Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Proyecto Terminal en Ingeniería en Computación

Simulación de la solución al problema directo y el problema inverso de reflectrometría acústica

Proyecto que presenta:
Ana Rosa Espinosa Sánchez

para obtener el título de:
Ingeniero en Computación

Asesores de Proyecto:
M. en C. Oscar Alvarado Nava
Dr. Ernesto Rodrigo Vázquez Cerón

México, D.F. Enero de 2013

RESUMEN

La técnica de reflectometría acústica consiste en la estimación del área transversal de una cavidad cilíndrica en función de la distancia mediante el adecuado procesamiento digital y análisis de una onda acústica reflejada respecto a una onda acústica incidente, este proceso es llamado problema inverso, así mismo, el problema directo consiste en calcular la onda acústica reflejada por medio de una cavidad cilíndrica y una onda acústica incidente. En este documento se muestra el diseño y la implementación de los diferentes algoritmos necesarios para la solución del problema directo usando como herramienta de desarrollo MATLAB 7.0.12.

AGRADECIMIENTOS

- A COMIPEMS por la beca para la realización de este proyecto.

- A mis padres, quienes han sido mi más grande fuente de inspiración, los admiro y amo.
- A mis amigos y asesores Oscar Alvarado Nava y Ernesto Rodrigo Vázquez Cerón, por su paciencia, sus palabras y el apoyo que me han brindado.
- A mi primo Alberto Rodríguez Sánchez, quien ha sido una parte de suma importancia en esta etapa de mi vida.
- A todos mis amigos y familiares por su apoyo incondicional.

ÍNDICE GENERAL

Existen diversos algoritmos diseñados para calcular una cavidad cilíndrica en función de la distancia, sin embargo, uno de los más reconocidos es el algoritmo Ware-Aki, debido a que ha sido el más utilizado en aplicaciones biomédicas para el estudio de la reconstrucción de cavidades humanas con exitosos resultados [7].

El algoritmo Ware – Aki es un proceso recursivo por medio del cual se pueden calcular los coeficientes de reflexión de una cavidad cilíndrica en cada cambio de impedancia, donde se genera una onda acústica reflejada y una onda acústica trasmitida; este cambio de impedancia depende, a su vez de un cambio en el área transversal del segmento S0 al área transversal en el segmento S1.

Como se puede observar en la Fig. 2, la onda entrante que viaja en el segmento S0 tiene una presión acústica de P0+, en el punto x=0 se divide en una onda transmitida con una presión acústica de P1+ en el segmento S1 y una onda reflejada que viaja en el segmento S0.

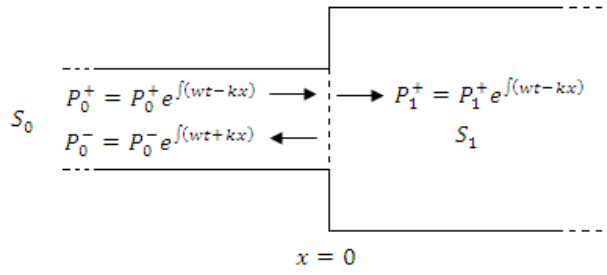

Fig. 2. *Variación del área transversal dentro de la cavidad cilíndrica.*

Para una cavidad cilíndrica, las características de impedancia acústica (Z) se definen como la razón del producto de la densidad del medio por donde se propaga la onda (ρ) y la velocidad del sonido (c) entre el área transversal del segmento S como se muestra en la siguiente expresión:

$$Z = \frac{c\rho}{S} \qquad (1)$$

Considerando la *ec.1*, se tiene una impedancia acústica para cada segmento de:

$$Z_0 = \frac{c\rho}{S_0}\ , \quad Z_1 = \frac{c\rho}{S_1}\ , \ldots, \quad Z_n = \frac{c\rho}{S_n} \qquad (2)$$

Los coeficientes de trasmisión se representan con la siguiente expresión:

$$r_{01} = \frac{P_0^-}{P_0^+} = \frac{Z_1 - Z_0}{Z_1 + Z_0} \qquad (3)$$

Entonces, para una onda acústica que se propaga dentro de una cavidad cilíndrica y atraviesa del segmento S_0 al segmento S_1 se puede deducir la expresión:

$$r_{01} = \frac{S_0 - S_1}{S_0 + S_1} \qquad (4)$$

Es posible aplicar esta teoría y considerar una cavidad cilíndrica compuesta por varios segmentos S_n, considerando para cada uno de estos segmentos diferentes áreas transversales (Fig. 3).

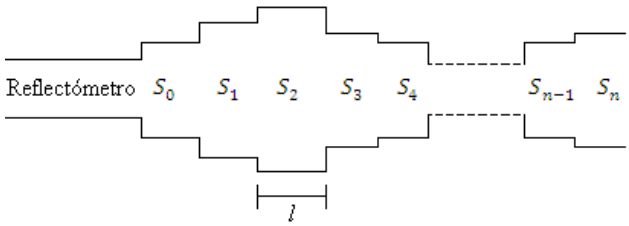

Fig. 3. *La resolución de una cavidad cilíndrica es representada mediante segmentos cilíndricos.*

Considerando un impulso acústico como onda incidente, con la siguiente forma:

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \qquad (5)$$

La onda reflejada será la respuesta impulso *iir[nT]*.

De aquí que la longitud de cada segmento está dada por:

$$l = \frac{cT}{2} \qquad 6$$

Donde:

l = Longitud del segmento.

c= Velocidad del sonido en el aire.

T=Es el tiempo de separación entre cada iir.

También se tiene que:

$$T = \frac{1}{F} \qquad (7)$$

Para la interfaz gráfica se usó como ambiente de desarrollo MATLAB 7.0.12, ya que permite un fácil procesamiento de señales y es una herramienta comúnmente utilizada por los principales usuarios a los que se encuentra dirigida la interfaz.

En el laboratorio de sensores y señales de la Universidad Autónoma Metropolitana se realiza el experimento para resolver el problema inverso, el cual se lleva a cabo mediante un reflectómetro acústico (Fig. 11).

***Fig. 11**. Prototipo del reflectómetro acústico.*

La puesta en marcha de este dispositivo consiste en hacer llegar una onda digital, previamente diseñada por software y procesada por un amplificador de potencia, a una bocina a través de una tarjeta de adquisición de datos.

La onda acústica generada se propaga a lo largo de un tubo fuente que sirve como guía de onda. En el otro extremo del tubo fuente se acopla la cavidad de estudio donde es impactada la onda acústica.

Como se describió en la sección de fundamentos teóricos, debido al cambio de impedancia que detecta la onda incidente se genera una onda reflejada, la cual se propaga en dirección opuesta a la onda incidente.

Un micrófono colocado al ras de la pared interna del tubo fuente registra ambas ondas acústicas, las cuales son amplificadas utilizando un amplificador de instrumentación y almacenadas en una computadora usando la misma tarjeta de adquisición de datos.

Para el diseño y construcción del reflectómetro acústico se utilizó un tubo fuente largo. De esta manera se puede identificar la onda incidente de la reflejada.

Sin embargo, debido a que en este trabajo se realizó el análisis de una región en el estado estacionario alrededor del pico de la onda para obtener su amplitud y fase, el largo del tubo fuente permitió generar un tiempo para llevar a cabo el registro para su posterior análisis.

Un diagrama esquemático del reflectómetro acústico se muestra en la siguiente Fig. 12.

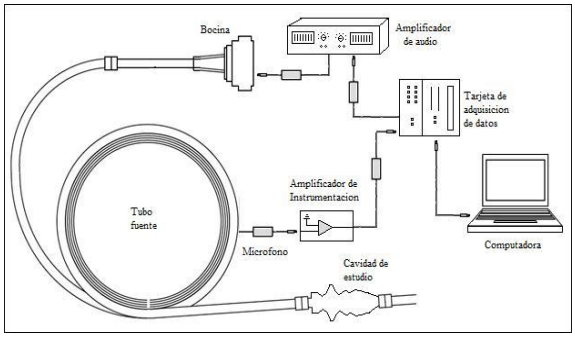

Fig. 12. *Diagrama esquemático*

Para llevar a cabo la implementación de ésta propuesta en ingeniería, se usó un modelo in vitro de una vía aérea humana [9].

Las dimensiones de esta cavidad de estudio representan la forma del área de un sujeto normal de 40 años, ver Fig. 13.

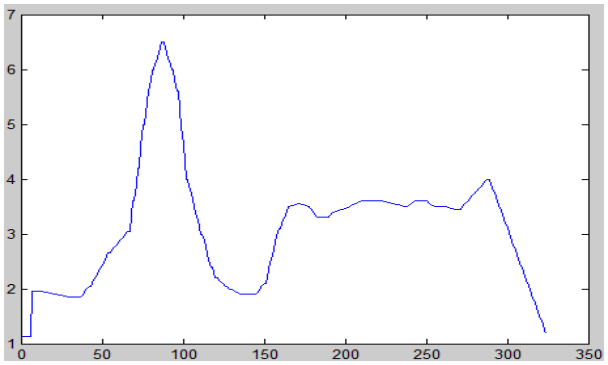

Fig. 13. *Área de cavidad cilíndrica.*

4.2. COMPARACIÓN DE RESULTADOS

4.3 CONCLUSIONES

CAPÍTULO 5

5.1. MANUAL DE USUARIO

5.1.1 PROBLEMA DIRECTO

5.1.2 PROBLEMA INVERSO

CAPÍTULO 6

CÓDIGO FUENTE

6.1 Inicio.m

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1.1. INTRODUCCIÓN

La técnica de reflectometría acústica consiste en la estimación del área transversal de una cavidad cilíndrica en función de la distancia mediante el adecuado procesamiento digital y análisis de una onda acústica reflejada respecto a una onda acústica incidente.

La técnica de reflectometría acústica surgió como un requerimiento en ingeniería para evaluar la resolución axial de cavidades cilíndricas, sin la necesidad de introducir un dispositivo electrónico (endoscopio) o recurrir a técnicas no invasivas de radiación (rayos X) para estimar el diámetro en función de la distancia.

La técnica de reflectometría acústica presenta ciertas ventajas sobre las existentes, ya que además de ser relativamente barata, rápida y no invasiva, los resultados obtenidos proporcionan directamente una perspectiva tridimensional de la resolución axial.

El correcto análisis de las señales acústicas permite entonces que la técnica también pueda ser empleada para determinar fugas y/o obstrucciones en ductos y tuberías [1]. Se ha utilizado también en la caracterización de instrumentos musicales de viento mediante el análisis de impedancia acústica[2][3]. Otra de las aplicaciones interesantes consiste en evaluar el área transversal de cavidades humanas como puede ser la vía aérea superior [4][5][6]

A pesar de que la técnica ha sido llevada a cabo, en todos los casos anteriormente mencionados con buenos resultados, es necesario analizar el alcance para mejorar la resolución axial obtenida en una cavidad y evitar los complicados inconvenientes que se presentan al utilizar un pulso acústico.

Para poder llevar a cabo un análisis y evaluar una serie de señales con distinto contenido de frecuencia para diferentes resoluciones axiales, es conveniente implementar una simulación apegándose a la teoría matemática y a las condiciones experimentales tanto como sea posible.

La simulación de reflectometría acústica se conforma a través de la síntesis y análisis tanto del problema directo y el problema inverso, respectivamente.

El problema inverso consiste en estimar la resolución axial de la cavidad cilíndrica por medio de la onda incidente y la onda reflejada. El problema directo, que es el objeto de estudio de este trabajo, consiste en calcular la respuesta al impulso de una cavidad por medio del algoritmo Ware-Aki [7] y, una vez obtenida, se realiza la convolución con la onda incidente para obtener la onda reflejada.
Esta técnica presenta ciertas ventajas sobre las existentes, ya que además de ser relativamente barata, rápida y no invasiva, los resultados obtenidos proporcionan directamente una perspectiva tridimensional de la resolución axial, entregando resultados en condiciones ideales.

1.2. OBJETIVOS

- Con el propósito de obtener la respuesta generada por un pulso acústico en una cavidad simétrica, implementar el algoritmo de Ware-Aki.

- Con el propósito de obtener la onda reflejada por medio de la respuesta generada por un pulso acústico y una onda incidente, implementar el algoritmo de convolución.
- Con la finalidad de que el usuario seleccione fácilmente una cavidad simétrica, se pretende implementar un catálogo que contenga por lo menos seis cavidades diferentes en función de la distancia.
- Con el objeto de permitir al usuario definir las características de una onda incidente y la frecuencia de muestreo, diseñar e implementar una interfaz gráfica.
- Con el propósito de generar en el usuario un mayor entendimiento del problema directo en reflectometría acústica, se pretende diseñar e implementar un módulo que permita representar gráficamente en un diagrama espacio-tiempo la propagación de las ondas acústicas a lo largo de una cavidad simétrica, además de construir una tabla con los datos obtenidos por el algoritmo de convolución.

1.3. ANTECEDENTES

Continuous and discrete inverse scattering problems in a stratified elastic medium.I: Planes at normal incidence

La solución del problema inverso en sismología consistió con la determinación de los coeficientes de reflexión, a partir de la respuesta impulso generada por la propagación de un pulso acústico (Explosiones de dinamita) a través de las capas de la tierra [8]. El objetivo en aquel entonces fue analizar los coeficientes de reflexión para identificar yacimientos de petróleo y/o agua.

Reflectómetro Acústico

La técnica de reflectometría acústica posteriormente fue adecuada para estimar la resolución axial en cavidades cilíndricas [9]. De esta manera se diseñó y construyó un reflectómetro acústico para generar ondas incidentes con bocinas y/o descargas eléctricas, ver figura 1.

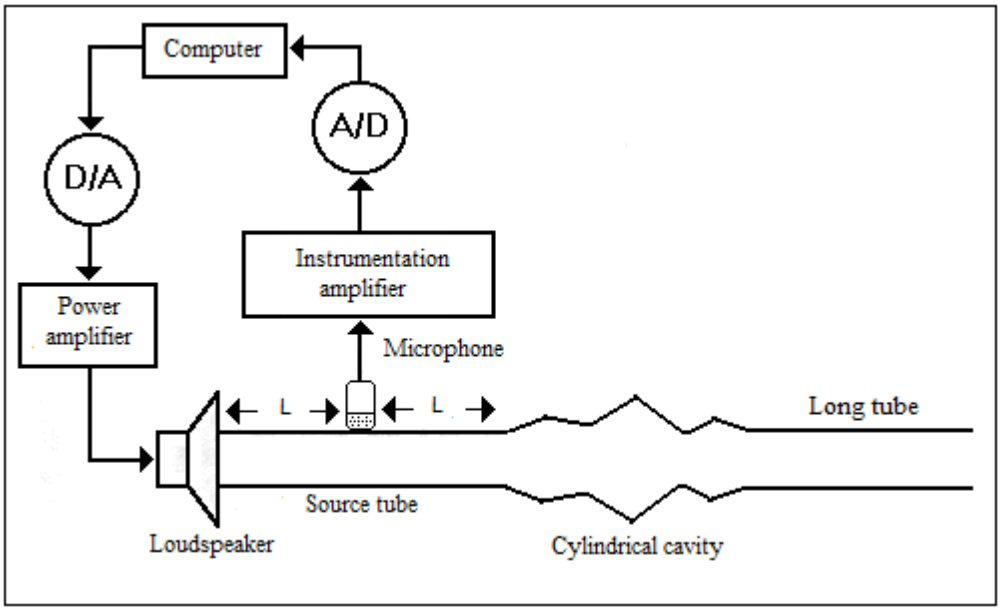

Fig. 1 *Diagrama esquemático de un reflectómetro acústico.*

El funcionamiento de un reflectómetro acústico consiste en generar ondas incidentes que son conducidas a lo largo de un tubo fuente, el cual sirve como guía de onda. De esta manera, al encontrar una variación en la impedancia, la cual se encuentra asociada con un cambio en el área, se genera una onda reflejada, la cual se propaga en dirección opuesta a la onda incidente. Un micrófono colocado al ras de la pared interna realiza el registro de ambas ondas las cuales son almacenadas en una computadora para su posterior análisis.

