5. DESARROLLO DEL EXPERIMENTO

Una vez que ya se ha hablado del OTDR, de la fibra y del procesado digital de señales, no queda más que explicar como se llevó a cabo este experimento. Haremos un poco de hincapié en el lenguaje de programación que se usó y la forma como se capturaron las señales para ser procesadas por este lenguaje. Antes de hablar del lenguaje de programación MATLAB, de forma que haya una secuencia en el proceso de este experimento, hay que explicar la configuración del OTDR antes de realizar cualquier medición. En seguida de la adquisición de los datos de tal medición y finalmente del procesado de estos datos para la obtención de los resultados deseados.

5.1 Puertos de entrada y salida del OTDR

La forma más sencilla de interactuar entre el OTDR y una computadora es la comunicación serial. El OTDR dentro de sus puertos y formas de intercambiar datos cuenta con un puerto serial, un puerto paralelo y un lector de discos 3.5 pulgadas. Desde luego que la única manera de obtener un formato es mediante el disco 3.5, ya que éste ingresa a una computadora y es leído por algún software ya existente, sin embargo, la extensión o el formato de las trazas (muestras) que se realizaron no son legibles para ningún lenguaje, a excepción de la misma marca: HP Agilent Technologies. Por este motivo, la siguiente solución se encontraba entre el puerto paralelo y el puerto serial. Se eligió el puerto serial porque al buscar la forma de configurar el puerto serial en MATLAB pareció muy sencillo y no se consideró indagar más en la comunicación paralela. Además, el software OTDR Trace Viewer II plus, del cual hablaremos más adelante, también se comunica de forma serial con el OTDR.

5.2 Configuración del OTDR y de la conexión serial

El OTDR cuenta con un cable RS-232 y un puerto para éste, de la misma forma la computadora. Únicamente se requiere de un código que sirva de intérprete entre el OTDR y la computadora. Mediante MATLAB se llevó a cabo la configuración del puerto serial, así como de otros detalles de la comunicación: 19,200 Baudios, 8 bits por dato, control de flujo por hardware, sin paridad, lectura asíncrona y continua, etc. Estas y más características de la configuración el puerto serial se pueden apreciar en el APÉNDICE A. De la misma manera podemos ver cómo se configuró el OTDR a través de MATLAB. Características como: brillo y contraste de la pantalla del OTDR, lugar de inicio de la medición, distancia a medir, resolución, índice de refracción, número de puntos por traza, etc. En el APÉNDICE A, reitero, se encuentra el código de la conexión serial y configuración del OTDR.

Las instrucciones utilizadas para la configuración del OTDR están basadas en el estándar IEEE 488.2 y de la SCPI (Standard Commands for Programmable Instruments). Obtenidas del manual de programación *HP Agilent Technologies OTDR E0401*.

5.3 Obtención de trazas

La obtención de la medición de las trazas que anteriormente configuramos de forma serial en el OTDR, no se realizó bajo el mismo lenguaje de programación MATLAB, fue necesario un intermediario, como ya se había comentado, las trazas solo se pueden leer por software de la misma marca HP Agilent Technologies, este puente entre el OTDR y MATLAB se llama OTDR Trace Viewer II plus. Una vez que configuramos el aparato mediante MATLAB y a través de una conexión serial, únicamente se inicia el software OTDR Trace Viewer II plus sin tener que hacer ninguna alteración física a la conexión. Debidamente terminada la traza en el OTDR, es decir una vez transcurridos

los 3 minutos recomendados para una buena traza, el software OTDR Trace Viewer II plus se conecta con el OTDR de forma serial igualmente y obtiene la gráfica de la traza que en ese momento despliegue el ODTR en pantalla. La Figura (5.1) muestra como se ve una traza obtenida desde el OTDR.

