Escuela Politécnica Superior de Linares

Grado en Ingeniería Mecánica

Universidad de Jaén
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CREACIÓN DE APLICACIÓN GRÁFICA CON MATLAB PARA EL ANÁLISIS DE UNA BOMBA CENTRÍFUGA

Alumno: Galiano Martínez, Andrés Jesús

Tutor: Prof. D. Mario Miró Barnés
Depto.: Ingeniería Mecánica y Minera

Junio del 2014

Universidad de Jaén
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

Curso 2013 - 2014

CREACIÓN DE APLICACIÓN GRÁFICA CON MATLAB PARA EL ANÁLISIS DE UNA BOMBA CENTRÍFUGA

Vº Bº **Vº Bº**

Alumno: Galiano Martínez, Andrés Jesús

Tutor: Prof. D. Mario Miró Barnés

Linares, 23 de Junio del 2014

INDICE

1.- Resumen 5

2.- Introducción 5

3.- Objetivos. 6

4.- Aplicaciones gráficas en Matlab. 6

4.1.- El entorno GUIDE. 7

4.1.1.- Organización del entrono GUIDE. 9

4.1.2.- Property Inspector 10

4.2.- Estructura de las GUIs en Matlab 12

4.2.1.- Los handles y las funciones Get y Set. 13

4.3.- Matlab compiler 15

5.- Clasificación de las máquinas elevadoras de líquidos. 17

5.1.- Bombas gravimétricas 18

5.2.- Bombas volumétricas. 18

5.2.1.- Bombas Volumétricas alternativas. 19

5.2.1.- Bombas Volumétricas rotativas. 20

5.3.- Bombas rotodinámicas 21

5.3.1.- Bombas hélice. 21

5.3.2.- Bombas centrífugas 22

5.3.3.- Bombas Helicocentrífugas. 26

5.3.4.- Velocidad específica. 27

6.- Descripción del banco de ensayo. 28

6.1.- Descripción de los sensores 30

6.2.- Toma de datos. 31

7.- Estructura del programa. 34

7.1.- Importación de datos. 35

7.2.- Pestaña Curvas dimensionales. 38

7.2.1.- Gráficas en dos dimensiones. 39

7.2.2.- Gráficas en tres dimensiones. 41
7.3.- Pestaña curvas adimensionales. 42
7.4.- Pestaña semejanza. 46
7.5.- Exportar gráficas. 49
8.- Conclusiones y futuras líneas de trabajo. 51
9.- Glosario de términos. 54
10.- Bibliográfia. 55
10.1.- Páginas webs visitadas: 56
11.- Anexos a la memoria. 57
11.1.- Anexo 1. Ayuda del programa. 57
11.1.1.- Importar datos. 57
11.1.2.- Unidades. 61
11.1.3.- Orden de ajuste. 62
11.1.4.- Curvas características. 62
11.1.5.- Seleccionar valores a representar. 64
11.1.6.- Representación de contornos de Isorendimiento. 65
11.1.7.- Representación de contornos de Isoconsumo. 67
11.1.8.- Representación de curvas en 3-D. 69
11.1.9.- Representación de grupos adimensionales. 71
11.1.10.- Obtener curvas por semejanza. 72
11.1.11.- Comparar puntos experimentales con curvas obtenidas por semejanza. 73
11.1.12.-Exportación de gráficas. 75
11.2.- Anexo 2. Código del programa. 77
11.2.1.- Función “Cargar_datos” 77
11.2.2.- Código de la GUI “dialgtable” 78
11.2.3.- Código de la GUI “dialginputimagen” 80
11.2.4.- Código de la GUI principal “Caracterizador de bombas centrífugas”. ..84

1.- RESUMEN

Este trabajo fin de grado consiste en la creación de una aplicación gráfica, en el entorno de programación de Matlab, para el análisis y caracterizado de bombas centrifugas.

Desde un principio el diseño de la interfaz gráfica persigue fines tanto docentes como prácticos por lo que además de las típicas curvas características utilizadas en la selección de bombas centrífugas, nos permite representar los grupos adimensionales de relevancia en el estudio de bombas y predecir el comportamiento de una bomba semejante trabajando en un régimen de giro y con un diámetro de rodete distinto del ensayado.

Para dar mayor versatilidad a la aplicación se le confiere la capacidad de importar datos tanto de ficheros "*.DAT", que son los generados por el software de adquisición de datos del banco de ensayo del que disponemos en la Escuela, como de ficheros "*.XLS", con los que se abre la posibilidad al estudio de otras bombas.

2.- INTRODUCCIÓN

Para seleccionar el equipo de bombeo más adecuado para una instalación y predecir el comportamiento que este va a tener, es necesario disponer de las curvas características de las distintas bombas que existen en el mercado. Estas curvas definen el comportamiento hidráulico de una bomba relacionando el caudal con el resto de parámetros de interés como son la altura manométrica, el rendimiento hidráulico, la potencia consumida, o la altura útil de aspiración.

En el presente trabajo fin de grado se desarrollará una interfaz gráfica de usuario (Graphical User Interface → GUI) denominada "Caracterizador de bombas Centrifugas", que permita analizar y obtener las curvas características de este tipo de máquinas hidráulicas de forma rápida y sencilla.

Para su programación el departamento de Ingeniería Mecánica y Minera ha elegido el entorno de programación de Matlab, ampliamente extendido y utilizado en investigación e ingeniería y que permite la generación de aplicaciones gráficas ejecutables en ordenadores sin la necesidad de que estos tengan instalado Matlab, y sobre todo sin la necesidad de que estos dispongan de una licencia de Matlab.

En este documento se realizará una introducción al entorno de generación de aplicaciones de Matlab, se describirán los conceptos básicos en el estudio de bombas rotodinámicas, y se profundizará en la estructura de la interfaz gráfica objeto de este proyecto, describiendo las partes más importantes de su código y las posibles modificaciones que se podrían incluir para aumentar su potencial y rendimiento.

3.- OBJETIVOS.

Generar una aplicación gráfica para la caracterización de bombas centrifugas los suficientemente intuitiva como para que, aquellas personas que no estén familiarizadas con los lenguajes de programación sean capaces de utilizarla. Y con un código abierto que permita a usuarios familiarizados con el entorno de programación de Matlab adaptarlo a sus necesidades.

Conferir a aquellos alumnos que cursen asignaturas de máquinas hidráulicas una herramienta rápida para conocer el comportamiento de una bomba centrifuga en los distintos escenarios de trabajo que se le pueden plantear y comprobar que el comportamiento de las bombas centrífugas se asemeja con bastante fidelidad a lo estudiado en el aula.

4.- APLICACIONES GRÁFICAS EN MATLAB.

Matlab (termino procedente de la unión de las primeras letras de “MATrix LABoratory”) es un programa de uso ampliamente extendido en ingeniería e investigación, que permite realizar una gran variedad de cálculos matemáticos y técnicos.

Matlab, además de contribuir un entorno de programación con su propio lenguaje que permite el desarrollo de funciones y algoritmos para la manipulación, visualización y representación de datos, puede ser utilizado para el desarrollo de aplicaciones graficas como la desarrollada en este trabajo.

En este capítulo se describirá cómo se estructuran y desarrollan las GUIs usando el entorno GUIDE de Matlab. No obstante, su finalidad es la de transmitir al lector una serie de conceptos básico que le permitan entender el funcionamiento del programa que se desarrolla en este trabajo fin de grado, tratando de no profundizar más de lo necesario.

4.1.- El entorno GUIDE.

Es posible desarrollar GUIs en Matlab usando solo las funciones “UIcontrol” y “UIMenu”. Con estas órdenes el usuario puede crear una interfaz gráfica usando cualquiera de los componentes para GUIs disponibles en Matlab, únicamente escribiendo el código en un fichero “*.m”. Este era en las primeras versiones de Matlab, el único camino para el desarrollo de una GUI, sin embargo, en las versiones de más recientes se ha integrado el entorno GUIDE (Graphical User Interface Development Environment → Entorno para el desarrollo de interfaces gráficas de usuario).

Entre las ventajas del uso del entorno GUIDE podemos destacar:

- **Menor tiempo requerido para el desarrollo de GUIs**, sobre todo en aquellas que poseen un grado de complejidad considerable.
- **Mayor facilidad para el diseño.** La distribución de los distintos elementos de control (botones, menús desplegables,…) de una GUI mediante su colocación con el ratón en una presentación gráfica es mucho más rápido e intuitivo que especificarlo mediante coordenadas en el código del programa.
- **Generación automática de parte del código**. El entorno GUIDE genera la función principal del programa e introduce las funciones “callback” de los distintos elementos de control, permitiendo al programador centras sus esfuerzos en el diseño de las funciones que se ejecutarán cuando el usuario interacciones con ellos.

El programa fruto del presente trabajo se ha desarrollado utilizando este entorno, así que de aquí en adelante, en esta memoria no se hará referencia a la metodología para el desarrollo de GUIs sin el uso del entorno GUIDE.

Existen varias formas de acceder al entorno GUIDE de Matlab. Escribiendo en la “Command Windows” de Matlab la palabra “guide” y pulsado la tecla intro, en la barra de menús en “File → New → GUI” (ver figura 4.1), o haciendo clic en el icono especifico situado en la barra de herramientas (ver figura 4.2).

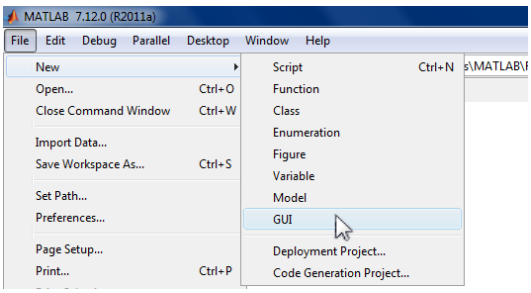


Figura 4.1

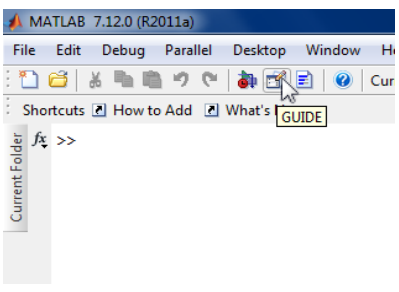


Figura 4.2

Indiferentemente del método utilizado para acceder al entorno, se nos abrirá una ventana como la que se muestra en la figura 4.3. En esta se nos da la posibilidad de elegir cuatro plantillas diferentes para el inicio de nuestra GUI:

- **Blank GUI (Default):** es la utilizada por defecto y viene totalmente en blanco.
- **GUI with Uicontrols:** es una GUI de ejemplo con varios textos y botones ya introducidos.
- **GUI with Axes and Menu:** es una GUI de ejemplo con una gráfica y un menú desplegable ya introducidos.
- **Modal Question Dialog:** es un ejemplo de GUI, del tipo utilizado en las ventanas emergentes, que viene con una pregunta y los botones para responder afirmativa o negativamente ya introducidos.

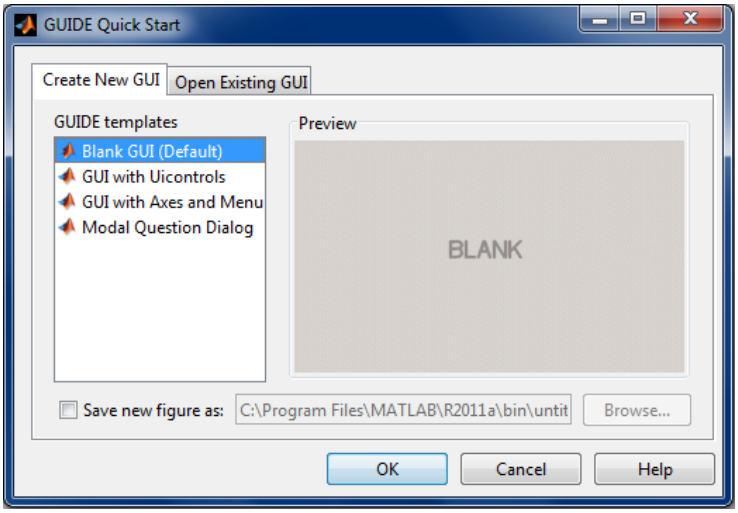


Figura 4.3

Eligiendo la opción marcada por defecto y haciendo clic en “OK” accedemos al entorno GUIDE, mostrado en la figura 4.4

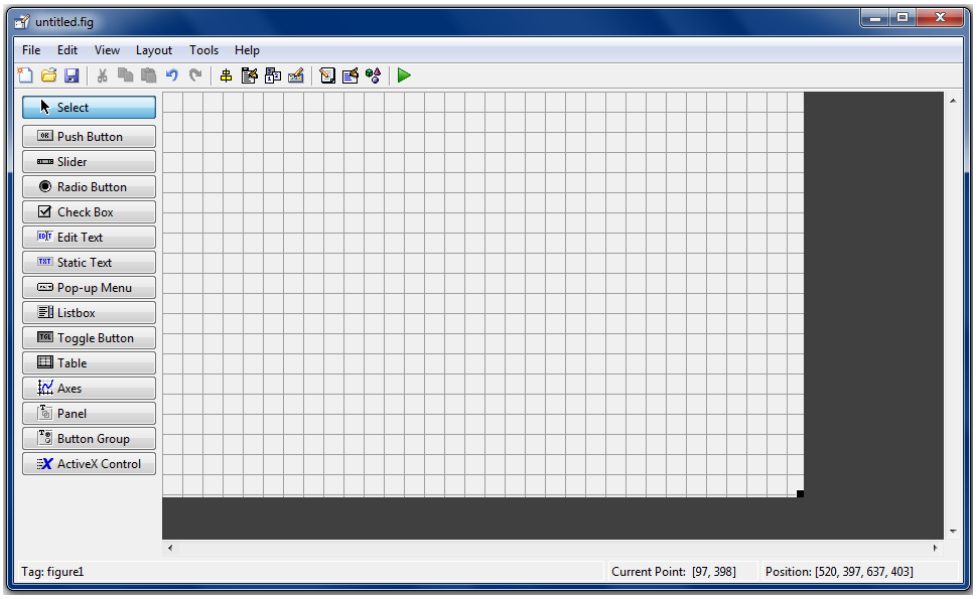


Figura 4.4

4.1.1.- Organización del entrono GUIDE.

Las zonas más importantes del entorno GUIDE (figura 4.4) son:

- La zona de trabajo: es una representación de lo que se verá en pantalla al ejecutar la GUI a falta de la barra de menús y de la barra de título (ver figuras 4.5 y 4.6). Es en esta área donde se distribuyen los distintos elementos de control que forman la GUI.

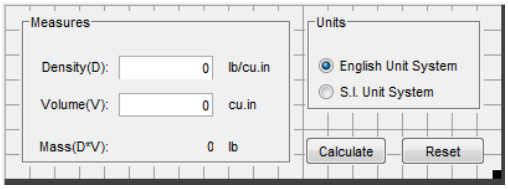


Figura 4.5 Zona de trabajo

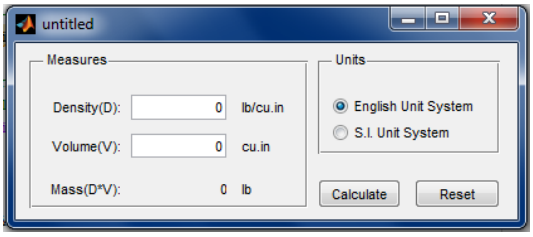


Figura 4.6

- Paleta de componentes: Muestra los diferentes controles que Matlab permite introducir en una interfaz gráfica (Figura 4.7). Para introducir uno de ellos basta con arrástralo a la zona de trabajo.

Select	Herramienta de selección
Push Button	Botón de presión
Slider	Barra de deslizamiento,
Radio Button	Botón circular, como un botón de presión con enclavamiento
Check Box	Casilla de verificación, similar a botón circular
Edit Text	Texto editable, permite al usuario de la GUI introducir valores
Static Text	Texto estático, proporciona información o resultados.
Pop-up Menu	Menú desplegable, permite elegir una opción de entre varias
Listbox	Lista, muestra al usuario una lista de variables
Toggle Button	Botón de presión con enclavamiento
Table	Tabla, con datos que el usuario puede ver y modificar.
Axes	Axes, en estos se representan las imágenes y gráficas
Panel	Panel, sirve para organizar un grupo de botones
Button Group	Grupo de botones circulares, mutuamente exclusivos.
ActiveX Control	Control ActiveX, sirve para introducir videos, …

Tabla 4.1 Elementos de la paleta de componentes

- Barra de Herramientas: En ella se encuentran los botones mostrados en la tabla 4.2.

	**New Figure:** crea una nueva interfaz gráfica.
	**Open Figure:** Abre una GUI ya existente
	**Save Figure:** guarda la GUI con la que se está trabajando.
	**Cut:** Corta el objeto seleccionado
	**Copy:** Copia el objeto seleccionado
	**Paste:** Pega el objeto que se halle en el portapapeles
	**Undo:** Deshace la última acción realizada
	**Redo:** Rehace la última acción deshecha
	**Align Objects:** Sirve para alinear los objetos introducidos entre si
	**Menu Editor:** Abre una ventana para crear la barra de menús
	**Tab Order Editor:** ordena los objetos que se superponen entre si
	**Toolbar Editor:** Abre una ventana para crear la barra de herramientas
	**Editor:** Abre una ventana que permite editar el código de la GUI
	**Property Inspector:** permite modificar las propiedades de cada objeto.
	**Object Browser:** Muestra la vista en árbol de todos los objetos de la GUI
	**Run Figure:** Ejecuta la GUI

Tabla 4.2 Elementos de la barra de herramientas

4.1.2.- Property Inspector.

El “Property Inspector” es la herramienta que nos permite modificar las propiedades de cada uno de los objetos insertados. Como puede verse en la figura 4.7 está compuesto por una tabla con dos columnas. En la columna de la izquierda figura el nombre de las propiedades del objeto que hayamos seleccionado, mientras que a la derecha de esta tenemos el valor o característica asignada.

Las propiedades que nos muestre y permita modificar dependerán del objeto que hayamos seleccionado (Push Button, Slider, etc.), no obstante existen algunas propiedades comunes a todos los elementos y que por su importancia para la comprensión del funcionamiento del código de una GUI voy a citar:

- **Tag:** es el nombre que se le asigna al objeto en cuestión. Matlab asigna uno por defecto sin embargo es recomendable darle a este un nombre que

describa el significado del elemento, ya que este valor se utilizará en el código del programa para obtener y/o modificar la información del objeto.

- **String:** es el nombre que aparecerá en el objeto al ejecutar la GUI, en el caso de tratarse de un menú desplegable, tendrá una fila por cada fila que deba de aparecer en el menú.
- **BackgroundColor:** es el color de fondo del elemento.
- **Visible:** permite dar o quitar la visibilidad a un objeto. Se utiliza para hacer desaparecer y aparecer objetos cuando así lo requiera el funcionamiento de la GUI.

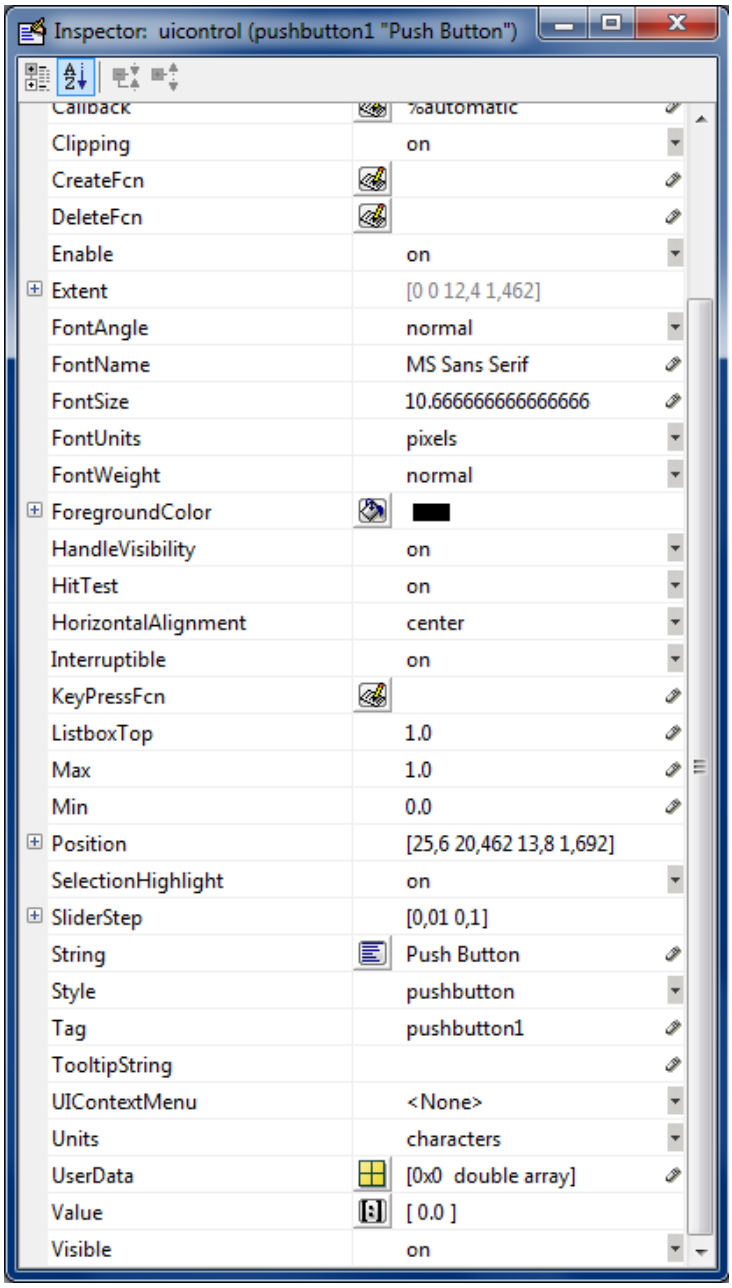


Figura 4.7

4.2.- Estructura de las GUIs en Matlab

Las GUIs desarrolladas en Matlab mediante el entorno GUIDE constan de dos ficheros, uno de extensión "*.fig" y otro de extensión "*.m"

El fichero ".fig", es un fichero binario con toda la información referente a la ventana grafica de la GUI. En el figuran todos los objetos insertado en la zona de trabajo y las propiedades que se le han asignado (entre las que se incluyen la posición que va a ocupar en la venta de la GUI, el color del objeto, etc.). Este fichero se encarga de actualizarlo el propio entorno GUIDE cada vez que guardamos o ejecutamos la GUI, por lo que el programador no ha de manipularlo.

El fichero "*.m", es un fichero de texto que contiene el código de la GUI. Su presentación puede recordar a los típicos "Script" utilizados en Matlab, sin embargo su forma de trabajo es muy distinta. En un "Script" el código se ejecuta de arriba abajo, mientras que el código de una GUI está formado por funciones (ver figura 4.8), que funcionan de la siguiente manera:

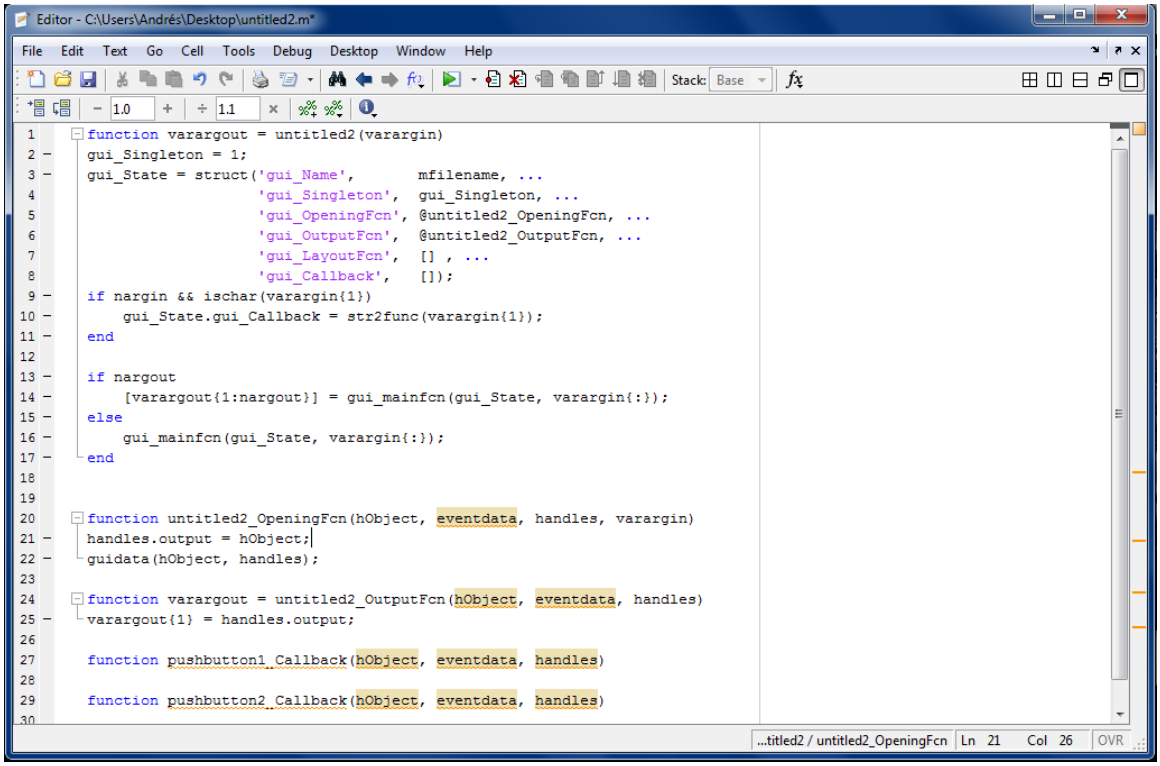


Figura 4.8

- **Función varagout:** es la función principal de la GUI. La genera automáticamente el entorno GUIDE basándose en las opciones de la GUI y no debe ser modificada.

- **Función NombredelaGUI_OpeningFcn:** En esta se escriben las líneas que queremos que se ejecuten un instante antes de que la GUI se habrá y el usuario pueda empezar a interactuar con ella.
- **Función NombredelaGUI_OutputFcn:** cuando llamamos a la interfaz gráfica de usuario desde la command Windows de Matlab, podemos hacerlo como si se tratase de una función con argumentos de entrada y de salida. Un ejemplos son los cuadros de dialogo o las ventanas emergentes, en las que el usuario responde a una pregunta y al cerrarla, esta retorna una variable de salida al programa principal que la ha invocado. Para que una GUI devuelva una variable o vector de salida al cerrarse, basta con que en alguna de las funciones callback asignemos el valor o valores que queremos que devuelva al handles.output.
- **Función tag_del_objeto_ callback:** tenemos tantas funciones de este tipo como objetos con los que se puedan interactuar al ejecutar la GUI. Cuando el usuario de la GUI interactúe con un objeto (Pulse un botón, seleccione una opción en un menú desplegable, etc.) se ejecutara las líneas que hay en la función callback que valla precedida por el nombre de su tag. Por tanto, es en estas funciones donde se programa lo que queremos que ocurra cuando el usuario interactué con cada uno de los elementos con los que puede hacerlo.

4.2.1.- Los handles y las funciones Get y Set.

En el apartado anterior se ha visto que el código de una GUI creada desde el entorno GUIDE está formado por una serie de funciones. En este subapartado se describirá el flujo de información entre estas funciones.

Las variables que definimos en una función del código de la GUI son variables locales, es decir, solo están disponibles para su uso dentro de esa función y durante el tiempo que esta tarda en ejecutarse, ya que una vez se haya ejecutado, aunque el usuario volviese a interactuar con el objeto que la invoca, este valor ya no estaría disponible para su uso. Por tanto, cuando queramos que uno o varios valores estén disponibles para todas las funciones de la GUI, deberemos asignar el valor a una variable global.

En las GUIs de Matlab la asignación de un valor a una variable global se hace escribiendo el siguiente código:

```
handles.Nombre_de_la_variable=Valor;
guidata(hObject, handles);
```

Donde el valor podrá ser una variable local de esa función, un número, un vector de números, una cadena, o un vector de cadenas siempre que estas posean el mismo tamaño. La función “guidata”, se encarga de actualizar la estructura del handles, y es necesario utilizarla después de haber definido una variable global o haber modificado el valor de una ya definida para que los cambios surtan efecto.

La utilidad de la estructura handles en una GUI va más allá de definir variables locales. Como puede verse en la figura 4.9 es esta la que controla el flujo de información de la GUI, y además de las variables globales creadas por el usuario tenemos una para cada tag (un tag por cada objeto insertado en la GUI).

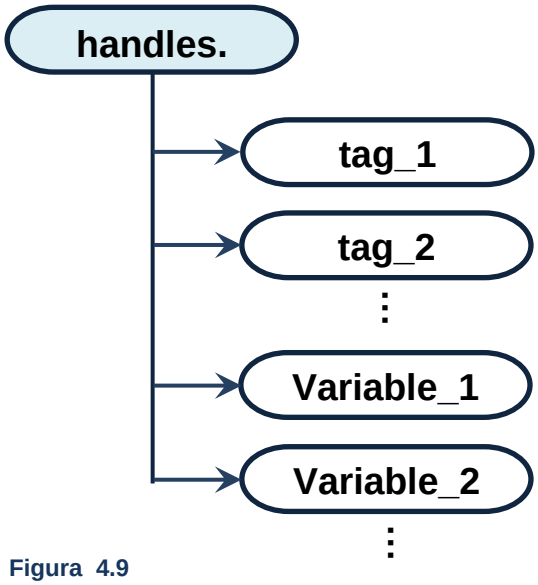


Figura 4.9

En los handles.tag se almacena toda la información referente al objeto designado por ese tag, existiendo dos funciones que nos permiten obtener el estado de un objeto o modificar este a partir de ese handles.

