UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

GENERADOR DE TRENES DE IMPULSOS SINTÉTICOS A PARTIR DE ESPIGAS CEREBRALES DIGITALIZADAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

IRMA DOMÍNGUEZ SOLÓRZANO

DIRECTOR: DR. JOSE ISMAEL ESPINOSA ESPINOSA

MÉXICO D.F. 1997

DEDICATORIA

A mis padres :

Isabel Solórzano Nambo
Antonio Domínguez Piñón

por darme la vida, su confianza, cariño y apoyo incondicionales,
por ser un ejemplo de trabajo, esfuerzo y dedicación a seguir,

A mis hermanos :

Jorge
Antonia

por todos esos momentos que hemos compartido.

Con todo mi cariño y respeto.

AGRADECIMIENTOS

A mis padres, Toña, Jorge, Lupe, Mary, Toñito, Luz, Faustino

por su cariño, apoyo, paciencia y compañía. Y por siempre tener una palabra de apoyo.

Al Dr. José Ismael Espinosa Espinosa

por darme su amistad y confianza,
por permitirme formar parte de su equipo de trabajo,
por acompañarme en esta última etapa como estudiante universitaria,
y sobre todo por la enorme paciencia que tuvo para conmigo durante este tiempo.

A mis amigos y compañeros

Carmen, Rosario, Minerva por ser mis mejores amigas.

Alberto, César, Eduardo, Fidel, Javier, Jordi, Jorge, Juan Manuel, Luigi, Malors, Mota, Omar, Pepe, Rafael, Ritaluz, Roberto, Rubén, Rul, Tere por darme su amistad y por compartir conmigo la magia de pertenecer al Laboratorio de Cibernética.

Héctor L, Ian, José Luis, Luis Guillermo, Manuel, Víctor y todos mis compañeros y amigos de DGSCA por todos esos momentos que hemos compartido.

Alfredo, Gerardo, Pedro y a todos mis compañeros de facultad, gracias por su amistad, enseñanzas, apoyo y compañía.

Cesar, Chris, Eamon, Goran, Jean Pierre, Manuela, Mario, Roger y todos aquellos que han compartido su amistad y cariño conmigo.

A DGAPA

Por la beca de licenciatura que me fue asignada, como parte del proyecto DGAPA-UNAM IN-100593 a cargo del Dr. José Ismael Espinosa Espinosa.

PROGRAMA

El programa EXTRACEL, desarrollado como parte de esta tesis, está disponible en:

Laboratorio de Cibernética
Departamento de Física
Facultad de Ciencias
UNAM
Tel. 622-4871

Correo electrónico :

Dr. José Ismael Espinosa Espinosa
espin@servidor.unam.mx

Irma Domínguez Solórzano
irma@ciber2.fciencias.unam.mx

CONTENIDO

CAPÍTULO 1

INTRODUCCIÓN

En la actualidad, un simulador es un programa de computadora que, basándose en un modelo matemático, permite seguir los cambios de una o varias variables conforme transcurre el tiempo.

Un modelo matemático está constituido por una o más ecuaciones, usualmente diferenciales, por medio de las cuales las variables de importancia pueden monitorearse por un intervalo determinado de tiempo.

Los modelos matemáticos son muy importantes en la ciencia y en la tecnología, porque permiten conocer mejor los procesos físicos y los dispositivos tecnológicos de tal forma que puede predecirse su funcionamiento y mejorarse su desempeño. Uno de los primeros modelos matemáticos fue el que James Clerk Maxwell hizo para el regulador de Watt de la máquina de vapor en el siglo pasado [Maxwell 68]. En estas máquinas la variable que se quería controlar era la velocidad de rotación y Watt inventó un regulador de la entrada de vapor de manera que, si aumentaba la velocidad, se reducía la entrada de vapor, y si disminuía la velocidad, se aumentaba la entrada de vapor. Este tipo de regulador hace la tarea de retroalimentación, de ahí que se pueda regular la velocidad, pero la regulación lleva consigo la posibilidad de oscilación. Con estas condiciones no era fácil regular la velocidad y eso motivó a Maxwell para desarrollar un modelo matemático del regulador. Con el modelo matemático fue posible determinar las condiciones adecuadas para controlar la velocidad.

En las Neurociencias, es decir, las disciplinas que se ocupan de estudiar el funcionamiento del cerebro normal y del cerebro enfermo, fue Nicolás Rashevsky el que empezó a proponer modelos matemáticos de neuronas, de redes neuronales y de la excitación nerviosa, entre muchas otras cosas, en su libro: "Biofísica Matemática: Fundamentos físico-matemáticos de la Biología" y que publicó en 1938 [Rashevsky 38].

Un modelo matemático es una abstracción del proceso viviente o del proceso tecnológico de que se ocupa. El proceso debe estudiarse cuidadosamente y las variables de interés deben medirse bajo condiciones controladas. Nunca es posible hacer un modelo que tome en cuenta toda la complejidad de un proceso, pero si se pueden identificar las variables más significativas y con ellas formar el modelo matemático.

Para tratar de entender el funcionamiento del cerebro también se han hecho modelos matemáticos basados en la experimentación. Seguramente el más famoso de ellos es el modelo de Hodgkin y Huxley que simula la propagación de los impulsos eléctricos en el nervio gigante del calamar [Hodgkin 52]. Estos científicos obtuvieron con este modelo el premio Nobel de Fisiología y Medicina en 1962. La variable de importancia en este modelo es el voltaje de la membrana del nervio. Hay que notar que no se modeló un cerebro, ni siquiera una parte anatómicamente importante, sino sólo un nervio viviente. Esto nos puede dar una idea de la complejidad del funcionamiento cerebral.

35 años después de Hodgkin y Huxley hemos avanzado razonablemente en el conocimiento anatómico y funcional del cerebro gracias a la capacidad de cálculo y de visualización con las computadoras modernas. Es decir, tenemos atlas computarizados del cerebro humano gracias a la resonancia magnética, la tomografía y la cámara positrónica. Sin embargo, seguimos sin tener modelos matemáticos del cerebro completo, es una tarea que continúa [Sanderson 85].

A pesar de las limitaciones, se han desarrollado una gran cantidad de modelos matemáticos para los componentes básicos del cerebro, es decir, las neuronas. Y, en años recientes, para redes neuronales. Estos modelos han permitido generar simuladores neuronales para estudiar las propiedades que poseen las neuronas cuando se comunican entre ellas.

Existen dos clases principales de simuladores basados en modelos matemáticos de la actividad neuronal. Uno de ellos son las Redes Neuronales Artificiales(RNA) que presentan

propiedades tales como aprendizaje, generalización, memoria y capacidad de optimización. Estos simuladores son más de lo que indica su nombre, son también herramientas tecnológicas poderosas en tareas como clasificación de patrones, memorias asociativas y control automático. Su origen se remonta al modelo de neuronas formales de McCulloch y Pitts publicado en 1943 [McCulloch 43]. El otro de ellos son los simuladores que utiliza la Neurociencia Computacional(NC). Aquí de lo que se trata es de modelar con bastante detalle la realidad biológica de una neurona específica o de una red neuronal. Este modelado va fuertemente apareado con experimentación cuidadosa de la región que se trata de modelar. El objetivo de los simuladores de la NC es conocer el funcionamiento, los mecanismos de una neurona o red neuronal en particular. Estos simuladores son los herederos directos del modelo de Hodgkin y Huxley.

Existe una gran cantidad de simuladores de RNA, unos son de dominio público y con fines didácticos y otros son comerciales y con finalidades tecnológicas y pueden alcanzar precios muy altos. Los simuladores de NC son principalmente herramientas de investigación básica y algunos con fines didácticos y la mayoría de ellos de dominio público. Dos han alcanzado notoriedad y son el llamado NEURONA desarrollado por Hines [Hines 89] y el otro es GENESIS desarrollado por Bower [Bower 88], los dos en los Estados Unidos. El simulador GENESIS permite simular la actividad eléctrica de una neurona con gran detalle biofísico y morfológico. Y también permite construir pequeñas redes neuronales. Este simulador prácticamente permite realizar experimentos biofísicos virtuales. El simulador NEURONA es hasta cierto punto similar a GENESIS. Estos simuladores usualmente requieren de estaciones de trabajo y sistema operativo UNIX para ser manejados satisfactoriamente y ya existen versiones en paralelo para ser trabajadas en supercomputadora.

Por otra parte, existen simuladores neuronales que no están basados en un modelo matemático, sino que están constituidos por componentes obtenidos, por así decirlo, directamente del sistema biológico. Un simulador prominente de esta clase es el conocido como retina neuromórfica desarrollada por Carver Mead [Mead 89], este es un modelo de una retina hecho en hardware, es decir, con la tecnología para desarrollar circuitos integrados con alta miniaturización y siguiendo el diseño biológico, no una ecuación. A esta clase, aunque no

necesariamente desarrollados en hardware, pertenecen también los simuladores de la actividad eléctrica de las neuronas en los cuales se toma como componente base a los potenciales de acción [Calvin 75] registrados experimentalmente por medio de microelectrodos de metal o de vidrio. Con microelectrodos de metal se registra la actividad llamada extracelular y que incluye a las neuronas que están cerca del microelectrodo. La señal obtenida de esta manera está compuesta de las contribuciones de cada neurona y representa el potencial de campo. Por otra parte, si se utiliza una micropipeta de vidrio lo que se registra es la actividad eléctrica de la neurona en que éste se clava y que se conoce como registro intracelular. La señal es el potencial de la membrana. En este trabajo nuestro interés está centrado en los registros extracelulares porque son los que llevan información de varias neuronas y, por eso, pueden servir para conocer si tales neuronas se comunican entre ellas. Si se hace, por medio de registro extracelular, un muestreo suficientemente amplio de la actividad eléctrica neuronal de una zona, es posible reunir una base de señales suficientemente grande que puede servir tanto como para reconocimiento de potenciales como para producir simuladores de señales extracelulares, entre otras aplicaciones [Abeles 75, Quiza 91]. Así pues, es posible simular señales como las que se obtienen al registrar con microelectrodos de metal. La disponibilidad de señales sintéticas facilita las tares de análisis e interpretación de los datos que se obtienen en experimentos de registro extracelular [Gerstein 83, Serna 94] y ese es precisamente el objetivo de este trabajo.

CAPÍTULO 2

OBJETIVOS

Desarrollar en PC un programa (EXTRACEL), para generar trenes de impulsos sintéticos. Generar trenes de impulsos que sigan un patrón o un comportamiento conocido. Para lograrlo los trenes de impulsos serán generados por un simulador de redes neuronales artificiales o podrán improvisarse según las necesidades. Para esto se cuenta con una base de datos de espigas de neuronas de la corteza auditiva del gato (Véase Apéndice A), previamente digitalizadas y en base a las cuales se generarán los trenes de impulsos siguiendo el comportamiento de la red neuronal que se haya elegido para tal fin o siguiendo un patrón temporal seleccionado a voluntad con tiempos predeterminados o aleatorios.

Realizar análisis de trenes de impulsos que hayan sido generados por el programa, ya sea por medio del paquete DISCOVERY [Serna 94], de DataWave o por medio del paquete CLASIF [Quiza 91].

En el Laboratorio de Cibernética se cuenta con una gama de paquetes de registro, simulación, y análisis de redes neuronales, tanto biológicas como artificiales. Sin embargo, los diversos programas no son compatibles entre sí. Por lo que este programa generador de trenes es un auxiliar para enlazarlos cuando se requieren analizar o evaluar registros electrofisiológicos extracelulares con varios microelectrodos que no es fácil de hacer experimentalmente.

El programa EXTRACEL cuenta con varios módulos:

En el primero de ellos, útil en la generación de trenes que incluyan diferentes secuencias de espigas, el primer paso es elegir qué espigas, de la base de datos, serán incluidas. El sub-módulo de lectura de espigas se encarga de solicitar los datos referentes a qué espigas y en qué tiempos se presentarán los disparos, que pueden ser predeterminados o aleatorios.

El módulo dos permite leer los datos de un tren generado anteriormente, para modificarlo o simplemente desplegarlo en pantalla.

La insercion de espigas adicionales en un tren es controlada por el tercer módulo.

El módulo cuatro proporciona la posibilidad de generar trenes ruidosos con la posibilidad de controlar el porcentaje de ruido incluido en éstos.

El módulo cinco permite graficar en pantalla los trenes generados.

El tren puede ser almacenado, en diferentes formatos, para su análisis posterior, en el módulo seis.

Las diferencias y semejanzas que existen entre las espigas seleccionadas, se observan al traslaparlas con el módulo siete.

Los tiempos de ocurrencia de las espigas y otra información relevante sobre el tren puede ser proporcionada por el módulo ocho.

Utilizando el programa descrito se presentan ejemplos de trenes de espigas multi-unitarias generadas por el mismo.

Los trenes generados serán útiles para la validación y entrenamiento en el uso de paquetes especializados de análisis de señales con los que ya cuenta el Laboratorio de Cibernética.

Los trenes de impulsos sintéticos proporcionarán marcos de referencia a partir de los cuales será posible determinar con mayor certeza la conectividad funcional entre neuronas a partir de las señales reales registradas durante los experimentos neurofisiológicos realizados en dicho laboratorio.

CAPÍTULO 3

ANTECEDENTES

Desde sus orígenes las computadoras fueron diseñadas para formar parte de las diversas herramientas de apoyo a investigadores.

A medida que se ha avanzado en el desarrollo de nuevas tecnologías los costos de los equipos computacionales han disminuido.

Es gracias a esto que actualmente los equipos de cómputo son ampliamente utilizados en un sinfin de aplicaciones. Permitiendo que cada vez más personas tengan acceso a equipos que les permitan realizar sus trabajos de una manera más fácil y rápida.

El uso de computadoras como herramienta en las investigaciones neurofisiológicas tiene ya una historia de casi 40 años [O'Connell 73].

Un área importante de la aplicación del cómputo en la neurofisiología es la adquisición de señales de tejido neuronal y su procesamiento.

En las investigaciones sobre redes neuronales biológicas es posible realizar registros extracelulares de la actividad eléctrica de las neuronas mediante el uso de microelectrodos de tungsteno y de otros materiales.

Las neuronas están interconectadas entre sí y se comunican mediante impulsos eléctricos los cuales tienen formas y amplitudes particulares. Es decir cada neurona genera impulsos eléctricos con amplitud y forma característica.

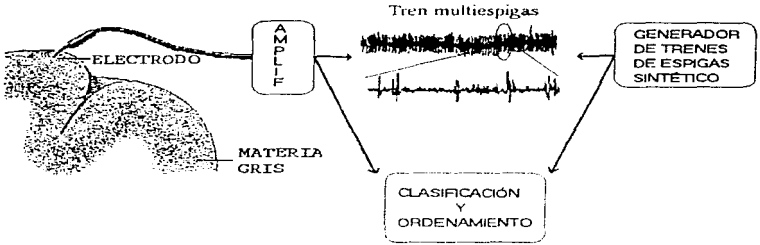


Fig. 3.1 Dado un tren multiespigas detectar e identificar las diferentes clases de espigas

El electrodo, al ser introducido en el área donde se desea hacer el experimento, registra o sensa los impulsos eléctricos de las neuronas cercanas a éste (Véase Fig.3.1).

Estos impulsos o espigas tienen una duración que varía de 0.5 a 5.0 msegs, y amplitudes de 50 a 500 μV, son capturados por el microelectrodo en forma de trenes de impulsos, que están compuestos por la suma de las espigas producidas por las neuronas cercanas a la zona de registro.

El proceso de registro y la apariencia característica de los registros obtenidos experimentalmente se muestran en la Fig.3.1.

La actividad de las neuronas se origina ya sea como una respuesta a estímulos externos o como actividad espontánea propia de cada una de ellas, y puede proporcionar información acerca de la interacción funcional que existe entre las neuronas localizadas en la zona donde se colocó el microelectrodo.

Es posible registrar diferentes tipo de potenciales en las células neuronales, sin embargo es ampliamente aceptado que el potencial de acción, es decir la espiga de una neurona, representa la unidad elemental de información en el sistema nervioso.

Debido al tamaño de las neuronas es sumamente difícil llevar a cabo registros en neuronas individuales, por lo que se ha enfatizado en el desarrollo de técnicas para aislar células nerviosas y realizar así estudios acerca de su comportamiento eléctrico.

El número de espigas registradas por el microelectrodo durante los experimentos, depende en gran medida del tamaño de las células del tejido nervioso que se esté estudiando así como de la calidad del registro; de esta manera, si se registran de 5 a 10 espigas diferentes, realizar un estudio individual es difícil de llevar a cabo, por lo que en estos casos se trabaja en análisis poblacionales como son: frecuencia y energía totales.

Si el número de espigas registradas varía de 2 a 5, entonces se cuenta, generalmente, con ciertas diferencias en forma y amplitud, que pueden ser identificadas con relativa facilidad, y que hacen factible la separación y clasificación de los impulsos presentes en el registro, para su análisis posterior.

El investigador puede observar algunas de estas diferencias durante el desarrollo del experimento, sin embargo, se tienen grandes ventajas tanto en rapidez como en poder de análisis, si la separación y discriminación de los potenciales se realizan con ayuda de una computadora.

El creciente interés en contar con herramientas que dieran apoyo en los experimentos realizados por los investigadores ha permitido el desarrollo de un gran número de dispositivos y programas orientados a la separación y clasificación de espigas neuronales. Ya sea en tiempo real o posterior al experimento.

O'Connell desarrolló una microcomputadora para realizar la identificación de impulsos nerviosos mezclados en un canal de registro [O'Connell 73].

Wodleriger desarrolló un sistema de extracción para simplificar la medición de cinco de los parámetros de las espigas [Wodleriger 78], mediante un circuito analógico. Estos parámetros son: La amplitud del pico del potencial de acción, la máxima inclinación del disparo, el umbral detectado en la inflexión durante la fase ascendente de la espiga, el intervalo de tiempo entre la inflexión y el pico del impulso y la constante de tiempo de la "exponencialidad" de la base de incremento del potencial.

Kreiter diseñó un sistema para la clasificación de espigas registradas por un sólo electrodo, en tiempo real [Kreiter 89]. Basado en un microprocesador MC68000 que fue montado en una tarjeta, auxiliado por otros elementos de hardware, que se comunicaba con una microcomputadora a través de un puerto serial RS232. El funcionamiento de este sistema está basado en dos fases principales:

- La fase de aprendizaje, durante la cual el sistema adquiere la información necesaria para distinguir entre las diferentes formas de las espigas. El entrenamiento se realiza usando espigas muestra, que han sido digitalizadas de tal manera que cada espiga está formada por 64 puntos.

El sistema determina e identifica los (ocho) puntos determinantes, en la diferenciación de las espigas, que servirán de referencia al sistema. Para poder llevar a cabo, posteriormente, una discriminación entre las espigas del grupo de aprendizaje y las que se estén registrando para su clasificación, durante el experimento.

- La fase de clasificación en tiempo real. Durante la cual, ante una señal activada por el sistema al detectar una espiga, activa la captura de los puntos principales que caracterizan la espiga, en base a los cuales es posible realizar la clasificación de las mismas.

La clasificación está basada en la mínima distancia Euclidiana que exista entre los puntos de la espiga detectada, y el conjunto utilizado durante el aprendizaje.

La información que proporcionan los trenes de impulsos es extraída mediante el análisis de los registros realizados.

Existen paquetes que permiten realizar los análisis necesarios, pero para dar una correcta interpretación a los resultados obtenidos es necesario contar con un marco de referencia adecuado.

Es aquí donde radica la importancia de este proyecto, ya que al desarrollar el programa que generará los trenes de impulsos sintéticos éstos serán útiles como entradas conocidas y controladas a los programas de análisis. Así se tendrá una herramienta de verificación de resultados de los programas utilizados y que servirá de guía cuando se analicen los registros con señales reales [Lara 96].

Ahora bien, la generación de los trenes de impulsos puede realizarse siguiendo el patrón proporcionado por algún simulador de redes neuronales. En cuyo caso la finalidad del análisis al que dichos trenes sean sometidos será la reconstrucción de la red que los generó.

O bien es posible generar trenes de impulsos con un comportamiento aleatorio, semejante al que se tiene en los registros reales.

Cabe mencionar que aunque puede considerarse conceptualmente contradictorio el utilizar una de las herramientas más precisas y determinísticas, como es el caso de las computadoras, para generar números aleatorios. Los generadores de números aleatorios son comúnmente utilizados.

No debe olvidarse, sin embargo, el hecho de que lo que puede ser considerado suficientemente aleatorio para una aplicación, puede no serlo para otras aplicaciones.

Después de todo, cualquier programa genera una salida predecible en su totalidad, es decir, no se trata de números realmente aleatorios, es por lo que se habla de números pseudo-aleatorios.

CAPÍTULO 4

ESTRATEGIAS DE PROGRAMACIÓN

En el Laboratorio de Cibernética se cuenta con una gama de paquetes de análisis, registro y simulación de redes neuronales. Los cuales son herramientas útiles en los proyectos que ahí se realizan.

Sin embargo son herramientas que se utilizan independientemente. Ya que los formatos que utilizan no son compatibles entre sí.

Por otro lado, como ya se ha mencionado anteriormente, uno de los principales intereses dentro de las investigaciones neurofisiológicas es lograr mediante el análisis de los registros que se llevan a cabo, la identificación y separación de las espigas que constituyen dichos registros, así como la reconstrucción de las redes que generaron los impulsos registrados.

En este capítulo se presenta la descripción del software desarrollado para generar trenes de impulsos sintéticos.

El programa fue desarrollado en lenguaje C y para ser ejecutado en computadoras personales.

Será utilizado como una herramienta de apoyo en el entrenamiento para el uso de paquetes de análisis electrofisiológico.

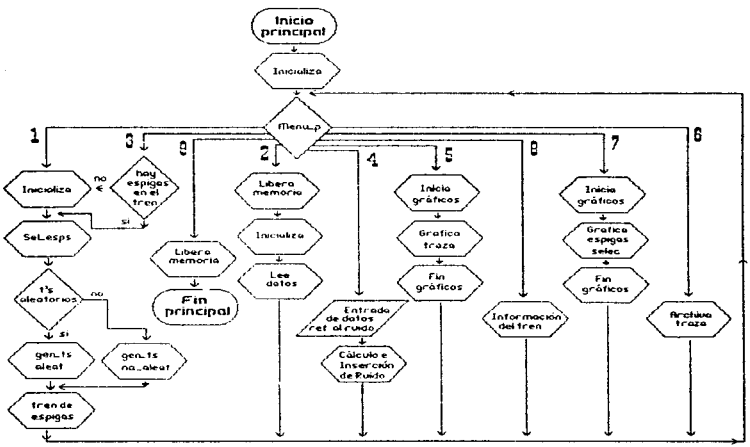


Fig. 4.1 Diagrama de Flujo del programa con rutinas principales.

4.1 METODOLOGÍA

La programación estructural que se utilizó fue con la finalidad de desarrollar módulos que permitieran un manejo sencillo del sistema y una mayor facilidad en el mantenimiento del mismo.

Este programa fue pensado de manera tal que las diferentes rutinas que lo componen puedan ser usadas entre sí. Evitando así redundancia en el código (Véase Fig.4.1).

Los datos del tren de espigas con que se está trabajando pueden ser modificados únicamente por las funciones relacionadas con la creación del tren y adición de nuevas espigas al mismo.

Se buscó proporcionar toda la información importante relacionada con el tren, para darle al investigador el control total sobre el proceso de generación de los trenes, que es la finalidad de este proyecto. Y es aquí donde se originaron los diferentes módulos que componen el sistema creado.

Parte de los recursos de información con que ya se cuenta es una base de datos de espigas digitalizadas. Está formada por setenta y dos archivos binarios. Cada uno de los cuales contiene 128 datos de tipo entero, que describen la forma de la espiga correspondiente.

La información necesaria para generar los trenes será solicitada y/o accesada dentro de cada uno de los módulos a medida que sea requerida.

De esta manera, se cuenta con la materia prima necesaria para poder realizar simulaciones de trenes de espigas. Los trenes de espigas se denominarán indistintamente trazas o señales, que sirvan a su vez para interactuar con los diferentes programas ya existentes.

Los módulos del sistema se muestran en la Fig.4.2 y son:

1) Generar Traza
2) Cargar Traza
3) Agregar una espiga a la traza
4) Agregar ruido a la señal
5) Graficar traza
6) Salvar
7) Traslapar las espigas
8) Información de la traza
9) Salir

Al ejecutar el sistema se debe de elegir el idioma en el que los mensajes de peticiones y de información del sistema serán desplegados. La función *textos* es la encargada de llevar a cabo el despliegue de los mensajes, eligiendo la cadena adecuada en base al indicador que se le pasa como parámetro y al idioma seleccionado.

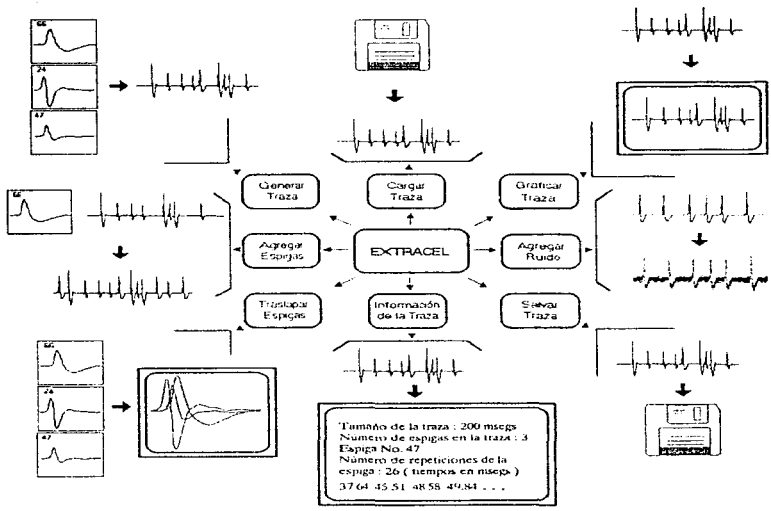


Fig. 4.2 Módulos del programa. En el orden de las manecillas del reloj :

Generar Traza.- Permite a partir de las espigas seleccionadas generar un tren de espigas en memoria.

Cargar traza.- Carga un tren de espigas de disco a memoria.

Graficar traza.- Permite desplegar en pantalla la traza almacenada en memoria.

Agregar ruido.- Agrega ruido a la traza que está en memoria.

Salvar traza.- Almacena la información de la traza que está en memoria en un archivo en disco.

Información de la traza.- Despliega la información relativa a la traza que está en memoria en pantalla.