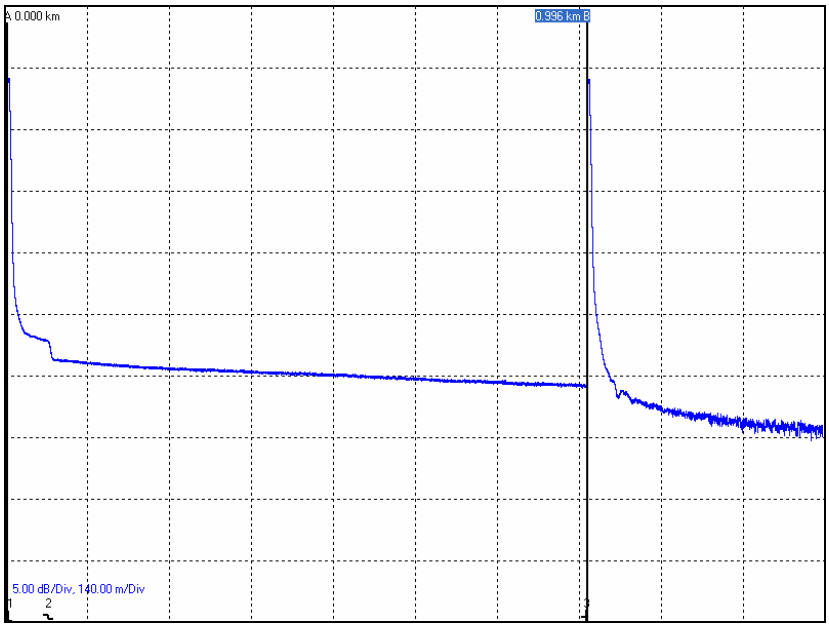

Figura 5.1 Traza del OTDR vista en el OTDR Trace Viewer II plus

El paso siguiente es muy sencillo, dado que el programa OTDR Trace Viewer II plus tiene la opción de exportar gráficas. Por otro lado, MATLAB tiene la opción de importar datos, solamente necesitamos saber de qué forma nos conviene trabajar con la gráfica, probablemente no haya necesidad de importar, sino hacer de los datos ASCII una tabla de datos en formato *.txt* .

Como vimos en el capítulo 4, hay dos gráficas que nos interesan particularmente , una es la del pulso de entrada (Figura 4.3) y la otra la del pulso de salida (Figura 4.4). Ahora a partir de la Figura 5.1, debemos seleccionar la opción de exportar traza en código ASCII y seleccionar las siguientes opciones (Figura 5.2) para mejor manipulación de los datos.

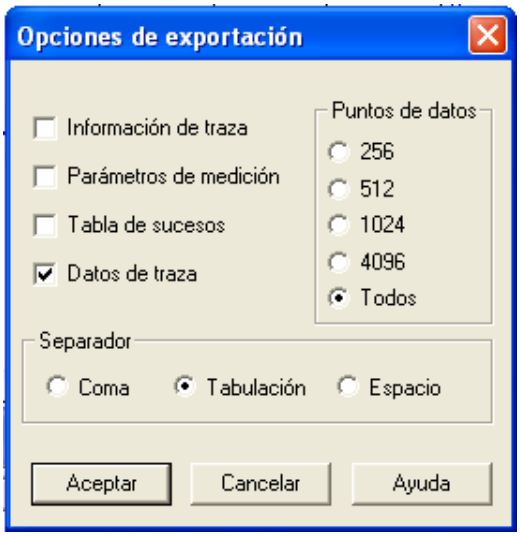

Figura 5.2 Configuración de la exportación de una traza desde el OTDR Trace Viewer II plus

5.3.1 Manipulación de la traza

La traza que vemos en la Figura 5.1 no es precisamente lo que necesitamos para poder trabajar, sino como acabamos de mencionar, nos interesa el enfoque del pulso de entrada y el de salida. Para lograr obtener estos pulsos, primero necesitamos la traza en forma de tabla de datos, es decir una matriz de 2 columnas, una con el vector de N muestras de la distancia recorrida y la segunda con el vector que contenga N muestras de la información en el eje Y, es decir la energía de la luz emitida dentro de la fibra, vista en forma de pulsos y sucesos no reflectivos, y la atenuación entre dos puntos.

Para realizar dos gráficas, una conteniendo el pulso de entrada y otra con el pulso de salida, es necesario basarnos en la traza original. Requerimos medir exactamente, la distancia que recorre nuestro pulso el tiempo de su duración. Una vez teniendo la distancia, es necesario ver de qué punto a qué punto no hay pérdida de energía es decir, que lo que seleccionemos tenga forma de una curva gaussiana, iniciando y terminando en el mismo nivel. Este procedimiento se realiza para los dos pulsos, sin embargo, el segundo pulso será siempre más ancho, por la dispersión y el ensanchamiento del haz de

luz. No nos preocuparemos tanto por la distancia que recorra, sino que tengan el mismo nivel sus extremos (en el eje de las ordenadas). Este procedimiento se puede realizar en el cuaderno de notas de Windows, no requiere más que comprobar los puntos que se midieron en la gráfica original vista en el OTDR Trace Viewer II plus, con los datos que tengamos en nuestros dos vectores. La nueva tabla, que contiene el pulso únicamente, debe ser mucho más pequeña que los miles de datos de la original y se guarda con la extensión *.txt*. Una vez realizado lo anterior, ahora solamente es tarea de MATLAB realizar los cálculos necesarios para completar el experimento.

5.4 Creación de la Interfaz Gráfica en MATLAB

MATLAB es un lenguaje de alto nivel para uso computacional técnico. Se enfoca a cálculos computacionales, visualización y programación en un ambiente amigable al usuario, donde los problemas y soluciones se expresan en una notación matemática familiar. Dentro de sus aplicaciones destacan:

-Cálculos computacionales y matemáticas

-Desarrollo de Algoritmos

-Desarrollo de modelos y prototipos, simulaciones

-Análisis de datos, exploración y visualización

-Gráficas científicas e ingenieriles

-Desarrollo de aplicaciones, como GUI (graphical user interface)

MATLAB es un sistema interactivo cuyo elemento básico de datos es un vector que no requiere de dimensionamiento. Esto permite resolver varios tipos de problemas técnicos computacionales, especialmente en los que se trabaja con matrices y vectores, en una fracción de tiempo, a diferencia de los lenguajes C o Fortran, que trabajan con escalares

no interactivos. Dada su aplicación principal es de donde surge el nombre MATLAB: matrix laboratory [12].