Por ejemplo, si durante la programación del código de una determinada función el comportamiento de esta ha de estar condicionado al estado de un botón con enclavamiento, utilizaremos el siguiente código para conocer si está o no pulsado:

```
get(handles.togglebutton1,'value')
```

Esta función devolverá un uno en caso de que el botón de tag togglebutton1 se encuentre pulsado, y un cero en caso contrario. Combinando esta función con una

condicional conseguiremos que la función se comporte de una u otra forma según sea el estado del botón con enclavamiento.

Para que la propiedad de un objeto como el color de un botón, el contenido de un texto, el tamaño de letra de un botón,… y prácticamente cualquiera de las propiedades de un objeto que puedan ser modificadas desde el property inspector sea modificada desde una función del código de la GUI, utilizaremos la función set. De sintaxis:

```
set(handles.tag,'propiedad','valor')
```

4.3.- Matlab compiler.

Las GUIs realizada con el entorno GUIDE pueden ser ejecutadas en ordenadores que tengan instalado Matlab con tan solo situar el directorio de trabajo en la carpeta en la que se encuentre los ficheros punto “fig” y punto “m”, y escribir el nombre de esta en la Command Windows de Matlab. Sin embargo, si queremos que pueda ser ejecutada sin la necesidad de abrir Matlab, o incluso hacerlo en un ordenador que no lo tenga instalado, será necesario que compilemos la aplicación.

Para ello Matlab dispone del Matlab compiler, al que se puede acceder escribiendo deploytool en la Command Windows y pulsando enter. Con lo que se nos abre la ventana mostrada en la figura 4.10, en la que se le da nombre al proyecto, se escribe el directorio en el que se situara, y que tipo de aplicación vamos a crear (para la GUI de este trabajo se utiliza la opción “Windows Standalone Application”).

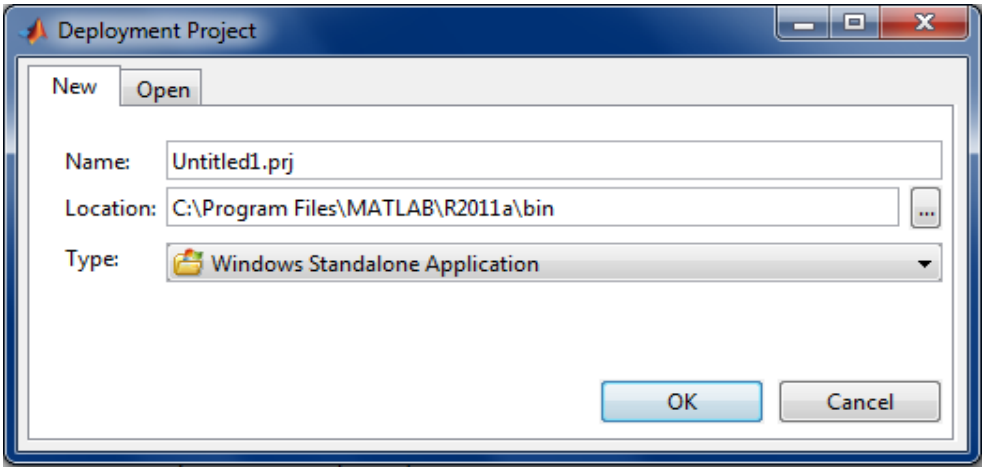


Figura 4.10

Al hacer clic en OK se nos abre la ventana del Matlab compiler, mostrada en la figura 4.11. La presentación de esta ventana ha cambiado bastante de unas versiones a

otras del programa, no obstante la filosofía de trabajo y los iconos que contiene son los mismos indiferentemente de los cambios de presentación que hayan realizado sobre ella.

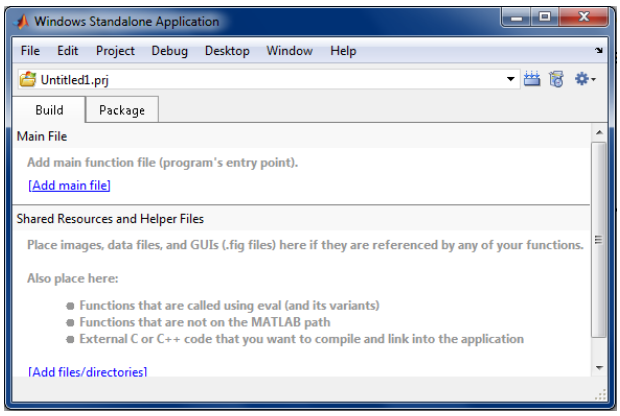


Figura 4.11

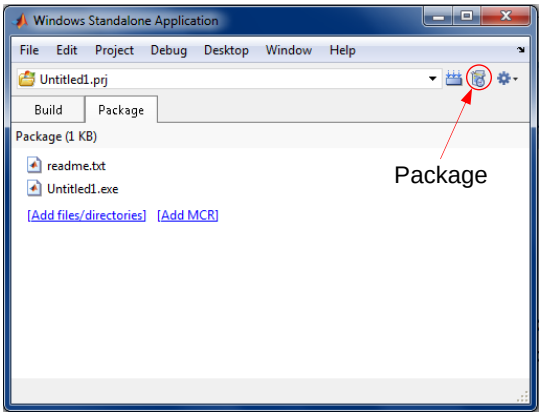


Figura 4.12

En la pestaña “Build” se ha de hacer clic en la opción “Add main file”, para añadir el código del programa (fichero “*.m”), y en “Add file/directories” para añadir el fichero “*.fig” y el resto de ficheros que necesite la GUI para funcionar, como son aquellas funciones de Matlab que no vienen por defecto con Matlab (en mi caso la función gridfit, y funciones que he realizado para esta).

Con esto podríamos copilar la interfaz y obtener un ejecutable. Sin embargo, este no podría trabajar en ordenadores sin Matlab. Por ello debemos de incluir en el paquete el MCR (Matlab Compiler Runtime), que es un conjunto independiente de bibliotecas compartida que permiten la ejecución de aplicaciones y componentes creados en Matlab en ordenadores que no lo tienen instalado y sin necesidad de tener licencia de Matlab. Para añadir estas bibliotecas se hace clic en “Add MCR”, en la pestaña “Package” (figura 4.12).

Para compilar la GUI y que nos agrupe todos los ficheros en uno solo, se ha de hacer clic en “Add file/directories” (en la pestaña Package) para añadir los ficheros con imágenes y datos (en mi caso ficheros “.DAT”) que necesite la GUI para funcionar. Hacer clic en el icono “Package”, figura 4.12. e indicarle el lugar en el que queremos que guarde el fichero cuando nos lo pregunte.

De esta forma se obtiene el fichero “*.exe” de la GUI, que al ejecutarse instalará el MCR en el ordenador, y desempaquetara junto al ejecutable que abre la GUI todos aquellos ficheros que hallamos agregado en la pestaña “Package”.

5.- CLASIFICACIÓN DE LAS MÁQUINAS ELEVADORAS DE LÍQUIDOS.

La bomba hidráulica es posiblemente la máquina más antigua que se conoce. Un ejemplo lo constituyen las norias que empezaron a utilizarse hace 3000 años, o el tornillo de Arquímedes, que data del siglo III antes de cristo y que aún hoy sigue teniendo un papel muy importante en las depuradoras de aguas residuales.

Incluso con el gran avance tecnológico acontecido desde que empezaran a utilizarse las primeras máquinas elevadoras de fluidos, hoy en días siguen ocupando un lugar de privilegio, siendo solo por detrás del motor eléctrico las máquinas de uso más común.

Puesto que el objetivo principal de este trabajo fin de grado es la creación de una aplicación gráfica para el caracterizado de bombas centrífugas. En este capítulo y pesé a la gran respeto que me inspiran estas máquinas, solo haré una breve introducción a los principales tipos de bombas, haciendo algo más de hincapié en las bombas centrífugas que son las que nos atañen.

Existen distintos criterios para la clasificación de las bombas hidráulicas, siendo el utilizado en esta memoria uno de los más extendidos. (Ver esquemas en figura 5.1)

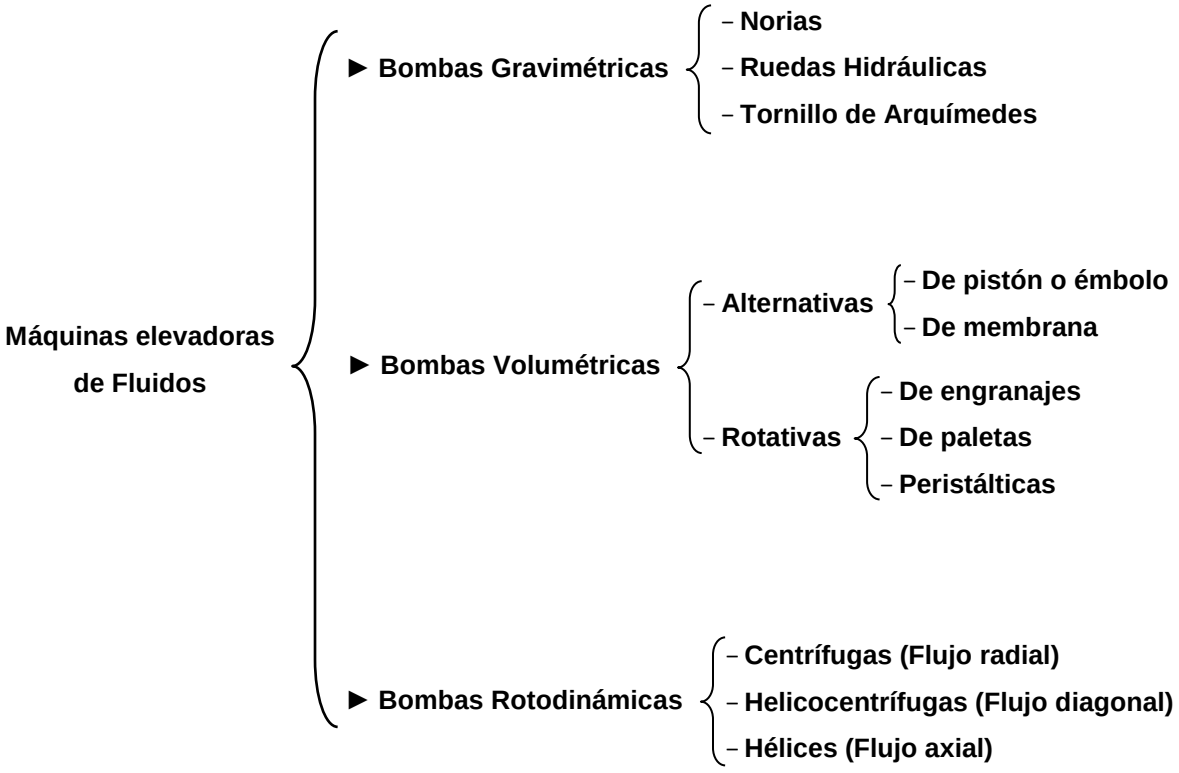


Figura 5.1: Clasificación de las bombas hidráulicas

5.1.- Bombas gravimétricas.

Las bombas gravimétricas son máquinas que actúan simplemente sobre el término de posición de la ecuación de Bernoulli (Ec.1) para transferir energía al fluido. Esto lo consiguen trasladando el fluido de un punto a otro de mayor cota (con lo que aumenta la energía potencial).

$$g \cdot \mathrm{H_m} = \frac{\mathrm{P_2} - \mathrm{P_1}}{\rho} + \mathrm{g} \cdot (\mathrm{Z_2} - \mathrm{Z_1}) + \frac{\mathrm{V_2^2} - \mathrm{V_1^2}}{2} \qquad (1)$$

El ejemplo clásico de este tipo de máquinas lo constituyen las norias (ver figura 5.2), hoy prácticamente en desuso. Otro tipo de bomba gravimétrica que pese a ser muy antigua hoy en día se sigue utilizando para elevar aguas residuales, fluidos cargados de solidos e incluso el transporte de agua con peces vivos, son las bombas de tornillo de Arquímedes, figura 5.3.

Figura 5.2: Noria de la Ñora (Murcia)

Figura 5.3: Bombas de tornillo de Arquímedes

5.2.- Bombas volumétricas.

Las bombas volumétricas o de desplazamiento positivo están compuestas por una serie de cavidades o cámaras que se llenan y vacían periódicamente. En estas bombas el fluido entra en la cámara, y es impulsado hacia fuera por la presión que el cuerpo móvil ejerce sobre él, por lo que solo aportan energía de presión.

A diferencia de las bombas rotodinámicas, las bombas de desplazamiento positivo producen un caudal prácticamente independiente de la presión de impulsión, por lo que

se suele dotar a la tubería de impulsión de una válvula de contrapresión que alivia (retorna) parte del caudal desplazado cuando la presión excede el valor al que se ha tarado la válvula.

Estas bombas se clasifican en alternativas y rotativas en función de que el movimiento del elemento impulsor sea alternativo o rotativo, y suelen utilizarse en aplicaciones en las que se requiere alta o media presión y bajos caudales,

5.2.1.- Bombas Volumétricas alternativas.

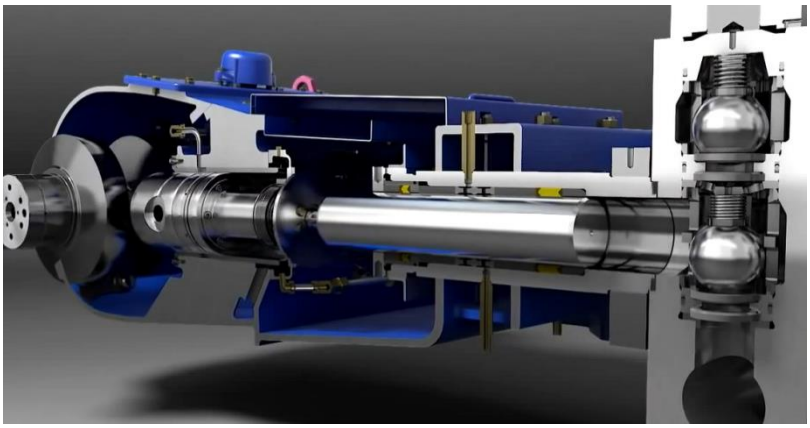


Figura 5.4: Bomba de pistones de la marca “Uraca”

Este tipo de máquinas bombean el fluido aprovechando el movimiento alternativo de una pared móvil (pistón) o deformable (membrana) y dos válvulas antirretorno, una que permite el ingreso del fluido a la cámara a través de la tubería de aspiración y la otra que permite la salida de este a través de la tubería de impulsión. (Ver figuras 5.4 y 5.5)

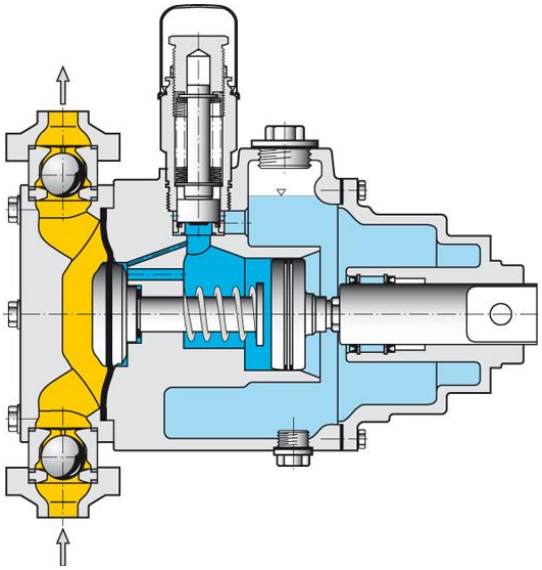


Figura 5.5: Esquema de una bomba de membrana

5.2.1.- Bombas Volumétricas rotativas.

En estas bombas existe uno o varios rotores que al girar capturan el fluido entre ellos, o entre ellos y la carcasa de la máquina, y lo transportan hasta la tubería de impulsión. Ofreciendo de esta forma un caudal más uniforme que el obtenido mediante una bomba volumétrica alternativa.

Su empleo está ampliamente extendido en la industria y el transporte, siendo utilizadas para el engrase de motores y compresores, como bomba hidráulica en las transmisiones hidráulicas de potencia, e incluso en la industria alimentaria y farmacéutica para el transporte de fluidos sensibles.

Como puede observarse en las siguientes imágenes, existen multitud de modelos en función de la geometría del rotor o rotores:

- Bombas lobulares. (Figura 5.6)

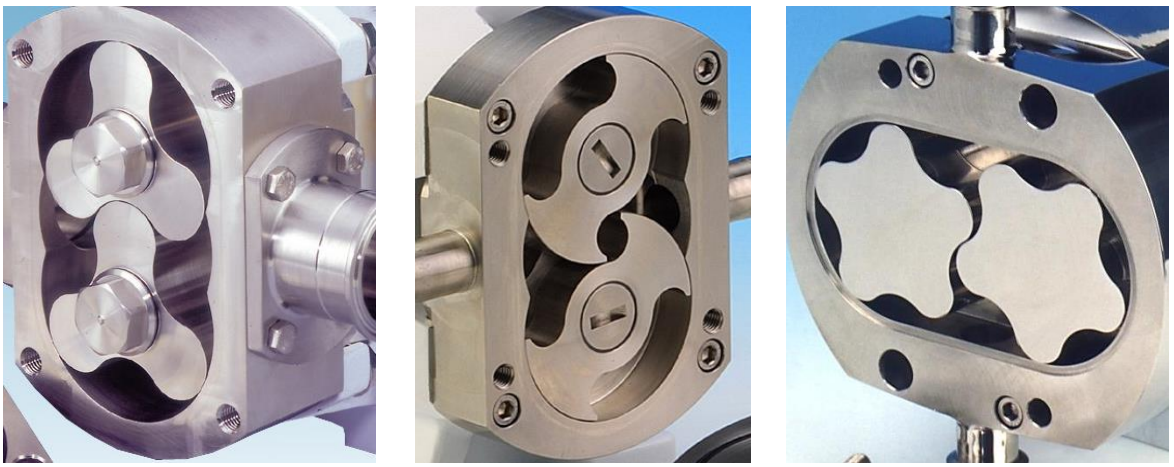


Figura 5.6: Distintos tipos de bombas lobulares

- Bombas de engranajes. (Figura 5.7)

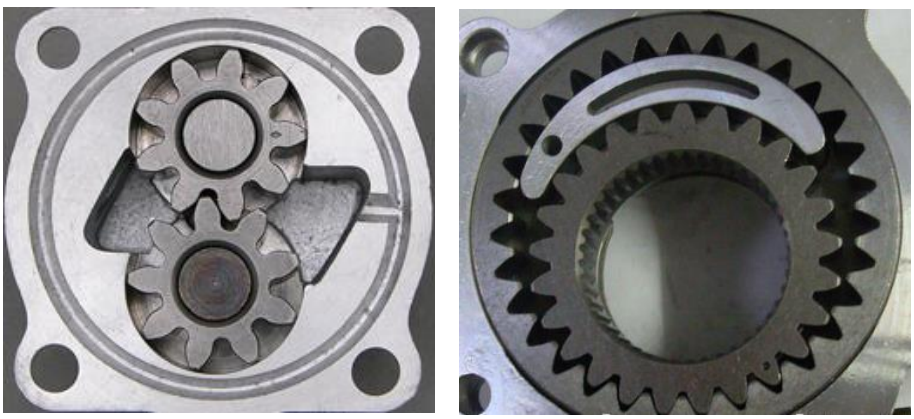


Figura 5.7: Bomba de engranajes exteriores, y bomba de engranaje interior

- Bombas peristálticas o de tubo flexible. (Figura 5.8)
- Bomba de paletas. (Figura 5.9)

Figura 5.8: Bomba peristáltica

Figura 5.9: Bomba de paletas

5.3.- Bombas rotodinámicas.

Las bombas rotodinámicas añaden cantidad de movimiento al fluido al circular este a través de un conjunto de álabes giratorios llamado rodete (de ahí que a estas máquinas también se las denomine bombas de intercambio de cantidad de movimiento).

Estas bombas se clasifican según la dirección del fluido a su paso por el rodete en:

- Bombas hélice o de flujo axial.
- Bombas centrífugas o de flujo radial.
- Bombas helicocentrífugas, de flujo diagonal, o de flujo mixto.

5.3.1.- Bombas hélice.

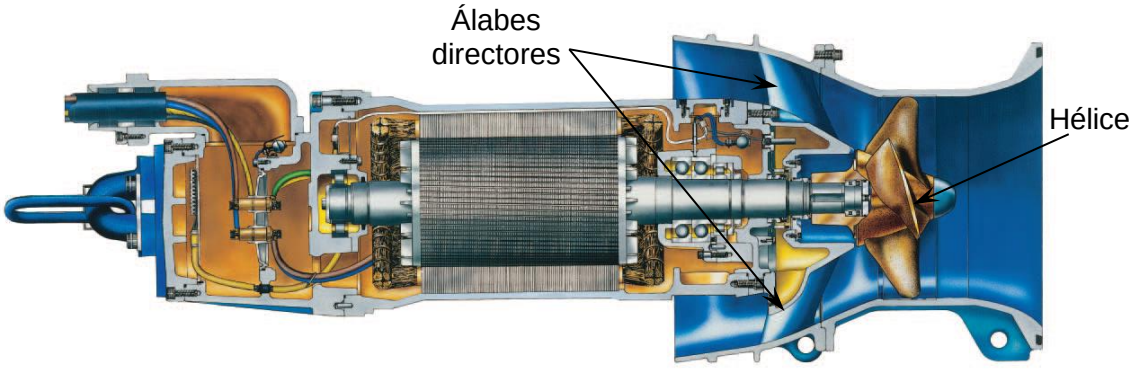


Figura 5.10: Sección de una bomba de hélice de la marca “FLYGT”

Las bombas de hélice (figura 5.10), son idóneas para aquellas aplicaciones en las que se requiere bombear grandes caudales a poca altura de elevación, de ahí que se empleen para regadíos, drenaje de terrenos y la manipulación de aguas residuales.

En estas máquinas el fluido entra axialmente al impulsor (también denominado rodete o hélice) y los álabes le imprimen una componente rotacional que hacen que las partículas describan una trayectoria de hélice circular a su paso por el impulsor, no obstante, el aumento de presión se debe a la acción impulsora o de sustentación ejercida por los alabes, de ahí que se denominen bombas de flujo axial pese a que en su paso por el impulsor el fluido no recorra una trayectoria puramente axial al eje.

A estas bombas se les dotan de unos álabes fijos a la carcasa, alabes directores, que dirigen el flujo en dirección axial a la salida del rodete, con lo que se elimina la componente rotacional de la velocidad, disminuyendo así la energía cinética y ganando energía de presión a la salida de estos.

5.3.2.- Bombas centrífugas

La bomba centrifuga es actualmente el tipo de bomba más utilizado. Como dato, decir que en la mayoría de plantas petrolíferas entre el 80 y el 90% de las bombas utilizadas son centrifugas. El uso de estas máquinas está tan ampliamente extendido debido a su simplicidad constructiva, alta eficiencia, holgado rango de trabajo, flujo homogéneo (libre de pulsaciones) y por su facilidad de utilización y de mantenimiento.

El funcionamiento de una bomba centrífuga es relativamente sencillo, cuando el impulsor gira transfiere aceleración radial al fluido que se encuentra en su interior, lo que fuerza a este a desplazarse hacia la voluta (figura 5.11). El fluido que se encuentra junto a la entrada del impulsor (ojo del rodete), se desplaza hacia el interior de este para ocupar el volumen que queda libre, estableciéndose de esta manera un flujo a través del rodete que adquiere energía cinética gracias a la acción mecánica ejercida por el impulsor. Al ingresar el fluido en la cámara en espiral, la geometría de esta (difusor) logra transformar parte de la energía cinética en energía de presión, que es tipo de energía que se pretende conseguir al utilizar una bomba.

El régimen de giro de trabajo para una bomba centrífuga está comprendido entre las 1500 y 3600 rpm. Sin embargo, se tiene constancia de la existencia de bombas diseñadas para operar en el rango comprendido entre las 5000 y 25000 rpm.

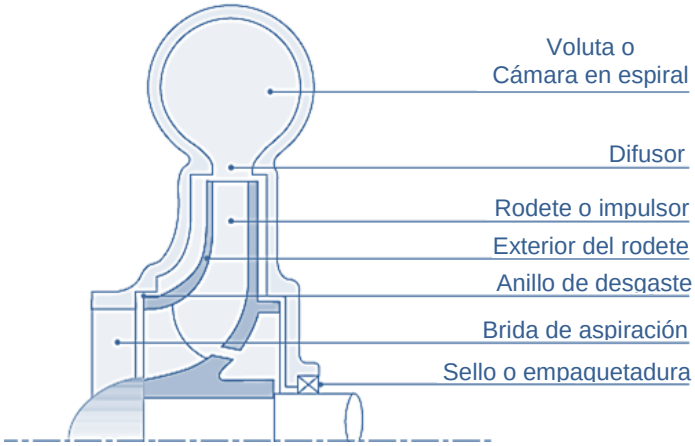


Figura 5.11: Partes de una bomba centrífuga

Las partes más importantes en el funcionamiento de una bomba centrífuga son las siguientes:

- **Rodete o impulsor**, como ya se ha explicado antes, su función es la de transmitir energía cinética al fluido, con lo que logra su impulsión. En función del tipo de fluido a bombear, podemos encontrarnos distintas tipologías (figura 5.12):
    - Abierto: Se utiliza para el bombeo de fluidos viscosos o que arrastren fibras y cuerpos similares con tendencia a atascar el impulsor. Posee menos rendimiento que uno de tipo cerrado, por lo que solo se utiliza en aquellas aplicaciones en las que uno de este tipo no podría funcionar porque se atascaría, o cuando el fluido de trabajo requiera limpiar la máquina en cada parada. (Como ocurre en el bombeo de pinturas, lacas y barnices)
    - Semi-abiertos: más resistente que un impulsor de tipo abierto y con prácticamente las mismas ventajas que uno abierto.
    - Cerrado: permiten obtener mayores rendimientos que los dos anteriores.

Abierto Semi-abierto Cerrado

Figura 5.12: Tipología de rodetes para bombas centrífugas

- **Anillo de desgaste:** como se muestra en la figura 5.13, existe una gran diferencia de presión entre la entrada del rodete y la salida de este, lo que produce una recirculación dentro de la bomba que disminuye notablemente el rendimiento. Para disminuir en la medida de lo posible esta pérdida, se colocan los anillos de desgaste, que pueden ser de bronce, del mismo material que la carcasa, o incluso estar dotados de unos laberintos que dificultan el paso del fluido. Uno de los principales motivos por el que los rodetes abiertos y semi-abiertos poseen menos rendimiento, es que no se les puede añadir este elemento.

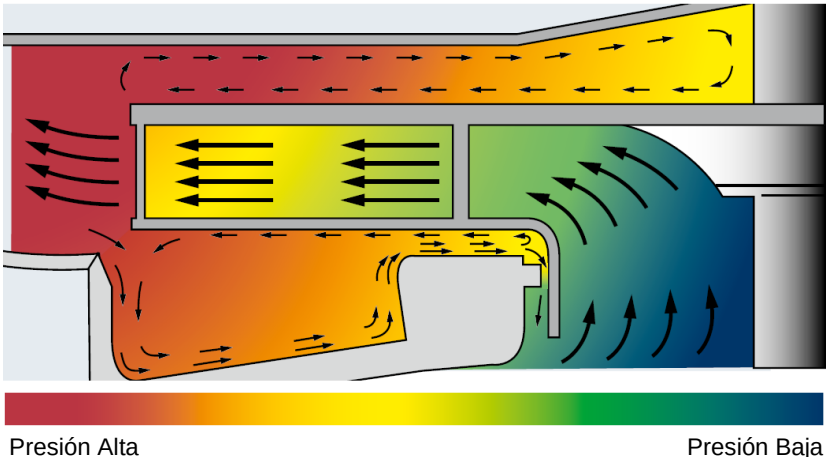


Figura 5.13: Contorno de presiones

- **Sello o empaquetadura mecánica:** Al ser la carcasa de la bomba fija y el eje de esta movil, es necesario colocar entre ambos un elemento que impida que escape el fluido bombeado por esa zona. En las primeras bombas centrifugas se colocaban unos anillos contituidos por unas trenzas cuadradas de estopa lubricadas con cera de abeja o grasa animal (figuras 5.14). Estos anillos se colocaban en un asiento mecanizado en la carcasa para tal fin (estopero) , donde se prensaba con tornillos (prensa estopas, figura 5.15). Con el tiempo se fueron sustituyendo las fibras naturales por

Figura 5.14: Estopas

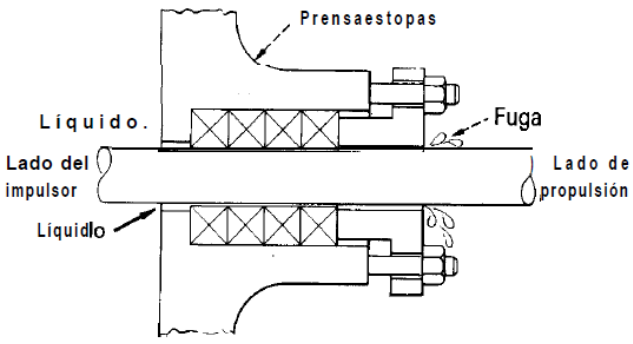


Figura 5.15: Empaquetadura mecánica

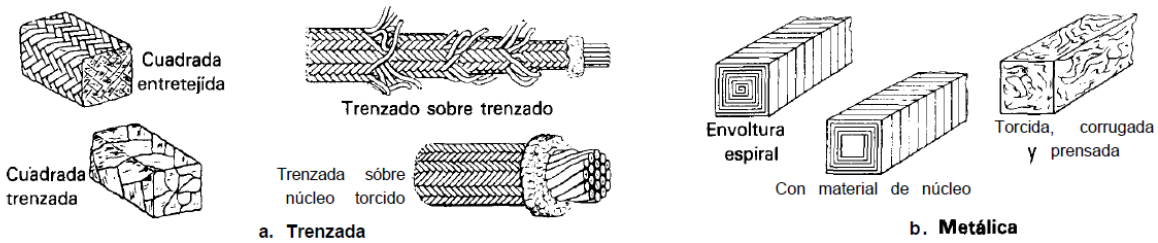


Figura 5.16: Empaquetadura mecánica

fibras sinteticas, metálicas, de vídrio, e incluso de cintas de grafito (para más detalle, ver figura 5.16). Pese a que aún se utiliza este sistema en bombas centrifugas por poseer ciertas ventajas en aplicaciones muy específicas, está siendo prácticamente sustituido por los sellos mecánicos.