1.4. CONCEPTOS CLAVE

Reflectometría. En su definición más amplia, la reflectometría en el dominio del tiempo es una técnica de detección remota en la cual las características de la reflexión de una señal

conocida dentro de un medio conocido se pueden utilizar para determinar de manera exacta la localización espacial y la naturaleza de las discontinuidades en el medio[10].

Cavidad cilíndrica. Hueco cilíndrico que se abre dentro de un cuerpo o en una superficie.

Onda incidente. Onda que se propaga hacia la superficie de separación de dos medios o hacia una discontinuidad en una línea de transmisión.

Onda reflejada. Onda que aparece cuando una onda encuentra una superficie de separación de dos medios diferentes, que se aleja de la superficie en el mismo medio que la onda incidente y que es interpretable por la óptica geométrica.

Graficación. La graficación es el arte o la ciencia de producir imágenes gráficas con la ayuda de la computadora, su principal objetivo es establecer los principios, técnicas y algoritmos para la generación y manipulación de imágenes mediante una computadora.

CAPÍTULO 2

2.1. FUNDAMENTO TEÓRICO

Existen diversos algoritmos diseñados para calcular una cavidad cilíndrica en función de la distancia, sin embargo, uno de los más reconocidos es el algoritmo Ware-Aki, debido a que ha sido el más utilizado en aplicaciones biomédicas para el estudio de la reconstrucción de cavidades humanas con exitosos resultados [7].

El algoritmo Ware – Aki es un proceso recursivo por medio del cual se pueden calcular los coeficientes de reflexión de una cavidad cilíndrica en cada cambio de impedancia, donde se genera una onda acústica reflejada y una onda acústica trasmitida; este cambio de impedancia depende, a su vez de un cambio en el área transversal del segmento S0 al área transversal en el segmento S1.

Como se puede observar en la Fig. 2, la onda entrante que viaja en el segmento S0 tiene una presión acústica de P0+, en el punto x=0 se divide en una onda transmitida con una presión acústica de P1+ en el segmento S1 y una onda reflejada que viaja en el segmento S0.

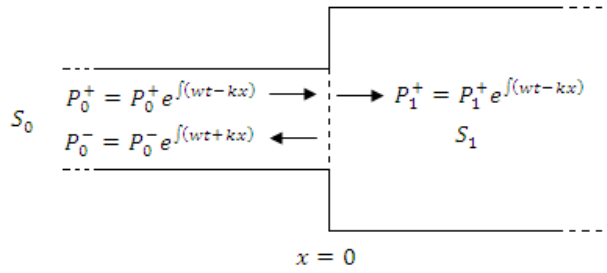

Fig. 2. *Variación del área transversal dentro de la cavidad cilíndrica.*

Para una cavidad cilíndrica, las características de impedancia acústica (Z) se definen como la razón del producto de la densidad del medio por donde se propaga la onda (ρ) y la velocidad del sonido (c) entre el área transversal del segmento S como se muestra en la siguiente expresión:

$$Z = \frac{c\rho}{S} \qquad (1)$$

Considerando la *ec.1*, se tiene una impedancia acústica para cada segmento de:

$$Z_0 = \frac{c\rho}{S_0}\ ,\quad Z_1 = \frac{c\rho}{S_1}\ ,\ldots,\ Z_n = \frac{c\rho}{S_n} \qquad (2)$$

Los coeficientes de trasmisión se representan con la siguiente expresión:

$$r_{01} = \frac{P_0^-}{P_0^+} = \frac{Z_1 - Z_0}{Z_1 + Z_0} \qquad (3)$$

Entonces, para una onda acústica que se propaga dentro de una cavidad cilíndrica y atraviesa del segmento S_0 al segmento S_1 se puede deducir la expresión:

$$r_{01} = \frac{S_0 - S_1}{S_0 + S_1} \qquad (4)$$

Es posible aplicar esta teoría y considerar una cavidad cilíndrica compuesta por varios segmentos S_n, considerando para cada uno de estos segmentos diferentes áreas transversales (Fig. 3).

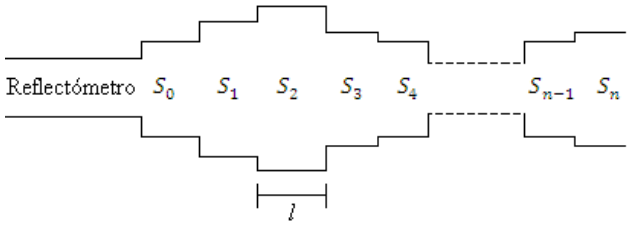

Fig. 3. *La resolución de una cavidad cilíndrica es representada mediante segmentos cilíndricos.*

Considerando un impulso acústico como onda incidente, con la siguiente forma:

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \qquad (5)$$

La onda reflejada será la respuesta impulso *iir[nT]*.

De aquí que la longitud de cada segmento está dada por:

$$l = \frac{cT}{2} \qquad 6$$

Donde:

l = Longitud del segmento.

c= Velocidad del sonido en el aire.

T=Es el tiempo de separación entre cada iir.

También se tiene que:

$$T = \frac{1}{F} \qquad (7)$$

2.2. PROBLEMA DIRECTO

Para la solución del problema directo se cuenta con dos valores iniciales que son: el área transversal de la cavidad en cada uno de sus segmentos (vector que llamaremos *cavidad*$_i$ en

el i-ésimo segmento) y la onda acústica incidente (vector que llamaremos *ondaIncidente*$_j$ en la j-ésima posición).

Los coeficientes de reflexión serán obtenidos por medio de la cavidad de la siguiente manera:

$$r_{i,i+1} = \frac{cavidad_i - cavidad_{i+1}}{cavidad_i + cavidad_{i+1}} \qquad (8)$$

Las dos primeras respuestas al impulso serán calculadas de la siguiente forma:

$$iir_0 = r_{0,1}$$
$$iir_1 = r_{1,2}\ 1 - r_{0,1}^2 \qquad (9)\ .$$

Para los siguientes coeficientes de reflexión haremos uso de la siguiente expresión:

$$iir_n = r_{n,n+1} \prod_{m=0}^{n-1} 1 - r_{m,m+1}^2 - \sum_{m=0}^{n-2} A_m iir_{n-1} \qquad (10)$$

Donde:

$$A_I = A_{I-1} + r_{I,I+1} B_{I-1}$$
$$B_I = r_{I,I+1}\ A_{I-1} + B_{I-1} \qquad (11)$$

Para I>0, como valores iníciales en la realización de estos cálculos se considera que:

$$A_0 = 1$$
$$B_0 = r_{1,1} \qquad (12)$$

El algoritmo finaliza cuando todos los valores del vector cavidad han sido procesados, enviando al algoritmo de convolución la respuesta al impulso (vector iir).

Una vez que se cuenta con la respuesta al impulso es necesario hacer la convolución con la onda incidente para obtener la respuesta del sistema. El cálculo de la convolución, se hará por medio de la función filter [8] con orden uno, que se encuentra en el repertorio de funciones de MATLAB 7.0.12.

2.3. PROBLEMA INVERSO

Para resolver el problema inverso se cuenta con dos valores iniciales, que son la onda acústica que ha sido introducida al sistema, a la cual llamaremos onda incidente, y la onda reflejada.

Como primer instancia se ejecuta el script llamado *fitsine*, por medio del cual se estima la frecuencia, amplitud y la fase de la onda; este proceso es aplicado a las ondas incidentes y a las ondas reflejadas introducidas por el usuario.

Debemos entender que la onda reflejada está compuesta por la convolución de la respuesta al impulso (iir) y la onda incidente, es por esto que el primer paso a realizar será una descomposición de valor singular.

Por medio de la onda incidente se obtiene la matriz triangular superior, aplicando a ésta la función *svd* contenida en el repertorio de matlab[8], que produce los tres factores en la descomposición de valores singulares.

Una vez que se ha terminado la descomposición del valor singular, se obtienen las razones de la amplitud y la fase de la onda reflejada con respecto a la amplitud y la fase de la onda incidente a los cuales llamaremos *A* y *P* respectivamente, en caso de tratarse de un barrido en frecuencia de ondas gaussinas sinusoidales se hará en cada una de las frecuencias.

Se obtiene el valor de la variable iforurier, por medio del script *ifourier*, que recibe como parámetros la frecuencia de muestreo, A, P; en caso de que el usuario introduzca un barrido en frecuencia de ondas gaussianas sinusoidales, también será necesario especificar el tamaño de la división entre cada frecuencia.

Una vez obtenido el valor de ifourier, se manda a llamar a la función *wasp*, quien recibe como parámetros la frecuencia de muestreo y el valor ifourier, obteniendo con ello los coeficientes de reflexión y el área tranversal de la cavidad cilíndrica.

Por último la cavidad cilíndrica es graficada, y sus valores númericos se pueden observar en la simulación.

CAPÍTULO 3

3.1. DESARROLLO DEL SIMULADOR GRÁFICO

Para la interfaz gráfica se usó como ambiente de desarrollo MATLAB 7.0.12, ya que permite un fácil procesamiento de señales y es una herramienta comúnmente utilizada por los principales usuarios a los que se encuentra dirigida la interfaz.

3.1.1. PROBLEMA DIRECTO

Como resultado se cuenta con una interfaz gráfica capaz de representar cualquier tipo de cavidad cilíndrica, para ello se le proporciona como valor de entrada un arreglo unidimensional que representa el área de la cavidad en función de la distancia, ésta puede ser introducida por el usuario a través del mouse seleccionando las coordenadas en un área de dibujo o bien al seleccionar un archivo con los datos prestablecidos directamente. Fig.4.

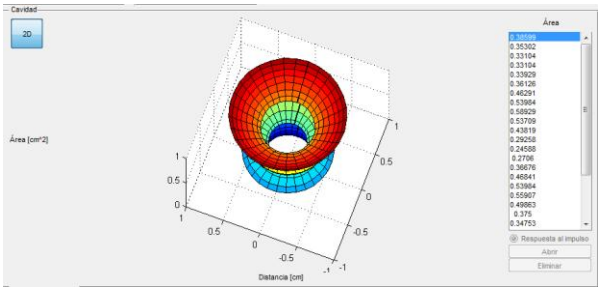

Fig. 4. *La cavidad en 3D rotada. En la parte superior izquierda está el control que define si la cavidad será representada en dos ó tres dimensiones, en la parte derecha se puede apreciar un control listbox por medio del cual se muestran las áreas de cada uno de los segmentos en la cavidad.*

Una vez que se ha definido la cavidad, es posible calcular y generar la gráfica de la respuesta al impulso como se muestra en la *Fig.5*.

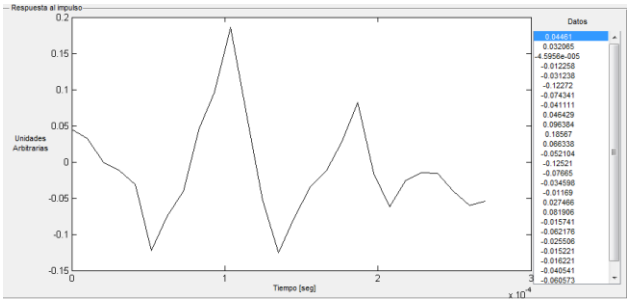

Fig. 5. *Respuesta al impulso generada con la cavidad ya antes definida, en la parte derecha de la imagen se encuentra un listbox que muestra los valores representados en la grafica.*

Onda incidente

Es posible introducir tres diferentes tipos de ondas. El primer tipo de onda es arbitraria, la cual se construye a partir de datos en un rango de 1 a 500 valores, los cuales pueden ser enteros ó flotantes tomando sólo cuatro dígitos fraccionarios de estos valores.

El segundo tipo de onda es un pulso acústico, el cual se genera en base a la longitud del vector cavidad.

El tercer tipo de ondas son las gaussianas sinusoidales (Fig. 6), estas ondas son las más importantes para la investigación, ya que nos permiten designar el rango de frecuencia en el que deseamos conocer la respuesta del sistema, además de ser más fáciles de manipular en la parte experimental. La simulación crea un barrido de señales gaussianas sinusoidales que van desde una frecuencia mínima a una frecuencia máxima, el cual se almacena en una matriz.

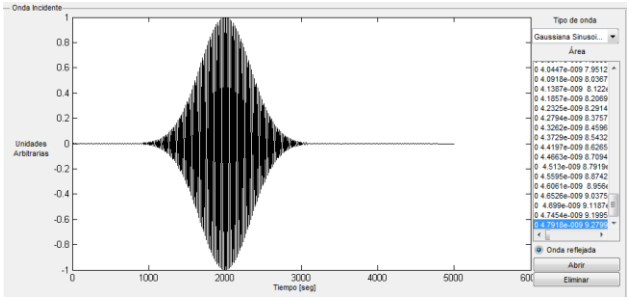

Fig. 6. *Onda incidente Gaussiana Sinusoidal.*

Para definirlas a través de la simulación es necesario introducir la frecuencia mínima de banda ancha y la frecuencia máxima de banda ancha, las cuales deben estar comprendidas entre los 50 Hz y 10,000 Hz; así como también es necesario definir el tamaño de los pasos en la frecuencia, es decir, los intervalos en los que incrementará la frecuencia y por último el número de muestras que se desean obtener. Como respuesta, la herramienta de simulación genera una gráfica de la onda incidente en la parte inferior derecha y muestra los valores de la misma dentro de un objeto de selección gráfico listbox que se encuentra del lado derecho de la gráfica, como se muestra en la Fig. 7.

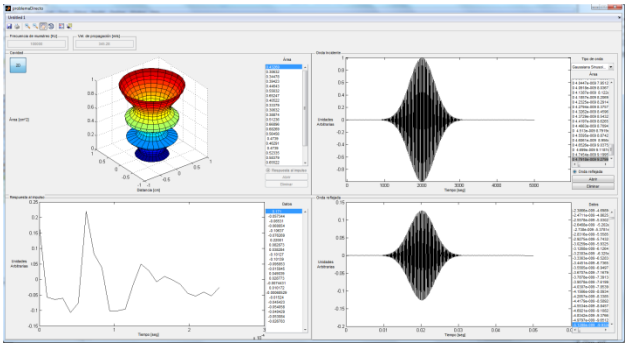

Fig. 7. *Interfaz gráfica completa*

3.1.2. PROBLEMA INVERSO

Onda incidente

Al igual que en el problema directo, en el problema inverso es posible introducir la onda incidente que el usuario desea, existiendo la posibilidad de tres tipos de ondas diferenres como son: un pulso acústico, un barrido en frecuencia de gaussianas sinusoidales o una onda acústica arbitraria.

