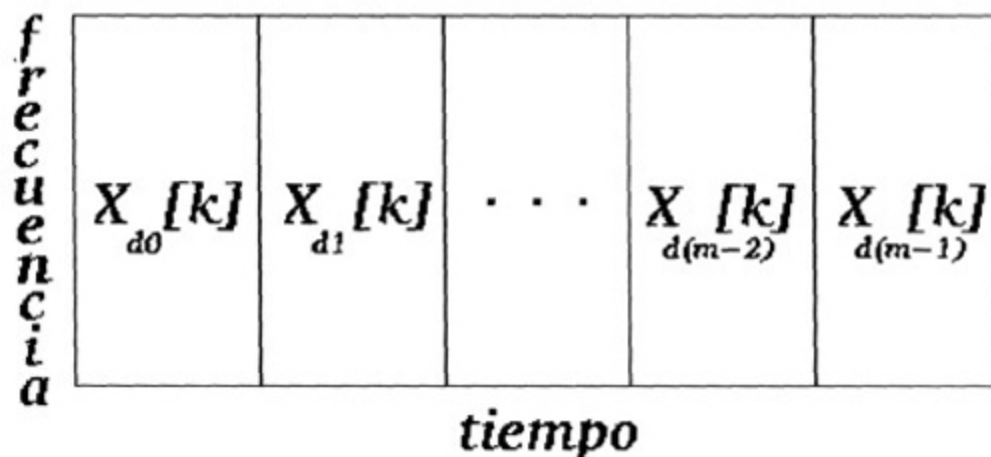


Figura 2.8: Construcción del espectrograma utilizando los  $X_{d_j}(k)$ . En el diagrama, el tiempo corre a través del eje horizontal, y la frecuencia se mide en el eje vertical de 0 a  $2\pi$ .

## 2.5 Algoritmos para generación de espectrogramas

Cuando se quiere analizar una señal en tiempo-frecuencia, el SATF llama al comando de Tcl `fftspc` o al comando de Tcl `fpbsp`. Estos comandos



crean un archivo que contiene el nombre de todos los colores que dibujarán el espectrograma de la señal no estacionaria analizada. Los comandos `fftspc` y `fpbspc` difieren en el tipo de interpolación utilizada para el cálculo de las ventanas.

Los comandos tienen la siguiente sintaxis:

`fftspc` ‘‘lista de parametros’’,

`fpbspc` ‘‘lista de parametros’’,

donde la lista de parámetros de los comandos se muestra en la tabla 2.3.

No.	Parámetro
1	Archivo.
2	Paso de la ventana.
3	Ventana.
4	Tamaño de ventana.
5	Resolución de la DFT.
6	Representación de la TF.
7	Bandera de ajuste.
8	No. de datos.
9	Modo.
10	Valor máximo de la TF.
11	Valor mínimo de la TF.

Tabla 2.3: Lista de parámetros de la sintaxis de los comandos de Tcl `fftspc` y `fpbspc`.

Se describe la sintaxis de los parámetros de los comandos `fftspc` y `fpbspc` que serán utilizados en las funciones que se explican en esta sección:

**Archivo** Conjunto de caracteres que forman el nombre del archivo, que contiene la señal que será analizada en tiempo-frecuencia.

**Paso** Valor entero que es el número de datos con los que va moviendo la ventana de tiempo.

**Ventana** Número entero del 0 al 3 que identifica al tipo de ventana que se desea utilizar para generar el espectrograma.

**Tamaño de ventana** Valor entero que representa el tamaño de la ventana como porcentaje de la señal completa no estacionaria que se desea analizar.

**Resolución (res)** Valor entero que representa el tamaño solicitado en número de datos de la transformada de Fourier discreta.

**Representación de la TF** Número entero del 0 al 2 que identifica a la forma en que va ser representado un número complejo de la TF. Las opciones son magnitud, magnitud en decibels y fase.

**No. de datos** La cantidad de datos que contiene el archivo que almacena la señal que se desea analizar.

Las funciones que manejan el proceso general de los comandos `fftspec` y `fpbspec` respectivamente son:

```
void Espectro(float *xintp, float *yintp, int inf[])  
void Espectro_FPB(float *xintp, float *yintp, int inf[])
```

La función `Espectro` utiliza la interpolación polinomial y la función `Espectro_FPB` la fórmula de reconstrucción de Shannon, sus argumentos representan:

`xintp` puntero al arreglo de flotantes que contiene el valor de las abscisas de la señal no estacionaria que se desea analizar.

`yintp` puntero al arreglo de flotantes que contienen el valor de las ordenadas de la señal no estacionaria que se desea analizar.

`inf` arreglo de datos enteros que contiene información característica del espectrograma a generar.

Nota: Al hablar de la función `Espectro` ( o del comando `fftspec`) también me estaré refiriendo a la función `Espectro_fpb` ( o al comando `fpbspec`) al menos que se mencione alguna diferencia.

Los pasos principales que realiza la función `Espectro` son:

- Lectura del archivo de datos que almacena la señal que se desea analizar.
- Calculo de la ventana de tiempo que será utiliza para generar el espectrograma.

- Búsqueda del valor de la frecuencia máximo en la representación de la magnitud o la magnitud en decibeles de la TF. Con el objetivo de calcular la escala utilizada para representar en colores cada una de las frecuencias requeridas en la generación del espectrograma. Este paso es explicado en la sección 2.5.2.
- Cálculo de los espectros requeridos al mover la ventana a través del tiempo.

### Lectura de datos

El primer paso que realiza el comando `fftspc`, es leer el archivo de datos que contiene la señal no estacionaria que se desea analizar en tiempo-frecuencia. Utiliza a la función:

```
Lec_Datos( float *xintp, float *yintp, char *arch ),
```

donde los argumentos `xintp` y `yintp` representan lo mismo que los argumentos `xintp` y `yintp` de la función `Espectro`. Y el argumento de la función `Datos` que falta explicar es:

`arch` puntero al arreglo de caracteres que contiene el nombre del archivo que almacena los datos de la señal que se desea analizar.

### Cálculo de la ventana de tiempo

El siguiente paso que realiza la función `Espectro` es calcular y almacenar en el arreglo `vent[n]`, los valores de la ventana que será utilizada para generar el espectrograma. Según el tipo de ventana seleccionada, se llama a una de las 4 funciones descritas en las sección 2.2.1 que calculan los valores de la ventana escogida. El tamaño en número de datos de la ventana será igual al valor de la *resolución* `res` de la DFT solicitada. Se presenta la parte del programa que realiza este paso:

```
switch( select)
{ case 0 : Rect(vent,res);
  break;
  case 1 : Hamming(vent,res);
  break;
  case 2 : Hanning(vent,res);
  break;
  case 3 : Blackman(vent,res);
  break;
}
```

### 2.5.1 Cálculo de los espectros

Después de haber calculado la ventana de tiempo, la función `Espectro` calcula el número total de segmentos que se obtienen con el traslado de la ventana a través del tiempo, realizando la siguiente operación:

```
ciclos = ( no_datos - tam_vent ) / paso;
```

donde:

`no_datos` es el número de datos que contiene el archivo que almacena la señal no estacionaria que se desea analizar.

`tam_vent` representa el tamaño en número de datos de la ventana de tiempo solicitada.

`paso` es el número de datos que se traslada la ventana de tiempo.

En un ciclo de `ciclos` iteraciones, `Espectro` obtiene el contenido de frecuencia de los `ciclos` segmentos requeridos para generar el espectrograma. Realiza los siguientes pasos:

- 1 Almacena el segmento de datos de la señal no estacionaria que le corresponde multiplicarse por la ventana de tiempo (ver figura 2.5).
- 2 Interpola el segmento de datos a una secuencia de *res* datos.
- 3 Multiplica el segmento interpolado a *res* datos, por la ventana de tiempo.
- 4 Se obtiene y representa en colores la transformada de Fourier de la secuencia que resulta de la multiplicación del segmento interpolado por la ventana de tiempo.

Se describen a continuación los pasos para calcular los  $m - 1$  espectros, mencionados anteriormente.

#### 1.- Almacenamiento del segmento de la señal

Se almacena en `x[n]` y `y[n]` la porción de datos de la señal no estacionaria que le corresponde multiplicarse por la ventana de tiempo. La señal no estacionaria se encuentran almacenada en los arreglos `xintp[n]`, `yintp[n]`. El arreglo `xintp[n]` almacena la secuencia de los tiempos en los que fue

muestreada la señal no estacionaria. El arreglo `yintp[n]` almacena los valores de la señal no estacionaria ocurridos en los tiempos de muestreo almacenados en `xintp[n]`. Por lo tanto la muestra  $l$  de la señal no estacionaria es representado por el par de datos  $(xintp[l], yintp[l])$ . A continuación se describe la manera de como se almacena el segmento de la señal:

`x[n] = xintp[n + paso*j]` para  $n = 0, 1, \dots, tam\_vent - 1$

`y[n] = yintp[n + paso*j]` para  $n = 0, 1, \dots, tam\_vent - 1$

`j` representa el número de ciclo.

La parte del programa que realiza la función es presentada a continuación:

```
for( i = 0; i < tam_vent; i++)
  { *(x + i) = xintp[i + paso*d] ;
    *(y+i)= yintp[i + paso*d];
  }
```

## 2.- Interpolación del segmento de la señal

Teniendo ya almacenado en `x[n]` y `y[n]` el segmento de la señal que se le va a calcular su espectro , existe el problema de que la ventana de tiempo con la cual se va multiplicar el segmento de la señal es de `res` datos, y el segmento es de `tam_vent` datos. Por lo tanto, es necesario interpolar los datos del segmento de la señal a `res` para que pueda ser multiplicado por la ventana de tiempo.

Para interpolar las secuencias de `tam_vent` datos (`x[n]` y `y[n]`) Espectro utiliza la función:

```
void Interpolacion( float x[], float y[], float xres[],
                  float yres[],int inf[], float dx)
```

mientras que `Espectro_FPB` interpola con la función:

```
InterFPB2(float *y,float *yres,int inf[],float periodo[])
```

Se describirán los argumentos que no han sido mencionados:

`dx` intervalo de tiempo entre 2 muestras consecutivas de la secuencia interpolada.

```
dx = ( tam_vent - 1.0 ) / ( res - 1.0 );
```

`periodo` arreglo que contiene el valor del periodo de la secuencia no interpolada (`periodo[0]`) y el periodo la secuencia interpolada (`periodo[1]`).

Las funciones `Interpolacion` y `InterFPB2` tienen el mismo algoritmo de cálculo que las funciones `Interpolacion` de la sección 2.3.1, y la función `InterFPB` de la sección 2.3.2 respectivamente.

### 3.- Multiplicación por la ventana

Se multiplica `yres[n]` por `vent[n]` donde según la ecuación 2.26:

$$x_{dj}[n] = yres[n] * vent[n]$$

El algoritmo en C que realiza este paso es:

```
for( j = 0; j < res; j++)
    //coloca los valores en arreglos para la FFT
    { imfft[j] = 0;    // ya multiplicados por la ventana
      refft[j] = yres[j]*vent[j];
    }
```

En el algoritmo anterior, podemos ver que las secuencias  $x_{dj}[n]$  son reales ya que:

`refft[n] = Re{xdj[n]}` (parte real de  $x_{dj}[n]$ ), y los datos de `Im{xdj[n]}` (parte imaginaria de  $x_{dj}[n]$ ) que se encuentra en la secuencia `imfft[n]` son igual a 0 para  $n = 0, \dots, tam\_vent - 1$ .

### 4.- Obtención del espectro

Se obtiene y representa en colores la transformada de Fourier de la secuencia `yres[n]*vent[n]`, llamando a la función:

`FFT( float *refft, float *imfft, int *inf, FILE *fp, int pot) ,`  
sus argumentos representan:

`refft` Parte real de la secuencia  $x_{dj}[n]$ .

`imfft` Parte imaginaria de la secuencia  $x_{dj}[n]$ .

`pot` Potencia de 2 tal que:  $res = 2^{pot}$

`fp` Puntero `FILE` al archivo donde se grabarán los colores del espectrograma.

La función `FFT` realiza la siguiente serie de pasos:

- Obtiene la transformada de Fourier de la secuencia de datos con parte real almacenada en `refft[n]` y la parte imaginaria almacenada en `imfft[n]`, llamando a la función `Fft` explicada en la sección 2.2. Como se vio la parte real y la parte imaginaria de la transformada serán regresadas en los arreglos `refft[n]` e `imfft[n]` respectivamente.

- Representa en colores cada una de las frecuencias de la transformada de Fourier obtenida en el paso anterior, explicándose a continuación la manera de como FFT realiza este paso.

### Representación en colores

La intensidad de cada frecuencia  $X_{d_j}(k)$  ( $j = 0, 1, \dots, res-1; k = 0, 1, \dots, m-1$ ) se visualiza por medio de un color. Este color es seleccionado de una escala de colores.

Las tres funciones que obtienen e imprimen en un archivo la representación en colores de una transformada  $X_{d_j}[k]$  son:

```
void Colorfase( float *refft, float *imfft, int inf[], FILE *fp)
void Colormag( float *refft, float *imfft, int inf[], FILE *fp)
void Colorlog( float *refft, float *imfft, int inf[], FILE *fp)
```

Todos los argumentos de las funciones fueron explicados anteriormente. Se describe la tarea principal de cada función:

- Colormag** obtiene los colores cuando se representa la magnitud (mag) de  $X_{d_j}(k)$ .
- Colorlog** obtiene los colores cuando se representa la magnitud en decibeles ( $20 * \log(mag)$ ) de  $X_{d_j}(k)$ .
- Colorfase** obtiene los colores cuando se analiza la señal observando la fase de  $X_{d_j}(k)$ .

Debido a que la DFT es una función par, solamente se determinan los colores para las primeras  $res/2$  frecuencias. La obtención de los colores de una secuencia de frecuencias  $X_{d_j}[k]$  (para  $k = 0, 1, \dots, res/2 - 1$ ) es realizada mediante un ciclo de  $res/2$  repeticiones. En una repetición se realiza:

- El cálculo del número de color (**no\_color**) de una de las frecuencia  $X_{d_j}(k)$  a representar, tal que:

$$no\_color = \frac{\text{valor de } X_{d_j}(k)}{\text{valor de la escala}},$$

donde *valor de la escala* puede ser *escmag* (función **Colormag**), *esclog* (función **Colorlog**), *escflog* (función **Colorfase**), que serán se

descritos en la sección 2.5.2. El *valor de  $X_{d_j}(k)$*  se refiere al valor de la representación de  $X_{d_j}(k)$ . Puede ser magnitud, magnitud en decibels o fase.

- El almacenamiento del color, llamando a la función:  
`void Sel_Color(int no_color, FILE *fp),`  
 que imprime el color, registrado con `no_color` en el archivo abierto con el puntero `FILE fp`.

El almacenamiento de los colores para una frecuencia  $X_{d_j}[k]$ , va de la menor frecuencia  $X_{d_j}(0)$  a la mayor frecuencia  $X_{d_j}(M/2 - 1)$ , empezando por la transformada de  $X_{d_0}[k]$ , hasta la transformada  $X_{d_{(m-1)}}[k]$ .

Los algoritmos en lenguaje C, programados para realizar las representaciones en colores son:

```
for( j = 0; j < res/2; j++) //imprime los valores
{ absoluto = sqrt( refft[j]*refft[j] + imfft[j]*imfft[j] );
  no_color = (int )( absoluto / unimax );
  Sel_Color(no_color, fp);
}
```

para la función `Colormag`,

```
for( j = 0; j < res/2; j++) //imprime los valores
{ absoluto = sqrt( refft[j]*refft[j] + imfft[j]*imfft[j] );
  if( absoluto < 1.0) absoluto = 1.0;
  logabs = 20*log(absoluto);
  no_color = (int )( (logabs - valmin) / unimax );
  Sel_Color(no_color, fp);
}
```

para la función `Colorlog`,

```
for( j = 0; j < res/2; j++) //imprime los valores
{ fase = atan2( imfft[j], refft[j] );
  no_color = (int )( fase / escala );
  Sel_Color(no_color, fp);
}
```



para la función **Colorfase**.

Como se ve en los algoritmos en lenguaje de las funciones **Colormag**, **Colorlog** y **Colorfase** presentados anteriormente, la única adaptación que se realiza para representar las frecuencias con colores, es en la función **Colormag**. Antes de convertir la magnitud de alguna frecuencia  $X_{d_j}(k)$  obtenida a decibelios, se verifica que  $|X_{d_j}(k)| \geq 1$ , si  $|X_{d_j}(k)| < 1$ , entonces  $|X_{d_j}(k)| = 1$ .

Una vez que se tienen las frecuencias representadas en colores de todas las transformadas  $X_{d_j}[k]$  obtenidas, sigue el proceso de pintar el espectrograma. El procedimiento en lenguaje tcl encargado de pintar el espectrograma es:

```
proc Colorea { NoDatos estado }
```

donde **NoDatos** es la cantidad de datos del archivo que contiene la señal no estacionaria que se desea analizar, y **estado** maneja el estado de permanencia, dentro de una ventana gráfica, del espectrograma elaborado.

El procedimiento **Colorea** crea primero el cuadro (canvas) donde el espectrograma será bosquejado. Después abre el archivo donde se escribió el nombre de todos los colores que pintan el diagrama del espectrograma. Para cada frecuencia  $X_{d_j}(k)$  (para  $j = 0, \dots, m-1$ ;  $k = 0, \dots, M/2-1$ ), lee el color y lo pinta sobre el canvas. La figura 2.9 muestra el orden de colocación de cada frecuencia.