Los sellos mecánicos, figura 5.17, están constituidos principalmente por una parte fija unida a la carcasa de la bomba, y otra movil solidaria al eje del impulsor. La dos caras principales del cierre se presionan entre sí por medio del resorte y por la presión ejercida por el fluido, dejado pasar una fina capa de fluido entre las caras, que lo lubrica y se evapora antes de llegar a la atmosfera. Por eso, que este tipo de sellos se consideren estancos aunque en realidad dejen escapar una insignificante cantidad de fluido en forma de vapor.

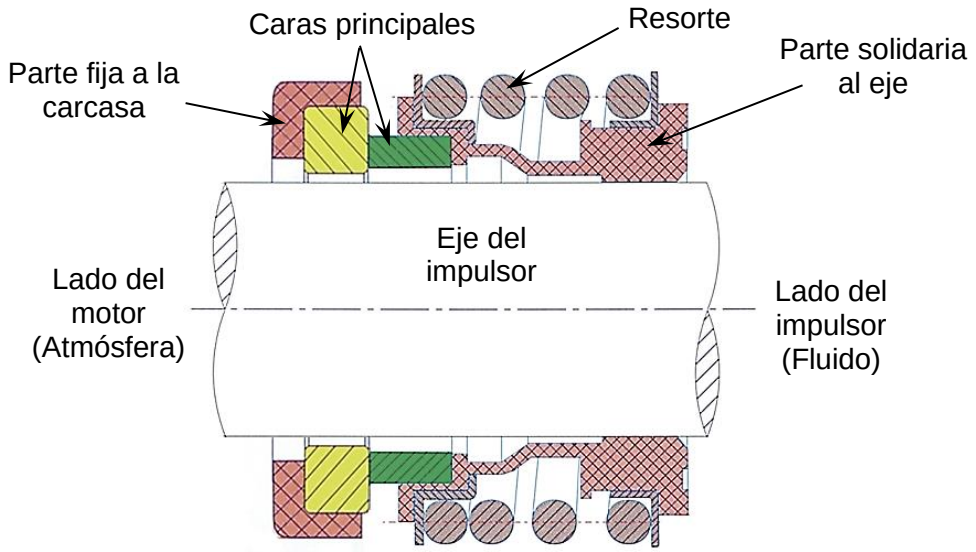


Figura 5.17: Sello mecánico

- **Carcasa o voluta:** Tambien conocida como camara en espiral. Su función es dirigír el fluido que sale de los alabes hasta la tubería de impulsión, y ha de hacerlo produciendo las menor perdida de carga posible y transformando la mayor parte de energía cinética que se pueda en energía de presión.

Aunque se le llame camara en espiral, no es obligatorio que tenga esta forma, ya que es posible obtener un efecto similar utilizando un difusor de alabes (ver figura 5.18). De hecho, esta es la forma utilizada en las bombas multietapa, en las que el fluido que sale de un rodete a de ingresar en el siguiente.

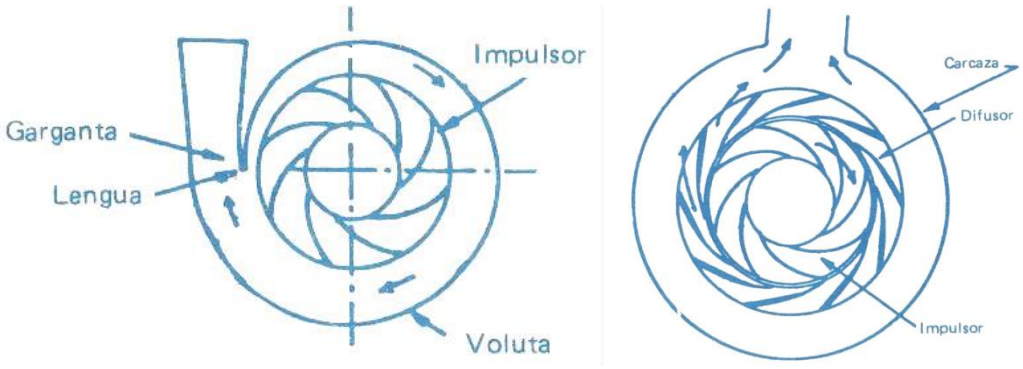


Figura 5.18: Bomba con difusor de cámara en espiral, y bomba con difusor de álabes.

5.3.3.- Bombas Helicocentrífugas.

Las bombas helicocentrífugas o de flujo mixto están constituidas por los mismos elementos que una bomba centrifuga, diferenciándose tan solo en la geometría del rodete que hace que el fluido circule diagonalmente al eje de la bomba y en las prestaciones.

En cuanto a prestaciones, una bomba helicocentrífuga está a medio camino entre las bombas de flujo axial y de flujo radial, dando un mayor caudal que una bomba centrifuga pero menor que el de una axial, y proporcionado una altura manométrica mayor a la que nos proporcionaría una bomba axial, pero menor que la que puede proporcionarnos una centrífuga.

Como puede verse en la figura 5.19, el rodete de esta bombas es muy similar al de una bomba centrifuga, salvo por que en este, los álabes en la parte de la aspiración (intradós), tiene una forma que recuerda a una hélice.

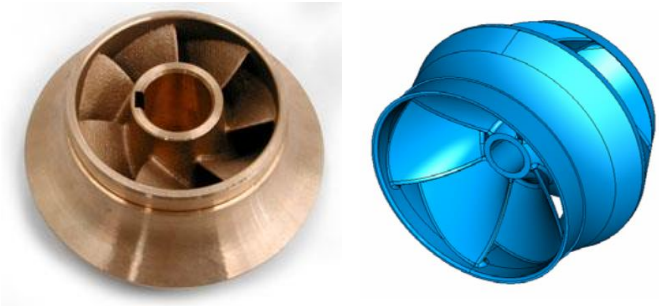


Figura 5.18: Rodete bomba helicocentrífuga

5.3.4.- Velocidad específica.

La velocidad específica es un factor de diseño indicativo de las características de una bomba, y se define como la velocidad a la que sería necesario girar una bomba centrífuga semejante para que proporcionara un caudal de 1m^3/s a 1 mca. de altura. La ecuación 2 permite obtener la velocidad específica, introduciendo el caudal en m^3/s, la altura manométrica en metros de columna de agua, y la velocidad nominal en revoluciones por minuto.

$$n_q = n \cdot \frac{\sqrt{Q}}{H^{3/4}} \tag{2}$$

La figura 5.19 nos muestra el rendimiento máximo de las bombas en función de la velocidad específica. Aunque estos valores se corresponden a bombas de gran tamaño

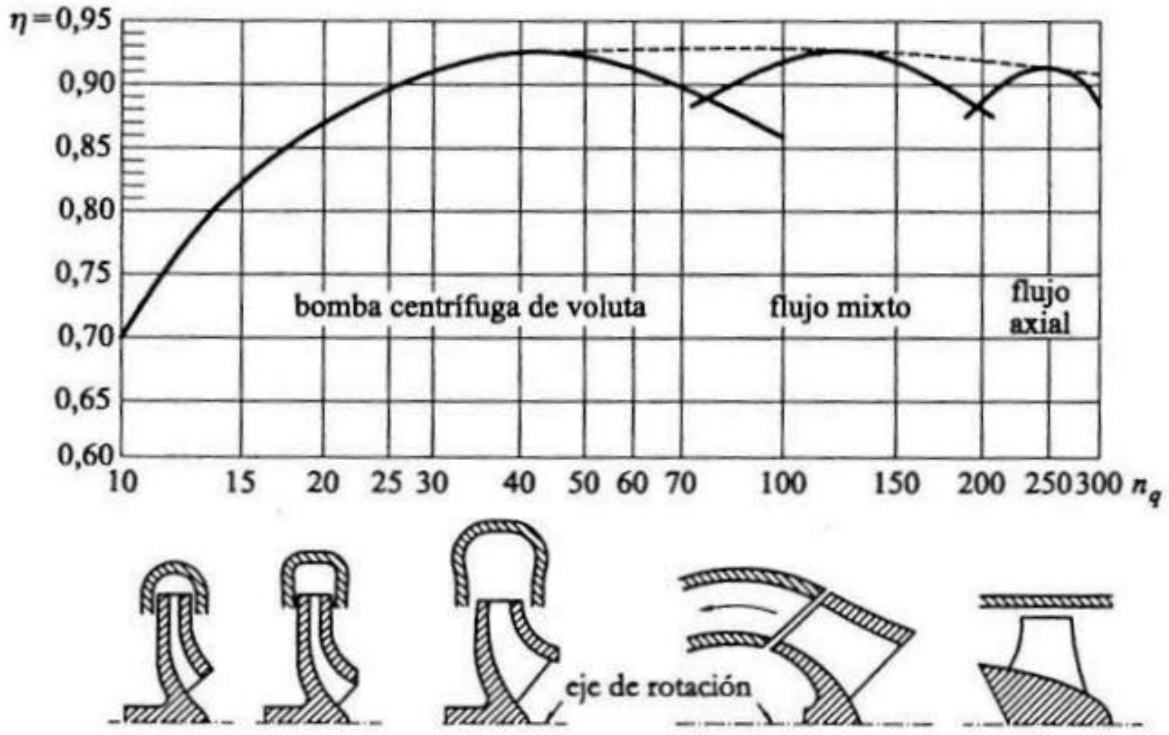


Figura 5.19: Rendimiento máximo de una bomba según familia y velocidad específica.

(en las que se obtienen mayores rendimientos), se puede deducir que el rendimiento máximo se obtiene:

- Para bombas centrífugas cuando $n_q \approx 50$
- Para bombas helicocentrífugas cuando $n_q \approx 130$
- Para bombas hélice cuando $n_q \approx 250$

6.- DESCRIPCIÓN DEL BANCO DE ENSAYO.

El equipo utilizado para la realización de este trabajo fin de grado es el banco de bomba centrifuga controlado por computadora (figura 6.1), del fabricante “Edibon”.

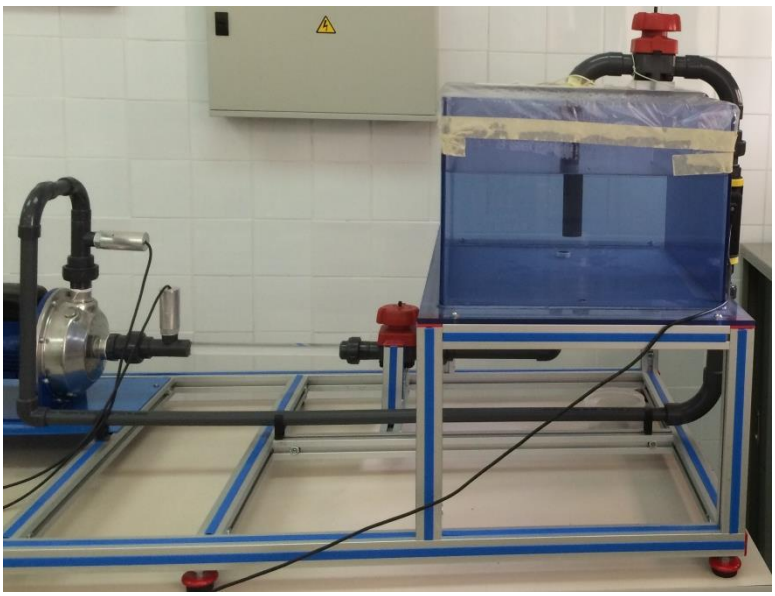


Figura 6.1: Banco de ensayo del laboratorio de fluidos y termotecnia

Este banco consta de:

- **Una bomba centrífuga:** Es la que se caracteriza con el banco de ensayo (figura 6.2). Para más detalle sobre el funcionamiento y construcción de estas bombas, ver apartado “5.3.2.- Bombas centrífugas”.

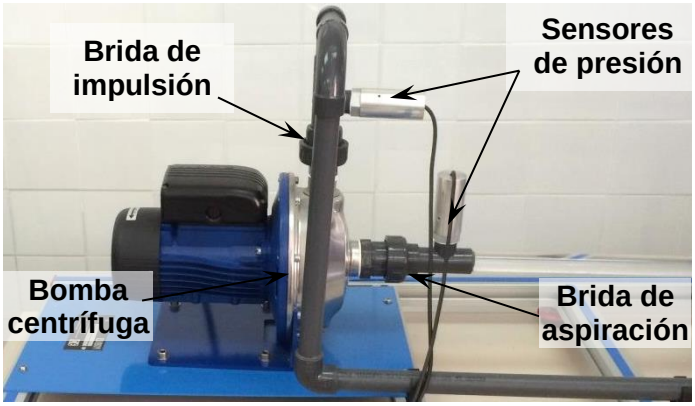


Figura 6.2: Bomba centrífuga y sensores de presión

- **Dos sensores de presión:** una para medir la presión en la brida de aspiración, y otra para medirla en la brida de impulsión, figura 6.2.
- **Sensor de caudal:** mide el caudal que circula por la bomba, figura 6.3.

Figura 6.3: Sensor de caudal

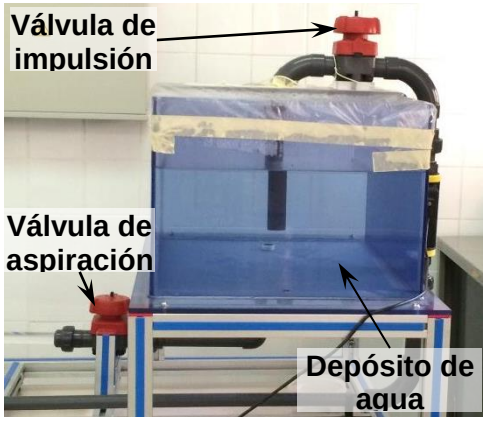


Figura 6.4: Depósito de agua

- **Depósito de agua:** la bomba funciona tomando y cediendo el fluido de trabajo de este (figura 6.4). Se encuentra tapado para impedir que el fluido se evapore.
- **Válvulas de diafragma:** (figura 6.4) sirven para estrangular el paso de fluido a través de la tubería de aspiración y de impulsión independientemente, con lo que se consigue controlar la caída de presión y con ello el caudal suministrado por la bomba.
- **Caja interface de control:** (figura 6.5) se encarga de transmitir al ordenador la información que recoge de los distintos sensores, y de regular la velocidad de giro del motor, controlando la frecuencia de la corriente eléctrica que le suministra.
- **Ordenador:** para la adquisición de datos y regulación de velocidad de la bomba, se utiliza un ordenador al que se le ha instalado el software específico del fabricante, y una tarjeta para la adquisición de datos provenientes de la caja interface de control.

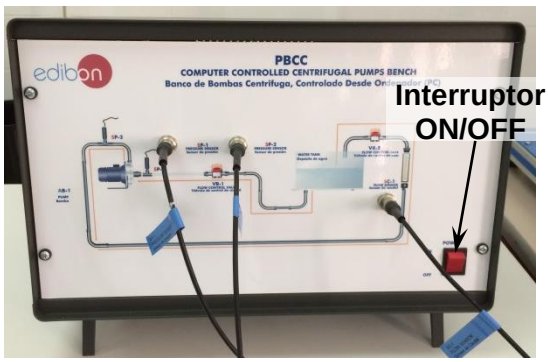


Figura 6.5: Caja-interface de control

En el diagrama de la figura 6.6 se puede ver el esquema de la instalación, así como el recorrido del fluido y la situación de los distintos sensores.

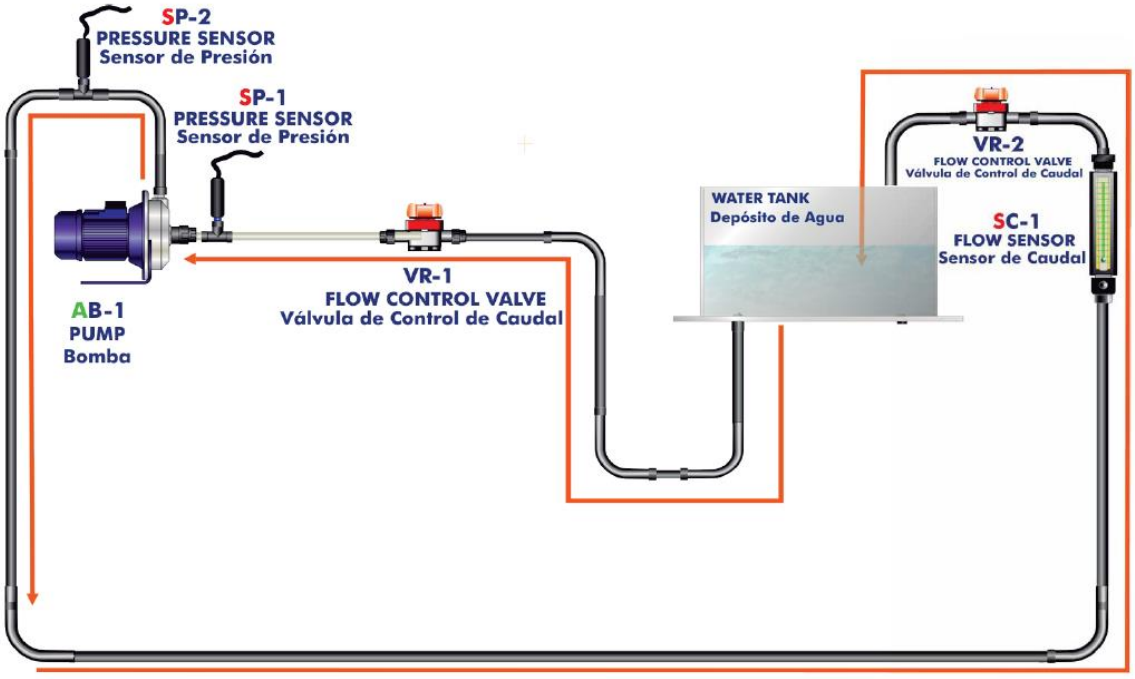


Figura 6.6: Esquema de la instalación

6.1.- Descripción de los sensores

Según el fabricante del banco de ensayo, este dispone de instrumentación y sensores que le permiten medir los parámetros más representativos de la bomba centrífuga, que son:

- Velocidad del motor.
- Caudal.
- Presión de admisión y de descarga.
- Par.

Y calcular a partir de esto:

- Altura total.
- Potencia hidráulica.
- Potencia mecánica.
- Rendimiento.

Para la adquisición de estos parámetros se dispone de los siguientes sensores:

- **Sensor de caudal SC-1:** (figura 6.6) con un rango de medida que va desde cero hasta 150 litros/minuto.
- **Sensor de presión en la aspiración SP-1:** (figura 6.6), con un rango de medida que va desde -1 bar hasta cero (Presión de vacío).

- **Sensor de presión en la impulsión SP-2:** (figura 6.6), con un rango de medida que va desde cero hasta 2,5 bares (Presión manométrica).

Como se puede observar, el banco no dispone de sensores para medir la velocidad de giro o el par motor, por lo que intuyo que estos valores los obtiene indirectamente a partir de la potencia consumida por el motor eléctrico, la frecuencia a la que los está alimentado, y los datos de ensayo del motor.

6.2.- Toma de datos.

Para la adquisición de valores de ensayo que más tarde puedan ser utilizados por la aplicación grafica “Caracterizador de bombas centrífugas”, se ha de seguir el siguiente procedimiento:

1) Conectar el equipo a la red eléctrica.
2) Arrancar el ordenador situado junto al banco de ensayo.
3) Abrir la válvula de la tubería de aspiración y la de la tubería de impulsión al máximo. (Para más detalle de la situación de estas válvulas, ver figura 6.4)
4) Activar interruptor de la caja-interface de control, figura 6.5.
5) Abrir software de adquisición de datos del fabricante, “BOMBA_SACED_PBCC” figura 6.7.

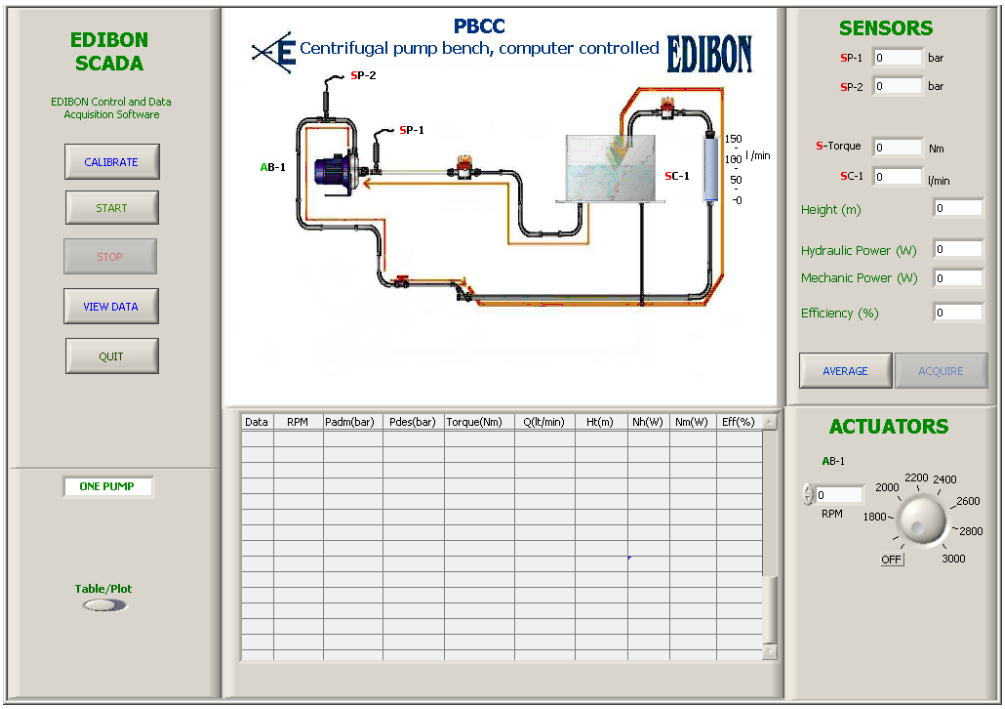


Figura 6.7: Ventana principal del software de adquisición de datos “PBCC”

6) Introducir el valor numérico de revoluciones a las que se quiere realizar el ensayo en la esquina inferior derecha de la ventana principal, figura 6.7. (Rango admisible de 1500 a 3000 rpm).

7) Hacer clic en el botón “START” y pulsar “Ok” en la venta que se muestra en la figura 6.8.

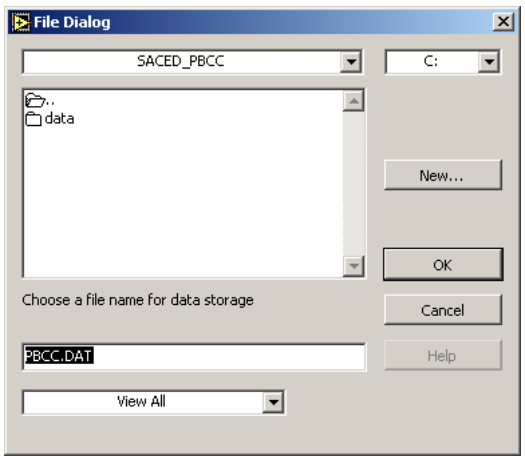


Figura 6.8:

8) Nos aparecerá una ventana como la mostrada en la figura 6.9, en la que podremos elegir el nombre y la dirección en la que queremos que nos guarde el fichero con los datos que ensayemos. La bomba se pondrá automáticamente en marcha.

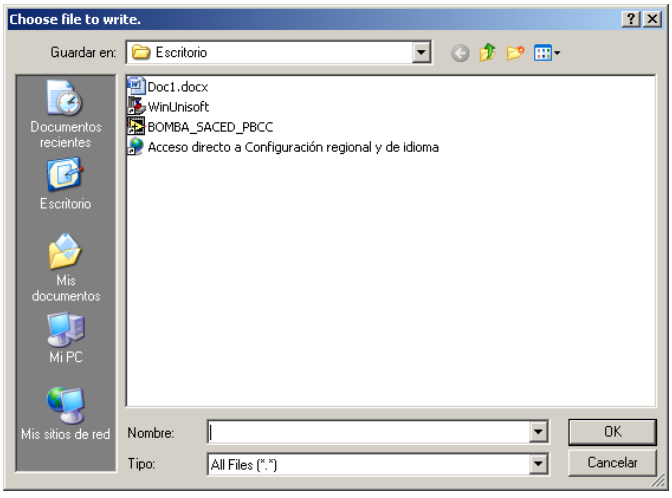


Figura 6.9:

9) Esperaremos a que los valores de caudal, presión,… se estabilicen, y se hará clic en el botón “AVERAGE”, que realiza una media del valor tomado por los sensores.

10) Hacer clic en el botón “ADQUIRE” para tomar el valor, y el botón “AVERAGE” para que se desactive la opción de promedio.

11) Con esto se tendrá el punto de mayor caudal. Para calcular de cuanto en cuanto tenemos que disminuir el caudal para pasar de un punto al siguiente, podemos utilizar la expresión (Ec. 3):

$$\Delta Q = \frac{Q_{Máximo}}{N° \, de \, puntos} \tag{3}$$

12) Cerrar poco a poco la válvula de impulsión, hasta que la lectura indicada para el caudal disminuya la cantidad deseada.
13) Esperar a que las lecturas se estabilicen, y hacer clic en los botones “AVERAGE”, “ADQUIRE”, y de nuevo “AVERAGE”.
14) Repetir los puntos 12 y 13 hasta llegar a caudal cero, y hacer clic en el botón “STOP”.
15) Para obtener los datos referentes a otro régimen de giro se debe abrir por completo la válvula de impulsión, y repetir los pasos del 6 al 15.
16) Una vez, hayamos terminado de tomar datos se hará clic en el botón “STOP”.
17) Se cerrará el programa y guardarán los datos en un pendrive.
18) Se apagará el ordenador, y se cortará la caja-interface de control de su interruptor, figura 6.5.
19) Se desconectará el equipo de la red eléctrica.

En este capítulo no se ha explicado el procedimiento de toma de datos para la obtención de las curvas de NPSH, ya que pese a la importancia de estas curvas para la selección de bombas centrífugas, en la GUI objeto de este proyecto no se trabaja con ellas.

7.- ESTRUCTURA DEL PROGRAMA.

El programa realizado en este trabajo fin de grado ha sido estructurado en tres pestañas, una para las curvas dimensionales, otra para las curvas adimensionales y finalmente una dedicada al estudio de semejanza (Ver figura 7.1). Aunque todas estas pestañas van enfocadas al análisis de una bomba centrífuga, cada una lo hace con objetivos distinto, y es por ello que desde el primer boceto de la interfaz gráfica consideré esta división del programa como la mejor opción para facilitar su uso.

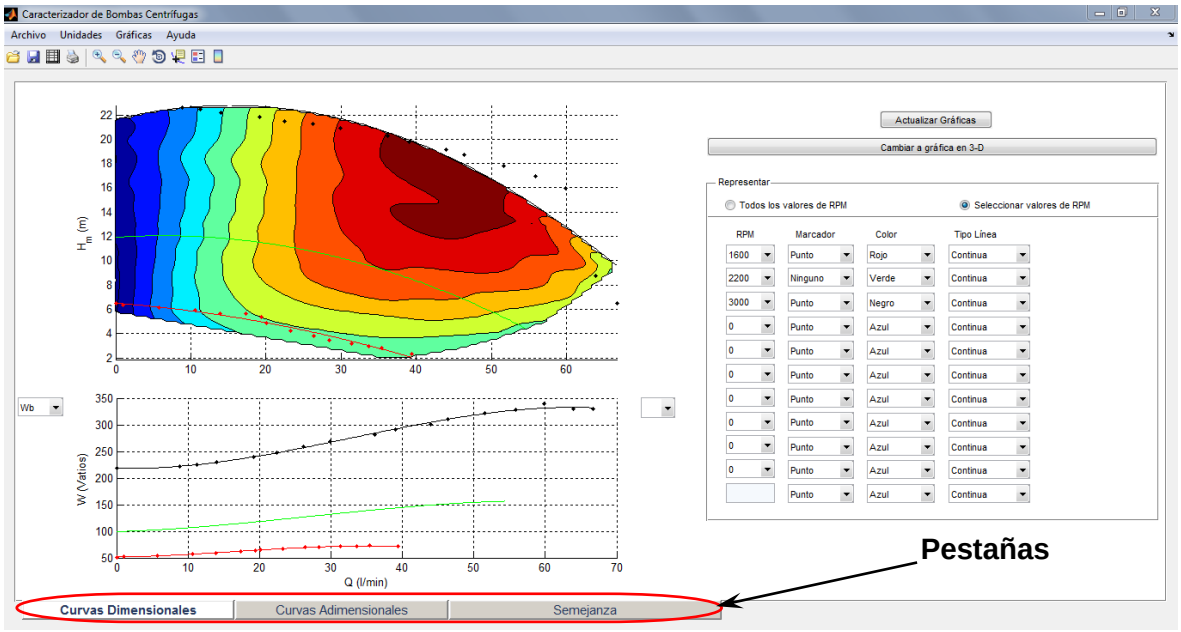


Figura 7.1: Ventana de la GUI "Caracterizador de bombas centrífugas"

Aquellas personas que estén familiarizadas con el uso de GUIs generadas en Matlab, sabrán que no son muy usuales los diseños con pestañas y que en su lugar, se prefiere realizar una GUI principal con un menú que nos dirige a otras GUIs secundarias en función de lo que deseemos hacer. Esta forma de trabajo se debe básicamente a que el entorno de generación de GUIs de Matlab no nos da la opción de insertar pestañas directamente, como ocurre por ejemplo en Java.