Esta información es solicitada al usuario por medio de la siguiente ventana:

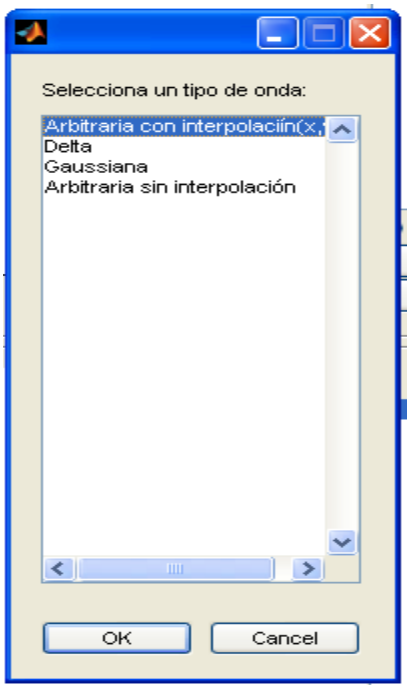

Fig 8. *Ventana para seleccionar el tipo de onda incidente*

Una vez que el tipo de onda ha sido introducido, es necesario especificar la posición del pico y el número de muestras, en caso de que la onda incidente sea una barrido en frecuencia de ondas gaussianas sinusoidales.

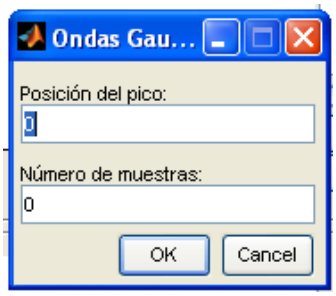

Fig.9. *Ventana para introducir el pico y el número de muestras*

Onda reflejada.

La onda reflejada será considerada del mismo tipo que la onda incidente, solo es necesario especificar el nombre del archivo que la contiene.

IFourier

Es el nombre del control de interfaz de usuario del tipo “Select button” que por medio del evento “callback”manda a llamar todo el proceso para resolver el problema inverso, como último parámetro es necesario introducir el orden de la cavidad, esto es el número aproximado de segmentos que tiene la cavidad que se desea estimar.

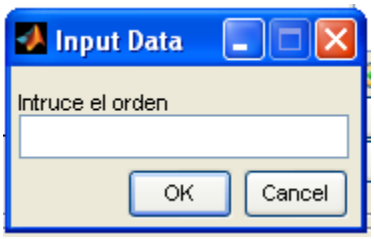

Fig.10. *Aproximación del número de segmentos que contiene la cavidad.*

CAPÍTULO 4

4.1. INSTRUMENTACIÓN DEL PROBLEMA INVERSO

En el laboratorio de sensores y señales de la Universidad Autónoma Metropolitana se realiza el experimento para resolver el problema inverso, el cual se lleva a cabo mediante un reflectómetro acústico (Fig. 11).

***Fig. 11**. Prototipo del reflectómetro acústico.*

La puesta en marcha de este dispositivo consiste en hacer llegar una onda digital, previamente diseñada por software y procesada por un amplificador de potencia, a una bocina a través de una tarjeta de adquisición de datos.

La onda acústica generada se propaga a lo largo de un tubo fuente que sirve como guía de onda. En el otro extremo del tubo fuente se acopla la cavidad de estudio donde es impactada la onda acústica.

Como se describió en la sección de fundamentos teóricos, debido al cambio de impedancia que detecta la onda incidente se genera una onda reflejada, la cual se propaga en dirección opuesta a la onda incidente.

Un micrófono colocado al ras de la pared interna del tubo fuente registra ambas ondas acústicas, las cuales son amplificadas utilizando un amplificador de instrumentación y almacenadas en una computadora usando la misma tarjeta de adquisición de datos.

Para el diseño y construcción del reflectómetro acústico se utilizó un tubo fuente largo. De esta manera se puede identificar la onda incidente de la reflejada.

Sin embargo, debido a que en este trabajo se realizó el análisis de una región en el estado estacionario alrededor del pico de la onda para obtener su amplitud y fase, el largo del tubo fuente permitió generar un tiempo para llevar a cabo el registro para su posterior análisis.

Un diagrama esquemático del reflectómetro acústico se muestra en la siguiente Fig. 12.

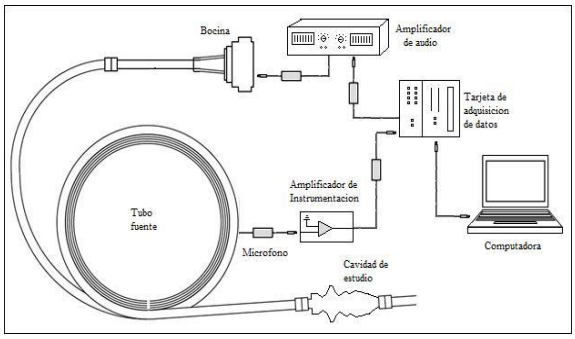

Fig. 12. *Diagrama esquemático*

Para llevar a cabo la implementación de ésta propuesta en ingeniería, se usó un modelo in vitro de una vía aérea humana [9].

Las dimensiones de esta cavidad de estudio representan la forma del área de un sujeto normal de 40 años, ver Fig. 13.

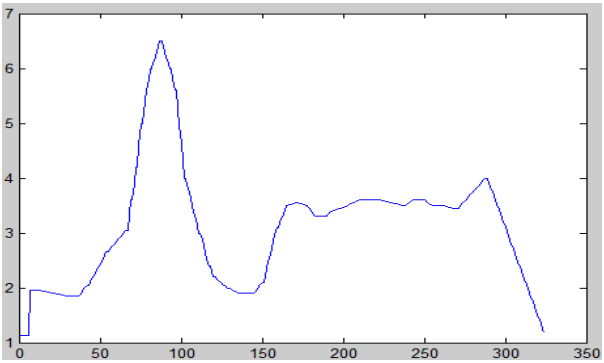

Fig. 13. *Área de cavidad cilíndrica.*

4.2. COMPARACIÓN DE RESULTADOS

Durante el proceso experimental se utilizó la cavidad propuesta en la Fig. 9 y se introdujo una onda gaussiana sinusoidal en un rango de frecuencia de 50 Hz hasta 500 Hz, con intervalos de 50 Hz.

En la Fig. 14 se muestra la onda incidente usada para el experimento, la amplitud de la onda se encuentra en unidades arbitrarias mientras que el eje de las absisas muestra el tiempo en segundos.

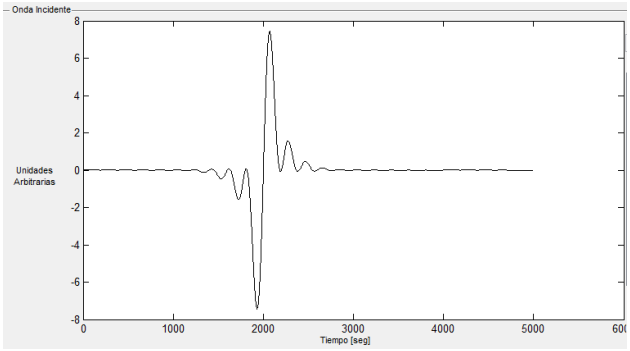

Fig. 14. *Onda incidente propuesta para el experimento.*

El experimento arrojó como resultado la onda reflejada que se muestra en la Figura 15.

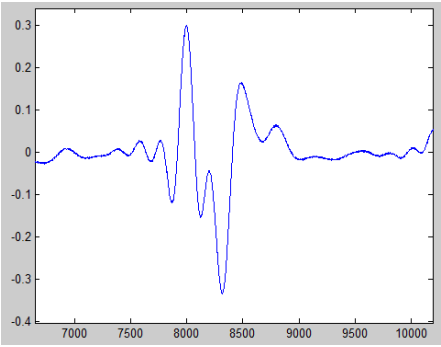

Fig. 15. *Resultado del experimento.*

La simulación fue alimentada con los datos de la cavidad y de la onda incidente usados en el experimento, obteniendo la siguiente onda reflejada mostrada en la Figura 16.

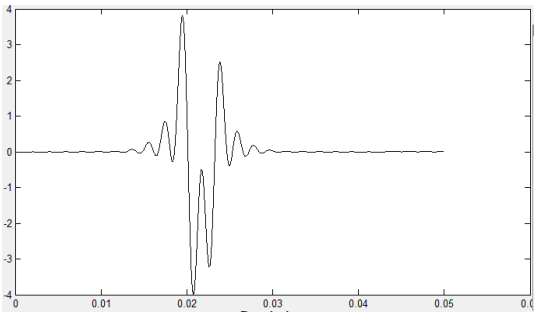

Fig. 16. *Resultado de la simulación*

Comparando ambos resultados podemos observar que son muy similares; las diferencias entre ambas ondas reflejadas se deben al ruido y a algunas condiciones que interfieren en el experimento, en cambio en la simulación la respuesta obtenida se encuentra en condiciones ideales.

4.3 CONCLUSIONES

Por medio de los experimentos se ha podido comprobar que los resultados son exitosos, la onda reflejada generada por la interfaz muestra gran similitud a la onda reflejada obtenida experimentalmente, por lo tanto la interfaz gráfica proporciona un medio eficiente y confiable para reproducir la onda reflejada a partir de los datos proporcionados al sistema, lo cual abarata costos en la experimentación y brinda la posibilidad de obtener ondas reflejadas fácilmente en condiciones ideales que no podrían ser reproducidas en el laboratorio. Además de que proporciona un medio propedéutico amigable para la enseñanza de reflectometría acústica.

CAPÍTULO 5

5.1. MANUAL DE USUARIO

Como requerimiento es necesario contar con el software Matlab 7.8.0 ó R2009a.

Es necesario ejecutar inicio.m, esto se puede hacer por medio de la tecla F5 cuando el script se encuentra abierto ó bien tecleando inicio en línea de comandos, obteniendo la siguiente ventana:

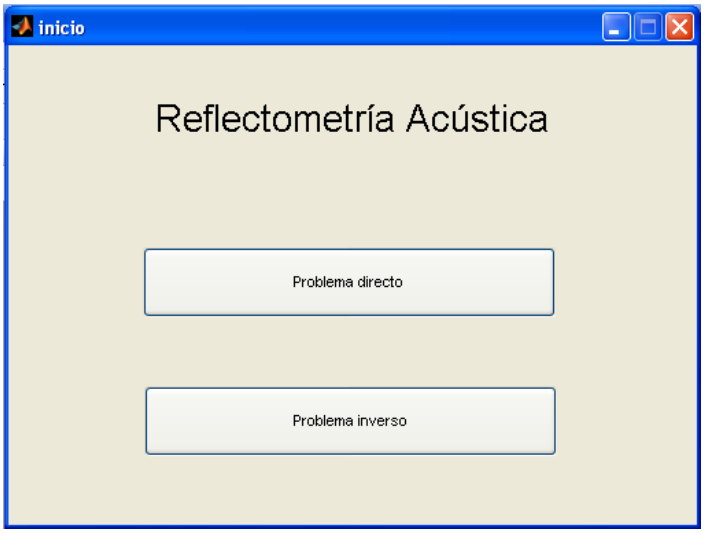

Fig.17 *Ventana de inicio*

En este punto el software proporciona dos opciones, el problema directo y el problema, solo es necesario seleccionar el botón de su preferencia.

Existen algunos controles en la parte superior de la ventana en el problema directo y en el problema inverso, como es la opción de guardar, imprimir, zoom, rotar el sólido de revolución o bien sólo desplazar una grafica, además de contar con controles para modificar la frecuencia de muestreo y la velocidad del sonido, estos pueden ser modificadas sólo si la cavidad no se encuentra en 3D, ya que si la opción de 3D se encuentra activada, estos controles se desactivan automáticamente para no generar errores.

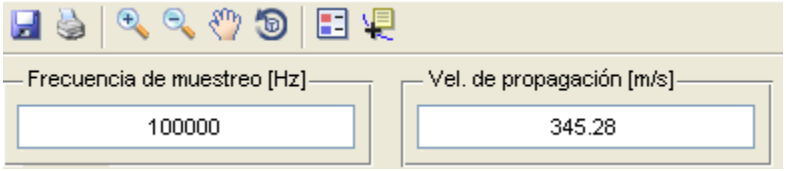

Fig.18 *Controles de la parte superior*

5.1.1 PROBLEMA DIRECTO

El sistema mostrará la siguiente ventana, por medio de la cual se puede ingresar el largo de la cavidad así como también el área transversal de la misma.

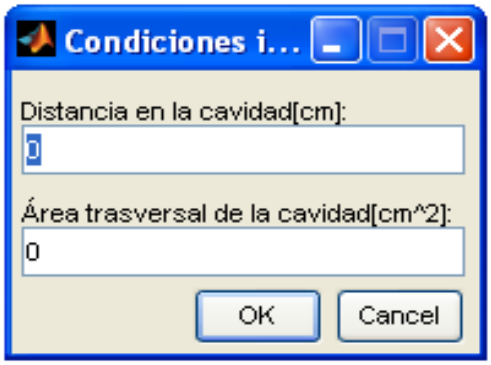

Fig.19. *Condiciones iniciales para la cavidad cilíndrica.*

También será necesario introducir las condiciones iniciales para la onda incidente, como se muestra en la figura 20.

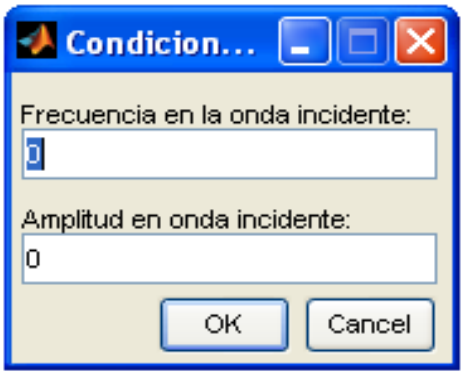

Fig.20. *Condiciones iniciales para la onda incidente.*

Después será necesario introducir el área transversal de la cavidad cilíndrica en cm, existen dos formas de hacerlo, la primera es por medio del mouse seleccionando los puntos en la siguiente área de dibujo:

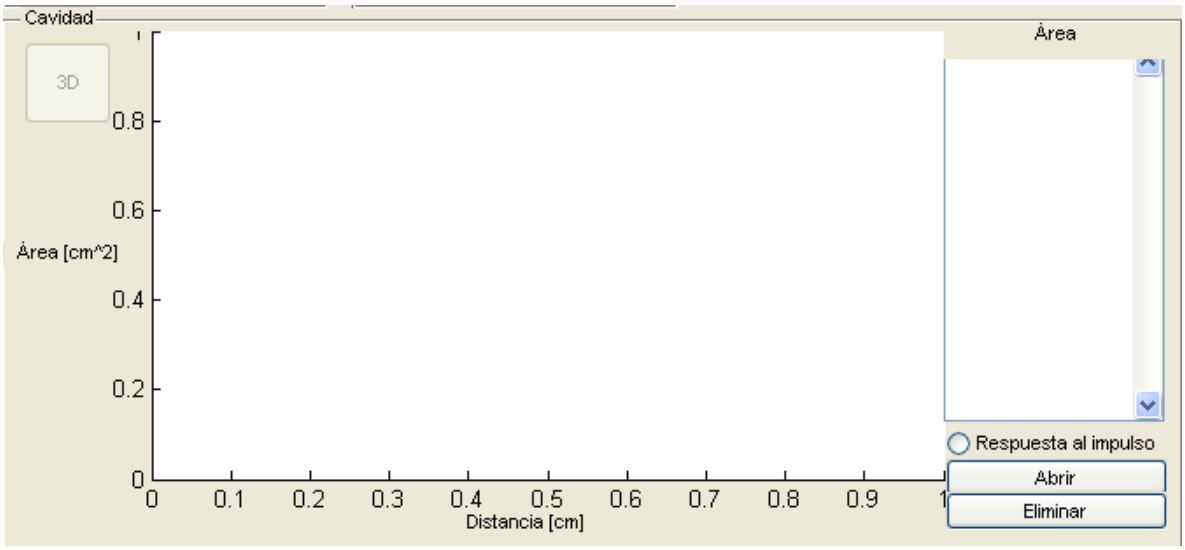

Fig.21. *Área para introducir la cavidad cilíndrica*

5.1.1.1. Botón abrir

O bien presionando el botón Abrir, el cual mostrará un cuadro de dialogo para poder seleccionar el archivo *.mat que se desea abrir.