Traslapar espigas.- Permite desplegar en pantalla, las espigas que forman el tren que está en memoria.

Agregar espigas .- Inserta una nueva espiga en el tren que ya está en memoria.

Por el momento se tienen dos opciones disponibles en la selección : Español e Inglés.

Por la forma en que está programada esta función se requieren de cambios mínimos en el código del programa, así como la adición de los textos en el idioma que se desea incluir en el cuerpo de la función *textos* y la posterior recompilación del programa.

1) GENERACIÓN DE TRENES DE ESPIGAS (Generar Traza)

Al seleccionar este módulo, se verifica si ya la memoria estaba ocupada con alguna otra traza, de ser así dicha memoria es liberada, para asignársela a la nueva Traza. La asignación de la memoria se lleva a cabo con ayuda de la función *inicializa*.

En la generación de los trenes es importante conocer en que tiempos es que se presentarán cada uno de los pulsos que se incluirán en el tren.

La serie de tiempos que será utilizada puede seguir un patrón establecido por algún programa de simulación de redes neuronales, o bien seguir el patrón de ocurrencias que el investigador determine al proporcionarle los datos al programa.

Se tienen dos opciones principales en la generación de los trenes y éstas se refieren a la entrada de datos de *tiempos predeterminados*, o bien a la generación de los tiempos de manera *aleatoria*.

Es por esto que se tienen dos rutinas de lectura-generación de tiempos (Véase Fig.4.1).

El primer paso en generación de datos es común a los dos procesos, y es la selección de las espigas que serán incluidas en el tren de impulsos. Esta acción es llevada a cabo por la función *sel_esps*, y es la que se encarga de almacenar en un vector las espigas que fueron seleccionadas para ser insertadas en el tren.

Al seleccionar cada espiga se verifica que las espigas que estamos seleccionando no hayan sido elegidas previamente.

Una vez que ya se cuenta con la información acerca de qué espigas serán incluidas deben calcularse los tiempos en que se presentarán, siempre y cuando esto sea necesario.

Este proceso es llevado a cabo por las rutinas correspondientes, las cuales son descritas a continuación.

•Generación de Tiempos Predeterminados

La rutina asociada a este proceso (*gen_ts_noal*), es la encargada de solicitar al usuario del sistema los tiempos asociados a cada una de las espigas que hayan sido seleccionadas.

Las neuronas pueden presentar disparos de una manera periódica, es decir que cada 5 msegs se presente una espiga. O bien puede darse el caso de que se presente una serie de disparos y que estos se repitan de una manera periódica también. A este comportamiento se le conoce como disparos en ráfagas.

Aunque también puede darse el caso de que el tiempo de disparo de una espiga no tenga relación alguna con el tiempo de disparo de la siguiente.

El primer dato que nos interesa conocer es el del período de disparo de la neurona, en caso de que no se desee un comportamiento periódico el asignar un valor de 0 (cero) a este dato funciona como una bandera que le indica al proceso que solo capturará los datos que sean proporcionados y que no se repetirán.

El manejar un período de repetición tiene la finalidad de facilitar al usuario la utilización del sistema, reduciendo así el número de datos que tendrá que proporcionar.

A continuación se proporcionan los tiempos en que se presentarán las espigas dentro de lo que sería el primer período de la neurona. Si se manejó un periodo igual a cero, entonces los tiempos que se proporcionen en este caso serán los únicos que se presentarán en el tren.

Al finalizar la entrada de datos se tiene la posibilidad de modificarlos en caso de que se haya presentado un error.

Los impulsos que se tendrán por período son almacenados en un vector. Se verifica que de acuerdo a los tiempos proporcionados por el usuario, no haya posibles traslapes de cada una de las espigas de una misma neurona, ya que esto no es permitido.

La neurona no puede producir un segundo disparo antes de que termine el inmediato anterior.

Se verifica también que cuando se deseen comportamientos periódicos el período que se desea manejar sea de un tamaño adecuado al tiempo requerido para que presenten todas las espigas especificadas para el primer período, de no ser así este período es ajustado automáticamente.

Una vez que los tiempos que se tienen son los que se desean manejar, se procede a calcular los tiempos para los periodos subsecuentes. Se calcula el número de impulsos totales en el tren de acuerdo a la magnitud de este último. Y se crea el vector en el cual será almacenada la información de esta espiga.

Para almacenar los tiempos de ocurrencia de cada una de las espigas se usa memoria dinámica, y las localidades asignadas dependen del número de datos que sea necesario almacenar.

Los procesos de generación de tiempos predeterminados así como el de tiempos aleatorios son ejecutados para cada una de las espigas, lo que facilita el ser utilizados de manera indistinta, tanto en el módulo de Generación de Trenes como en el de Adición de Espigas a la traza (éste último descrito más adelante).

•Generación de Tiempos Aleatorios

Existe la posibilidad de que sea el sistema el que determine los tiempos en que se presentará cada una de las espigas que serán incluidas en el tren. Es la función *gen_ts_aleat* la que se encarga de llevar a cabo el proceso.

La generación de números aleatorios no es trivial, y existen un gran número de trabajos dedicados al estudio de la generación de números aleatorios.

Generalmente los compiladores cuentan con una librería que permite generar números aleatorios. Comúnmente conocida como : *random(semilla)*. En la cual la semilla permite generar diferentes secuencias de números aleatorios, que serán las mismas cada vez que se utilice la misma semilla.

Los trenes de espigas reales siguen una distribución de Poisson, es por esto que se acudió a la literatura correspondiente para auxiliarnos de las investigaciones realizadas previamente acerca del tema.

Se trata, casi siempre, de generadores lineales congruentes que generan una secuencia de números enteros $I_1, I_2, I_3, \ldots$, cada uno entre 0 y $m-1$ basada en la relación de recurrencia.

$$I_{j+1} = a\ I_j + c \quad (\text{mod } m) \qquad \text{(ec. 1)}$$

Donde m es el módulo y a y c son enteros positivos llamados multiplicador e incremento, respectivamente. La recurrencia se repite eventualmente a sí misma con un período que es no mayor que m . Si m, a y c son elegidos apropiadamente, entonces el período será el máximo posible, de longitud m en éste caso.

La ventaja de éste método es que es bastante rápido y no requiere de muchas operaciones a realizar en cada llamada. Sin mencionar que es casi de uso universal. La desventaja que tiene es que puede presentarse una correlación secuencial en llamadas sucesivas del generador.

Hay evidencia tanto teórica como práctica de que el algoritmo congruente multiplicativo [Press 92]

$$I_{j+1} = a\ I_j \quad (\bmod\ m) \qquad \text{(ec.2)}$$

puede ser tan bueno como el mejor de los generadores congruenciales lineales que tienen $c \neq 0$ (ec.1), siempre y cuando m y a sean elegidos adecuadamente.

Los números aleatorios con desviación uniforme, son aquellos que caen dentro de un intervalo específico, en el que todos los números tienen la misma probabilidad de ocurrir.

Sin embargo, en ocasiones no es suficiente que los números aleatorios que hayan sido generados tengan una desviación uniforme. Es por esto que el método del rechazo es una herramienta muy útil, para generar números aleatorios cuya función de distribución $p(x)dx$ (probabilidad de que ocurra un valor entre x y $x+dx$), es conocida y calculable.

Este método está basado en un simple argumento geométrico:

Si se dibuja la gráfica de la distribución de probabilidad $p(x)$, que se desea obtener(Ver Fig.4.3). Entonces el área bajo la curva en cualquier intervalo de x, corresponde a la probabilidad que se quiere que tenga la ocurrencia de un valor en ese intervalo.

En la misma gráfica se dibuja una curva $f(x)$ de área finita, ubicada en la parte superior de la función original de probabilidad, a la cuál llamaremos función de comparación. Si generamos números aleatorios en dos dimensiones, que tengan una distribución uniforme bajo el área de la curva de comparación, entonces, siempre que un punto quede por arriba de la curva de probabilidad original ese punto será rechazado, y los puntos aceptables serán aquellos

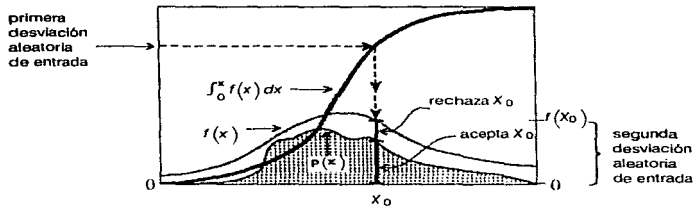


Fig. 4.3 Método del rechazo.

que estén ubicados bajo la curva de probabilidad original. De ésta manera los puntos que sean seleccionados tendrán la distribución deseada.

La distribución de Poisson se refiere a la probabilidad de que un cierto número entero m dentro del intervalo de eventos aleatorios de Poisson ocurra en un intervalo de tiempo x.

Es importante hacer notar que m toma solamente valores ≥ 0, es por eso que la distribución de Poisson, vista como una función de distribución $p_x(m)dm$ es cero en cualquier punto que m no sea un entero ≥ 0.

Tenemos que la integral de la probabilidad en una región que contiene un entero es un número finito. La probabilidad total para un entero j es:

$$\text{Prob}\,(j) = \int_{j-\varepsilon}^{j+\varepsilon} p_x\,(m)\,dm = \frac{x^{\,j} e^{-x}}{j\,!} \qquad \text{(ec.3)}$$

A simple vista puede no parecer una distribución adecuada para el método del rechazo, ya que no es posible realizar una comparación fuera de áreas infinitamente altas pero sumamente estrechas. Sin embargo es posible distribuir el área finita de j en el intervalo localizado entre j y $j+1$. Con lo que se define la distribución continua

$$q_x\,(m)\,dm = \frac{x^{\,[m]} e^{-x}}{[m]\,!}\,dm \qquad \text{(ec.4)}$$

donde $[m]$ representa el número entero más grande menor que m.

Las funciones que están descritas en el programa *dist_poi.c* son las encargadas de generar los números aleatorios que siguen una distribución de Poisson. En base a los cuales se determinará en que tiempos se colocará cada uno de las espigas de cada neurona.

Sin embargo la función *poidev* sólo se encarga de determinar qué valores, de los aleatorios que le son proporcionados, y que siguen una distribución normal, deben de ser considerados dentro de los que siguen una distribución de Poisson.

El generador de números aleatorios con distribución uniforme está implementado en la función *ran1*. Ésta rutina usa el Mínimo Standard para generar el número aleatorio pero mezcla las salidas para eliminar las correlaciones seriales de bajo orden. Una desviación aleatoria derivada del j-ésimo valor en la secuencia I_j, es la salida en la j-ésima llamada en una llamada posterior, $j + 32$ en promedio.

La función Gamma está definida por :

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt \qquad \text{(ec.5)}$$

Ahora bien cuando z es un entero, la función gamma es equivalente al factorial pero con un corrimiento de 1.

$$n! = \Gamma(n+1) \qquad \text{(ec.6)}$$

Por otro lado, la función gamma satisface la relación de recurrencia :

$$\Gamma(z+1) = z\ \Gamma(z) \qquad \text{(ec.7)}$$

Si la función es conocida para valores de $z > 1$, es posible calcular los valores para $z < 1$ por la fórmula de reflejo

$$\Gamma(1-z) = \frac{\pi}{\Gamma(z)\, sen(\pi z)} = \frac{\pi z}{\Gamma(1+z)\, sen(\pi z)} \qquad \text{(eq.8)}$$

Lanczos [Lanczos 64] propone la siguiente aproximación numérica para la que la función gamma está dada por

$$\Gamma(z+1) = \left(z+\gamma+\frac{1}{2}\right)^{z+\frac{1}{2}} e^{-\left(z+\gamma+\frac{1}{2}\right)} \qquad \text{(ec.9)}$$

$$x\sqrt{2\pi}\left[C_0 + \frac{C_1}{z+1} + \frac{C_2}{z+2} + \ldots + \frac{C_N}{z+N} + \varepsilon\right] \qquad (z > 0)$$

Es mejor implementar $\ln \Gamma(x)$ en lugar de $\Gamma(x)$, ya que ésta última genera sobrecargas por los valores generados aún cuando se trata de valores muy pequeños de x. Es por esto que generalmente $\Gamma(x)$ es utilizada cuando los valores de la función serán divididos por valores comparablemente grandes, y dicha operación es llevada a cabo como una resta de logaritmos.

Es posible calcular el logaritmo de la función gamma teniendo en cuenta la ec.9.

Y el valor es calculado por la función *gammln*.

Ya que los números aleatorios son calculados, entonces son ordenados y se verifica que las diferentes ocurrencias no originen traslapes de las espigas. En cuyo caso se eliminan los tiempos que provoquen este problema.

El número de tiempos que serán generados depende de un valor también aleatorio. Se crea el espacio para alojar dicho número de datos y una vez que se determina que valores de los presentes son válidos, entonces si es necesario se reorganiza la información en la memoria, después de eliminar los tiempos que no serán utilizados.

Los valores válidos son aquellos que no originarán traslapes de las espigas, y aquellos que se refieren a tiempos dentro del intervalo que se está manejando. Es decir si se está manejando un tren con longitud máxima de 200 msegs (el intervalo válido será de 1 a 198.7 milisegundos[1]), no es posible insertar en él una espiga cuyo tiempo de ocurrencia sea el milisegundo 250, o incluso el milisegundo 201.

•Inserción de las Espigas en la Traza

Las funciones de generación de tiempos, crean los vectores donde los tiempos de ocurrencia de las espigas son almacenados.

El siguiente paso está a cargo de la función *tren_espigas* que es la encargada de "vaciar" los datos de cada espiga en la traza. la traza es un vector de tamaño variable, el cual almacena los valores de la traza punto a punto.

[1] La duración de cada espiga es de 1.3 milisegundos. Si el tiempo de ocurrencia de una espiga rebasara el límite máximo, en éste caso los 198.7 milisegundos, la espiga aparecería truncada.

Este proceso consiste en cargar la información que describe a la espiga con la que se va a trabajar, es decir se lee el archivo que describe la forma de onda de la espiga correspondiente, se almacena en un vector de datos y esta información es "vaciada" en el vector de la traza.

El "vaciado" es en realidad la suma de los valores de la espiga con los de la traza, punto a punto, a partir de la localidad que corresponde al tiempo en el cual se presentará la espiga en la traza.

El vector de traza tiene un valor inicial de cero, en cada una de sus localidades.

Si la espiga de otra neurona se presentó en el mismo tiempo o en un tiempo cercano al de la espiga de la presente neurona, al sumar los valores de ambos vectores, se tendrá el mismo efecto que se tiene al realizar los registros de las espigas mediante el electrodo, el cual sensa la actividad en la zona de registro, como la suma de los impulsos eléctricos (potenciales de acción) de las neuronas cercanas a él.

2) LECTURA DE UN TREN ALMACENADO EN DISCO (Cargar Traza)

Este es el segundo módulo del programa y permite leer la información que ya había sido creada anteriormente.

El primer paso es liberar la memoria en caso de que esté ocupada por otra traza, crea e inicializa los vectores principales, y *lee_datos* se encarga de accesar los datos que están almacenados en el archivo.

El archivo almacenado en formato EXTRACEL (formato de éste programa) genera dos tipos de archivos, el archivo de datos y el archivo descriptor (Ver Salvar Traza), en el momento en que se está haciendo la lectura de los datos, la función *lee_datos* busca en primera instancia al archivo descriptor, si lo encuentra lee la información que está almacenada en él y crea los vectores correspondientes a los tiempos de ocurrencia de cada una de las espigas.

Este archivo es importante ya que en él se guarda toda la información referente a la traza correspondiente.[2]

[2] Ver Contenido de un archivo descriptor, en Salvar Traza.

Entonces, el archivo descriptor nos permite reconstruir la traza a partir de los datos contenidos en él. Y lo anterior es muy útil si el archivo con los datos de la traza en sí se daña o se pierde. Por otro lado, sin este archivo no es posible determinar que espigas están presentes en la traza que se está leyendo y mucho menos se sabe en qué tiempos se presentan. Aunque, si es posible cargar únicamente el archivo con los datos de la traza.

El archivo descriptor nos proporciona también varias ventajas como son:

- Ahorro de espacio en disco, los archivos descriptores ocupan un espacio mucho menor que el que ocupan las trazas, y éstas son recuperables a partir del archivo descriptor, así que con conservar éste último es suficiente.

- Si el investigador no desea dar los datos de la traza que desea generar, usando el módulo diseñado para tal fin, entonces es posible generar las trazas mediante la edición de un archivo de este tipo, teniendo cuidado de respetar el formato anteriormente descrito.

- La posibilidad de modificar la traza que ya se generó de acuerdo a los requerimientos del usuario.

3) AGREGAR UNA ESPIGA A LA TRAZA

Ya que se ha generado una traza y que nos interesa insertar una o más espigas en ésta, entonces utilizamos el presente módulo.

El funcionamiento de éste es practicamente el mismo que se utiliza para generar la traza, de hecho utiliza las mismas funciones y el proceso se lleva a cabo de la misma manera.[3]

La diferencia básica que existe es que al iniciar el proceso no se inicializa el área de memoria donde serán almacenados los datos. Es decir cada vez que se selecciona el módulo de Generación de Traza se destruye la información que estaba en memoria y se asigna el espacio a la nueva traza, y al seleccionar este módulo de Agregar una Espiga entonces se verifica si ya hay información en memoria relacionada a un tren previamente creado o que haya sido recuperado de la memoria secundaria. Si ya existe algún tren en memoria entonces se procede a solicitar la información relacionada con las nuevas espigas a insertar en el tren.

[3] Ver sección GENERACIÓN DE TRENES DE ESPIGAS, para más detalle.

En caso de que no se tenga en memoria un tren, es decir que la memoria esté vacía, entonces se crean los arreglos dinámicos, en los cuales será almacenada la información relativa al tren que se creará con las espigas seleccionadas mediante la función *inicializa*. Es decir este módulo es capaz de insertar una o más espigas a un tren ya existente o bien puede generar un nuevo tren a partir de la espigas que se seleccionen en el proceso, si es que no había ya un tren en memoria al cual insertarle la(s) nueva(s) espiga(s). Esto último equivaldría a insertar una serie de espigas a una traza vacía.

Este módulo permitirá al usuario realizar mezclas de señales, es decir, si el usuario necesita contar con un tren de espigas en el que se tengan disparos de espigas que estén colocadas en tiempos específicos con espigas que disparen de una manera aleatoria, éste módulo hace posible generar tales trenes.

Se debe mencionar que, tanto en el módulo de generación de traza como en este, si una espiga ya había sido incluida en el tren, entonces ya no podrá ser insertada nuevamente.

Y también se debe de tener en cuenta que si al cargar un tren generado anteriormente, desde disco con ayuda del segundo módulo, y si no se encontró el archivo descriptor sino únicamente el archivo de datos, entonces se tendrá en memoria la traza, pero no se podrá determinar que espigas estaban ya contenidas en tal traza. Con lo que podría estarse insertando una espiga que ya había sido incluida en la traza.

4) ADICIÓN DE RUIDO A LA SEÑAL

Este módulo nos proporciona la facilidad de generar trenes afectados por ruido gaussiano, y nos permite así tener trenes que sean más semejantes a los registros que se obtienen en los experimentos.

La función *ins_ruido* es la encargada de calcular punto a punto el valor a agregar como componente del ruido en ese punto.

Se maneja un valor de la varianza, para determinar el nivel del ruido a ser adicionado al tren.

5) DESPLIEGUE DEL TREN DE IMPULSOS (Graficar la Traza)

La graficación de la traza generada con ayuda de este programa, es posible con la ayuda de este módulo.

La función asociada al proceso es básicamente la función *pinta* la cual por la forma en que está programada es la misma que permite realizar la graficación de las espigas en el despliegue de las espigas de la base de datos de espigas con que se cuenta.

El primer paso en la graficación de la señal es cambiar al modo gráfico del sistema, con la ayuda de la función *inic_graf* .

Posteriormente la función *pinta* es la que se encargará del control del despliegue de la señal.

Pinta básicamente realiza la graficación de un cierto número de puntos, valor que es pasado como parámetro de dicha función, los valores de estos puntos son pasados como otro de los parámetros de la función en un vector. Se le indica también a la función las coordenadas iniciales de graficación xy. Así como una bandera en base a la cual es posible determinar si en el momento de graficar los valores deberá desplegar o no una etiqueta, estos valores son también parámetros de la función.

Al terminar el despliegue de la traza, se procede a cerrar el modo gráfico y regresar al modo texto, con la ayuda de la función *fin_graf*.

6) ALMACENAMIENTO EN DISCO (Salvar)

El almacenamiento de la información está a cargo de la función *salvar*. Esta función nos va a permitir salvar la información que tengamos almacenada en memoria.

Se tienen disponibles tres formatos de archivo de salida diferentes:

- EXTRACEL
- CLASIF
- DISCOVERY

El formato que sea elegido depende de en qué aplicación posterior se desee utilizar la traza que haya sido generada con el presente sistema.

20001	*Tamaño del vector de la traza*
3	*Número de espigas que contiene la traza*
1	10 1501 1634 1891 3918 7542 8036 8444 8666 9009 9374
Identificador de la espiga.	*Número de veces que se presenta la espiga en el tren.* *Tiempos en los que se presenta la espiga en la traza.*
15 9 8413 9005 9347 10644 10977 11551 11897 12218 12486	*(ídem)*
25 4 5558 5734 6610 7512	*(ídem)*
0	*Si a la espiga se le adicionó ruido este valor es igual a 1 (uno) y el siguiente valor que se presente es el de la varianza asociada al ruido que se agregó. En éste no se agregó ruido a la señal por lo que el valor es 0 (cero).*
esp: 1(10) 15(9) 25(4) &&	*Comentario acerca del archivo actual. && marca de fin de comentario, debe estar separada del texto por un espacio.*

Recuadro A. Ejemplo del Contenido de un archivo descriptor.

Para archivos a ser analizados con ayuda de CLASIF [Quiza 91] la información de la traza es almacenada en un archivo binario.

Los archivos para ser usados en EXTRACEL y en DISCOVERY son archivos de texto.

Esta función requiere la entrada del nombre del archivo en el cual se va a almacenar la información. Y automáticamente genera el del archivo en el cual se va a almacenar el archivo de control a partir del nombre que se le proporcionó, asignándole la extensión .DES.

El nombre de los archivos por convención usan las siguientes extensiones: EXT para EXTRACEL, DAT para CLASIF y TXT para DISCOVERY.

En general se le puede asignar cualquier extensión a los archivos que se generen si así se desea. Siempre y cuando no sea la del archivo de control (DES).

Una vez que se tiene el identificador del archivo, entonces se determina el identificador para el archivo de control y se salvan los datos de (Ver Recuadro A.) :

- Número de puntos del vector asociado a la traza
- Número de espigas en la traza

Para cada espiga :

- Identificador de la espiga, número de veces que se presenta en el tren, y tiempos de las ocurrencias.

Finalmente :

- Si se agregó ruido a la señal (inserta un uno en el archivo), o no (inserta un cero en el archivo), y si se agregó ruido, que varianza se usó.

- Así como un comentario, que permita posteriormente determinar los criterios que se tomaron en cuenta al generar la traza.

7) TRASLAPE DE LAS SEÑALES (Traslapar Espigas)

El traslape de las espigas es el despliegue de cada una de las espigas que serán insertadas en el tren a partir de una misma referencia. De ésta forma las espigas quedan una sobre otra lo que facilita la apreciación de las semejanzas y diferencias entre éstas.

La función *graf_esps_sel* es la encargada de este proceso.

Muestra además, el número de cada una de las espigas que está desplegando.

Las funciones asociadas a las gráficas están contenidas en el archivo *grafica.c*.

El proceso cambia inicialmente a modo gráfico con la función *inic_graf*, y al finalizar el despliegue regresa al modo texto mediante la ejecución de la función *fin_graf*.

8) INFORMACIÓN DE LA TRAZA

La presentación de la información relativa a la traza está a cargo de la función *info* y presenta :

- Longitud de la traza en milisegundos

- Número de espigas en la traza

- El comentario asociado a la traza, que permite conocer los criterios que se tomaron en cuenta al generar la traza.

Para cada espiga :

- Identificador de la espiga, número de veces que se presenta en el tren, y tiempos de las ocurrencias.

Finalmente :

- Si se agregó ruido a la señal (inserta un uno en el archivo), o no (inserta un cero en el archivo), y si se agregó ruido, que varianza se usó.

- Esta información es más precisa que si sólo se observara el tren generado, ya que aunque se cuenta con una escala, se pueden conocer los tiempos de ocurrencia de las espigas pero sólo de una manera aproximada.

9) SALIDA DEL SISTEMA (Salir)

Esta opción es la que nos permite finalizar el programa, y antes de terminar la ejecución libera la memoria que estaba ocupada, con ayuda de la función *libera*.

CAPÍTULO 5

EJEMPLOS DE TRENES OBTENIDOS CON EL GENERADOR

La finalidad de este trabajo es contar con una herramienta que nos permita generar trenes de espigas, que cumplan con ciertas características. Las cuales serán determinadas por el usuario.

A fin de ilustrar la manera en que funciona el programa se generaron una serie de trenes. De esta manera será más fácil visualizar los resultados generados.

La forma en que se presentan los ejemplos es la siguiente :

* La primera imagen, corresponde a la gráfica de traslape de espigas, la cual permite observar las diferencias existentes entre las espigas incluidas en el tren. Aquí el punto de referencia para la comparación son las espigas mismas, ya que las espigas son graficadas bajo un mismo punto de referencia.

En base a las diferencias existentes, es que podemos estimar si una separación posterior del tren será factible o no.

** La imagen siguiente, corresponde al tren generado. Y nos permite visualizar la distribución de las espigas en el tren de impulsos.

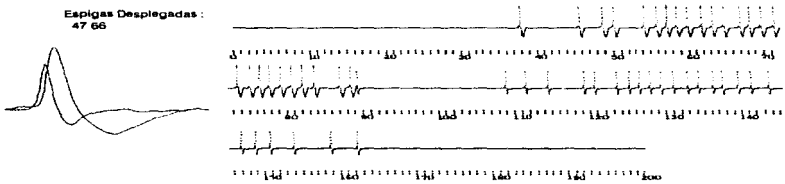
*** En la página opuesta se presenta la descripción correspondiente a cada ejemplo. Aquí se presenta la justificación y características del tren generado, así como el contenido del archivo descriptor, correspondiente a dicho ejemplo.