Pese a que el diseño de una GUI principal con un menú, me hubiese resultado más simple que realizar una con pestañas, desde un principio me negué a esta filosofía por considerarla menos elegante, y sobre todo, porque el traspaso de datos entre las distintas GUIs sería mucho más lento. Así que para hacer posible la operatividad de estas pestañas, he hecho uso de un artificio consistente en utilizar tres botones de presión. Con un código en sus "callback" tal que al pulsar cualquiera de ellos cambie el color de fondo y el tipo de letra, dándole al botón pulsado un fondo blanco y un tipo de letra negrita, y a

los otros dos botones un color de fondo oscuro y un tipo de letra normal, con lo que se consigue dar el aspecto de pestaña activa. Para modificar lo que aparece en la ventana gráfica, el código del programa hace uso de la propiedad de visibilidad de aquellos objetos que la requieren, y actualiza las gráficas para que muestren el contenido correspondiente a la pestaña activa.

Debido a la extensión y complejidad del código bajo el que se rige la interfaz gráfica diseñada, no puedo explicar detalladamente su funcionamiento en esta memoria sin extenderme más de lo que cabría esperar. Por lo que en este capítulo solo explicaré el modo de operar, y las expresiones utilizadas para el cálculo y representación de las distintas gráficas.

7.1.- Importación de datos.

No se puede conferir una interfaz gráfica para el análisis y caracterizado de bombas centrifugas sin la capacidad de importar o adquirir datos de alguna manera, por lo que antes incluso de realizar el primer boceto de la ventana gráfica, llegue a la conclusión de que era necesario realizar una función para importarlos.

El banco de ensayo utilizado en el desarrollo de este trabajo (descrito en el capítulo “6.- Descripción del banco de ensayo.”) nos proporciona los datos de ensayo en uno o varios ficheros de extensión “.DAT”, no obstante, desde un primer momento quise dotar a la GUI de la capacidad de analizar y caracterizar bombas distintas a la ensayada en este banco, por lo que realicé una función que pudiera importar datos tanto de ficheros de extensión “.DAT” como de hojas de cálculo de extensión “.XLS”, e incluso de ambos tipos de ficheros a la vez.

La función realizada para cubrir estas necesidades es la que se expone en el apartado “11.2.1.- Función “Cargar_datos””, con argumentos de entrada la dirección en la que se encuentran los ficheros a importar y el nombre de estos.

Para introducir al programa valores de ensayo tomados de otro banco o instalación, es necesario introducirlos en una hoja de cálculo con extensión “.XLS”, y siguiendo una estructura como la mostrada en la figura 7.2.

	A	B	C	D	E	F	G
1		RPM	P_{asp} (Bares)	P_{imp} (Bares)	Par (N·m)	Q (l/min)	
2		1600	-0,139273	0,086197	0,423827	39,443844	
3		1600	-0,11258	0,159586	0,433468	35,412975	
4		1600	-0,105575	0,182047	0,43207	33,667225	
5		1600	-0,09242	0,218512	0,424624	31,380857	
6		1700	-0,018241	0,674439	0,33127	5,469934	
7		1700	-0,013613	0,711771	0,32098	0	
8		1600	-0,02555	0,528723	0,352877	13,880367	
9		1600	-0,018796	0,557047	0,337545	10,586291	
10		1600	-0,020612	0,576701	0,32468	5,733153	
11		1600	-0,017968	0,600987	0,314228	0,965912	
12		1600	-0,008137	0,623481	0,303109	0	
13		1700	-0,064408	0,449506	0,431057	24,109428	
14		1700	-0,054049	0,519901	0,412447	22,669384	
15		1700	-0,037211	0,567415	0,391901	18,290831	
16		1700	-0,035276	0,588313	0,391723	16,858679	
17		1700	-0,030246	0,611335	0,37491	13,437775	
18		1700	-0,025173	0,627449	0,357461	10,846169	
19		1700	-0,018447	0,656826	0,341928	8,380448	
20							

Figura 7.2: Modelo de hoja de calculo de la que se podrían importar datos.

Como puede observarse en la figura 7.2, el programa puede recibir en un mismo fichero "*.XLS" datos referentes a distintas velocidades de giro, e igual se puede hacer con los ficheros de datos de extensión ".DAT". Esto consigue hacerlo, gracias a que el programa solamente posee una matriz de datos para todos los valores de ensayo, en la que la primera columna se corresponde al régimen de giro, la segunda a la presión de admisión, la tercera a la presión de la impulsión, la cuarta al par, y la quinta al caudal.

Cuando el programa recibe datos de la función "Cargar_Datos", lee el valor de la primera columna de la matriz de datos, identifica los distintos valores de régimen de giro y crea un vector formado por estos valores. De tal forma que cuando el programa necesita hacer uso de los datos referentes a un único régimen de giro, utiliza la indexación lógica para discriminar al resto de valores. Teniendo las siguientes expresiones para las variables de ensayo:

- Para la velocidad de giro tenemos el vector (Ec. 4):

$$RPM \Leftrightarrow Vector\ con\ las\ distintas\ velocidades\ de\ giro \tag{4}$$

- Para el caudal en l/min correspondiente al régimen de velocidades que se encuentre en la componente “i” de este vector, tenemos (Ec. 5):

$$datos(datos(:,1) == RPM(i), 5) \quad (5)$$

- Para la presión de aspiración en bares correspondiente al régimen de giro que se encuentra en la componente “i” del vector RPM, tenemos (Ec. 6):

$$datos(datos(:,1) == RPM(i), 2) \quad (6)$$

- Para la presión de impulsión en bares correspondiente al régimen de giro que se encuentra en la componente “i” del vector RPM, tenemos (Ec. 7):

$$datos(datos(:,1) == RPM(i), 3) \quad (7)$$

- Para el par en N·m correspondiente al régimen de giro que se encuentra en la componente “i” del vector RPM, tenemos (Ec. 8):

$$datos(datos(:,1) == RPM(i), 4) \quad (8)$$

Además de estos valores, para el cálculo y representación de los parámetros de interés de una bomba centrifuga, es necesario conocer ciertas características del fluido de trabajo y de la instalación en la que se realiza el ensayo. Para las constantes de fluido como son la densidad y la viscosidad, el programa utiliza la función “inputdlg” de Matlab, que muestra en pantalla una ventana emergente en la que el usuario puede indicárselas. Mientras que para el ingreso de los datos referente a la instalación, se vale de una GUI auxiliar de tipo modal (código en apartado “11.2.3.- Código de la GUI “dialginputimagen”“) , que muestra en pantalla una ventana con una imagen explicativa (figura 7.3), en la que el usuario puede introducir el diámetro de rodete, los diámetros de

las tuberías de impulsión y aspiración, y la diferencia de cota entre los puntos en los que se toma la presión de aspiración y de impulsión (ΔZ).

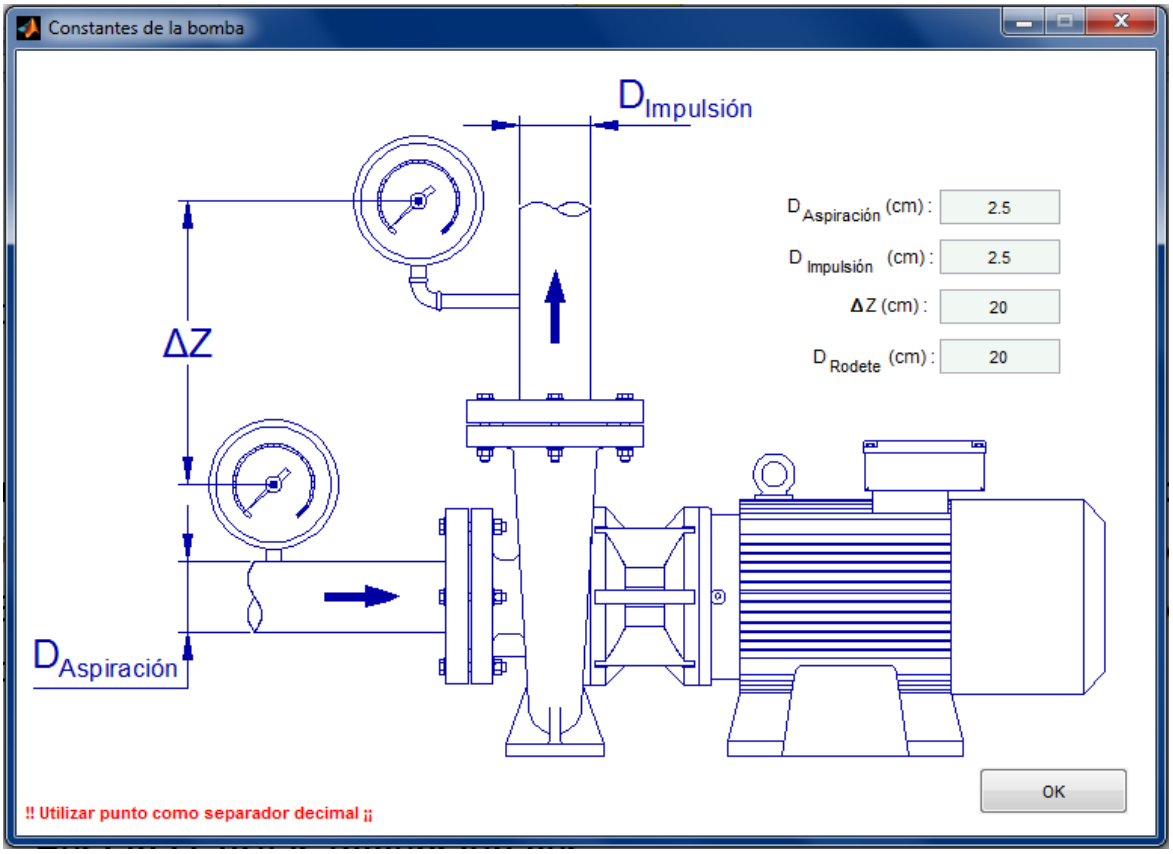


Figura 7.3: Vista de la ventana para introducir los valores de la instalación

De esta forma, el programa es capaz de adquirir nuevos datos de trabajo y representarle al usuario las gráficas que él necesita. No obstante, cuando el programa se inicia lo hace con unos valores ya introducidos, correspondientes al banco de ensayo de la EPS de Linares, que nos permiten analizar esta bomba sin necesidad de importar valores.

7.2.- Pestaña Curvas dimensionales.

Las pestaña “curvas dimensionales” es la que aparece activa cuando arrancamos la GUI, y nos permite representar e importar las gráficas más significativas en el estudio de bombas centrifugas, como son las de altura manométrica, potencia consumida, potencia hidráulica, par y rendimiento, frente al caudal proporcionado.

Esta pestaña permite su representación en dos o tres dimensiones pulsando el botón “Cambiar a grafica 3-D” (figura 7.4) cuando está mostrando gráficas 2-D, o el botón “Cambiar a grafica 2-D” cuando está mostrando gráficas 3-D. Puesto que las posibilidades que ofrece para cada tipo de representación son muy distintas, este apartado se ha subdividido en dos atendiendo a estas diferencias.

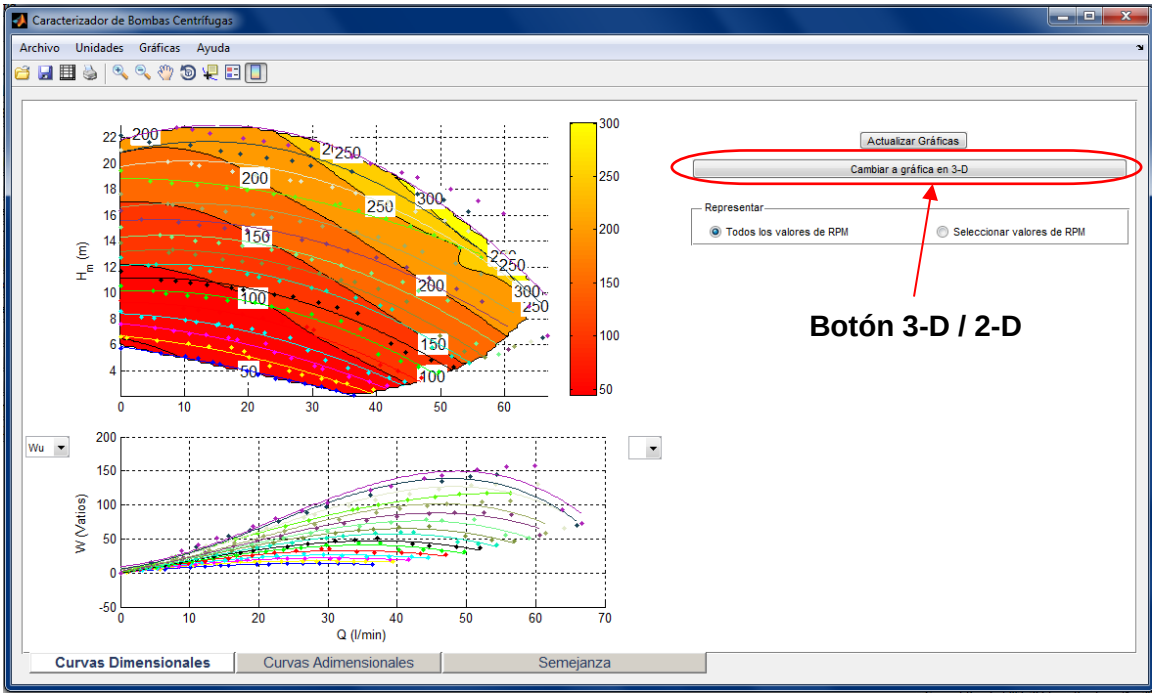


Figura 7.4: Vista de la GUI con la pestaña “curva dimensionales” activa

7.2.1.- Gráficas en dos dimensiones.

Es la opción activada por defecto de la pestaña “Curvas dimensionales”, y permite representar las curvas más utilizadas en selección de bombas. Para el cálculo de los parámetros representados en función de los valores de ensayo, el código hace uso de las siguientes expresiones:

- **Altura manométrica**. Aplicando las ecuaciones de conservación, tenemos:

$$H_m = \left[\left[\frac{V^2}{2 \cdot g} + Z + \frac{P}{\rho \cdot g}\right]\right]_E^S = \left[\left[\frac{Q^2 \cdot 4}{2\pi \cdot g \cdot D^2} + Z + \frac{P}{\rho \cdot g}\right]\right]_E^S \qquad (9)$$

Que expresado en función de las variables definidas en las ecuaciones 4, 5, 6 y 7 del apartado “7.1.- Importación de datos.”, queda (Ec. 10):

```
H=(datos(datos(:,1)==RPM(i),3)-datos(datos(:,1)==RPM(i),2))/(g*rho*...
    10^-5)+dz+(((4/(pi*Di^2)-4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
```
(10)

- **Potencia hidráulica.** Por definición, la potencia hidráulica o potencia útil es la cantidad de energía por unidad de tiempo que aporta la bomba al fluido, que escrito en forma de ecuación (Ec.11):

$$W_u = G\left[\left[\frac{V^2}{2} + g \cdot Z + \frac{P}{\rho}\right]\right]_E^S = \rho \cdot g \cdot Q \cdot H_m \quad (11)$$

Expresando esto en función de las variables del programa tenemos:

```
Wu=g*rho*(1/60000)*q.*h;
```
(12)

- **Potencia absorbida.** Se define la potencia absorbida como el par por la velocidad angular, que expresado en función de las variables de la GUI queda (Ec. 13):

```
Wb=datos(datos(:,1)==RPM(i),4).*datos(datos(:,1)==RPM(i),1)*(2*pi/60);
```
(13)

- **Rendimiento:** se obtiene del cociente entre la potencia útil o hidráulica y la potencia consumida por la bomba, que expresado en función de las variables de la GUI, queda (Ec. 14):

```
n=100*Wu./Wb
```
(14)

Con estas variables, un bucle y las funciones “plot” y “plotyy” (para las gráficas con dos ejes verticales) de Matlab, la GUI representa las gráficas en dos dimensiones.

Además, también hace uso de las funciones “griddata”, “gridfit” y “contourf”, para representar los contornos de isorendimiento e isoconsumo.

Para más detalle sobre el manejo de esta pestaña, ver el apartado “” de la ayuda del programa.

7.2.2.- Gráficas en tres dimensiones.

Las gráficas en tres dimensiones (figura 7.5), no se suelen mostrar en los catálogos de bombas ni en los libros de mecánica de fluidos. Sin embargo, resultan muy interesantes a la hora de evaluar la evolución del rendimiento o el consumo con el régimen de giro, la altura o el caudal, y pueden ser sobre todo para usuarios inexpertos más fáciles de interpretar que los isocontornos. Es por este motivo, y pese a que estas gráficas no se utilizan desde un punto de vista técnico, que decidí incluir un pequeño apartado dedicado a su representación.

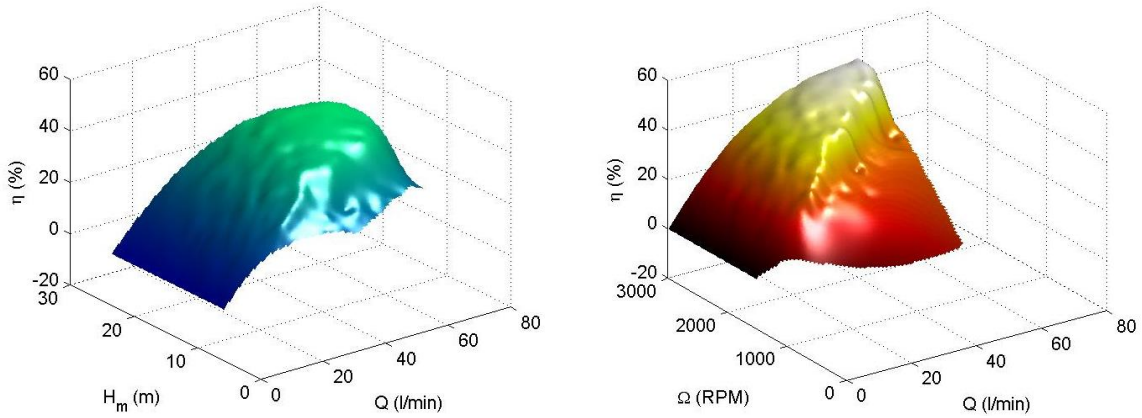


Figura 7.5: Gráficas en 3-D obtenidas con la GUI “Caracterizador de bombas centrífugas”

Para su obtención el código del programa hace uso de las funciones utilizadas para la representación de los isocontornos (“griddata”, “gridfit” y “meshgrid”), y de la función de Matlab “surf” para representar la superficie.

Como se puede observar en la figura 7.6, entre las distintas opciones que nos ofrece la aplicación en las gráficas en tres dimensiones tenemos una que dice: “Ajustar superficie a:”, y nos da las siguientes opciones:

- **Nube de puntos.** Esta opción ajusta la superficie que representa a los valores obtenidos de los puntos de ensayo.

- **Curvas interpoladas:** Esta opción ajusta la superficie que representa sobre los valores de las curvas de ajuste obtenidas de aplicar la función “polyfit” a los puntos obtenidos de los datos de ensayo. En algunos casos puede ofrecernos curvas mejor suavizadas que la opción “Nube de puntos”. Sin embargo, se ha de tener en cuenta que en el ajuste por mínimos cuadrados que se utiliza, se ajusta la curva de ajuste para minimizar la variación de altura, potencia y rendimiento con respecto al caudal, y esta no es la opción más acertada para este tipo de superficies.

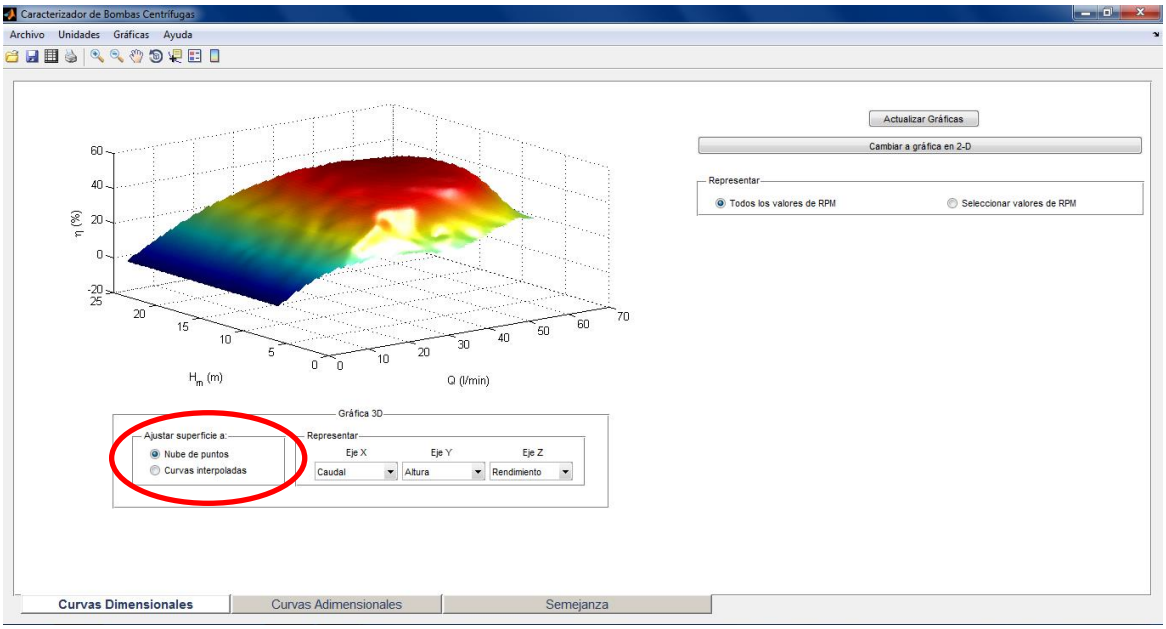


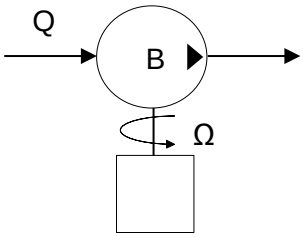
Figura 7.6: Gráficas en 3-D en el entorno de la GUI

Para más detalle sobre las posibilidades y manejo de las gráficas 3-D, consultar el apartado “” de la ayuda del programa.

7.3.- Pestaña curvas adimensionales.

El teorema de Π o de Vaschy-Buckingham permite, mediante la adimensionalización de las ecuaciones, identificar el mínimo número de parámetros que influyen en el comportamiento de una bomba, y a partir de estos prever el funcionamiento de esta u otra bomba semejante en situaciones de trabajo distintas a las ensayadas.

Para aplicar el teorema de Vaschy-Buckingham a una Bomba centrífuga, antes debemos de conocer que (figura 7.7):



$\Omega \equiv Velocidad\ de\ giro\ del\ impulsor\ [s^{-1}]$

$Q \equiv Caudal\ [M \cdot L^{-3}]$

$D \equiv Diámetro\ de\ rodete\ [L]$

Figura 7.7:

Y teniendo en cuenta que las variables que influyen en la altura proporcionada por una bomba centrífuga son (Ec. 15):

$$gH_m \equiv fun\ (Q, \Omega, \rho, \mu, D, l_1, l_2, \dots, l_n) \tag{15}$$

Tomamos, ρ como unidad de masa, Ω como unidad de tiempo y D como unidad de longitud, por ser las más significativas y comprobamos que son independientes (Ec. 16):

$$\begin{array}{c|ccc} & M & L & T \\ \rho & 1 & -3 & 0 \\ \Omega & 0 & 0 & -1 \\ D & 0 & 1 & 0 \end{array} = 1 \neq 0 \tag{16}$$

Según el teorema de Vaschy-Buckingham Π_{gH_m} va a depender de (Ec. 17):

$$\Pi_{gH_m} \equiv f_1(\Pi_Q, \Pi_\mu, \Pi_{l1}, \Pi_{l2}, \dots, \Pi_l) \tag{17}$$

Que son lo que se conocen como adimensionales o grupos Pi, y que tienen como expresión (Ec. 18, 19, 20, 21):

$$\Pi_{gH_m} \equiv \frac{gH_m}{\rho^a \cdot \Omega^b \cdot D^c} \Leftrightarrow \left[\frac{L^2}{T^2}\right] = \left[\frac{M}{L^3}\right]^a \cdot \left[\frac{1}{T}\right]^b \cdot [L]^c \Leftrightarrow \Pi_{gH_m} = \frac{gH_m}{\Omega^2 \cdot D^2} \tag{18}$$

$$\Pi_Q \equiv \frac{Q}{\rho^a \cdot \Omega^b \cdot D^c} \Leftrightarrow \left[\frac{L^3}{T}\right] = \left[\frac{M}{L^3}\right]^a \cdot \left[\frac{1}{T}\right]^b \cdot [L]^c \Leftrightarrow \Pi_Q = \frac{Q}{\Omega \cdot D^3} \quad (19)$$

$$\Pi_\mu \equiv \frac{\mu}{\rho^a \cdot \Omega^b \cdot D^c} \Leftrightarrow \left[\frac{M}{LT}\right] = \left[\frac{M}{L^3}\right]^a \cdot \left[\frac{1}{T}\right]^b \cdot [L]^c \Leftrightarrow \Pi_\mu \equiv \frac{\mu}{\rho \cdot \Omega \cdot D^2} \quad (20)$$

$$\Pi_{l1} = \frac{l_1}{D}, \Pi_{l2} = \frac{l_2}{D}, \dots, \Pi_{ln} = \frac{l_n}{D} \quad (21)$$

No obstante, siempre que haya semejanza geométrica los grupos pi de las longitudes que caracterizan la máquina van a permanecer constantes. Y en el estudio de bombas, el número de Reynolds normalmente alcanzan un valor suficientemente alto como para poder despreciar los efectos de la viscosidad, por lo que en principio, podemos considerar que:

$$\Pi_{gH_m} \equiv f_1(\Pi_Q) \quad (22)$$

Extendiendo este procedimiento al resto de variables de interés, tenemos para la potencia útil la Ec. 23 y 24, y para la potencia absorbida la Ec. 25 y 26:

$$\Pi_{W_u} \equiv f_2(\Pi_Q) \quad (23)$$

$$\Pi_{W_u} \equiv \frac{W_u}{\rho^a \cdot \Omega^b \cdot D^c} \Leftrightarrow \left[\frac{ML^2}{T^3}\right] = \left[\frac{M}{L^3}\right]^a \cdot \left[\frac{1}{T}\right]^b \cdot [L]^c \Leftrightarrow \Pi_{W_u} \equiv \frac{W_u}{\rho \cdot \Omega^3 \cdot D^5} \quad (24)$$

$$\Pi_{W_b} \equiv f_3(\Pi_Q) \quad (25)$$

$$\Pi_{W_b} \equiv \frac{W_b}{\rho^a \cdot \Omega^b \cdot D^c} \Leftrightarrow \left[\frac{ML^2}{T^3}\right] = \left[\frac{M}{L^3}\right]^a \cdot \left[\frac{1}{T}\right]^b \cdot [L]^c \Leftrightarrow \Pi_{W_b} \equiv \frac{W_b}{\rho \cdot \Omega^3 \cdot D^5} \quad (26)$$

Para el caso del caso del rendimiento, tenemos la Ec. 27:

$$\eta = \frac{W_u}{W_b} \equiv f_4(\Pi_Q) \tag{27}$$

Sustituyendo en estas expresiones las variables utilizadas por el código del programa para los distintos parámetros y utilizando un bucle, la GUI es capaz de representar los parámetros adimensionales de interés frente al grupo Π_Q, facilitándonos las gráficas que se muestran en la figura 7.8.