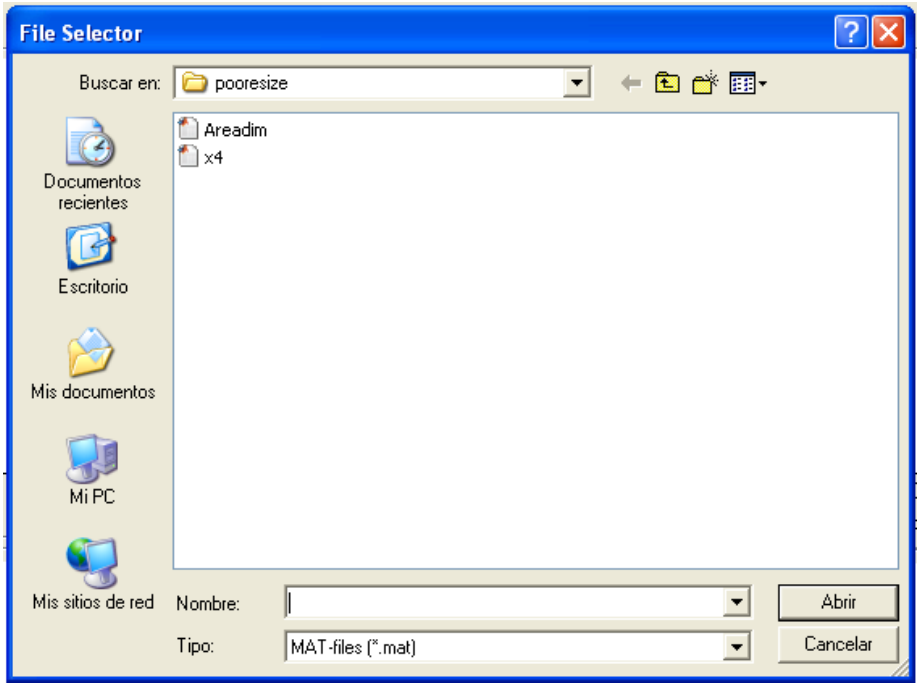

Fig.22 *Ventana para seleccionar el archivo que se desea abrir*

La siguiente ventana muestra los variables que contiene el archivo *.mat

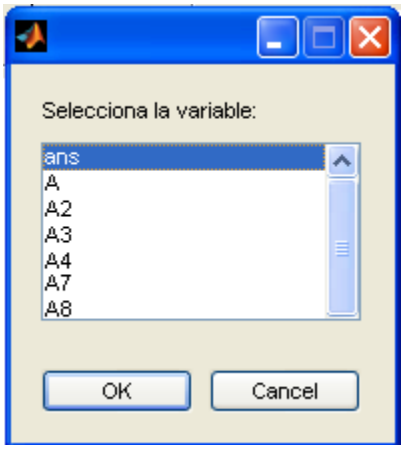

Fig.23. *Ventana para seleccionar una variable dentro de un archivo *.mat*

Una vez que se ha seleccionado se mostrará en el área de dibujo.

RESPUESTA AL IMPULSO

El siguiente paso es obtener la respuesta al impulso de la cavidad propuesta, pa esto sólo es necesario seleccionar de respuesta al impulso, como se muestra a continuación:

Fig.24. *Control que genera la respuesta al impulso*

ONDA INCIDENTE

El sistema trabaja con tres tipos de ondas incidentes, como se muestra en la figura 24.

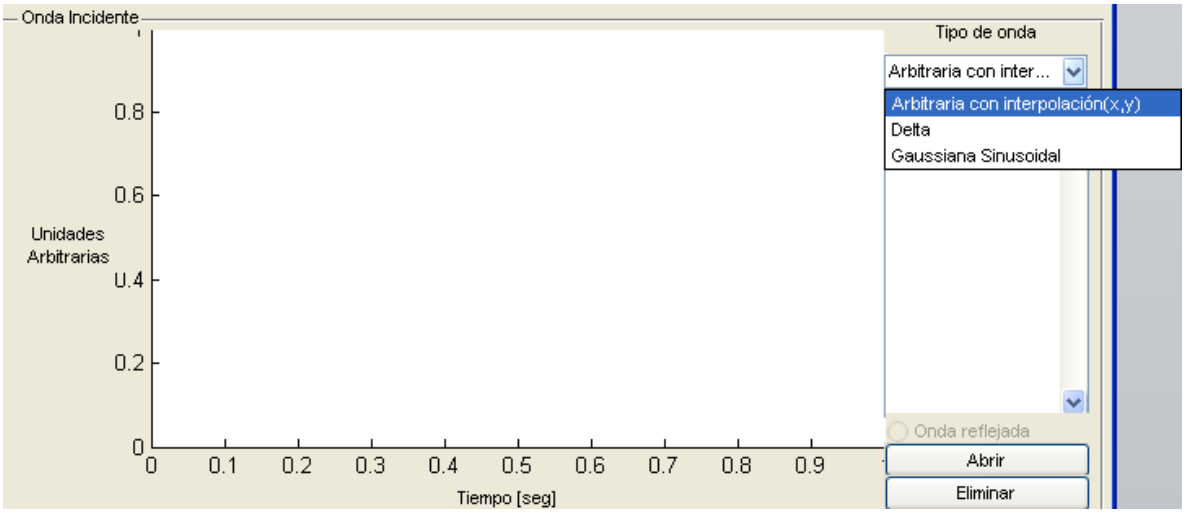

Fig.25 *Área para introducir la onda incidente.*

Onda Arbitraria

Se genera la onterpolación de los puntos introducidos para la onda arbitraria, estos puntos pueden ser introducidos por medio del mouse o con un archivo ya existente como se muestra en la sección 5.1.1.1.

Delta

Se genera en base a la longitud de la cavidad, basta con sólo ser seleccionada.

Gaussiana Sinusoidal

Requiere ciertos valores de entrada, que pueden ser introducidos por medio de la siguiente ventana:

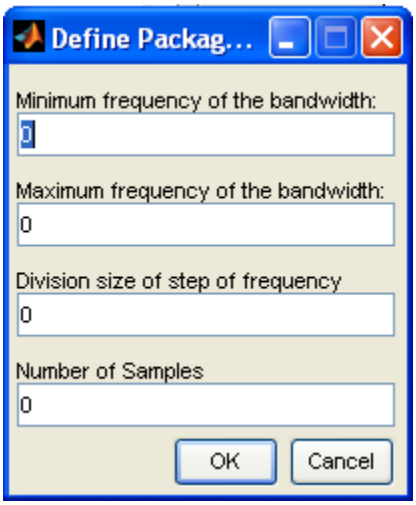

Fig.26 *Valores requeridos para generar el barrido de ondas gaussianas sinusoidales.*

Una vez que han sido introducidos los valores, es necesario seleccionar el botón de Onda Reflejada (Fig. 26), el cual se activará cuando los datos introducidos sean correctos.

Fig. 27 *Botón para gráficar la onda reflejada.*

Se puede observar un ejemplo del problema directo funcionando con ondas gaussianas sinusoidales en la **Fig. 7.** Interfaz *gráfica completa*

5.1.2 PROBLEMA INVERSO

El problema inverso comienza mostrando

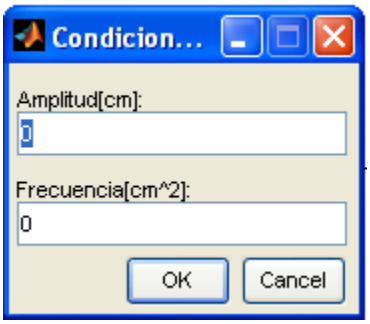

Fig.28 *Condiciones iniciales para la onda incidente en el problema inverso.*

Una vez que se han proporcionado estos datos es necesario introducir la amplitud y la frecuencia para la onda reflejada, como se muestra en la figura 28.

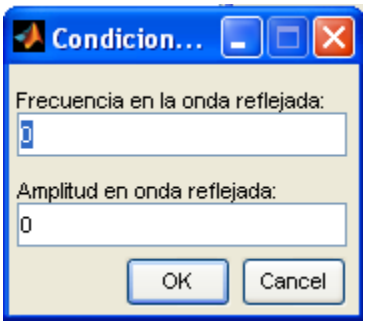

Fig.29 *Condiciones iniciales para la onda reflejada en el problema inverso.*

Inmediatamente después se mostrará la ventana para resolver el problema inverso, una vez aquí será necesario introducir la onda incidente, por medio del botón abrir como se mostró en la sección 3.1.2. PROBLEMA INVERSO, donde debemos seleccionar el tipo de onda que se desea comparar, para después seleccionar el archivo que deseamos abrir y por último la variable que se encuentra dentro del archivo seleccionado, y la cual deseamos graficar.

Para introducir la onda reflejada sólo será necesario seguir los pasos mencionados en la sección *5.1.1.1. Botón abrir*

Una vez que han sido introducidos todos estos datos, se debe presionar el botón de iFourier (Fig 30), por medio del cual se genera la respuesta al impulso y la cavidad en estudio.

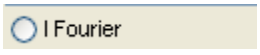

Fig. 30 *Botón para resolver el problema inverso*

Cavidad

La cavidad cuenta con un control para poder observar la gráfica en 2 y 3 dimensiones, este control se encuentra en la parte izquierda.

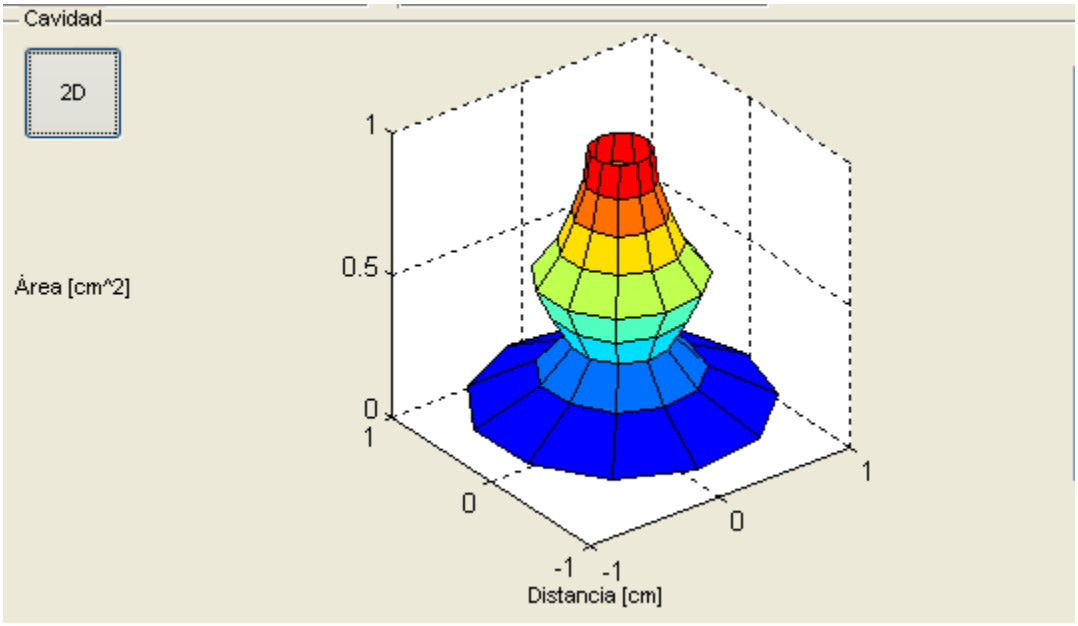

Fig.31 *Cavidad en 3D*

CAPÍTULO 6

CÓDIGO FUENTE

6.1 Inicio.m

```
function varargout = inicio(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @inicio_OpeningFcn, ...
                   'gui_OutputFcn',  @inicio_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before inicio is made visible.
function inicio_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = inicio_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;


% --- Executes on button press in Problema Directo.
function pushbutton1_Callback(hObject, eventdata, handles)
problemaDirecto;

% --- Executes on button press in Problema Inverso.
function pushbutton2_Callback(hObject, eventdata, handles)
```

```
problemaInverso;
```