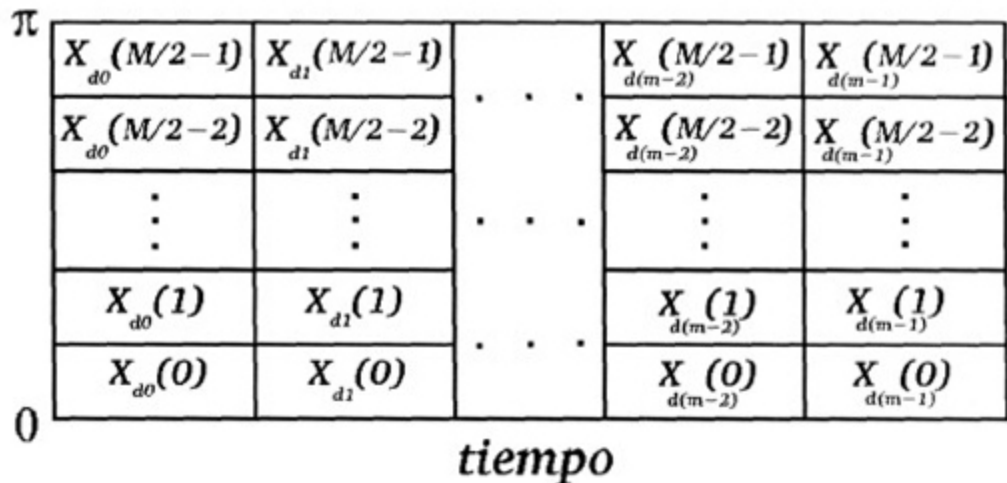


Figura 2.9: Orden de colocación de las frecuencias  $X_{d_j}(k)$  ( para  $j = 0, \dots, m-1$ ,  $k = 0, \dots, M/2-1$ ) representadas por colores, utilizadas para armar el espectrograma.

### 2.5.2 Escalas

Fue convenido que en el SATF se generaran espectrogramas con tres opciones en la representación de los número complejos  $X_{d_j}(k)$  para  $j = 0, \dots, m - 1$  y  $k = 0, \dots, M - 1$ , obtenidos en la tranformada de Fourier . Las opciones son: magnitud, magnitud en decibeles y fase.

Como se ha visto, para la construcción de un espectrograma, es necesario que cada una de las frecuencias obtenidas sea representada por un color. La figura 2.10 muestra la escala que utiliza el SATF para representar un  $X_{d_j}(k)$ :



Figura 2.10: Barra de colores utilizada para representar los espectrogramas.

Los colores en el lado izquierdo de la barra de colores de la figura 2.10 representarán a las frecuencias con valores mas bajos en magnitud, magnitud en decibeles y fase. Los colores del lado derecho de la barra representarán a las frecuencias con valores mas grandes en magnitud, magnitud en decibeles, y fase. El rango de valores de la fase será de 0 a  $2\pi$ .

Las escala para seleccionar el color de la *magnitud* de una frecuencia  $X_{d_j}(k)$ , es obtenida linealmente:

$$escmag = \frac{|X_{d_j}(k)|_{max}}{NoColor}, \quad (2.26)$$

y para la *magnitud en decibeles*

$$esclog = \frac{(20 * \log(|X_{d_j}[k]|))_{max} - (20 * \log(|X_{d_j}[k]|))_{min}}{NoColor}, \quad (2.27)$$

donde:

$(20 * \log(|X_{d_j}(k)|))_{max}$  : Se refiere a la frecuencia con magnitud en decibeles máxima de todas las frecuencia  $X_{d_j}(k)$  calculadas para generar el espectrograma.

$(20 * \log(|X_{d_j}(k)|))_{min}$  : Se refiere a la frecuencia con magnitud en decibeles mínima de todas las frecuencia  $X_{d_j}(k)$  calculadas para generar el espectrograma.

*NoColor* : número de colores que contiene la barra de colores.

Para calcular la escala es necesario encontrar la  $|X_{d_j}(k)|_{max}$  cuando se genera un espectrograma con la magnitud, y es necesario encontrar el  $(20 * \log(|X_{d_j}(k)|))_{max}$  y el  $(20 * \log(|X_{d_j}(k)|))_{min}$  cuando se genera un espectrograma con la magnitud en decibeles. La función:

```
void Obtiene_Uni2( float *xintp, float *yintp,
                  float *vent, int inf[], int pciclos[], float dx)
```

realiza esta operación. Los argumentos de *xintp*, *yintp*, *vent*, *inf* y *dx* son los mismos que se han mencionado en esta sección. *Pciclos* contiene la potencia  $k$  de la resolución de la DFT tal que  $2^k = res$  (almacenado en *pciclos*[0]) y el valor *m* correspondiente al número segmentos de la señal no estacionaria que se utilizarán para generar el espectrograma (almacenado en *pciclos*[1])

La función *Obtiene\_Uni2* calcula el valor de todas las transformadas  $X_{d_j}[k]$  (para  $j = 0, \dots, m - 1$ ) representadas en magnitud o magnitud en decibeles, correspondientes a los *ciclos* espectros requeridos para generar el espectrograma, mencionados en la página 30 de la sección 2.5.1. Por lo tanto la función *Obtiene\_Uni2* realiza un ciclo similar al que realiza la función *Espectro* en la obtención de los *ciclos* espectros, explicados en la sección antes mencionada. Pero con la diferencia de que en cada ciclo iterativo la función *Obtiene\_Uni2* llama a la función *Unidad* mientras que la función *Espectro* llama a la función *FFT*.

La función *Unidad* es la encargada de obtener la  $X_{d_j}(k)$  que tenga la mayor magnitud dentro de una secuencia  $X_{d_j}[k]$  (para  $k = 0, \dots, M - 1$ ). También obtiene la  $X_{d_j}(k)$  que tenga la mayor magnitud en decibeles y la  $X_{d_j}(k)$  que tenga la menor magnitud en decibeles correspondientes a una  $X_{d_j}[k]$ . Se define de la siguiente manera:

```
void Unidad( float *reffft, float *imfft,
             float *max, int *inf, int pot)
```

Los argumentos `refft`, `imfft`, `inf`, `pot` son los mismos que se mencionaron para la función `Espectro_FPB`. El argumento `max` apunta a la posición donde la función `Unidad` tiene acceso al valor de la mayor magnitud (`max[0]`), al valor de la mayor magnitud en decibeles (`max[1]`) y al valor de la menor magnitud en decibeles (`max[2]`). Los valores de `max` pertenecen a alguna transformada  $X_{dj}[k]$  obtenida anteriormente. Antes de llamar por primera vez a la función `Unidad` los argumentos `max[0]`, y `max[1]` son inicializados a 0, y el argumento `max[2]` es inicializado a 10000. La función `Unidad` realiza los siguientes pasos en la obtención de la  $X_{dj}(k)$  con mayor magnitud :

- Obtiene la transformada de Fourier de la secuencia con parte real en `refft[n]` y parte imaginaria en `imfft[n]`, llamando a la función `Fft`.
- Busca la mayor magnitud `magmax` que se encuentra en la secuencia compleja `refft[n],imfft[n]`. Lo compara con la mayor magnitud obtenida anteriormente `max[0]`. Si `magmax > max[0]` entonces `max[0] = magmax`, de lo contrario `max[0]` se queda igual.

El algoritmo en lenguaje C que desarrolla los pasos explicados para obtener la frecuencia con mayor magnitud es:

```
for( j = 0 ; j < res / 2 ; j++)
  { absoluto = sqrt( refft[j]*refft[j] + imfft[j]*imfft[j] );
    if( magmax < absoluto ) magmax = absoluto;
  }
if (maxmax < magmax) maxmax = magmax;
```

Cuando se representa la transformada de Fourier con la *magnitud en decibeles*, la función `Unidad` además de buscar el mayor valor también obtiene el valor de la  $X_{dj}(k)$  que posea la menor magnitud en decibeles. No se deja de efectuar la adaptación realizada para los valores con  $|X_{dj}(k)| < 1.0$ , mencionada en la página 35. El algoritmo en lenguaje C, que lleva a cabo la búsqueda de la mayor y la menor magnitud en decibeles de una secuencia  $X_{dj}[k]$  es:

```

for( j = 0; j < res / 2; j++)
  { absoluto = sqrt( refft[j]*refft[j] + imfft[j]*imfft[j] );
    if( absoluto < 1.0) absoluto = 1.0;
    logabs = 20*log(absoluto);
    if( logmax < logabs    ) logmax = logabs;
    if( logmin > logabs    ) logmin = logabs;
  }
if (lgmxmax < logmax) lgmxmax = logmax;
if (lgmnmin > logmin) lgmnmin = logmin;

```

Los pasos desarrollados por el algoritmo anterior son muy parecidos a los pasos para encontrar la frecuencia con mayor magnitud. Solamente que ahora se busca la mayor magnitud en decibeles y se almacena en la variable `logmax`, y también se busca la menor magnitud en decibeles y se almacena en la variable `logmin`. Y si:

(`lgmxmax < logmax`) entonces `lgmxmax = logmax`,

(`lgmnmin > logmin`) entonces `lgmnmin = logmin`,

donde las variables `lgmxmax` y `lgmnmin` corresponden a las variables `max[1]` y `max[2]` del argumento `max` de la función `Unidad`.

Cuando se generan espectrogramas representando la fase de las  $X_{df}(k)$ , no es necesario obtener una fase máxima. La escala se obtiene de la siguiente manera:

$$escfase = \frac{4\pi}{NoColor} \quad (2.28)$$

## Capítulo 3

# Sistema de Herramientas Computacionales

El sistema de análisis en tiempo-frecuencia de señales no estacionarias (SATF) está desarrollado en lenguaje C con una interfaz gráfica en lenguaje TCL. El SATF contiene procedimientos para manejo de sus archivos de datos y control de sus funciones. Las funciones del SATF son: Interpolación de archivos de datos, obtención de la transformada de Fourier de archivos de datos interpolados y generación de los espectrogramas de archivos de datos interpolados.

El SATF tiene una ventana principal llamada *Análisis en tiempo-frecuencia de señales no estacionarias*, la cual se muestra en la figura 3.1. La ventana tiene una barra de menús con 5 opciones que divide a la ventana en 5 columnas o secciones. Cada opción del menú encabeza la columna que le corresponde. El menú de la primera columna maneja la selección e impresión de archivos de datos. Los menús de las siguientes columnas trabajan con los archivos que se coloquen en su área correspondiente.

### 3.1 Selección de Archivos

El menú que encabeza la primera columna se llama *Archivo*. Sus primeras cuatro son opciones para seleccionar 4 tipos de archivo que son: archivos de datos crudos (extensión *.raw*), archivos de datos interpolados (extensión *.intp*), archivos de datos con transformada de Fourier (TF) (extensión *.fft*) y archivos de datos que contienen los parámetros para generar un espectro-

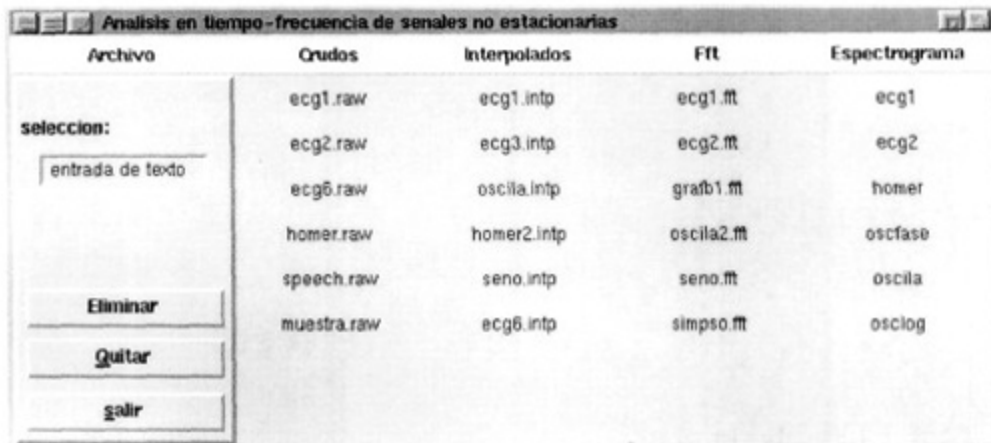


Figura 3.1: Ventana principal del sistema de análisis en tiempo-frecuencia.

grama (extensión .spc). La última opción del menú *Archivo* es *Imprimir*.

Al momento de elegir una opción de tipo de archivo de dato en el menú *Archivo*, el SATF abre otra ventana llamada *Selección de Archivo*. La ventana *Selección de Archivo*, que se muestra en la figura 3.2, lista los archivos existentes del tipo elegido. Al momento de escoger un archivo desde la ventana, este se ubicará en la ventana principal del SATF en la sección correspondiente a su tipo de archivo.

Además del menú *Archivo*, la primera sección contiene una entrada de texto donde se escribe el nombre del archivo o espectrograma que ha sido últimamente seleccionado en la ventana principal del SATF (Nota: últimamente seleccionado en la ventana principal del SATF se abreviará *usel*). La entrada de texto se encuentra ubicada abajo de la etiqueta "seleccion", como lo muestra la figura 3.1. La primera columna cuenta con los botones de "Eliminar", "Quitar" y "Salir". El botón "Eliminar" borra del disco duro el archivo *usel*. "Quitar" elimina el archivo *usel* de la ventana principal del SATF. "Salir" cierra el sistema.

La última opción del menú *Archivo Imprimir*, crea un archivo postscript ( extensión ps) cuando el archivo *usel* es un archivo de datos crudos, un

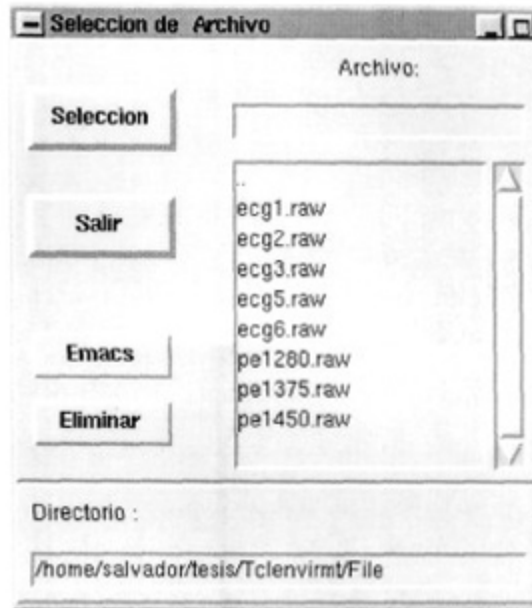


Figura 3.2: Ventana de selección de archivos.

archivo de datos interpolados o un archivo de datos con TF. Si el nombre del archivo *usel* pertenece a un espectrograma, *Imprimir* carga el espectrograma *usel* en el editor de imágenes *xv* para que éste sea editado convenientemente.

## 3.2 Manejo de archivo de datos

En esta sección se explicarán como se ordenan los archivos de datos en la ventana principal del SATF. Se describirá el formato de archivo convenido por el SATF, para archivos: de datos crudos, datos interpolados, y datos con TF. Además se explicará, como se obtienen los archivos de datos interpolados y archivos de datos con TF, y la forma en que pueden ser visualizados.

Las opciones de la barra de menús que se encuentra en la ventana principal del SATF son: *Archivo*, *Crudos*, *Interpolados*, *Fft* y *Espectrograma*. Un menú encabeza una columna de la ventana principal. En la primera columna se encuentra el menú *Archivo* que fue explicado en la sección anterior.



En la segunda columna se albergan los archivos de datos crudos seleccionados desde la ventana de *Selección de Archivo*. *Crudos* es el menú correspondiente a esta columna. Tiene las siguientes opciones: *Datos*, *Graficar* e *Interpolar*. *Datos* muestra, en el editor de texto del sistema, el contenido que tiene el archivo de datos crudos *usel*. *Graficar* gráfica la secuencia de datos que contiene el archivo de datos crudos *usel*. *Interpolar* es un submenú con opciones que interpolan la cantidad de datos de un archivo de datos crudos a la próxima potencia de 2, estas son:

- *Polinomial*: Utiliza el método de interpolación polinomial.
- *Shannon*: Utiliza la fórmula de reconstrucción de Shannon.

Al interpolarse un archivo de datos crudos, el nombre del archivo se ubicará en la tercera columna de la ventana principal del SATF, pero con extensión *intp*. La figura 3.3 muestra una prueba de la interpolación polinomial y la interpolación con la fórmula de Shannon.

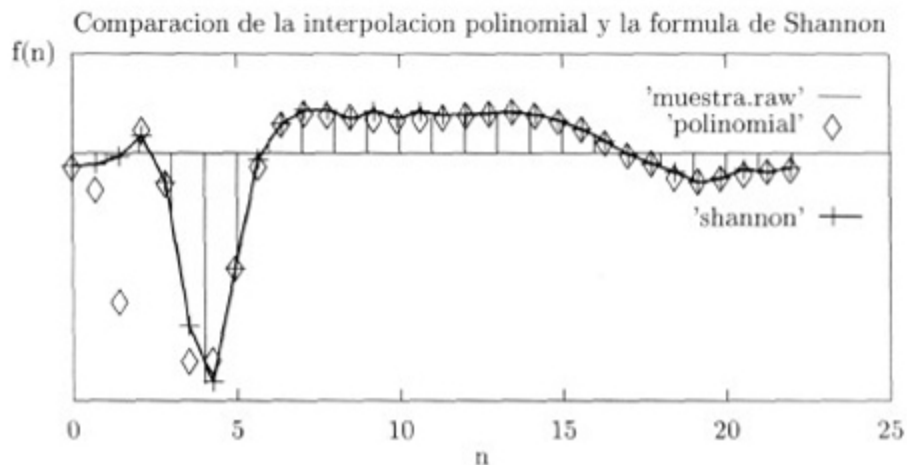


Figura 3.3: Señal no estacionaria, interpolada con la fórmula de interpolación polinomial y con la fórmula de reconstrucción Shannon. La líneas verticales (impulsos) representan a la señal no interpolada. La línea consecutiva que une cruces (+) es la señal interpolada con la fórmula de Shannon. Los rombos representan la señal interpolada con la fórmula polinomial.

La tercera columna está encabezada por el menú *Interpolados* con las opciones: *Datos*, *Graficar*, *Fft*, *entre2*, y *Espectrograma*. En esta columna

se alojan los nombres de los archivos de datos interpolados. Los archivos pueden ser ubicados en la columna de 2 maneras: seleccionando el archivo de datos interpolados desde la ventana de *Selección de Archivo* o al interpolar un archivo de datos crudos. Cada opción del menú *Interpolados* tiene una función que actúa solamente sobre archivos de datos interpolados *usel*. Las opciones son:

**Datos** Visualiza el contenido del archivo utilizando el editor de texto del sistema.

**Grafica** Bosqueja la secuencia de datos del archivo de datos.

**Fft** Obtiene la TF del archivo.

**entre2** Reduce a la mitad los datos que contiene el archivo, a través de la eliminación de los datos impares.

**espectrograma** Llama a otra ventana donde se proporcionan los parámetros para generar el espectrograma.

Como ejemplo ilustrativo de la obtención de la transformada de Fourier de un archivo de datos interpolados (opción *Fft*), la figura 3.4 muestra la gráfica de una señal de electrocardiograma, y su transformada de Fourier se muestra en la figura 3.5 .