Por varias de las características enunciadas correspondientes a este lenguaje de alto nivel, es que no se dudó en escogerlo como herramienta principal del software a desarrollar: elaboración de algoritmos, cálculos matemáticos, gráficos y una interfaz gráfica amigable al usuario. Cabe destacar que el enfoque principal del uso de este lenguaje de programación de alto nivel, es realizar una interfaz gráfica que de forma sencilla nos muestre como transformar la información que obtuvimos con antelación del OTDR (en el dominio del tiempo) al dominio de la frecuencia, mediante algoritmos para la FFT (transformada rápida de Fourier) y finalmente poder ver la respuesta en frecuencia de la fibra que se haya medido. A continuación se explicará un poco el funcionamiento de este programa y cómo se desarrolló interfaz gráfica de esta tesis.

5.4.1 El código principal en MATLAB

Para entender mejor el funcionamiento de este lenguaje, un ejemplo viene muy bien. A continuación se explicará el código dividido en partes (figuras) con una descripción debajo de cada figura (para ver el código completo sin fragmentar revisar APÉNDICE B). Este código es un archivo de tipo *.m* de MATLAB, es decir que se ejecuta al escribir su nombre en la ventana de comandos, ya que hace referencia a si mismo.

```
function p3(action);

if nargin<1,
action='start';
end;

global P3_DAT
global entrada
global salida

if strcmp(action,'start'),
```

Figura 5.3 Función de inicio.

Es una forma muy común de comenzar un programa. Comprueba que se ha llamado a *p3* y se le nombra '*start*' a la acción, lo demás son variables globales que se declaran para usarse en cualquier parte de *p3*.

```
oldFigNumber = watchon;
figNumber = figure;
set(gcf, ...
'NumberTitle','off', ...
'Name','Respuesta en frecuencia de una fibra mediante FFT',...
'backingstore','off',...
'Units','normalized');
```

Figura 5.4 Inicialización de la Interfaz Gráfica.

Estas instrucciones llaman a la interfaz (figura) actual, es decir si existe una, sino MATLAB la crea

```
arriba=0.95;
abajo=0.05;
izq = 0.78;
anchoEtiq=0.18;
alturaEtiq=0.05;
anchoBoton = 0.18;
alturaBoton=0.07;
offset=0.003;
espacio=0.05;
```

Figura 5.5 Medidas.

Para hacer más fácil el diseño de la interfaz gráfica, es posible construirla dentro del GUIDE, que es un programa de MATLAB especial para crear interfaces, es decir, donde se pueden agregar botones, cuadros, ejes, cajas de texto, barras, etc. Muy parecido a la forma de trabajar del lenguaje de programación Visual Basic.Tal vez no parece muy práctico realizar una interfaz de forma programada, en si no es difícil, lo único que hay que tener cuidado, es en las medidas de forma que quede lo más simétrica posible, para eso se definen arriba en la Figura 5.5. El resto son simplemente instrucciones recurrentes ya sea de un botón, o una caja de texto, o una etiqueta.

```
margen=0.02;
posY=0.05-margen;
posConsola=[izq-margen posY anchoBoton+2*margen 0.9+2*margen];
h=uicontrol( ...
   'Style','frame', ...
   'Units','normalized', ...
   'Position',posConsola);
```

Figura 5.6 Consola.

Por ejemplo, aquí (Figura 5.6) acabamos de construir una consola (frame), sin importar las medidas y posición.

```
numBoton=1;
posYEtiq=arriba-(numBoton-1)*(alturaBoton+alturaEtiq+espacio);
posEtiq=[izq posYEtiq-alturaEtiq anchoEtiq alturaEtiq];
uicontrol( ...
   'Style','text', ...
   'Units','normalized', ...
   'Position',posEtiq, ...
   'HorizontalAlignment','left', ...
   'String',' Importar Archivo');

posBoton=[izq posYEtiq-alturaEtiq-alturaBoton-offset anchoBoton alturaBoton];
popup=uicontrol('Style','Popup','String','Pulso de entrada|Pulso de salida',...
'Units','normalized',...
      'Position', posBoton, ...
      'CallBack','p3("leeArchivo")');
```

Figura 5.7 Menú Pop-up.

En la Figura 5.7 vemos dos párrafos, el primero hace una caja de texto no editable (un título o etiqueta) y el segundo crea un *popup menu,* esto es un menú que contiene 2 o más opciones, donde solamente aparece una en el recuadro y al presionar la flecha aparecen todas las opciones, como un menú de opciones de cualquier programa en Windows. Lo importante en este menú, es lo que queramos que realice al ser elegido. Aquí es donde viene la aplicación de los '*Callbacks*' en MATLAB. Un *Callback* es quien se encarga de llamar a cierta rutina bajo una acción determinada, como el presionar un botón, o elegir una opción de un menú, o el deslizar la barra movible, o hacer clic sobre una figura, etc. Es decir recurre a la acción que responde bajo el mismo nombre, como aquí la acción '*leeArchivo*'.

```
numBoton=3;
posYEtiq=arriba-(numBoton-1)*(alturaBoton+alturaEtiq+espacio);
posEtiq=[izq posYEtiq-alturaEtiq anchoEtiq alturaEtiq+margen];
texto_freq = uicontrol( ...
   'Style','text', ...
   'Units','normalized', ...
   'Position', posEtiq, ...
   'String','Ancho de banda en Hz');

posBoton=[izq+margen/2  posYEtiq-alturaEtiq-alturaBoton-offset ...
      0.8*anchoBoton+margen  alturaBoton];
campo_freq = uicontrol( ...
   'Style','edit', ...
   'Units','normalized', ...
   'Position', posBoton, ...
   'BackgroundColor','w',...
   'String','',...
   'CallBack','');
```