6.2.ProblemaDirecto.m

```
function varargout = problemaDirecto(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@problemaDirecto_OpeningFcn, ...
                   'gui_OutputFcn',
@problemaDirecto_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function problemaDirecto_OpeningFcn(hObject, eventdata,
handles, varargin)

handles.output = hObject;

guidata(hObject, handles);


function varargout = problemaDirecto_OutputFcn(hObject,
eventdata, handles)

varargout{1} = handles.output;



function edVelprop_Callback(hObject, eventdata, handles)

function edVelprop_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edFrecmues_Callback(hObject, eventdata, handles)

function edFrecmues_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function lbOndaref_Callback(hObject, eventdata, handles)

function lbOndaref_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function lbIir_Callback(hObject, eventdata, handles)

function lbIir_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in lbOndainc.
function lbOndainc_Callback(hObject, eventdata, handles)

if get(handles.puOndaincTipo,'Value')==3
    cla(handles.axOndainc);
    plot(handles.axOndainc,...

handles.dOi_xy(get(hObject,'Value'),:),'k');

    if(size(handles.w)~=[1 1])
            cla(handles.axOndaref);
        l=length(handles.w(get(hObject,'Value'),:));
```

```
tiempo=linspace(0,(l/str2num(get(handles.edFrecmues,'String')
)),l)
        plot(handles.axOndaref,...

tiempo,handles.w(get(hObject,'Value'),:),'k');
        set(handles.lbOndaref,'Value',get(hObject,'Value'));
    end %if

end %%if
%        contents{get(hObject,'Value')} returns selected item
from lbOndainc


% --- Executes during object creation, after setting all
properties.
function lbOndainc_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in rbOndainc.
function rbOndainc_Callback(hObject, eventdata, handles)

    if get(hObject,'Value') == get(hObject,'Min')
            %cla(CavityAx); Aquí falta definir nuevamente los
puntos de
            cla(handles.axOndaref);
            set(handles.lbOndaref,'String',' ');
            %set(src,'Value',get(src,'Min'));
            return;
    else
            if  handles.dOi_xy==0
            errordlg('Es necesario proponer una onda
incidente','Error');
            set(hObject,'Value',get(hObject,'Min'));
            else
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Frecmues=str2num(get(handles.edFrecmues,'String'));
            Velprop=str2num(get(handles.edVelprop,'String'));
            puOndaincValor =
get(handles.puOndaincTipo,'Value');
            switch (puOndaincValor);
```

```
                case 1 % User selects Arbitrary function.
                    hold on;
                    cs1 =
csapi(handles.dOiArbitraria(:,1),handles.dOiArbitraria(:,2));
                    handles.dOi_xy=
fnval(cs1,linspace(handles.dOiArbitraria(1,1),handles.dOiArbi
traria(handles.dOi_n-1,1),5000));
                    hold on;
                    handles.t=1;
                    fnplt(cs1);
                    hold off;
                    handles.w=[];

handles.w=filter(handles.iir1,1,handles.dOi_xy);
                case 2 % User selects delta function
                    handles.w=[];
                    handles.t=2;
                    disp('disp')
                    disp(handles.dOi_xy);

handles.w=filter(handles.iir1,1,handles.dOi_xy);
                case 3 % User selects gaussian function
                    [i2,j2]=size(handles.dOi_xy);
                    handles.w=[];
                    handles.t=3;
                    %matlabpool open local 4
                    hwait = waitbar(0,'Please wait...');
                    steps=i2;
                    for n=1:i2 %parfor

w(n,:)=filter(handles.iir1,1,handles.dOi_xy(n,:));
                        %fnplt(RefwavAx,...
                        %        w,'k');
                        waitbar(n / steps);
                    end %parfor
                    handles.w=w;
                    close(hwait)
                    %matlabpool close;
                case 4 % User selects open file
                    handles.w=[];

handles.w=filter(handles.iir1,1,handles.dOi_xy);
            end
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            [izqi,deri]=size(handles.iir1);
            if (izqi>1 || deri>1)
                [izq,der]=size(handles.w);
```

```
                if(izq>der)
                    strtxt = num2str(handles.w);
                else
                    strtxt = num2str(handles.w');
                end

set(handles.lbOndaref,'String',strtxt);
                l=length(handles.w);
                tiempo=linspace(0,(l/Frecmues),l)
                plot(handles.axOndaref,...
                    tiempo,handles.w,'k');
                disp('si entre')

            end

    end
    guidata(hObject, handles);
end

function pbOndaincAbrir_Callback(hObject, eventdata, handles)

[filename, pathname] = ...
        uigetfile({'*.mat'},'File Selector');
        if isequal(filename,0)
            disp('User selected Cancel')
            set(src,'Enable','On');
        else

            disp(['User selected', fullfile(pathname,
filename)]);
            fname = load([pathname filename]);
            [s,v] = listdlg('PromptString','Selecciona un
tipo de onda:',...
                'SelectionMode','single',...
                'ListString',{'Arbitraria con
interpolaciín(x,y)','Delta','Gaussiana','Arbitraria sin
interpolación'})
            set(handles.puOndaincTipo,'Value',s);
            names=fieldnames(fname);
            [name_var,V2]= listdlg('PromptString','Selecciona
la variable:',...
                'SelectionMode','single',...
                'ListString',names)
            handles.dOi_xy=fname.(names{name_var});
            handles.dOi_n=size(handles.dOi_xy);
            switch s
                case 1
```

```
                    a(:,1)=handles.dOi_xy(:,1);
                    a(:,2)=handles.dOi_xy(:,2);
                    cs1 =
csapi(handles.dOi_xy(:,1),handles.dOi_xy(:,2));
                    fnplt(cs1);
                case 2
                    stem(handles.axOndainc,handles.dOi_xy);
                case 3
                    x=sum(handles.dOi_xy(1,:));
                    y=sum(handles.dOi_xy(:,1));
                    loni=x-y;
                    if((loni+y)==0)
                        handles.dOi_xy=handles.dOi_xy';
                    end
                    plot(handles.axOndainc,...
                                    handles.dOi_xy','k');
                case 4
                    plot(handles.axOndainc,...
                                    handles.dOi_xy','k');
            end
            strtxt = num2str(handles.dOi_xy);
            set(handles.lbOndainc,'String',strtxt);
            guidata(hObject, handles);
        end

% --- Executes on button press in pbOndaincEliminar.
function pbOndaincEliminar_Callback(hObject, eventdata,
handles)

handles.dOi_xy=0;
handles.dOi_n=1;
set(handles.lbOndainc,'String',[]);
set(handles.lbOndaref,'String',[]);
cla(handles.axOndainc);
cla(handles.axOndaref);
set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Min'));
guidata(hObject, handles);


function rbCavidad_Callback(hObject, eventdata, handles)
    if get(hObject,'Value') == get(hObject,'Min')
            %cla(CavityAx); Aquí falta definir nuevamente los
puntos de
            cla(handles.axCavidad);
            cla(handles.axIir);
            set(handles.lbIir,'String',' ');
            %set(handles.rbCavidad3d,'Enable','off');
```

```
            set(handles.tbCavidadDim,'Enable','off');
            set(handles.rbOndainc,'Enable','off');
            %set(src,'Value',get(src,'Min'));
    else
            if  handles.dC_xy==0
                errordlg('It is necessary to propose the
cavity','Error');
                set(hObject,'Value',get(hObject,'Min'));
            else
                set(handles.tbCavidadDim,'Enable','on');
                set(handles.rbOndainc,'Enable','on');

Frecmues=str2double(get(handles.edFrecmues,'String'));

Velprop=str2double(get(handles.edVelprop,'String'));
            area1=handles.dC_xy/2;
            lon2=Velprop./(2*Frecmues);
            cla(handles.axCavidad);
            %%hold off;

handles.lon=linspace(0,(lon2*handles.dC_n),handles.dC_n-1);
            hold on;
            stairs
(handles.axCavidad,handles.lon',area1,'k','LineWidth',3,'Butt
onDownFcn','down_Cavity=1;');
            hold on;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Iir
            [coefi,handles.iir1]=iir(handles.dC_xy,Frecmues);
                [n2,m2]=size(handles.iir1);
                lon4=linspace(0,(m2/Frecmues),m2);
                [izq,der]=size(handles.iir1);
            if(izq>der)
                strtxt = num2str(handles.iir1);
            else
                strtxt = num2str(handles.iir1');
            end
                    set(handles.lbIir,'String',strtxt);
                    cla(handles.axIir);
                    hold off;
                plot(handles.axIir,...
                    lon4,handles.iir1,'k');
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            guidata(hObject, handles);
            end
end
```

```
function pbCavidadAbrir_Callback(hObject, eventdata, handles)

[filename, pathname] = ...
        uigetfile({'*.mat'},'File Selector');
        if isequal(filename,0)
            disp('User selected Cancel')
            set(src,'Enable','On');
        else
            fname = load([pathname filename]);
            names=fieldnames(fname);
            [name_var,V2]= listdlg('PromptString','Selecciona
la variable:',...
                'SelectionMode','single',...
                'ListString',names)
            handles.dC_xy=fname.(names{name_var});
            [izq,der]=size(handles.dC_xy);
            if(izq>der)
                handles.dC_n=izq;
            else
                handles.dC_n=der;
                handles.dC_xy=handles.dC_xy';
            end
            handles.dC_n=handles.dC_n+1;
            strtxt = num2str(handles.dC_xy);
            set(handles.lbCavidad,'String',strtxt);
            guidata(hObject, handles);
        end

% --- Executes on button press in pbCavidadCerrar.
function pbCavidadCerrar_Callback(hObject, eventdata,
handles)

handles.dC_xy=0;
handles.dC_n=1;
handles.iir1=0;
set(handles.lbCavidad,'String',[]);
set(handles.lbIir,'String',[]);
set(handles.lbOndaref,'String',[]);
set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Min'));
set(handles.tbCavidadDim,'Value',get(handles.tbCavidadDim,'Mi
n'));
set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Min'));
cla(handles.axIir);
cla(handles.axOndaref);
cla(handles.axCavidad);
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all
properties.
function axCavidad_CreateFcn(hObject, eventdata, handles)

 set(hObject,'XLim',[0 handles.DCavDistancia],'YLim',[0
handles.DCavArea])
 handles.txtCavidad=text(0,0,'(0,0)','Visible','off');
 guidata(hObject, handles);

% --- Executes during object creation, after setting all
properties.
function figure1_CreateFcn(hObject, eventdata, handles)

handles.dC_xy=0;
handles.dC_n=1;
handles.dClb=0;
handles.w=0;
handles.dOi_xy=0;
handles.dOiArbitraria=[0,0];
handles.dOi_n=1;
%%%%%%%%%%%%%regresa = true;
regresa = true;
    while regresa,
        prompt = {'Distancia en la cavidad[cm]:','Área
trasversal de la cavidad[cm^2]:'};
                    dlg_title = 'Condiciones iniciales
Cavidad';
                    num_lines = 1;
                    def = {'0','0','0','0'};
                    answer =
inputdlg(prompt,dlg_title,num_lines,def);

handles.DCavDistancia=str2double(answer{1});
                    handles.DCavArea=str2double(answer{2});
         if ((handles.DCavDistancia+handles.DCavArea)>=2)
                    regresa = false;
         end
    end
%%%%%%%%%5%%%
guidata(hObject, handles);

% --- Executes during object creation, after setting all
properties.
function axOndainc_CreateFcn(hObject, eventdata, handles)

regresa = true;
    while regresa,
```

```
        prompt = {'Unidades Arbitrarias:','Tiempo [Seg]:'};
                    dlg_title = 'Condiciones iniciales Onda
incidente';
                    num_lines = 1;
                    def = {'0','0'};
                    answer =
inputdlg(prompt,dlg_title,num_lines,def);
                    DOIncFrecuencia=str2double(answer{1});
                    DOIncAmplitud=str2double(answer{2});
         if ((DOIncFrecuencia+DOIncAmplitud)>=2)
                    regresa = false;
         end
    end
 set(hObject,'XLim',[0 DOIncFrecuencia],'YLim',[0
DOIncAmplitud])
 handles.txtOndainc=text(0,0,'(0,0)','Visible','off');
 guidata(hObject, handles);


% --- Executes on mouse motion over figure - except title and
menu.
function figure1_WindowButtonMotionFcn(hObject, eventdata,
handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
%set(hObject,'XLim',[0 DOIncFrecuencia],'YLim',[0
DOIncAmplitud])
% pt1 = get(handles.axCavidad,'CurrentPoint');
% pt1x=get(handles.axCavidad,'XLim');
% pt1y=get(handles.axCavidad,'YLim');
% pt2 = get(handles.axOndainc,'CurrentPoint');
% pt2x=get(handles.axOndainc,'XLim');
% pt2y=get(handles.axOndainc,'YLim');
% if (pt1(1)<=pt1x(2)) && (pt1(3)<pt1y(2)) && pt1(1)>0 &&
pt1(3)>0
%           pt=pt1;
%           h=handles.txtCavidad;
%           set(h,'Visible','off')
% elseif (pt2(1)<=pt2x(2)) && (pt2(3)<=pt2y(2)) && pt2(1)>0
&& pt2(3)>0
%           pt=pt2;
%           h=handles.txtOndainc;
%           set(h,'Visible','off')
% else
```

```
%     set(handles.txtOndainc,'Visible','off')
%     set(handles.txtCavidad,'Visible','off')
%     return;
% end
%         strx = num2str(pt(1),'%10.2f');
%         stry = num2str(pt(3),'%10.2f');
%         str = strcat('(',strx,',',stry,')');
%         res(1)=pt(1);
%         res(2)=pt(3);
% set(h,'Position',res,'String',str,'Visible','on')
% guidata(hObject, handles);

% --- Executes on mouse press over axes background.
function axCavidad_ButtonDownFcn(hObject, eventdata, handles)

pt = get(hObject,'CurrentPoint');
text(pt(1),pt(3),'°','Color','Red');
handles.dC_xy(handles.dC_n,:)=pt(3);
strtxt = num2str(handles.dC_xy);
set(handles.lbCavidad,'String',strtxt);
handles.dC_n=handles.dC_n+1;
guidata(hObject, handles);

% --- Executes on mouse press over axes background.
function axOndainc_ButtonDownFcn(hObject, eventdata, handles)

pt = get(hObject,'CurrentPoint');
text(pt(1),pt(3),'°','Color','Blue');
handles.dOi_xy=1;
handles.dOiArbitraria(handles.dOi_n,:)=[pt(1);pt(3)];
strtxt = num2str(handles.dOiArbitraria);
set(handles.lbOndainc,'String',strtxt);
handles.dOi_n=handles.dOi_n+1;
guidata(hObject, handles);

% --- Executes on selection change in puOndaincTipo.
function puOndaincTipo_Callback(hObject, eventdata, handles)

switch get(hObject,'Value');
    case 2
        cla(handles.axOndainc);
        hold on;
        handles.dOi_xy=[1 zeros(1,(handles.dC_n-2))];
        cs=handles.dOi_xy;
        stem(handles.axOndainc,handles.dOi_xy)
        guidata(hObject, handles);
    case 3
```

```
        %User selects Gaussian wave
        reset(handles.axOndainc);

        handles.dOi_xy=0;
        prompt = {'Minimum frequency of the
bandwidth:','Maximum frequency of the bandwidth:','Division
size of step of frequency','Number of Samples'};
                dlg_title = 'Define Package of Gaussian
waves';
                num_lines = 1;
                def = {'0','0','0','0'};
                answer =
inputdlg(prompt,dlg_title,num_lines,def);
        handles.dOi_Fsmin=str2double(answer{1});
        handles.dOi_Fsmax=str2double(answer{2});
        handles.dOi_division=str2double(answer{3});
        handles.dOi_numsamples=str2double(answer{4});

Frecmues=str2double(get(handles.edFrecmues,'String'));

handles.dOi_xy=modulated(handles.dOi_Fsmin,handles.dOi_Fsmax,
handles.dOi_division,handles.dOi_numsamples,Frecmues);
                strtxt = num2str(handles.dOi_xy);

set(handles.lbOndainc,'String',strtxt);
                plot(handles.axOndainc,...
                        handles.dOi_xy','k');
                [n,handles.dOi_n]=size(handles.dOi_xy);
        guidata(hObject, handles);
end

% --- Executes during object creation, after setting all
properties.
function puOndaincTipo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on edVelprop and none
of its controls.
function edVelprop_KeyPressFcn(hObject, eventdata, handles)

if get(handles.rbCavidad,'Value') ==
get(handles.rbCavidad,'Max')
    if(strcmp(eventdata.Key,'return'))
```

```
set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Min'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Max'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);
    end %if
end %if rbcavidad


% --- Executes on key press with focus on edFrecmues and none
of its controls.
function edFrecmues_KeyPressFcn(hObject, eventdata, handles)
if get(handles.rbCavidad,'Value') ==
get(handles.rbCavidad,'Max')
    if(strcmp(eventdata.Key,'return'))

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Min'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Max'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);
    end
end %if rbcavidad
if get(handles.rbCavidad,'Value') ==
get(handles.rbCavidad,'Max')
    if(strcmp(eventdata.Key,'return'))

set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Min'));
    rbOndainc_Callback(handles.rbOndainc,eventdata,handles);

set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Max'));
    rbOndainc_Callback(handles.rbOndainc,eventdata,handles);
    end
end

% --- Executes on button press in tbCavidadDim.
function tbCavidadDim_Callback(hObject, eventdata, handles)
button_state = get(hObject,'Value');
    if button_state == get(hObject,'Max')
        set(hObject,'String','2D')
        set(handles.edFrecmues,'Enable','off');
        set(handles.edVelprop,'Enable','off');
        set(handles.rbCavidad,'Enable','off');
        set(handles.pbCavidadCerrar,'Enable','off');
        set(handles.pbCavidadAbrir,'Enable','off');
```

```
        if get(handles.rbCavidad,'Value') ==
get(handles.rbCavidad,'Max')
        [X,Y,Z] = cylinder(handles.dC_xy,handles.dC_n);
        surf(X,Y,Z)
        hold on;
        axis square
        end %if rbCavidad
    elseif button_state == get(hObject,'Min')
        reset(handles.axCavidad);

%%axCavidad_DeleteFcn(handles.axCavidad,eventdata,handles);

axCavidad_CreateFcn(handles.axCavidad,eventdata,handles);
        hold on;
        set(hObject,'String','3D')
        set(handles.edFrecmues,'Enable','on');
        set(handles.edVelprop,'Enable','on');
        set(handles.rbCavidad,'Enable','on');
        set(handles.pbCavidadAbrir,'Enable','on');

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Min'));

rbCavidad_Callback(handles.rbCavidad,eventdata,handles);

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Max'));

rbCavidad_Callback(handles.rbCavidad,eventdata,handles);
        set(handles.pbCavidadCerrar,'Enable','on');
    end

% --- If Enable == 'on', executes on mouse press in 5 pixel
border.
% --- Otherwise, executes on mouse press in 5 pixel border or
over lbCavidad.

% --- Executes on selection change in lbCavidad.
function lbCavidad_Callback(hObject, eventdata, handles)
disp(get(hObject,'String'));
disp(get(hObject,'Value'));
% --- Executes during object deletion, before destroying
properties.
function axCavidad_DeleteFcn(hObject, eventdata, handles)


% --- Executes during object creation, after setting all
properties.
function lbCavidad_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all
properties.
function rbOndainc_CreateFcn(hObject, eventdata, handles)


% ------------------------------------------------------------
---------
function uipushtool4_ClickedCallback(hObject, eventdata,
handles)
ondaref=handles.w;
ondainc = handles.dOi_xy;
cavidad=handles.dC_xy;
iir=handles.iir1
FolderName = inputdlg('Nombre del proyecto','Input Data');
[stat,struc] = fileattrib;
 PathCurrent = struc.Name;
 PathFolder=[PathCurrent '\PD' FolderName{1} '\']
 mkdir([PathFolder]);
%%save [PathFolder 'convolucionOinc'];
 save ([PathFolder
FolderName{1}],'ondaref','ondainc','cavidad','iir');

% --- Executes on mouse press over axes bmackground.
function axIir_ButtonDownFcn(hObject, eventdata, handles)
% --- Executes during object creation, after setting all
properties.
function pbOndaincEliminar_CreateFcn(hObject, eventdata,
handles)
```