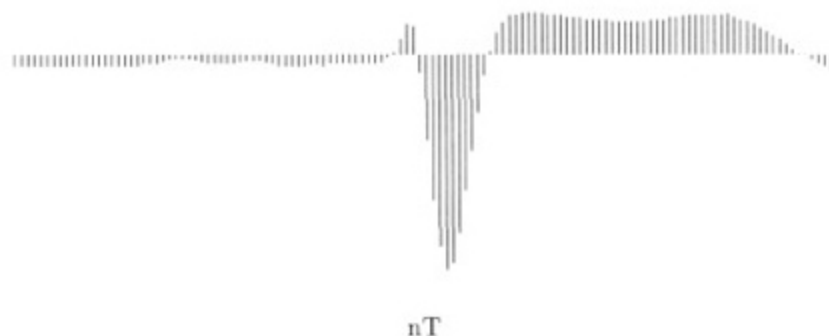


Figura 3.4: Señal de una derivación de un electrocardiograma. Los ejes del diagrama representan amplitud contra tiempo en unidades arbitrarias.

En la cuarta columna de la ventana principal del SATF se colocan los nombres de los archivos de datos con TF. La manera de como pueden ser

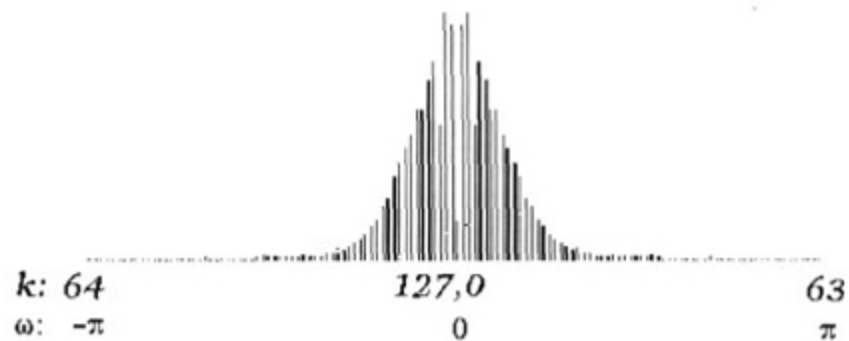


Figura 3.5: Transformada Discreta de Fourier de la señal del electrocardiograma que se encuentra en la figura 3.4.

colocados los archivos es seleccionándolos desde la ventana de *Selección de Archivo*, o calculando la TF de un archivo de datos interpolados. La columna está encabezada por el menú *Fft* que tiene las opciones: *Datos* y *Graficar*. Las opciones del menú *Fft* solamente son para examinar los archivos de datos con TF. La opción *Datos* visualiza el contenido del archivo *usel* utilizando el editor de texto del sistema. La opción *Graficar* es un submenú con tres opciones. Cada opción se refiere a la parte de la TF que se desea graficar. Estas opciones son:

**Re e Im** grafica la parte real e imaginaria de la TF.

**Magnitud** grafica la magnitud (mag) de la TF.

**Fase** grafica la fase de la TF.

**20\*log(mag)** grafica la magnitud en decibeles de la TF.

Los nombres de los espectrogramas generados o seleccionados desde la ventana de *Selección de Archivo*, son almacenados en la quinta columna (última columna). *Espectrograma* es el nombre del menú que encabeza la última columna. Las opciones que tiene *Espectrograma* son:

**Ver** Imprime en una ventana gráfica el espectrograma cuyo nombre ha sido últimamente seleccionado en la ventana principal del SATF.

**Salvar** Guarda los parámetros que generan el espectrograma *usel*.

**Imagen** Carga en el editor de imágenes *xv* el espectrograma *usel*.

El editor de imágenes *xv* es utilizado con el fin de que el espectrograma sea editado convenientemente.

### 3.2.1 Formato de Archivo

El formato de archivo que acepta el SATF para los archivos de datos crudos y los archivos de datos interpolados es:

- Los comentarios se colocan en las primeras líneas del archivo de datos, empiezan con el signo gato (#).
- Después que terminan los comentarios, se escribe el número de datos (`no_datos`) y el modo en que están escritos los datos:  
`#/no_datos modo`
- Por último se escriben los datos numéricos (en total `no_datos`). En un renglón se escribe un valor de la señal, según el `modo` propuesto. Hay 4 modos para leer los datos crudos e interpolados. Se describen en la tabla 2.2 de la sección 2.3.1 en la página 17.

En los archivos de datos con TF, primero se colocan los comentarios con el mismo formato que el de los archivos de datos crudos o archivos de datos interpolados. Después solamente se coloca el número de datos que tendrá el archivo de datos con TF con el siguiente formato `#/no_datos`. Al final se guarda en el archivo los valores de la transformada de Fourier en 6 columnas de datos, donde en un renglón se almacena solamente el valor correspondiente a una frecuencia de la TF. La tabla 3.1 describe el contenido de cada columna.

Columna	Descripción
1	Posición del dato en el eje $x$ .
2	Parte real de la TF.
3	Parte imaginaria de la TF.
4	Magnitud de la TF.
5	Fase de la TF.
6	Magnitud en decibeles de la TF.

Tabla 3.1: Contenido de las columnas de datos en los archivos de datos con transformada de Fourier.

### 3.3 Características para generar un espectrograma.

En esta sección se explicará la forma en que el SATF maneja los espectrogramas: características, parámetros, opciones, y la manera en que son almacenados en disco.

Por convención, solamente a los archivos de datos interpolados se les puede generar un espectrograma. En el menú *Interpolados* que se encuentra en la tercera columna, está la opción llamada *Espectrograma* que inicia el proceso de generar un espectrograma.

Como se vio en la sección 2.5, los comandos de tcl que se utilizan para generar el espectrograma de una señal no estacionaria almacenada en un archivo de datos interpolados (archivo fuente), son `fftspc` y `fftspb`. Y las funciones que manejan el proceso general de los comandos, son las funciones `Espectro` y `Espectro_Fpb` respectivamente. Los comandos utilizados para generar los espectrogramas requieren la lista de parámetros que se muestra en la tabla 2.3. Parámetros que son proporcionados por el usuario a través de la ventana de *Caracterización de espectrograma*.

Al seleccionar la opción *Espectrograma* del menú *interpolados*, el SATF abre la ventana llamada *Caracterización de Espectrograma* que se muestra en la figura 3.6. En esta ventana se proporciona las características deseadas para generar un espectrograma. Las opciones de los parámetros que solicita la ventana para la caracterización de espectrograma se muestran en la tabla

Parámetro	opciones
resolución	baja, media, alta.
Parte de la TF	magnitud, fase, $20 * \log(mag)$ .
Interpolación	Polinomial, Shannon.
Ventana	rect, hanning, hamming, blackman.
Tamaño de Ventana	1 a 100 % de la cantidad de datos que contiene el archivo que almacena la señal que se desea analizar.
Paso	de <i>un</i> dato a la mitad de datos que contiene el archivo que almacena la señal que se desea analizar.

Tabla 3.2: Opciones de los parámetros requeridos en la ventana de caracterización.

3.2. A continuación se describen brevemente estos parámetros:

**resolución** Tamaño de la DFT en número de datos (potencia de 2).

**Parte de la TF** Representación de la transformada de Fourier, requerida para generar el espectrograma. Puede ser magnitud, magnitud en decibels o fase.

**Interpolación** Técnica utilizada para la interpolación de datos.

**Ventana** Nombre de la ventana de tiempo utilizada para generar el espectrograma.

**Tamaño de Ventana** Tamaño de la ventana de tiempo como porcentaje del número de datos del archivo que contiene a la señal no estacionaria que se desea analizar.

**Paso** Cantidad de datos que brincaré la ventana de tiempo, en cada movimiento que vaya a realizar.

La ventana de caracterización contiene tres modos para generar un espectrograma que son: el modo *Derivado*, el modo de *Ajuste* y el modo de *Recortar*. Los modos se describen a continuación.

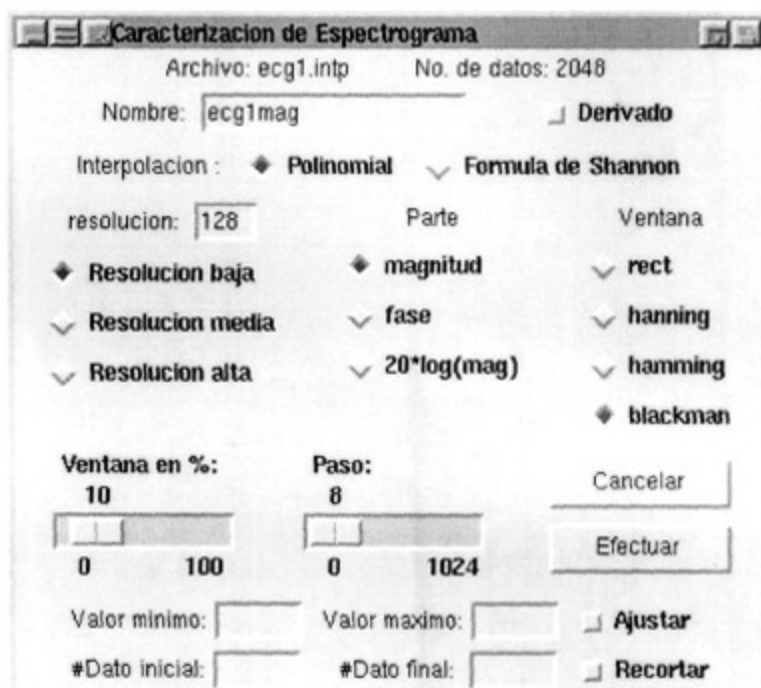


Figura 3.6: Ventana de Caracterización de Espectrograma.

El **modo derivado** genera el espectrograma de la señal derivada. La secuencia de datos de la señal no estacionaria  $x[n]$ , es derivada de la siguiente manera:

$$x[n] = x[n] - x[n - 1] \quad (3.1)$$

El comando de tcl que realiza esta operación es **deriva**. La sintaxis del comando es la siguiente:

```
deriva 'archivo'
```

donde 'archivo' es el nombre del archivo de datos interpolados cuyos valores van a ser derivados.

La función en lenguaje C que maneja el proceso del comando **deriva** es: **Restador(char \*arch)**

donde **arch** es el puntero al arreglo donde se ubica el nombre del archivo que contiene los datos a ser derivados. Los pasos que realiza la función **Restador** son los siguientes:

- Lectura, y almacenamiento en dos arreglos globales, de los datos que almacena el archivo cuyo nombre se ubica en la posición que apunta `arch`.
- Resta los datos según el algoritmo de la ecuación 3.1, y los almacena en el archivo llamado *derivado*.

A continuación se describe el modo para generar un espectrograma llamado *Recortar*:

**Modo recortar.**- Genera el espectrograma de la señal no estacionaria reducida en cantidad de datos. Los datos que son excluidos de la secuencia completa, se encuentran al inicio y al final de la secuencia de datos del archivo analizado. Con la reducción de datos se produce una nueva secuencia de datos (señal reducida) para ser analizada. A través de la ventana de *Caracterización de Espectrograma* se proporcionan el número de dato en la secuencia completa que será el primer dato de la secuencia reducida (dato inicial), y el número de dato en la secuencia completa que será el último dato de la secuencia reducida (dato final). Los datos que formarán a la secuencia reducida estarán en el intervalo cerrado [dato inicial, dato final].

El comando de tcl `tronca` realiza el recorte de datos de un archivo de datos interpolados. Su sintaxis es la siguiente:

```
tronca archivo1 archivo2 modo dat_ini dat_fin
```

Sus parámetros son descritos a continuación:

`archivo1` Nombre del archivo de datos que va a ser recortado.

`archivo2` Nombre del archivo donde se va almacenar la secuencia recortada de datos.

`modo` Indica el modo de como serán leídos los valores `dat_ini` y `dat_fin`. Si `modo = 0` indica que `dat_ini` y `dat_fin` serán leídos en número de dato (valor entero), y `modo = 1` serán leídos como tiempo (valor flotante).

`dat_ini` Número de dato o tiempo con el que iniciará el archivo recortado en datos.

`dat_fin` Número de dato o tiempo con el que finalizará el archivo recortado en datos.



La función en lenguaje C que realiza el proceso del comando **tronca** es:  
`Recortar( double datext[], char *intp, char *tronca ),`  
 sus argumentos significan:

**datext** Contiene los valores extremos con los que iniciará y terminará el archivo recortado en número de datos.

**intp** Puntero al arreglo de caracteres que forman el nombre del archivo que se desea recortar en número de datos.

**tronca** Puntero al arreglo de caracteres que forman el nombre del archivo que almacenará la secuencia recortada en número de datos.

Efectúa los siguientes pasos:

- Lectura de datos del archivo a ser recortado, utiliza la función **Lec\_Datos**.
- Almacenamiento de la nueva secuencia recortada de datos en el archivo cuyo nombre se ubica en el lugar donde apunta **tronca**.

El algoritmo programado para realizar el recorte de datos es mostrado a continuación:

```

inicio = (int) datext[0];
fin     = (int) datext[1];

for ( i = inicio - 1 ; i < fin ; i++)
    fprintf( fp, "%e %e\n", xintp[i] , yintp[ i ] );

```

Y para terminar se describe el último modo para genera espectrogramas que es:

**Modo de ajuste.**- Produce el espectro de la secuencia de datos con valores ajustados en la magnitud de la transformada de Fourier o la magnitud en decibels de la transformada de Fourier. Los ajustes son proporcionados en la *Ventana de Caracterización*. Para ajustar la TF se proporciona un valor mínimo, y un valor máximo. El programa ajustará las frecuencias  $X(k)$  de la TF de la siguiente manera:

Si  $|X(k)| > \text{valor máximo}$  entonces  $|X(k)| = \text{valor máximo}$ .  
 igualmente para el  $20 * \log |X(k)|$  (magnitud en decibels).

Si  $20 * \log |X(k)| < \text{valor mínimo}$  entonces  $20 * \log |X(k)| = \text{valor mínimo}$ .  
para la magnitud de  $X(k)$  el valor mínimo siempre será cero.

Cuando se solicita un espectrograma con el valor máximo y/o mínimo ajustado en la transformada de Fourier, se activa la *bandera de ajuste* con valor igual a 1. Esta bandera es el sexto parámetro que se le proporciona al comando `fftspc`, como lo muestra la tabla 2.3 en la página 27. Los valores extremos máximo y mínimo le son pasados al comando mencionado en los parámetros noveno y décimo de la lista de parámetros mostrada en la misma tabla.

La función `Espectro` o la función `Espectro_fpb`, detectan por medio de la *bandera de ajuste*, el requerimiento de que el espectrograma sea generado con los valores máximos y mínimos ajustados. Cuando se requiere un espectrograma con el *modo de ajuste*, la función `Espectro` se salta el paso de llamar a la función `Obtiene_Uni2` (explicada en la sección 2.5.2 en la página 37) que obtiene la frecuencia  $X_{df}(k)$  con *magnitud* o *magnitud en decibeles* máxima, y procede a calcular que forman el espectrograma. Al momento de leer el valor de la *magnitud* o la *magnitud en decibeles* de cada frecuencia obtenida para generar el espectrograma, se verifica que no sobrepase los valores máximo y mínimo requeridos. El algoritmo que desarrolla esta acción se encuentra dentro de las funciones: `colormag`, y `colorlog`.

El algoritmo de la función `colormag` se presenta a continuación:

```
if( ajuste == 1)
{
  valmin = (float ) inf[8];
  unimax = (float ) ( (valmax - valmin + 1.0) / COLOR );
  //selecciona e imprime color
  for( j = 0; j < res/2; j++) //imprime los valores
    { absoluto = sqrt( refft[j]*refft[j] + imfft[j]*imfft[j] );
      if( absoluto > valmax ) absoluto = valmax;
      if( absoluto < valmin ) absoluto = valmin;
      no_color = (int)(absoluto/unimax);
      Sel_Color(no_color, fp);
    }
}
```

y el algoritmo de la función `colorlog` es muy parecido a este anterior, pero en vez de verificar que el valor `absoluto` no sobrepase a `valmax`, verifica al valor  $20 * \log(\text{absoluto})$ .

**Acoplamiento del tamaño de ventana.**

Cuando se proporciona un tamaño de ventana que representa una cantidad de datos (*tam\_vent*) mayor que la resolución deseada (*res*), o sea

$$tam\_vent > res,$$

el comando `fftspc` llama a la función `Fix_res` para obtener la nueva resolución *res'* que es la próxima potencia de 2 mayor que *tam\_vent*.

Con la nueva resolución, la función `fftspc` realiza una serie de cambios para obtener la representación en colores de los espectros requeridos para generar el espectrograma. En primer lugar genera la ventana de tiempo con *res'* datos en lugar de *res* datos. La parte de la señal no estacionaria (de *tam\_vent* datos) que le corresponde multiplicarse por la ventana de tiempo es interpolada a *res'* datos. El resultado de la multiplicación entre la señal no estacionaria y la ventana de tiempo (secuencia producto), que corresponde a la secuencia  $x_{d_j}[n]$  de la ecuación 2.26, es una secuencia de *res'* datos. Las DFT solicitada es de *res* datos, por lo tanto la función `fftspc` ajusta la secuencia  $x_{d_j}[n]$  a *res* datos llamando a la función

```
void Ajustador(float *realnew, float *real, float *im, int inf[])
```

obtiene una nueva secuencia de *res* datos  $x_{n_j}[m]$  a partir de la reducción de datos de la secuencia  $x_{d_j}[n]$  antes mencionada. La parte real de la nueva secuencia  $x_{n_j}[m]$  de *res* datos es almacenada en arreglo que apunta `realnew`, y la parte imaginaria en el arreglo que apunta `im`.