Figura 5.8 Caja de texto editable.

Igualmente, en estos dos párrafos de la Figura 5.8, el primero es un texto o etiqueta y el segundo, es una caja de texto editable (‘*edit*’), es decir que se puede modificar ya sea externa o internamente. Externamente el usuario modifica o ingresa algún valor que desee. Internamente, es el programa el que modifica dicho valor dependiendo de la aplicación. En este caso, no hay *Callback* es decir, que el uso de la caja de texto es exclusivo de algún resultado o dato modificable del programa.

```
boton_cerrar=uicontrol('Style','Pushbutton', ...
    'Units','normalized',...
    'Position',[izq abajo anchoBoton 2*alturaEtiq], ...
    'Callback','p3("acabar")','String','Cerrar');

boton_BW= uicontrol( ...
    'Style','Pushbutton', ...
    'Units','normalized', ...
    'Position',[izq-2*offset/2abajo+(4*alturaEtiq)+(2*espacio) anchoBoton+3*offset 2*alturaEtiq],...
    'String','Resp. en Frecuencia', ...
    'Callback','p3("bw")');
```

Figura 5.9 Creación de botones.

En la Figura 5.9 tenemos dos párrafos, los dos son botones, el primero es un botón que sirve para cerrar el programa (quitar la aplicación) y el segundo para que realice, una vez teniendo la información necesaria, la respectiva rutina que nos entregue la gráfica y dato de la respuesta en frecuencia. El botón es muy sencillo en MATLAB, como se puede ver en el primer párrafo, fuera de las medidas, lo importante es el *Callback*, la llamada a una rutina, ‘*acabar*’ en el caso del primer párrafo y ‘*bw*’ en el segundo.

```
val = get(popup,'Value');
[f,N]=carga_dato(val);
```

Figura 5.10 Llenado de *f* llamando a función *carga_dato*.

Lo primero que hará el programa (Figura 5.10) es ver cual de las dos opciones del menú se eligió: pulso de entrada o pulso de salida. Si acaba de comenzar, no preguntará por alguna opción, sino que inmediatamente asumirá que es el pulso de entrada el que se requiere y aparecerá el archivo done se encuentran la señales de los pulsos. Inmediatamente después de saber qué pulso es, el programa llamará a la función

carga_dato. Abajo, en la Figura 5.11, se pueden ver las instrucciones para poder importar una tabla de datos que contenga la gráfica de algún pulso, esta tabla es un vector que será almacenado en f , donde N es la longitud de ese vector (número de muestras):

a)

```
function [f,N] = carga_dato(val);

if (val == 1),      %pulso de entrada
        [filename,pathname]=uigetfile('*.txt','Busque la Señal Pulso de Entrada');
    if ~ischar(filename),return,end

    %lee archivo (usa la segunda columna solamente)
    longfilename=strcat(pathname,filename);
    s=load(longfilename);
    if size(s,2)>1
       fn=s(:,end);
       N=length(fn);
       f=10.^(fn/20);
    end
```

b)

```
elseif (val == 2),   % pulso de salida
   [filename,pathname]=uigetfile('*.txt','Busque la Señal Pulso de Salida');
   if ~ischar(filename),return,end

   %lee archivo (usa la segunda columna solamente)
   longfilename=strcat(pathname,filename);
   s=load(longfilename);
   if size(s,2)>1
      fn=s(:,end);
      N=length(fn);
      f=10.^(fn/20);
   end
end;
```

Figura 5.11 Función para importar pulso de a) entrada y b) salida.

El pulso que capturamos del OTDR está en decibeles por lo tanto que hay que desnormalizarlo. Este procedimiento está en las rutinas que se acaban de mostrar. Se aprecia que son dos rutinas iguales, lo único que cambia es el título del recuadro que nos aparecerá preguntando por “la señal deseada” (entrada o salida).

```
M=2048;
t0=0;
```

```
t1=10e-9;
t=linspace(t0,t1,N)';
T= (t1-t0)/N;
```

Figura 5.12 Configuración de variables no globales.