6.3. ProblemaInverso.m

```
function varargout = problemaDirecto(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@problemaDirecto_OpeningFcn, ...
                   'gui_OutputFcn',
@problemaDirecto_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before problemaDirecto is made visible.
function problemaDirecto_OpeningFcn(hObject, eventdata,
handles, varargin)
handles.output = hObject;


guidata(hObject, handles);


function varargout = problemaDirecto_OutputFcn(hObject,
eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;



function edVelprop_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function edVelprop_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edFrecmues_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function edFrecmues_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in lbOndaref.
function lbOndaref_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function lbOndaref_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in lbIir.
function lbIir_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function lbIir_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in lbOndainc.
function lbOndainc_Callback(hObject, eventdata, handles)
if get(handles.puOndaincTipo,'Value')==3
```

```
    cla(handles.axOndainc);
    plot(handles.axOndainc,...

handles.dOi_xy(get(hObject,'Value'),:),'k');

    if(size(handles.dOref_xy)~=1)
            cla(handles.axOndaref);
        l=length(handles.dOi_xy(get(hObject,'Value'),:));

tiempo=linspace(0,(l/str2num(get(handles.edFrecmues,'String')
)),l)
        plot(handles.axOndaref,...

tiempo,handles.dOi_xy(get(hObject,'Value'),:),'k');
        set(handles.lbOndaref,'Value',get(hObject,'Value'));
    end %if

end %%if%  cellstr(get(hObject,'String')) returns lbOndainc
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from lbOndainc


% --- Executes during object creation, after setting all
properties.
function lbOndainc_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in rbOndainc.
function rbOndainc_Callback(hObject, eventdata, handles)
    if get(hObject,'Value') == get(hObject,'Min')
            %cla(CavityAx); Aquí falta definir nuevamente los
puntos de
            cla(handles.axOndaref);
            set(handles.lbOndaref,'String',' ');
            %set(src,'Value',get(src,'Min'));
            return;
    else

            set(handles.tbCavidadDim,'Enable','on');

Frecmues=str2double(get(handles.edFrecmues,'String'));
```

```
Velprop=str2double(get(handles.edVelprop,'String'));
            puOndaincValor =
get(handles.puOndaincTipo,'Value');
            disp('El orden deseado es: ');
                    disp(size(handles.dOi_xy));
                    OincOrden = inputdlg('Intruce el
orden','Input Data');

handles.OincOrden=str2double(OincOrden{1});
            switch (puOndaincValor);
                case 1 % User selects Arbitrary function.
                    hold on;


handles.ifourier=svd6(handles.dOi_xy',handles.dOref_xy,handle
s.OincOrden)%handles.OincOrden);%%%%%%%%%%%%%%%%%%%%%%%%%%%%Aq
ui
                    ifourier=handles.ifourier;
                    [n,m]=size(handles.ifourier);
                    t=linspace(0,m/Frecmues,m);
                    plot (handles.axIir,t,handles.ifourier);

[c,diametro,d2]=wasp(handles.ifourier,Frecmues);
                    handles.diametro=diametro;
                    lon=(Velprop.*100)./(2.*Frecmues);

x4=linspace(0,length(diametro)*lon,length(diametro));%.5754
                    stairs
(handles.axCavidad,x4,diametro,'k','LineWidth',3,'ButtonDownF
cn','down_Cavity=1;');
                    strtxt1 = num2str(handles.ifourier');
                    set(handles.lbIir,'String',strtxt1);
                    strtxt2 = num2str(diametro');
                    set(handles.lbCavidad,'String',strtxt2);

                case 2 % User selects delta function


handles.ifourier=svd6(handles.dOi_xy,handles.dOref_xy,handles
.OincOrden)%handles.OincOrden);%%%%%%%%%%%%%%%%%%%%%%%%%%%%Aqu
i
                    lon=(Velprop.*100)./(2.*Frecmues);
                    [n,m]=size(handles.ifourier);
                    t=linspace(0,m/Frecmues,m);
                    plot (handles.axIir,t,handles.ifourier)
```

```
[c,diametro,d2]=wasp(handles.ifourier,Frecmues);
                    handles.diametro=diametro;

x4=linspace(0,length(diametro)*lon,length(diametro));%.5754
                    plot(handles.axCavidad,x4,diametro,'k');
                    strtxt1 = num2str(handles.ifourier');
                    set(handles.lbIir,'String',strtxt1);
                    strtxt2 = num2str(diametro');
                    set(handles.lbCavidad,'String',strtxt2);
                case 3 % User selects gaussian function
                    for i=1:handles.dOref_izq

handles.A(i)=handles.dOref_a(i)/handles.dOi_a(i);
                           handles.P(i)=handles.dOref_p(i)-
handles.dOi_p(i);

                    end %for
                    %%%%%%%%%%%%%%%%%%%Cerón
                       OincDivin = inputdlg('Intruce la
división','Input Data');
                    OincDiv=str2double(OincDivin{1});
                    handles.ifourier =
iFourierG(Frecmues,handles.A,handles.P,OincDiv);


                    %%%%%%%%%%%%%%%%%%%%%%%%%%
                    %handles.ifourier =
iFourier(Frecmues,handles.A,handles.P);
                         lon=(Velprop.*100)./(2.*Frecmues);
                    [n,m]=size(handles.ifourier);
                    t=linspace(0,m/Frecmues,m);
                    plot (handles.axIir,t,handles.ifourier)

[coef,diametro,diametro1]=wasp(handles.ifourier,Frecmues);
                          handles.diametro=diametro;

x4=linspace(0,length(diametro)*lon,length(diametro));%.5754
                    plot(handles.axCavidad,x4,diametro,'k');
                    strtxt1 = num2str(handles.ifourier');
                    set(handles.lbIir,'String',strtxt1);
                    strtxt2 = num2str(diametro');
                    set(handles.lbCavidad,'String',strtxt2);
%
            end

 guidata(hObject, handles);
```

```
end
% --- Executes on button press in pbOndaincAbrir.
function pbOndaincAbrir_Callback(hObject, eventdata, handles)

[filename, pathname] = ...
        uigetfile({'*.mat'},'File Selector');
        if isequal(filename,0)
            disp('User selected Cancel')
            %%set(src,'Enable','On');
        else

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            disp(['User selected', fullfile(pathname,
filename)]);
            fname = load([pathname filename]);
            [s,v] = listdlg('PromptString','Selecciona un
tipo de onda:',...
                'SelectionMode','single',...
                'ListString',{'Arbitraria con
interpolaciín(x,y)','Delta','Gaussiana','Arbitraria sin
interpolación'})
            names=fieldnames(fname);
            [name_var,V2]= listdlg('PromptString','Selecciona
la variable:',...
                'SelectionMode','single',...
                'ListString',names)
            dOiArbitraria=fname.(names{name_var});
            [izq,der]=size(dOiArbitraria);
            handles.OndaInc_izq=izq;
            handles.OndaInc_der=der;
            handles.dOi_n=izq;
            handles.dOi_GMuestras=0;
            handles.dOi_GPico=0;
            handles.dOi_a=0;
            handles.dOi_w=0;
            handles.dOi_p=0;
            handles.dOi_b=0;
            switch s
                case 1
                    if(izq==2)
                        dOiArbitraria=dOiArbitraria';
                        handles.dOi_n=der;
                    end
                    cs1 =
csapi(dOiArbitraria(:,1),dOiArbitraria(:,2));
```

```
                    handles.dOi_xy=
fnval(cs1,linspace(dOiArbitraria(1,1),dOiArbitraria(handles.d
Oi_n-1,1),5000));
                    handles.dOi_xy=handles.dOi_xy';
                    hold on;
                    plot(handles.axOndainc,...
                                    handles.dOi_xy,'k');
                    hold off;
                case 2
                    if(izq>der)
                        dOiArbitraria=dOiArbitraria';
                        handles.dOi_n=der;
                    end
                    handles.dOi_xy=dOiArbitraria;
                    stem(handles.axOndainc,handles.dOi_xy);
                case 3
                    handles.dOi_xy=dOiArbitraria;
                    prompt = {'Posición del pico:','Número de
muestras:'};
                dlg_title = 'Ondas Gaussianas';
                num_lines = 1;
                def = {'0','0'};
                answer =
inputdlg(prompt,dlg_title,num_lines,def);
                handles.dOi_GPico=str2double(answer{1});

handles.dOi_GMuestras=str2double(answer{2})/2;
                    x=sum(handles.dOi_xy(1,:));
                    y=sum(handles.dOi_xy(:,1));

                    if(x==0)
                        for i=1:izq


[handles.dOi_b(i),handles.dOi_a(i),handles.dOi_w(i),handles.d
Oi_p(i)]=fitsine(handles.dOi_xy(i,(handles.dOi_GPico-
handles.dOi_GMuestras):(handles.dOi_GPico+handles.dOi_GMuestr
as)));

                        end %for
                    else
                        handles.dOi_xy=handles.dOi_xy';
                        for i=1:izq


[handles.dOi_b(i),handles.dOi_a(i),handles.dOi_w(i),handles.d
Oi_p(i)]=fitsine(handles.dOi_xy(((handles.dOi_GPico-
```

```
handles.dOi_GMuestras):(handles.dOi_GPico+handles.dOi_GMuestr
as)),i));
                        end %for
                    end
                    plot(handles.axOndainc,...
                                    handles.dOi_xy,'k');

                    wf=((handles.dOi_w(3)-
handles.dOi_w(2))*180)/pi;
                    disp(wf);

                case 4
                     if(izq>der)
                        dOiArbitraria=dOiArbitraria';
                        handles.dOi_n=der;
                    end
                    handles.dOi_xy=dOiArbitraria;
                    plot(handles.axOndainc,...
                                    handles.dOi_xy,'k');
                                s=1;
            end
                        set(handles.puOndaincTipo,'Value',s);
            strtxt = num2str(handles.dOi_xy');
            set(handles.lbOndainc,'String',strtxt);
            guidata(hObject, handles);
        end
% --- Executes on button press in pbOndaincEliminar.
function pbOndaincEliminar_Callback(hObject, eventdata,
handles)
handles.dOi_xy=[0,0];
handles.dOi_n=1;
handles.ifourier=0;
handles.OincOrden=0;
handles.diametro=0;
set(handles.lbOndainc,'String',[]);
set(handles.lbIir,'String',[]);
set(handles.lbCavidad,'String',[]);
cla(handles.axOndainc);
cla(handles.axIir);
cla(handles.axCavidad);
set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Min'));
guidata(hObject, handles);


function rbCavidad_Callback(hObject, eventdata, handles)
    if get(hObject,'Value') == get(hObject,'Min')
```

```
            %cla(CavityAx); Aquí falta definir nuevamente los
puntos de
            cla(handles.axCavidad);
            cla(handles.axIir);
            set(handles.lbIir,'String',' ');
            %set(handles.rbCavidad3d,'Enable','off');
            set(handles.tbCavidadDim,'Enable','off');
            set(handles.rbOndainc,'Enable','off');
            %set(src,'Value',get(src,'Min'));
    else
            if  handles.dC_xy==0
                errordlg('It is necessary to propose the
cavity','Error');
                set(hObject,'Value',get(hObject,'Min'));
            else
                set(handles.tbCavidadDim,'Enable','on');
                set(handles.rbOndainc,'Enable','on');

Frecmues=str2double(get(handles.edFrecmues,'String'));

Velprop=str2double(get(handles.edVelprop,'String'));
            area1=handles.dC_xy/2;
            lon2=Velprop./(2*Frecmues);
            cla(handles.axCavidad);
            %%hold off;

handles.lon=linspace(0,(lon2*handles.dC_n),handles.dC_n-1);
            hold on;
            stairs
(handles.axCavidad,handles.lon',area1,'k','LineWidth',3,'Butt
onDownFcn','down_Cavity=1;');
            hold on;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Iir
            [coefi,handles.iir1]=iir(handles.dC_xy,Frecmues);
                [n2,m2]=size(handles.iir1);
                lon4=linspace(0,(m2/Frecmues),m2);
                strtxt = num2str(handles.iir1);
                    set(handles.lbIir,'String',strtxt);
                    cla(handles.axIir);
                    hold off;
                plot(handles.axIir,...
                    lon4,handles.iir1,'k');
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            guidata(hObject, handles);
            end
end
```

```
% Hint: get(hObject,'Value') returns toggle state of
rbCavidad


% --- Executes on button press in pbCavidadAbrir.
function pbCavidadAbrir_Callback(hObject, eventdata, handles)

[filename, pathname] = ...
        uigetfile({'*.mat'},'File Selector');
        if isequal(filename,0)
            disp('User selected Cancel')
            set(src,'Enable','On');
        else
           fname = load([pathname filename]);
            names=fieldnames(fname);
            [name_var,V2]= listdlg('PromptString','Selecciona
la variable:',...
                'SelectionMode','single',...
                'ListString',names)
            handles.dOref_xy=fname.(names{name_var});
            [izq,der]=size(handles.dOref_xy);

            if(izq>der)
                handles.dOref_xy=handles.dOref_xy';
            end
            handles.dOref_n=size(handles.dOref_xy);
            strtxt = num2str(handles.dOref_xy);
            set(handles.lbOndaref,'String',strtxt);

Frecmues=str2double(get(handles.edFrecmues,'String'));
                    l=length(handles.dOref_xy);
                    tiempo=linspace(0,(l/Frecmues),l)
                    plot(handles.axOndaref,...
                         tiempo,handles.dOref_xy,'k');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            if get(handles.puOndaincTipo,'Value')==3
                handles.dOref_izq=izq;
                 x=sum(handles.dOref_xy(1,:));
                    y=sum(handles.dOref_xy(:,1));
                    if(x==0)
                        for i=1:izq


[handles.dOref_b(i),handles.dOref_a(i),handles.dOref_w(i),han
dles.dOref_p(i)]=fitsine(handles.dOref_xy((handles.dOi_GPico-
```

```
handles.dOi_GMuestras):(handles.dOi_GPico+handles.dOi_GMuestr
as),i));

                        end %for
                    else
                        handles.dOref_xy=handles.dOref_xy';
                        for i=1:izq


[handles.dOref_b(i),handles.dOref_a(i),handles.dOref_w(i),han
dles.dOref_p(i)]=fitsine(handles.dOref_xy((handles.dOi_GPico-
handles.dOi_GMuestras):(handles.dOi_GPico+handles.dOi_GMuestr
as),i));
                        end %for
                    end
                    plot(handles.axOndaref,...
                                    handles.dOref_xy,'k');
            end



set(handles.orden,'String',num2str(handles.dOref_n(:,2)));
            guidata(hObject, handles);
        end
% --- Executes on button press in pbCavidadCerrar.
function pbCavidadCerrar_Callback(hObject, eventdata,
handles)

handles.ifourier=0;
handles.OincOrden=0;
handles.diametro=0;
handles.dOref_xy=0;
set(handles.lbOndaref,'String',[]);
cla(handles.axOndaref);
set(handles.lbCavidad,'String',[]);
set(handles.lbIir,'String',[]);
set(handles.lbOndaref,'String',[]);
set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Min'));
set(handles.tbCavidadDim,'Value',get(handles.tbCavidadDim,'Mi
n'));
set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Min'));
cla(handles.axIir);
cla(handles.axOndaref);
cla(handles.axCavidad);
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all
properties.
function axCavidad_CreateFcn(hObject, eventdata, handles)

 set(hObject,'XLim',[0 handles.DCavDistancia],'YLim',[0
handles.DCavArea])
 handles.txtCavidad=text(0,0,'(0,0)','Visible','off');
 guidata(hObject, handles);


% --- Executes during object creation, after setting all
properties.
function figure1_CreateFcn(hObject, eventdata, handles)

handles.dC_xy=0;
handles.dC_n=1;
handles.diametro=0;
handles.dClb=0;
handles.dOi_xy=0;
handles.axOndaref=0;%%%%%%%%%%%%%%%
handles.dOref_xy=0;
handles.dOi_n=1;
handles.dOiArbitraria=[0,0];
handles.ifourier=0;
handles.OincOrden=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
handles.dOi_GMuestras=0;
handles.dOi_GPico=0;
handles.dOi_a=0;
handles.dOi_w=0;
handles.dOi_p=0;
handles.dOi_b=0;
handles.dOref_a=0;
handles.dOref_w=0;
handles.dOref_p=0;
handles.dOref_b=0;
%%%%%%%%%%%%%%regresa = true;
regresa = true;
    while regresa,
        prompt = {'Distancia en la cavidad[cm]:','Área
trasversal de la cavidad[cm^2]:'};
                    dlg_title = 'Condiciones iniciales
Cavidad';
                    num_lines = 1;
                    def = {'0','0','0','0'};
                    answer =
inputdlg(prompt,dlg_title,num_lines,def);
```

```
handles.DCavDistancia=str2double(answer{1});
                    handles.DCavArea=str2double(answer{2});
          if ((handles.DCavDistancia+handles.DCavArea)>=2)
                    regresa = false;
          end
    end
%%%%%%%%%5%%%
guidata(hObject, handles);

% --- Executes during object creation, after setting all
properties.
function axOndainc_CreateFcn(hObject, eventdata, handles)

regresa = true;
    while regresa,
        prompt = {'Frecuencia en la onda
incidente:','Amplitud en onda incidente:'};
                    dlg_title = 'Condiciones iniciales Onda
incidente';
                    num_lines = 1;
                    def = {'0','0'};
                    answer =
inputdlg(prompt,dlg_title,num_lines,def);
                    DOIncFrecuencia=str2double(answer{1});
                    DOIncAmplitud=str2double(answer{2});
          if ((DOIncFrecuencia+DOIncAmplitud)>=2)
                    regresa = false;
          end
    end
 set(hObject,'XLim',[0 DOIncFrecuencia],'YLim',[0
DOIncAmplitud])
 handles.txtOndainc=text(0,0,'(0,0)','Visible','off');
 guidata(hObject, handles);


% --- Executes on mouse motion over figure - except title and
menu.
function figure1_WindowButtonMotionFcn(hObject, eventdata,
handles)

% --- Executes on mouse press over axes background.
function axCavidad_ButtonDownFcn(hObject, eventdata, handles)
pt = get(hObject,'CurrentPoint');
text(pt(1),pt(3),'°','Color','Red');
handles.dC_xy(handles.dC_n,:)=pt(3);
strtxt = num2str(handles.dC_xy);
```

```
set(handles.lbCavidad,'String',strtxt);
handles.dC_n=handles.dC_n+1;
guidata(hObject, handles);

% --- Executes on mouse press over axes background.
function axOndainc_ButtonDownFcn(hObject, eventdata, handles)


pt = get(hObject,'CurrentPoint');
text(pt(1),pt(3),'°','Color','Blue');
handles.dOi_xy=1;
handles.dOiArbitraria(handles.dOi_n,:)=[pt(1);pt(3)];
strtxt = num2str(handles.dOiArbitraria);
set(handles.lbOndainc,'String',strtxt);
handles.dOi_n=handles.dOi_n+1;
guidata(hObject, handles);


% --- Executes on selection change in puOndaincTipo.
function puOndaincTipo_Callback(hObject, eventdata, handles)

switch get(hObject,'Value');
    case 2
        cla(handles.axOndainc);
        hold on;
        ceros = inputdlg('función delta','Número de ceros');
        handles.dOi_xy=[1 zeros(1,str2double(ceros{1}))];
        cs=handles.dOi_xy;
        stem(handles.axOndainc,handles.dOi_xy)
        guidata(hObject, handles);
    case 3
        %User selects Gaussian wave
        reset(handles.axOndainc);

        handles.dOi_xy=0;
        prompt = {'Minimum frequency of the
bandwidth:','Maximum frequency of the bandwidth:','Division
size of step of frequency','Number of Samples'};
                dlg_title = 'Define Package of Gaussian
waves';
                num_lines = 1;
                def = {'0','0','0','0'};
                answer =
inputdlg(prompt,dlg_title,num_lines,def);
        handles.dOi_Fsmin=str2double(answer{1});
        handles.dOi_Fsmax=str2double(answer{2});
        handles.dOi_division=str2double(answer{3});
```

```
        handles.dOi_numsamples=str2double(answer{4});

Frecmues=str2double(get(handles.edFrecmues,'String'));

handles.dOi_xy=modulated(handles.dOi_Fsmin,handles.dOi_Fsmax,
handles.dOi_division,handles.dOi_numsamples,Frecmues);
                strtxt = num2str(handles.dOi_xy);

set(handles.lbOndainc,'String',strtxt);
                plot(handles.axOndainc,...
                        handles.dOi_xy','k');
                [n,handles.dOi_n]=size(handles.dOi_xy);
        guidata(hObject, handles);

end
% --- Executes during object creation, after setting all
properties.
function puOndaincTipo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on edVelprop and none
of its controls.
function edVelprop_KeyPressFcn(hObject, eventdata, handles)
if get(handles.rbCavidad,'Value') ==
get(handles.rbCavidad,'Max')
    if(strcmp(eventdata.Key,'return'))

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Min'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Max'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);
    end %if
end %if rbcavidad


% --- Executes on key press with focus on edFrecmues and none
of its controls.
function edFrecmues_KeyPressFcn(hObject, eventdata, handles)

if get(handles.rbCavidad,'Value') ==
get(handles.rbCavidad,'Max')
```

```
    if(strcmp(eventdata.Key,'return'))

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Min'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);

set(handles.rbCavidad,'Value',get(handles.rbCavidad,'Max'));
    rbCavidad_Callback(handles.rbCavidad,eventdata,handles);
    end
end %if rbcavidad
if get(handles.rbCavidad,'Value') ==
get(handles.rbCavidad,'Max')
    if(strcmp(eventdata.Key,'return'))

set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Min'));
    rbOndainc_Callback(handles.rbOndainc,eventdata,handles);

set(handles.rbOndainc,'Value',get(handles.rbOndainc,'Max'));
    rbOndainc_Callback(handles.rbOndainc,eventdata,handles);
    end
end

% --- Executes on button press in tbCavidadDim.
function tbCavidadDim_Callback(hObject, eventdata, handles)
button_state = get(hObject,'Value');
    if button_state == get(hObject,'Max')
        set(hObject,'String','2D')
        set(handles.edFrecmues,'Enable','off');
        set(handles.edVelprop,'Enable','off');
        set(handles.rbCavidad,'Enable','off');
        set(handles.pbCavidadCerrar,'Enable','off');
        set(handles.pbCavidadAbrir,'Enable','off');
        set(handles.rbOndainc,'Enable','off');
        set(handles.pbOndaincEliminar,'Enable','off');
        set(handles.pbOndaincAbrir,'Enable','off');
        if get(handles.rbOndainc,'Value') ==
get(handles.rbOndainc,'Max')
            [n1,n2]=size(handles.diametro)
        [X,Y,Z] = cylinder(handles.diametro,n2);
        surf(X,Y,Z)
        hold on;
        axis square
        end %if rbCavidad
    elseif button_state == get(hObject,'Min')
        reset(handles.axCavidad);


Frecmues=str2double(get(handles.edFrecmues,'String'));
```

```
Velprop=str2double(get(handles.edVelprop,'String'));
                    lon=(Velprop.*100)./(2.*Frecmues);

x4=linspace(0,length(handles.diametro)*lon,length(handles.dia
metro));%.5754
                    stairs
(handles.axCavidad,x4,handles.diametro,'k','LineWidth',3,'But
tonDownFcn','down_Cavity=1;');
        hold on;
        set(hObject,'String','3D')
        set(handles.edFrecmues,'Enable','on');
        set(handles.edVelprop,'Enable','on');
        set(handles.rbCavidad,'Enable','on');
        set(handles.pbCavidadAbrir,'Enable','on');
        set(handles.rbOndainc,'Enable','on');
        set(handles.pbOndaincEliminar,'Enable','on');
        set(handles.pbOndaincAbrir,'Enable','on');
        set(handles.pbCavidadCerrar,'Enable','on');
    end


% --- Executes on selection change in lbCavidad.
function lbCavidad_Callback(hObject, eventdata, handles)

% --- Executes during object deletion, before destroying
properties.
function axCavidad_DeleteFcn(hObject, eventdata, handles)


% --- Executes during object creation, after setting all
properties.
function lbCavidad_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all
properties.
function rbOndainc_CreateFcn(hObject, eventdata, handles)


% --- Executes on mouse press over axes background.
function axOndaref_ButtonDownFcn(hObject, eventdata, handles)
```

```
% --- Executes on selection change in listbox5.
function listbox5_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function listbox5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% ------------------------------------------------------------
---------
function uipushtool5_ClickedCallback(hObject, eventdata,
handles)
% ------------------------------------------------------------
---------
function uipushtool4_ClickedCallback(hObject, eventdata,
handles)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muestras=handles.dOi_GMuestras;
pico=handles.dOi_GPico;
dOi_a=handles.dOi_a;
dOi_w=handles.dOi_w;
dOi_p=handles.dOi_p;
dOi_b=handles.dOi_b;
dOref_a=handles.dOref_a;
dOref_w=handles.dOref_w;
dOref_p=handles.dOref_p;
dOref_b=handles.dOref_b;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ifourier=handles.ifourier;
OndaIncidente=handles.dOi_xy;
OndaReflejada=handles.dOref_xy;
Orden=handles.OincOrden;
diametro=handles.diametro;
FolderName = inputdlg('Nombre del proyecto','Input Data');
[stat,struc] = fileattrib;
 PathCurrent = struc.Name;
 PathFolder=[PathCurrent '\PI' FolderName{1} '\']
 mkdir([PathFolder]);
%%save [PathFolder 'convolucionOinc'];
 save ([PathFolder
FolderName{1}],'OndaReflejada','OndaIncidente','ifourier','Or
```

```
den','diametro','muestras','pico','dOi_a','dOi_w','dOi_p','dO
i_b','dOref_a','dOref_w','dOref_p','dOref_b');


% --- Executes during object creation, after setting all
properties.
function pbCavidadAbrir_CreateFcn(hObject, eventdata,
handles)
```