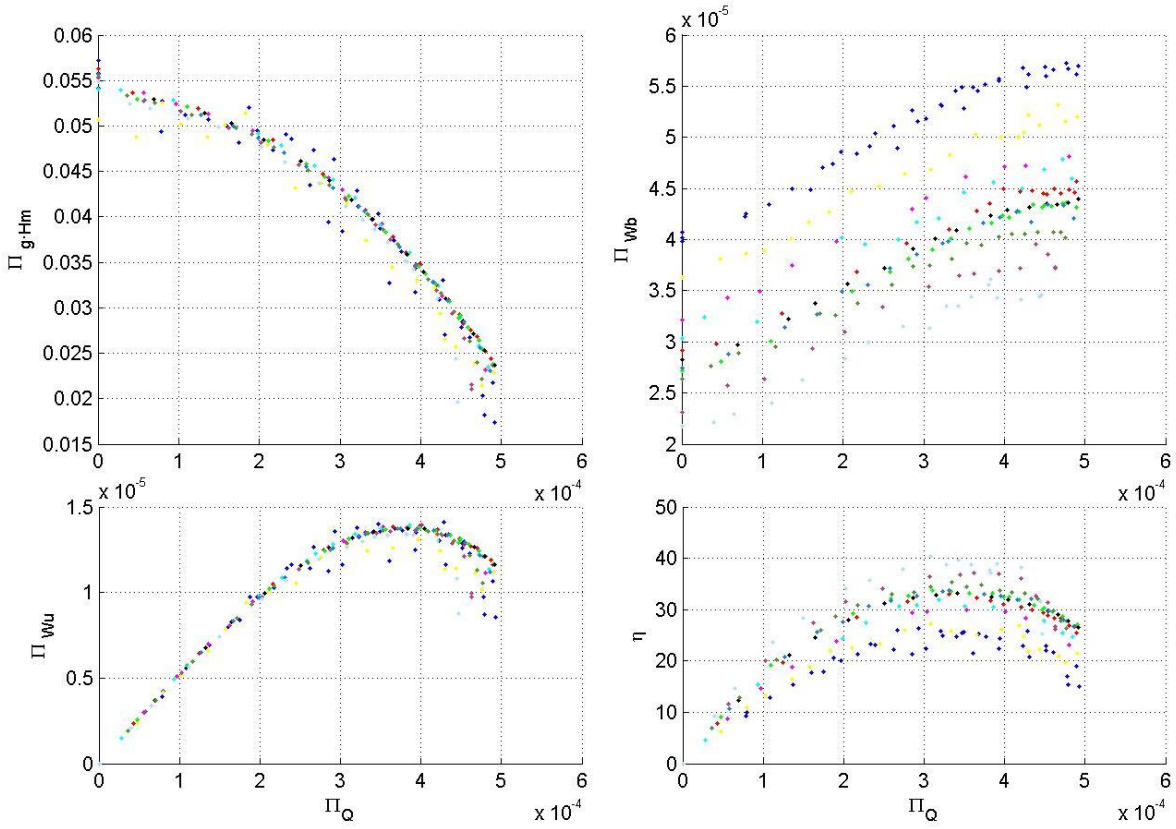


Figura 7.8: Representación de los distintos grupos adimensionales

Estas gráficas permiten comprobar que los grupos $\Pi_{g \cdot H_m}$ y Π_{w_u} dependen del grupo Π_Q, pero no de la velocidad de giro, por eso los puntos tienden a aglutinarse alrededor de una misma curva tal y como cabría esperar (los puntos de distinto color se corresponden a distinto régimen de giro). Sin embargo, en el caso del grupos Π_{W_b} y del rendimiento (η), se puede observar que los puntos de cada color se aglutinan entorno a una curva independiente, lo que quiere decir que el grupo adimensional correspondiente a la potencia absorbida, y el rendimiento, depende tanto de Π_Q como de la velocidad de giro.

Esta dependencia del adimensional de la potencia absorbida y del rendimiento de la velocidad de giro, nos demuestra que despreciar los efectos de la viscosidad es una buena aproximación para la altura manométrica y para la potencia útil, pero que no lo es para la potencia consumida y el rendimiento. Lo que es normal si se tiene en cuenta que la potencia consumida por una bomba, viene condicionada por las pérdidas viscosas que se producen en ella. Y puesto que el rendimiento no deja de ser una herramienta para cuantificar las pérdidas de potencia que se tienen, incluyéndose dentro de estas pérdidas las pérdidas viscosas, es normal que también dependa de Π_Q.

El objetivo que pretende esta pestaña, es precisamente demostrar al alumno que las leyes de semejanza (que se estudiarán con más detenimiento en el siguiente apartado) son útiles para predecir la altura y la potencia hidráulica que proporcionaría una bomba semejante a la ensayada a distintas velocidades de giro, pero que no lo son tanto para predecir el consumo o rendimiento que va a tener, ya que para esto tendríamos que mantener el número de Reynolds del ensayo igual al que tendríamos en régimen de giro para el que queremos predecir el comportamiento de la bomba, lo que nos obligaría a realizar ensayos con un fluido de viscosidad y densidad difícilmente conseguibles.

El uso de esta pestaña es bastante intuitivo, no obstante, para más detalles de manejo de ésta, ver el apartado “” de la ayuda de programa.

7.4.- Pestaña semejanza.

Esta pestaña está dedicada al estudio de semejanza en bombas centrifugas, permitiéndonos obtener las curvas de altura manométrica y potencia útil que tendría la bomba ensayada a un régimen de giro distinto al que hemos ensayado, en incluso para una bomba de distinto tamaño, siempre que se cumplan las condiciones de semejanza geométrica, es decir, que todas las dimensiones de la bomba ensayada sean proporcionales a las que tiene la bomba para la que se pretenden predecir las curvas de funcionamiento.

Se dedujo en el apartado anterior que $\Pi_{gH_m} \equiv f_1(\Pi_Q)$ y $\Pi_{W_u} \equiv f_1(\Pi_Q)$, por lo que si Π_Q permanece constante también lo hará Π_{gH_m}. Si además tenemos en cuenta que se trata de mismo fluido y que por tanto la densidad (ρ) permanece constante, se puede deducir las ecuaciones 28, 29 y 30:

$$\Pi_Q = \frac{Q_1}{\Omega_1 \cdot D_1^3} = \frac{Q_2}{\Omega_2 \cdot D_2^3} \Leftrightarrow \mathrm{Q}_2 = \mathrm{Q}_1 \cdot \left(\frac{\Omega_2}{\Omega_1}\right) \cdot \left(\frac{\mathrm{D}_2}{\mathrm{D}_1}\right)^3 \quad (28)$$

$$\Pi_{gH_m} = \frac{gH_{m,1}}{{\Omega_1}^2 \cdot D_1^2} = \frac{gH_{m,2}}{{\Omega_2}^2 \cdot D_2^2} \Leftrightarrow gH_{m,2} = gH_{m1}\left(\frac{\Omega_2}{\Omega_1}\right)^2 \cdot \left(\frac{\mathrm{D}_2}{\mathrm{D}_1}\right)^2 \quad (29)$$

$$\Pi_{W_u} \equiv \frac{W_{u,1}}{\rho \cdot {\Omega_1}^3 \cdot D_1^5} = \frac{W_{u,2}}{\rho \cdot {\Omega_1}^3 \cdot D_2^5} \Leftrightarrow W_{u,2} = W_{u,1}\left(\frac{\Omega_2}{\Omega_1}\right)^3 \cdot \left(\frac{\mathrm{D}_2}{\mathrm{D}_1}\right)^5 \quad (30)$$

Las expresiones que utiliza el código del programa para obtener el caudal, la altura y la potencia hidráulica de la bomba semejante en función del valor que tienen estos para la curva de ensayo que elegimos como referencia son:

- Para el caudal, Ec. 31

```
q*((n1/RPMo(i)))*((D1/Dr)^3)*qx;
```
(31)

- Para la altura manométrica, Ec. 32

```
h*((n1/RPMo(i))^2)*((D1/Dr)^2)*px
```
(32)

- Para la potencia útil, Ec. 33

```
wu*((n1/RPMo(i))^3)*((D1/Dr)^5);
```
(33)

Donde “qx” y “px” son factores de conversión, “D1” es el diámetro de rodete de la máquina semejante, “Dr” el diámetro de rodete de la máquina ensayada, “n1” el régimen de giro para el que queremos obtener las curvas, y “RPMo(i)” el régimen de giro elegido como referencia. Para más detalle consultar el callback “representar” del código principal del programa, apartado “11.2.4.- Código de la GUI principal “Caracterizador de bombas centrífugas”.”.

Nota: No se debe confundir una bomba geométricamente semejante con distinto diámetro de rodete (todas las dimensiones multiplicadas por el factor de escala D_1/D_2), con un rodete recortado, que es a groso modo, un rodete al que se le ha reducido el diámetro sin tocar el resto de dimensiones. Los fabricantes de bombas, suelen diseñar un rodete para que proporcione el máximo rendimiento posible en un determinado punto de funcionamiento, y recortan el diámetro para cubrir la gama de altura y caudal que se encuentran por debajo de las que este ofrece. Esta disminución de diámetro, conlleva una reducción del rendimiento, así que cuando esta reducción justifica los gastos de diseñar un nuevo rodete, diseñan uno más pequeño y eficiente. Según el libro de hidráulica de bombas ideal (disponible en: http://www.bombasideal.com/libro-de-hidraulica/), la relaciones para la altura y caudal entre un rodete que ha sido recortado y del que deriva, se corresponde con la ecuaciones 34 y 35, distintas de las que tenemos entre bombas semejantes, Ec. 29 y 28.

$$\frac{H_2}{H_1} = \left(\frac{D_2}{D_1}\right)^2 \tag{34}$$

$$\frac{Q_2}{Q_1} = \left(\frac{D_2}{D_1}\right)^2 \tag{35}$$

Esta pestaña consta de un grupo de botones, figura 7.9, que permiten elegir dos modos de funcionamiento que cambia significativamente el comportamiento de la pestaña:

- **Opción "Obtener curvas por semejanza"**, basándose en las ecuaciones anteriormente descritas y en la curva de ensayo elegida como referencia, permite representar el comportamiento que tendrían dos bombas de tamaño distinto y girando a una velocidad distinta de las ensayadas.
- **Opción "Comparar puntos experimentales con curvas obtenidas por semejanza",** esta opción representa los puntos experimentales obtenidos para la atura manométrica y la potencia útil a la velocidad o las velocidades de giro ensayadas que le especifiquemos, y nos representa las curvas de ajuste que tendrían según se deduce del estudio de semejanza y usando como curva de referencia el régimen ensayado que le especifiquemos. Esto nos permite evaluar cualitativamente la separación entre la realidad y lo que prevé el análisis por semejanza, Constituyendo una vez más, una

herramienta que permite a los alumnos valorar la utilidad de los teoremas estudiados en el aula.

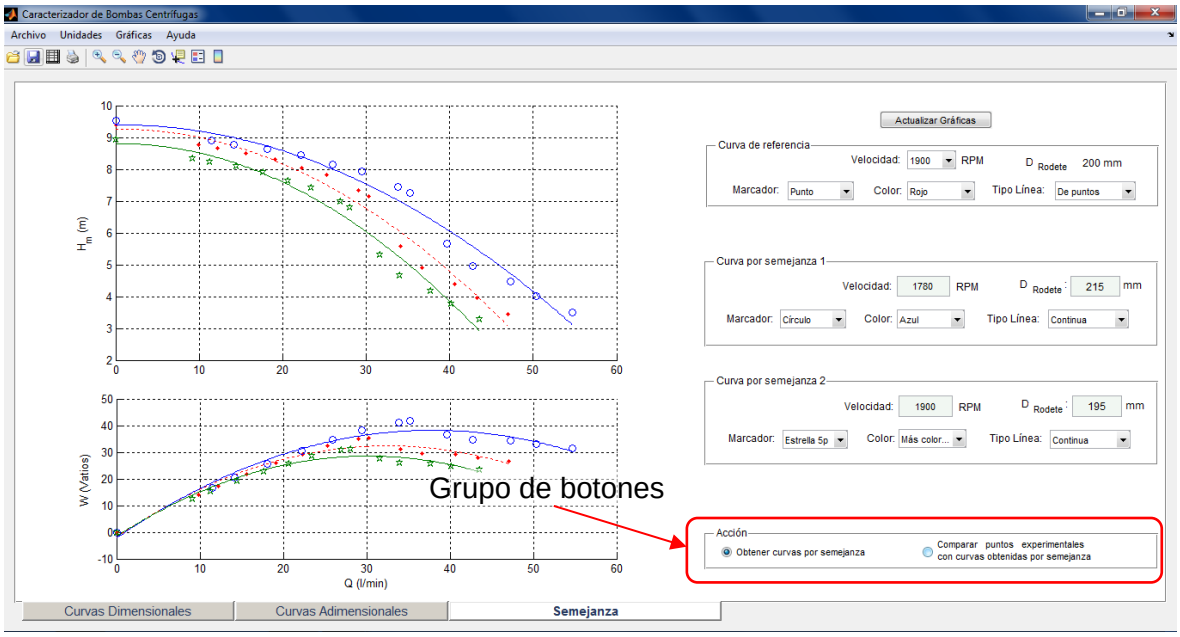


Figura 7.9: Vista de la pestaña "Semejanza"

7.5.- Exportar gráficas.

Todas las utilidades descritas en los apartados anteriores sirven para demostrar la capacidad que posee la interfaz "Caracterizador de bombas centrífugas" para el análisis y caracterización de bombas. No obstante, resulta de gran utilidad disponer de una herramienta que permita al usuario exportar las gráficas que puede ver en pantalla a otros programas. Si bien es cierto que éstas se podrían tomar realizando una captura de pantalla, ofrecería una calidad de imagen baja y sobre todo daría una mala imagen de la interfaz desarrollada

Para exportar la gráficas, el código de la interfaz se vale de las funciones "copyobj" y "saveas". La función "copyobj", copia la gráfica del "axes" de la GUI a una nueva ventana, mientras que la función "saveas" se encarga de guardar la gráfica de esta nueva ventana en la dirección y con la extensión que el usuario le ha proporcionado. Para más detalle ver la función "guardargrafica_Callback" , en el apartado "11.2.4.- Código de la GUI principal "Caracterizador de bombas centrífugas".".

Los formatos que he habilitado para exportar las gráficas son: JPG, TIF, BMP, FIG, PBM, PDF, PNG y PPM. Siendo la extensión ".fig" especialmente interesante para

aquellos usuarios que posean Matlab, ya que pueden ser abiertas desde este y modificadas en tamaño, grosor de línea, tipo de letra, resolución,… utilizando el “Property editor-figure”.

El tamaño que se obtiene al exportar las gráficas a formatos de imagen responde al que tiene una imagen mediana en un documento de texto (figura 7.10), no teniendo suficiente resolución como para aumentarla. Por lo que para obtener una imagen de mayor tamaño y resolución, aquellos usuarios que utilicen la GUI y no dispongan de licencia de Matlab, se verán obligados a exportar la imagen a PDF, formato al que la función “saveas” de Matlab le da bastante más resolución, y extraer la imagen de este con las herramientas que nos proporcionan los programas que trabajan con ficheros PDF.

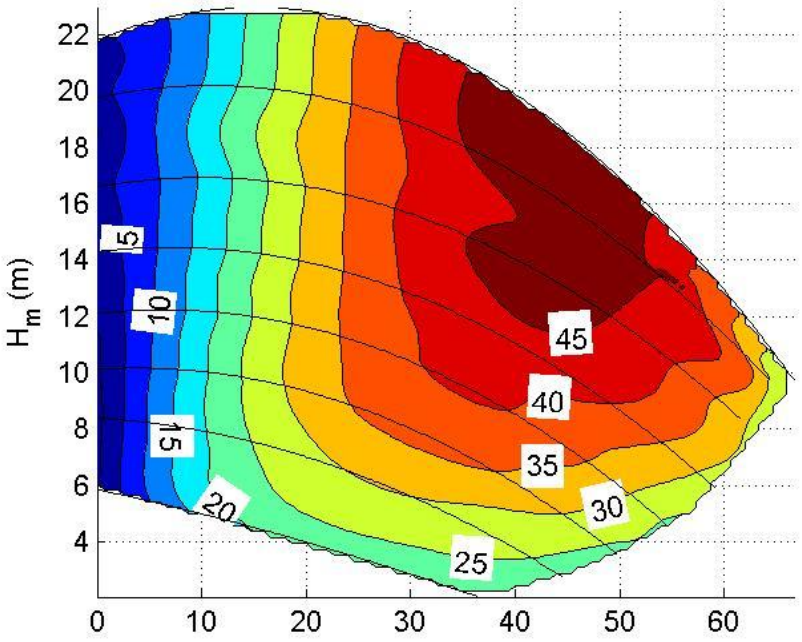


Figura 7.10: Grafica exportada a formato JPG

Quiero resaltar en este apartado, que por algún motivo la función “copyas” no es capaz de copiar las gráficas con dos ejes verticales, y por tanto, al exportar estas gráficas solo se exportará el contenido referente al eje izquierdo.

8.- CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO.

La finalidad de este trabajo fin de grado era lograr una aplicación gráfica que permitiera analizar y caracterizar una bomba centrífuga, en concreto, la bomba centrífuga computarizada de la que disponemos en el laboratorio de fluidos y termotecnia de la Escuela.

En lo que respecta al cumplimiento de esto objetivos, tengo que decir que con la opción de importar datos desde ficheros de extensión ".xls", la aplicación adquiere la capacidad de analizar bombas centrífugas ensayadas en cualquier banco de ensayo o instalación, siempre que ésta permita medir la velocidad de giro, la presión en las bridas de aspiración e impulsión, el caudal, y el par proporcionado por el motor. Siendo por tanto la aplicación, capaz de analizar bombas más allá de lo que se propuso en los objetivos iniciales del trabajo.

Además de poder utilizar la aplicación gráfica para el análisis de bombas centrífugas, muchas de las posibilidades que ofrece pueden ser aplicadas también al estudio de bombas de hélice y helicocentrífugas, lo que también contribuye una mejora respecto a los objetivos iniciales de este TFG.

Aunque la interfaz cumple con las funciones de representación para las que fue diseñada, existen ciertos factores que podrían implementarse para aumentar su capacidad de trabajo, como pueden ser:

- **Añadir una pestaña destinada a la obtención de las curvas de la NPSHr**. Las curvas de cavitación son fundamentales para la selección de bombas, prueba de ello es que todos los fabricantes las proporcionan en su catálogo. Sin embargo, debido a la mayor dificultad que ofrecen para su ensayo, y sobre todo por el tiempo que requiere la adaptación de la GUI para poder analizarlas, no se han incluido.
- **Lograr que la GUI proporcione valores calculados**. Como puede observarse en la figura 8.1, en el cuadro para la selección de las velocidades de giro a representar, se dejó hueco para dos columnas más. La idea era, que en la primera columna figurara la velocidad específica de la bomba para cada una de las filas, mientras que en la segunda columna, aparecería un icono "f_x" por cada fila (figura 8.2), de tal forma que cuando se hiciera clic sobre uno de ellos se nos abriera un cuadro de diálogo en el que se mostrara los polinomios de ajuste, coeficiente de bondad de ajuste, y

máximo y mínimo relativo para las curvas de $H_m(Q)$, $W_u(Q)$, $W_b(Q)$, $T(Q)$ y $\eta(Q)$, correspondiente a ese régimen de giro.

Debido al tiempo disponible, estás dos columnas no se llegaron a implementar.

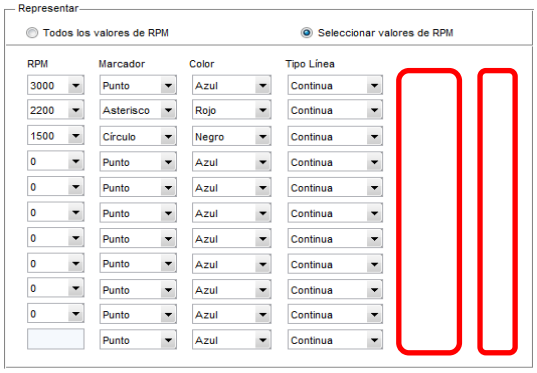


Figura 8.1: Situación actual

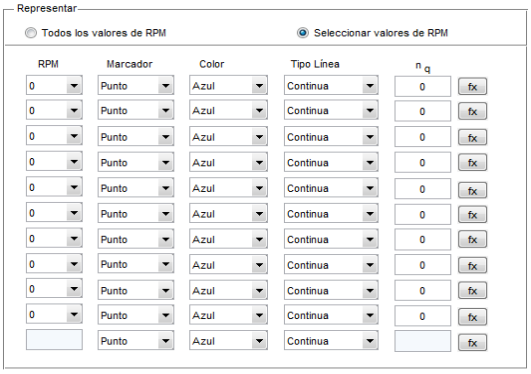


Figura 8.2: Posible mejora

- **Mejorar el menú ayuda**. Añadiendo algunos complementos que ayuden a entender el funcionamiento de las bombas centrifugas, el proceso de adquisición de datos, y sobretodo que proporcionen información que enlace los cálculos que realiza el programa con la teoría de bombas. Si bien esto no es algo imprescindible, podría constituir una ayuda en el ámbito de la docencia.
- **Inserción de leyenda**. La GUI permite insertar leyenda a las gráficas, aunque en ésta no figura el valor del régimen de giro con el que se corresponde, por lo que son de poca utilidad. Durante la realización de esta interfaz, implementé unas líneas en el código que permitían precisamente que esta información estuviese disponible, sin embargo, provocaron tal ralentización en la ejecución del programa que tuve que eliminarlas. Esto no quiere decir que no se pueda realizar, sino que habría que estudiar una forma alternativa de hacerlo.

Otro factor que sería interesante estudiar, aunque escapa completamente de los objetivos de este proyecto y sobre todo de nuestras posibilidades, es el ensayo de una bomba centrifuga de distinto tamaño de la que tenemos. Con esto se pueden comparar los resultado procedentes de dos estudios, teórico y experimental, para poder validar las curvas obtenidas por semejanza entre dos bombas de tamaño distinto y geométricamente semejantes. Sin embargo, tenemos una dificulta debido a que sería necesario encontrar una bomba realizada a escala de la que tenemos, y puesto que es dificil encontrar una

bomba con estas características en el mercado, nos veríamos obligados a fabricar una única unidad a medida, lo que supondría un coste añadido.

Durante el desarrollo de una interfaz de usuario van surgiendo nuevas ideas que contribuirían a mejorar la base que se tiene, prueba de ello son las actualizaciones y nuevas versiones que sacan al mercado los grandes fabricantes de software. Sin embargo, en un TFG en el que el tiempo es limitado y no se puede sacar actualizaciones anuales, es necesario alcanzar un compromiso entre la introducción de nueva mejoras y la consecución de los objetivos fijados. Algo con lo que cumple la GUI “Caracterizador de Bombas centrífugas”, pese a todos los aspectos en los que se podría mejorar.

9.- GLOSARIO DE TÉRMINOS.

GUI: Graphical User Interface → interfaz gráfica de usuario. Es un programa que utiliza una ventana con distintos elementos (Botones, menús, tablas, etc.) para interactuar con el usuario.

GUIDE: Graphical User Interface Development Environment → Entorno para el desarrollo de interfaces gráficas de usuario. Es el entorno de Matlab para el desarrollo de GUIs.

NPSH: Net Positive Suction Head → Altura neta positive en la aspiración. Las curvas de la NPSH, o curvas de cavitación, permiten conocer si una bomba va a sufrir cavitación, en una determinada instalación.

n_q: Velocidad específica.

MCR: Matlab Compiler Runtime. Es un conjunto independiente de bibliotecas compartida que permiten la ejecución de aplicaciones y componentes creados en Matlab en ordenadores que no lo tienen instalado

Script: fichero de texto (en Matlab con extensión ".m"), en el que se introducen una serie de secuencias u órdenes que constituyen un programa o función. En Matlab se puede realizar una GUI usando solo un Script, sin embargo el fichero "*.m" creado por el entorno GUIDE para las GUIs, no se puede considerar un Script debido a su particular estructura.

TFG: Trabajo Fin de Grado

10.- BIBLIOGRÁFIA.

1) J. V. Romero, M. D. Roselló, R. Zalaya. “Fundamentos matemáticos para la ingeniería con MATLAB”, Univ. Politécnica de Valencia, 2002. (Vista previa en: http://books.google.es/books?id=5yFgI6NTFgwC&printsec=frontcover&hl=es#v=onepage&q&f=true)

2) Scott T. Smith. “MATLAB: Advanced GUI Development”. Dog Ear Publishing, 2006 (Vista previa en: http://books.google.es/books?id=lSGEIZscN_gC&printsec=frontcover&hl=es#v=onepage&q&f=true)

3) Stephen Chapman. “MATLAB Programming for Engineers”. Cengage Learning, 2007. (Vista previa en: http://books.google.es/books?id=fhpotPvv7v8C&printsec=frontcover&hl=es#v=onepage&q&f=true)

4) J. García, J. I. Rodríguez, J. Vidal. “Aprenda Matlab 7.0 como si estuviera en primero”. Universidad Politécnica de Madrid, 2005. (Disponible en: http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab70/matlab70primero.pdf)

5) A. García Prats. “Hidráulica: prácticas de laboratorio”. Univ. Politécnica de Valencia, 2006. (Vista previa en: http://books.google.es/books?id=CnEPGohcQWsC&printsec=frontcover&hl=es#v=onepage&q&f=true)

6) J. Agüera Soriano. “Mecánica de Fluidos Incompresibles y Turbomáquinas hidráulicas”. 5ª edición. Ciencia 3 S.A. 2002. (Disponible en: http://www.uco.es/termodinamica/)

7) Paresh Girdhar, O. Moniz. “Practical Centrifugal Pumps”. Elsevier, 2011. (Vista previa en: http://books.google.es/books?id=3RjnmvQSFvcC&printsec=frontcover&hl=es#v=onepage&q&f=true)

8) Kenneth J. y el cuerpo de redactores de Chemical, “Bombas. Selección, uso y mantenimiento”. McGraw-Hill, 1989.

9) Claudio Mataix, “Mecánica de fluidos y máquinas hidráulicas”. Marcombo, 2004.

10.1.- Páginas webs visitadas:

1) www.vilmat-pres.com/pbombasCatalogo.aspx

2) www.tecnicafluidos.es

3) www.grundfos.com

4) www.bombasideal.com

5) www.grundfos.com

6) http://ocwus.us.es

7) www.mathworks.es

8) www.bocasa.com.mx/es/images/bronce.jpg

9) www5.uva.es/guia_docente/uploads/2011/375/51412/1/Documento15.pdf

10) http://i00.i.aliimg.com/img/pb/211/221/418/418221211_057.JPG

11) http://biblioteca.sena.edu.co/exlibris/aleph/u21_1/alephe/www_f_spa/icon/15067/vol17/volumen17.html#

12) https://www.septicsewagepumps.com/pages/comparison

13) http://www.erica.es/web/wp-content/gallery/empaquetaduras-tecnicas/EMPAQ.-018.JPG

14) http://energiare.blogspot.com.es/2013/08/energia-hidroelectrica-pequenas.html

15) http://dim.usal.es/eps/mmt/?p=1096

16) www.flygt.com

17) www.virtualplant.net/vptd/includes/equipo.php?id_equipo=P-0101#

18) www.edibon.com

11.- ANEXOS A LA MEMORIA.

11.1.- Anexo 1. Ayuda del programa.

Este apartado, dedicado al usuario de la GUI, explica el modo de uso de la aplicación “Caracterizador de bombas centrífugas”, constituyendo en sí mismo un manual de instrucciones. Debido a esto, el documento no sigue una estructura lineal, de tal forma que el usuario que busque consultar una duda en un apartado concreto, pueda resolverla sin tener que recurrir a consultar todos los apartados anteriores.

Aunque esta ayuda es un anexo a la memoria del trabajo fin de grado, está pensada para que se acceda a ella desde la propia aplicación. Y es por esto, que no explica ni el proceso para instalarla en ordenadores que no poseen Matlab, ni para su ejecución en ordenadores que sí que dispongan de una versión instalada. Dejando la explicación de ese proceso al fichero “Léame.txt” que se encuentra en este DVD.

11.1.1.- Importar datos.

Cuando abrimos la aplicación, vemos que ya existen unas gráficas en la pantalla, y que todas las opciones que ofrece son completamente funcionales. Esto se debe, a que al iniciarse la GUI, carga los datos que se han tomado durante la realización de este trabajo fin de grado (que se encuentran en la carpeta del programa).

Para importar datos distintos de estos, debemos hacer clic en Archivo → Datos curvas características → Importar ficheros de datos (figura 11.1), o hacer clic en el icono “Importar Datos” (figura 11.2).

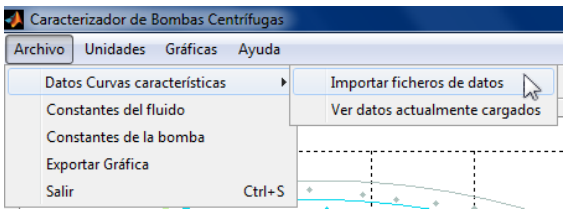


Figura 11.1:

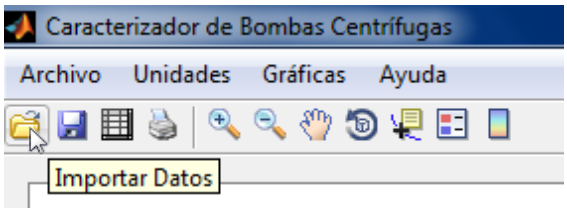


Figura 11.2:

Independientemente de la opción que elijamos, se nos abrirá una ventana para indicar el fichero o los ficheros que queremos importar. Esta ventana (figura 11.3) funciona igual que la ventana “Abrir” que utilizan los programas de uso común. Se busca la carpeta en la que se encuentra el fichero que queremos importar, y se selecciona el

fichero haciendo clic con el ratón. Para seleccionar más de un fichero, se puede hacer una ventana con el ratón, que comprenda aquellos ficheros que queremos seleccionar, o hacer clic en los ficheros que queremos a la vez que se mantiene la tecla “control” pulsada (la selección múltiple solo permite seleccionar ficheros que se encuentran en el mismo directorio).