En la Figura 5.12 vemos variables que representan: *M* número de muestras, mientras mayor sea este número y potencia de 2, más resolución tendremos en los cálculos de la DFT; sin embargo, más lento es el cálculo. *T* es el periodo de muestreo, este valor varía dependiendo del ancho de pulso que se esté usando y el número de muestras *N* , en este caso tomamos 10ns.

```
if (val==1),
ejes_freq=axes('Position',[.12 .14 .6 .3],'XLim',...
      [0 1/(2*T)],'YLim',[-5 10]);
axis([0 1/(2*T)  -5 10]);
[FF1,w]=haz_fft(f,M);
inf_freq=plot(w/(2*pi*T),(FF1),'EraseMode','xor');

elseif (val==2),
ejes_freq=axes('Position',[.12 .14 .6 .3],'XLim',...
      [0 1/(2*T)],'YLim',[-5 10]);
axis([0 1/(2*T)  -5 10]);
[FF2,w]=haz_fft2(f,M);
inf_freq=plot(w/(2*pi*T),(FF2),'EraseMode','xor');
salida=[FF2(:)];

end

grid on;
ylabel('Magnitud');
xlabel('Frecuencia (Hertz)');
title('Dominio de la Frecuencia');
```

Figura 5.13 Elección para graficar pulso de entrada o salida en el dominio de la frecuencia.

Estas dos rutinas en la Figura 5.13 hacen lo mismo, pero una para el pulso de entrada y la otra para el pulso de salida, dependiendo de nuestra elección. Lo que harán es llamar a la función *haz_fft* y *haz_fft2* respectivamente. Se requiere hacer dos diferentes cálculos, ya que cada señal tiene diferente número de muestras *N*. Sin embargo *M* será

igual siempre en las dos. El resto de las instrucciones son para crear un eje de coordenadas y finalmente la función *plot*, que se encarga de graficar la DFT (la transformada en el dominio de la frecuencia de *f*) en dicho eje.

```
ejes_tiempo=axes('Position',[.12 .58 .6 .3],'XLim',[t0 t1],'YLim',[10 240]);
inf_tiempo=plot(t,f,'EraseMode','xor');
axis([t0 t1 10 240]);
grid on;
ylabel('Magnitud');
xlabel('Tiempo (Segundos)');
title('Dominio del tiempo');
```

Figura 5.14 Instrucciones para graficar en el dominio del tiempo.

Arriba, en la Figura 5.14, vemos como se grafica el vector *f*, es decir el pulso en el dominio del tiempo.

```
elseif strcmp(action,'bw'),
   %realiza los calculos para la respuesta en frecuencia
   %al presionar el boton 'BW'

   FF1=entrada;
   FF2=salida;
   campo_freq=P3_DAT(3)
   BW=(FF2./FF1);
   BWdB=20*log10(BW);
   BWdB=BWdB(1:55);
   maxBW=max(BWdB)
   BWdB3=maxBW-3
   c=find(BWdB3 -0.3 < BWdB & BWdB < BWdB3 +0.3)
   c=mean(c)
   freq=70e9*(0:c)/55;
   frecuencia= freq(c)
   set(campo_freq,'String',num2str(frecuencia));
   plot(freq,(BWdB(1:c+1)),'EraseMode','xor');
   Grid on;
   title('Espectro de frecuencias del canal');
   ylabel('Magnitud en dB');
   xlabel('Respuesta en frecuencia (Herz)');
```

Figura 5.15 Acción que realiza el cálculo del ancho de banda en -3dB.

Esta es la rutina de otra acción (Figura 5.15), que es diferente de una función, la acción depende de que suceda un evento que la invoque, como en este caso el presionar el botón *Resp. en Frecuencia*. La función es invocada desde una acción o desde cualquier parte del programa. La acción que se muestra arriba, es la que se encarga de realizar el procedimiento para calcular la respuesta en frecuencia a partir de *H(f)=Y(f)/X(f)*, donde *Y(f)* es la transformada DFT del pulso de salida (*FF2* en el programa) y *X(f)* es la

transformada DFT del pulso de entrada (*FF1* en el programa). A partir de ahí, solo necesitamos encontrar la frecuencia de corte (-3dB) y obtener el valor correspondiente. La frecuencia de muestreo *freq* que se usa para esta gráfica está basada en la frecuencia de muestreo del obturador del osciloscopio del OTDR, aproximadamente 70 GHz [5].

```
function [FF1,w] = haz_fft(f,M)

   N=length(f);
   F=fft(boxcar(N).*f,2*M);
   F=F(1:M);
   w=(0:M-1)*pi/M;
   indxs = find(F==0);
   F(indxs) = eps;       % Evita dividir entre cero
   FF1=abs(F)/M;
```

Figura 5.16 Función para calcular la FFT de un pulso *f*.

En la Figura 5.16 está la función comentada antes *haz_fft,* para el pulso de entrada, que en realidad es igual a la función para el pulso de salida, simplemente cambian los nombres de los vectores resultantes que contengan la respectiva DFT. Los vectores (*f,M*) son traídos a la función, para obtener [*FF1,w*] los vectores correspondientes al dominio de la frecuencia. *FF1* es la DFT de *f*, y *w*, es el vector ω, es decir quien contiene las frecuencias discretas de 0 a π (no se requiere 2π, porque se aplica la propiedad de simetría). El resultado del algoritmo FFT que use MATLAB para realizar la DFT, son dos vectores, uno con números reales y el otro con números imaginarios, por eso es necesario obtener la magnitud con el comando *abs(F)*. Los demás comandos se encargan de realizar la DFT, y buscar que no haya ceros en el vector para evitar un error por división entre cero.