6.4 Fitsine.m

```
function [b,a,w,p] = fitsine(y)
% [b,a,w,p] = fitsine(y) fits a sinusoidal function
%             of the form b + a*sin(w*n + p) to the
%             data vector y using least squares.


% Estimate and remove bias
b = (max(y) + min(y))/2;
y = y - b;

% Estimate amplitude
a = (max(y) - min(y))/2;

% Linearize the model computing the arcsine
% and unwrapping the angle
asy = asin(y/a);
TOL = -0.002;
ofs = pi;
while any(diff(asy) < TOL)
   inx = find(diff(asy) < TOL);
   asy(inx(1):end) = ofs - asy(inx(1):end);
   ofs = 2*pi + ofs;
end;

% Estimate frequency and phase through least squares
params = polyfit([0:length(asy)-1]',asy,1);
w = params(1);
p = params(2);
```

6.5 Gen_sinxexp.m

```
function sig=gen_sinxexp(fs,fd,n,k)
%fd=100 % frequency desired
%k is a vector which represent the numeber of samples of the
signal
%n=4    % number of cycles inside the Gaussian
%SIN(2*3.1416*fd*TIME)*EXP(-18*(TIME-4/fd)^2/(4/fd)^2)
%2*n/fd % total length to show

Td=1/fd;
sig=sin(2*pi*fd*k/fs).*exp(-18*(k/fs-n*Td).^2/(n*Td)^2);
```

6.6 iFourier.m

```
function [ifourierRafB3] = iFourier(Fs,ACompRaf,PCompRaf)
%function [b,a,w,p] = iFourier(Fs,ACompRaf,PCompRaf)

%Pruebas

%clear all
%clc
%señales w*1801/pi
%load Compensacion

Amp1=[0 ACompRaf];
Fas1=[0 PCompRaf];

%Fs=100000;
%50 es la division entre cada frecuencia
intervalos=(Fs/2)./50;
Amp11=zeros(1,intervalos-length(Amp1));

Amp1=[Amp1 Amp11];
Fas1=[Fas1 Amp11];

[n3,m3]=size(Amp1);

cond=2;
if cond==1
    Amp2=fliplr(Amp1(1,2:m3));
    Fas2=(fliplr(Fas1(1,2:m3)))*(-1);
else
    Fas1(1,m3)=0;
    Amp2=fliplr(Amp1(1,2:m3-1));
```

```
    Fas2=(fliplr(Fas1(1,2:m3-1)))*(-1);
end

Amp3=[Amp1, Amp2];
Fas3=[Fas1, Fas2];

[n1,m1]=size(Amp3);
fourier=zeros(1,m1);

for u=1:m1
    Amp4=Amp3(1,u)*cos(Fas3(1,u));
    Fas4=Amp3(1,u)*sin(Fas3(1,u));
    fourier(1,u)=Amp4+(Fas4*j);
end

fourier;
ifourierRafB3=ifft(fourier);
end
```