La manera que `Ajustador` reduce los datos es:

- Busca la relación *mult* entre *res* y *res'* de manera que:

$$mult = \frac{res'}{res}.$$

- En un ciclo iterativo de *res* iteraciones, se almacenan los valores en la secuencia de *res* datos de la siguiente forma:

$$x_{n_j}[m] = x_{d_j}[m * mult] \text{ para } m = 0, \dots, res - 1.$$

Los valores de la secuencia que corresponden a `im` se igualan a 0 debido a que las funciones que deseamos analizar son reales.

Archivo fuente (*.intp). Tipo de interpolación. Cantidad de datos que contiene el archivo fuente. Ventana de tiempo utilizada. Tamaño de la ventana ( en %). Paso (en # de datos). Resolución. Parte de la TF. Valor máximo y mínimo ( $X(k)$ ).
--

Tabla 3.3: Parámetros, que se muestran en las ventanas gráficas donde los espectrogramas son presentados .

### Presentación y almacenamiento de los espectrogramas

Al generarse un espectrograma este es presentado dentro de una ventana gráfica. La ventana es llamada *Espectrograma: nombre*, donde *nombre* identifica al espectrograma bosquejado. La ventana contiene también información del valor de los parámetros listados en la tabla 3.3, utilizados para generar el espectrograma muestra además la escala de colores empleada para colorearlo.

Para salvar un espectrograma en disco, el SATF almacena el valor de los parámetros que lo generaron. Son salvados en archivos con extensión .spc. Se coloca un parámetro por renglón , en el siguiente orden:

**paso, tipo de ventana, resolución, bandera de derivada (der), tamaño de ventana, bandera de parcial (par), bandera de ajuste, nombre del espectrograma y nombre del archivo interpolado analizado.**

La lista de anterior de datos corresponde al formato de archivo de datos con extensión .spc . Se describirán los parámetros que no habían sido mencionados:

**der** Indica si se obtuvo el espectrograma del archivo de datos derivado ( $der = 1$ ) o sin derivar ( $der = 0$ ).

**par** Indica si se obtuvo el espectrograma del archivo de datos troncado ( $par = 1$ ) o sin trincar ( $par = 0$ ).

Como ejemplo ilustrativo se presenta el espectrograma de la figura 3.7.

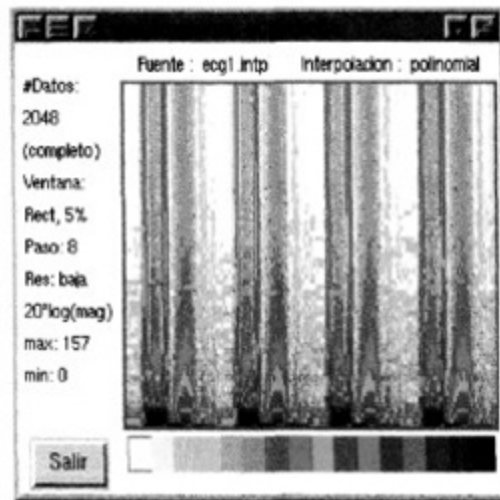


Figura 3.7: Figura que ilustra la manera de como son las ventanas gráficas donde los espectrogramas son presentados.

La columna de datos de la parte izquierda en la figura 3.7 contiene la información de los parámetros que generaron el espectrograma. La etiqueta *(completo)* en la columna indica que el archivo de datos interpolados que contiene a la señal analizada no ha sido reducido en número de datos. En la parte superior de la ventana se etiquetan el archivo fuente y el tipo de interpolación utilizada en la generación del espectrograma. En la parte inferior se encuentra la escala de colores utilizada para pintar el espectrograma y el botón “salir” que destruye a la ventana *Espectrograma: prueba*.

El espectrograma de la figura 3.7 analiza una señal no estacionaria del tipo biomédico: una derivación de un electrocardiograma [12].

# Capítulo 4

## Resultados

El sistema de análisis en tiempo–frecuencia permite manejar diversos parámetros para la generación de espectrogramas. Los parámetros que se listan en la tabla 4.1 producen las diferentes formas para construir el espectrograma de una señal no estacionaria. Los algoritmos y sus programas descritos en los capítulos anteriores se ponen a prueba usando tres tipos señales diferentes. Se analizan los resultados obtenidos en este capítulo en base a las opciones disponibles.

<b>Representación de la TF</b>	<b>Tamaño de la ventana</b>
<b>Ventana</b>	<b>Paso</b>
<b>Resolución</b>	<b>Interpolación</b>

Tabla 4.1: Lista de parámetros para generar un espectrograma.

Los parámetros de la tabla 4.1 se describen a continuación.

**Representación de la TF.** Identifica a la forma de representar gráficamente los valores complejos de la transformada de Fourier. Sus opciones son la representación por magnitud, por magnitud en decibels y por la fase.

**Ventana.** Tipo de ventana de tiempo utilizada en la generación del espectrograma. Sus opciones son la ventana rectangular, la ventana de Hanning, la ventana de Hamming y la ventana de Blackman.

**Resolución.** Tamaño de la DFT en número de datos, que debe ser una potencia de 2.

**Tamaño de ventana.** Tamaño de la ventana de tiempo como porcentaje del número total de datos donde se almacena la señal no estacionaria que se desea analizar.

**Paso.** Número de datos que avanza la ventana en cada movimiento realizado a través del tiempo.

**Interpolación.** Algoritmo matemático utilizado para interpolar los datos. Sus opciones son la interpolación polinomial y la fórmula de reconstrucción de Shannon.

En las secciones 4.1 y 4.2 de este capítulo se examina la manera en que la variación de los parámetros para generar espectrogramas, que se muestran en la tabla 4.1, afectan el diagrama obtenido. Se presentan varios grupos de espectrogramas. Los espectrogramas de un grupo son generados variando el valor de uno de sus parámetros de generación. El valor de los parámetros restantes es el mismo para todos los espectrogramas del grupo.

Además de los parámetros mencionados, el programa cuenta con los dos siguientes *modos para generar espectrogramas*:

**Modo de juste de la TF.** Ajuste del valor máximo en la magnitud de la TF, o ajuste del valor máximo y mínimo en la magnitud en decibeles de la TF.

**Modo derivado.** Espectrograma de la señal derivada.

En la última sección de este capítulo se generan espectrogramas de una señal de voz.

## 4.1 Análisis de la señal $\text{sen}(1/t)$

La primer señal que será analizada en esta sección es  $\text{sen}(1/t)$ . Es bosquejada en la figura 4.1.

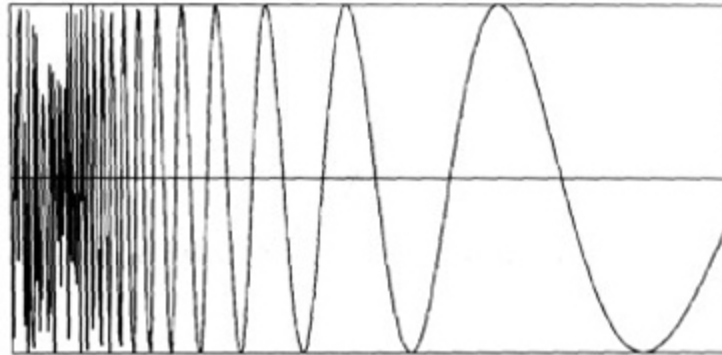


Figura 4.1: Señal  $\text{sen}(1/t)$  muestreada a tiempos de  $t = 0.001, 0.001 + T, \dots, 0.001 + nT, \dots, 0.1032$ , donde  $T = 0.002$  segundos.

Los espectrogramas que se generan en esta sección analizan la manera en que afecta:

- La variación en la representación de la TF.
- La variación en la resolución de la DFT.
- La variación en la interpolación.
- El modo de ajuste de la TF y el modo derivado.

### 4.1.1 Variación en la representación de la TF y el modo derivado

Se generan espectrogramas de la señal  $\text{sen}(1/t)$  correspondientes a la *fase* de la TF, a la *magnitud* de la TF y a la *magnitud en decibeles* de la TF. Además se genera un espectrograma de la señal derivada. Los diagramas obtenidos son presentados en la figura 4.2



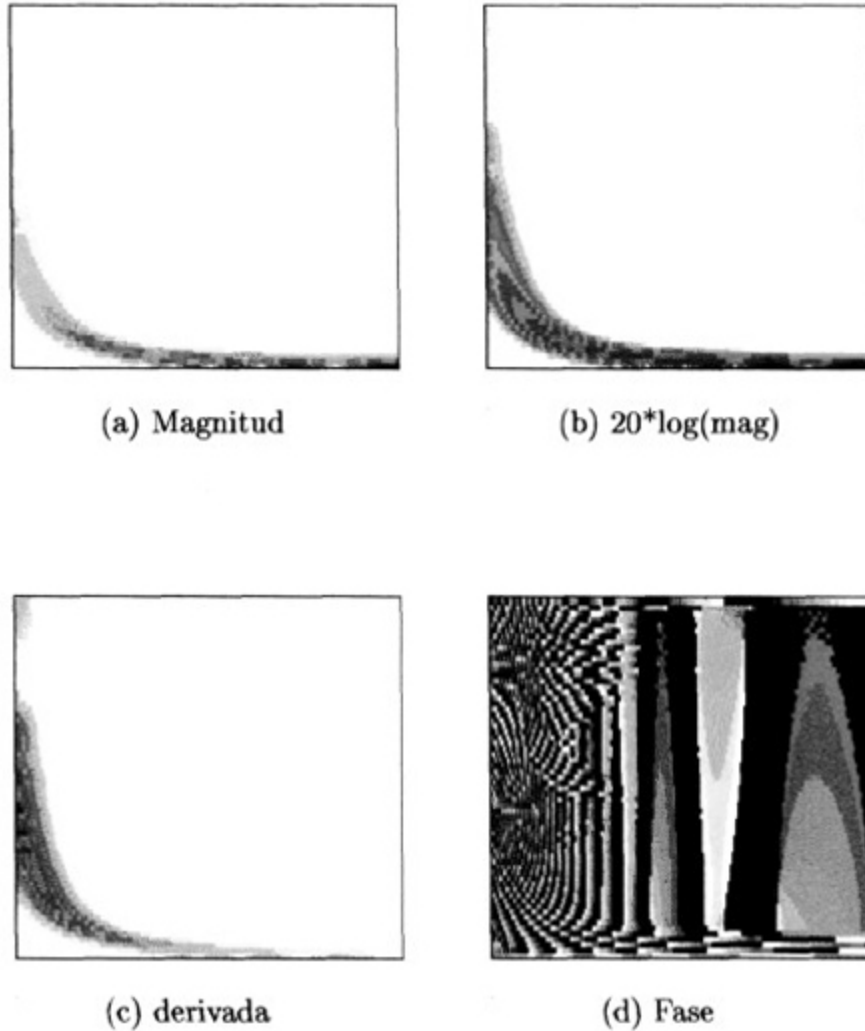


Figura 4.2: Espectrogramas de las señal  $\sin(1/t)$  correspondientes a la magnitud, a la magnitud en decibels y a la fase (diagramas a,b,d respectivamente) de la transformada de Fourier. Además el espectrograma de las señal  $\sin(1/t)$  derivada generado con la magnitud de la TF (diagrama d). Las características comunes de los espectrogramas son: ventana de Blackman, tamaño de ventana = 20% resolución baja, paso = 2, e interpolación polinomial.

### 4.1.2 Variación en la resolución

A continuación se generan cuatro espectrogramas de la señal  $\text{sen}(1/t)$ , utilizando 4 resoluciones para la DFT, estas son 64, 128 (baja), 256 (media), y 512 (alta). La figura 4.3 muestra los diagramas obtenidos.

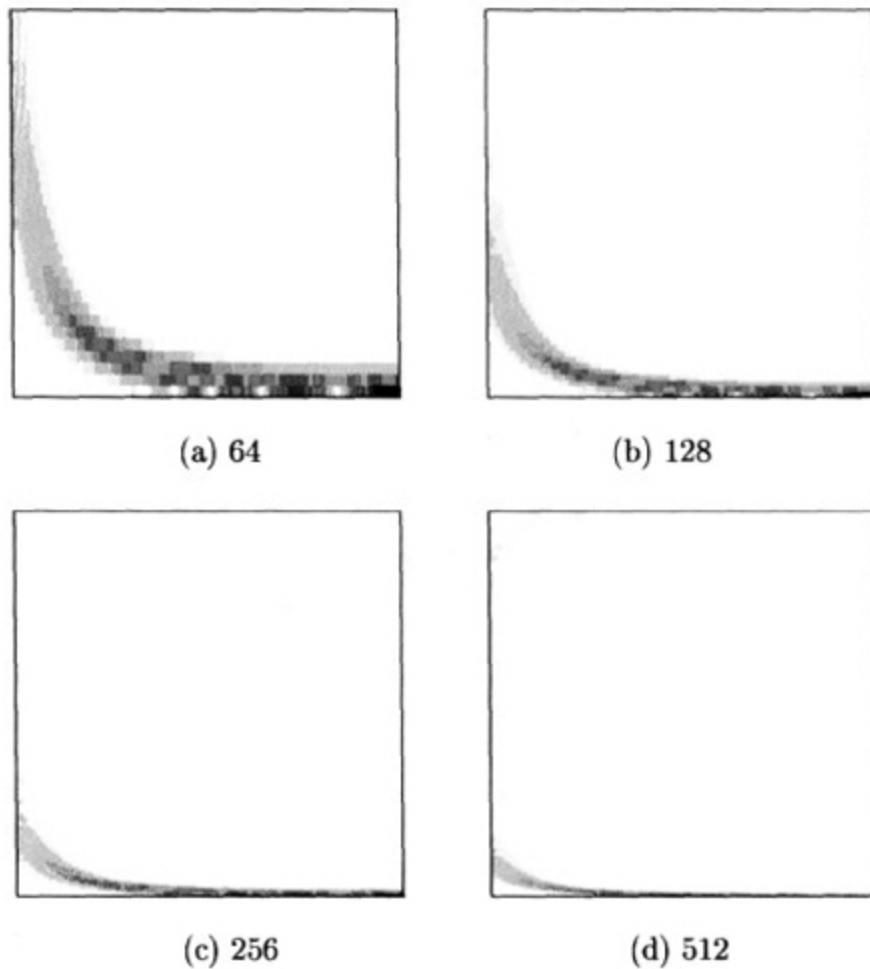


Figura 4.3: Espectrograma de las función  $\text{sen}(1/t)$  utilizando cuatro resoluciones diferentes: 64, 128, 256, y 512 (imágenes a,b,c y d respectivamente). Las características comunes de los espectrogramas son: representación de la magnitud de la TF, ventana de Blackman, tamaño de ventana = 20%, paso = 2, e interpolación polinomial.

### 4.1.3 Modo de ajuste al máximo

El siguiente espectrograma de la señal  $\sin(1/t)$  es generado ajustando el valor máximo en la *magnitud* de la TF. Pero primero es necesario conocer el valor máximo de la *magnitud* del espectrograma que ha sido generado sin ajustar dicho valor. Se selecciona el espectrograma de la figura 4.2 (a) para representar el espectrograma no ajustado. Su *magnitud* máxima es de 78 (*magnitud máxima inicial*). Se genera el espectrograma de la figura 4.4, con las mismas características que el espectrograma de la figura 4.2 (a) pero ajustando la *magnitud máxima inicial* a un 6% de su valor.

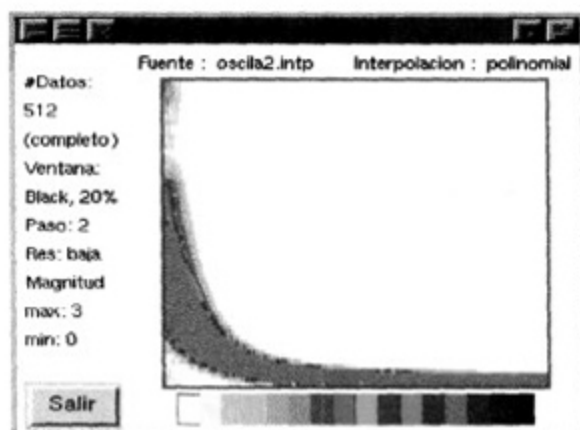


Figura 4.4: Espectrograma generado con el valor máximo de la *magnitud* ajustado a un 6% del valor máximo obtenido en el espectrograma sin ajustar, mostrado en el diagrama (a) de la figura 4.2. El espectrograma fue generado con las mismas características que el espectrograma de la figura 4.2 (a).

### 4.1.4 Variación en la interpolación

Hasta el momento, todos los espectrogramas han sido generados con la interpolación polinomial. Se genera ahora un espectrograma con la interpolación de la fórmula de Shannon (fórmula de reconstrucción de Shannon), mostrado en la figura 4.5. El espectrograma tiene las mismas características que el espectrograma de la figura 4.2 (b).

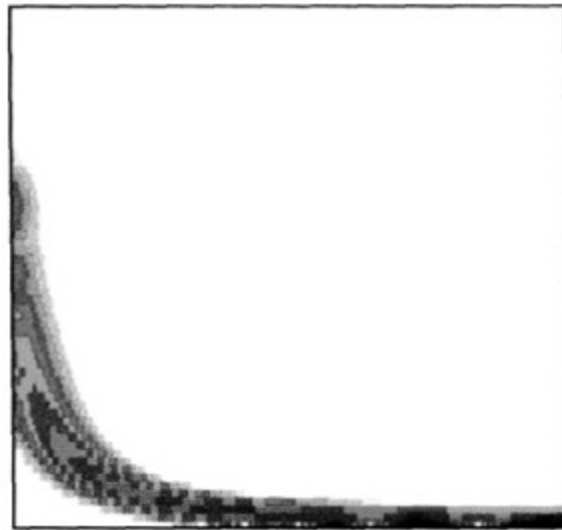


Figura 4.5: Espectrograma de la señal  $\text{sen}(1/t)$  generado con la opción de la interpolación con la fórmula de reconstrucción de Shannon. Las características del espectrograma son las mismas que tiene el espectrograma de la figura 4.2 (b), excepto la interpolación.

## 4.2 Espectrogramas de electrocardiogramas

Se analizarán en tiempo-frecuencia señales de electrocardiogramas. La figura 4.6 muestra la señal de electrocardiograma 1 (ECG 1) [12] que será analizada con algunas opciones del SATF .