Todas las simulaciones fueron realizadas para trenes de 200 milisegundos de longitud; los ejemplos 1 al 26, fueron generados de manera aleatoria; los ejemplos 27 al 29 fueron generados con tiempos predeterminados y finalmente los ejemplos 30 al 34 son trenes a los cuales se les agregó ruido.

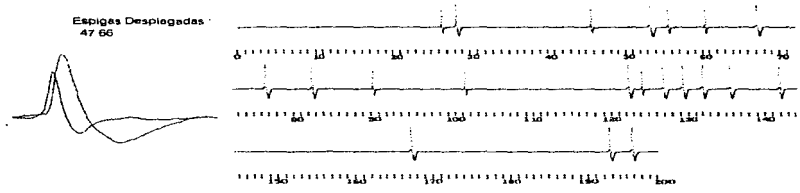
Se utilizó un máximo de cinco espigas (tomadas de la base de señales mostrada en el apéndice A) y un mínimo de dos para generar los trenes. Los ejemplos 1 al 9, 28 y 29 fueron generados en base a dos espigas; los ejemplos 10 al 21, 27 y 30 al 34 se creron en base a tres espigas; los ejemplos 22 al 24 consideran cuatro espigas en su generación y finalmente para los ejemplos 25 y 26 se seleccionaron cinco espigas.

5.1 EJEMPLOS

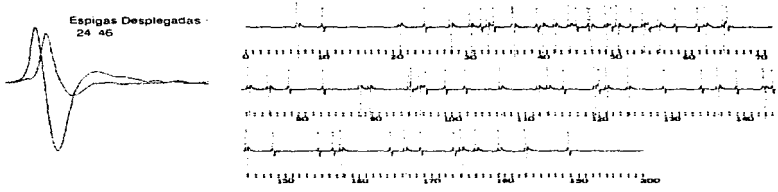
EJEMPLO 1



EJEMPLO 2



EJEMPLO 3



EJEMPLO 1

Espigas con amplitudes diferentes, con forma de onda semejante.
Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 2
Espiga No. 47 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 25
89.10 108.70 111.26 114.27 119.02 120.44 123.13 124.79 126.30 127.71 129.16 130.77 132.54 133.96 136.02 137.33 139.35 141.11 143.35 146.74 148.59 150.60 153.67 158.62 162.05
Espiga No. 66 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 26
37.64 45.41 48.51 49.84 54.01 55.66 56.99 58.27 59.64 61.46 63.00 64.58 66.67 67.98 69.51 71.09 73.33 74.87 76.22 77.52 79.07 80.44 81.78 83.38 86.84 88.22

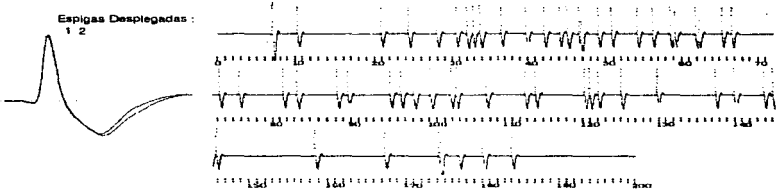
EJEMPLO 2

Una segunda simulación en base a las espigas usadas en el tren anterior.
Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 2
Espiga No. 47 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 7
26.25 45.57 55.49 60.28 90.17 102.18 125.00
Espiga No. 66 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 14
28.06 53.15 66.98 76.45 82.38 123.19 127.69 130.10 132.80 136.33 142.73 168.14 193.45 196.57

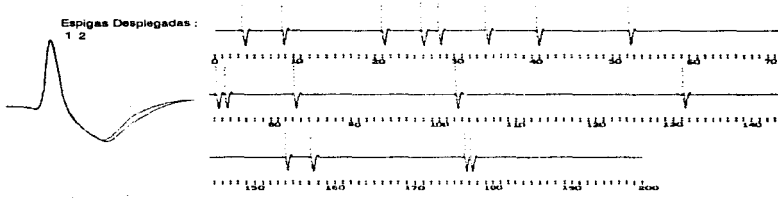
EJEMPLO 3

Espigas con amplitudes y formas diferentes. Espiga 24 mucho más grande que la espiga 46.
Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 2
Espiga No. 24 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 37
6.87 20.70 27.76 30.38 32.67 36.29 40.19 41.94 43.90 46.65 48 87 50.57 54.05 56.04 62.66 65.15 73.07 75.66 88.05 89.41 94.89 96.25 107.35 112.55 113.87 120.10 121.88 124.90 137.31 143.68 145.72 158.44 167.25 174.91 176.84 180.02 183.86
Espiga No. 46 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 36
6.79 10.27 24.20 31.85 33.55 36.18 39.63 44.88 46.36 49.73 52.48 54.82 58.43 61.63 65.74 78.50 83.24 88.06 95.30 97.17 99.98 102.65 116.11 120.78 129.70 135.07 139.82 143.56 144.87 149.17 155.49 157.38 165.37 169.41 174.05 189.62

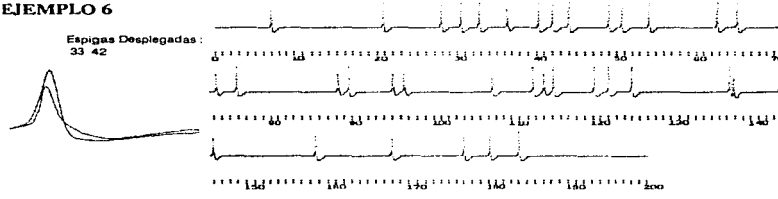
EJEMPLO 4



EJEMPLO 5



EJEMPLO 6



EJEMPLO 4

Dos espigas con amplitud y forma de pico semejantes. Valles ligeramente diferentes.
Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 2

Espiga No. 1 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 44
6.87 10.27 20.70 24.20 27.76 30 38 32.67 36.29 39.63 41.94 43.90 46.65 48.87 50.57 54.05 56.04 59.02 62.05 65.15 66.50 73.07 75.66 81.31 88.05 89.41 94.89 96.25 103.44 107.35 112.55 113.87 120.10 121.88 124.90 129.70 137.31 143.68 145.72 158.44 167.25 174.30 176.84 180.02 183.86

Espiga No. 2 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 15
6.79 31.85 33.55 44.88 46.36 58.43 61.63 83.24 97.89 99.98 102.65 120.78 139.82 144.87 174.05

EJEMPLO 5

Segunda simulación en base a las dos espigas anteriores.
Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 2

Espiga No. 1 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9
8.35 27.65 33.84 52.05 74.17 132.03 154.89 158.02 177.23

Espiga No. 2 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 8
3.32 20.70 25.58 40.40 73.06 83.00 103.06 177.96

EJEMPLO 6

Espigas con forma y amplitudes diferentes.
Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 2

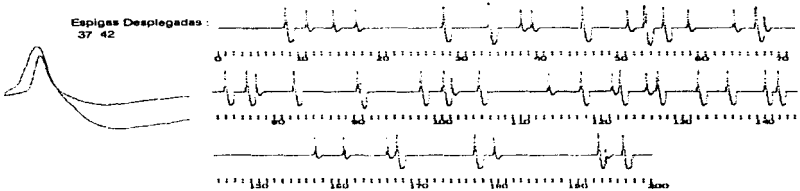
Espiga No. 33 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 28
6.87 20.70 27.76 30.38 32.67 40.19 41.94 43.90 48.87 50.57 54.05 62.66 65.15 75.66 89.41 107.35 112.55 115.08 120.10 121.88 124.90 137.31 143.68 158.44 167.84 176.84 180.02 183.86

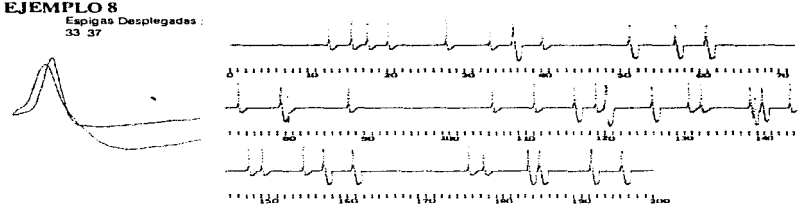
Espiga No. 42 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9
36.29 73.07 88.05 94.89 96.25 113.87 137.73 144.27 145.72

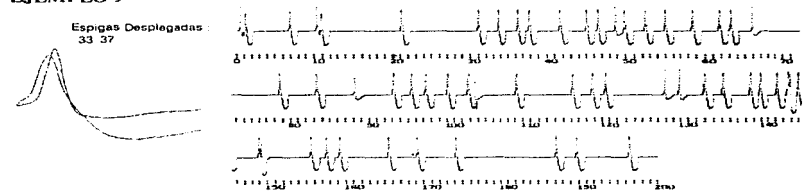
EJEMPLO 7



EJEMPLO 8



EJEMPLO 9



EJEMPLO 7

Dos espigas con forma semejante y amplitudes diferentes.
Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 2
Espiga No. 37 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 25
8.35 28.06 33.84 45.57 53.15 55.49 66.86 73.84 76.45 82.38 90.33 98.20 101.04 105.53 118.34 123.19 127.69 132.93 136.33 140.97 142.73 168.14 177.96 193.45 196.57
Espiga No. 42 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 22
11.01 14.28 17.14 33.75 37.94 39.34 51.18 53.40 58.65 64.16 67.86 77.60 90.68 102.06 114.44 122.22 126.36 158.02 161.52 166.92 180.32 194.29

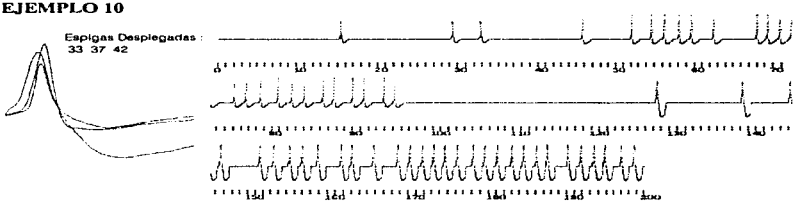
EJEMPLO 8

Dos espigas con forma y amplitudes diferentes.
Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 2
Espiga No. 33 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 24
12.49 15.42 17.36 20.08 27.50 33.10 36.16 40.03 73.92 80.15 88.03 106.39 112.00 119.69 121.15 131.49 133.09 139.58 144.38 148.23 150.02 155.30 176.40 178.29
Espiga No. 37 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 16
36.37 50.99 56.72 60.69 79.35 116.95 120.94 126.67 139.02 140.51 157.79 161.54 184.22 185.64 192.09 196.03

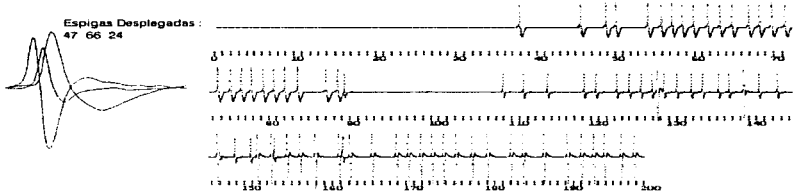
EJEMPLO 9

Espigas con forma y amplitud diferente.
Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 2
Espiga No. 33 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 11
0.42 10.27 0.00 48.63 65.74 88.06 103.44 127.83 129.70 143.81 149.17
Espiga No. 37 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 45
1.02 6.79 10.76 21.01 30.90 33.55 36.18 37.71 41.48 44.88 46.36 49.73 52.23 54.82 58.43 61.63 63.03 78.50 83.24 92.99 95.30 97.17 99.98 102.65 109.03 116.11 118.65 120.36 132.85 135.07 138.52 139.82 142.14 143.56 144.87 148.83 155.49 157.38 159.15 165.37 169.06 174.05 186.99 189.62 196.30

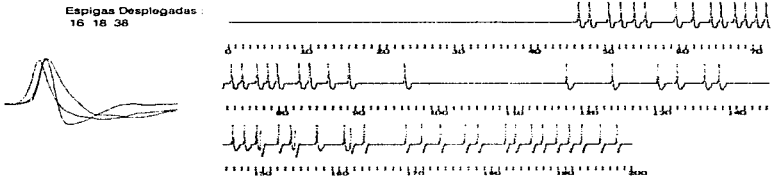
EJEMPLO 10



EJEMPLO 11



EJEMPLO 12



EJEMPLO 10

Espigas con amplitud de pico escalonada. Forma de valle diferente.
Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 3

Espiga No. 33 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 23
45.66 51.99 54.54 56.14 57.96 59.48 62.33 67.86 69.22 70.65 72.03 75.01 76.51 78.44 80.69 82.44 84.00 86.25 87.55 89.92 91.30 93.80 95.10

Espiga No. 42 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 3
15.24 29.23 32.67

Espiga No. 37 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 31
128.36 139.18 145.02 146.48 151.46 153.27 155.25 156.85 158.71 161.56 162.93 165.65 168.60 170.08 171.68 173.10 174.52 176.09 178.02 179.44 180.96 182.87 184.34 185.96 187.29 190.04 191.69 193.30 194.65 196.83 198.56

EJEMPLO 11

Simulación en base al ejemplo 1. Se agregó una espiga al tren (24), cuya amplitud de valle es mucho mayor que la de las espigas 47 y 66.

Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 3

Espiga No. 47 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 25
89.10 108.70 111.26 114.27 119.02 120.44 123.13 124.79 126.30 127.71 129.16 130.77 132.54 133.96 136.02 137.33 139.35 141.11 143.35 146.74 148.59 150.60 153.67 158.62 162.05

Espiga No. 66 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 26
37.64 45.41 48.51 49.84 54.01 55.66 56.99 58.27 59.64 61.46 63.00 64.58 66.67 67.98 69.51 71.09 73.33 74.87 76.22 77.52 79.07 80.44 81.78 83.38 86.84 88.22

Espiga No. 24 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 28
128.36 139.18 146.48 151.46 153.27 155.25 156.85 158.71 161.56 162.93 165.67 168.60 170.08 171.68 173.10 174.52 176.09 178.02 179.86 182.87 184.34 186.96 190.04 191.81 193.30 194.65 196.83 198.56

EJEMPLO 12

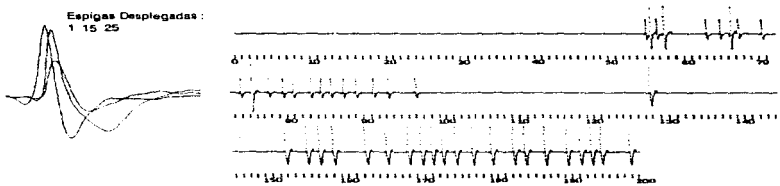
Espigas con amplitud, tiempo de pico y valle semejantes; forma de los valles diferente.
Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 16 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 24
46.28 47.68 50.24 51.98 53.83 55.32 59.33 61.60 64.06 65.41 67.37 68.69 70.20 71.48 73.33 74.75 76.84 78.30 79.61 82.49 83.81 86.20 88.95 96.16

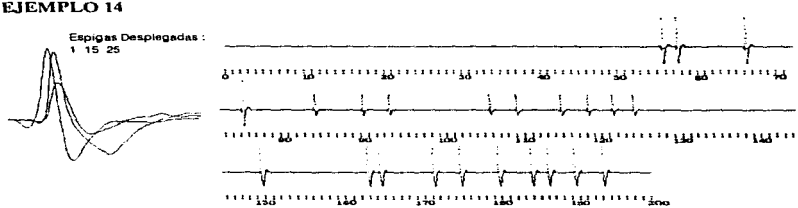
Espiga No. 18 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 19
150.52 152.96 155.11 162.34 164.12 169.55 171.65 174.15 177.40 179.04 182.73 184.06 186.22 187.64 189.66 191.38 193.04 195.47 197.77

Espiga No. 38 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 12
117.87 123.99 130.21 132.73 136.43 138.37 146.66 148.26 149.97 154.58 157.92 161.51

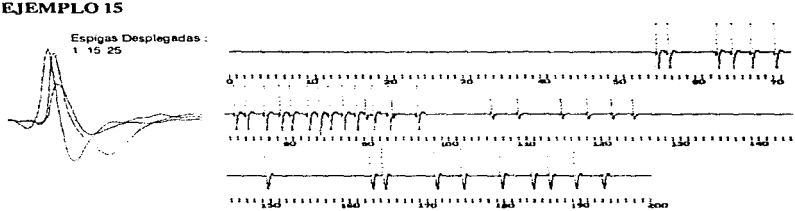
EJEMPLO 13



EJEMPLO 14



EJEMPLO 15



EJEMPLO 13

Dos espigas semejantes en amplitud (1 y 25), diferentes en tiempo de ocurrencia de los valles. La tercera espiga (15) es marcadamente más pequeña.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 1 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 22
128.36 152.50 155.25 156.85 158.71 162.93 165.67 168.60 170.30 171.68 173.10 174.93 177.22 179.86 182.87 184.34 186.96 190.04 191.81 193.30 194.65 198.56

Espiga No. 15 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 19
54.95 56.40 62.74 64.65 67.11 70.12 73.54 75.00 77.39 79.28 80.61 82.85 84.21 85.52 87.18 88.85 91.06 93.02 96.72

Espiga No. 25 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 4
55.58 57.34 66.10 75.12

EJEMPLO 14

Segunda simulación en base a las espigas utilizadas en el tren anterior. Presenta un bajo porcentaje de ocurrencias de espigas en el tren.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 1 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 10
150.30 163.57 165.15 171.86 175.42 180.36 184.44 186.66 190.09 193.74

Espiga No. 15 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9
84.13 90.05 93.47 106.44 109.77 115.51 118.97 122.18 124.86

Espiga No. 25 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 4
55.58 57.34 66.10 75.12

EJEMPLO 15

Tercera simulación en base a las espigas 1, 15 y 25. Se observan ráfagas de las dos espigas mayores, con algunas ocurrencias de la espiga pequeña(15), intercaladas.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 1 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 10
150.30 163.57 165.15 171.86 175.42 180.36 184.44 186.66 190.09 193.74

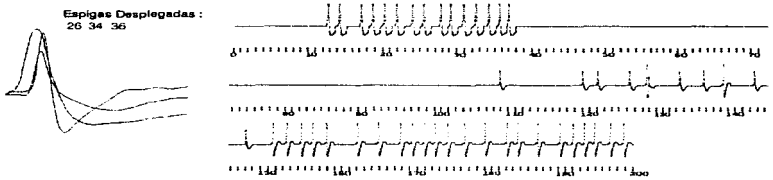
Espiga No. 15 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9
84.13 90.05 93.47 106.44 109.77 115.51 118.97 122.18 124.86

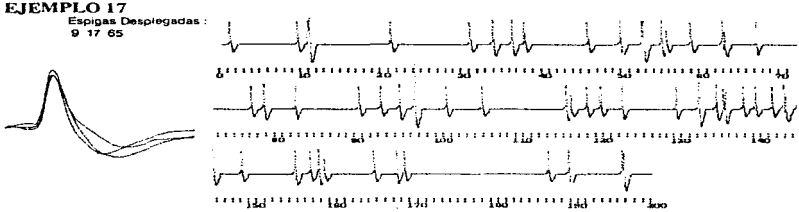
Espiga No. 25 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 19
54.95 56.40 62.74 64.65 67.11 70.12 73.54 75.00 77.39 79.28 80.61 82.85 84.21 85.52 87.18 88.85 91.06 93.02 96.72

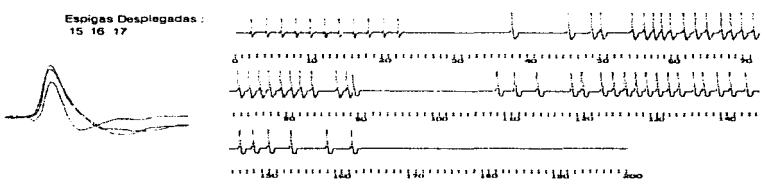
EJEMPLO 16



EJEMPLO 17



EJEMPLO 18



EJEMPLO 16

Espigas con amplitud de valles semejantes, formas diferentes. Dos espigas con amplitud de pico semejante (26 y 36); la tercera con amplitud de pico menor (34).

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 26 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 25

128.36 139.18 151.46 153.27 155.25 156.85 158.71 162.93 165.67 168.60 170.08 171.68 173.10 174.93 177.22 179.86 182.87 184.34 186.96 190.04 191.81 193.30 194.65 196.83 198.56

Espiga No. 34 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9

108.67 119.84 121.87 126.10 128.68 133.21 136.39 143.38 147.90

Espiga No. 36 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 15

12.53 14.07 17.02 18.51 20.04 21.71 23.77 25.25 27.61 29.00 30.71 32.27 33.95 35.48 36.76

EJEMPLO 17

Dos espigas con la misma forma de pico, diferente valle. Tercera espiga (9), con una amplitud un poco mayor y valle semejante a una de las anteriores(65).

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 9 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 17

10.95 33.67 36.18 49.73 52.48 54.82 62.68 78.50 95.30 97.17 116.11 135.07 143.56 145.31 155.49 158.49 189.62

Espiga No. 17 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 33

1.02 9.38 10.76 21.01 30.90 37.71 45.57 52.23 55.49 58.69 63.03 76.88 90.33 92.99 97.21 101.04 105.53 116.74 118.65 120.36 132.85 135.97 138.52 140.04 142.14 143.89 148.83 157.38 159.15 165.37 169.06 186.99 196.30

Espiga No. 65 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 8

66.98 82.38 123.19 130.10 132.93 136.33 168.14 196.57

EJEMPLO 18

Dos espigas muy semejantes en forma y amplitud (16 y 17). Tercera espiga más pequeña (15).

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 15 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 11

2.00 4.00 6.00 8.00 10.00 12.00 14.00 16.00 18.00 20.00 22.00

Espiga No. 16 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 25

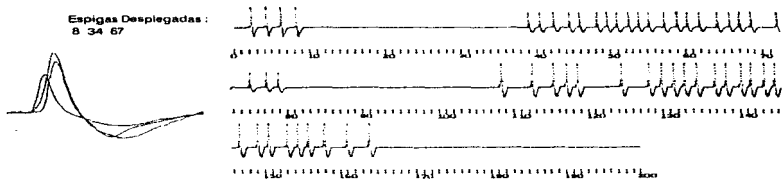
89.10 108.70 111.26 114.27 119.02 120.44 123.13 124.79 126.30 127.71 129.16 130.77 132.54 133.96 136.02 137.33 139.35 141.11 143.35 146.74 148.59 150.60 153.67 158.62 162.05

Espiga No. 17 (tiempo de las ocurrencias en msegs)

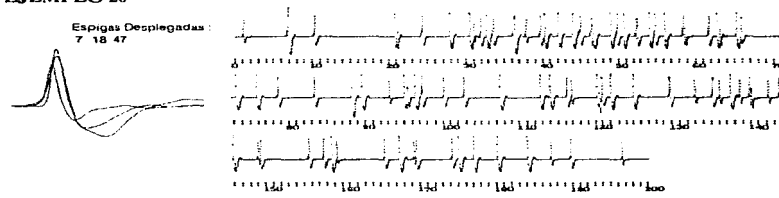
Número de repeticiones de la espiga : 26

37.64 45.41 48.51 49.84 54.01 55.66 56.99 58.27 59.64 61.46 63.00 64.58 66.67 67.98 69.51 71.09 73.33 74.87 76.22 77.52 79.07 80.44 81.78 83.38 86.84 88.22

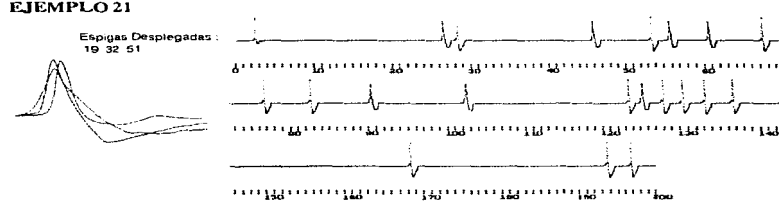
EJEMPLO 19



EJEMPLO 20



EJEMPLO 21



EJEMPLO 19

Dos espigas con ligeras diferencias en amplitud de valle y pico (8 y 67). Una tercera totalmente diferente a las anteriores (34).

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 8 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 26

108.21 112.25 115.12 116.88 118.37 124.25 128.03 129.70 131.30 132.68 134.24 136.61 138.05 139.96 141.27 142.99 144.35 146.57 148.89 150.35 152.83 154.21 155.57 157.73 160.82 163.73

Espiga No. 34 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 23

38.77 40.27 42.16 44.27 45.67 48.02 49.35 50.64 52.19 53.60 55.01 57.06 58.74 60.02 61.60 63.92 65.46 66.76 68.27 71.62 74.87 77.06 78.65

Espiga No. 67 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 4

2.00 4.00 6.00 8.00

EJEMPLO 20

Tres espigas diferentes. Una de amplitud muy grande (7), una segunda (18) de mediana amplitud y una tercera (47) muy pequeña.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 7 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 37

6.87 20.70 27.76 30.38 32.67 36.29 40.19 41.94 43.90 46.65 48.87 50.57 54.05 56.04 62.66 65.15 73.07 75.66 88.05 89.41 94.89 96.25 107.35 112.55 113.87 120.10 121.88 124.90 137.31 143.68 145.72 158.44 167.25 174.91 176.84 180.02 183.86

Espiga No. 18 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 36

6.79 10.27 24.20 31.85 33.55 36.18 39.63 44.88 46.36 49.73 52.48 54.82 58.43 61.63 65.74 78.50 83.24 88.06 95.30 97.17 99.98 102.65 116.11 120.78 129.70 135.07 139.82 143.56 144.87 149.17 155.49 157.38 165.37 169.41 174.05 189.62

Espiga No. 47 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 21

1.02 21.01 30.90 37.71 52.23 58.69 63.03 92.99 116.74 120.36 132.85 135.97 138.52 140.04 142.14 143.89 148.83 159.15 169.06 186.99 196.30

EJEMPLO 21

Tres espigas con forma de onda diferente. Amplitudes semejantes.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 19 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 1

2.29

Espiga No. 32 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 7

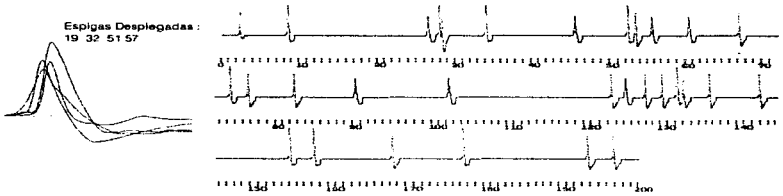
26.25 45.57 55.49 60.28 90.17 102.18 125.00

Espiga No. 51 (tiempo de las ocurrencias en msegs)

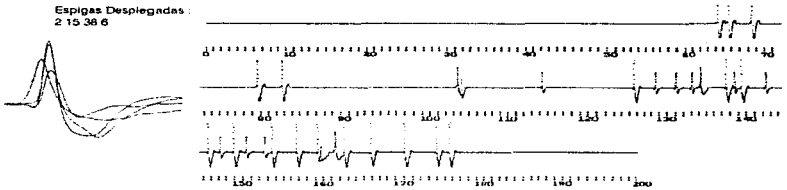
Número de repeticiones de la espiga : 14

28.06 53.15 66.98 76.45 82.38 123.19 127.69 130.10 132.80 136.33 142.73 168.14 193.45 196.57

EJEMPLO 22



EJEMPLO 23



EJEMPLO 22

Segunda simulación en base al tren anterior. Se realizó la inserción de una cuarta espiga (57) en dicha traza.