5.4.2 Los algoritmos FFT de MATLAB

En el inciso anterior mostramos la forma de proceder de MATLAB a través de un ejemplo: el código para calcular la respuesta en frecuencia, sin embargo, dicho ejemplo no nos dice cómo es que MATLAB realiza la DFT. En el capítulo 3, vimos algunos

algoritmos FFT. Principalmente los algoritmos *Cooley-Tukey* y *base 2*, a partir de estos algoritmos surgen los demás que MATLAB usa [12].

El algoritmo *base 4*, es muy similar al *base 2*, simplemente se basa en el hecho de que el número de muestras *N* sea una potencia de 4 [3]. Las reducción de tiempo al usar un algoritmo *base 4* contra uno *base 2*, es del 25% [3] en realidad no es tanto como podría esperarse. Por esta misma razón se experimentó con más bases, como *base 8*, por ejemplo, y se llegó a la conclusión que a partir de una *base 4*, no hay mejora considerable en la simplificación del cálculo de la DFT [3]. En seguida surge otro algoritmo que se llama SRFFT (Split Radix Fast Fourier Transform) o algoritmo FFT de base partida. Este algoritmo combina *base 4* y la *base 2*, de esta forma agiliza mucho más el proceso [7]. MATLAB tiene una librería llamada *fftw* que se encarga de descomponer *N* con el algoritmo Cooley-Tukey: calculando *N1* transformadas de tamaño *N2* y después *N2* transformadas de tamaño *N1*. Este procedimiento es recursivo hasta que tengan un tamaño adecuado para ser resueltos por los "*codelets*" (mini-códigos). Estos "*codelets*" pueden usar a su vez diferentes combinaciones de algoritmos incluyendo una variación del Cooley-Tukey, un algoritmo de factor primo, y el SRFFT (Algoritmo de Base Partida). Como ya sabemos mientras menos valores tenga *N* , y *N* no sea número primo y sí potencia de 2, el tiempo de ejecución para la FFT se reducirá notablemente [12].

5.5 Pruebas y Resultados

Una vez analizados los comandos más importantes de nuestro código en MATLAB, no queda más que ver como funciona la interfaz gráfica. La siguiente gráfica (Figura 5.17) muestra el inicio del programa, nos preguntará inmediatamente que busquemos la señal

del pulso de entrada que queramos y aparecerá un fólder con muchos archivos. Ya que hayamos escogido la señala con la que queramos trabajar para analizar la fibra, se le da clic al botón “abrir”.

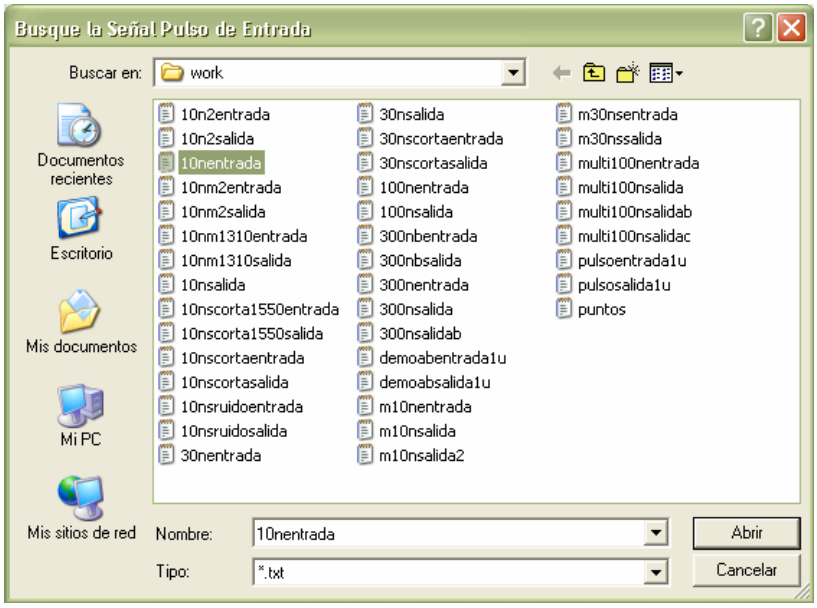

Figura 5.17 Ventana para seleccionar el pulso de entrada deseado.

En seguida nos aparecerá la interfaz con el pulso graficado en el dominio del tiempo arriba y en el dominio de la frecuencia abajo (Figura 5.18), el paso siguiente será selección del menú “*importar archivo*”, la señal del pulso de salida.