6.7 Iir.m

```
function [coefi,iir1]=iir(a,q);

% iir1=respuesta impulso del sistema
% coefi=coeficietes de reflexion
% area=es el vector-renglon que representa el area de los i-
segmentos que
%  q=sampling frequency
% forman la cavidad cilindrica a evaluar
%
%  [coefi,iir1]=iir(area);
%
% este programa calcula los valores de la respuesta impulso
% entrada a partir de los i-segmentos que componen el
cilindro
% acustico
%
iir(I,z)=iir[0T]+iir[1T]z+iir[2T](z.^2)+iir[3T](z.^3)+.......
...
%
%
area=a';

[n,m]=size(area);
coefi=zeros (n,m-1);
```

```
iir1=zeros(n,m-1);
%
% se obtienen los coeficientes de reflexión a partir de la
% formula recursiva
%                   r(i,i+1)=(S(i)-S(i+1))/(S(i)+S(i+1))
% donde
%      r(i,i+1) son los coeficientes de reflexion
%      S es el area del segmento i que compone al cilindro
acustico
%
%
for i=1:m-1
   coefi(1,i)=(area(1,i)-
area(1,i+1))/(area(1,i)+area(1,i+1));
end
coefi;
%
%   Ware y Aki encontraron una formula recursiva para obtener
los coeficientes
%   de reflexion
%
%   r(0,1)= iir[0T]
%   r(1,2)= iir[1T]/(1-(r(0,1)^2))
despejo iir[1T]
%   r(1,3)= (iir[2T]+(r(0,1)*r(1,2)*(iir[1T])) / ((1-
r(0,1)^2)(1-r(1,2)^2))   despejo iir[2T]
%   .....
%
%   esta es una formula recursiva en donde se despeja el
coeficiente de la
%   respuesta impulso de entrada
%
%


%
% esta parte del programa calcula el denominador de la
formula recursiva
%

for i=1:m-1
   denominador(1,i)=(1-(coefi(1,i))^2);
end
denominador;
denominador3=zeros(1,m-1);
denominador3(1,1)=denominador(1,1);

for i=1:m-2
```

```
   denominador3(1,i+1)=denominador3(1,i)*denominador(1,i+1);
end
denominador3;


%
% esta parte del programa calcula los tres primeros
coeficientes de la
% respuesta impulso de entrada


iir1(1,1)=coefi(1,1);
iir1(1,2)=coefi(1,2)*(1-(coefi(1,1)^2));

A0=1;
B0=coefi(1,1);

m1=2;
a=A0;
b=coefi(1,m1).*B0;
c=coefi(1,m1).*A0;
d=B0;

A1=[a b];
B1=[c d];
C1=A1(1,2)*iir1(1,2);
denominador=(1-(coefi(1,1))^2).*(1-(coefi(1,2))^2);
iir1(1,3)=(coefi(1,3)*denominador)-(C1);

%
% se calcula el numerador de la formula recursiva que es
restado
% a su vez por el denominador para obtener la
iir[3T],iir[4T],....

for k=4:m-1
   a=A1;
   a=[a,0];
   b=coefi(1,k-1).*B1;
   b=[0,b];
   A2=a+b;

   c=coefi(1,k-1).*A1;
   c=[c,0];
   d=B1;
   d=[0,d];
   B2=c+d;
```

```
   A1=A2;
   B1=B2;

   C2=iir1(2:k-1);
   C2=fliplr(C2);
   C2=[0 C2];
   numerador=sum(A1.*C2);
   iir1(1,k)=(coefi(1,k)*denominador3(1,k-1))-(numerador);
end
%iir1

%
% aqui se presenta la respuesta impulso entrada
```

6.8 Modulated.m

```
%division=50;
%Fsmin frecuencia miníma del ancho de banda a analizar
%división tamaño de paso en frecuencia **Multiplos de
%50**
%Fsmax frecuencia max del ancho de banda a analizar
%q frecuencia de muestreo
%x numero de muestras de cada señal
function [inc]=modulated(Fsmin,Fsmax,division,x,q)
global n_FunIncident;
global xy_FunIncident;
freqma=Fsmin:division:Fsmax;%9950;
inc=[];
x1=0:x;
hwait = waitbar(0,'Please wait...');
                    steps=length(freqma);
%matlabpool open local 2
    %parfor k1=1:length(freqma)
    for k1=1:length(freqma)

inc(k1,:)=gen_sinxexp(q,freqma(1,k1),freqma(1,k1)/50,x1);
        %50 is used to keep the number of cycles for each
frequency
        waitbar(k1 / steps);
    end
%matlabpool close;
close(hwait)
end %function
```

6.9 Svd6.m

```
function TT4=svd6(e2,e3,orden);

% TT4 respuesta impulso del sistema
% e2 = señal senoidal generada (entrada)
% e3 = convolucion de la onda acustica incidente con la
respuesta impulso del sistema (salida)
% orden=orden del filro
%     TT4=svd6(e2,e3,orden);

% tomo la entrada y saco la matriz triangular superior
% tomo esa matriz superior y aplico SVD
p=(tril(toeplitz(e2)));

%selecciono el orden del filtro, cortando la matriz Toeplitz
T1=p(:,1:orden);
[n,m]=size(T1);%

%tomo el SVD de la matriz anterior
% u son los eigenvectores de TT*
% v son los eigenvectores de T*T
% s son los eigenvalores de la matriz diagonal
[u,s,v]=svd(T1);
diag(s);
s1=sum(diag(s));
%se elige una posicion de los m-eigenvectores
k1=1;%input ('introduce la posición inicial de eigenvalor
deseado');
k2=orden;%input ('introduce la posición final de eigenvalor
deseado');

T2=zeros(m,n);
for i=k1:k2;
   TT1=(1/s(i,i))*v(:,i)*u(:,i)';
   T2=TT1+T2;
end

%hace la deconvolucion cortando el vector de salida
TT4=(T2*e3')';

% [n,m]=size(TT4);
% t=linspace(0,m/Fs,m);
% figure
% plot (t,TT4)
% title('impulse response determined by SVD')
```

```
% xlabel('time [sec]')
% ylabel('arbitrary units')
%



% Una vez deconvolucionado se realiza la reconstruccion
usando Ware & Aki
%[f1,f2,f3]=wasp(TT4);
```

6.10 Wasp.m

```
function [coef,diametro,diametro1]=wasp(iir,q);

%,area);

% [coef,diametro,diemtro1]=wasp (iir)
%
% coef son los coeficientes de reflexion
% diametro es la pared superior
% diametro1 es la pared inferior
% iir es la respuesta impulso a ser examinada
%
% este programa calcula el area de cada segmento apartir de
una respuesta impulso,
% esta iir es la señal que se adquiere en el microfono, en la
cual se separa la señal
% incidente de la señal reflejada.
%
% la longitud de cada segmento esta dado por l=cT/2, donde
T=1/F
%  c es la velocidad de sonido en el aire
%  F es la frecuencia de muestreo
%  T es el tiempo de separacion entre de cada iir
%



%se genera un area de seccion de cruce variabl3
%a1=[12 12 12];% 12 12 12 12 12 12 12];% 12 12 12 12 12 12 12
12 12 12 12 12 12 12 12 12 12];
%a2=[19 19 19];% 19 19 19 19 19 19 19];% 19 19 19 19 19 19 19
19 19 19 19 19 19 19 19 19 19];
%a3=[27 27 27];% 27 27 27 27 27 27 27];% 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27 27];
```

```
%area=[a3 a2 a1 a2 a3 a2 a1 a2 a3 a2 a1 a2 a3 a2 a1];

%area=[12 12 19 19 27 27 19 19 19 12 12 12 12 19 19 19 19 19
27 27 27 27 19 19 19 12 12 19 19 27 27];


%area=[a3 a2 a1];% a2 a3 a2 a1 a2 a3];
%area=[a1 a2 a3 a2 a3 a2 a1 a2 a3 a2 a1];

%a1=[12 12 12];% 12 12 12 12];% 12 12 12 12 12 12 12 12 12 12
12 12 12 12 12 12 12 12 12];
%a2=[19 19 19];% 19 19 19 19];% 19 19 19 19 19 19 19 19 19 19
19 19 19 19 19 19 19 19 19];
%a3=[27 27 27];% 27 27 27 27];% 27 27 27 27 27 27 27 27 27 27
27 27 27 27 27 27 27 27 27];
%area=[a3 a2 a1 a2 a3 a2 a1 a2 a3 a2 a1 a2 a3 a2 a1];
%area=[a3 a2 a1];% a2 a3 a2 a1 a2 a3];
%area=[a1 a2 a3 a2 a3 a2 a1 a2 a3 a2 a1];
% a2 a3 a2 a1 a2 a3 a2 a1];

%iir=[-0.2258 -0.1650 0.1664 0.1911 -0.1960];
%iir=[-0.2258 -0.1650 0.1665 0.1935 -0.1962 -0.1074 0.1112
0.1185 -0.1224 -0.0274 0.0316 0.0424];
%iir=iir1;
%iir=iir1;
[n,m]=size(iir);
coef=zeros(1,m);

%  aqui se define los tres primeros coeficientes de reflexion

r01=iir(1,1);
r12=iir(1,2)/(1-(r01^2));
coef(1,1)=r01;
coef(1,2)=r12;

A0=1;
B0=coef(1,1);

m1=2;
a=A0;
b=coef(1,m1).*B0;
c=coef(1,m1).*A0;
d=B0;
A1=[a b];
B1=[c d];
C1=iir(2:m1+1);
C1=fliplr(C1);
```

```
numerador=sum(A1.*C1);
denominador=(1-(coef(1,1))^2).*(1-(coef(1,2))^2);
coef(1,3)=numerador/denominador;

%
%   Ware y Aki encontraron una formula recursiva para obtener
los coeficientes
%   de reflexion
%
%   r(0,1)= iir[0T]
%   r(1,2)= iir[1T]/(1-(r(0,1)^2))
%   r(1,3)= (iir[2T]+(r(0,1)*r(1,2)*(iir[1T])) / ((1-
r(0,1)^2)(1-r(1,2)^2))
%   .....
%
%   con los ceficientes de reflexion se usa la formula
recursiva
%
%                       S(i+1)=Si * (1-r(i,i+1)) /
(1+(r(i,i+1))
%
%   para obtener el area del siguiente segmento, tomando como
condicion inical que
%   se conoce el area del primer segmento
%


%
% aqui se calculan los coeficientes de reflexion de iir[3T],
iir[4T], iir[5T],.....
%
for k=4:m
   a=A1;
   a=[a,0];
   b=coef(1,k-1).*B1;
   b=[0,b];
   A2=a+b;

   c=coef(1,k-1).*A1;
   c=[c,0];
   d=B1;
   d=[0,d];
   B2=c+d;
   A1=A2;
   B1=B2;
   C2=iir(2:k);
   C2=fliplr(C2);
```

```
   numerador=sum(A1.*C2);
   for i=1:k
      denominador(1,i)=(1-(coef(1,i))^2);
   end
   denominador2=denominador;
   denominador3=zeros(1,k);
   denominador3(1,1)=denominador(1,1);
   for i=1:k-1

denominador3(1,i+1)=denominador3(1,i)*denominador2(1,i+1);
   end
   denominador3;
   coef(1,k)=numerador/denominador3(1,k);
end

%
% este el el vector de los coeficientes de reflexion que sera
usado para calcular
% el area del segmento cilindrico
%
coef;

%
% aqui se introduce la condicion inicial que es el area del
primer segmento
%
[n2,m2]=size(coef);
s=zeros(1,m2);


%introducimos el primer elemento del area como condicion
inical
s1=
0.0536;%1.1;%0.9852+0.12;%area(1,1);%%0.9852%+0.12;%0.05;%+0.
12;%0.13;%+0.0704;%+0.15;%0.0704;%The dc component is added
when the fisrt cylidnricalcavity is 1.14 instead of 1.1
;%input ('introduce el area del primer segmento........');
s(1,1)=s1;


%
%calcula el calibre del segmento cilindrico en forma
recursiva
%
for g=1:m2
   x=s(1,g)*(1-coef(1,g));
   s(1,g+1)=x/(1+coef(1,g));
```

```
end
s;


%
%   en principio asi se representarian los reultados donde
indica elarea de cada segmento
%   como un impulso
%
%figure
%subplot (223)
%plot (coef)
%title('coeficientes de reflexion')

%figure
diametro=s;%./2
diametro1=diametro*(-1);
%subplot (224)

diametro=diametro(1,1:end);
diametro1=diametro*(-1);

%sampling frequency
%q=100000;
lon=(345.28.*100)./(2.*q);%0.5754;

% qr=100000;%2000;%input('sampling frequency used to estimate
the cavity.....');%2000;
% lonr=(345.28.*100)./(2.*qr);
%
%
%
x4=linspace(0,length(diametro).*lon,length(diametro));%.5754
% plot(x4,diametro,'k')
% hold on
% %stairs(x4,diametro1,'g')
% title ('Comparing both area')
% xlabel('Distance [ cm ]')
% ylabel('Area [ cm^2 ]')

diametro2=0;
diametro2=diametro2+diametro;

%x3=linspace(0,length(area).*lon,length(area));
%%%hold on; stairs(x3,-area/2,'r')
%hold on; plot(x3,(area),'-r');drawnow;
```

BIBLIOGRAFÍA

[1]Xuesong Wang, Gordon Short and Karl Dawson, "Acoustic reflectometry for gas pipelines – Monitoring features in gas pipelines using acoustek ". Department of Acoustic, The University of Manchester, UK, 2010.

[2] A. H. Benade and J. H. Smith, "Brass wind instrument impulse response measurements". J. Acoust. Soc. Am., Vol. 70, S22, 1981.

[3] A. P. Watson and J. M. Bowsher, "Impulse measurements on brass musical instruments.Austica".Acustica, Vol. 66, no. 3, pp. 170-174, 1988.

[4] A. C. Jackson and D. E. Olson, "Comparison of direct and acoustical area measurements in physical models of human central airways". J. Appl. Physiol., Vol. 48, no. 5, pp. 896-902, 1980.

[5] J. J. Fredberg, M. E. Wohl, G. M. Glass, and H. L. Dorkin, "Airway area by acoustic reflections measured at the mouth". J. Appl. Physiol., Vol. 48, no. 5, pp. 749-758, 1980.

[6] L. J. Brooks, R. G. Castile, G. M. Glass, N. T. Griscom, M. E. Wohl, and J. J. Fredberg, "Reproducibility and accuracy of airway area by acoustic reflection". J. Appl. Physiol., Vol. 57, no. 3, pp. 777-787, 1984.

[7] Ernesto Rodrigo Vázquez Cerón, "Reconstrucción de cavidades con sección trasversal variable mediante reflectometría acústica", M. MasteryThesis, Universidad Autónoma Metropolitana, Unidad Iztapalapa, Mexico City, July, 2002.

[8] MathWorks, «Product Documentation,» [En línea]. Available: http://www.mathworks.com/help/matlab/index.html. [Último acceso: 01 08 2012].

[9] A. D. Durzo, I. Rubinstein, V. G. Lawson, K. P. Vassal, A. S. Rebuck, A. S. Slutsky, and V. Hoffstein, Comparison of glottis areas measured by acoustic reflections vs. computerized tomography. J. Appl. Physiol., Vol. 64, no. 1, pp. 367-370, 1988.

[10] "Time Domain Reflectometry Theory", Application Note 1304-2, Hewlett- Packard Company, U.S.A., 1998