Tamaño de la traza : 200.0 msegs.
Número de espigas que contiene : 4

Espiga No. 19 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 1
2.29

Espiga No. 32 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 7
26.25 45.57 55.49 60.28 90.17 102.18 125.00

Espiga No. 51 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 14
28.06 53.15 66.98 76.45 82.38 123.19 127.69 130.10 132.80 136.33 142.73 168.14 193.45 196.57

Espiga No. 57 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 9
8.35 27.65 33.84 52.05 74.17 132.03 154.89 158.02 177.23

EJEMPLO 23

Tres amplitudes de pico diferentes. Dos espigas (2 y 6) con amplitud de pico semejante, con forma de valles diferentes. Las dos espigas restantes (15 y 38) con amplitud de valle semejante, amplitud de pico diferente.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 4

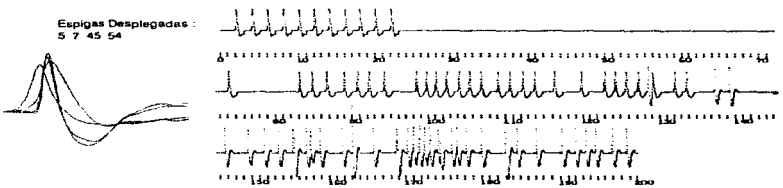
Espiga No. 2 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 13
126.40 138.30 140.17 146.39 149.68 154.26 157.49 160.09 163.28 166.66 170.85 174.67 176.37

Espiga No. 15 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 10
104.81 115.09 129.34 131.87 133.81 139.32 143.13 147.98 151.17 153.39

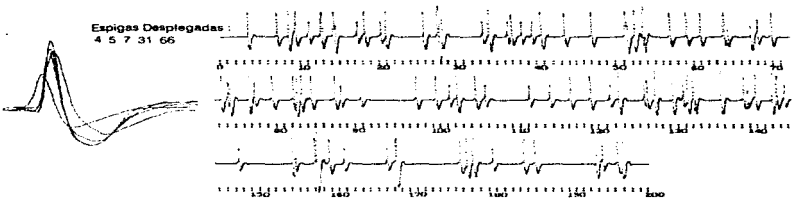
Espiga No. 38 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 4
104.34 135.03 160.46 162.25

Espiga No. 6 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 5
4.10 65.41 68.25 79.40 82.49

EJEMPLO 24



EJEMPLO 25



EJEMPLO 24

Dos espigas con forma semejante (5 y 7). Dos espigas con amplitudes semejantes, tiempos de ocurrencia de los picos diferente (45 y 54).

Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 4

Espiga No. 5 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 28
128.36 139.18 146.48 151.46 153.27 155.25 156.85 158.71 161.56 162.93 165.67 168.60 170.08 171.68 173.10 174.52 176.09 178.02 179.86 182.87 184.34 186.96 190.04 191.81 193.30 194.65 196.83 198.56

Espiga No. 7 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 11
137.28 149.39 155.09 157.42 162.95 168.76 170.87 172.36 174.22 176.79 182.83

Espiga No. 45 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 28
74.06 83.03 84.74 86.66 88.85 90.68 92.17 94.11 98.33 99.70 100.99 102.41 104.13 105.54 106.94 108.96 110.46 112.19 113.79 116.22 119.69 122.91 124.27 125.59 127.03 128.99 132.00 133.56

Espiga No. 54 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 11
2.00 4.00 6.00 8.00 10.00 12.00 14.00 16.00 18.00 20.00 22.00

EJEMPLO 25

Espiga 31, muy pequeña y con forma diferente. Las cuatro espigas restantes con forma semejante. Espiga 4 con amplitud de pico mayor.

Tamaño de la traza : 200.0 msegs.
Número de espigas que contiene : 5

Espiga No. 4 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 14
28.06 53.15 66.98 76.45 82.38 123.19 127.69 130.10 132.80 136.33 142.73 168.14 193.45 196.57

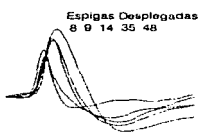
Espiga No. 5 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 9
8.35 27.65 33.84 52.05 74.17 132.03 154.89 158.02 177.23

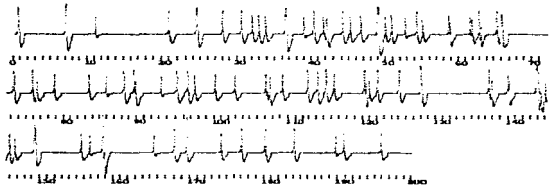
Espiga No. 7 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 8
3.32 20.70 25.58 40.40 73.06 83.00 103.06 177.96

Espiga No. 31 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 35
11.01 12.52 14.28 17.14 20.08 27.50 33.10 36.16 37.94 39.34 51.18 53.40 64.16 67.86 73.92 77.17 88.03 90.58 102.06 106.39 112.00 114.44 122.22 126.36 131.49 133.09 139.58 144.38 148.23 155.30 158.02 161.52 166.92 180.32 194.29

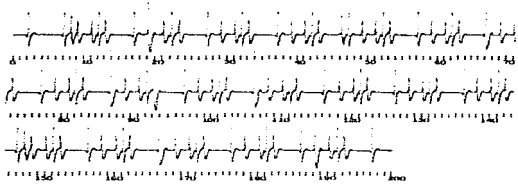
Espiga No. 66 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 35
6.91 9.25 12.60 14.08 18.03 27.40 36.60 43.14 46.82 50.99 56.72 58.89 60.69 63.67 69.61 79.35 81.93 84.24 87.20 97.59 100.21 105.19 116.95 119.69 126.67 139.02 140.51 143.27 157.77 159.50 167.96 176.40 184.22 185.64 196.03

EJEMPLO 26





EJEMPLO 27



EJEMPLO 26

Espiga 48, más pequeña con valle diferente al de las cuatro restantes. Espiga 14 pico con mayor amplitud. Espigas 8 y 9 amplitudes semejantes.

Tamaño de la traza : 200.0 msegs.

Número de espigas que contiene : 5

Espiga No. 8 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 28

6.87 20.70 27.76 30.38 32.67 40.19 41.94 43.90 48.87 50.57 54.05 62.66 65.15 75.66 89.41 107.35 112.55 115.08 120.10 121.88 124.90 137.31 143.68 158.44 167.84 176.84 180.02 183.86

Espiga No. 9 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9

36.29 73.07 88.05 94.89 96.25 113.87 137.73 144.27 145.72

Espiga No. 14 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 7

0.42 24.20 48.63 65.74 127.83 143.81 149.17

Espiga No. 35 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 31

6.79 31.85 33.55 36.18 41.48 44.88 46.36 49.73 54.82 58.43 61.63 76.24 78.50 83.24 89.69 97.17 99.98 102.65 106.84 109.03 116.11 120.78 139.82 144.87 146.48 155.49 158.49 169.41 174.05 190.60 195.67

Espiga No. 48 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 15

10.95 38.78 52.48 58.69 61.39 63.03 85.70 92.99 120.36 140.04 156.65 158.77 165.37 169.41 189.62

EJEMPLO 27

Espigas con forma y amplitudes diferentes. Espiga 3 (de mediana amplitud), se presenta siguiendo los tiempos proporcionados de manera aperiódica. Espigas 23 y 30 tienen un comportamiento periódico de 5 y 10 milisegundos respectivamente. La espiga 23 (la de mayor amplitud), presenta únicamente un disparo por periodo y la primera ocurrencia de la espiga en el tren es en el milisegundo 2; la espiga 30 (de menor amplitud), presenta ráfagas compuestas por tres impulsos cada una, que se repiten periódicamente. El disparo inicial ocurre en el milisegundo 9 y los siguientes cada dos milisegundos. Esto es en los milisegundos 11 y 13 respectivamente.

Tamaño de la traza : 200.0 msegs.

Número de espigas que contiene : 3

Espiga No. 3 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 10

8.00 19.00 46.00 67.00 87.00 93.00 107.00 148.00 167.00 189.00

Espiga No. 23 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 40

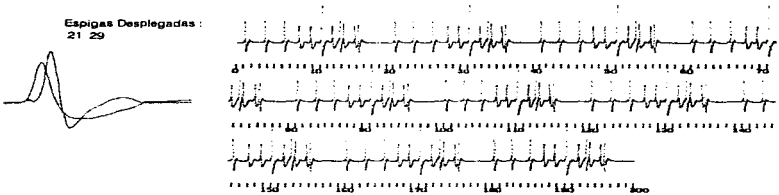
2.00 7.00 12.00 17.00 22.00 27.00 32.00 37.00 42.00 47.00 52.00 57.00 62.00 67.00 72.00 77.00 82.00 87.00 92.00 97.00 102.00 107.00 112.00 117.00 122.00 127.00 132.00 137.00 142.00 147.00 152.00 157.00 162.00 167.00 172.00 177.00 182.00 187.00 192.00 197.00

Espiga No. 30 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 57

9.00 11.00 13.00 19.00 21.00 23.00 29.00 31.00 33.00 39.00 41.00 43.00 49.00 51.00 53.00 59.00 61.00 63.00 69.00 71.00 73.00 79.00 81.00 83.00 89.00 91.00 93.00 99.00 101.00 103.00 109.00 111.00 113.00 119.00 121.00 123.00 129.00 131.00 133.00 139.00 141.00 143.00 149.00 151.00 153.00 159.00 161.00 163.00 169.00 171.00 173.00 179.00 181.00 183.00 189.00 191.00 193.00

EJEMPLO 28



EJEMPLO 29

EJEMPLO 28

Dos espigas diferentes. Espiga 21 (la más grande), se presenta en ráfagas compuestas por 3 impulsos cada una, con una separación de 2.5 milisegundos entre ellos, y con un comportamiento periódico. Tiempos de ocurrencia para el primer periodo : 1, 3.5 y 6 milisegundos, periodo igual a 10 milisegundos. Espiga 29 (la más pequeña). Se presenta en ráfagas periódicas compuestas, cada una de ellas, por 6 impulsos a una distancia de 1.5 milisegundos entre ellas, tiempo inicial de disparo de ésta espiga : milisegundo 8; periodo de repetición de las ráfagas : 20 milisegundos.

Tamaño de la traza : 200.0 msegs.

Número de espigas que contiene : 2

Espiga No. 21 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 60

1.00 3.50 6.00 11.00 13.50 16.00 21.00 23.50 26.00 31.00 33.50 36.00 41.00 43.50 46.00 51.00 53.50 56.00 61.00 63.50 66.00 71.00 73.50 76.00 81.00 83.50 86.00 91.00 93.50 96.00 101.00 103.50 106.00 111.00 113.50 116.00 121.00 123.50 126.00 131.00 133.50 136.00 141.00 143.50 146.00 151.00 153.50 156.00 161.00 163.50 166.00 171.00 173.50 176.00 181.00 183.50 186.00 191.00 193.50 196.00

Espiga No. 29 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 60

8.00 9.50 11.00 12.50 14.00 15.50 28.00 29.50 31.00 32.50 34.00 35.50 48.00 49.50 51.00 52.50 54.00 55.50 68.00 69.50 71.00 72.50 74.00 75.50 88.00 89.50 91.00 92.50 94.00 95.50 108.00 109.50 111.00 112.50 114.00 115.50 128.00 129.50 131.00 132.50 134.00 135.50 148.00 149.50 151.00 152.50 154.00 155.50 168.00 169.50 171.00 172.50 174.00 175.50 188.00 189.50 191.00 192.50 194.00 195.50

EJEMPLO 29

Espiga 69 (la más grande), se presenta en ráfagas periódicas, con ocho impulsos cada una ellas, los impulsos están a una distancia de 1.5 milisegundos y las ráfagas se presentan cada 30 milisegundos, tiempo inicial de disparo : milisegundo 1. La espiga más pequeña (espiga 70), presenta también ráfagas con un periodo de 30 milisegundos, compuestas, cada una de ellas, por 5 impulsos a una distancia de 2 milisegundos entre ellos. La primer ráfaga se presenta a partir del milisegundo 13.

Tamaño de la traza : 200.0 msegs.

Número de espigas que contiene : 2

Espiga No. 69 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 56

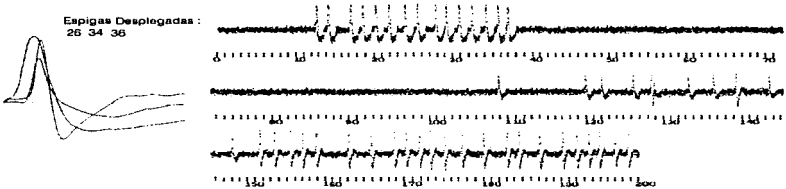
1.00 2.50 4.00 5.50 7.00 8.50 10.00 11.50 31.00 32.50 34.00 35.50 37.00 38.50 40.00 41.50 61.00 62.50 64.00 65.50 67.00 68.50 70.00 71.50 91.00 92.50 94.00 95.50 97.00 98.50 100.00 101.50 121.00 122.50 124.00 125.50 127.00 128.50 130.00 131.50 151.00 152.50 154.00 155.50 157.00 158.50 160.00 161.50 181.00 182.50 184.00 185.50 187.00 188.50 190.00 191.50

Espiga No. 70 (tiempo de las ocurrencias en msegs)

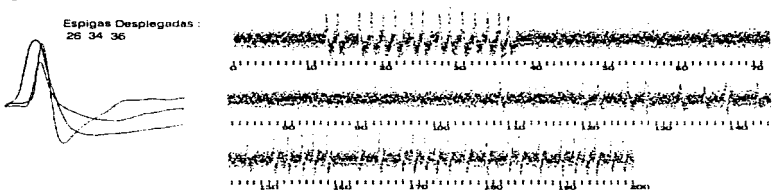
Número de repeticiones de la espiga : 33

13.00 15.00 17.00 19.00 21.00 43.00 45.00 47.00 49.00 51.00 73.00 75.00 77.00 79.00 81.00 103.00 105.00 107.00 109.00 111.00 133.00 135.00 137.00 139.00 141.00 163.00 165.00 167.00 169.00 171.00 193.00 195.00 197.00

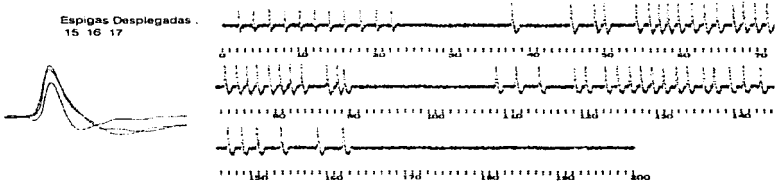
EJEMPLO 30



EJEMPLO 31



EJEMPLO 32



EJEMPLO 30

Se adicionó ruido al tren del ejemplo 16 con una varianza de 20%. El tren está compuesto por espigas con amplitudes de valles semejantes, formas diferentes. Dos de las espigas con amplitud de pico semejante (26 y 36); la tercera con amplitud de pico menor (34).

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 26 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 25

128.36 139.18 151.46 153.27 155.25 156.85 158.71 162.93 165.67 168.60 170.08 171.68 173.10 174.93 177.22 179.86 182.87 184.34 186.96 190.04 191.81 193.30 194.65 196.83 198.56

Espiga No. 34 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9

108.67 119.84 121.87 126.10 128.68 133.21 136.39 143.38 147.90

Espiga No. 36 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 15

12.53 14.07 17.02 18.51 20.04 21.71 23.77 25.25 27.61 29.00 30.71 32.27 33.95 35.48 36.76

EJEMPLO 31

Al igual que en el ejemplo anterior, se adicionó ruido al tren del ejemplo 16 pero en este caso con una varianza del 50%, para observar el efecto de la varianza seleccionada sobre el nivel de ruido de la señal.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 26 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 25

126.36 139.18 151.46 153.27 155.25 156.85 158.71 162.93 165.67 168.60 170.08 171.68 173.10 174.93 177.22 179.86 182.87 184.34 186.96 190.04 191.81 193.30 194.65 196.83 198.56

Espiga No. 34 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 9

108.67 119.84 121.87 126.10 128.68 133.21 136.39 143.38 147.90

Espiga No. 36 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 15

12.53 14.07 17.02 18.51 20.04 21.71 23.77 25.25 27.61 29.00 30.71 32.27 33.95 35.48 36.76

EJEMPLO 32

En éste caso se agregó ruido a la traza del ejemplo 18, con una varianza del 10 %. El tren está formado por dos espigas muy semejantes en forma y amplitud (16 y 17). Tercera espiga más pequeña (15).

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 15 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 11

2.00 4.00 6.00 8.00 10.00 12.00 14.00 16.00 18.00 20.00 22.00

Espiga No. 16 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 25

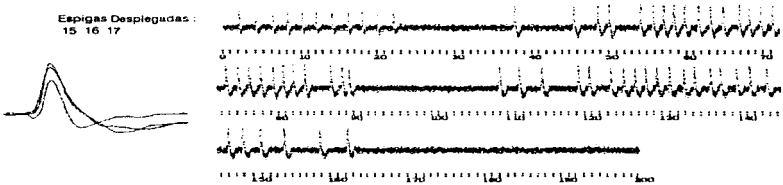
89.10 108.70 111.26 114.27 119.02 120.44 123.13 124.79 126.30 127.71 129.16 130.77 132.54 133.96 136.02 137.33 139.35 141.11 143.35 146.74 148.59 150.60 153.67 158.62 162.05

Espiga No. 17 (tiempo de las ocurrencias en msegs)

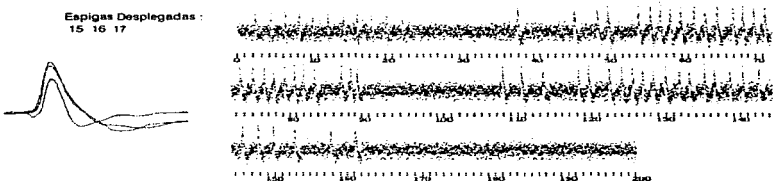
Número de repeticiones de la espiga : 26

37.64 45.41 48.51 49.84 54.01 55 66 56.99 58.27 59.64 61.46 63.00 64.58 66.67 67.98 69.51 71.09 73.33 74.87 76.22 77.52 79.07 80.44 81.78 83.38 86.84 88.22

EJEMPLO 33



EJEMPLO 34



EJEMPLO 33

Se agregó ruido a la señal del ejemplo 18, en ésta ocasión con una varianza del 20%. El tren está formado por dos espigas muy semejantes en forma y amplitud (16 y 17). Que si inicialmente era difícil separarlas, ahora con el ruido es aún más difícil diferenciarlas. Tercera espiga más pequeña (15).

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 15 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 11
2.00 4.00 6.00 8.00 10.00 12.00 14.00 16.00 18.00 20.00 22.00

Espiga No. 16 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 25
89.10 108.70 111.25 114.27 119.02 120.44 123.13 124.79 126.30 127.71 129.16 130.77 132.54 133.96 136.02 137.33 139.35 141.11 143.35 146.74 148.59 150.60 153.67 158.62 162.05

Espiga No. 17 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 26
37.64 45.41 48.51 49.84 54.01 55.66 56.99 58.27 59.64 61.46 63.00 64.58 66.67 67.98 69.51 71.09 73.33 74.87 76.22 77.52 79.07 80.44 81.78 83.38 86.84 88.22

EJEMPLO 34

Nuevamente se utilizó la traza del ejemplo 18, a la cual se le agregó ruido con una varianza del 55%. Puede observarse como el ruido puede ir degradando la señal hasta hacer prácticamente irreconocibles las espigas que lo componen.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 3

Espiga No. 15 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 11
2.00 4.00 6.00 8.00 10.00 12.00 14.00 16.00 18.00 20.00 22.00

Espiga No. 16 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 25
89.10 108.70 111.25 114.27 119.02 120.44 123.13 124.79 126.30 127.71 129.16 130.77 132.54 133.96 136.02 137.33 139.35 141.11 143.35 146.74 148.59 150.60 153.67 158.62 162.05

Espiga No. 17 (tiempo de las ocurrencias en msegs)

Número de repeticiones de la espiga : 26
37.64 45.41 48.51 49.84 54.01 55.66 56.99 58.27 59.64 61.46 63.00 64.58 66.67 67.98 69.51 71.09 73.33 74.87 76.22 77.52 79.07 80.44 81.78 83.38 86.84 88.22

CAPÍTULO 6

VALIDACIÓN DE EXTRACEL

EXTRACEL debe de generar trenes de impulsos en base a las necesidades del investigador. Los cuales deben y pueden ser analizados por programas que permitan llevar a cabo la separación y clasificación de las espigas que componen dicho tren, o bien aplicarle análisis que nos permitan tener idea del número de espigas que componen el tren en cuestión.

Como ya se ha mencionado anteriormente, los trenes generados por EXTRACEL pueden ser almacenados en diferentes formatos de manera tal que los trenes puedan ser analizados por otros paquetes, como son el CLASIF [Quiza 91], y el DISCOVERY [Serna 94].

CLASIF es un programa que permite clasificar las espigas contenidas en una traza. Está basado en una red neuronal artificial de capas múltiples, entrenada con el algoritmo de retropropagación.

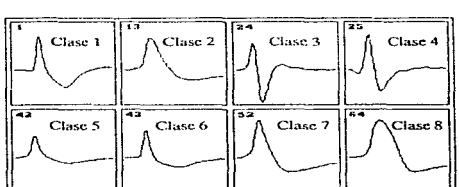


Fig. 6.1. Grupo de espigas utilizadas en el entrenamiento de la red. Se consideraron 8 clases. El número en la esquina superior izquierda corresponde al identificador de cada espiga dentro de la base de espigas utilizada. (Ver apéndice A).

Fig. 6.2. Traza generada por EXTRACEL para ser clasificada con CLASIF.

Tamaño de la traza : 200.0 msegs
Número de espigas que contiene : 2
Espiga No. 24 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 37
6.87 20.70 27.76 30.38 32.67 36.29 40.19 41.94 43.90 46.65 48.87 50.57 54.05 56.04 62.66 65.15 73.07
75.66 88.05 89.41 94.89 96.25 107.35 112.55 113.87 120.10 121.88 124.90 137.31 143.68 145.72
158.44 167.25 174.91 176.84 180.02 183.86
Espiga No. 42 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 36
6.79 10.27 24.20 31.85 33.55 36.18 39.63 44.88 46.36 49.73 52.48 54.62 58.43 61.63 65.74 78.50 83.24
88.06 95.30 97.17 99.98 102.65 116.11 120.78 129.70 135.07 139.82 143.56 144.87 149.17 155.49
157.38 165.37 169.41 174.05 189.62

Recuadro 6.1. Información de la traza generada por EXTRACEL.

Los patrones de entrenamiento utilizados se muestran en la Fig. 6.1.

Con ayuda de EXTRACEL se procedió a generar una traza, para lo cual se utilizaron las espigas 24 y 42 de la base de datos, que corresponden a las clases 3 y 5 respectivamente, de los patrones de entrenamiento. Los tiempos de ocurrencia de cada espiga, fueron generados de manera aleatoria considerando un porcentaje de concentración de impulsos del 40%.

En la Fig. 6.2 podemos observar el tren generado y las espigas que se incluyeron en el tren.

La información que describe la traza generada se muestra en el Recuadro 6.1.

La traza generada fue almacenada en formato CLASIF, y posteriormente se utilizó como entrada a dicho programa.

Los parámetros utilizados en la clasificación se muestran en el Recuadro 6.2.

Red : e36
Umbral de Ruido : 0
Límite de amplitud : 30
Puntos antes de la detección : 17

Recuadro 6.2. Parámetros del análisis a realizar en CLASIF.

Es importante mencionar que al seleccionar los parámetros a considerar en la clasificación de una traza, es necesaria la inspección visual de la señal, para así tener una idea de cuántas espigas están presentes y de los valores que dichos parámetros deben tomar.

El resultado de la clasificación con CLASIF, fue exitosa y los resultados son los mostrados en el Recuadro 6.3.

En el Recuadro 6.1., se muestran los tiempos de ocurrencia de las espigas en milisegundos, y en el Recuadro 6.3. se presentan los puntos del vector de datos en los cuales se detectaron las espigas en la señal con ayuda de CLASIF.

Podemos darnos cuenta de que 1 milisegundo para EXTRACEL, equivale a 100 puntos en CLASIF. De tal manera que, para tener la equivalencia basta con multiplicar los milisegundos de EXTRACEL por 100, para tener los puntos de CLASIF.

Recuadro 6.3. Resultado de la clasificación realizada por CLASIF. La información es presentada por columnas siguiendo el orden de aparición de cada espiga en el tren.

Clase = CL	Tiempo = T		
CL T	CL T	CL T	CL T
4 678	5 4973	3 9490	4 14355
5 1027	3 5058	3 9626	5 14487
3 2071	5 5248	5 9716	3 14573
5 2420	3 5406	5 9998	5 14917
3 2777	5 5481	5 10265	5 15549
3 3039	3 5605	3 10736	5 15738
5 3185	5 5843	3 11256	3 15844
3 3268	5 6163	3 11388	5 16537
5 3355	3 6267	5 11611	3 16726
4 3617	3 6516	3 12011	5 16941
5 3963	5 6574	5 12078	5 17405
3 4020	3 7308	3 12189	3 17492
3 4195	3 7567	3 12491	3 17685
3 4391	5 7850	5 12970	3 18003
5 4488	5 8324	5 13507	3 18387
4 4636	4 8804	3 13732	5 18962
3 4888	3 8942	5 13982	

Es decir, si una espiga ocurre en el milisegundo 45.25 en EXTRACEL, CLASIF lo detectará en el mejor de los casos en el punto 4525.

Si comparamos los resultados de CLASIF (Recuadro 6.3.) con los datos de la traza generada por EXTRACEL (Recuadro 6.1.) podemos darnos cuenta de que se clasificó correctamente un alto porcentaje de las espigas contenidas en la señal.