Figura 4.6: Electrocardiograma 1 (amplitud vs tiempo en unidades arbitrarias)

En las siguientes 2 subsecciones se generan espectrogramas de la señal ECG 1 que muestran la manera de como afecta en el diagrama obtenido:

- La variación del tipo de ventana.
- La variación del paso.
- La variación del tamaño de ventana.

En la última subsección se muestran los espectrogramas de otras dos señales de electrocardiogramas.

### 4.2.1 Variación del tipo de ventana

En el primer grupo de espectrogramas que analizan la señal del ECG 1, se generan diagramas utilizando las cuatro ventanas de tiempo que contiene el SATF, estas son: la ventana rectangular, la ventana de Hanning, la ventana de Hamming y la ventana de Blackman. Los espectrogramas obtenidos se muestran en la figura 4.7.

Se observa en la figura 4.7, que los diagramas con las ventanas de Hanning, Hamming y Blackman son muy parecidos entre sí. El espectrograma con la ventana rectangular, que se encuentra en el diagrama (a), es el que difiere un poco con respecto a los otros bosquejos.

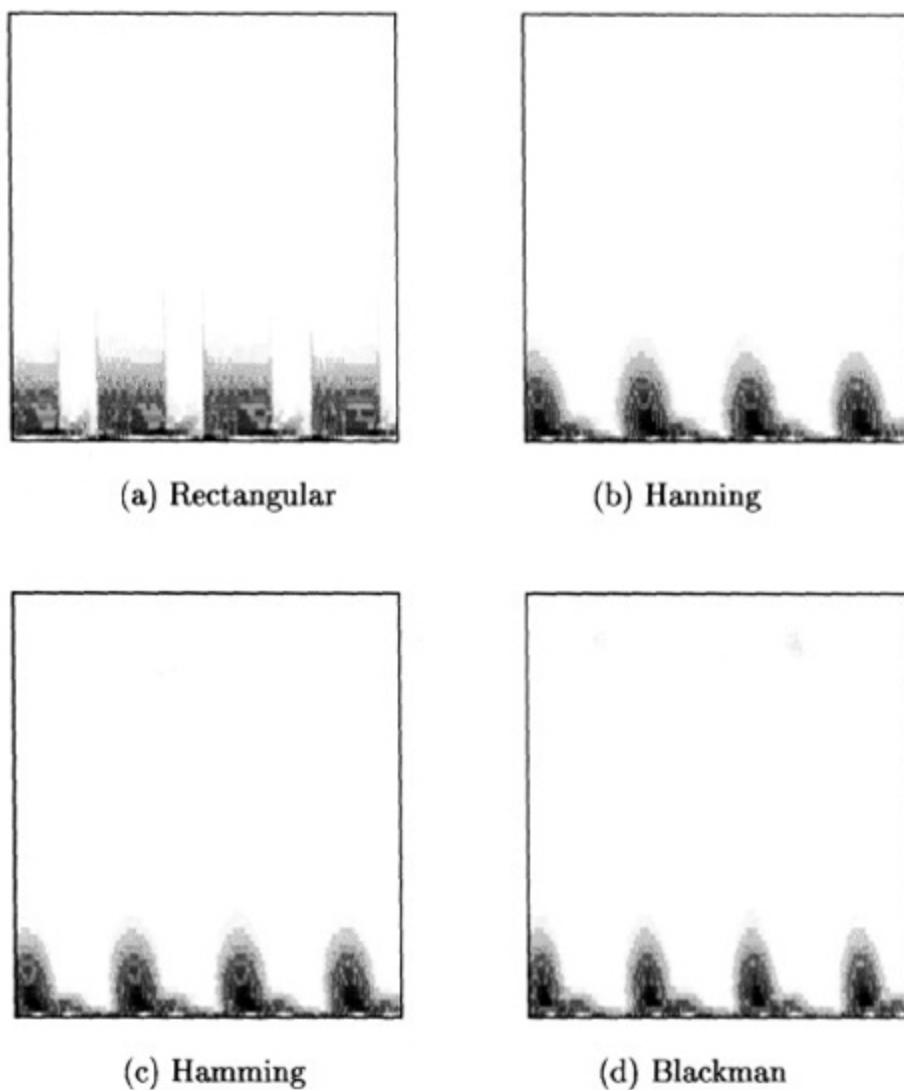
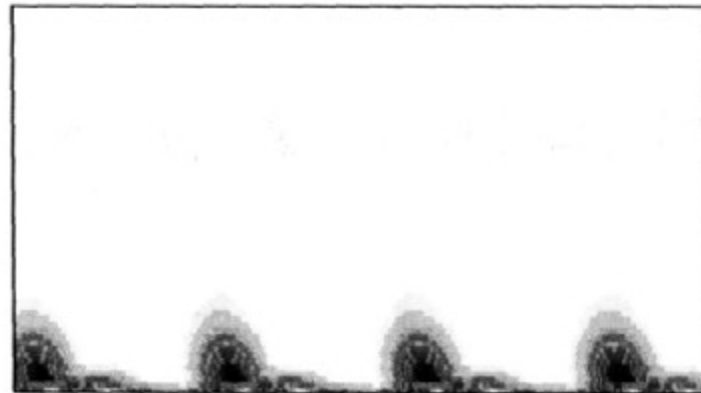


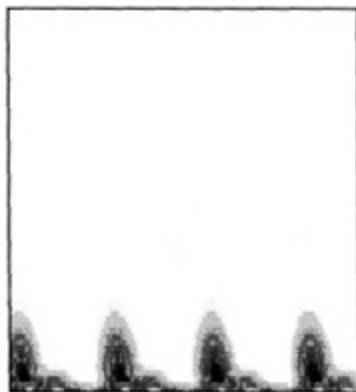
Figura 4.7: Espectrogramas que analizan la señal del electrocardiograma 1 utilizando la 4 ventanas de tiempo del SATF: rectangular, Hanning, Hamming y Blackman. Las características comunes de los espectrogramas son: representación de la magnitud de la TF, resolución = 128, tamaño de ventana = 15 %, paso = 10 e interpolación polinomial.

### 4.2.2 Variación del tamaño de ventana y el paso

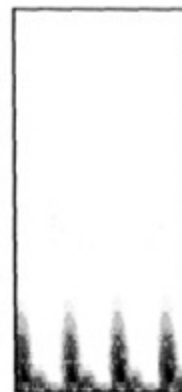
A continuación se analiza la manera en que afectan la variación del *paso* y el *tamaño de ventana* en la forma del espectrograma de la señal ECG 1 de la figura 4.6. Se presentan tres espectrogramas variando el parámetro del *paso*, son mostrados en las figuras 4.8. En la figura 4.9 se presentan dos espectrogramas generados con diferente valor en el parámetro del *tamaño de ventana*.



(a) *paso* = 5



(b) *paso* = 10



(c) *paso* = 20

Figura 4.8: Espectrogramas que analizan la señal del electrocardiograma 1. Muestran la manera de como el cambio del *paso* afecta el bosquejo del mismo. Las características comunes de los espectrogramas son: representación de la magnitud de la TF, ventana de Blackman, resolución = 128, tamaño de ventana = 15% e interpolación polinomial .

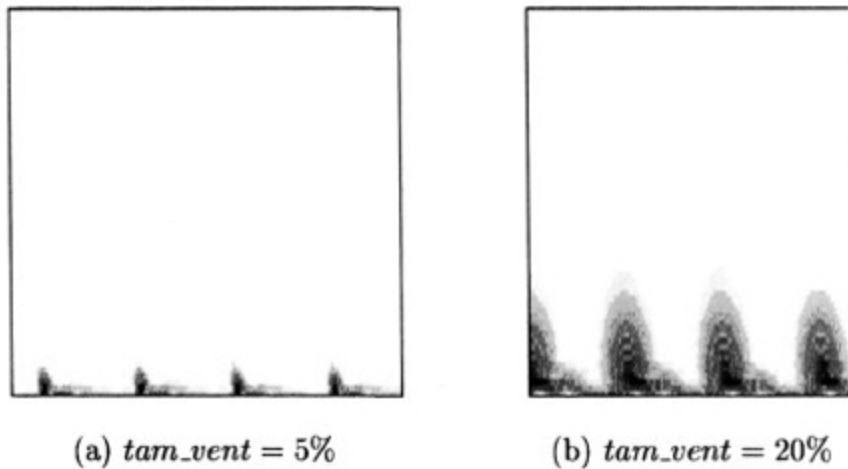
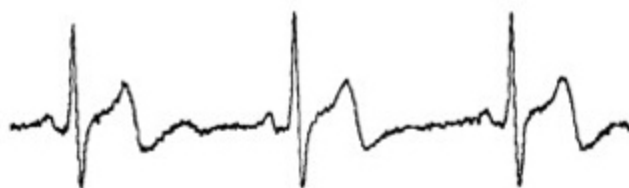


Figura 4.9: Espectrogramas de la señal del electrocardiograma 1 que muestran la manera de como afecta el cambio del tamaño de ventana al bosquejo del mismo. Las características comunes de los espectrogramas son: representación de la magnitud de la TF, ventana de Blackman, resolución = 128, paso = 10 e interpolación polinomial.

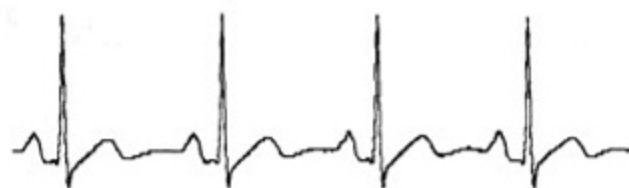
### 4.2.3 Dos electrocardiogramas diferentes

Se analizan en tiempo-frecuencia otras 2 señales de electrocardiogramas, el electrocardiograma 2 (ECG 2) y el electrocardiograma 3 (ECG 3) [12], que se muestran en los diagramas (a) y (b) de la figura 4.10. Los espectrogramas de las señales, que se muestran en los diagramas (c) y (d) de la figura 4.10, fueron generados con las mismas características del espectrograma del diagrama (d) de la figura 4.7.

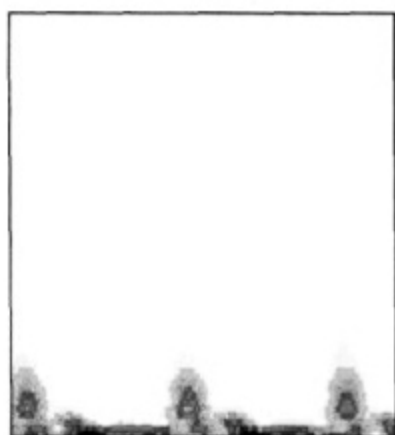




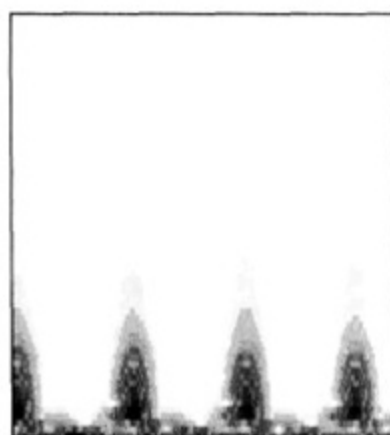
(a) Electrocardiograma 2 (amplitud vs tiempo).



(b) Electrocardiograma 3 (amplitud vs tiempo).



(c) Espectrograma de ECG 2



(d) Espectrograma del ECG 3

Figura 4.10: Señales del electrocardiograma 2 y el electrocardiograma 3 [12] (diagramas (a) y (b) respectivamente) y sus los espectrogramas que las analizan (diagramas (c) y (d) respectivamente). Los 2 espectrogramas están generados con las mismas características del diagrama (d) de la figura 4.7.

### 4.3 Análisis de una señal de voz

En esta sección se generan espectrogramas de la señal de voz de Homero Simpson (personaje de la caricatura americana titulada “The Simpsons”). La figura 4.11 contiene la señal muestreada de 39954 datos que será representada en tiempo-frecuencia.

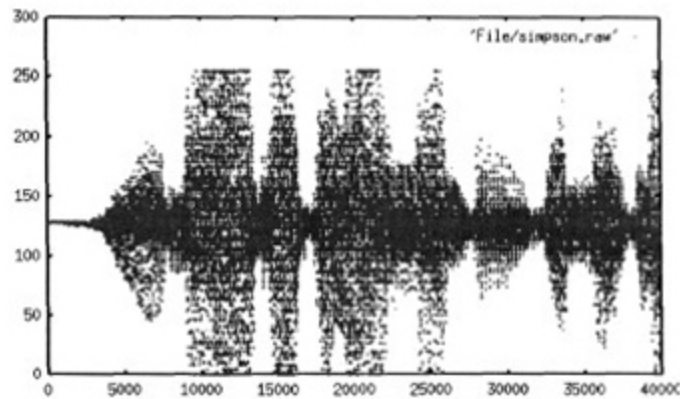


Figura 4.11: Señal de voz de Homero Simpson de 39954 datos (amplitud vs tiempo).

La señal de voz de Homero Simpson fue interpolada a 65536 datos. Para trabajar con un archivo de menos datos, la señal fue reducida a 2048 datos (señal de voz reducida) con la opción de *entre2* del menú *interpolados* de la ventana principal del SATF. La señal está graficada en la figura 4.12.



Figura 4.12: Señal de voz de Homero Simpson de 2048 datos (amplitud vs tiempo).

Se genera el espectrograma de la señal de voz reducida mostrado en la figura 4.13. Tiene las siguientes características:

<b>ventana</b> Blackman.	<b>resolución</b> baja (128 datos).
<b>tamaño</b> 10%.	<b>parte de la TF</b> Magnitud.
<b>paso</b> 10 datos.	<b>interpolación</b> polinomial.



Figura 4.13: Espectrograma de señal de voz de Homero simpson. Las características con las que se generó el espectrograma se encuentran listadas en la página 70 de esta sección.

Como se ve en la figura 4.13, el espectrograma de la señal de voz de Homero Simpson presenta principalmente un bosquejo con manchas cafés y moradas, además dos líneas de colores claros en la parte inferior del cuadro.

Experimentaremos ahora generando el espectrograma de una sección, de la señal de la voz de H. Simpson mostrada en la figura 4.11. Se extrae del archivo que almacena a la señal de voz, los datos correspondientes a la sección de la señal que se muestra en la figura 4.14. El nuevo archivo que contiene la sección de voz extraída es de 4449 datos. Se interpola a 8192 datos y después se reduce con la opción *entre2* a 2048 datos. La porción de voz ya interpolada y reducida se muestra en la figura 4.15.

El espectrograma que analiza la sección de la señal la voz de H. Simpson se encuentra en la figura 4.16. Las características de generación del espectrograma se encuentran especificadas en la misma figura.



Figura 4.14: Señal completa de la voz de H. Simpson (amplitud vs tiempo) mostrando la sección de la señal que será analizada.

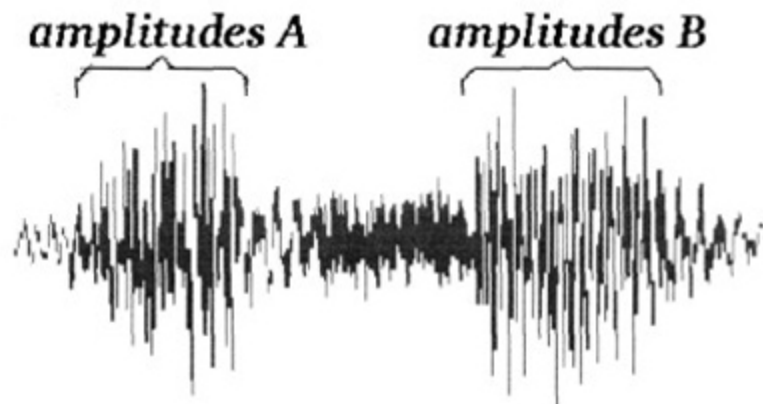


Figura 4.15: Sección extraída de la señal de voz completa de H. Simpson (amplitud vs tiempo), ya interpolada, y reducida a 2048 datos. En la señal se indican la amplitudes que tienen mayor magnitud, distinguidas como amplitudes A y amplitudes B. La sección de la señal entera de donde fue extraído este segmento de voz se muestra en la figura 4.14.

Examinando el espectrograma de la figura 4.16 observamos que el bosquejo presenta principalmente dos conjuntos de figuras amorfas, pintadas con color amarillo y un poco de color rosa. Un conjunto se encuentra en el costado izquierdo del espectrograma y el otro conjunto en el lado derecho. Cada conjunto se extienden verticalmente desde la parte inferior del diagrama hasta



Figura 4.16: Espectrograma que analiza la sección de señal extraída de la voz de Homero Simpson. Los características del espectrograma son: representación de la magnitud TF, ventana de Blackman, resolución = 64, *paso* = 10 e interpolación polinomial.

un poco mas de la mitad del mismo. Además de las 3 líneas dibujadas (con colores amarillo, verde y negro) en la parte inferior del espectrograma, que también presenta el espectrograma de la señal completa mostrado en la figura 4.13. Comparando el espectrograma de la figura 4.16 con la señal en tiempo que representa mostrada en la figura 4.15, deducimos que los conjuntos de figuras amorfas del espectrograma son debidas a las amplitudes distinguidas como A y B mostradas en la figura 4.15.

## Capítulo 5

### Conclusiones

Se construyó un sistema en ambiente gráfico para el análisis en tiempo-frecuencia de señales no estacionarias mediante la generación de espectrogramas a partir de archivos de datos. El sistema se probó usando una señal simple de prueba, señales de electrocardiogramas y señales de voz.

Los espectrogramas generados con la ventana de Hanning, la ventana Hamming y la ventana Blackman presentan bosquejos más limpios y parecidos entre sí, en comparación con los espectrogramas generados con la ventana rectangular. Los espectrogramas de algunas señales generados con la *magnitud en decibeles* pueden tener la misma forma de los espectrogramas generados con *magnitud normal* pero con mejor diferenciación en los colores, como lo muestran los resultados obtenidos con la señal  $\sin(\frac{1}{t})$ . El mejoramiento de la diferenciación en los colores es debido a que el rango de valores en la intensidad de la magnitud es extendido logarítmicamente.

Las diferencias observadas entre la interpolación polinomial o al utilizar fórmula de reconstrucción de Shannon en la elaboración de un espectrograma es mínima. La mejor opción de interpolación se debe seleccionar a base de prueba y error. Los espectrogramas construidos con la fase de la TF, regularmente presentan bosquejos amorfos. No es simple su interpretación para ver la identificación directa de algún patrón en una señal no estacionaria.