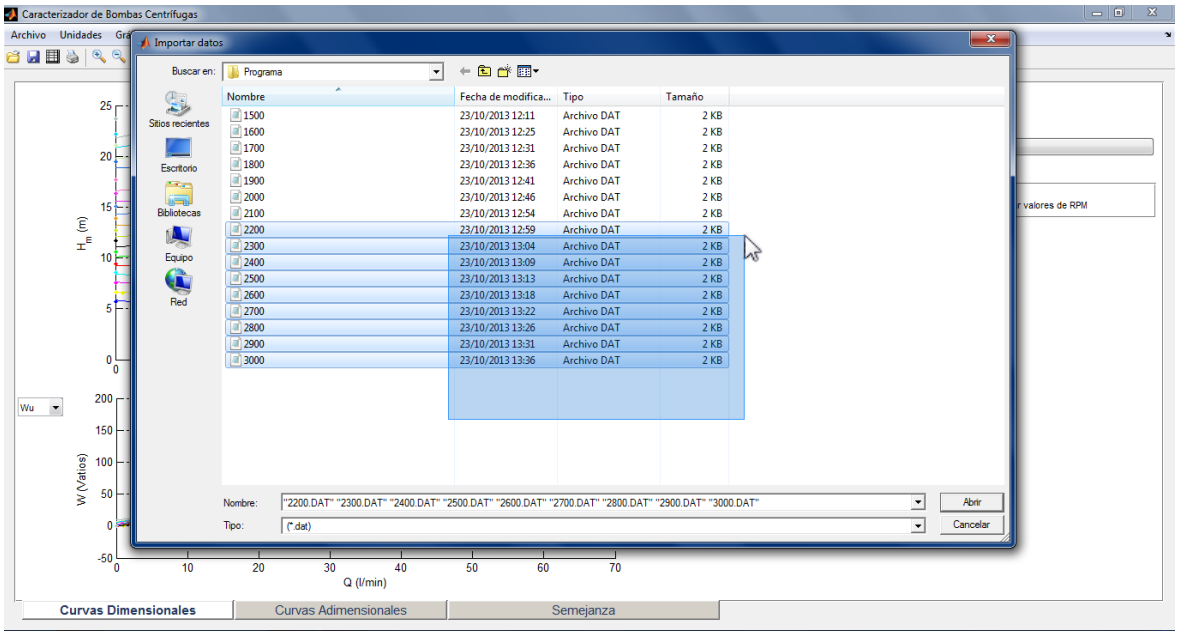


Figura 11.3: Importar datos.

Cuando se hayan seleccionado los ficheros que se quieren importar, se hará clic en abrir y se esperará a que los cargue, figura 11.4. (No se debe pulsar la “X” de esta ventana).

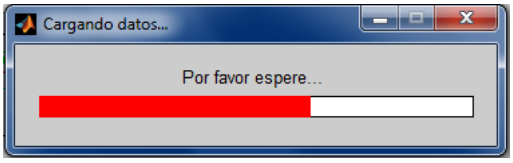


Figura 11.4: Ventana con Barra de progreso

Al elegir los ficheros de datos se ha de tener en cuenta que:

- No es necesario que en un fichero de datos, haya un único régimen de giro ensayado.
- Se pueden cargar a la vez, más de un fichero que contenga datos del mismo régimen de giro.
- No es necesario que los ficheros de datos tengan nombres especiales, es más, el nombre del fichero puede contener espacios y la letra ñ.
- Pada cada régimen de giro ensayado se deben de tomar al menos cuatro mediciones.

La GUI está diseñada para permitir la importación de datos provenientes de un banco de ensayo distinto al de la bomba computarizada. Por ello, es capaz de importar datos de una o varias hojas de cálculo con extensión “.xls”, siempre que éstas tengan la estructura y unidades que se muestran en la figura 15.5. (Solo se importarán los datos de la primera hoja).

	A	B	C	D	E	F
1		RPM	P_{adm} (Bares)	P_{mp} (Bares)	T (N·m)	Q (l/min)
2		1500	-0,11997	0,067787	0,377487	36,473122
3		1500	-0,079097	0,191564	0,383072	28,832111
4		1500	-0,06335	0,250309	0,375011	24,177164
5		1500	-0,052366	0,289416	0,368396	21,449535
6		1500	-0,032034	0,402943	0,341004	14,478582
7		1500	-0,026036	0,441014	0,331759	12,183938
8		1500	-0,009195	0,532709	0,279816	0
9		1600	-0,139273	0,086197	0,423827	39,443844
10		1600	-0,105575	0,182047	0,43207	33,667225
11		1600	-0,080709	0,25933	0,420023	28,375912
12		1600	-0,036601	0,512528	0,368477	17,348822
13		1600	-0,02555	0,528723	0,352877	13,880367
14		1600	-0,017968	0,600987	0,314228	0,965912
15		1600	-0,008137	0,623481	0,303109	0
16		1700	-0,13607	0,155192	0,465438	38,969746
17		1700	-0,11584	0,211672	0,45977	35,681751
18		1700	-0,054049	0,519901	0,412447	22,669384
19		1700	-0,030246	0,611335	0,37491	13,437775
20		1700	-0,025173	0,627449	0,357461	10,846169
21		1700	-0,018447	0,656826	0,341928	8,380448
22		1700	-0,013613	0,711771	0,32098	0

Figura 11.5:

Para importar datos desde ficheros con extensión “.xls”, se debe seleccionar el tipo (*.xls) en la ventana de importar datos (ver figura 15.6). Tenga paciencia, los ficheros con extensión “.xls” son algo más lentos de importar que los “.dat”.

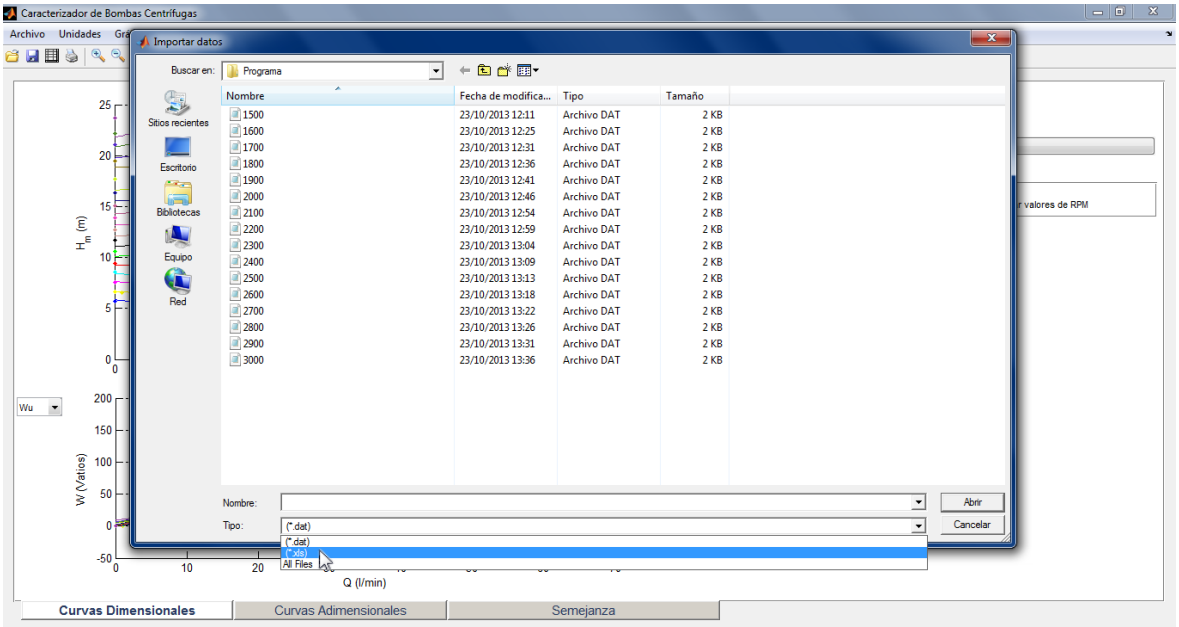


Figura 11.6:

Nota: cuando se importan nuevos datos el programa solo considera estos valores, obviando los que se cargaron por defecto, o los que hubiésemos importado antes. Para conocer con más detalle los datos que se encuentran cargados, hacer clic en

Archivo → Datos curvas características → Ver datos actualmente cargados (figura 11.7), o hacer clic en el icono “Ver datos cargados” (figura 11.8).

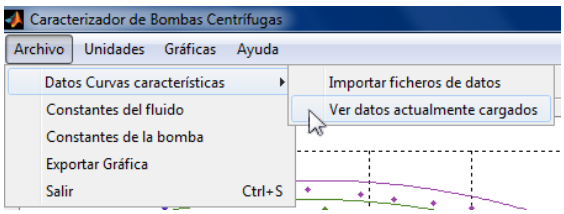


Figura 11.7:

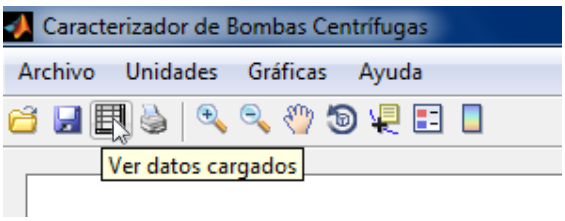


Figura 11.8:

De esta forma se nos abrirá un cuadro de diálogo, como el que se muestra en la figura 11.9, en el que aparece una tabla con los valores que esté utilizando el programa.

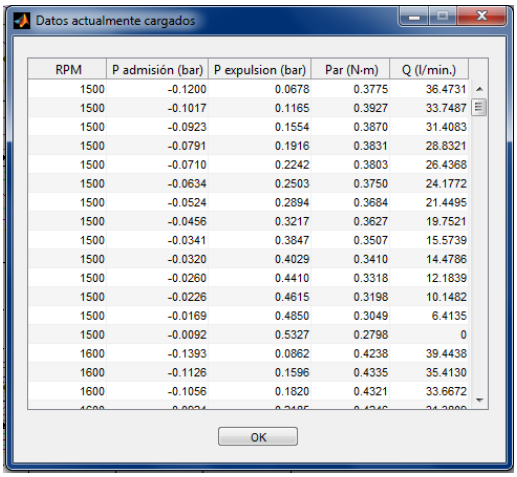


Figura 11.9:

Si los datos importados se han obtenido de ensayar con un fluido distinto del agua, se deberá de hacer clic en Archivo → Constantes del fluido, y especificar los nuevos valores en la ventana que nos aparece (figura 11.10).

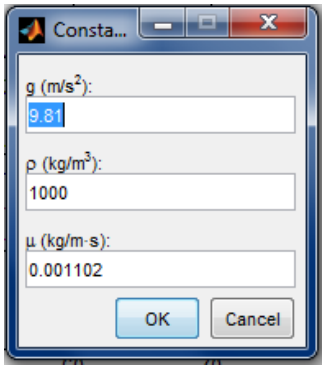


Figura 11.10:

De igual forma, si se utilizara un banco de ensayo distinto al de la bomba computarizada, habría que hacer clic en Archivo → Constantes de la bomba, y en la

ventana que nos aparece (figura 11.11), especificar el diámetro de las tuberías de impulsión y aspiración, la diferencia de cota entre los puntos en los que se miden la presión de aspiración e impulsión, y el diámetro del rodete de la bomba.

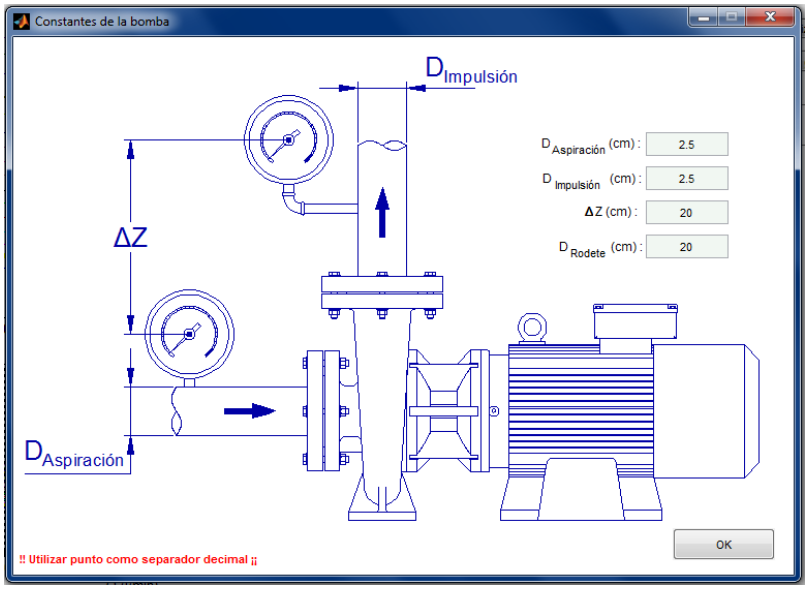


Figura 11.11: Ventana de adquisición de datos referentes a la instalación

11.1.2.- Unidades.

Esta interfaz gráfica permite cambiar las unidades que aparecen en las distintas gráficas. Para ello, debemos irnos al menú “Unidades”, seleccionar la magnitud que queremos que nos cambie de unidad, y hacer clic en la unidad que queremos que utilice. (Ver ejemplo en figura 11.12).

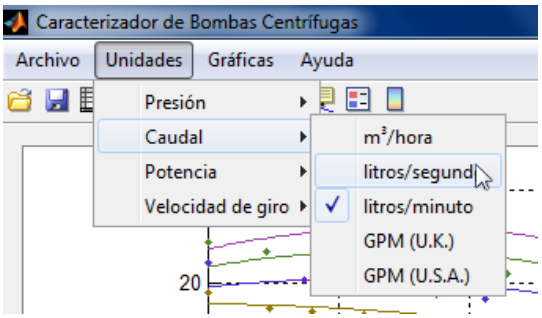


Figura 11.12

Nota: Este cambio de unidades solo afecta a lo que vemos en pantalla. Independientemente de las unidades que elijamos, seguirá siendo necesario importar los datos con las unidades que se especificaron el en apartado anterior, es decir, el caudal en l/min, las presiones en bares, el régimen de giro en rpm. y el par en N·m.

11.1.3.- Orden de ajuste.

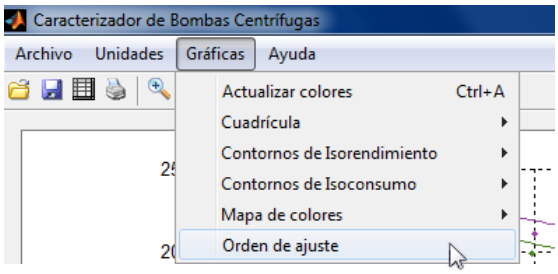


Figura 11.13

Para cambiar el orden de los polinomios utilizados para el ajuste de las curvas de $H_m(Q)$, $W_u(Q)$, $W_b(Q)$, $T(Q)$ y $\eta(Q)$, se ha de hacer clic en el menú Gráficas → Orden de ajuste (figura 11.13), y en el cuadro de dialogo que se nos abre, especificar el grado de los distintos polinomios de ajuste, figura 11.14.

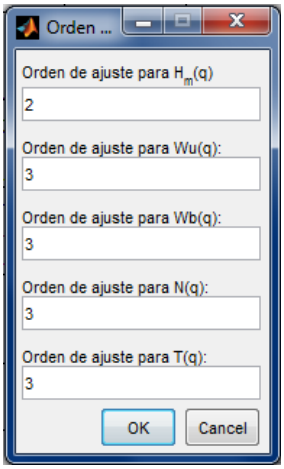


Figura 11.14

El orden de ajuste utilizado para los parámetros adimensionales, será el mismo que el elegido para su homólogo dimensional.

11.1.4.- Curvas características.

Las curvas características de la bomba se representan en la pestaña “Curvas Dimensionales”, que es la que se encuentra activa cuando arrancamos el programa. Si por algún motivo no es esta la pestaña activa, se hará clic en el botón “Curvas Dimensionales” para activarla.

Las gráficas que nos muestra se corresponden con los datos que viene precargados en la GUI, así que si queremos trabajar con otros deberemos importarlos. (En caso de duda, ver apartado “11.1.1.- Importar datos.”).

Las gráficas se actualizan automáticamente cada vez que importamos datos, modificamos el orden de ajuste, o cambiamos de pestaña, representándonos automáticamente las curvas de altura manométrica y de potencia útil. Para que nos muestre la curva de potencia absorbida, par o rendimiento, deberemos hacer uso de los menús desplegables que se encuentran a la izquierda (figura 11.15) y a la derecha (figura 11.16) de la gráfica inferior.

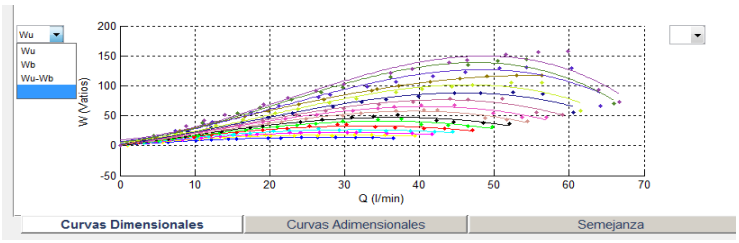


Figura 11.15

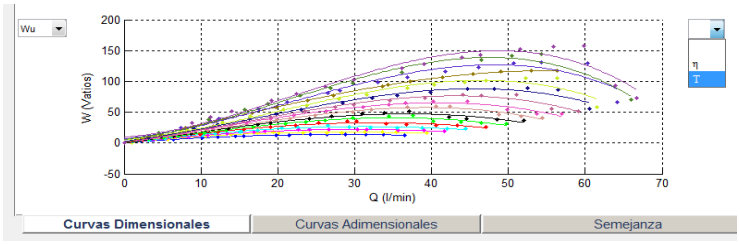


Figura 11.16

Los efectos de combinar ambos menús, son los que se muestran en la tabla 11.1.

Menú desplegable de la:		Grafica que se obtiene:
Derecha	Izquierda	
Wu		Potencia útil en eje izquierdo
Wb		Potencia absorbida en eje izquierdo
Wu-Wb		Potencia útil y potencia absorbida en eje izquierdo
	η	Rendimiento en eje izquierdo
Wu	η	Potencia útil en eje izquierdo y rendimiento en eje derecho
Wb	η	Potencia absorbida en eje izquierdo y rendimiento en eje derecho
Wu-Wb	η	Potencia útil y absorbida en eje izquierdo y rendimiento en eje derecho
	T	Par en eje izquierdo
Wu	T	Potencia útil en eje izquierdo y par en eje derecho
Wb	T	Potencia absorbida en eje izquierdo y par en eje derecho
Wu-Wb	T	Potencia útil y absorbida en eje izquierdo y para en eje derecho
		Gráfica en blanco

Tabla 11.1

11.1.5.- Seleccionar valores a representar.

Para elegir las series de datos que queremos que se representen, opción disponible en las pestañas “Curvas Dimensionales” y “Curvas Adimensionales”, deberemos de utilizar el panel de botones “Representar”, situado en la parte derecha de la ventana (figura 11.17). En este, podemos elegir una de las siguientes opciones:

- **“Todos los valores de RPM”.** Nos representará una curva por cada régimen de giro del que hayamos importado datos, y utilizará como marcador el punto. Debido a que el número de series que representa depende de los datos que importemos, el programa hace uso de una función de Matlab que genera números pseudoaleatorios, para elegir los distintos colores. Esto produce, que en algunas ocasiones se generen colores muy similares entre sí para las distintas series. En caso de suponer un problema, se podrán actualizar estos colores, pulsando la teclas “Control” y “A” simultáneamente, o bien, en el menú Gráficas → Actualizar colores.

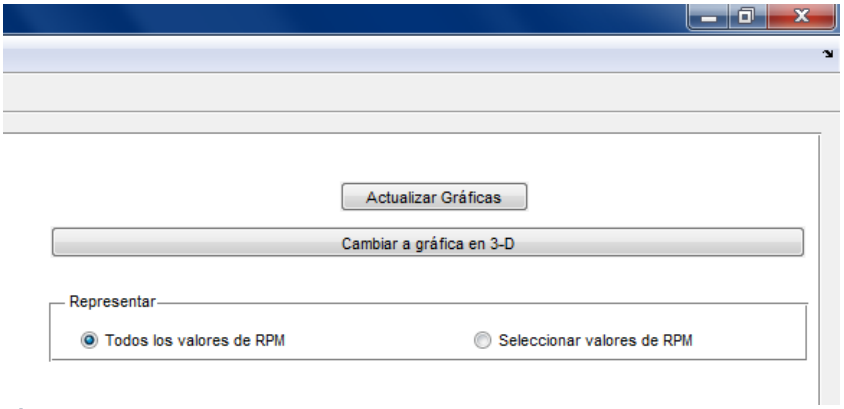


Figura 11.17

- **“Seleccionar valores de RPM”.** Esta opción abre un panel que permite elegir once o menos, dependiendo de nuestras necesidades, valores a representar y los atributos que tendrán (color, tipo de marcador, y tipo de línea), ver figura 11.18. En los menús desplegables se eligen las velocidades de giro que queremos que nos represente, los colores y los tipos de línea, actualizándose las gráficas automáticamente cada vez que modifiquemos alguno. Como puede observarse, la última fila, en vez de contener un menú desplegable para seleccionar el régimen de giro, contiene un texto editable. Si introducimos en este un régimen de giro del que hallamos importado datos, nos lo representará, sino, nos representara las curvas obtenidas por semejanza con relación al régimen de giro más

cercano. (Téngase en cuenta las limitaciones a la hora de aplicar semejanza, más información en “7.3.- Pestaña curvas adimensionales.”).

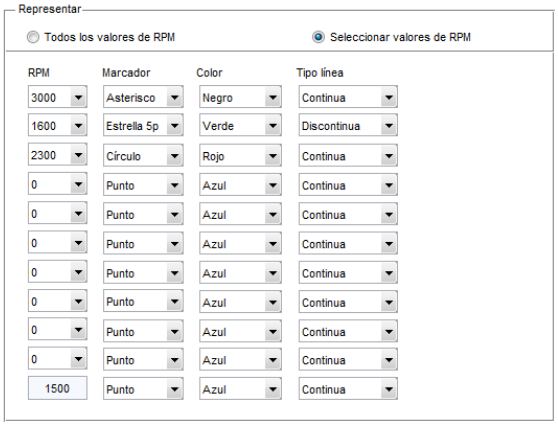


Figura 11.18

11.1.6.- Representación de contornos de Isorendimiento.

Los contornos de Isorendimiento solo se representan en la pestaña “Curvas Dimensionales”, y para ello deben de estar representadose al menos 3 velocidades de giro distintas (en caso de duda, consultar apartado “11.1.5.- Seleccionar valores a representar.”).

Si la pestaña activa no es la de “Curvas Dimensionales” tendremos que activarla haciendo clic en el boton “Curvas Dimensionales” situado en la parte inferior de la ventana. Para representar los contornos de isorendimiento, haremos clic en menú Gráficas → Contornos de Isorendimiento → Contornos de Isorendimiento (Mostar), figura 11.19.

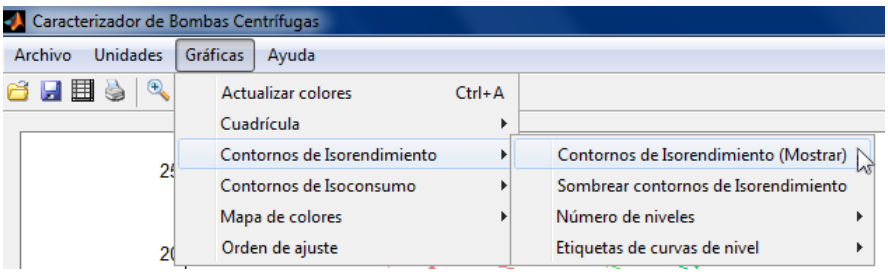


Figura 11.19

Con esto nos mostrará los contornos de isorendimiento en la gráfica superior (similar a figura 11.20). Los cambios que podemos realizar sobre estos son:

- **Sombrearlos**. Para ello hacer clic en menú Gráficas → Contornos de Isorendimiento → Sombrear contornos de Isorendimiento.

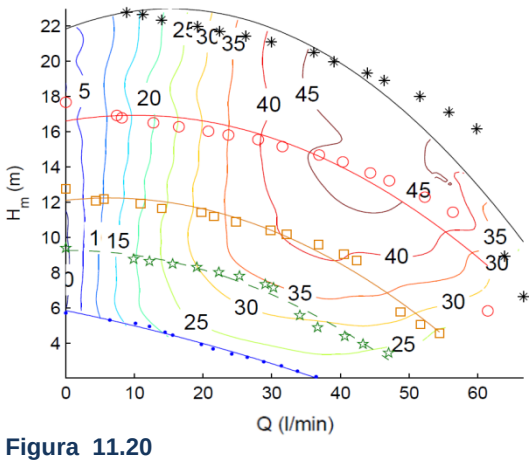


Figura 11.20

- **Cambiar número de niveles.** En el menú Gráficas → Contornos de Isorendimiento → Número de niveles, podemos elegir dos opciones:
  - Automático: es la opción activada por defecto. Selecciona automáticamente el número de niveles, para evitar el uso de decimales.
  - Introducir Valor: si hacemos clic en esta opción, se nos abrira una ventana como la que se muestra en la figura 11.21, que nos pedirá que le indiquemos el número de curvas que queremos.

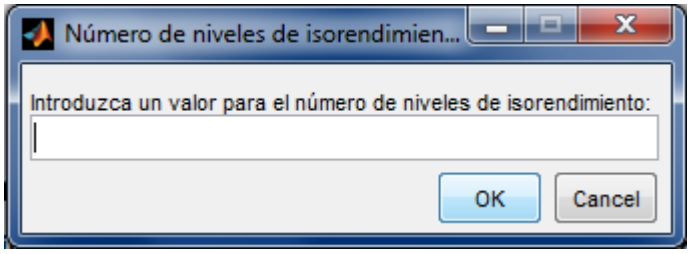


Figura 11.21

- **Etiquetas de curvas de nivel.** En el menú Gráficas → Contornos de Isorendimiento → Etiquetas de curvas de nivel, podemos elegir tres opciones:
  - Sin etiquetas. Si activamos esta opción, no mostrará etiquetas.
  - Posicionamiento automático: es la opción seleccionada por defecto, y distribuye las etiquetas de forma equiespaciada.
  - Posicionamiento manual: al elegir ésta, se nos abre un cuadro de dialogo con las intrucciones para colocar las etiquetas. Permitiendonos esta opción distribuir las etiquetas a nuestro antojo.
- **Cambiar mapa de colores.** El mapa de colores utilizado para colorear los contornos se elije en el menú Gráficas → Mapa de colores.

Para desactivar los contornos de Isorendimiento, se hace clic en menú Gráficas → Contornos de Isorendimiento → Contornos de Isorendimiento (Ocultar).

Se pueden activar simultaneamente los contornos de Isorendimiento e Isoconsumo, sin embargo, no se recomienda esta opción, ya que las gráficas son dificilmente interpretables.

11.1.7.- Representación de contornos de Isoconsumo.

Los contornos de Isoconsumo solo se representan en la pestaña “Curvas Dimensionales”, y para ello deben de estar representadose al menos 3 velocidades de giro distintas (en caso de duda, consultar apartado “11.1.5.- Seleccionar valores a representar.”).

Si la pestaña activa no es la de “Curvas Dimensionales” tendremos que activarla haciendo clic en el boton “Curvas Dimensionales” situado en la parte inferior de la ventana. Para representar los contornos de isoconsumo, haremos clic en menú Gráficas → Contornos de Isoconsumo → Contornos de Isoconsumo (Mostar), figura 11.22.

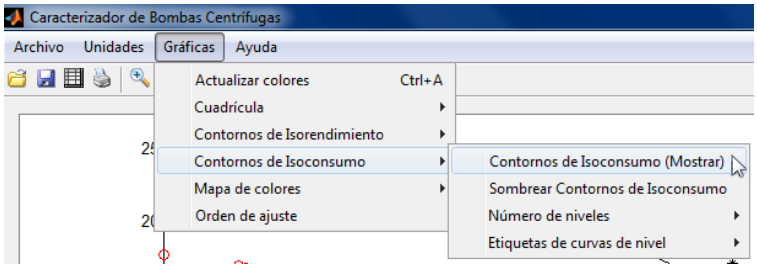


Figura 11.22

Con esto nos mostrará los contornos de isoconsumo en la gráfica superior (similar a figura 11.23). Los cambios que podemos realizar sobre estos son:

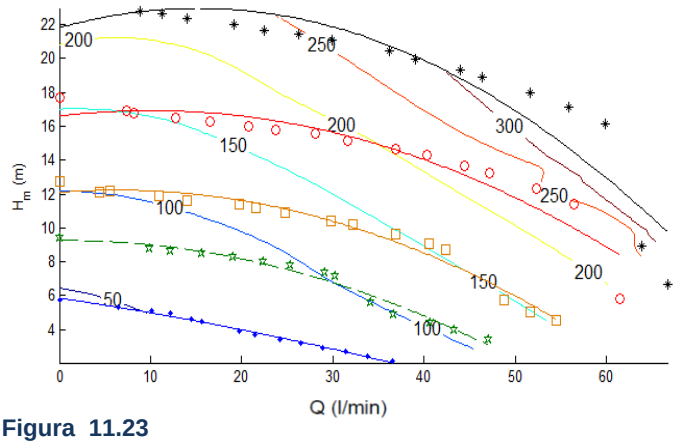


Figura 11.23

- **Sombrearlos**. Para ello hacer clic en menú Gráficas → Contornos de Isoconsumo → Sombrear contornos de Isoconsumo.
- **Cambiar número de niveles.** En el menú Gráficas → Contornos de Isoconsumo → Número de niveles, podemos elegir dos opciones:
    - Automático: es la opción activada por defecto. Selecciona automáticamente el número de niveles, para evitar el uso de decimales.
    - Introducir Valor: si hacemos clic en esta opción, se nos abrira una ventana como la que se muestra en la figura 11.24, que nos pedirá que le indiquemos el valor de curvas que queremos.