A excepción de 5 casos a los que se les asignó la clase 4. Si observamos las espigas clasificadas en la clase 4 ocurren en los puntos: 678, 3617, 4636, 8804 y 14355.

Y si cotejamos estos datos contra la información de la traza notamos que se trata de puntos en los cuales se presentó un traslape de las espigas 24 y 42.

En la generación de la traza, cuando se presenta un traslape las señales son sumadas, por lo que no es raro encontrar que en el traslape de estas dos espigas se presente una forma de espiga diferente a las dos originales y que haya sido clasificada dentro de una tercera clase.

Finalmente la espiga que se presenta en el milisegundo 95.30 no fue detectada por CLASIF, y si observamos la señal (Fig. 6.2)la zona en la que se presenta el impulso mencionado, es un área muy difícil, ya que presenta muchos traslapes, razón a la que se atribuye el que no haya sido detectado.

Ahora bien, ya se ha descrito, la manera en la cual se pueden realizar análisis sobre señales generadas con EXTRACEL con ayuda de CLASIF, así como los resultados que este último genera.

A continuación hablaremos acerca de DISCOVERY, que es un programa que permite hacer análisis y adquisición de señales. En este caso haremos uso de su capacidad de análisis de señales fuera de línea[1] y nos enfocaremos en el análisis FFT [2] [Discovery 94].

El análisis FFT es la transformada de una señal en particular o de una función en el dominio del tiempo, a su correspondiente en el dominio de la frecuencia. Los términos de "análisis espectral" o "espectro de potencia" son usados comúnmente para describir éste proceso.

[1] DISCOVERY permite hacer adquisición de datos, proceso conocido como en línea, y permite también hacer el análisis de dichas señales, sin embargo, este proceso no necesariamente tiene que ser realizado al mismo tiempo que los experimentos y se le conoce como análisis fuera de línea.
[2] Fast Fourier Transform, Transformada Rápida de Fourier .

La FFT de una señal "separará" las potencias de las frecuencias que componen la señal. Los resultados de la FFT son usados típicamente, para determinar las frecuencias dominantes de una señal y la potencia asociada a cada una de ellas.

La FFT es calculada en base a conjuntos de datos, donde cada conjunto debe contener un número de datos que sea potencia exacta de dos, esto se debe a la naturaleza del algoritmo que el paquete emplea en el cálculo de la FFT.

Nuevamente, con ayuda de EXTRACEL se generó una serie de señales a las que se les aplicó la FFT con DISCOVERY.

Para llevar a cabo este proceso es necesario :

* Guardar la señal generada por EXTRACEL en formato DISCOVERY.

* Importar la información de la señal con DISCOVERY, en el Módulo de *Replay - Data_Editor*. Es necesario definir los parámetros para llevar a cabo la importación de los datos.

En la opción *Define - Import_ASCII*, se presentan todos los parámetros que deben ser definidos antes de llevar a cabo la importación de la información. Se debe tener en cuenta que, el archivo en formato DISCOVERY, no tiene más que la información que describe la

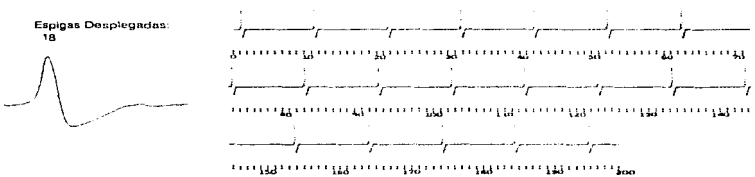


Fig. 6.3. Traza generada por EXTRACEL para ser analizada por DISCOVERY.

Tamaño de la traza : 200.0 msegs — Número de espigas que contiene : 1
Espiga No. 18 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 20
1.00 11.24 21.48 31.72 41.96 52.20 62.44 72.68 82.92 93.16 103.40 113.64 123.88 134.12 144.36
154.60 164.84 175.08 185.32 195.56

Recuadro 6.4. Información de la traza (Ver Fig. 6.3.), generada por EXTRACEL.

señal generada, en formato ASCII, Es decir no tiene una cabecera (Header) que reconozca DISCOVERY.

* Ya definidos los parámetros de la importación, ésta se lleva a cabo con la opción *Options - Import_ASCII*.

Y la información estará lista para ser usada por DISCOVERY.

Al igual que en la importación es necesario definir el análisis a realizar, así como la ventana en la que será desplegado el resultado, antes de poder ejecutar dicho análisis.

La opción *Analysis - FFT_Analysis* o en su defecto *Transfer - Analysis - FFT_Analysis* nos permite accesar el área de trabajo de la FFT.

Primeramente se generaron tres señales, la primera de ellas con la espiga número 18 (Ver Fig. 6.3. y Recuadro 6.4.)

La segunda de ellas con la espiga número 38 (Ver Fig. 6.4. y Recuadro 6.5.).

Y finalmente la tercera señal es una combinación de ambas espigas (Ver Fig. 6.5. y Recuadro 6.6.).

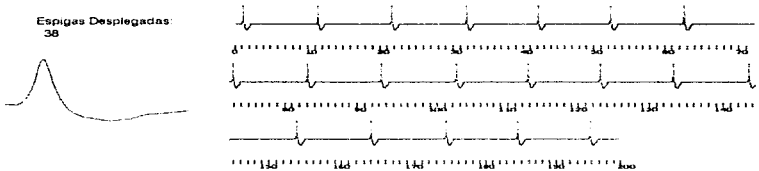


Fig. 6.4. Traza generada por EXTRACEL para ser analizada por DISCOVERY.

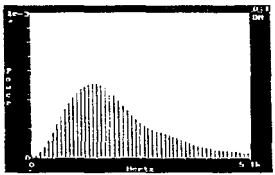
Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 1
Espiga No. 38 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 20
1.00 11.24 21.48 31.72 41.96 52.20 62.44 72.68 82.92 93.16 103.40 113.64 123.88 134.12 144.36
154.60 164.84 175.08 185.32 195.56

Recuadro 6.5. Información de la traza (Ver Fig. 6.4.), generada por EXTRACEL.

Fig. 6.5. Traza generada por EXTRACEL para ser analizada por DISCOVERY.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 2
Espiga No. 18 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 10
1.00 21.48 41.96 62.44 82.92 103.40 123.88 144.36 164.84 185.32
Espiga No. 38 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 10
10.24 30.72 51.20 71.68 92.16 112.64 133.12 153.60 174.08 194.56

Recuadro 6.6. Información de la traza (Ver Fig. 6.5.), generada por EXTRACEL.



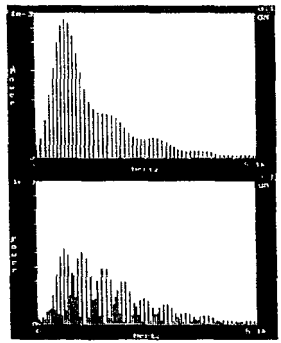


Fig. 6.6 Espectros de potencia obtenidos con ayuda de Discovery. En el orden de las manecillas del reloj, los espectros corresponden a: señal generada con la espiga 18, señal generada con la espiga 38 y señal generada con las espigas 18 y 38.

Al aplicar la FFT a cada una de las señales, los resultados obtenidos se presentan en la Fig.6.6.

En la cual podemos observar que los espectros obtenidos marcan potencias dominantes en diferentes lugares para las señales compuestas por las espigas 18 y 38 de manera independiente.

Y al analizar la señal que contiene ambas espigas, se nota la presencia de dos frecuencias dominantes, sin embargo por el gran número de frecuencias que compone cada señal y por la cercanía de los espectros correspondientes a cada una de las espigas, podemos darnos cuenta de que es difícil determinar cuantas espigas componen el tren por la simple inspección del espectro de potencias.

Por otro lado a mayor actividad, es decir a mayor número de espigas en la señal, la energía de la señal se incrementa lo que se ve reflejado en el espectro. Sin embargo se conservan las frecuencias dominantes que componen la señal aunque la potencia en cada una de ellas varíe.

Para ilustrar lo anterior se generaron dos señales más, las cuales contienen la misma espiga, pero los impulsos se presentan a diferentes intervalos en cada una de ellas. (Ver Fig. 6.8. y Recuadros 6.7. y 6.8.).

Los espectros correspondientes a estas señales pueden observarse en la Fig. 6.7.

Sin embargo debemos mencionar que el espectro de la señal se mantiene constante si la señal presenta ruido.

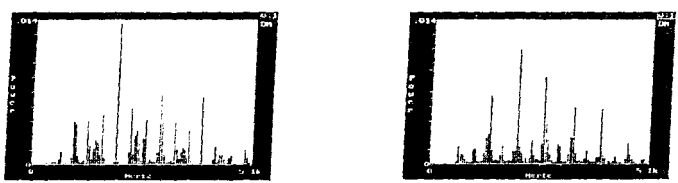


Fig. 6.7. Espectros de potencia obtenidos con ayuda de Discovery. El espectro de la izquierda corresponde al primer tren mostrado en la Fig. 6.8., y el de la derecha al segundo tren de la figura mencionada. Nótese la presencia de casi las mismas frecuencias dominantes, y la diferencia en la energía de cada una de ellas.

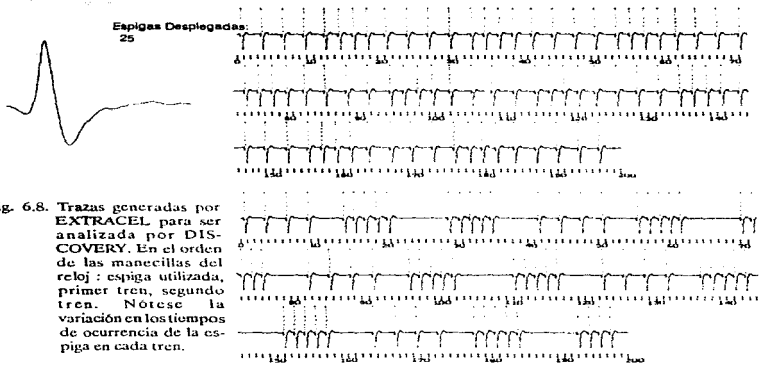


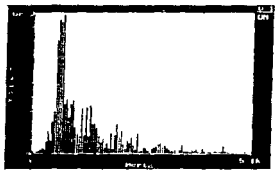
Fig. 6.8. Trazas generadas por EXTRACEL para ser analizada por DISCOVERY. En el orden de las manecillas del reloj : espiga utilizada, primer tren, segundo tren. Nótese la variación en los tiempos de ocurrencia de la espiga en cada tren.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 1
Espiga No. 25 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 77
0.50 3.50 6.50 9.50 12.50 14.50 16.50 18.50 20.98 23.98 26.98 29.98 32.98 34.98 36.98 38.98 41.46
44.46 47.46 50.46 53.46 55.46 57.46 59.46 61.94 64.94 67.94 70.94 73.94 75.94 77.94 79.94 82.42
85.42 88.42 91.42 94.42 96.42 98.42 100.42 102.90 105.90 108.90 111.90 114.90 116.90 118.90
120.90 123.38 126.38 129.38 132.38 135.38 137.38 139.38 141.38 143.86 146.86 149.86 152.86
155.86 157.86 159.86 161.86 164.34 167.34 170.34 173.34 176.34 178.34 180.34 182.34 184.82
187.82 190.82 193.82 196.82

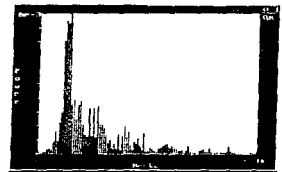
Recuadro 6.7. Información de la traza (Ver Fig. 6.8. primer tren), generada por EXTRACEL.

Tamaño de la traza : 200.0 msegs Número de espigas que contiene : 1
Espiga No. 25 (tiempo de las ocurrencias en msegs)
Número de repeticiones de la espiga : 69
0.50 3.50 6.50 9.50 14.50 16.00 17.50 19.00 20.50 29.00 30.50 32.00 33.50 35.00 41.46 44.46 47.46
50.46 55.46 56.96 58.46 59.96 61.46 69.96 71.46 72.96 74.46 75.96 82.42 85.42 88.42 91.42 96.42
97.92 99.42 100.92 102.42 110.92 112.42 113.92 115.42 116.92 123.38 126.38 129.38 132.38 137.38
138.88 140.38 141.88 143.38 151.88 153.38 154.88 156.38 157.88 164.34 167.34 170.34 173.34
178.34 179.84 181.34 182.84 184.34 192.84 194.34 195.84 197.34

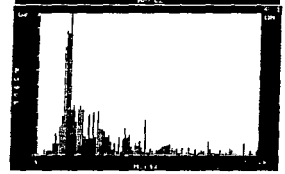
Recuadro 6.8. Información de la traza (Ver Fig. 6.8. segundo tren), generada por EXTRACEL.



0 %



10 %



55 %

Fig. 6.9. Espectros de potencia obtenidos con ayuda de Discovery. En el orden de las manecillas del reloj, los espectros corresponden a: ejemplo 18, ejemplo 32 y ejemplo 34. Podemos observar prácticamente el mismo espectro a pesar de que la señal presenta diferentes niveles de ruido en cada caso.

Para mostrar la invariabilidad ante el ruido se utilizaron los ejemplos 18, 32 y 34 del Capítulo 5. Y los espectros resultantes se muestran en la Fig. 6.9.

Al observar la Fig. 6.9. notamos que el espectro no varía notoriamente al pasar de 0% de varianza, a 10% y luego a 55%. Sin embargo, no es posible determinar el número de espigas que componen la señal a partir de la inspección del mismo. En esta señal hay tres espigas : 15, 16 y 17.

CAPÍTULO 7

DISCUSIÓN Y CONCLUSIONES

El paquete desarrollado (EXTRACEL) ha sido parcialmente evaluado en el Laboratorio de Cibernética y el resultado es bastante satisfactorio en cuanto a la generación de señales extracelulares. Sin embargo, se necesita la retroalimentación proveniente de usuarios externos para conocer mejor los pros y los contras del simulador en diferentes aplicaciones. Para obtener dicha retroalimentación se presentará el trabajo en diferentes foros y EXTRACEL se colocará en el dominio publico a través de Internet.

El paquete EXTRACEL permite simular señales extracelulares con patrones de disparo especificados, ya sean del tipo marcapaso o del tipo ráfaga, y también con patrones de disparo aleatorios. Aunque la base de señales pertenece a la corteza auditiva del gato [Abeles 75], es posible, por medios experimentales, obtener bases de señales de otras regiones del cerebro. Hay que agregar que obtener una base de señales cerebrales no es una tarea trivial, sino complicada y costosa. La base de señales utilizada representa cuando menos diez años de experimentos en gatos [Abeles 75, Abeles 77].

El objetivo principal de desarrollar este simulador fue separar espigas en señales compuestas por medio de paquetes especializados como DISCOVERY [Serna 94] o con paquetes experimentales como CLASIF [Quiza 91, Lara 96]. La separación de espigas o potenciales de acción en una señal compuesta es un problema añejo que se ha ido volviendo más y más complicado conforme la tecnología ha avanzado y se han podido registrar más neuronas

simultáneamente [Glaser 71, Abeles 77, Gerstein 83, Sanderson 85]. Se puede decir que el problema no esta resuelto del todo y, por eso, hemos probado, con cierto éxito, nuevas tecnologías como las redes neuronales artificiales [Quiza 91, Lara 96] y paquetes comerciales como DISCOVERY [Serna 94]. En estos dos últimos proyectos ha sido de mucha utilidad el paquete EXTRACEL. En este trabajo se presenta un catálogo de señales generadas aleatoriamente, debe entenderse que tales ejemplos son ilustrativos de las capacidades de EXTRACEL, pero no siempre pueden repetirse exactamente debido precisamente a que las señales son generadas aleatoriamente. Los ejemplos determinísticos si pueden repetirse exactamente como se muestran.

Al desarrollar y probar el programa nos dimos cuenta de que, además, el paquete EXTRACEL puede tener otras aplicaciones interesantes, como podrían ser las siguientes:

* Simulación de señales ruidosas para ser estudiadas con métodos espectrales como FFT y otros.

* Visualización de señales para paquetes de redes neuronales que únicamente generan los tiempos de disparo de las neuronas. En nuestro caso hemos utilizado esta aplicación con el paquete de redes neuronales biológicas NEURORED [Alcántara 92].

* Simulación de patrones de disparo de acuerdo con observaciones experimentales en regiones específicas del cerebro. En este caso, es muy importante que el usuario tenga experiencia electrofisiológica para que las señales diseñadas tengan un significado biológico y no sean únicamente señales generadas al azar. Es usual que el fisiólogo conozca muy bien el tipo de actividad extracelular de la zona cerebral que estudia.

De esta manera se cuenta ahora con una herramienta que es útil para familiarizarnos con trenes de espigas que presenten una diversidad considerable en la forma en que están distribuidas las espigas contenidas en él. Lo que nos llevará a una mejor interpretación de los registros reales que se realicen y, así, se aprovecharán al máximo los registros que se llevan a cabo durante los experimentos, los cuales involucran costos y niveles de complejidad muy altos.

CAPÍTULO 8

REFERENCIAS

[Abeles 75] ABELES, M., *A journey into the brain*, in INBAR, G.F.(ED.), Signal analysis and pattern recognition in biomedical engineering, Wiley, pp. 41-59. 1975.

[Abeles 77] ABELES, M. AND GOLDSTEIN, M.H., *Multispike train analysis*, Proc. IEEE 65: 762-773, 1977.

[Alcántara 92] ALCÁNTARA G., M.A., *NEURORED: Simulador y analizador de redes neuronales artificiales tipo biológico*, Tesis Licenciatura en Ing. en Computación, Fac. Ingeniería, UNAM, 1992.

[Bower 88] BOWER, J.M., *GENESIS*, Division of Biology, California Institute of Technology, Pasadena CA, 1988.

[Calvin 75] CALVIN, W.H., *Generation of spike trains in CNS neurons*, Brain Res., 84: 1-22, 1975.

[Discovery 94] *DISCOVERY 4.0 Users guide Section 3.* Data Wave Technologies Corporation,USA 1988-1994.

[Gerstein 83] GERSTEIN, G.L., BLOOM, M.J., ESPINOSA E., I., EVANCZUK, S., AND TURNER, M.R., *Design of a laboratory for multineuron studies*, IEEE Trans. Sys., Man, and Cybern. 13: 668-676, 1983.

[Glaser 71] GLASER, E.M., *Separation of neuronal activity by waveform analysis*, 77-136, Advances in Biomedical Engineering, Kenedi R.M. (Edit), Vol. 1, Ac. Press, 1971.

[Hines 89] HINES, M.L., *NEURON 1989*, Neuroengineering and Neuroscience Center, Yale Univ., New Haven, CT.

[Hodgkin 52-1] HODGKIN, A.L., HUXLEY, A.F., AND KATZ, B., *Measurement of current-voltage relations in the membrane of the giant axon of Loligo*, J. Physiol. 116: 424-448, 1952.

[Hodgkin 52-2] HODGKIN, A.L., HUXLEY, A.F., *Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo*, J. Physiol. 116: 449-472, 1952.

[Hodgkin 52-3] HODGKIN, A.L., HUXLEY, A.F., *The components of membrane conductance in the giant axon of Loligo*, J. Physiol. 116: 473-496, 1952.

[Hodgkin 52-4] HODGKIN, A.L., HUXLEY, A.F., *The dual effect of membrane potential on sodium conductance in the giant axon of Loligo*, J. Physiol. 116: 497-506, 1952.

[Kreiter 89] KREITER,A.K., AERTSEN, A. M.H.J. AND GERSTEIN,G. L. *A low-cost single-board solution for real-time, unsupervised waveform classification of multineuron recordings.* Journal of Neuroscience Methods, 30 (1989) 59-69

[Lanczos 64] LANCZOS, C. *SIAM Journal on Numerical Analysis*, ser. B, vol 1, pp. 8696, 1964.

[Lara 96] LARA V., R. Y ESPINOSA E., I. *Evaluación de un clasificador neuronal por medio de trenes multiespigas sintéticos*. Supl. Bol. Soc. Mex. Fis, XXXIX Congr. Nac. Física, p. 48, 1996.

[Maxwell 68] MAXWELL, J.C., *On Governors*, Proc. Roy. Soc. London 16: 270-283, 1868.

[McCulloch 43] McCULLOCH, W.S. AND PITTS, W., *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics 5: 115-133,1943.

[Mead 89] MEAD, C., *Analog VLSI and Neural Systems*, Addison-Wesley, 1989.

[O'Connell 73] O'CONNELL, R.J., KOCSIS, W.A. AND SCHOENFELD, R.L., *Minicomputer Identification and Timing of Nerve Impulses Mixed in a Single Recording Channel*, Proceedings of the IEEE, vol. 61, no. 11, November 1973.

[Press 92] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T. AND FLANNERY, B. P., *NUMERICAL RECIPES IN FORTRAN The Art of Scientific Computing*, Second Edition, 966 pp, Cambridge University Press, USA 1992.

[Quiza 91] QUIZA T., J., *Clasificación de potenciales de acción con una red neuronal que emplea retropropagación*, Tesis Maestría en Ing. Biomédica, UAM-Iztapalapa, 1991.

[Rashevsky 38] RASHEVSKY, N., *Mathematical Biophysics: Physico-Mathematical Foundations of Biology*, Univ Chicago Press, 1938.

[Sanderson 85] SANDERSON, A.C. AND PETERKA, R.J., *Neural modeling and model identification*, CRC Crit. Revs. Biom. Eng. 12: 237-309, 1985.

[Serna 94] SERNA H., R., SANTAMARÍA P., F., DOMÍNGUEZ S., I. Y ESPINOSA E., I., *Integración y validación de un sistema de adquisición y procesamiento de señales neurofisiológicas*. Memorias Congreso de Instrumentación, SOMI IX, Sociedad Mexicana de Instrumentación, A.C., pp. 98-101, 1994.

[Wodleriger 78] WODLERIGER, H., KUUOV, H. AND ATWOOD, H. L., *An electronic feature extractor for action potentials* , IBME and Department of Zoology, University of Toronto (1978).

APÉNDICE A

BASE DE SEÑALES

ESPIGAS DE LA CORTEZA AUDITIVA DEL GATO

Este trabajo se realizó tomando como materia prima para la generación de los trenes, una base de datos compuesta por 72 archivos. Cada uno contiene la información que describe la espiga correspondiente.

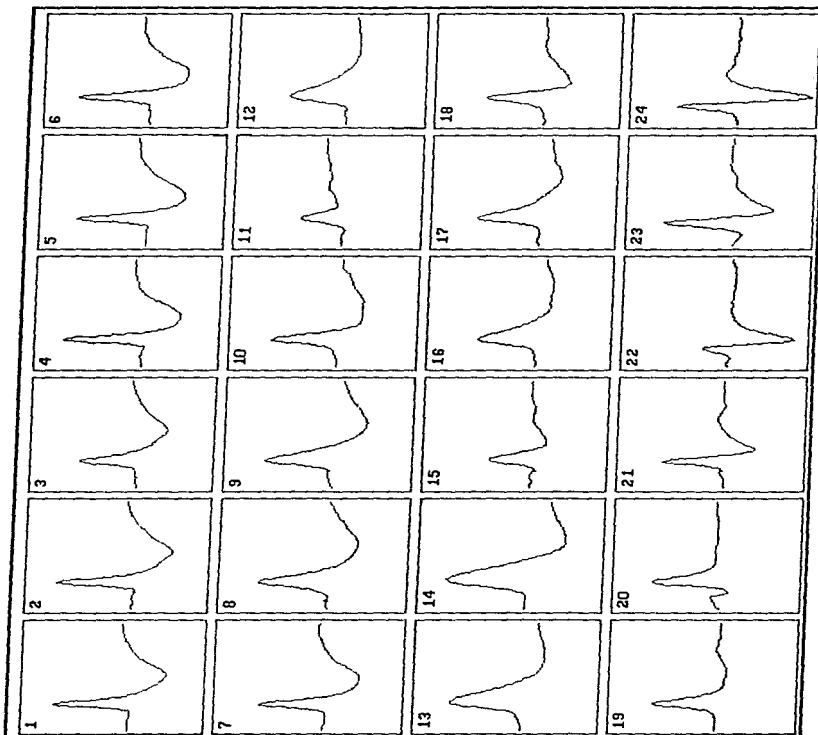
Los datos contenidos en estos archivos son de tipo entero, y cada uno de ellos contiene 128 datos.

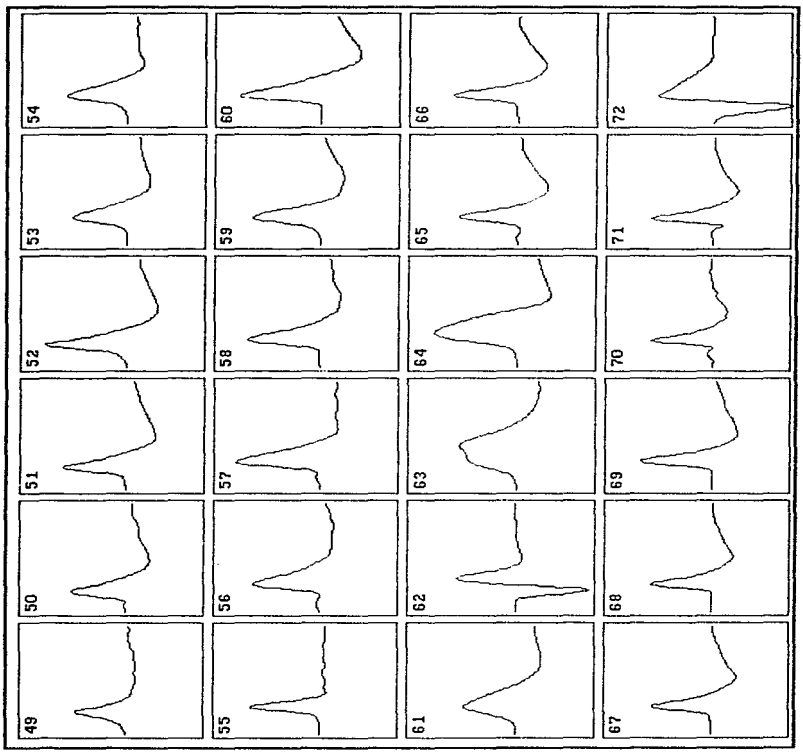
A continuación se muestran las espigas que constituyen la base de datos.

Éstas han sido desplegadas en orden ascendente y cada una de ellas está identificada por el número del archivo que contiene la información correspondiente.

El identificador de los archivos de datos, que contienen las espigas esta formado por la raíz: "spike", el número de la espiga correspondiente y la extensión : ".dat".