El efecto principal de aumentar la *resolución* en un espectrograma, es el de reducir la dimensión vertical del bosquejo, como lo muestran los espectrogramas de la señal  $\sin(1/t)$ . Mientras que el efecto principal de aumentar

el paso en un espectrograma, es la reducción en la dimensión horizontal del bosquejo, como se observa en los espectrogramas de la señal del electrocardiograma 1. El incremento en el valor del parámetro en el tamaño de ventana produce el aumento en el tamaño del bosquejo en el espectrograma generado.

Los espectrogramas de los electrocardiogramas tienen una apariencia sencilla. Los tres espectrogramas correspondientes a la señal del electrocardiograma 1, la señal del electrocardiograma 2 y la señal del electrocardiograma 3, analizados en el Capítulo anterior, tienen una forma global típica y difieren en particularidades como el color que presentan los bosquejos, el tamaño y detalles en la forma de los mismos. Por lo observado en los espectrogramas obtenidos de los electrocardiogramas analizados, vemos la posibilidad de desarrollar un análisis en tiempo-frecuencia con el objetivo de reconocer patrones que identifiquen a una clase de electrocardiograma en particular. Por ejemplo una clase de electrocardiograma que genere una persona con alguna enfermedad del corazón.

Los espectrogramas de señales de voz son mucho más complejos y deben analizarse por segmentos.

La variación en el valor de los parámetros, que ofrece el SATF, para generar espectrogramas crean una gran variedad de representaciones en tiempo-frecuencia para una sola señal no estacionaria. Conociendo la forma en que las opciones del SATF para generar espectrogramas modifican la apariencia del diagrama de tiempo-frecuencia elaborado, podemos crear espectrogramas con un bosquejo sencillo que proporcione la estructura particular de la señal no estacionaria analizada. Entonces estos espectrogramas tendrán las cualidades ideales para la identificación de patrones en la señal analizada.

# Apéndice A

## Funciones en lenguaje Tcl

### VARIABLES GLOBALES

Las variables globales pueden ser accedidas por cualquiera de los procedimientos y comandos en lenguaje Tcl descritos en este apéndice. Son las siguientes:

#### **arch**

Variable que contiene el nombre del archivo que ha sido últimamente seleccionado en la ventana principal del SATF.

#### **cont1**

Variable que incrementa su valor en uno, cada vez que ha sido utilizada por un procedimiento de tcl.

#### **alist**

Lista utilizada para guardar el valor de los parámetros que generan un espectrograma.

#### **path**

Variable que almacena el nombre del directorio donde se encuentran los archivos de datos a que el SATF tiene acceso.

#### **valor**

Variable que contiene el “nombre-trayectoria” del botón que ha sido últimamente seleccionado en la ventana principal del SATF.

#### **listname**

Lista que contiene los nombres de todos los archivos datos y espectrogramas que se encuentran en la ventana principal del SATF.



**archtipo**

Variable que contiene el tipo de archivo que ha sido últimamente seleccionado dentro de la ventana principal del SATF.

**val\_min**

Variable que almacena el mínimo valor (magnitud o magnitud en decibeles) de todas las frecuencias correspondientes a las transformadas de Fourier obtenidas en la elaboración de un espectrograma.

**val\_max**

Variable que almacena el máximo valor (magnitud o magnitud en decibeles) de todas las frecuencias correspondientes a las transformadas de Fourier obtenidas en la elaboración de un espectrograma.

**comentario**

Variable que almacena una cadena de texto.

**paso**

Variable que contiene el número de datos con los que se irá moviendo la ventana a través del tiempo.

**tam\_vent**

Variable que contiene el número de datos del tamaño de la ventana solicitado para generar un espectrograma.

**res**

Variable que contiene el número de datos del tamaño de la DFT solicitado para generar un espectrograma.

**ajust**

Variable que contiene el valor de la bandera de ajuste, utilizada para identificar si se requiere algún ajuste en el valor máximo de la TF (magnitud o magnitud en decibeles) en la elaboración de un espectrograma.

**der**

Variable que contiene el valor de la bandera de derivación, utilizada para identificar si se requiere el espectrograma de la señal derivada.

## A.1 Manual de procedimientos en Tcl

---

### NOMBRE

Grafica - grafica un archivo de datos.

### SINTAXIS

Fuente: ventmej.tcl

Grafica {archivo}

### DESCRIPCIÓN

Grafica a **archivo**, donde **archivo** puede contener datos crudos (extensión **.raw**) o datos interpolados (extensión **intp**).

---

### NOMBRE

GrafReIm - grafica la parte real e imaginaria de archivo de datos con TF.

### SINOPSIS

Fuente: ventmej.tcl

GrafReIm {archivo}

### DESCRIPCIÓN

Grafica la parte real e imaginaria del archivo de datos con TF llamado **archivo** (extensión **fft**).

---

### NOMBRE

GrafFase - grafica la fase de los archivos de datos con TF.

### SINOPSIS

Fuente: ventmej.tcl

GrafFase {archivo}

### DESCRIPCIÓN

Grafica la fase del archivo de datos con TF llamado **archivo** (extensión **fft**).

---

**NOMBRE**

GrafMag - grafica la magnitud de los archivos de datos con TF.

**SINOPSIS**

Fuente: ventmej.tcl  
GrafMag {archivo}

**DESCRIPCIÓN**

Grafica la magnitud del archivo de datos con TF llamado **archivo** (extensión ft).

---

**NOMBRE**

GrafLog - grafica el  $20 \cdot \log(\text{magnitud})$  de un archivo de datos con TF.

**SINOPSIS**

Fuente: ventmej.tcl  
GrafLog {archivo}

**DESCRIPCIÓN**

Grafica la magnitud del archivo de datos con TF llamado **archivo** (extensión ft).

---

**NOMBRE**

activa Activa el menú correspondiente al tipo de archivo seleccionado.

**SINOPSIS**

Fuente: ventmej.tcl  
activa {w}

**DESCRIPCIÓN**

Adquiere el tipo de archivo correspondiente al archivo últimamente seleccionado en la ventana principal (*usel*) del SATF y lo guarda en **archtipo**. Y activa el menú correspondiente al archivo *usel* a través de la función **Menuon**.

---

**NOMBRE**

interpol.- Maneja el proceso de interpolación de un archivo de datos.

**SINTAXIS**

Fuente: ventmej.tcl  
 interpol {q}

**DESCRIPCIÓN**

Interpola el archivo de datos crudos llamada **arch** llamando al comando **poli1** cuando **q = 0**, o al comando **fpb** cuando **q = 1**. El nombre del archivo (extensión **intp**), donde se almacenaron los datos interpolados es colocado en **listname**.

---

**NOMBRE**

**Seleccion** - Coloca en la ventana principal los archivos seleccionados desde la ventana de *Seleccion de Archivo*.

**SINOPSIS**

Fuente: ventmej.tcl  
**Seleccion** {w}  
 w: Nombre del archivo seleccionado desde la ventana de *Seleccion de Archivo*.

**DESCRIPCIÓN**

Coloca en la *ventana principal* del SATF los archivos seleccionados desde la ventana de *Seleccion de Archivo*, en el lugar correspondiente al tipo de archivo seleccionado. Si el archivo es de extensión **spc**, llama a la función **memspc** para que los datos de este archivo sean colocados en memoria.

---

**NOMBRE**

**Fft** - Maneja el proceso de obtener la transformada de Fourier de un archivo de datos interpolados.

**SINOPSIS**

Fuente: ventmej.tcl  
**Fft** { }

**DESCRIPCIÓN**

Llama al commando de tcl **fft** para obtener la transformada de Fourier del archivo de datos interpolados llamado **arch**. El nombre del archivo (extensión **fft**), donde se almacenaron los datos interpolados es colocado en **listname**

---

**NOMBRE**

Espectro - Maneja el proceso para generar un espectrograma.

**SINOPSIS**

Fuente: ventmej.tcl

Espectro { }

**DESCRIPCIÓN**

Genera el espectrograma de la señal almacenada en el archivo interpolado **arch** realizando la siguiente serie de pasos:

-Obtiene el número de datos que contiene **arch**, llamando al procedimiento **Obt\_NoDatos**.

-Llama al procedimiento **VentEspect** para obtener todas las caracterizaciones del espectro.

-Según las características que se solicitaron del espectrograma llama a los comandos de tcl **tronca**, **deriva**, **fftspc** o **fftspb**, con el objetivo de obtener el archivo con todos colores que pintan espectrograma.

-Al final llama al procedimiento **Colorea**, que imprime el espectrograma dentro de una ventana gráfica.

---

**NOMBRE**

Quitar - Quita un nombre de archivos de la ventana principal del SATF.

**SINOPSIS**

Fuente: ventmej.tcl

Quitar { }

**DESCRIPCIÓN**

Quita el nombre del archivo **arch**. Llama a la función **menuoff**, para desactivar los menús *Crudos*, *Interpolados*, *fft* y *Espectrograma* que se encuentran en la barra de menús de la ventana principal.

---

**NOMBRE**

Paint - Controla los pasos para construir un espectrograma que ya ha sido generado.

**SINTAXIS**

Fuente: ventmej.tcl  
**Paint** {{imprimir 0}}  
**imprimir** - variable que indica el estado de impresión. Su default es 0 e indica que no se ha solicitado la impresión de algún espectrograma.

**DESCRIPCIÓN**

Busca en **alist** el nombre del espectrograma solicitado y el valor de los parámetros del espectrograma requerido. Dependiendo de las características del espectrograma se llaman a los comandos de tcl: **tronca**, **deriva**, **fftspc** o **fftspb** con el objetivo de obtener todos los colores que pintan el espectrograma. Al final llama al procedimiento **Colorea**, para imprimir el espectrograma dentro de una ventana gráfica.

---

**NOMBRE**

**Memspc** - Coloca en memoria los datos de un archivo con extensión spc

**SINOPSIS**

Fuente: ventmej.tcl  
**Memspc** {valor archivo}

**DESCRIPCIÓN**

Almacena en **alist** los datos de un archivo con extensión spc que ha sido seleccionado desde la ventana de *Selección de Archivo*. Los datos corresponden a los parámetros de generación de un espectrograma.

---

**NOMBRE**

**archselect** - Crea una ventana donde se seleccionan archivos de datos con los que el usuario desea trabajar.

**SINTAXIS**

Fuente: seleccion.tcl  
**archselect** { w .sel }  
**w** (con default **.sel** ) es el "nombre-valor" del toplevel, donde se construye la ventana gráfica para selección de archivos.

**DESCRIPCIÓN**

Crea una ventana llamada *Selección de Archivo* mostrada en la figura 3.2 de la sección 3.1. En esta ventana se seleccionan los archivos de datos con los que el usuario desea trabajar.

---

**NOMBRE**

Mitad - Maneja el proceso de la reducción de datos de un archivo de datos interpolados.

**SINOPSIS**

Fuente: seleccion.tcl  
Mitad { }

**DESCRIPCIÓN**

Reduce a la mitad el número de datos de `arch` llamando al comando `entredos`, que borra los datos impares.

---

**NOMBRE**

Imprimir - Crea un postscript para el archivo `arch`.

**SINOPSIS**

Fuente: seleccion.tcl  
Imprimir { }

**DESCRIPCIÓN**

Lee el tipo de archivo seleccionado en `archtipo`. Si `archtipo` vale 0 o 1, crea un postscript de la grafica de la señal almacenada en `arch`. Si `archtipo` es 3 (indentifica a los espectrogramas) llama a la función `post`.

---

**NOMBRE**

Imprimir2 - Crea postscripts para archivos de datos con transformada de Fourier.

**SINOPSIS**

Fuente: seleccion.tcl  
Imprimir2 {parte}  
parte - variable que indica la parte de la TF del archivo de datos que se quiere graficar

## DESCRIPCIÓN

Crea un postscript de la grafica de la **parte** de la TF, cuyo datos estan almacenados en **arch**.

---

## NOMBRE

**VentEspect2** - Crea una ventana donde se proporcionan las características del espectrograma a generar.

## SINOPSIS

Fuente: seleccion.tcl

**VentEspect2** {No\_datos {w .carac}}

**No\_datos** : número de datos del archivo interpolado, que almacena la señal que será analizada en tiempo-frecuencia.

**w** : nombre-valor del top-level donde se construye la ventana "Caracterizacion del espectrograma" su default es **.carac**.

## DESCRIPCIÓN

Crea la ventana mostrada en la figura 3.6 que contiene distintos dispositivos gráficos donde proporcionan el valor deseado en los parámetros que caracterizan el espectrograma a generar. Estas variables son:

**tipo**, **vent**, **tam\_vent**, **paso**, **inter**, **res**, **der**, **par**, **name**, **dat\_ini**, **dat\_fin**, **val\_min**, **val\_max**, **cancel**.

---

## NOMBRE

**Actlista** - Realiza los comandos de los dispositivos gráficos correspondientes a los parámetros *ajuste*, *resolución* y *tipo de ventana*, pertenecientes a la ventana *Caracterización de Espectrograma*.

## SINOPSIS

Fuente: seleccion.tcl

**Actlista** {a tiporadio}

**tiporadio**: variable que identifica al dispositivo gráfico.

**a** : variable que contiene el valor del botón.

## DESCRIPCIÓN

Identifica la opción **tiporadio**, y después realiza su comando.

---



**NOMBRE**

**ErrorVentSpc** - Checa errores en la ventana de caracterización de espectrograma

**SINOPSIS**

Fuente: seleccion.tcl

**ErrorVentSpc** {No\_datos}

**No\_datos**: número de datos del archivo datos interpolados. El archivo del cual va a ser generado un espectrograma

**DESCRIPCIÓN**

Checa errores posibles que pueden ser efectuados por el usuario, al proporcionar un parámetro en la ventana de *Caracterización de Espectrograma*. A las variables que se le checa algún error son:

**paso**, **tam\_vent**, **dat\_min**, **dat\_max**, **val\_min**, **val\_max**, y el nombre del espectrograma.

---

**NOMBRE**

**Obt\_NoDatos** - Obtiene el número de datos que almacena un archivo.

**SINOPSIS**

Fuente: seleccion.tcl

**Obt\_NoDatos** { }

**DESCRIPCIÓN**

Obtiene el número de datos que contiene **arch**. La función regresa el valor obtenido.

---

**NOMBRE**

**Colorea** - Dibuja dentro de una ventana gráfica el espectrograma requerido.

**SINTAXIS**

Fuente: procedimientos.tcl

**Colorea** {NoDatos estado}

**NoDatos**: número de datos que contiene el archivo de datos interpolados. De este archivo se va a generar el espectrograma

**estado**: variable que indica si el espectrograma se encuentra graficado (**estado** = 1) o no (**estado** = 0), o si solamente se requiere generar el espectrograma para editarlo en el *xv* (**estado** = 2).

**DESCRIPCIÓN**

Primero crea el cuadro gráfico (canvas) donde el espectrograma será bosquejado. Después da lectura a los colores de los rectángulos que formarán el espectrograma, al mismo tiempo de ir los colocando dentro del canvas. Si la variable de **estado** es igual a 2, al terminar de construir el espectrograma, se llama al procedimiento **Post**.

---

**NOMBRE**

**VentText** - Crea un editor de texto para visualizar los datos de un archivo.

**SINOPSIS**

Fuente: procedimientos.tcl  
**VentText**{**w** **.texto**}}  
**w** - variable que recibe el nombre-valor de la top-level donde se crea el editor de texto, su default es **.texto**.

**DESCRIPCIÓN**

Se genera una ventana gráfica que contiene un marco en la parte superior, donde hay un menú con las opciones de *salvar*, *salvar como* y *salir*. Abajo del marco superior, el procedimiento **VentText** construye un editor de texto, que tiene un scroll en posición vertical. Al terminar de construir el editor de texto llama al procedimiento **Abrir**.

---

**NOMBRE**

**Post** - Maneja el proceso de crear un archivo postscript del espectrograma seleccionado .

**SINOPSIS**

Fuente: procedimientos.tcl  
**Post**{ **{quitar 0}** **{bandera 0}** }  
**quitar**: variable que indica si el espectrograma creado va a ser destruido. Cuando su valor es uno (activado), el espectrograma va a ser borrado de pantalla. Su default es cero (desactivado).  
**bandera**: variable que indica si el espectrograma está siendo construido, para así esperar a que éste sea terminado.

**DESCRIPCIÓN**

Crea un archivo postscript del espectrograma que ha sido últimamente seleccionado en la ventana principal del SATF.

**NOMBRE**

**Saveim** - Almacena los parámetros que caracterizan un espectrograma dentro un archivo de datos (extensión spc).

**SINOPSIS**

Fuente: procedimientos.tcl

**Saveim** { }

**DESCRIPCIÓN**

Busca en **alist**, los parámetros que caracterizan el espectrograma cuyo nombre se encuentra almacenado en **arch**. Crea el archivo **arch.spc**. Lee los parámetros de la lista y los va salvando en el archivo abierto.

**NOMBRE**

**Comment** - Crea una ventana para escribir un comentario.

**SINOPSIS**

Fuente: rawsel.tcl

**Comment** {{**purpose** ‘Comentario:’}{**w** .fileSelectWindow}} **purpose**: comentario que etiquetará la ventana gráfica que se construye en este procedimiento. Su default es ‘Comentario’.

**w**: contiene el nombre-valor del toplevel donde se crea la ventana gráfica para el comentario.

**DESCRIPCIÓN**

Crea una ventana gráfica que solamente contiene una etiqueta que cuyo texto se encuentra en el argumento **purpose**, y una entrada de texto. En la entrada de texto es donde se coloca el comentario deseado.

**NOMBRE**

**Listar** - Activa el proceso de listar los archivos de datos.

**SINOPSIS**

Fuente: listas.tcl

**Listar** {tipodata}

**tipodata**: parámetro donde se recibe que tipo de archivo de dato se desea

listar. Sus opciones son: 0 para archivos de datos crudos, 1 para archivos de datos interpolados, 2 para archivos de datos con TF, 3 para archivos de datos para espectrogramas.

### DESCRIPCIÓN

Listar coloca la extensión del tipo de archivo de datos seleccionado, en la variable global `tipo`. Las extensiones pueden ser: `raw` para archivos de datos *crudos*, `intp` para archivo de datos interpolados, `fft` para archivos de datos con TF, y `spc` para archivo de datos que contienen los parámetros que caracterizan un espectrograma. Después llama al procedimiento `archselect` que construye la ventana de *Selección de Archivo* mostrada en la figura 3.2 en la sección 3.1.