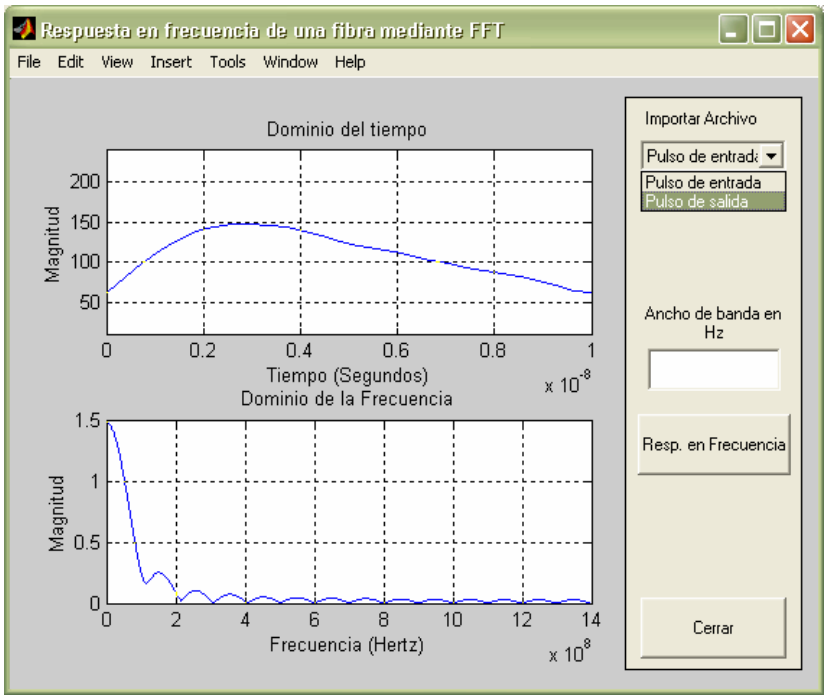

Figura 5.18 Interfaz gráfica con el pulso de entrada.

Nos volverá a aparecer la ventana con los archivos conteniendo los pulsos, pero esta vez habrá que seleccionar el correspondiente pulso de salida de la misma medición. Como se muestra en la Figura 5.19.

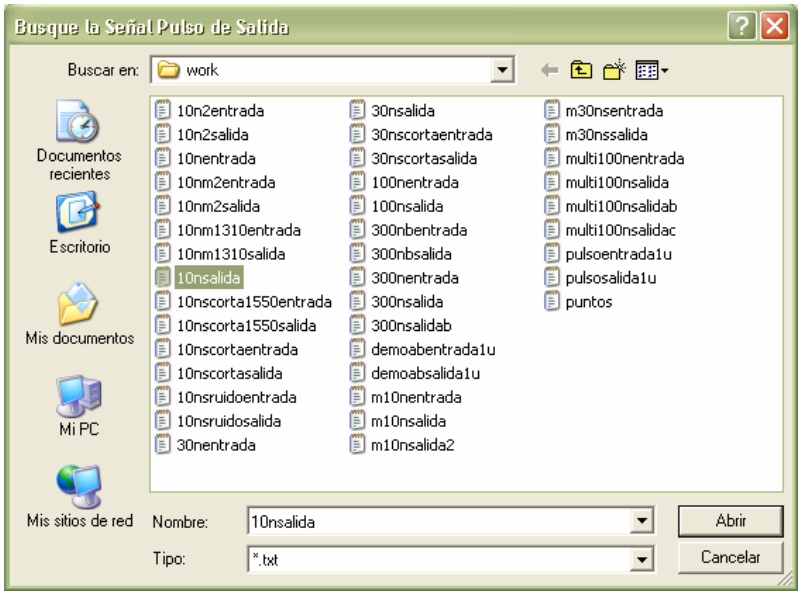

Figura 5.19 Ventana para seleccionar el pulso de salida deseado.

Volveremos a ver dos gráficas en la interfaz (Figura 5.20), una que muestra el pulso de salida en el dominio del tiempo y debajo, la gráfica que muestra el pulso de salida en el dominio de la frecuencia. En seguida, lo que queremos hacer es obtener a partir de estas dos señales, la respuesta en frecuencia de la fibra óptica, de la cual obtuvimos tales muestras.

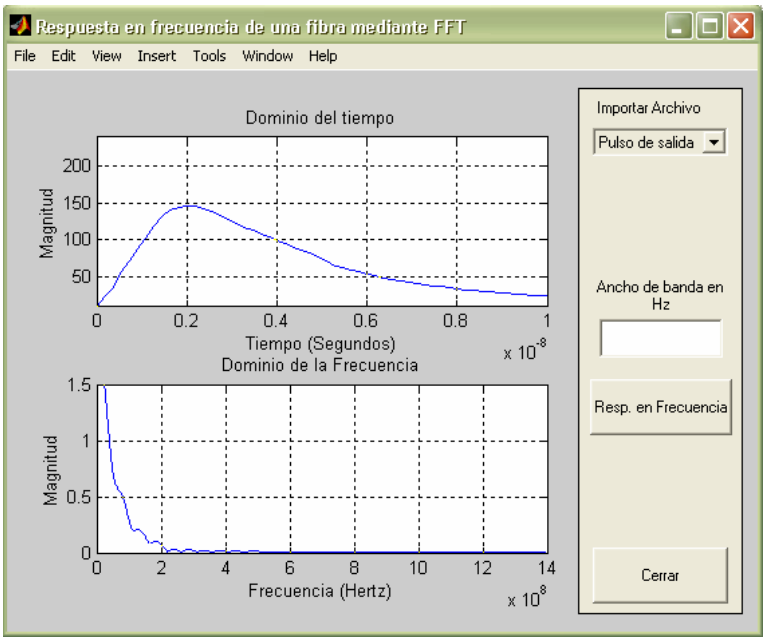

Figura 5.20 Interfaz gráfica con el pulso de salida.