De esta manera, si deseamos conocer los datos que describen la espiga número 8, basta con accesar el archivo "spike8.dat", para obtener dicha información.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24


49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

APÉNDICE B

MANUAL DEL USUARIO

El programa cuenta con varios módulos de procesamiento, dentro del menú principal, que permiten manipular los trenes generados. Desde su generación, ampliación (inserción de impulsos de espigas al tren ya generado), adición de ruido, almacenamiento y recuperación de trenes grabados en unidades de almacenamiento secundario, visualización de las formas de espigas presentes en el tren de impulsos.

El desarrollo de este programa está orientado a que el manejo del mismo sea fácil y flexible.

La finalidad del sistema es la de generar los trenes de impulsos que serán procesados por los paquetes de análisis ya existentes.

Cuando ejecutamos el sistema es necesario elegir el idioma en el que los mensajes de las peticiones, esto es mensajes de opciones de cada menú y las peticiones de entrada de datos; y

```
**************
Selecciona el idioma a utilizar
**************
Please, select the language you want to use
**************
 1 ) Español
 2 ) English
 1
```

Recuadro 1. **Selección de idioma. Cursivas indican la selección hecha por el ususario.**

de información del sistema, como son mensajes de error y mensajes de espera, serán desplegados (Ver. Recuadro 1.).

El programa maneja ahora, sólo dos opciones en la selección y éstas son: Español e Inglés.

Es posible adicionar otro idioma, solo se requiere de la modificación de la función asociada,[1] y recompilando el programa.

La implementación de esta facilidad del sistema, de manejar al menos dos idiomas, está basada en la experiencia que se ha tenido en el Laboratorio, la que nos ha enseñado que en ciertas ocasiones, como es la presentación de artículos en el extranjero, el campo de utilización del sistema se amplia hacia gente fuera de las fronteras de nuestro país, quienes no necesariamente hablan nuestro idioma; y una manera de apoyar lo anterior, es tener la posibilidad de ofrecerle al usuario un sistema que le sea fácil y accesible de utilizar. El manejo de un programa en un idioma conocido, es de gran ayuda en la operación de cualquier sistema de cómputo.

1) GENERACIÓN DE TRENES DE ESPIGAS (Generar Traza)

```
1) Generar Traza
2) Cargar Traza
3) Agregar una espiga a la traza
4) Agregar ruido a la señal
5) Graficar traza
6) Salvar
7) Traslapar las espigas
8) Información de la traza
9) Salir
                /
```

Recuadro 2. Menú

La primera opción que ofrece el sistema es la de generar un tren de impulsos (Ver Recuadro 2).

Este módulo permite generar el tren bajo dos políticas diferentes, la primera de ellas se refiere al uso de tiempos predeterminados por el usuario; y la segunda utiliza los tiempos de

[1] Ver sección 4.1 Metodología.

ocurrencia generados aleatoriamente por el sistema, en éste último el número de ocurrencias de cada espiga también es determinado aleatoriamente.

Recuadro 3. Forma de generar los tiempos de ocurrencia de las espigas a insertar en el tren.

```
Generar Traza
1 ) Con tiempos aleatorios
0 ) Con tiempos predeterminados
        0
```

- El primer paso después de elegir este módulo es elegir alguna de las políticas para generar el tren de impulsos (Ver Recuadro 3).

Esto es, decidir si la generación de la traza será en base a tiempos aleatorios o tiempos predeterminados.

- El siguiente paso, si se eligió *Generación de Tiempos Predeterminados,* consiste en

Recuadro 4. Selección de espigas. En éste caso se seleccionó la espiga 12 y al dar el 0 (cero) se terminó la entrada de datos.

```
Selección de espigas a insertar en el tren
Para finalizar la entrada de datos
            espiga = 0
cuales son las espigas a seleccionar.
        12  0
```

proporcionar los identificadores de las espigas que se utilizarán en la construcción del tren (Ver Recuadro 4). Es decir indicar qué espigas son las que incluiremos en nuestra traza.

- Como se eligió *Generación de Tiempos Predeterminados.* Esto es trabajar con tiempos de ocurrencia proporcionados por el usuario, se deberán de entrar los datos correspondientes a los tiempos de cada una de las espigas.

El primer valor que se solicita es el del período de la señal (Ver Recuadro 5), es decir, si suponemos una neurona que está bajo una excitación sostenida, presentará espigas periódicas.

Recuadro 5. Selección del período de la señal. En éste caso es de 5 msegs.

```
Cual es el período de repetición de los
impulsos de la espiga : 12 en msegs
período = 0  = sin período
          5
```

Entonces, si deseamos que los disparos de la espiga se presenten cada 5 milisegundos. El período de disparo de la espiga es 5 (Ver Recuadro 5).

El uso del período al entrar los datos relativos a los tiempos de ocurrencia de las neuronas es, con miras a facilitar la entrada de datos al sistema. Ya que si por ejemplo se quieren entrar los tiempos de una espiga que se presenta cada 2 milisegundos, en un tren de 200 milisegundos, se tendrían que entrar al menos 100 datos, y al usar el período basta con indicar el valor de período igual a 2, y proporcionar tiempo en el que se desea que se presente la primera espiga, y el programa calculará los tiempos restantes.

Si el patrón de disparo que estamos considerando es aperiódico, entonces con proporcionar un período igual a cero, es suficiente.

En ocasiones la actividad de ciertas neuronas está conformada por ráfagas de espigas, por

```
Cuales son los tiempos del primer ciclo
para la espiga No. 12 en msegs
Para finalizar la entrada de datos
de cada espiga : t = 0
        4  0
```

Recuadro 6. Tiempos del primer ciclo. En éste caso se tiene sólo un disparo en cada periodo, a partir del milisegundo 4.

esta razón se pueden proporcionar varios tiempos de disparo para el primer período de la neurona (Ver Recuadro 6).

Una vez que se han dado los tiempos de cada espiga, se pasa a la siguiente etapa del proceso proporcionando un cero en la línea de datos, el cual le indica al sistema que ya no se proporcionará más información acerca de la espiga actual y se procederá, así, a generar el tren de impulsos. Y posteriormente si se seleccionaron más espigas se repetirá el proceso anterior para cada una de ellas.

Al generar el tren cada pulso de la ráfaga de la espiga será repetido cada *n* milisegundos, donde *n* es el período de repetición que se proporcionó inicialmente (Ver Recuadro 6).

El tiempo inicial para cada espiga será el proporcionado para el primer ciclo y el tiempo para el segundo periodo será igual al tiempo del primer disparo más el periodo, y así sucesivamente hasta insertar las espigas en todo el tren.

Al entrar los tiempos para cada neurona, se debe de tener en cuenta que:

* El período que se determine, en caso de que se requiera, debe de ser adecuado al número de veces que se repetirá la espiga en cada ciclo. La duración de cada espiga es de 1.3 milisegundos.

Es decir si se selecciona un período de 5 milisegundos y deseamos tener 5 espigas en cada período esto no será posible. Con lo que o se determina un período más grande o se disminuye el número de espigas que se quiere en cada período.

El programa automáticamente calculará un período adecuado al número de espigas que se desee, en caso de que el periodo seleccionado inicialmente sea muy pequeño para albergar las espigas que se desea insertar en el mismo.

```
Los tiempos que serán utilizados
para cada periodo son :  ( en msegs )
                4
el periodo de repetición de los
impulsos de la espiga : 12 es de 5 msegs
presiona 'C' para continuar
                'R' para redefinir los tiempos
```

Recuadro 7. Despligue de la información para generar el tren.

Los valores que se manejarán en el proceso de generación de tiempos serán desplegados en pantalla por el programa, y se dará opción a aceptar los valores o a modificarlos (Ver Recuadro 7).

* Los tiempos en que se presentará una misma espiga, debe de estar a una distancia mínima de 1.3 milisegundos. Debido a que no es posible tener espigas a una distancia menor.

Esto es, el período será ajustado para evitar que se presente una superposición de las ráfagas o de los impulsos individuales de la espiga. Ya que de otra forma no se tendría una simulación válida equiparable al comportamiento de las neuronas.

Es decir no es posible que una misma neurona dispare dos veces dentro del mismo intervalo de tiempo en el que ocurriría un sólo impulso de dicha neurona. Ya que las condiciones que deben reunirse a nivel fisiológico, requieren de un tiempo mínimo para reunir las condiciones de disparo que puedan proporcionar la energía necesaria para generar un segundo impulso.

* Si los datos que fueron proporcionados no son los correctos es posible re-entrar los datos para adecuarlos a las necesidades del usuario.

Generar Traza
1) Con tiempos aleatorios
0) Con tiempos predeterminados
1
Porcentaje de concentración
de generación de tiempos
40

Recuadro 8. Generación de tren con tiempos aleatorios.

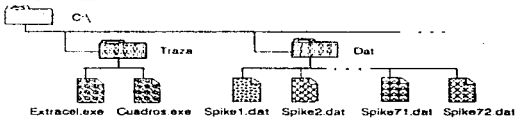


Fig. B.1. Ubicación de los directorios que contienen los programas y la base de señales.

- Si se eligió *Generación de Tiempos Aleatorios* entonces el programa se encargará de generar los tiempos correspondientes, una vez que se le ha proporcionado el porcentaje de concentración de impulsos (valores recomendados entre 30% y 60 %) a considerar en el tren (Ver Recuadro 8).

- El siguiente paso es seleccionar las espigas a insertar en la traza (Ver Recuadro 4), ya con los identificadores de las espigas, el programa genera los tiempos en los cuales ocurrirán cada una de ellas a lo largo de la traza.

- Ya con la información acerca de los tiempos, el programa hace el vaciado de éstos en la traza.

- Después de lo cual puede ser desplegado en pantalla o bien ser salvado en un archivo, para poder así ser utilizado como entrada para los programas de análisis que así lo requieran.

No debe olvidarse que :

* La ruta de las espigas a seleccionar está predeterminada, por lo que si los archivos de descripción de cada espiga no se encuentran ahí, entonces no se podrá generar la traza. Las rutas son :

...\traza\EXTRACEL.exe

...\dat\spike#.dat donde: # es el número de la espiga a la que se desea hacer referencia.

En general la base de datos de las espigas estará en un subdirectorio llamado DAT, y que está ubicado al mismo nivel del directorio donde se encuentra el ejecutable de nuestra aplicación (Ver Fig B.1).

El programa debe de ser ejecutado desde el directorio en el cual esta archivado, de otra manera no encontrara los archivos que se le estan solicitando, y desplegara un mensaje de error (Ver Recuadro 9).

```
Selección de espigas a insertar en el tren
Para finalizar la entrada de datos
espiga = 0
cuales son las espigas a seleccionar.
                23 0

          **********

No fue posible abrir el archivo:
        ..\dat\spike23.dat
          **********
```

Recuadro 9. Error al ejecutar el programa diferente al de "Traza'.

2) LECTURA DE UN TREN ALMACENADO EN DISCO (Cargar Traza)

El segundo módulo permite que el sistema lea los datos de un tren generado anteriormente para que pueda ser modificado, o quizás únicamente para ser desplegado en pantalla.

Es importante mencionar que al leer un tren que haya sido almacenado en un archivo de datos, la información que ya estaba almacenada en memoria será reemplazada con la del archivo que sea leído.

Por lo que, se debe de tener cuidado en almacenar la traza que está en memoria, en un archivo, antes de realizar la lectura del siguiente archivo que nos interesa.

- El sistema solicita el nombre del archivo a cargar en memoria, se debe de proporcionar el nombre tal y como se dió para salvarlo, esto es el nombre y la extensión que identifican el archivo deseado (Ver Recuadro 10).

Si el archivo no se encuentra en el directorio actual, es decir no se encuentra en el directorio desde el cual se ejecutó el programa, entonces será necesario proporcionar la ruta completa, que será utilizada para localizar los archivos de datos que definen el tren de impulsos.

```
1) Generar Traza
2) Cargar Traza
3) Agregar una espiga a la traza
4) Agregar ruido a la señal
5) Graficar traza
6) Salvar
7) Traslapar las espigas
8) Información de la traza
9) Salir
                2

Cual es el nombre del archivo :
        ejems\e_10.ext
```

Recuadro 10. Recuperación de archivos almacenados en disco, a memoria.

Al llevar a cabo la búsqueda del archivo a cargar en memoria, desde el disco duro o el disquete colocado en la computadora, el sistema determina qué archivos, relacionados con el que nos interesa cargar, están presentes.

Cuando se graba el tren en un archivo, para ser almacenado en un dispositivo de memoria secundaria, se generan dos archivos, uno con extensión .EXT y el otro con extensión .DES.[2]

Por lo que al llevar a cabo la lectura del archivo, se tienen básicamente tres casos :

* Que estén presentes los dos archivos, el que contiene los datos de la traza en sí (.EXT) y el archivo descriptor (.DES), en cuyo caso se procede a cargar la información en los arreglos correspondientes para ser almacenados en memoria, con lo que el tren queda totalmente descrito, es decir, se sabe qué espigas componen el tren de espigas y se conocen los tiempos de disparo de cada una de ellas, así como los valores de cada uno de los puntos que forman el tren de impulsos.

** Que sólo se encuentre el archivo .EXT, en este caso, se podrá tener acceso sólo a los valores de cada uno de los puntos que forman el tren de impulsos. Pero no se podrá determinar qué espigas y en qué tiempos se dieron los disparos de éstas.

*** Cuando el sistema detecta la existencia del archivo .DES únicamente, entonces es posible reconstruir totalmente el tren ya que se cuenta con los identificadores de las espigas

[2] Ver sección ALMACENAMIENTO EN DISCO, para más detalles.

involucradas, así como con la información relativa a los tiempos en que se realizan los disparos de las neuronas que forman el tren.

- Si se desea reconstruir un archivo a partir del archivo descriptor, sólo se proporciona el nombre del archivo, sin extensión. Y el programa generará el tren a partir del archivo descriptor.

Por ejemplo si se tiene el archivo *prueba.DES*, para generar la traza asociada a este archivo, en la petición del nombre del archivo se indica : *prueba*.

Se debe de tener en cuenta:

* La extensión del archivo con los datos de la traza, en formato EXTRACEL, es EXT, por convención, pero puede ser cualquier otra.

* La extensión del archivo descriptor debe ser forzosamente DES. Si se le asigna otra extensión el programa no tendrá forma de saber que es el archivo descriptor. Y el nombre es el mismo del archivo de datos de la traza.

3) ADICIÓN DE UNA O MÁS ESPIGAS EN EL TREN

En la Generación de la traza es posible elegir, ya sea tiempos aleatorios o tiempos predeterminados únicamente, es decir no se puede seleccionar un proceso diferente para cada espiga, al menos no con el primer módulo.

Y cada vez que se desea generar un nuevo tren el anterior es eliminado, y la información del nuevo tren sustituye la del anterior.

Por otro lado existe la posibilidad de que al investigador después de generar un tren le interese agregarle una o varias espigas más a la traza previamente generada.

Es en estas situaciones en los que el presente módulo puede ser utilizado.

En el primer caso, si deseamos combinar neuronas cuyos tiempos de ocurrencia sean aleatorios con neuronas con tiempos de disparo predeterminados, una opción es:

- Con ayuda del primer módulo generar la traza que contendrá las espigas con tiempos aleatorios (o predeterminados según se prefiera)[3].

[3] Ver sección Generación de Trenes de Espigas, para más detalles.

- Posteriormente con ayuda de este módulo (inserción de espigas), agregar las espigas que se presentarán de acuerdo a tiempos predeterminados por el usuario (o aleatorios, según sea el caso).

- El resultado será la traza con espigas que se presenten cada cierto tiempo (especificado por el usuario), y aquellas que se presenten de una manera aleatoria.

El proceso anterior describe la manera en que se puede llevar a cabo la inserción de espigas en el tren.

Pero se debe de tener en cuenta :

* El proceso de selección de generación de tiempos, de selección de espigas y de asignación de tiempos (si es requerido), se lleva a cabo exactamente de la misma forma en que se lleva a cabo en el módulo de Generación de traza.

* Si el tren que se tiene actualmente en memoria fue leído de un archivo, y durante la lectura de la información que lo define sólo se encontró el archivo con extensión .EXT, entonces no se tendrá información acerca de las espigas que lo componen, y en tal caso no podrá determinarse si las espigas que se están insertando estaban o no, presentes en el tren inicial.

Sin embargo serán insertadas, por lo que en este caso existe la posibilidad de tener impulsos de una misma neurona siguiendo diferentes comportamientos y en el peor de los casos, podrían ocurrir traslapes de la misma espiga, lo que provocará una situación de inconsistencia del tren generado.

4) ADICIÓN DE RUIDO A LA SEÑAL

Cuando se realizan registros de trenes de impulsos durante los experimentos, las señales registradas presentan un nivel de ruido, que es ruido gaussiano.

Ahora bien las espigas que componen la base de datos con que se cuenta han sido separadas e identificadas a partir de registros que se hicieron en corteza cerebral de gato. Estas espigas han sido identificadas y separadas previamente por lo que están libres de ruido

Como podemos darnos cuenta las espigas que constituyen la base de datos de señales a partir de las cuales se generarán los trenes, no presentan ruido, entonces por lo tanto, los trenes

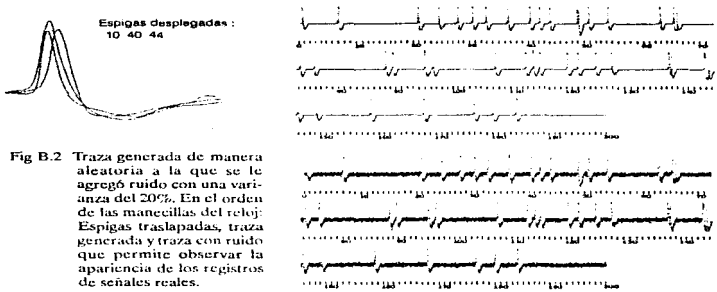


Fig B.2 Traza generada de manera aleatoria a la que se le agregó ruido con una varianza del 20%. En el orden de las manecillas del reloj: Espigas traslapadas, traza generada y traza con ruido que permite observar la apariencia de los registros de señales reales.

generados, mediante los módulos correspondientes (léase: Creación de trenes e Inserción de espigas), serán generados como idealmente se conciben los trenes de impulsos, es decir sin ruidos de ninguna especie y formados únicamente por las espigas de las neuronas aledañas a la zona del registro.

Sin embargo en la realidad no ocurre así. Ya que por la manera en que funciona el electrodo, éste capta las espigas, de las neuronas que están en la zona de registro. Y las espigas de las neuronas que están muy lejanas se captan pero como señales muy débiles las cuales forman lo que se considera como ruido nervioso y es el que les da ésta apariencia a las señales registradas (Ver Fig B.2).

Si deseamos que nuestra señal presente esta característica, presencia de ruido, entonces utilizaremos este módulo para agregarle al tren que tengamos en memoria el ruido deseado.

En este proceso será necesario indicarle al sistema la varianza que deseamos manejar al calcular el ruido.

No se debe olvidar :

* Que este módulo adiciona ruido al tren que generamos con anterioridad, el cual debe estar en memoria. Si no hay alguna traza en memoria, entonces no habrá tren al cual agregarle el ruido.

5) DESPLIEGUE DEL TREN DE IMPULSOS (Graficar la Traza)

El despliegue de la traza que se ha generado es importante, ya que es una de las formas más tangibles de darse cuenta acerca de que fue lo que ocurrió con el proceso, es decir si fue exitoso o no.

Es una de las formas más tangibles ya que, a simple vista el usuario del sistema se da cuenta si lo que está viendo, es lo que el esperaba generar. Si al usuario se le proporcionara un listado conteniendo los valores de los puntos de la traza, no le dirían gran cosa una serie de diez o veinte mil puntos.

Por otro lado, al saber que es lo que contiene la traza que generó, sabe en cierta medida, que es lo que debe esperar de los programas que usarán tales datos.

Por el ejemplo, Clasif permite observar el tren de espigas que leyó, así que si el investigador compara la gráfica que genera este módulo contra la gráfica que esté generando Clasif, podrá darse cuenta en un principio, si los datos están siendo leídos, por Clasif, de una manera adecuada.

Así que una vez que el tren de espigas ha sido generado, el investigador podrá ver como quedaron distribuidas las espigas a lo largo del tren que se generó.

Graficar la traza es el módulo encargado de llevar a cabo este proceso, y basta tan sólo con seleccionarlo para que se lleve a cabo el despliegue de la señal.

Se debe tener presente que :

* El archivo *egavga.bgi*, esté presente en el directorio donde se tiene el ejecutable de este programa.

6) ALMACENAMIENTO EN DISCO (Salvar)

A pesar de que la generación de los trenes es mucho más rápida de lo que es el realizar los experimentos, siempre es importante tener manera de guardar el trabajo que ya ha sido desarrollado para evitar generar un mismo tren cada vez que éste sea requerido.

La información del tren que está en memoria, puede ser almacenada en disco en tres formatos diferentes:

- EXTRACEL

- CLASIF
- DISCOVERY

El usuario debe de seleccionar uno de los tres formatos y posteriormente asignar el nombre del archivo en el cual se desea que se haga el vaciado de datos correspondientes a la traza y al archivo de control (Ver Recuadro 11).

Formato de Salida
1) EXTRACEL
2) CLASIF
3) DISCOVERY
1

Cual es el nombre del archivo de salida en formato EXTRACEL :
prueba.ext

Descripcion del archivo a grabar
Ejemplo de proceso de grabacion de una traza.

Recuadro 11. Almacenamiento de la traza en disco. La 'Descripción del archivo a grabar' es almacenada, al final del archivo con extensión .DES.

El archivo de control es generado en los tres casos. Y la traza puede ser reconstruida por EXTRACEL a partir de éste, aún si no se tiene el archivo de los datos que definen la traza en sí.

A este archivo, se le asigna el mismo nombre que se le dió al archivo de datos pero con extensión .DES, contiene la información que describe el tren, es decir, longitud en número de puntos que contiene el tren de impulsos, número de espigas que lo componen y sus identificadores. Si se le agregó o no ruido a la señal y el valor de la varianza del mismo.[4]

Se debe de tener en cuenta :

* Por convención los archivos en formato EXTRACEL se identificarán mediante el uso de la extensión .EXT, aunque puede usarse otra extensión si así se desea.

[4] Ver Almacenamiento en Disco, en la sección 4.1, para una descripción más detallada del formato del archivo descriptor.

* Para los archivos en formato CLASIF, la extensión que deberá ser utilizada es .DAT, debido a que de otra manera, el programa CLASIF no identificará tal archivo como un archivo de datos en su formato. Por lo que en este caso lo más recomendable es usar .DAT como la extensión que denote archivos en este formato. Ahora bien si se desea se puede usar otra extensión, siempre y cuando se tenga presente, en el momento de tratar de analizar dicha traza con ayuda de CLASIF, el cambiar el nombre del archivo, por otro en el que se use la extensión correspondiente.

* Para los archivos en formato DISCOVERY lo mas recomendable es hacer uso de la extensión .TXT que denota el uso de archivos en ascii, ya que para hacer uso de estos archivos en tal programa, es necesario importar la información como tipo ascii.

* En los tres casos es posible, dentro de lo que cabe, determinar la extensión que se desee, como parte del nombre que se le asigne al archivo en el que se almacenarán los datos correspondientes. Pero por ningún motivo debe de ser utilizada la extensión .DES como parte del nombre del archivo de datos que definen la traza. Ya que ésta es la extensión que se le asigna al archivo de control de la traza.

7) TRASLAPE DE LAS SEÑALES (Traslapar Espigas)

Este módulo permite al usuario observar las diferencias de forma que existen entre las diferentes espigas que fueron utilizadas como base al generar el tren de impulsos.

La función de este módulo es la de desplegar las espigas que fueron seleccionadas para formar parte de la traza, y graficarlas a partir de un mismo punto de referencia, Y es este despliegue el que permite comparar de una manera más fácil las diferencias en tiempos de disparo, amplitud de los picos de las espigas, la forma y duración de los valles, entre otras características.

Comparaciones que nos permiten determinar si las espigas son semejantes o no.

Se debe de tomar en cuenta :

* Las espigas podrán ser desplegadas siempre y cuando se cuente con la información respectiva, es decir que se tengan almacenados en memoria los identificadores de las espigas utilizadas.

Y dicha información estará disponible, siempre y cuando el tren de impulsos haya sido generado en la presente sesión o bien, si el archivo fue cargado desde un archivo de datos, que éste haya sido recuperado totalmente a partir del archivo de datos de la traza (extensión .EXT) y el archivo de control (extensión .DES) ó al menos del archivo de control.[5]

8) INFORMACIÓN DE LA TRAZA

La información de la traza es toda aquella información relativa al tamaño del tren en milisegundos; el número de espigas que contiene la traza; cuántas veces se repite cada espiga en la traza y en qué tiempos; si se agregó o no ruido a la señal, y en tal caso qué varianza se utilizó (Ver Recuadro 12).

```
        Información de la traza
Tamaño de la traza : 200.0 msegs
Señal con ruido, varianza: 20
Número de espigas que contiene : 1
Espiga No. 36      (tiempo de las
                   ocurrencias en msegs )
Número de repeticiones de la espiga : 15
12.53  14.07  17.02  18.51  20.04  21.71
23.77  25.25  27.61  29.00  30.71  32.27
33.95  35.48  36.76
```

Recuadro 12.
Información de la traza.

Esta información es de utilidad cuando se tiene una traza de la que no se sabe o el usuario no recuerda con qué espigas se generó.

Es útil también para que el investigador se de cuenta de una manera precisa en qué tiempos se están presentando cada una de la espigas.

Es importante hacer notar :

* Si la traza que está actualmente en memoria fue leída de algún archivo, es decir que se utilizó el segundo módulo para cargar la traza en memoria, entonces esta información estará

[5] Debe tenerse en cuenta que al recuperar un archivo que haya sido salvado con anterioridad, para reconstruir la información en su totalidad se debe de contar al menos con el archivo de extensión .DES. Ver Lectura de un tren almacenado en disco, en la sección 4.1.

disponible únicamente en el caso de que se haya tenido acceso al archivo de control en el momento de cargar dicha traza.

9) SALIDA DEL SISTEMA (Salir)

Como su nombre lo indica, esta opción nos permitirá salir del sistema.

Antes de dar por terminada la presente sesión de trabajo con el sistema, éste se encargará de liberar la memoria que haya sido utilizada durante el proceso.