---

### NOMBRE

`Abrir` - Coloca en el editor de texto del sistema el contenido del archivo `usel`.

### SINOPSIS

Fuente: `listas.tcl`

`Abrir {w} {w}`: "nombre-valor" del editor de texto del sistema.

### DESCRIPCIÓN

Abre el archivo `arch` para solo lectura y coloca el contenido en `w`.

## A.2 Manual de comandos

### NOMBRE

`poli1` - Interpola un archivo de datos, utilizando la interpolación polinomial.

### SINOPSIS

Fuente: `poli.c`

`poli1 archivo1 archivo2`

`archivo1`: archivo que contiene los datos no interpolados (datos crudos).

`archivo2`: archivo donde se almacenarán los datos interpolados.

### DESCRIPCIÓN

Interpola la secuencia de datos almacenada en `archivo1`, y la guarda en `archivo2`.

---

### NOMBRE

`fpb` - Interpola un archivo de datos, utilizando la fórmula de reconstrucción de Shannon.

### SINOPSIS

Fuente: `interpola.c`

`fpb archivo1 archivo2`

`archivo1`: archivo que contiene los datos no interpolados (datos crudos).

`archivo2`: archivo donde se almacenarán los datos interpolados.

### DESCRIPCIÓN

Interpola la secuencia de datos almacenada en `archivo1`, y la guarda en `archivo2`.

---

### NOMBRE

`tronca` - Recorta los datos de un archivo.

### SINOPSIS

Fuente: `tronca.c`

`tronca archivo1 archivo2 datoini datofin`

`archivo1`: archivo que almacena la secuencia de datos a recortar.

`archivo2`: archivo donde se almacenarán la secuencia recortada de datos.

`modo`: modo de lectura de los datos inicial y final para la secuencia de datos reducida. Puede ser en tiempo o en número de dato.

`datoini`: tiempo o número de dato con el que iniciará la secuencia reducida.

`datofin`: tiempo o número de dato con el que finalizará la secuencia reducida.

**DESCRIPCIÓN**

Obtiene la secuencia reducida de datos con los datos que se encuentran en el intervalo [datoini,datofin], almacenados en `archivo1`, y los coloca en `archivo2`.

---

**NOMBRE**

`entredos` - Reduce a la mitad la secuencia de datos numéricos almacenada en un archivo, sin cambiar la forma de la señal que se encuentra dentro de la misma.

**SINOPSIS**

Fuente: `entredos.c`  
`entredos archivo`  
`archivo`: archivo que almacena la secuencia de datos a reducir.

**DESCRIPCIÓN**

Elimina los datos impares contenidos en `archivo`, donde el primer dato es par (dato 0).

---

**NOMBRE**

`fft` - Obtiene la transformada de Fourier de un archivo de datos interpolados.

**SINOPSIS**

Fuente: `fft.c`  
`fft archivo1 archivo2`  
`archivo1`: archivo que almacena la secuencia de datos interpolados. `archivo2`: archivo donde se almacenarán los datos con transformada de Fourier.

**DESCRIPCIÓN**

Calcula la transformada de Fourier de los datos que se encuentran en `archivo1`, y almacena el resultado en `archivo2`.

---

**NOMBRE**

`fftspec`, `fpbspec` - Obtiene los colores en la generación de un espectrograma.

**SINTAXIS**

Fuente: `fftspc.c`

`fftspc parametros`

`fpbspc parametros`

`parametros`: lista de parámetros que caracterizan el espectrograma a generar, se describen en la tabla 2.3.

**DESCRIPCIÓN**

Obtiene los colores utilizados para generar el espectrograma de la señal almacenada en `arch`, con las características que se le proporcionan en `parametros`.

Los comandos difieren en el tipo de interpolación utilizada:

`fftspc`- utiliza la interpolación polinomial.

`fpbspc`- utiliza la fórmula de reconstrucción de Shannon.

# Apéndice B

## Manual de funciones en lenguaje C

---

### NOMBRE

**Datos** - Lee un archivo de datos crudos.

### SINOPSIS

Fuente: poli.c, interpola.c

**Datos** ( int \*res, int \*no\_datos, char \*raw)

**res**: puntero a la ubicación donde la función regresará el número de datos a los que se va a interpolar el archivo de datos crudos (extensión raw).

**no\_datos**: puntero a la ubicación donde la función regresa el número de datos que contiene el archivo de datos crudos.

**raw**: puntero a la cadena de caracteres donde se encuentra el nombre del archivo que contiene los datos que serán leídos.

### DESCRIPCIÓN

Abre el archivo, cuyo nombre se encuentra en la posición que apunta **raw**, para solo lectura. Ignora los comentarios que se encuentran en el archivo, y lee el modo en que vienen los datos almacenados en el archivo. El modo puede ser:

modo 0.- tiene una columna de datos que son valor de la función ( $f(t)$ ) en formato entero.

modo 1.- son 2 columnas de datos, en cada renglón se encuentra el par  $(t, f(t))$  en formato flotante.

modo 2.- 1 columna de datos, contiene  $f(t)$  en formato flotante.

modo 3.- 2 columnas de datos, en cada renglón se encuentra el par  $(t, f(t))$  en formato entero.



Después de que se lee el modo, aloja memoria para los arreglos donde se almacenarán los datos crudos. Se da lectura a los datos, y se almacenan dentro de los arreglos.

---

## NOMBRE

**Interpolación** - Interpola una secuencia de datos.

## SINOPSIS

Fuente: poli.c

```
void Interpolacion( int NNP, int NP, float dx, char *intp)
```

**NNP**: número de datos que almacenará la secuencia datos interpolados.

**NP**: número de datos que almacena de la secuencia de datos no interpolados.

**dx**: Separación (en el eje horizontal) entre dos muestras consecutivas de la secuencia de datos interpolados.

**intp**: puntero a la cadena de caracteres que contiene el nombre del archivo (ext intp) donde se almacenará la secuencia datos interpolados.

## DESCRIPCIÓN

La función **Interpolacion** abre el archivo correspondiente a **intp** para solo escritura, donde se imprimirán los datos interpolados. Se imprime en el archivo que ha sido abierto, el número de datos interpolados **NNP** que contendrán el archivo (formato de archivo). Después la función **Interpolacion**, dentro un ciclo de iterativo, interpola los **NP** datos crudos, a **NNP** datos interpolados. Para estimar cada valor de la secuencia interpolada, se llama función **polint**.

---

## NOMBRE

**Principal** - Maneja el proceso de interpolar un archivo de datos crudos.

## SINOPSIS

Fuente: poli.c

```
void Principal(char *raw, char *intp)
```

**raw**: Puntero a la cadena de caracteres que contiene el nombre del archivo de datos no interpolados (datos crudos).

**intp**: Puntero a la cadena de caracteres que contiene nombre del archivo donde se almacenarán los datos interpolados.

## DESCRIPCIÓN

La función **Principal** llama a la función **Datos**, que obtiene el intervalo entre dos muestras consecutivas de la secuencia interpolada. Y además

la función **Datos**, coloca en memoria los datos crudos almacenados en el archivo correspondiente a **raw**, dentro de 2 arreglos globales. Después la función **Principal** llama a la función **Interpolacion** para interpolar los datos crudos y almacenarlos en el archivo cuyo nombre se encuentra en la dirección que apunta **intp**.

---

## NOMBRE

**Espectro**, **Espectro\_FPB** - Maneja el proceso de obtener los colores de un espectrograma.

## SINOPSIS

Fuente: `fftspec.c`

```
void Espectro(float *xintp, float *yintp, int inf[ ])
```

```
void Espectro_FPB(float *xintp, float *yintp, int inf[])
```

**xintp**: Arreglo donde se almacenan las abscisas de la señal con la cual se va a generar un espectrograma.

**yintp**: Arreglo donde se guardan las ordenadas de las señal con la cual se va a generar un espectrograma.

**inf**: Arreglo que contiene los parámetros de caracterización del espectrograma a generarse. Los parámetros son los siguientes (en el orden creciente del arreglo): *paso*, *selección de ventana*, *tamaño de ventana*, *resolución de la DFT*, *parte de la TF*, *bandera de ajuste*, *número de datos*, *modo*, *val\_max* y *val\_min*.

## DESCRIPCIÓN

Las funciones **Espectro** y **Espectro\_FPB** realizan la siguiente serie de pasos.  
 -Cuando la resolución de la DFT es menor que el tamaño de ventana, la función **Espectro** incrementa el valor de la resolución a la próxima potencia de 2 mayor que el tamaño de la ventana.

-Aloja memoria para los distintos arreglos que se necesitan para interpolar y obtener la TF.

-Obtiene el número de segmentos de la señal no estacionaria que se requieren para generar el espectrograma. También obtiene el intervalo entre dos muestras consecutivas, correspondientes al segmento de señal no estacionaria, cuyos datos serán interpolados a la resolución con la cual va a ser obtenida la DFT.

-Obtiene y almacena en un arreglo, los valores de la ventana de tiempo que va a ser multiplicada por el segmento de la señal no estacionaria de datos interpolados. Hay 4 tipos distintos de ventana: *Rectangular*, *Hamming*, *Hanning* y *Blackman*.

-Si al espectrograma no se le ha proporcionado valor máximo y mínimo para la TF, obtiene los valores máximo, y mínimo en la magnitud o magnitud en decibelios en la TF. Para la obtención de la de estos valores se utiliza la

función `Obtiene_Uni2`.

-Abre un archivo de solo escritura llamado `espectral`. Imprime en este archivo el formato de archivo que espera leer el procedimiento de `tcl Colorea`.

-Realiza el proceso de cargar en un arreglo, interpolar, multiplicar por la ventana de tiempo, obtener y representar en colores la TF cada segmento de la señal no estacionaria que se requiere para generar el espectrograma.

`Espectro` y `Espectro_FPB` difieren en que llaman a funciones distintas al interpolar los arreglos de datos de los segmentos de la señal.

`Espectro` llama a la función de de Interpolacion.

`Espectro_FPB` llama a la función de `InterFPB`.

## NOMBRE

`Obtiene_Uni2` - Calcula los valores máximo y mínimo en la magnitud o la magnitud en decibels en todas transformadas de Fourier que se obtienen al generar un espectrograma.

## SINOPSIS

Fuente: `ftspc.c`

```
void Obtiene_Uni2( float *xintp, float *yintp, float *vent,
int inf[ ], int pciclos[ ], float dx)
```

`xintp`: Arreglo donde se almacenan las abscisas de la señal con la cual se va a generar un espectrograma.

`yintp`: Arreglo donde se guardan las ordenadas de las señal con la cual se va a generar un espectrograma.

`vent`: Puntero al arreglo donde se guardan los valores de la ventana que será utilizada para obtener el espectrograma.

`inf`: Puntero al arreglo que contiene los parámetros de caracterización del espectrograma a generar. Los parámetros son los siguientes (en el orden creciente del arreglo): *paso*, *selección de ventana*, *tamaño de ventana*, *resolución de la DFT*, *parte de la TF*, *bandera de ajuste*, *número de datos*, *modo*, *val.max* y *val.mi n*.

`pciclos`: Arreglo que contiene el número total de segmentos de la señal no estacionaria, requeridos para generar el espectrograma, y la potencia de dos a la que está elevada la resolución deseada en la TF.

`dx`: intervalo entre dos muestras consecutivas de los segmentos de datos interpolados de la señal no estacionaria.

## DESCRIPCIÓN

`Obtiene_Uni2` primero aloja memoria para los distintos arreglos que se necesitan para interpolar y obtener la TF. En un proceso iterativo carga en un arreglo, interpola y multiplica por la ventana de tiempo cada segmento que se requiere para generar el espectrograma de la señal. En cada iteración llama a la función `Unidad`, para ir calculando los valores máximos y mínimos

en la *magnitud* o *magnitud en decibeles* de la TF. Al terminar el proceso iterativo, `Obtiene.Uni2` libera la memoria alojada. Y por último regresa en el arreglo de `inf[ ]`, el valor máximo (en `inf[8]`) y el valor mínimo (en `inf[9]`) obtenidos.

---

## NOMBRE

`Lec.Datos` - Lee un archivo de datos interpolados.

## SINOPSIS

Fuente: `fftspc.c`

```
int Lec_Datos( float *xintp, float *yintp, char *arch )
```

`xintp`: puntero al arreglo donde se almacenarán los valores de las abscisas de la señal almacenada en el archivo de datos interpolados.

`yintp`: puntero al arreglo donde se almacenarán los valores de las ordenadas de la señal almacenada en el archivo que será de datos interpolados.

`arch`: puntero a la cadena de caracteres que contiene el nombre del archivo que almacena los valores de la señal no estacionaria que serán leídos.

## DESCRIPCIÓN

`Lec.Datos` abre el archivo cuyo nombre está en la posición que apunta `arch`. Empieza a leer el archivo ignorando los comentarios que contiene (empiezan con el carácter gato `#`). Dentro de un ciclo, da lectura a los datos que se encuentran dentro del archivo. Estos datos son alojados en los arreglos ubicados en las posición que apuntan los argumentos `xintp` y `yintp`.

---

## NOMBRE

`Fix.Res` - Ajusta la resolución en la generación del espectrograma cuando se proporciona un tamaño de ventana mayor que la resolución de la DFT deseada.

## SINOPSIS

Fuente: `ajuste.c`

```
void Fix_Res(int inf[ ])
```

`inf`: arreglo que contiene el valor de la resolución requerida para la obtención del espectrograma, y también donde la función regresa el valor de la resolución ajustada.

## DESCRIPCIÓN

Calcula la nueva resolución con la característica de que es la próxima potencia de dos tal que:

nueva resolución > tamaño de la ventana requerida para la obtención del espectrograma.

La nueva resolución es regresada en el arreglo `inf[ ]`, en la posición `inf[NR + 1]`. Así, la resolución requerida ubicada en `inf[3]`, no se borra. Activa la bandera de nueva resolución (`inf[NR] = 1`), donde NR es una constante definida al principio del programa.

## NOMBRE

**Ajustador** - Reduce el número de datos de una secuencia a la resolución requerida para generar el espectrograma.

## SINOPSIS

Fuente: ajuste.c

```
void Ajustador(float *realnew, float *real, float *im, int inf[ ])
```

**real**: puntero al arreglo que contiene `inf[nR+1]` valores, que forman la secuencia de datos no reducida.

**realnew**: puntero al arreglo donde se almacenarán los datos de la secuencia reducida a la resolución requerida.

**inf**: arreglo que contiene el valor resolución requerida (`inf[3]`) y el valor de la resolución actual almacenada en `inf[NR+1]`.

## DESCRIPCIÓN

Calcula un número  $v$  tal que  $2^v \text{inf}[3] = \text{inf}[NR + 1]$ . En un ciclo de `inf[3]` oscilaciones, almacena la secuencia reducida en número de datos en el arreglo que apunta `realnew`. Los valores de la secuencia reducida, son obtenidos a partir de la secuencia sin reducir de la siguiente manera:

```
realnew[j] = real[v*j] .
```

Donde  $j$  representa el número de dato de la secuencia reducida.

## NOMBRE

**Pasabaja** - Interpola un dato utilizando la fórmula de reconstrucción de Shannon.

## SINOPSIS

Fuente: ajuste.c

```
void pasabaja( float *y, float periodo[ ], float *ym, int m, int n0)
```

**y**: Puntero al arreglo de la secuencia no interpolada.

**periodo**: Contiene el valor de los periodos de la secuencia no interpolada, y de la secuencia interpolada.

*ym*: Parámetro donde la función regresa el valor del dato estimado.  
*m*: Número de dato a estimar de la secuencia interpolada. El valor de este dato será *\*ym*.  
*n0*: posición del dato central (secuencia no interpolada) de la sumatoria utilizada en el algoritmo de obtención del dato interpolado ( con filtro pasabajas).

### DESCRIPCIÓN

Primero se inicializa *\*ym* a 0.0. Después se realiza el proceso de cálculo utilizando el algoritmo de la fórmula de reconstrucción de Shannon, para estimar el valor del dato *m* de la secuencia interpolada. El valor estimado es regresado en la posición que apunta *ym*.

---

### NOMBRE

**InterFPB** - Interpola un arreglo de datos, utilizando la fórmula de reconstrucción de Shannon.

### INTAXIS

Fuente: ajuste.c

```
void InterFPB( float y[ ], float yres[ ],int inf[ ], float p[ ] )
```

*y[ ]*: arreglo de la secuencia que contiene los valores de los datos no interpolados.

*yres[ ]*: arreglo donde se alojará la secuencia de datos interpolados.

*inf[ ]*: arreglo que contiene el número de datos de la secuencia no interpolada, y el número de datos que tendrá la secuencia interpolada.

*p[ ]*: arreglo que contiene los periodos de la secuencia no interpolada y la secuencia interpolada.

### DESCRIPCIÓN

Interpola la secuencia de datos *y[ ]* con periodo de muestreo de *p[0]*, a una secuencia de *inf[3]* datos (o la interpola a una secuencia de *inf[NR+1]* datos si *inf[NR] = 1*), con periodo de muestreo en *p[1]*. Los datos interpolados son almacenados en *yres[ ]*. En cada iteración se llama a la función *pasabaja* para que se encargue de estimar el dato solicitado.

---

### NOMBRE

**Hanning** - Obtiene la ventana de Hanning.

**SINTAXIS**

Fuente: ventcol.c  
`void Hanning(float *f, int size)`  
`f` : puntero al arreglo donde se almacenará la secuencia de valores de la ventana de Hanning.  
`size` : tamaño en número de datos que contendrá la ventana de Hanning.

**DESCRIPCIÓN**

En un ciclo de `size` iteraciones, se calcula cada valor de la ventana de Hanning y se coloca en el arreglo correspondiente a `f`.

---

**NOMBRE**

`Hamming` - Obtiene la ventana de Hamming.

**SINOPSIS**

Fuente: ventcol.c  
`void Hamming(float *f, int size)`  
`f` : puntero al arreglo donde se almacenará la secuencia de valores de la ventana de Hamming.  
`size` : tamaño en número de datos que contendrá la ventana de Hamming.