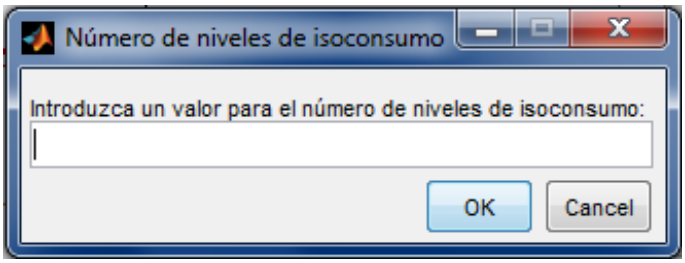


Figura 11.24

- **Etiquetas de curvas de nivel.** En el menú Gráficas → Contornos de Isoconsumo → Etiquetas de curvas de nivel, podemos elegir tres opciones:
    - Sin etiquetas. Si activamos esta opción, no mostrará etiquetas.
    - Posicionamiento automático: es la opción seleccionada por defecto, y distribuye las etiquetas de forma equiespaciada.
    - Posicionamiento manual: al elegir ésta, se nos abre un cuadro de dialogo con las intrucciones para colocar las etiquetas. Permitiendonos esta opción distribuir las etiquetas a nuestro antojo.
- **Cambiar mapa de colores.** El mapa de colores utilizado para colorear los contornos se elije en el menú Gráficas → Mapa de colores.

Para desactivar los contornos de Isoconsumo, se hace clic en menú Gráficas → Contornos de Isoconsumo → Contornos de Isoconsumo (Ocultar).

Se pueden activar simultaneamente los contornos de Isorendimiento e Isoconsumo, sin embargo, no se recomienda esta opción, ya que las gráficas son dificilmente interpretables.

11.1.8.- Representación de curvas en 3-D.

Para representar curvas en tres dimensiones debemos estar en la pestaña “Curvas Dimensionales”. En caso contrario, hacer clic en el botón “Curvas Dimensionales” para activarla.

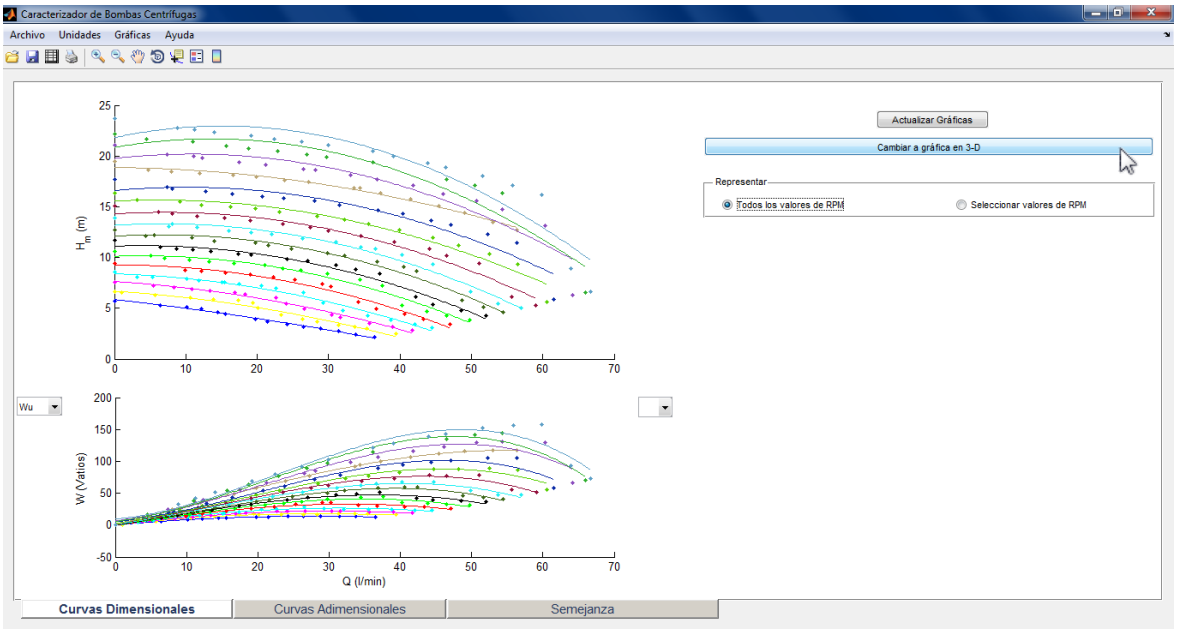


Figura 11.25

Es necesario hacer clic en el botón “Cambiar a gráfica en 3-D”, figura 11.25, con lo que cambiará el aspecto de la pestaña, al que se muestra en la figura 11.26

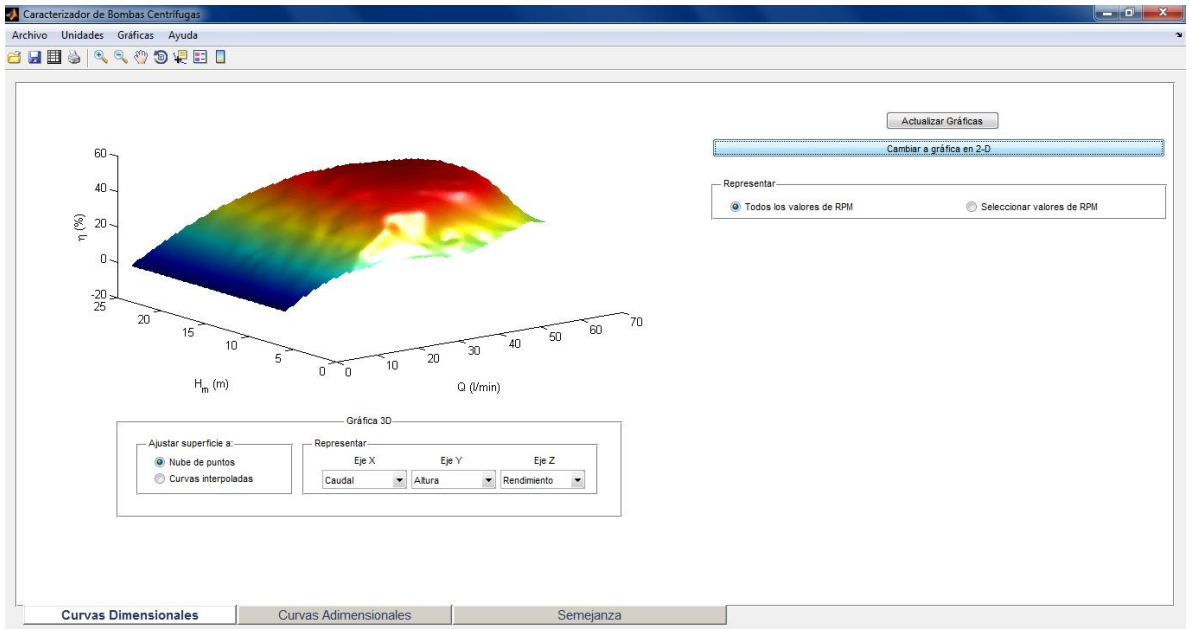


Figura 11.26

Para seleccionar las velocidades de giro que queremos que nos represente, usaremos cuadro de botones situado a la derecha de la gráfica, cuyo uso se define en el apartado “11.1.5.- Seleccionar valores a representar.”.

Las magnitudes representadas en cada uno de los ejes de la gráfica, se definen en los menús desplegables que están situados debajo de ésta. (Ver figura 11.27)

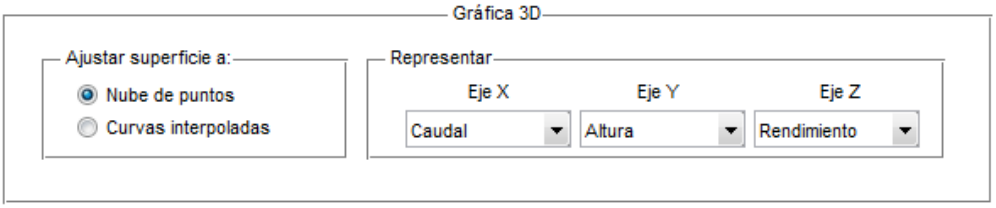


Figura 11.27

Como puede observarse en la figura 11.27, existe un panel de botones, “ajustar superficie a:”, que nos permite modificar la forma en la que la superficie de ajuste se adapta a los datos. Ofreciéndonos las siguientes posibilidades:

- **Nube de puntos.** Esta opción ajusta la superficie que representa a los valores obtenidos de los puntos de ensayo.
- **Curvas interpoladas:** Esta opción, ajusta la superficie que representa sobre los valores de las curvas de ajuste obtenidas de aplicar la función “polyfit” a los puntos obtenidos de los datos de ensayo. En algunos casos puede ofrecernos curvas mejor suavizadas que la opción “Nube de puntos”. Sin embargo, se ha de tener en cuenta que en el ajuste por mínimos cuadrados que se utiliza, se ajusta la curva de ajuste para minimizar la variación de altura, potencia y rendimiento con respecto al caudal, y esta no es la opción más acertada para este tipo de superficies.

El color utilizado por la superficie se puede modificar cambiando el mapa de colores, en menú Gráficas → Mapa de colores.

Para volver a representar gráficas en dos dimensiones se hará clic en el botón “Cambiar a gráfica en 2-D”, que estará situado en el mismo lugar en el que antes se encontraba el botón “Cambiar a gráfica en 2-D”.

11.1.9.- Representación de grupos adimensionales.

Para representar los grupos adimensionales, nos tenemos que situar en las pestaña “Curvas adimensionales”. Para ello, hacer clic en el botón “Curvas adimensionales”, situado en la parte inferior de la ventana.

El aspecto de esta pestaña es el que se muestra en la figura 11.28. En la parte derecha tenemos el panel que controla las series de datos que queremos que nos represente, su uso se detalla en el apartado “11.1.5.- Seleccionar valores a representar.”. En la parte izquierda, tenemos dos gráficas para la representación de los grupos adimensionales.

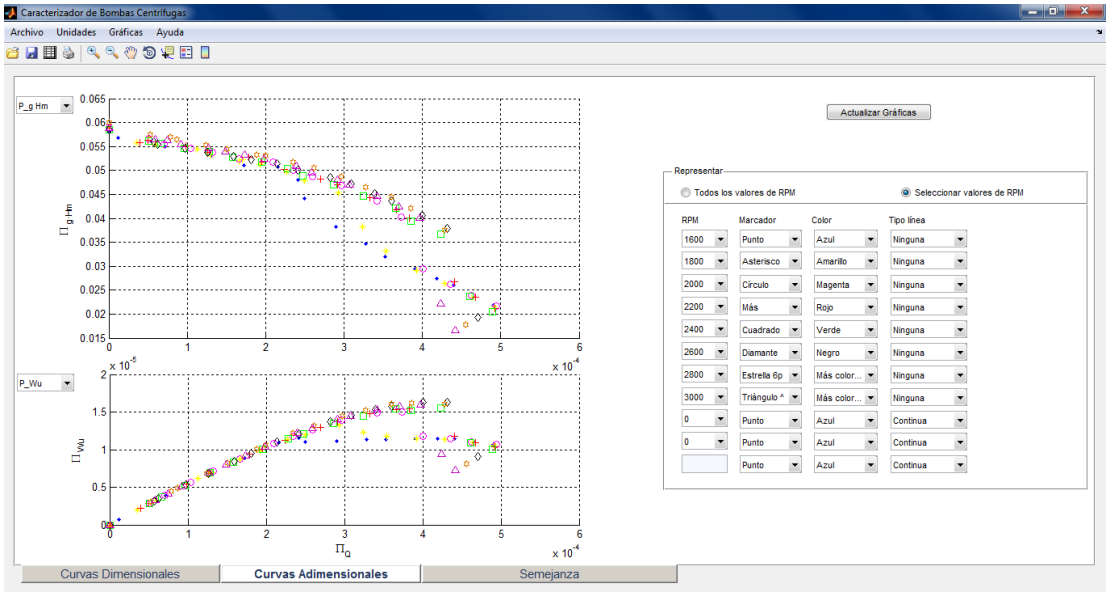


Figura 11.28

Junto a cada gráfica tenemos un menú desplegable, que es el que nos permite controlar lo que representamos en ellas. Para cambiar el contenido de una de las gráfica, nos basta con desplegar el menú que se encuentra junto a ella y elegir el grupo adimensional que queremos que no represente (las gráfica se actualizan automáticamente al seleccionar una opción del menú). Ver ejemplo en figura 11.29.

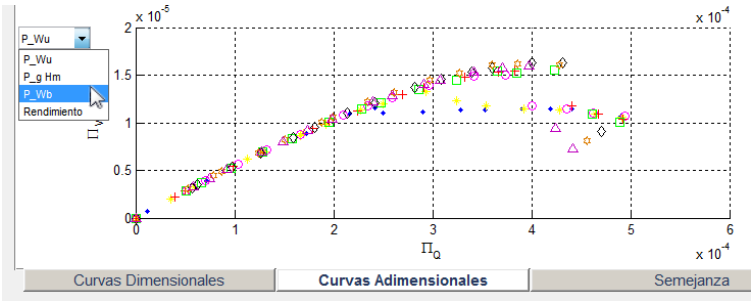


Figura 11.29

11.1.10.- Obtener curvas por semejanza.

La curvas por semejanza se obtiene de aplicar las ecuaciones que se presentan el en apartado “7.4.- Pestaña semejanza.”.

Para obtener curvas por semejanza debemos de situarnos en la pestaña “Semejanza”. Si no es esta la pestaña activa, deberemos hacer clic sobre el botón “Semejanza”, situado en la parte inferior de la ventana.

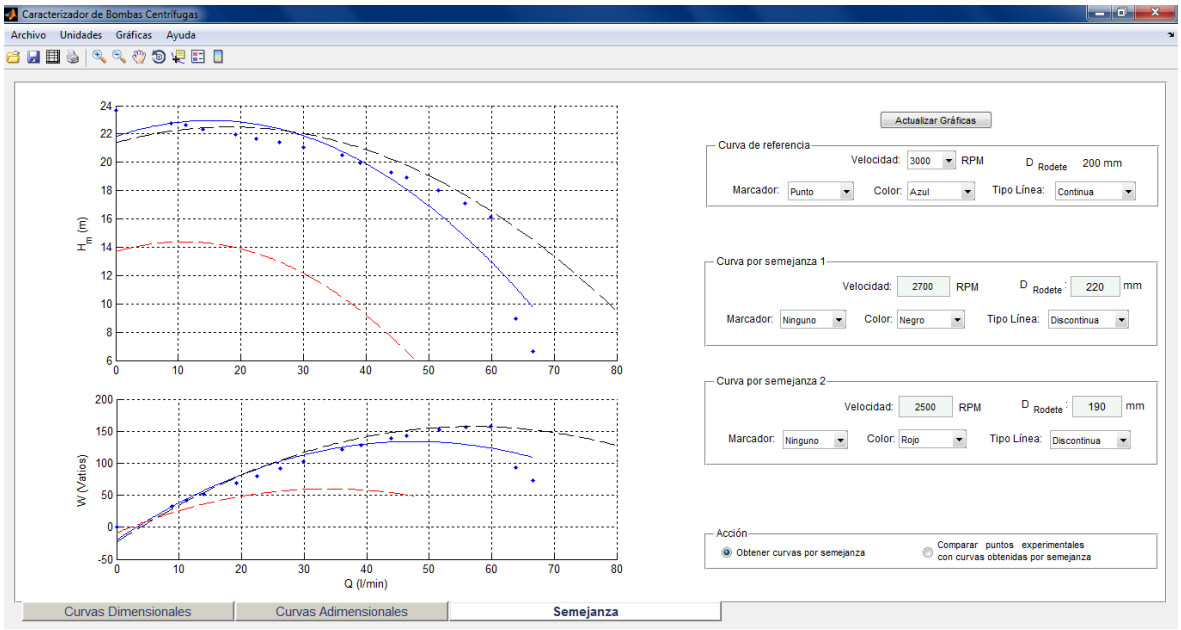


Figura 11.30. Pestaña Semejanza

El aspecto que presenta esta pestaña es el que se muestra en la figura 11.30. Teniendo dos graficas en la parte izquierda, la superior para la altura manométrica y la inferior para la potencia útil. En la parte derecha tenemos cuatro paneles:

- **Curva de referencia:** En este se especifica de entre los valores importados, cual es el régimen de giro que queremos que utilice como referencia. Esta curva también nos la representa, y es por eso que nos permite seleccionar el color, tipo de marcador y tipo de línea.
- **Curva por semejanza 1:** nos representa el comportamiento que tendría la bomba semejante a la de referencia, girando al número de revoluciones que le indiquemos, y con el diámetro de rodete que especifiquemos en este panel. Los puntos que representa, son puntos obtenidos por semejanza a partir de los puntos de la curva de referencia, por tanto, no se deben confundir con valores de ensayo. Este panel, también nos permite

seleccionar el color, tipo de marcador y tipo de línea, utilizados para la curva por semejanza 1.

- **Curva por semejanza 2:** Sirve para obtener otra curva por semejanza. El panel trabaja exactamente igual que el que tenemos para la “Curva por semejanza 1”.
- **Acción:** nos permite elegir el objetivo que perseguimos al utilizar el análisis por semejanza. Ofreciendo las siguientes opciones:
  - “Obtener curvas por semejanza”: Es la opción seleccionada por defecto y la que se está describiendo en este apartado. Sirve para predecir el comportamiento que tendría una bomba semejante a la ensayada en condiciones distintas de las de ensayo.
  - “Comparar puntos experimentales con curvas obtenidas por semejanza”: Como su nombre indica, sirve para comparar las curvas obtenidas por semejanza con los datos de ensayo.

11.1.11.- Comparar puntos experimentales con curvas obtenidas por semejanza.

Esta interfaz gráfica, para demostrar que el análisis por semejanza en bombas centrífugas es de gran utilidad a la hora de predecir la altura manométrica y la potencia útil que suministra una bomba, permite representar los valores obtenidos en el laboratorio frente a las curvas por semejanza que se obtendrían, para que así se pueda valorar de forma cualitativa la separación que existe entre teoría y práctica.

Hacer clic en el botón “Semejanza”, para activar esta pestaña, y en el cuadro de botones “Acción” hacer clic sobre el botón circular “Comparar puntos experimentales con curvas obtenidas por semejanza” figura 11.31. De esta manera el aspecto de la pestaña será el mostrado en la figura 11.32.

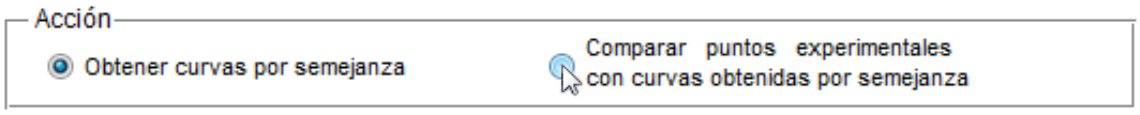


Figura 11.31

Nota: Para activar la opción “Compara puntos experimentales con curvas obtenidas por semejanza”, no es suficiente con hacer clic sobre el texto, es necesario hacer clic sobre el botón circular situado a la izquierda del texto. Esto se debe a la utilización del texto en dos líneas. No siendo necesario tomar esta precaución para el resto de botones circulares utilizados en este programa.

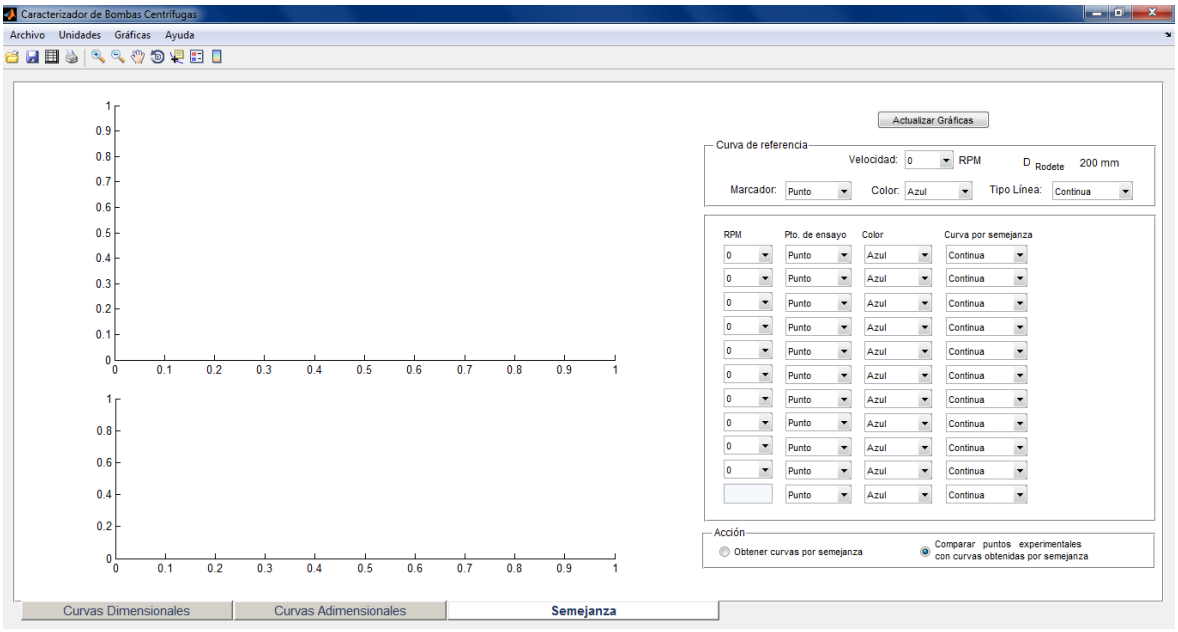


Figura 11.32

En el panel “Curva de referencia” se introduce el régimen de giro que queremos que utilice como referencia a la hora de aplicar semejanza, y el color, tipo de marcador y tipo de línea que queremos que utilice para su representación.

En el panel que se encuentra debajo, figura 11.33, se seleccionan las velocidades de giro en las que queremos que se realice la comparación, y el programa se encargará de representar, para cada régimen de giro seleccionado, los puntos obtenidos en el banco de ensayo y la curva por semejanza correspondiente a ese mismo régimen de giro.

Figura 11.33

La columna “Pto. de ensayo”, permite seleccionar el marcador que queremos que utilice para los puntos de ensayo correspondiente a cada una de las velocidades de giro seleccionadas. Mientras que, la columna “Curvas por semejanza”, permite seleccionar el tipo de línea para cada una de las curvas obtenidas por semejanza.

La figura 11.34 muestra un ejemplo de uso.

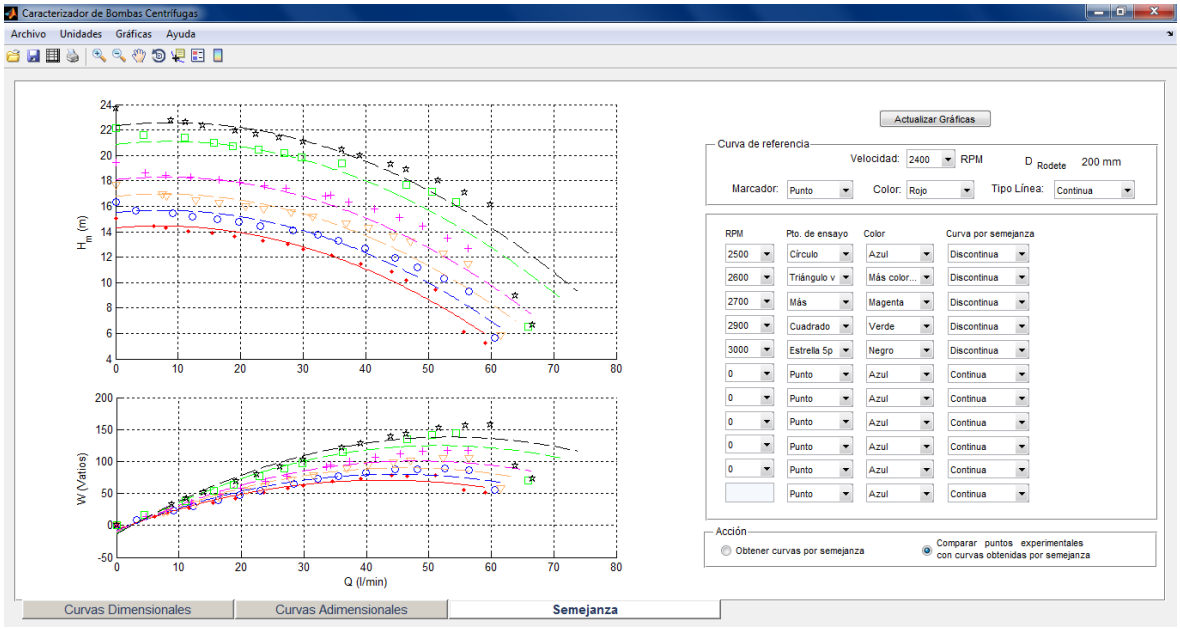


Figura 11.34

11.1.12.-Exportación de gráficas.

Para exportar las gráficas que el programa muestra en pantalla, se tiene que hacer clic en el menú Archivo → Exportar Gráfica, figura 11.35, o bien, hacer clic en el icono de guardar, figura 11.36.

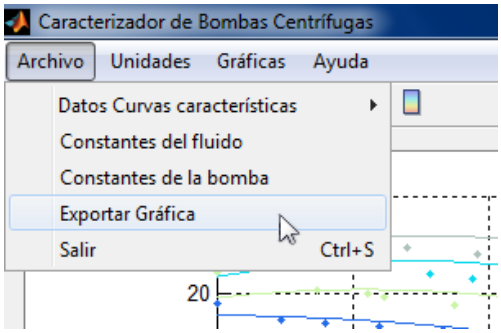


Figura 11.35

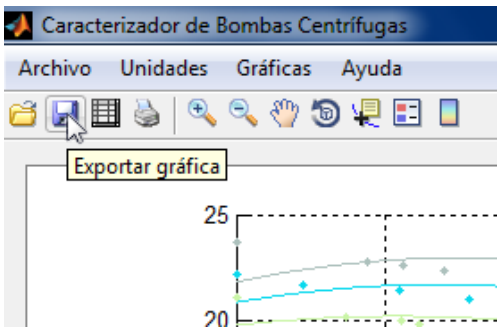


Figura 11.36

Si en la ventana gráfica que tenemos existen dos gráficas (siempre que no se esté trabajando con gráficas en 3-D), se nos abrirá un cuadro de diálogo como el que se muestra en la figura 11.37, en el que se elige cual es la gráfica que queremos exportar.

Figura 11.37

Al elegir la opción que necesitamos, se nos abre el cuadro de diálogo que se muestra en la figura 11.38. En este, tenemos que:

- Situar el directorio en el que queremos que nos guarde la gráfica
- Elegir el tipo de extensión (“.jpg”, “fig”, “”pdf”,…).
- Poner un nombre al fichero, (Solo el nombre, sin extensiones).
- Hacer clic en guardar.
- Esperar a que se cierre la ventana en la que se muestra la gráfica importada.

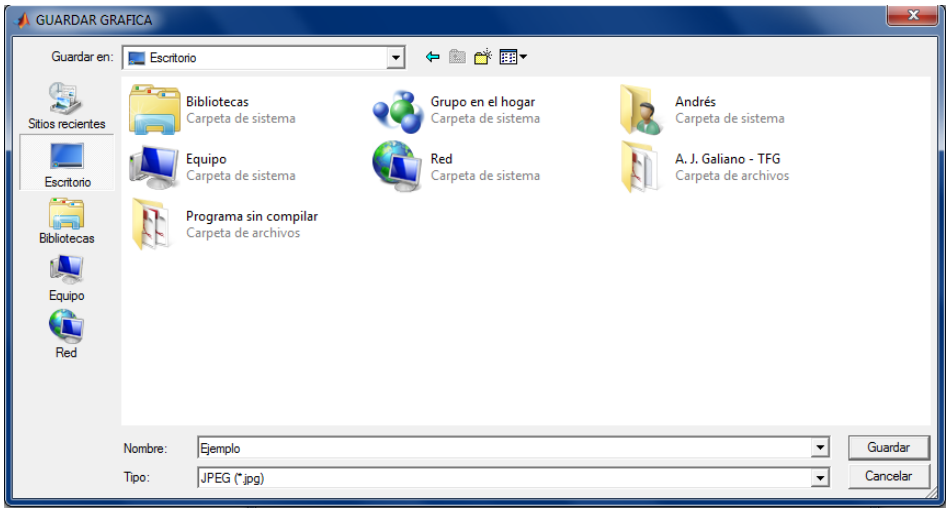


Figura 11.38

11.2.- Anexo 2. Código del programa.

En este capítulo se expondrá el código principal de la GUI, así como el de una función y el de dos GUIs auxiliares que utiliza como ventanas emergentes.