APÉNDICE C

CÓDIGO DEL PROGRAMA

EXTRACEL está formado por cuatro subprogramas, y un archivo de encabezados :

Prot.h	Es el archivo de encabezados.
Traza.c	Contiene las funciones relacionadas con la generación de los trenes. Y es el programa principal.
Dist_poi.c	Contiene las rutinas del generador de números aleatorios con distribución de Poisson.
Grafica.c	Contiene las rutinas relacionadas con la graficación de los trenes generados, así como, el despliegue del traslape de las espigas contenidas en dichos trenes.
Textos.c	Contiene los textos a ser deplegados en los mensajes usados por EXTRACEL.

El código de cada uno de éstos programas es listado a continuación.

Junto con EXTRACEL se desarrolló el progra llamado CUADROS, el cual permite visualizar las espigas de la base de datos.

C.1 PROT.H

```
/***** prot.h ********/
/**** Prototipos ******/
/****** Traza . c ******/

void dat_esp( int opc, int * arch);
float gen_ran (int *ent, float flota);
void gen_ruido ( int *ent);
void gen_ts_aleat(int opc);
void gen_ts_noal(int opc);
void info(void);
void inicializa(int opc);
void ins_ruido(void);
void lee_datos(void);
void libera(void);
void limp_i (int *vector, int tam);
void limp_v (int **vector, int tam);
void menup(void);
void normaliza(int* vector);
void ordena(int * vect,int elem_vect);
int p_max(int* vector, int val1, int val2);
int p_min(int* vector, int val1, int val2);
void salvar(int*vector, int tam);
void sel_esps(int band);
void tren_espigas(int opc);

/*** Variables globales **/

int ** esp = NULL;
 int ** t_ocur_esp = NULL;
 int *esps_traza = NULL;
 int *t_ocur_traza = NULL;

 int a_desc = 1;    /* existe archivo descriptor */
 int band_grab = 0;
 int ent[4] = {0,0,0,0}; /* valores necesarios para
                            la inserción de ruido */
 int esp_max = 72;  /* número de espigas en la base
                       de datos  */
 int genera = 0;    /* ya se ha generado el vector de
                       ocurrencias */
 int idioma = 1;        /* bandera para el despliegue
                           de títulos  */

 int n_es = 72;     /* número máximo de espigas
                       a insertar en la traza */
 int n_pt = 128;    /* número de puntos de cada
                       archivo de espigas */
 int n_rep_esp = 0; /* número de repeticiones de
                       la espiga en el tren*/
 int no_seles = 0;  /* número espigas seleccionadas
                       */

 int sel_menup = 0; /* selección en el menú
                       principal */
  int tam_traza = 20001; /* tamaño del tren en
puntos
                            100 ptos = 1 mseg */
  int ts_aleat = 0;   /* tipo de genración de pulsos
*/

  float esc = 0.6;  /* escala de graficación  */
  float esc_cuad = 0;
  float por_al = 10; /*  regulador en la generación
de
                        datos aleatorios */

  char t_cad[255] = "";    /* varible puente en los
                              despliegues de texto */
  char descrip[255] = ""; /*  Descripción del archivo
                              al salvarlo */

 /********** Grafica **********/

 void fin_graf(void) ;
 void graf_esps_sel(int opc, int x, int y);
 void inic_graf (void);
 void pinta( int * dat_esps,float fac_esc, int x, int
y,int tam,int t_gr);

  /********** TEXTOS.C **********/

 void textos (int tex);
```

C.2 TRAZA.C

```
/* Generación de la traza  TRAZA.C  */

#include <alloc.h>
#include <bios.h>
#include <conio.h>
#include <ctype.h>
#include <dos.h>
#include <graphics.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "prot.h"
#include "textos.c"
#include "grafica.c"
#include "dist_poi.c"

int maximo = 0;

/****  Selecciona el valor de la bandera que ****/
/****  Que indica el idioma en el cual serán ****/
/****  desplegados los mensajes.            ****/
/****  idioma = 0 : Inglés ; idioma = 1 : Español
*/
   void cam_id (void){
   char l[2];
     for (;;){
      textos(0);
      scanf("%s", l);
      switch(l[0]){
       case '2':
        idioma = 0;
        return;
       case '1':
        idioma = 1;
        return;
      }
      l[0] = '\0';
     }
   }

 /****  Lee los datos de cada uno de los archivos de
espigas ***/
 /****       num_esp : el número de archivo a leer
***/
 /****  almacena en vector[n_pt + 2] el número de
archivo    ***/

   void dat_esp( int opc, int * arch){

   char arch_inom[30];
   FILE *f;

   strcpy(arch_inom , "");
                          sprintf(arch_inom ,
"..\\dat\\spike%i",esps_traza[opc]);
   strcat(arch_inom,".dat");
   arch[n_pt + 2] = esps_traza[opc];
   if ((f = fopen(arch_inom, "rb")) == NULL)
   {
     printf( "\n\t  ***********");
     strcpy(t_cad, arch_inom);
     textos(2);
     printf("\n\t  ********** \n");
     exit(1);
   }
   fread(arch,sizeof(int), n_pt,f );
   fclose(f);
   normaliza(arch);
   }

 /****    Control del flujo principal del programa
****/

   void general(void){

   int k;
   static int no_sel_aux = 0;
   int x = 5 ,y = 100 ;

     clrscr();
     cam_id();
       inicializa(1);  /*  inicializa vector de
esps_traza(n_es) */
     for (;;){
      if (sel_menup == 9) break;
      menup();

      switch (sel_menup){
     case 9:
        break;
     case 1:
       sel_esps(0);
       if (no_selcs){
       if(!genera){
                inicializa(2); /*  inicializa matriz de
t_ocur_esp(n_es)*/
```

```
                inicializa(3); /*   inicializa
t_ocur_traza(tam_traza)*/
        for (k=0;k<no_selcs;k++) {
          if (ts_aleat)
          gen_ts_aleat(k);
           else
          gen_ts_noal(k);
          tren_espigas(k);
         }
         genera = 1;
         a_desc = 1;
       }
       }
       break;
    case 2:
       libera();
       inicializa(1);
       inicializa(2);
       inicializa(3);
       strcpy(descrip,"");
       lee_datos();
       genera = 1;
       break;
    case 3:
       if (!genera){
      inicializa(2);
      inicializa(3);
      genera = 1;
       }
       no_sel_aux = no_selcs;
       sel_esps(1);
       for (k = no_sel_aux;k<no_selcs;k++) {
      if (ts_aleat)
        gen_ts_aleat(k);
      else
        gen_ts_noal(k);
        tren_espigas(k);
       }
       a_desc = 1;
       break;
    case 4:
       maximo = 0;
       for(k=0;k<tam_traza;k++)
      maximo = p_min(t_ocur_traza,k,maximo);
      /* calculado con la función de mínimos por */
      /* que el máximo de la espiga está en el   */
      /*valor menor de y */
       maximo = t_ocur_traza [maximo];
       getch();
       ins_ruido();
       break;
    case 5:
       inic_graf();
       setbkcolor(15);
       setcolor(GREEN);
       pinta(t_ocur_traza, 1,x,y,tam_traza,0);
       setcolor(LIGHTBLUE);
       textos(31);
       getch();
       fin_graf();
       break;
    case 6:
       salvar(t_ocur_traza,tam_traza);
       break;
    case 7:
       inic_graf();
       setbkcolor(15);
       graf_esps_sel(1,100,250);
       setcolor(LIGHTBLUE);
       textos(31);
       getch();
       fin_graf();
       break;
    case 8:
       info();
       break;
     } /* switch */
   } /* for */
   libera();
}

float gen_ran (int *ent, float flota){
  ent[1]=ent[0]*899;
  if ( ent[1]<0 )
  ent[1]+=32767+1;
  flota=ent[1];
  flota/=32767;
  return flota;
}

void gen_ruido ( int *ent){
float flota;
int j;
float adm=0;

 for (j = 0; j < 12 ;j++){
   flota=gen_ran(ent,flota);
   ent[0] = ent[1];
   adm += flota;
 }
 ent[2] = (int)(((adm -6.0)*ent[3]) + 0.5);
}

void gen_ts_aleat(int opc){
int k,*vec,l=1;
static long r = -100;
static float xm;
float flota,a;
time_t t;
```

```
    xm =(tam_traza/1000);
    do {
    a = ran1(&r);
    n_rep_esp = a * por_al *2;
     }while(!n_rep_esp);
     inicializa(20 + opc);
     vec = &t_ocur_esp[opc][1];
     for (k = 0;k < t_ocur_esp[opc][0];k + + ){
     a = poidev(xm,&r);
     t_ocur_esp[opc][k + 1] = a/10 * tam_traza;
     if (t_ocur_esp[opc][k + 1] < 0)
        t_ocur_esp[opc][k + 1] + = 32767;
     }
     ordena(vec,t_ocur_esp[opc][0]);
     if (!vec[0])
     vec[0] + = 1;
     t_ocur_esp[opc][1] = t_ocur_esp[opc][1];
     l + + ;
     for (k = 0;k < t_ocur_esp[opc][0]-2;k + + ){
    if (t_ocur_esp[opc][l-1] > tam_traza){
      t_ocur_esp[opc][0] = l-1;
      break;
    }
    if (t_ocur_esp[opc][k + 2] < t_ocur_esp[opc][l-
1] + n_pt )
      continue;
    t_ocur_esp[opc][l] = t_ocur_esp[opc][k + 2];
    l + + ;
     }
     t_ocur_esp[opc][0] = l-1;
     if ( t_ocur_esp[opc][0] ! = n_rep_esp){
     n_rep_esp = l-1;
     if (!( vec = (int*) calloc (l,sizeof(int) ) ) ){
      t_cad[14] = '1'; /* tiempos */
      textos(4);
      exit(1);
      }
      for (k = 0;k < l;k + + )
         vec[k] = t_ocur_esp[opc][k];
      free (t_ocur_esp[opc]);
      inicializa(20 + opc);
      for (k = 1;k < l;k + + )
         t_ocur_esp[opc][k] = vec[k];
      free (vec);
      }

    }

    void gen_ts_noal(int opc){
    int j,k,k2,no_imp_per;
    int i1,per_aux ,no_per_traza;
    float t,periodo;
    int imp_per[100];
    char c = 'R';
     while (c = = 'R' || c = = 'r'){
    limp_i(imp_per,10);
    j = k = no_imp_per = 0;
    i1 = per_aux = no_per_traza = 0;
    t = periodo = 0;
    k2 = 1;

    clrscr();
    textos(5);
    printf(" %d ", esps_traza[opc]);
    textos(6);
    scanf("%f", &periodo);
    periodo * = 100;
    if (!periodo)
      periodo = tam_traza -1;
    sprintf(t_cad,"%d",esps_traza[opc]);
    textos(7);
    for (k = 0;k < 100;k + + ){
      scanf("%f", &t);
      t* = 100;
      if ((int)(t) = = 0) break;
      imp_per[k] = t;
      no_imp_per + + ;
    }
    ordena(imp_per,no_imp_per);
    textos(8);
    if (no_imp_per > 1){
      for (i1 = 0;i1 < no_imp_per-1;i1 + + ){
       if (imp_per[i1] + n_pt > imp_per[i1 + 1]){
        imp_per[i1 + 1] = imp_per[i1] + n_pt + 1 ;
       }
      }
        per_aux = imp_per[no_imp_per-1] + n_pt -
imp_per[0];
       if (per_aux > periodo)  periodo = per_aux;
     }
     else
     if (periodo < n_pt + 1 )
       periodo = n_pt + 1 ;
     for (i1 = 0;i1 < no_imp_per;i1 + + )
        printf( " %d \n ", imp_per[i1]);
     sprintf(t_cad,"%d",esps_traza[opc]);
     textos(9);
     sprintf(t_cad,"%8.0f",periodo);
     textos(10);
     scanf (" %c", &c);
      }
      no_per_traza = ((tam_traza - imp_per[0])/(peri-
odo)) + 1;
      n_rep_esp = (no_per_traza*no_imp_per) + 1;
      inicializa(20 + opc);
      for (k = 0;k < no_imp_per;k + + ){
     t_ocur_esp[opc][k2] = imp_per[k];
     k2 + + ;
      }
```

```
        /* no_per_traza*/
     for (j = 0;j < no_per_traza;j + + ){
     for (k = 0;k < no_imp_per;k + + ){
       if ( k2 + k  > t_ocur_esp[opc][0] )
          break;
        t_ocur_esp[opc][k2 + k] = t_ocur_esp[opc][k
2 + k-no_imp_per] + periodo;
      if (t_ocur_esp[opc][k2 + k]  > tam_traza - n_pt)
       t_ocur_esp[opc][k2 + k] =  0;
     }
     k2 + = no_imp_per;
     if ( k2  > t_ocur_esp[opc][0] )
       break;
      }
      k2  = t_ocur_esp[opc][0];
      for (j =  t_ocur_esp[opc][0]; j  > 0 ; j--){
     if ( t_ocur_esp[opc][j] < = 0 )
        k2--;
     else{
        t_ocur_esp[opc][0] = k2;
        break;
     }
      }
    }

    /*  Despliegue de la información relativa a los
tiempos de disparo */
   /*  De las espigas incluidas en el tren          */

   void info(void){
   int i,i1;
   char salva[2], sal_nom[200] = "";
   FILE *f;
    clrscr();
    if((!genera) | | (!a_desc)){
      textos(11);
      getch();
      return;
    }
    t_cad[14] = '0';
    textos(12);
    for (i = 0;i < no_seles ;i + + ){
       sprintf(t_cad," %d ", esps_traza[i]) ;
       t_cad[14]  = '0';
       textos(13);
       printf(" %d \n\n", t_ocur_esp[i][0] );
       for (i1 = 1 ; i1 < = t_ocur_esp[i][0] ; i1  + + )
       printf ( "%6.2f\t", t_ocur_esp[i][i1]/100.);
       textos(3);
       getch();
    }
    textos(33);
    scanf("%s",&salva);
   if ((( idioma) && ((salva[0] = = 'S') | | (salva[0]
= = 's'))) | |
        ((!idioma) && ((salva[0] = = 'Y') | | (salva[0]
= = 'y')))){
       textos(18);
       scanf("%s",sal_nom);
       if ((f = fopen(sal_nom, "wt")) = = NULL){
         printf("\n\t    **********");
         strcpy(t_cad,sal_nom);
         textos(2);
         printf("\n\t    ********** \n");
         exit(1);
       }
       t_cad[14] = '1';
       textos(12);
       fprintf(f,"%s %s",descrip,t_cad);
       for (i = 0;i < no_seles ;i + + ){
         sprintf(t_cad," %d ", esps_traza[i]) ;
         t_cad[14]  = '1';
         textos(13);
         fprintf(f,"%s",t_cad);
         fprintf(f," %d \n\n", t_ocur_esp[i][0] );
         for (i1 = 1 ; i1 < = t_ocur_esp[i][0] ; i1  + + )
        fprintf (f, "%6.2f\t", t_ocur_esp[i][i1]/100.);
         fprintf(f,"\n");
       }
     }
    }

  /****    Crea e inicializa los vectores de
**esp[no_esp_max][n_pt + 4] ***/
  /****          * t_ocur_traza [tam_traza[0]]    y *
esps_traza [n_es] ***/

    void inicializa(int opc){
      int l;
      if (opc > = 20){

     l = opc-20;
     opc - = l;
     if ( l > n_es ){
       textos(14);
       exit(1);
     }
      }
     switch (opc){
     case 1:
     if (!( esps_traza = (int*) calloc (n_es,sizeof(int)
) ) ){
       t_cad[14] = '2';
       textos(4);
       exit(1);
     }
     limp_i(esps_traza,n_es);
     break;
      case 2:
```

```
    if (!( t_ocur_esp =(int**) calloc (n_es,sizeof (
int*) ) )){
      t_cad[14]='3';
      textos(4);
      exit(1);
    }
    limp_v(t_ocur_esp,n_es);
    break;
    case 20:
          if (!( t_ocur_esp[l] = (int*)
calloc(n_rep_esp,sizeof(int)) ) ){
      sprintf(t_cad,"%d",l);
      t_cad[14] = '4';
      textos(4);
      exit(1);
    }
    limp_i(t_ocur_esp[l],n_rep_esp);
    t_ocur_esp[l][0] = n_rep_esp-1;
    break;
    case 3:
      if (!( t_ocur_traza = (int *) calloc(
tam_traza,sizeof(int) ) ) ){
      t_cad[14] ='5';
      textos(4);
      exit(1);
    }
    limp_i(t_ocur_traza,tam_traza);
    break;
    }
  }

  void ins_ruido(void){
  int k,l = 0;
  char cad[] = "_\|/";
   clrscr();
   if (t_ocur_traza = = NULL){
     textos (15);
     getch();
     return;
   }
   if( sel_menup = = 4 ){
     sprintf(t_cad,"%f",((maximo*0.2)/10.));
     textos(16);
     scanf("%d",ent + 3);
   }
   clrscr();
   textos(17);
   ent[0] = 1;
   for (k = 0;k < tam_traza;k + + ){
     gen_ruido(ent);
     t_ocur_traza[k] + = ent[2];
     if (l > = 4)
       l = 0;
     if (k%100 = = 0 ){
       gotoxy(39,5);
       printf("%c",cad[l]);
       l + + ;
     }
   }
  }

void lee_datos(void){
char arch_inom[100],arch_i[100];
FILE *f;
int i,i1, pos_pto,len;
char cad[30] = "";

 textos(18);
 scanf ("%s",arch_inom);
 strcpy(arch_i,arch_inom);
 len = strlen(arch_inom);
 for (i = len-4;i < len ;i + + ){
    if ( arch_inom[i] = = '.' ){
   arch_inom[i] = '\0';
   break;
    }
 }
 strcat(arch_inom,".des");
 if ((f = fopen(arch_inom, "rt")) = = NULL)
 {
   printf("\n\t    ***********");
   strcpy(t_cad,arch_inom);
   textos(19);
   printf("\n\t    ********** \n");
   a_desc = 0;
 }
 else
   a_desc = 1;
 if (a_desc){
   fscanf (f,"%d", &tam_traza );
   fscanf (f,"\n%d", &no_selcs );
   for (i = 0;i < no_selcs ;i + + ){
     fscanf ( f,"\n %d",&esps_traza[i]) ;
     fscanf (f, " %d ",&n_rep_esp );
     n_rep_esp + + ;
     inicializa(20 + i);
     for (i1 = 1 ; i1 < = t_ocur_esp[i][0] ; i1 + + )
     fscanf ( f," %d ", &t_ocur_esp[i][i1]);

   }
   i1 = 0;
   fscanf ( f," %d ", &i1);
   if (i1)
     fscanf ( f," %d ", &ent[3]);
   strcpy(descrip,"");
   while ( (strcmp( cad,"&&")) && (!(feof(f))) ){
     strcat(descrip ,cad);
     strcpy(cad,"");
     fscanf ( f," %s ", &cad);
```

```
      strcat(descrip ," ");
    }
  }
  fclose(f);
  if ((f = fopen(arch_i, "rb")) == = NULL)
  {
    printf( "\n\t    ***********");
    strcpy( t_cad,arch_i );
    textos(2);
    if (a_desc){
       strcpy(t_cad,arch_inom);
       textos(20);
    }
    printf("\n\t    ********** \n");
    if (a_desc){
       for (i = 0;i < no_selcs;i + + )
       tren_espigas(i);
       if (ent[3])
       ins_ruido();
       textos(3);
       getch();
       return;
    }
    else{
      textos(3);
      getch();
      return;
    }
  }
  fread(t_ocur_traza,sizeof(int), tam_traza,f );
  fclose(f);
  textos(3);
  getch();
}

void libera(void){
int k;
 if (esps_traza ! = NULL){
free(esps_traza);
esps_traza = NULL;
 }
 if (t_ocur_traza! = NULL){
free(t_ocur_traza);
t_ocur_traza = NULL;
 }
 for (k = 0;k < no_selcs;k + + ){
free (t_ocur_esp[k]);
t_ocur_esp[k] = NULL;
 }
 no_selcs = 0;
 ent[3] = 0;
 if (t_ocur_esp ! = NULL){
free  (t_ocur_esp);
t_ocur_esp = NULL;
 }
}

/**** Inicializa vectores de interos ****/

void limp_i (int *vector, int tam){
int k;
   for (k = 0;k < tam;k + + )
   vector[k] = 0;
}

/**** Inicializa vectores de apuntadores ****/

void limp_v (int **vector, int tam){
int k;

 for (k = 0;k < tam;k + + )
   vector[k] = NULL;
}

void menup(void){
char l[2],t[2];
int band = 0;
  for(;;){
 clrscr();
 textos(1);
 scanf ( "%s", l);
 if (l[0] = = '1' | | l[0] = = '3'){
  for (;;){
   if (l[0] = = '1'){
     genera = 0;
     libera();
     inicializa(1);
        textos(21);
   }
   else
     textos (22);
   textos (23);
   scanf("%s", t);
   switch(t[0]){
    case '0':
     ts_aleat = 0;
     band = 1;
     break;
    case '1':
     ts_aleat = 1;
     band = 1;
        textos(24);
     scanf("%f",&por_al);
     break;
   }
   if (band)
   break;
   t[0] = '\0';
  }
}
```

```
    switch(l[0]){
     case '1':
        sel_menup = 1;
        return;
     case '2':
        sel_menup = 2;
        return;
     case '3':
        sel_menup = 3;
        return;
     case '4':
        sel_menup = 4;
        return;
     case '5':
        sel_menup = 5;
        return;
     case '6':
        sel_menup = 6;
        return;
     case '7':
        sel_menup = 7;
        return;
     case '8':
        sel_menup = 8;
        return;
     case '9':
        sel_menup = 9;
        return;
    }
    l[0] = '\0';
     }
    }

  /***** almacena en : vector[n_pt] = punto donde
está el pico ***/
  /*****                vector[n_pt + 1] = punto donde
está el valle ***/
  /*****              vector[n_pt + 3] = vector [n_pt-1]
original ***/
  /*****      modifica los diez ultimos valores en el
vector, para ***/
  /*****                              eliminar los escalones
***/

    void normaliza(int* vector) {
    float aux;
    int i,j;
    int dif = 0;

    int dif1 = 0;
    int p_pico = 0,p_valle = 0;

      for (i = 1;i < n_pt;i + + ){
        p_valle = p_min(vector,i,p_valle);
        p_pico = p_max(vector,i,p_pico);
      }
      vector[n_pt] = p_pico;
      vector[n_pt + 1] = p_valle;
      vector[n_pt + 3] = vector[n_pt-1];
      if (sel_menup ! = 7){
        dif = vector[0] - vector[n_pt-1];
        dif/ = 10;
        j = n_pt -10;
        for (i = 0;i < 9 ;i + + )
        vector[j + i] + = (dif*(i + 1));
        vector[n_pt-1] = vector[0];
      }
      dif1 = vector[0];
      for (i = 0;i < n_pt;i + + )
        vector[i]- = dif1;
    }

    /***   Ordena los elementos del vector   ***/

    void ordena(int * vect,int elem_vect){
    int a,b,t;

      for(a = 1;a < elem_vect;a + + )
     for(b = elem_vect-1;b > = a;--b){
        if(vect[b-1] > vect[b]){
        t = vect[b-1];
        vect[b-1] = vect[b];
        vect[b] = t;
        }
      }
    }

  /******  p_max y p_min  regresan el punto en el que
*****/
  /******  estan el valor máximo y mínimo en el vector
*****/

    int p_max(int* vector, int val1, int val2){
        return ( (vector[val1] < vector[val2]) ? val1 :
val2);
    }


    int p_min(int* vector, int val1, int val2){
        return ( (vector[val1] > vector[val2]) ? val1 :
val2);
    }

    void salvar(int*vector, int tam){
                                                    char
arch_inom[30] = "",ar_nom[30] = "",l1[2] = "",tipo[12
] = "";
    char texto[255],car;
    FILE *f;
    int k,l,i,pos_pto,i1,opc,band = 0;
```

```
  for(;;){
 clrscr();
 textos(25);
 printf( "\n\n\t\t  1) EXTRACEL " );
 printf( "\n\n\t\t  2) CLASIF " );
 printf( "\n\n\t\t  3) DISCOVERY \n\n\t\t  " );
 scanf ( "%s", l1);

 switch (l1[0]){
  case '1':
   strcpy(tipo, "EXTRACEL");
   band = 1;
   break;
  case '2':
   strcpy(tipo, "CLASIF");
   for (k = 0;k < tam;k + + )
     vector[k]* = -1;
   band = 1;
   break;
  case '3':
   strcpy(tipo, "DISCOVERY");
   for (k = 0;k < tam;k + + )
     vector[k]* = -1;
   band = 1;
   break;
 }
 if (band)
   break;
 l1[0] = '\0';
  }
  strcpy(t_cad,tipo);
  textos (26);
  scanf("%s",arch_inom);
  if (l1[0] = = '3'){
 if ((f = fopen(arch_inom, "wt")) = = NULL){
   printf("\n\t    ***********");
   strcpy(t_cad,arch_inom);
   textos(2);
   printf("\n\t    ********** \n");
   textos(3);
   getch();
   return;

 }
 for (k = 0;k < tam;k + + )
   fprintf (f,"%d ", vector[k] );
  }
  else {
 if ((f = fopen(arch_inom, "wb")) = = NULL){
   printf("\n\t    ***********");
   strcpy(t_cad,arch_inom);
   textos(2);
   printf("\n\t    ********** \n");
   textos(3);
   getch();
   return;
 }
 fwrite(vector,sizeof(int),tam,f);
  }
  fclose(f);
  if (l1[0]! = '1')
 for (k = 0;k < tam;k + + )
    vector[k]* = -1;
 strcpy(ar_nom,arch_inom);
 l = strlen(ar_nom);
 for (i = l-4;i < l ;i + + ){
   if ( ar_nom[i] = = '.' ){
  ar_nom[i] = '\0';
  break;
   }
 }
 strcat(ar_nom,".des");
 if ((f = fopen(ar_nom, "wt")) = = NULL)
{
  printf("\n\t    ***********");
  strcpy(t_cad,ar_nom);
  textos(2);
  printf("\n\t    ********** \n");
  textos(3);
  getch();
  return;
}
fprintf (f,"%d", tam_traza );
fprintf (f,"\n%d", no_selcs );
for (i = 0;i < no_selcs ;i + + ){
    fprintf ( f,"\n %d", esps_traza[i]) ;
    for (i1 = 0 ; i1 < = t_ocur_esp[i][0] ; i1 + + )
    fprintf ( f," %d ", t_ocur_esp[i][i1]);
}
if (!ent[3])
  fprintf ( f,"\n %d",0);
else
  fprintf ( f,"\n %d %d", 1,ent[3]) ;
textos(32);
strcpy(texto,"\0");
for (i = 0 ; i < 254; i + + ){
  car =getche();
  switch (car){
    case '\r':
    texto[i] = '\0';
    i = 255;
    break;
    case '\b':
    i- = 2;
    break;
    default :
    texto[i] = car;
    texto[i + 1] = '\0';
    break;
```

```
    }
  }
  fprintf ( f,"\n%s &&",texto ) ;
  strcpy(descrip,texto);
  fclose(f);
}

void sel_esps(int band){

int t,i,existe = 0;

clrscr();
textos(27);
if (band && no_selcs){
  textos(28);
  for (i = 0;i< no_selcs;i + + )
    printf( "\t%d",esps_traza[i]);
  printf("\n\n");
}
textos(29);
for (;no_selcs < n_es;){
  scanf("%d", &t);
  if (t = = 0) break;
  if ((t > 0)&&(t < = esp_max)){
    for (i = 0;i< no_selcs;i + + )
   if (t = = esps_traza[i]){
     existe = 1;
     break;
   }
   else
     existe = 0;
    if(existe){
   sprintf(t_cad,"%d",t);
   textos(30);
    }
    else{
   esps_traza[no_selcs] = t;
   no_selcs + + ;
    }
  }
}
if (!band)
  ordena(esps_traza,no_selcs);
}
```

```
    /*** Hace el vaciado de los tiempos de ocurrencia
de las espigas ***/
      /***        en el tren de espigas final
***/

    void tren_espigas(int opc){
    int j,j1 = 0,n_pt_a,k,t,l;
    int *arch = NULL;

      n_pt_a = n_pt;
      if (!( arch = (int*) calloc( n_pt + 4 ,sizeof(int) )
) ){
      t_cad[14] = '6';
      textos(4);
      exit(1);
      }
      limp_i(arch,n_pt + 4);
      dat_esp(opc,arch);
      printf("%d\t",arch[n_pt]);
      j1 = arch[n_pt];
      for (j = 1;j< = t_ocur_esp[opc][0];j + + ){
     t = t_ocur_esp[opc][j];
     n_pt_a = n_pt;
     l = 0;
     if (t&& (t + n_pt < tam_traza) ){          /* t != 
0 */
       if( (t-j1) > 0 )
       t- = j1;
       else{
       l + = j1;
       n_pt_a = n_pt -j1;
       }
       for (k = t;k < t + n_pt_a || l< n_pt;k + + ,l + + )

       t_ocur_traza[k] + = arch[l];
     }
      }
     free(arch);
   }


 void main(){
   general();

    }
```