**DESCRIPCIÓN**

En un ciclo de `size` iteraciones, se calcula cada valor de la ventana de Hamming y se coloca en el arreglo correspondiente a `f`.

---

**NOMBRE**

`Blackman` - Obtiene la ventana de Blackman.

**SINOPSIS**

Fuente: ventcol.c  
`void Blackman(float *f, int size)`  
`f` : puntero al arreglo donde se almacenará la secuencia de valores de la ventana de Blackman.  
`size` : tamaño en número de datos que contendrá la ventana de Blackman.

**DESCRIPCIÓN**

En un ciclo de `size` iteraciones, se calcula cada valor de la ventana de Blackman y se coloca en el arreglo correspondiente a `f`.

---

**NOMBRE**

`Rect` - Obtiene la ventana de Rectangular.

**SINOPSIS**

Fuente: `ventcol.c`  
`void Rect(float *f, int size)`  
`f` : puntero al arreglo donde se almacenará la secuencia de valores de la ventana de Hanning.  
`size` : tamaño en número de datos que contendrá la ventana de rectangular.

**DESCRIPCIÓN**

Se igualan a uno todos los elementos del arreglo correspondiente a `f`.

---

**NOMBRE**

`Sel_Color` - Almacena un color dentro de un archivo de datos.

**SINOPSIS**

Fuente: `ventcol.c`  
`void Sel_Color(int no_color, FILE *fp)`  
`int no_color`: Selección del color que se va a ser impreso en el archivo de datos.  
`FILE *fp`: flujo al archivo donde se imprimirá el nombre del color identificado por `no_color`.

**DESCRIPCIÓN**

Selecciona el color identificado por `no_color` y lo almacena en el archivo abierto con flujo `fp`.

---

**NOMBRE**

`Colorfase` - Representa en colores la fase de cada una de las frecuencias de la TF de una secuencia de datos y almacena los nombres de los colores en un archivo de datos.

**SINOPSIS**

Fuente: `ventcol.c`  
`void Colorfase( float *refft, float *imfft, int inf[ ], FILE *fp)`  
`refft` : puntero al arreglo donde se almacenan los valores de la parte real



de la TF.

**imfft** : puntero al arreglo donde se almacenan los valores de la parte imaginaria de la TF.

**inf** : arreglo que contiene el tamaño de la DFT almacenado en **inf**[3].

**fp** : flujo al archivo de datos donde se almacenarán el nombre de los colores obtenidos.

## DESCRIPCIÓN

Obtiene la escala para la selección de color, dividiendo  $2\pi$  entre el *número de colores* que contiene la escala utilizada (se trabaja con una escala de 16 colores). Para el valor de la fase de cada frecuencia de la TF con parte real ubicada en la dirección que apunta **refft**, y parte imaginaria en la dirección que apunta **imfft**, obtiene el número de color. Llama a la función **Sel.Color** para almacenar, en el archivo de datos al cual se conectado a través de **fp**, el nombre correspondiente al número de color obtenido.

## NOMBRE

**Colormag** - Representa en colores la magnitud de cada una de las frecuencias de la TF de una secuencia de datos y almacena los nombres de los colores en un archivo de datos.

## SINOPSIS

Fuente: ventcol.c

```
void Colormag( float *refft, float *imfft, int inf[ ], FILE *fp)
```

**refft** : puntero al arreglo donde se almacenan los valores de la parte real de la TF.

**imfft** : puntero al arreglo donde se almacenan los valores de la parte imaginaria de la TF.

**inf** : arreglo que contiene el valor de la resolución almacenado en **inf**[3], el valor máximo de la TF en **inf**[9], y la bandera de ajuste en la magnitud máxima de la TF en **inf**[5].

**fp** : flujo al archivo donde se almacenarán el nombre de los colores obtenidos.

## DESCRIPCIÓN

Calcula el valor escala para la obtención de los colores. Obtiene el número de color de cada frecuencia de la TF correspondiente a **refft** e **imfft**. Llama a la función **Sel.color** para almacenar en el archivo de al cual se esta conectado a través **fp**, el nombre correspondiente al color obtenido.

Cuando la bandera de ajuste ha sido activada (**inf**[5] = 1), los valores de la magnitud de la TF son ajustados al valor máximo en la magnitud requerido (almacenado en **inf**[9]).

---

**NOMBRE**

**Colorlog** - Representa en colores la magnitud en decibeles de cada una de las frecuencias de la TF de una secuencia de datos y almacena los nombres de los colores en un archivo de datos.

**SINOPSIS**

Fuente: ventcol2.c

```
void Colorlog( float *refft, float *imfft, int inf[ ], FILE *fp)
refft : puntero al arreglo donde se almacenan los valores de la parte real de la TF.
```

```
imfft : puntero al arreglo donde se almacenan los valores de la parte imaginaria de la TF.
```

```
inf : arreglo que contiene el valor de la resolución almacenado en inf[3], el valor máximo de la TF en inf[9], y la bandera de ajuste en la magnitud máxima de la TF en inf[5].
```

```
fp : flujo al archivo donde se imprimirán los colores obtenidos.
```

**DESCRIPCIÓN**

La descripción es exactamente la misma que la función **Colormag**, pero con la diferencia de que en la función (**Colorlog**) trabaja con la *magnitud en decibeles* y la función **Colormag** simplemente con la *magnitud*.

---

**NOMBRE**

**Interpolacion** - Interpola un arreglo de datos utilizando la interpolación polinomial.

**SINTAXIS**

Fuente: ventcol2.c

```
void Interpolacion( float x[ ], float y[ ],
float xres[ ], float yres[ ], int inf[ ], float dx)
```

```
x[ ] : Arreglo que contiene el valor de las abscisas no interpoladas.
```

```
y[ ] : Arreglo que contiene el valor de las ordenadas no interpoladas.
```

```
xres[ ] : Arreglo donde se alojarán las abscisas de los valores interpolados.
```

```
yres[ ] : Arreglo donde se alojarán las ordenadas de los valores interpolados.
```

```
inf[ ] : arreglo que contiene el número de datos de la secuencia no interpolada (inf[2]), el número de datos que almacenará la secuencia interpolada (inf[3]), y la bandera de nueva resolución (inf[NR]).
```

```
dx : intervalo entre dos muestras consecutivas de la secuencia interpolada.
```

**DESCRIPCIÓN**

El número de datos que contienen los arreglos `x[ ]` y `y[ ]` es interpolado a `inf[3]` datos, o a `inf[NR+1]` si `inf[NR] = 1`. En un ciclo iterativo se va obteniendo el valor de cada uno de las ordenadas interpoladas. Para estimar cada ordenadas interpolada se llama a la función `polint`.

**NOMBRE**

`FFT` - Maneja el proceso de obtener y representar en colores la transformada de Fourier de una secuencia de datos.

**SINOPSIS**

Fuente: `ventcol2.c`

```
void FFT( float *refft, float *imfft, int *inf,
FILE *fp, int pot)
```

`refft` : Puntero al arreglo donde se aloja la parte real de la secuencia de datos.

`imfft` : Puntero al arreglo donde se aloja la parte imaginaria de la secuencia de datos.

`inf[ ]` : arreglo que contiene el valor de la resolución de la TF requerida.

`fp` : flujo al archivo donde serán almacenados el nombre de los colores que representan a la TF de la secuencia de datos.

`pot` : exponente al que está elevado .

**DESCRIPCIÓN**

Si la bandera de resolución está desactivada (`inf[NR] = 0`) se llama a la función `Fft`, para obtener la transformada de Fourier discreta de la secuencia de datos correspondiente a `*refft` e `*imfft`. Dependiendo de la representación solicitada para las frecuencias de la TF (magnitud, magnitud en decibeles o fase), se llama a una función que se encargue del proceso de seleccionar un color para cada uno de las componentes de la respuesta en frecuencia. Las funciones son `Colormag` (representación en magnitud), `Colorfase` (representación en fase) y `Colorlog` (representación de magnitud en decibeles). Si la bandera de resolución está activada (`inf[NR] = 1`), se llama a la función `Ajustador`, para que disminuya los datos, de `inf[NR+1]` datos a la resolución `inf[3]` solicitada para la TF. Una vez que se tiene una secuencia de `inf[3]` datos se sigue el mismo paso de llamar a la función `fft`, y después a una de las funciones `Colormag`, `Colorlog` o `Colorfase`.

---

**NOMBRE**

`Lec.Datos` - Lee un archivo de datos interpolados que se le va a obtener su transformada de Fourier.

## SINTAXIS

Fuente: `fft.c`  
`float **Lec_Datos( char *intp )`  
`intp` : Puntero a la cadena de caracteres que contiene el nombre del archivo de datos interpolados.

## DESCRIPCIÓN

Abre el archivo para solo lectura . Lee el numero de datos (`no_datos`) que contiene este archivo (formato de archivo). Aloja memoria para 2 arreglos de `no_datos` datos flotantes (arreglos globales). Dentro de un ciclo iterativo da lectura a todos los datos que se encuentran en el archivo. Coloca a las ordenadas en un uno arreglos globales (parte real). El otro arreglo que tiene apartada memoria se llena de ceros (parte imaginaria). El archivo regresa el puntero a los dos arreglos de datos flotantes.

---

## NOMBRE

`Tfourier` - Obtiene la transformada Discreta de Fourier de un arreglo de datos.

## SINOPSIS

Fuente: `fft.c`  
`int Tfourier(char *intp, char *fft)`  
`intp` : Puntero a la cadena de caracteres que contiene el nombre del archivo que almacena el arreglo de los valores que se les va obtener la transformada de Fourier. Los valores son presentados en dos columnas, en la primera columna representa las absisas, en la segunda columna se representa las ordenadas de los valores.  
`fft` : Puntero a la cadena de caracteres que contiene el nombre del archivo que contendrá el arreglo de datos con transformada de Fourier. Los datos serán almacenados en 5 columnas. Las columnas representan: la posición de la frecuencia en el eje horizontal, parte real, parte imaginaria, absoluto y la magnitud en decibeles, en el orden de la 1ª a la 5ª columna respectivamente

## DESCRIPCIÓN

Obtiene el puntero (variable global) al arreglo que contiene la parte real e imaginaria de la secuencia de datos que se le va a obtener su transformada de Fourier, obtenidas con la función `Lec_Datos`. Abre el archivo para solo escritura, correspondiente a `fft`. Calcula la potencia de dos a la que esta elevado el número de datos que contiene la secuencia que se le obtendrá su transformada. Llama a la función `Fft` para obtener la tranformada discreta de Fourier de la secuencia de datos. Almacena los valores de la respuesta en frecuencia en el archivo de datos abierto anteriormente.

**NOMBRE**

**Lec\_Datos** - Lee un archivo de datos interpolados que va a ser derivado, o reducido a la mitad de datos.

**SINOPSIS**

Fuente: `fft.c`, `entredos.c`, `deriva.c`

```
int Lec_Datos( char *comment[ ], int *inf, char *arch )
```

`comment` : Puntero al arreglo de caracteres donde se almacenarán temporalmente los renglones-comentario que contiene archivo de datos que será leído.  
`arch` : Puntero a la cadena de caracteres que contiene el nombre del archivo contiene datos que se le va a obtener su respuesta en frecuencia.

`inf[ ]` : Arreglo donde la función regresará el número de datos (en `inf[0]`) que contiene el archivo y el número de renglones-comentario (`inf[1]`) que tiene el mismo archivo.

**DESCRIPCIÓN**

El archivo correspondiente a `arch`, es abierto para solo lectura. Obtiene el número de renglones comentario que contiene el archivo y lo guarda en `inf[1]`. Aloja la memoria necesaria para el almacenamiento de estos renglones y los almacena los en `*comment[ ]`. Aloja memoria para los dos arreglos globales que almacenarán las columnas de datos del archivo abierto. Lee los datos del archivo, y los guarda en los arreglos globales.

**NOMBRE**

**Recortar** - Recorta los datos que contiene un archivo de datos interpolados.

**SINOPSIS**

Fuente: `tronca.c`

```
void Recortar( double datext[ ], char *intp,  
char *tronca )
```

`datext` : arreglo que contiene el número dato inicial, y el número de dato final que almacenará el nuevo archivo de datos recortados.

`intp` : Puntero a la cadena de caracteres que contiene el nombre del archivo que almacena la secuencia de datos que será recortada.

`tronca` : Puntero a la cadena de caracteres que contiene el nombre del archivo que almacenará la secuencia de datos recortada.

**DESCRIPCIÓN**

Lee los datos y los comentarios del archivo correspondiente `intp`, utilizando la función `Lec_datos` (todo lo que es leído se almacena en memoria en arreglos

globales). El archivo correspondiente a `tronca`, es abierto para solo escritura. Almacena en el archivo que ha sido abierto, los comentarios del archivo que contiene la secuencia completa de datos, el nuevo número de datos que tendrá la secuencia recortada y los valores de la misma.

---

#### NOMBRE

`Divide` - Reduce a la mitad el número de datos de una secuencia, eliminando los datos impares.

#### SINOPSIS

Fuente: `entredos.c`

```
void Divide( char **comment , int *inf, char *arch)
```

`comment` : puntero a arreglos de caracteres, que guarda los renglones comentario del archivo almacena la secuencia de datos interpolados no reducida.

`arch` : Cadena de caracteres que contiene el nombre del archivo almacenará el arreglo de datos reducido a la mitad.

#### DESCRIPCIÓN

El archivo correspondiente a `arch`, es abierto para solo escritura. Los renglones comentario, almacenados en `comment`, son grabados en el archivo abierto. Los datos no reducidos se encuentran alojados en dos arreglos globales (uno para las abscisas y otro para las ordenadas). Solamente son almacenados en el archivo abierto, los datos pares de los arreglo globales (inician con el dato cero).

---

#### NOMBRE

`Restador` - Deriva los datos que contiene un archivo de datos interpolados.

#### SINOPSIS

Fuente: `entredos.c`

```
void Restador( char *arch )
```

`arch` : Puntero a la cadena de caracteres que contiene el nombre del archivo que almacena los datos que van ser derivados.

#### DESCRIPCIÓN

Los datos son leídos y almacenados utilizando la función `Lec.Datos`. El archivo derivado es abierto para solo escritura. Los comentarios del archivo cuyo son grabados en derivado. En un ciclo iterativo, se restan ( o derivan) los datos, que están almacenados en dos arreglos globales (uno para las abscisas, y otro para las ordenadas) de la siguiente manera:  $y[j] - y[j - 1]$  (donde  $y[]$  representa a las ordenadas).

**NOMBRE**

**InterFPB** - Interpola los datos de un archivo de datos crudos, utilizando la fórmula de reconstrucción de Shannon.

**SINOPSIS**

Fuente: `interpola.c`

```
void InterFPB( char *raw, char *intp )
```

**raw** : Puntero a la cadena de caracteres que contiene el nombre del archivo de datos crudos.

**intp** : Puntero a la cadena de caracteres que contiene el nombre del archivo donde se almacenarán los datos interpolados.

**DESCRIPCIÓN**

La función **InterFPB** primero obtiene el incremento en el tiempo de la secuencia interpolada, llamando a la función **Datos** que también calcula los periodos para la secuencia no interpolada, y la secuencia interpolada. Los datos del archivo correspondiente a **raw** son almacenados en arreglos globales. En seguida, el archivo correspondiente a **intp**, es abierto para solo escritura. Después se obtienen los datos interpolados, al mismo tiempo que se almacenan en el archivo abierto. Para estimar cada uno de los datos interpolados se llama a la función **pasabaja**.

**NOMBRE**

**pasabaja** - Interpola un dato utilizando la fórmula de reconstrucción de Shannon.

**SINOPSIS**

Fuente: `interpola.c`

```
void pasabaja( float periodo[ ], float *ym, int m, int n0)
```

**periodo** : Arreglo que contiene el valor de los periodos de la secuencia no interpolada, y de la secuencia interpolada. **ym** : Puntero a la ubicación donde se regresa el valor del dato estimado.

**m** : número de dato de la secuencia interpolada, que será estimado.

**n0**: número de dato de la secuencia no interpolada, que será el valor central de la sumatoria en la fórmula de Shannon.

**DESCRIPCIÓN**

Primero se inicializa **\*ym** a 0.0. Después se realiza el proceso de cálculo utilizando la fórmula de reconstrucción de Shannon para estimar el valor **\*ym** del dato **m** de la secuencia interpolada.

## Bibliografía

- [1] V. Oppenheim and Ronald V. Schafer, Digital Signal Processing Alan, *Capítulo 7*, Prentice Hall, 1989, pp 444–449.
- [2] V. Oppenheim and Ronald V. Schafer, Digital Signal Processing Alan, *Capítulo 8*, Prentice Hall, 1989, pp 514–520.
- [3] V. Oppenheim and Ronald V. Schafer, Digital Signal Processing Alan, *Capítulo 9*, Prentice Hall, 1989, pp 581–582.
- [4] V. Oppenheim and Ronald V. Schafer, Digital Signal Processing Alan, *Capítulo 3*, Prentice Hall, 1989, pp 101 – 111.
- [5] Brent B. Welch, Practical Programing in Tcl and Tk, Second Edition
- [6] Peter Aitken/Bradley Jones, *Aprendiendo C en 21 dias*, Prentice Hall Hispanoamericana, 1994.
- [7] Willian H. Press Saul A. Teukoisky William T. Vettering, Numerical Recipies in C, *Capítulo 3 Interpolation and Extrapolation*, Brian P. Flannery, 1996, pp 105–110.
- [8] J.G.G. Dobe, Faster FFTs, *Dr. Dobb's Journal*, pp 125–133, febrero 1995.
- [9] Richard G. Barniuk, Time–Frequency Analysis Background, <http://www-dsp.rice.edu/software/TFA/tfa.html>, netscape
- [10] Teresa Ree Chay, Bifurcation in Heart Rhythms, *Bifurcation and Chaos*, diciembre 1995, volumen 5, número 6.
- [11] Metin Akay, Wavelet aplicacions in medicine, *IEEE spectrum*, pp 50-57, mayo de 1997.



- [12] Jose Agustin Elías Segura, *Desarrollo de una Herramienta Computacional basada en un RNA, para el prediagnóstico de cuatro tipos de infarto al miocardio usando señales de electrocardiogramas*, Centro de Investigación de Estudios Avanzados del Instituto Politécnico Nacional. 1997.