La siguiente figura (Figura 5.21) muestra el resultado final de este experimento, la respuesta en frecuencia de la fibra que estamos tratando. La gráfica que nos interesa es la que se encuentra en la parte superior y el dato que aparece en el recuadro en blanco, es el ancho de banda estimado en la frecuencia de corte de la fibra.

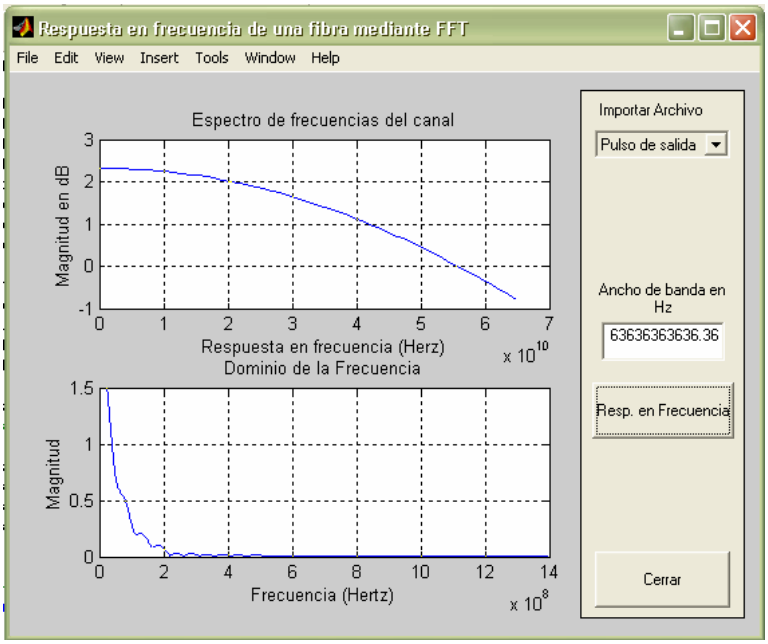

Figura 5.21 Interfaz gráfica con la respuesta en frecuencia de la fibra (recuadro superior).

Es necesario comentar que los valores y datos mostrados en este desarrollo corresponden a al mismo procedimiento (una sola medición). Desde la programación del OTDR, se tomaron valores reales y correspondientes a una fibra mono-modo de 1 km., con un ancho de pulso de 10ns. y con una longitud de onda de 1310 (la distancia es muy corta como para que afecte el hecho de usar 1310nm o 1550nm).

Las mediciones que se hicieron fueron hechas en el laboratorio de óptica de la UDLA, contando con 3 carretes de fibra: uno multi-modo de 100m, los otros dos de 1km y mono-modo. El riesgo de hacer mediciones tan cortas (especificado en el manual del OTDR) con el OTDR es que hay probabilidad de que los pulsos entrantes y salientes, como regresan al origen, se empalmen y nos trunquen la información. Sin embargo, al usar 100 m de fibra, al igual que con 1 km, se usó el ancho de pulso más corto 10 ns, esto nos dio una gran resolución en la traza y no tanto ruido como se esperaba, ya que las pruebas se realizaron en el modo promedio del OTDR. En sí la información que nos interesa es el pulso de entrada y el de salida, la atenuación que hay entre estos, al análisis de Fourier no le importa mucho. La captura de estos pulsos se logró de manera adecuada, de forma que los cálculos que realiza MATLAB son sobre datos precisos, lo

que es muy importante indicar es que la precisión con respecto al valor que se marca como “Ancho de Banda en Hz” en la interfaz gráfica, no es un dato exacto, ya que no se pudo averiguar cual es la velocidad real de muestreo del obturador del osciloscopio del OTDR que usamos, sino que tomamos un valor general de 15ps. [5]. Con este valor final, se obtiene el ancho de banda máximo a partir de una traza cualquiera del OTDR. Cada una de las fibras mostró un valor diferente de ancho de banda. Sin embargo los valores son muy parecidos entre las fibras mono-modo de 1 km: 63GHz y 54GHz. Cabe destacar que la segunda fibra mostró siempre mucho ruido, lo cual hace lógica tal diferencia de valores. La fibra multi-modo por su parte, nos indicó un ancho de banda de 43GHz, esto se debe a la longitud de la fibra, una vez más reitero que mientras más larga sea, más fiel es la medición Sin embargo, al ser tan corta el ancho de banda debe de ser más elevado. Por ejemplo, el OTDR tiene una traza de muestra sobre una fibra de 40km, con 1 μs como ancho de pulso y el ancho de banda que nos da MATLAB sorprendentemente es de 63 GHz. Lo que comprueba la veracidad del programa y por ende, de esta tesis.