C.3 DIST_POI.C

```
/********** DIST_POI.C ********/

 /* float ran1(long *idum) generador de números
aleatorios de Park y Miller. Regresa un random con
desviación estándar entre 0 y 1. Sin incluir los valores
de frontera. Con ayuda de inum, inicializa con un
 entero negativo, después de ésta etapa inum no
debe de ser alterado en una secuencia.          */

 #define IA 16807
 #define IM 2147483647
 #define AM (1.0/IM)
 #define IQ 127773
 #define IR 2836
 #define NTAB 32
 #define NDIV (1+(IM-1)/NTAB)
 #define EPS 1.2e-7
 #define RNMX (1.0-EPS)
 #include <<math.h>>
 #define PI 3.141592654

 float ran1(long *idum){
   int j;
   long k;
   static long iy=0;
   static long iv[NTAB];
   float temp;
   if (*idum <= 0 || !iy) {     /* Inicializa. */
    if (-(*idum) < 1)
      *idum=1;   /* Se asegura de que idum sea ! =
0.*/
    else
      *idum = -(*idum);

    for (j=NTAB+7;j>=0;j--) {
      k=(*idum)/IQ;
      *idum=IA*(*idum-k*IQ)-IR*k;
      if (*idum < 0) *idum += IM;
      if (j < NTAB) iv[j] = *idum;
    }
    iy=iv[0];
   }
   k=(*idum)/IQ;
     /*  arranca aquí cuando no se está inicializando
*/

   *idum=IA*(*idum-k*IQ)-IR*k;
   /* Calcula idum=(IA*idum) % IM   */

   if (*idum < 0)
    *idum += IM;
   j=iy/NDIV;
    /* Estará en el rango de 0..NTAB-1. */
   iy=iv[j];
    /* proporciona el valor qu estaba almacenado en
la tabla    y almacena el nuevo valor.*/

   iv[j] = *idum;
   if ((temp=AM*iy) > RNMX)
    return RNMX;
    /* Evita dar los valores de frontera.*/
   else return temp;
 }

 float gammln(float xx) {
   /* Regresa el valor de ln[Gamma(xx) para xx > 0.
*/
    /* La aritmética interna se lleva a cabo con en
doble precisión */
    double x,y,tmp,ser;
      static double cof[6]={76.18009172947146,-
86.50532032941677,
      24.01409824083091,-1.231739572450155,
      0.1208650973866179e-2,-0.5395239384953e-5};
    int j;
    y=x=xx;
    tmp=x+5.5;
    tmp -= (x+0.5)*log(tmp);
    ser=1.000000000190015;
    for (j=0;j<=5;j++)
      ser += cof[j]/++y;
    return -tmp+log(2.5066282746310005*ser/x);
 }

 float poidev(float xm, long *idum) {
   /* Regresa un valor entero que es aleatorio a partir
de una distribución de Poisson de media xm, usando
a ran1 (idum ) como generador de números aleatorios
con desviación uniforme. */

    static float sq,alxm,g,oldm=(-1.0);

       /* oldm permite verificar si xm ha cambiado
desde la última llamada.*/
    float em,t,y;
    if (xm < 12.0) {
      /* Usa el método directo. */
      if (xm != oldm) {
```

```
        oldm = xm;
        g = exp(-xm);
          /* Si xm es nuevo, calcula el exponencial.*/
      }
      em = -1;
      t = 1.0;
      do {
          /* Es equivalente multiplicar desviciones
uniformes a sumar desviaciones exponenciales. Aunque no es necesario calcular el log, sino únicamente
comparar el exponencial pre-calculado. */
        + + em;
        t * = ran1(idum);
      } while (t > g);
    }
    else {
      /* Usa el método de rechazo  */
      if (xm ! = oldm) {
          /*  si xm cambia, entonces se precalculan
algunas de las funciones descritas a continuación. */
        oldm = xm;
        sq = sqrt(2.0*xm);
        alxm = log(xm);
        g = xm*alxm-gammln(xm + 1.0);
          /* La función gammln es el logaritmo natural
de la funciñón gama */
      }
      do {
        do {
          y = tan(PI*ran1(idum));
          em = sq*y + xm;
           /* em es y, con un corrimiento y escalada
*/
        } while (em < 0.0);
        /* Se rechaza si está en una región de
        probabilidad 0 */
        em = floor(em);
              t = 0.9*(1.0 + y*y)* exp(em*alxm-
gammln(em + 1.0)-g);
          /* El radio de la función de distribución
requerida en la función de comparación, aceptamos
o rechazamos comparando con otra desviación uniforme. Se eligió el factor 0.9, de esta manera t nunca
          excede el 1. */
      } while (ran1(idum) > t);
    }
    return em;

  }
```

C.4 GRAFICA.C

```
        /************  grafica.c ***********/

  /*** cierra los gráficos y libera la memoria ocupada
***/
   void fin_graf(void) {
      closegraph();
   };

    /* Grafica  las espigas seleccionadas  */

   void graf_esps_sel(int opc, int x, int y){
   int   x1, y1,x2,y2,x22,i;
   char numer_esp[4];
   float esc_verd = 1;
   int *arch = NULL;

     if (!( arch = (int*) calloc( n_pt + 4 ,sizeof(int) )
) ){
      t_cad[14] = '6';
      textos(4);
      exit(1);
       }
      setcolor(LIGHTBLUE);
      textos(34);
     /* selecciona el tipo de linea  */
     setlinestyle(USERBIT_LINE,3, 3);
     setcolor(LIGHTBLUE);
     line(x-50,y,500,y);
     setlinestyle(SOLID_LINE, 1, 1);
     for (i = 0;i < no_seles;i + + ){
      limp_i(arch,n_pt + 4);
      dat_esp(i,arch);
      setcolor(i + 2);
      pinta(arch,esc_verd,x,y,n_pt,1);
      sprintf(numer_esp,"%i",esps_traza[i]);
      x2 = (i%10)*20 + 350;
      y2 = 50 + 8 + (i/10)*12;
      outtextxy(x2,y2,numer_esp);
     }
     free(arch);
   }
```

```
/**** Inicializa modo gráfico ***/

void inic_graf (void)
{

int ctrlr,modo,errorcode;

   ctrlr = DETECT;
   initgraph(&ctrlr, &modo,"");
   errorcode = graphresult();
   if (errorcode != grOk)
   {
  textos(35);
  printf(" : %s\n", grapherrormsg(errorcode));
  exit(1);
   }
}

/**** Pinta la espiga en los cuadritos ***/
/***      o en la traza          ***/


void pinta( int * dat_esps,float fac_esc, int x,
int y,int tam,int t_gr)
{

float auxy,auxx,fact_esc_x = 0, x_cuad;
char num_arch[2];
int j,col,y1;
int k2 = 0,k22 = 0,ban_et = 0,aux1,auy1;
char ray[2],tiempo[4];

    col = getcolor();
   col + + ;
       /* usar t_gr como bandera al graficar con
número o sin él */
      if (fac_esc = = esc_cuad){
      /***  despliega el número de espiga  ***/
      /***   en los cuadritos de selección ***/
      /***  y proporciona la escala        ***/
      strcpy(num_arch ,"");
      sprintf(num_arch ,"%i",dat_esps[130]);
      auxx = x + 4;
      auxy = y + 4;
      outtextxy(auxx,auxy,num_arch);
      fact_esc_x = fac_esc*8;
       } else
       if (!esc_cuad){
      if (t_gr){/*fac_esc = = esc_verd) */ /* grafica
traza o espigas solas */
      fac_esc* = 0.5;
      fact_esc_x = fac_esc*8;
      }
      else{
      /*** escala para dibujar en la traza ***/
      fact_esc_x = fac_esc*0.125;/**0.2;0.5;*/
      fac_esc = fac_esc*0.125; /* = 1.1*/
      }
       }
       auxx = (float)x + 3;
       /****  dibuja la espiga    ***/
       y1 = y;
       strcpy(ray,"|");
       if (!t_gr)
       for ( j = 0; j < tam; j + + ){

       auxy = ((float) dat_esps[j] * fac_esc) + y1;
       auxx = auxx + fact_esc_x*0.7;
          k22 = j/1000; /**** segs   **/
          k2 = j/100; /***  msegs  **/
          setcolor(LIGHTBLUE);
          if ((ban_et != k2) || (!j)){
        if (!t_gr){
         if (k22*1000 = = j){
           setcolor(11);
                 outtextxy(auxx-(textwidth("|")/2),(300*
fac_esc) + y1 ,ray);
           setcolor (LIGHTBLUE);
           sprintf(tiempo,"%i",k22*5*2);
                 outtextxy(auxx-(textwidth("|")/2),(360*
fac_esc) + y1 ,tiempo);
           ban_et = k2;
         }else
         if (k2*100 = = j)
                 outtextxy(auxx-(textwidth("|")/2),(300*
fac_esc) + y1 ,ray);
        }
         }
         setcolor(11);
       putpixel( auxx + 0.5,auxy,col);
       if ( auxx > 640 ){
        y1 + = y;
        auxx = 0;
       }
       }
       else {
       auy1 = ((float) dat_esps[0] * fac_esc) + y1;
       aux1 = auxx + fact_esc_x*0.8;
       moveto (aux1,auy1);
       for ( j = 1; j < tam; j + + ){
        auxy = ((float) dat_esps[j] * fac_esc) + y1;
        auxx = auxx + fact_esc_x*0.7;
        lineto( auxx + 0.5,auxy);
        if ( auxx > 640 ){
         y1 + = y;
         auxx = 0;
        }
       }
         }
      }
```

C.5 TEXTOS.C

```
/********** TEXTOS.C **********/

void textos ( int tex){
char aux[20];
   if (tex = = 0){
   printf( "\n\n\t      ***********");
   printf( "\n\tSelecciona el idioma a utilizar ");
   printf( "\n\n\t      ***********");
   printf( "\n\tPlease, select the language you want
to use ");
   printf( "\n\n\t      ***********");
   printf( "\n\n\t\t  1) Español" );
   printf( "\n\n\t\t  2) English \n\n\t\t\t\t") ;
   return;
   }
   if (idioma) {
     switch (tex){
    case 1:
      printf( "\n\n\t\t  1) Generar traza" );
      printf( "\n\n\t\t  2) Cargar traza" );
       printf( "\n\n\t\t  3) Agregar una espiga a la
traza");
       printf( "\n\n\t\t  4) Agregar ruido a la señal
");
      printf( "\n\n\t\t  5) Graficar la traza ");
      printf( "\n\n\t\t  6) Salvar traza");
      printf( "\n\n\t\t  7) Traslapar espigas");
     printf( "\n\n\t\t  8) Información de la traza ");
      printf( "\n\n\t\t  9) Salir \n\n\t\t\t" );
      break;
    case 2:
      printf("\n\t No fue posible abrir el archivo :
%s \n\t",t_cad);
      break;
    case 3:
      printf( "\n\n\t Presiona cualquier tecla para
continuar ...");
      break;
    case 4:
        printf( "\n\t      ***** No hay memoria
suficiente");
      printf( "\n\t      para ");
       switch( t_cad[14]){
       case '1':
          printf (" el vector de tiempos ****** \n");
          break;
       case '2':
          printf (" el vector de espigas  ****** \n");
          break;
       case '3':
            printf (" la matriz de tiempos    ******
\n");
          break;
       case '4':
          t_cad[14] = ' ';
              printf ( " el arreglo No. %s ******
\n",t_cad);
          break;
       case '5':
           printf ( " el arreglo de ocurrencias ******
\n");
          break;
       case '6':
             printf(" el arreglo de datos de la espiga
****** \n");
          break;
       }
      break;
    case 5:
      printf( "\n\n Cual es el periodo de repetición de
los \n");
      printf( " \t   impulsos de la espiga : ");
      break;
    case 6:
      printf(" en msegs \n\t");
      printf( "\n   periodo = 0   = > sin periodo \n
");
      break;
    case 7:
       printf( "\n Cuales son los tiempos del primer
ciclo \n" );
      printf( "    para la espiga No.%s en msegs \n\t
",t_cad);
        printf( "\n\t\t Para finalizar la entrada de
datos");
      printf( "\n\t\t    de cada espiga : t = 0 \n");
      break;
    case 8:
      printf( "\n\n Los tiempos que serán utilizados
\n");
      printf( "    para cada periodo son :   \n\t\
       ( dados en puntos dentro del arreglo) \n\t\t");
      break;
    case 9 :
     printf( "\n\n el periodo de repetición de los \n"
);
      printf( "  impulsos de la espiga  : %s",t_cad);
      break;
    case 10:
     printf( " es de  %s puntos",t_cad);
```

```
      printf("\n\n presiona 'C' para continuar ");
          printf("  \n          'R' para redefinir los
tiempos\n");
       break;
     case 11:
      printf(" \n\n\t\t No se ha generado algún tren
en memoria\
       \n\n\t\t o al cargar el tren no se encontró el
archivo \
       \n\n\t\t descriptor ");
       printf( "\n\n\t Presiona cualquier tecla para
continuar ...");
       break;
     case 12:
       if (t_cad[14] = = '1') {
       if (ent[3]){
            sprintf(t_cad,"\n\n\t Información de la
traza \
         \n\n\t Tamaño de la traza :  %6.1f msegs\
         \n\n\t Señal con ruido, varianza:  % d \
          \n\n\t Número de espigas que contiene :
%d",
         tam_traza /100., ent[3],no_seles );
       }
       else{
            sprintf(t_cad,"\n\n\t Información de la
traza \
         \n\n\t Tamaño de la traza :  %6.1f msegs\
          \n\n\t Número de espigas que contiene :
%d",
         tam_traza /100.,no_seles );
       }
       }
       else {
         printf("\n\n\t Información de la traza ");
         printf("\n\n\t %s ",descrip);
          printf("\n\n\t Tamaño de la traza :  %6.1f
msegs",
         tam_traza /100. );
         if (ent[3])
         printf("\n\n\t Señal con ruido, varianza:  %
d ",
         ent[3] );
         printf("\n\n\t Número de espigas que con-
tiene : %d",
         no_seles );
       }
       break;
     case 13:
       strcpy(aux,t_cad);

       if (t_cad[14] = = '1') {
       t_cad[14] = ' ';
       sprintf ( t_cad, "\n\n\t Espiga No. %s  \
         \t (tiempo de las ocurrencias en msegs )\n\
            \n  Número de repeticiones de la espiga
:",aux);
       }
       else {
       t_cad[14] = ' ';
       printf ( "\n\n\t Espiga No. %s",t_cad);
       printf ( "\t (tiempo de las ocurrencias en msegs
)\n\
          \n  Número de repeticiones de la espiga :");
       }
       break;
     case 14:
       printf( "\n\t     ***** el numero máximo de ");
      printf(" selecciones es  : %d ****** \n",n_es);
       break;
     case 15:
       printf(" \n\n\t\t No se ha generado o cargado \
       \n\n\t\t algún tren en memoria. \
        \n\n\t\t Por lo que no se puede ejecutar ,este
módulo ");
         printf( "\n\n\t Presiona cualquier tecla para
continuar ...");
       break;
     case 16:
         printf("\n\n\tEl valor de varianza para el
20%%");
       printf(" de la señal es:  %s \n\n",t_cad);
      printf("\n\n\t Cual es el valor de la varianza ? ");
       break;
     case 17:
       printf(" \n\n\t\t Agregando ruido a la señal ");
       break;
     case 18:
         printf("\n\t  Cual es el nombre del archivo
:\n\t");
       break;
     case 19:
       printf("\n\t No fue posible abrir el archivo %s
\n\t ",
       t_cad);
        printf("\n\t No se tendrá información de las
espigas que \n\
         componen el tren \n\t");
       break;
     case 20:
       printf("\n\t Se reconstruir  la traza a partir del
\n\
                  archivo descriptor de tiempos :
%s\n",t_cad);
       break;
     case 21:
       printf( "\n\n\t\t   Generar traza" );
       break;
     case 22:
       printf( "\n\n\t\t   Agregar espigas" );
```

```
        break;
      case 23:
       printf( "\n\n\t\t  1) Con tiempos aleatorios" );
            printf( "\n\n\t\t   0) Con tiempos pre-
determinados \n\n\t\t\t") ;
        break;
      case 24:
            printf( "\n\n\t\t    Porcentaje de
concentración \n\
                  \n\t\t    de generación de tiempos
\n\t\t" );
        break;
      case 25:
        printf( "\n\n\t\t\t\t   Formato de Salida " );
        break;
      case 26:
        printf( "\n\t Cual es el nombre del archivo de
salida \
          \n\t   en formato  %s :\n\t",t_cad);
        break;
      case 27:
        printf( "\n\n\t Selección de espigas a insertar
en el tren \n");
          printf( "\n\t\t Para finalizar la entrada de
datos");
        printf( "\n\t\t      espiga =0  \n");
        break;
      case 28:
        printf( "\n\t Las espigas seleccionadas hasta
el momento son :\n");
        break;
      case 29 :
        printf( "\n cuales son las espigas a seleccionar
\n");
        break;
      case 30 :
          printf(" \n\n la espiga %s   ya había sido
seleccionada \n",t_cad);
        break;
      case 31 :
         outtextxy(200,450, " Presiona cualquier tecla
para continuar ...");
        break;
      case 32 :
         printf( "\n Descripción del archivo a grabar
\n");
        break;
      case 33 :
       printf( "\n Deseas almacenar la información en
un archivo (S/N)\n");
        break;
      case 34 :
        outtextxy(350,35,"Espigas desplegadas : ");
        break;
      case 35 :
        printf("\nError de Gráficos");
        break;
       }
     }
     else {
      switch (tex){
      case 1:
        printf( "\n\n\t\t  1) Train Generation" );
        printf( "\n\n\t\t  2) Load train"  );
        printf( "\n\n\t\t  3) Add spikes ");
        printf( "\n\n\t\t  4) Add noise     ");
        printf( "\n\n\t\t  5) Display train");
        printf( "\n\n\t\t  6) Save ");
        printf( "\n\n\t\t  7) Overlap spikes  ");
        printf( "\n\n\t\t  8) Train information");
        printf( "\n\n\t\t  9) Quit  \n\n\t\t\t" );
        break;
      case 2:
       printf("\n\t Cannot access file : %s \n\t ",t_cad);
        break;
      case 3:
        printf( "\n\n\t Press any key to continue  ...");
        break;
      case 4:
        printf( "\n\t     ***** There is not enough mem-
ory ");
        printf( "       for the ");
        switch( t_cad[14]){
          case '1':
            printf (" time array ****** \n");
            break;
          case '2':
            printf (" spike array     ********* \n");
            break;
          case '3':
            printf (" time matrix    ****** \n");
            break;
          case '4':
            t_cad[14] = ' ';
            printf ( " array No. %s ****** \n",t_cad);
            break;
          case '5':
            printf ( " signal array      ****** \n");
            break;
          case '6':
            printf ( " data spike array  ****** \n");
            break;
        }
        break;
      case 5:
        printf( "\n\n\t\t\t For  SPIKE  : ");
        break;
      case 6:
        printf("\n\n\t Spike period in msecs \n\t");
```

```
        printf( "\n\t\t   period = 0,  for  no period
\n\n\t\t ");
        break;
      case 7:
       printf( "\n\t Enter spike times for the fist cicle
in msecs   \n" );
         printf( "\n\t\t  t = 0 : For finishing data en-
trance. \n\n\t");
        break;
      case 8:
        printf( "\n\n\t The spike times to be used in
each period are :\
          \n\t( values are done in points within the
memory array ) \n\t\t");
        break;
      case 9 :
          printf( "\n\n\t Spike period for spike   :
%s",t_cad);
        break;
      case 10:
        printf( " is %s points ",t_cad);
        printf("\n\n Press 'C' to continue");
        printf("  \n      'R' to change the data \n");
        break;
      case 11:
       printf(" \n\n\t\t There is not a train loaded on
memory\
         \n\n\t\t or there was not a descriptor file \
         \n\n\t\t when thr train was loaded ");
        printf( "\n\n\t Press any key to continue  ...");
        break;
      case 12:
        if (t_cad[14] = = '1') {
        if (ent[3]){
           sprintf(t_cad,"\n\n\t     Train Information
\
          \n\n\t Train size  :  %6.1f  msecs \
          \n\n\t Noisy signal, variance:  % d \
            \n\n\t Number of different spikes in the
train : %d",
          tam_traza /100., ent[3],no_seles );
        }
        else{
           sprintf(t_cad,"\n\n\t     Train Information
\
          \n\n\t Train size  :  %6.1f  msecs \
            \n\n\t Number of different spikes in the
train : %d",
          tam_traza /100.,no_seles );
        }
        }
        else {
        printf("\n\n\t     Train Information    \
          \n\n\t Train size  :  %6.1f  msecs ",
          tam_traza /100.);
        if (ent[3])
           printf("\n\n\t Noisy signal, variance :  % d
",ent[3] );
        printf("\n\n\t Number of different spikes in the
train : %d",
           no_seles );
        }
        break;
      case 13:
        strcpy(aux,t_cad);
        if (t_cad[14] = = '1') {
        t_cad[14] = ' ';
        sprintf ( t_cad,"\n\n\t Spike No. %s \
            \t ( time in msecs ) \n\
              \n    Total number of spikes in this class :
",aux);
        }
        else {
        t_cad[14] = ' ';
        printf ( "\n\n\t Spike No. %s",t_cad);
        printf ( "\t ( time in msecs ) \n\
            \n    Total number of spikes in this class : ");
        }
        break;
      case 14:
        printf( "\n\t     ***** maximum number ");
         printf( "  of selections is :  %d  spikes ******
\n",n_es);
        break;
      case 15:
        printf(" \n\n\t\t There is not a train on \
           \n\n\t\t memory to wich add the noise.");
        printf( "\n\n\t Press any key to continue  ...");
        break;
      case 16:
           printf("\n\n\t The variance value for the
20%%");
        printf(" of the signal is :  %s \n\n", t_cad);
        printf("\n\n\t New variance value :  ");
        break;
      case 17:
        printf(" \n\n\t\t Adding noise to the signal ");
        break;
      case 18:
        printf("\n\t Enter file name :  \n\t");
        break;
      case 19:
       printf("\n\t Cannot open file : %s  \n\t ",t_cad);
        printf("\n\t There will not be information avai-
ble about the \n\
            Spikes that have generated the train \n");
        break;
      case 20:
        printf("\n\t The train will be reconstructed by
\n\
```

```
            using the information on the descriptor \
            file : %s \n ",t_cad);
          break;
        case 21:
          printf( "\n\n\t\t   Train Generation");
          break;
        case 22:
          printf( "\n\n\t\t   Add spikes " );
          break;
        case 23:
          printf( "\n\n\t\t  1) Random firing times " );
           printf( "\n\n\t\t  0) Predefined firing times
 \n\n\t\t\t\t") ;
          break;
        case 24:
          printf( "\n\n\t\t   Percentage level to be con-
sidered   \n\
            \n\t\t for times generation \n\t\t");
          break;
        case 25:
          printf( "\n\n\t\t\t    Output Format " );
          break;
        case 26:
          printf(  "\n\t File name for output file \
            \n\t   in format %s : \n\t ",t_cad);
          break;
        case 27:
          printf( "\n\n\t Select the spikes to be inserted
in the train \n");
          printf( "\n\t\t Spike number = 0 : Finish data
entrance");
          break;
        case 28:
          printf( "\n\t Spikes already selected are   :\n");
          break;
        case 29 :
          printf( "\n\n\t Selected spikes :    \n\n\t\t");
          break;
        case 30 :
           printf(" \n\n Spike %s had already been se-
lected \n",t_cad);
          break;
        case 31 :
           outtextxy(200,450, " Press any key to continue
...");
          break;
        case 32 :
          printf( "\n File description  \n");
          break;
        case 33 :
          printf( "\n Do you want to save the information
in a file (Y/N)\n");
          break;
        case 34 :
          outtextxy(350,35,"Overlaped Spikes :");
          break;
        case 35 :
          printf("\nGraphics error ");
          break;
        }
      }
    }
```