Debido a las características del código no puedo cumplir con las normas de estilo, impuestas por la universidad para los trabajos fin de gradado sin comprometer la interpretabilidad del código. Por ello, para distinguir aquellos párrafos pertenecientes al código del programa y que por tanto no cumplen con parte de la citada normativa, voy a sombrearlos en un amarillo tenue.

11.2.1.- Función "Cargar_datos"

El código principal de la GUI utiliza esta función cada vez que el usuario le pide que importe datos. Tiene como argumentos de entrada la dirección en la que se encuentran los ficheros a importar y una lista con el nombre de los ficheros o el fichero a importar. Y devuelve como argumento de salida una matriz de cinco columnas y tantas filas como puntos de ensayo hubiera en los ficheros importados.

El código de esta función es el siguiente:

```
function datos=Cargar_Datos(directorio,nombre)
      a=0;h=waitbar(0,'Por favor espere...','Name','Cargando datos...');
      for i=1:1:length(nombre)
          if iscellstr(nombre),k=nombre(i);else k=nombre;end
          v=importdata([directorio '' char(k)],'\t',1);
          v=v.data;v=[v(:,2) v(:,3) v(:,4) v(:,5) v(:,6)];
          for j=a+1:1:a+length(v(:,1)),datos(j,:)=v(j-a,:);
              waitbar((i-1+(j-a)/length(v(:,1)))/length(nombre))
          end
          a=a+length(v(:,1));
      end
      datos=[datos(:,1) datos(:,2) datos(:,3) datos(:,4) datos(:,5)];
      close(h)
end
```

11.2.2.- Código de la GUI "dialgtable"

Esta es una GUI auxiliar que muestra en pantalla un cuadro de diálogo con una tabla, en la aparecen los datos importados (ver figura 11.35). Tiene como argumento de entrada una matriz numérica con los datos mostrados en la tabla, y no posee argumentos de salida.

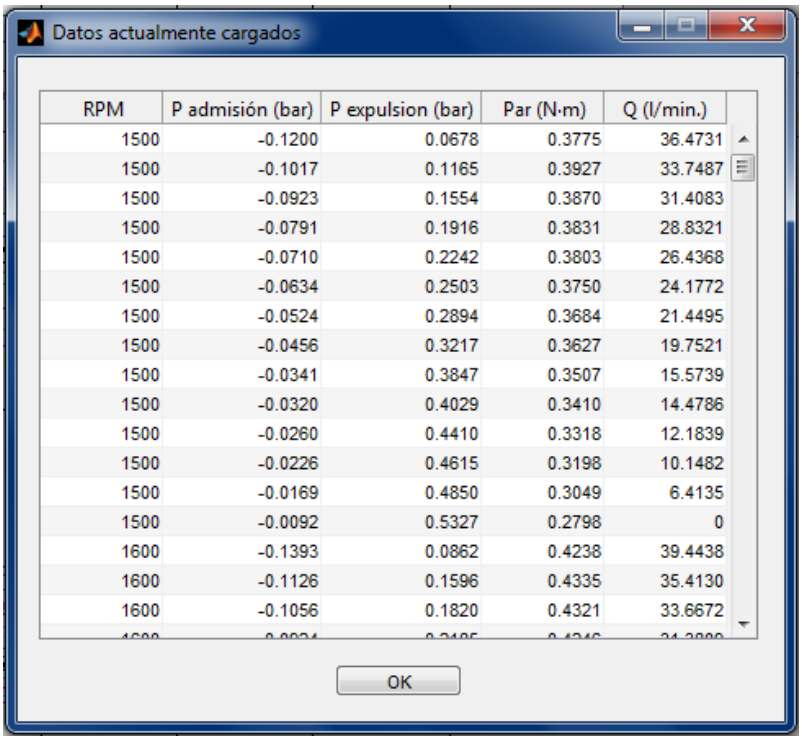


Figura 11.35: Ventana de la GUI "dialgtable"

El código de esta GUI auxiliar es:

```
function [datos] = dialgtable(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dialgtable_OpeningFcn, ...
                   'gui_OutputFcn',  @dialgtable_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 dialgtable_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% Choose default command line output for dialgtable
handles.output = 'Yes';
guidata(hObject, handles);
if(nargin > 3)
    for index = 1:2:(nargin-3),
        if nargin-3==index, break, end
        switch lower(varargin{index})
         case 'title'
          set(hObject, 'Name', varargin{index+1});
         case 'string'
          set(handles.text1, 'String', varargin{index+1});
        end
    end
end

FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);
if isempty(gcbf)
    ScreenUnits=get(0,'Units');set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');set(0,'Units',ScreenUnits);

    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);FigPos(2)=2/3*(ScreenSize(4)-
FigHeight);
else
    GCBFOldUnits = get(gcbf,'Units'); set(gcbf,'Units','pixels');
    GCBFPos = get(gcbf,'Position'); set(gcbf,'Units',GCBFOldUnits);
    FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
                   (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);set(hObject, 'Units', OldUnits);

set(handles.figure1,'WindowStyle','modal')% Make the GUI modal
set(handles.uitable1,'data',varargin{1})
uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = dialgtable_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
delete(handles.figure1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
handles.output = get(hObject,'String');
guidata(hObject, handles);uiresume(handles.figure1);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
handles.output = get(hObject,'String');
guidata(hObject, handles);uiresume(handles.figure1);

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
if isequal(get(hObject, 'waitstatus'), 'waiting')
    uiresume(hObject);
else
```

```
    delete(hObject);
end


% --- Executes on key press over figure1 with no controls selected.
function figure1_KeyPressFcn(hObject, eventdata, handles)
% Check for "enter" or "escape"
if isequal(get(hObject,'CurrentKey'),'escape')
    handles.output = 'No';guidata(hObject,
handles);uiresume(handles.figure1);
end

if isequal(get(hObject,'CurrentKey'),'return')
    uiresume(handles.figure1);
end
```

11.2.3.- Código de la GUI "dialginputimagen"

Esta GUI auxiliar muestra en pantalla una ventana emergente con las características de la bomba y del banco de ensayo, permitiendo al usuario modificarlas, figura 11.36.

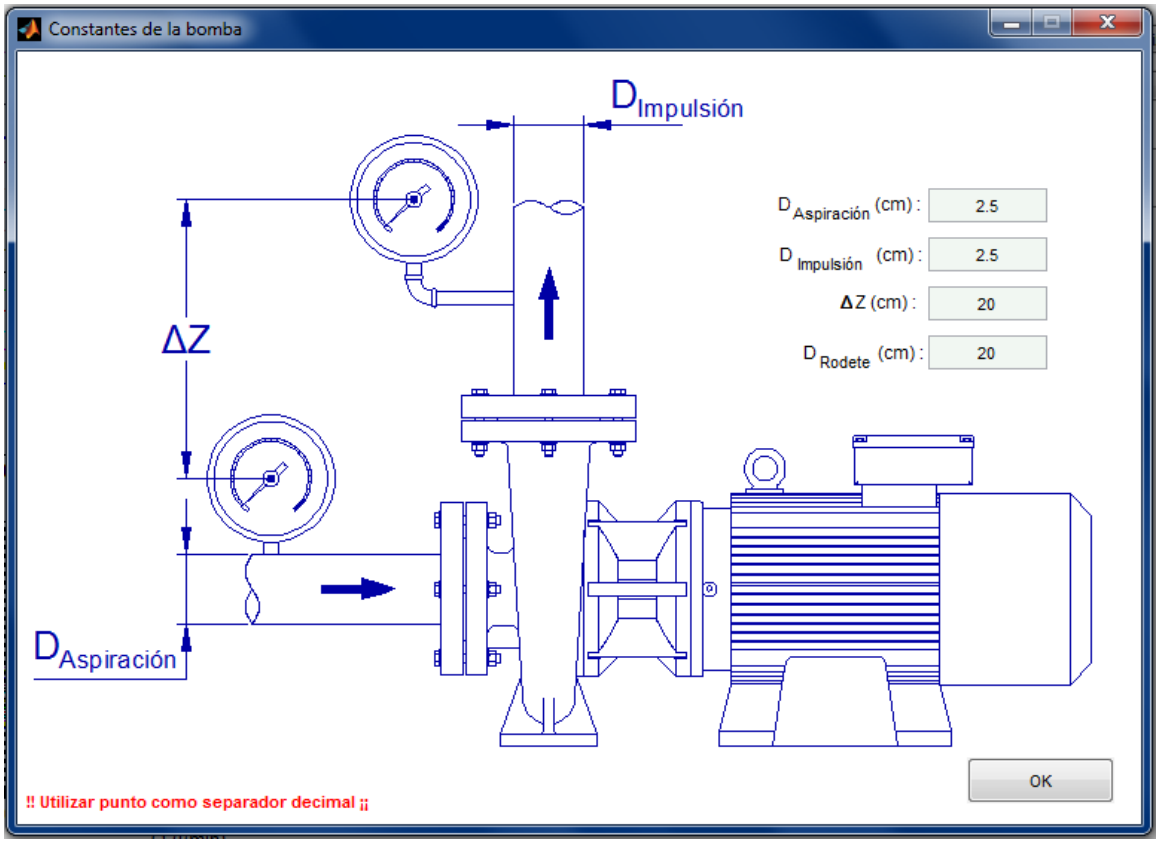


Figura 11.36: Ventana de la GUI "dialginputimagen"

Los argumentos de entrada son el diámetro del rodete, los diámetros de las tuberías de aspiración e impulsión y la diferencia de cota entres los puntos en los que se mide la presión, y devuelve como argumento de salida los valores que haya introducido el usuario en pantalla para estos parámetros.

El código de está GUI auxiliar es el que sigue:

```
function varargout = dialginputimagen(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dialginputimagen_OpeningFcn, ...
                   'gui_OutputFcn',  @dialginputimagen_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 dialginputimagen is made visible.
function dialginputimagen_OpeningFcn(hObject, eventdata, handles,
varargin)
% Choose default command line output for dialginputimagen
handles.output = 'Yes';guidata(hObject, handles);
if(nargin > 3)
    for index = 1:2:(nargin-3),
        if nargin-3==index, break, end
        switch lower(varargin{index})
         case 'title'
          set(hObject, 'Name', varargin{index+1});
         case 'string'
          set(handles.text1, 'String', varargin{index+1});
        end
    end
end

% Escribo valores que apareceran por defecto en la ventana (Da,Di,AZ)
set(handles.edit1,'string',num2str(varargin{1})) % Az (cm)
set(handles.edit2,'string',num2str(varargin{2})) % D_imp.(cm)
set(handles.edit3,'string',num2str(varargin{3})) % D_asp.(cm)
set(handles.edit4,'string',num2str(varargin{4})) % D_rod.(cm)

% if available, else, centered on the screen
FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');FigWidth = OldPos(3);FigHeight =
OldPos(4);
if isempty(gcbf)
    ScreenUnits=get(0,'Units');
```

```
    set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');
    set(0,'Units',ScreenUnits);

    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);
    FigPos(2)=2/3*(ScreenSize(4)-FigHeight);
else
    GCBFOldUnits = get(gcbf,'Units');
    set(gcbf,'Units','pixels');
    GCBFPos = get(gcbf,'Position');
    set(gcbf,'Units',GCBFOldUnits);
    FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
                   (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);set(hObject, 'Units', OldUnits);

% Muestra imagen de bomba centrifuga en axes 1
axes(handles.axes1)
background = imread('Pumb.bmp');
axis off;imshow(background);

set(handles.figure1,'WindowStyle','modal')% Make the GUI modal
uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = dialginputimagen_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;delete(handles.figure1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

AZ=str2double(get(handles.edit1,'String'));
Di=str2double(get(handles.edit2,'String'));
Da=str2double(get(handles.edit3,'String'));
Dr=str2double(get(handles.edit4,'String'));
if isnan(AZ)||isnan(Di)||isnan(Da)||isnan(Dr)
    errordlg({'Los campos deben ser', 'un valor numerico'}...
        ,'Error')
    return
elseif Da~=0&&Di~=0&&Dr~=0
    handles.output={num2str(AZ);num2str(Di);num2str(Da);num2str(Dr)};
elseif Di==0
    errordlg({' El diámetro de la tubería de impulsión debe',...
        'ser un valor numérico mayor que cero'},'Error')
    return
elseif Da==0
    errordlg({' El diámetro de la tubería de aspiración debe',...
        'ser un valor numérico mayor que cero'},'Error')
    return
elseif Dr==0
    errordlg({' El diámetro del rodete debe ser',...
        ' un valor numérico mayor que cero'},'Error')
    return
end
guidata(hObject, handles);

uiresume(handles.figure1);
```

```
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
if isequal(get(hObject, 'waitstatus'), 'waiting')
    uiresume(hObject);
else
    delete(hObject);
end

% --- Executes on key press over figure1 with no controls selected.
function figure1_KeyPressFcn(hObject, eventdata, handles)
% Check for "enter" or "escape"
if isequal(get(hObject,'CurrentKey'),'escape')
    handles.output = 'No';
    guidata(hObject, handles);uiresume(handles.figure1);
end
if isequal(get(hObject,'CurrentKey'),'return')
    uiresume(handles.figure1);
end

function edit1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2 Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

11.2.4.- Código de la GUI principal "Caracterizador de bombas centrífugas".

En este subapartado se expone el código principal de la aplicación gráfica objeto de este trabajo:

```
function varargout = Programa(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Programa_OpeningFcn, ...
                   'gui_OutputFcn',  @Programa_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 Programa_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;guidata(hObject, handles);
g=9.81;rho=1000;Da=0.025;Di=0.025;dz=0.2;Ucaudal='l/min';Upresion='H_m
(m)';
handles.g=g;handles.rho=rho;handles.Da=Da;handles.Di=Di;
handles.dz=dz;handles.R=0;
handles.Ucaudal=Ucaudal;handles.Upresion=Upresion;directorio=[pwd
'\'];handles.Upotencia='W (Vatios)';
handles.egrid='on';ficheros=dir(fullfile(directorio,'*.dat'));
nombre={ficheros.name};
datos=Cargar_Datos(directorio,nombre);handles.ordenh=2;handles.ordenwu=3;
handles.z=0;
handles.ordenwb=3;handles.ordenT=3;handles.ordenN=3;handles.Dr=0.2;
handles.mu=1.102*10^-3;
handles.datos=datos;handles.mapacolor='default';
handles.vgiro='\Omega (RPM)';
guidata(hObject,handles);
RPM(1)=datos(1,1);
for i=1:1:length(datos(:,1)),
if datos(i,1)~=RPM, RPM(length(RPM)+1)=datos(i,1); end,
end
handles.RPM=RPM;handles.RPMo=RPM;
color=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
for i=8:1:length(RPM),color(i,:)=rand(1,3);end
handles.color=color;
handles.color0=[0,0,1;0,0,1;0,0,1;0,0,1;0,0,1;0,0,1;0,0,1;0,0,1;0,0,1;0,0,1;0,0
,1;0,0,1];
axes(handles.axes1);
guidata(hObject,handles);
hold on
for i=1:1:length(RPM)
    q=datos(datos(:,1)==RPM(i),5);
```

```
    h=(datos(datos(:,1)==RPM(i),3)-
datos(datos(:,1)==RPM(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)*4/...
(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
    pol_h=polyfit(q,h,2);
    plot(q,h,'.','color',color(i,:))

plot(min(q):0.1:max(q),polyval(pol_h,min(q):0.1:max(q)),'color',color(i,:
))
end
ylabel(Upresion);grid on,hold off

axes(handles.axes2);
guidata(hObject,handles);
hold on
for i=1:1:length(RPM)
    q=datos(datos(:,1)==RPM(i),5);
    h=(datos(datos(:,1)==RPM(i),3)-
datos(datos(:,1)==RPM(i),2))/(g*rho*10^-5)...
        +dz+(((4/(pi*Di^2)-4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
    Wu=g*rho*(1/60000)*q.*h; pol_Wu=polyfit(q,Wu,3);
    plot(q,Wu,'.','color',color(i,:))

plot(min(q):0.1:max(q),polyval(pol_Wu,min(q):0.1:max(q)),'color',color(i,
:))
end
xlabel(['Q (',Ucaudal,')']); ylabel(handles.Upotencia) ;grid on, hold off

simbolo=zeros(1,length(RPM));
for i=1:1:length(RPM),simbolo(i)='.';end
handles.simbolo=simbolo;handles.tlinea=zeros(1,length(RPM));
set(handles.popupmenu57,'Value',1);b='0';
for i=1:1:length(RPM),b=[b,'|',num2str(RPM(i))];end
set(handles.popupmenu57,'String', b);handles.semejanzacolor1=[0,0,1];
handles.semejanzacolor2=[0,0,1];handles.semejanzacolor3=[0,0,1];
set(handles.text11,'String',[num2str(handles.Dr*1000),' mm']);
set(handles.text17,'string',num2str(handles.Dr*1000));set(handles.edit4,'
string',num2str(handles.Dr*1000));
guidata(hObject,handles);

function varargout = Programa_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes on button press in curvas_dimensional.
function curvas_dimensional_Callback(hObject, eventdata, handles)
set(handles.curvas_dimensional,'backgroundcolor',[1,1,1]);
set(handles.curvas_dimensional,'fontweight','bold');
cla(handles.axes1,'reset');cla(handles.axes2,'reset');
set(handles.uipanel13,'visible','off');
set(handles.semejanza,'backgroundcolor',[0.831,0.816,0.784]);
set(handles.semejanza,'fontweight','normal')
set(handles.curvasadimensionales,'backgroundcolor',[0.831,0.816,0.784]);
set(handles.curvasadimensionales,'fontweight','normal')
set(handles.axes1,'visible','on');set(handles.axes2,'visible','off');
set(handles.dosd,'visible','on');
set(handles.popupmenu52,'visible','off');
set(handles.popupmenu53,'visible','off')
if strcmp(get(handles.dosd,'string'),'Cambiar a gráfica en 2-D')==0

set(handles.axes2,'visible','on');set(handles.uipanel5,'visible','off');
set(handles.popupmenu1,'visible','on');
set(handles.popupmenu2,'visible','on');
```

```
end
if strcmp(get(handles.dosd,'string'),'Cambiar a gráfica en 2-D')==1
    cla(handles.axes2,'reset');set(handles.axes2,'visible','off');
set(handles.uipanel5,'visible','on');
    set(handles.popupmenu1,'visible','off');
set(handles.popupmenu2,'visible','off');
end
set(handles.uipanel3,'visible','on');
set(handles.uipanel8,'visible','off');
set(handles.uipanel9,'visible','off');
set(handles.uipanel10,'visible','off');
set(handles.text4,'string','Marcador');
set(handles.text6,'string','Tipo línea');
guidata(hObject,handles);uipanel3_SelectionChangeFcn(hObject,0,handles);

% --- Executes on button press in curvasadimensionales.
function curvasadimensionales_Callback(hObject, eventdata, handles)
cla(handles.axes1,'reset');cla(handles.axes2,'reset');
set(handles.uipanel13,'visible','off');
set(handles.semejanza,'backgroundcolor',[0.831,0.816,0.784]);
set(handles.semejanza,'fontweight','normal')
set(handles.curvas_dimensional,'backgroundcolor',[0.831,0.816,0.784]);
set(handles.curvas_dimensional,'fontweight','normal');
set(hObject,'backgroundcolor',[1,1,1]);set(hObject,'fontweight','bold');
set(handles.dosd,'visible','off');
set(handles.axes1,'visible','off');set(handles.axes2,'visible','off');
set(handles.popupmenu1,'visible','off');
set(handles.popupmenu2,'visible','off');
set(handles.uipanel5,'visible','off');
set(handles.uipanel8,'visible','off');
set(handles.uipanel9,'visible','off');
set(handles.uipanel10,'visible','off');
set(handles.popupmenu52,'visible','on');
set(handles.popupmenu53,'visible','on');
set(handles.uipanel3,'visible','on');
set(handles.text4,'string','Marcador');
set(handles.text6,'string','Tipo línea');
guidata(hObject,handles);uipanel3_SelectionChangeFcn(hObject,0,handles);

% --- Executes on button press in semejanza.
function semejanza_Callback(hObject, eventdata, handles)
cla(handles.axes1,'reset');cla(handles.axes2,'reset');
set(handles.uipanel13,'visible','on');
set(handles.semejanza,'backgroundcolor',[1,1,1]);
set(handles.semejanza,'fontweight','bold')
set(handles.curvasadimensionales,'backgroundcolor',[0.831,0.816,0.784]);
set(handles.curvasadimensionales,'fontweight','normal')
set(handles.curvas_dimensional,'backgroundcolor',[0.831,0.816,0.784]);
set(handles.curvas_dimensional,'fontweight','normal')
set(handles.axes1,'visible','off');set(handles.axes2,'visible','off');
set(handles.dosd,'visible','off');
cla(handles.axes1,'reset'),cla(handles.axes2,'reset');
set(handles.popupmenu52,'visible','off');
set(handles.popupmenu53,'visible','off')
set(handles.popupmenu1,'visible','off');
set(handles.popupmenu2,'visible','off');
set(handles.uipanel5,'visible','off');
set(handles.uipanel3,'visible','off');
set(handles.uipanel4,'visible','off');
set(handles.uipanel8,'visible','on');
```

```
set(handles.uipanel9,'visible','on');
set(handles.uipanel10,'visible','on');
set(handles.text4,'string','Pto. de
ensayo');set(handles.text6,'string','Curva por semejanza');
guidata(hObject,handles);
uipanel13_SelectionChangeFcn(hObject,0,handles);

% --- Executes on mouse press over figure background.
function figure1_ButtonDownFcn(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)

% --- Executes during object deletion, before destroying properties.
function axes1_DeleteFcn(hObject, eventdata, handles)

% --- Executes on mouse press over axes background.
function axes1_ButtonDownFcn(hObject, eventdata, handles)

% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)

% --------------------------------------------------------------------
function archivo_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function ayuda_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function ayuda_programa_Callback(hObject, eventdata, handles)
winopen(strcat(cd,'\AYUDA.pdf'))

% Muestra memoria Trabajon Fin de Grado
function memoria_TFG_Callback(hObject, eventdata, handles)
winopen(strcat(cd,'\MEMORIA TFG.pdf'))

% --------------------------------------------------------------------
function Imp_dat_cc_Callback(hObject, eventdata, handles)


% --------------------------------------------------------------------
function unidades_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function U_caudal_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function potencia_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function potencia_w_Callback(hObject, eventdata, handles)
set(handles.potencia_w,'Checked','on');
set(handles.potencia_kw,'Checked','off');
set(handles.potencia_cv,'Checked','off');
handles.Upotencia='W (Vatios)'; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
```

```
function potencia_kw_Callback(hObject, eventdata, handles)
set(handles.potencia_w,'Checked','off');
set(handles.potencia_kw,'Checked','on');
set(handles.potencia_cv,'Checked','off'); handles.Upotencia='W (kW)';
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function potencia_cv_Callback(hObject, eventdata, handles)
set(handles.potencia_w,'Checked','off');
set(handles.potencia_kw,'Checked','off');
set(handles.potencia_cv,'Checked','on');handles.Upotencia='W (CV)';
guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function U caudal m3h Callback(hObject, eventdata, handles)
Ucaudal='m^3/h';handles.Ucaudal=Ucaudal;guidata(hObject,handles);
set(handles.U_caudal_ls,'Checked','off');
set(handles.U_caudal_m3h,'Checked','on');
set(handles.U_caudal_lm,'Checked','off');
set(handles.gpmusa,'Checked','off');
set(handles.gpmi,'Checked','off');representar_Callback(hObject,0,handles)

% --------------------------------------------------------------------
function U_caudal_ls_Callback(hObject, eventdata, handles)
Ucaudal='l/s'; handles.Ucaudal=Ucaudal; guidata(hObject,handles);
set(handles.U_caudal_ls,'Checked','on');
set(handles.U_caudal_m3h,'Checked','off');
set(handles.U_caudal_lm,'Checked','off');
set(handles.gpmusa,'Checked','off');
set(handles.gpmi,'Checked','off');representar_Callback(hObject,0,handles)

% --------------------------------------------------------------------
function U_caudal_lm_Callback(hObject, eventdata, handles)
Ucaudal='l/min';handles.Ucaudal=Ucaudal;guidata(hObject,handles);
set(handles.U_caudal_ls,'Checked','off');
set(handles.U_caudal_m3h,'Checked','off');
set(handles.U_caudal_lm,'Checked','on');
set(handles.gpmusa,'Checked','off');
set(handles.gpmi,'Checked','off');representar_Callback(hObject,0,handles)

% --------------------------------------------------------------------
function Untitled_22_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function Copiar_Callback(hObject, eventdata, handles)



function salir_Callback(hObject, eventdata, handles)
opc=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(opc,'No'),return;end
clear,clc,close

function text1_ButtonDownFcn(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function text1_CreateFcn(hObject, eventdata, handles)

% Importar datos curvas dimensionales
```

```
function imp_cc_0_Callback(hObject, eventdata, handles)
[nombre directorio]=uigetfile({'*.dat';'*.xls'}, 'Importar
datos','MultiSelect','on');
if isequal(nombre,0),return
else
    datos=Cargar_Datos(directorio,nombre);handles.datos=datos;
RPM(1)=datos(1,1);
    for i=1:1:length(datos(:,1)), if datos(i,1)~=RPM,
RPM(length(RPM)+1)=datos(i,1); end, end
    handles.RPMo=RPM;handles.RPM=RPM;RPMo=RPM;b='0';
    for i=1:1:length(RPMo),b=[b,'|',num2str(RPMo(i))];end
    set(handles.popupmenu3,'Value',1);set(handles.popupmenu4,'Value',1);
    set(handles.popupmenu5,'Value',1);set(handles.popupmenu6,'Value',1);
    set(handles.popupmenu7,'Value',1);set(handles.popupmenu8,'Value',1);
    set(handles.popupmenu9,'Value',1);set(handles.popupmenu10,'Value',1);

set(handles.popupmenu11,'Value',1);set(handles.popupmenu12,'Value',1);
set(handles.popupmenu57,'Value',1);

set(handles.popupmenu3,'String',b);set(handles.popupmenu4,'String',b);

set(handles.popupmenu5,'String',b);set(handles.popupmenu6,'String',b);

set(handles.popupmenu7,'String',b);set(handles.popupmenu8,'String',b);

set(handles.popupmenu9,'String',b);set(handles.popupmenu10,'String',b);

set(handles.popupmenu11,'String',b);set(handles.popupmenu12,'String',b);
set(handles.popupmenu57,'String',b);

    simbolo=zeros(1,length(RPM));tlinea=zeros(1,length(RPM));
    for i=1:1:length(RPM),simbolo(i)='.';tlinea(i)='-';end
    if get(handles.dos,'value')||get(handles.radiobutton7,'value')
        a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
        simbolo(1)=a(get(handles.popupmenu13,'Value'));
        simbolo(2)=a(get(handles.popupmenu14,'Value'));
        simbolo(3)=a(get(handles.popupmenu15,'Value'));
        simbolo(4)=a(get(handles.popupmenu16,'Value'));
        simbolo(5)=a(get(handles.popupmenu17,'Value'));
        simbolo(6)=a(get(handles.popupmenu18,'Value'));
        simbolo(7)=a(get(handles.popupmenu19,'Value'));
        simbolo(8)=a(get(handles.popupmenu20,'Value'));
        simbolo(9)=a(get(handles.popupmenu21,'Value'));
        simbolo(10)=a(get(handles.popupmenu22,'Value'));
        simbolo(11)=a(get(handles.popupmenu23,'Value'));

RPM=0;RPM(11)=str2double(get(handles.edit1,'string'));handles.RPM=RPM;
        k=[0,1,2,3,4];
        tlinea(1)=k(get(handles.popupmenu35,'Value'));
        tlinea(2)=k(get(handles.popupmenu36,'Value'));
        tlinea(3)=k(get(handles.popupmenu37,'Value'));
        tlinea(4)=k(get(handles.popupmenu38,'Value'));
        tlinea(5)=k(get(handles.popupmenu39,'Value'));
        tlinea(6)=k(get(handles.popupmenu40,'Value'));
        tlinea(7)=k(get(handles.popupmenu41,'Value'));
        tlinea(8)=k(get(handles.popupmenu42,'Value'));
        tlinea(9)=k(get(handles.popupmenu43,'Value'));
        tlinea(10)=k(get(handles.popupmenu44,'Value'));
        tlinea(11)=k(get(handles.popupmenu45,'Value'));
    end
```

```
    if
get(handles.uno,'value')==1&&strcmp(get(handles.semejanza,'fontweight'),'
normal')
        color=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
        for i=8:1:length(RPM),color(i,:)=rand(1,3);end
        handles.color=color;
        tlinea=zeros(1,length(RPM));
    end
    handles.simbolo=simbolo;handles.tlinea=tlinea;
    guidata(hObject,handles);
    if strcmp(get(handles.semejanza,'fontweight'),'bold'),
        semejanza_Callback(hObject,0,handles);
    else
       representar_Callback(hObject,0,handles);
    end
end

function figure1_WindowButtonMotionFcn(hObject, eventdata, handles)

function cte_fluido_Callback(hObject, eventdata, handles)
prompt={'g (m/s^2):','\rho (kg/m^3):','\mu (kg/m·s):'};
name='Constantes de ensayo'; numlines=1;
defaultanswer{1}=num2str(handles.g);
defaultanswer{2}=num2str(handles.rho);
defaultanswer{3}=num2str(handles.mu);
options.Resize='off'; options.WindowStyle='normal';
options.Interpreter='tex';
[answer]=inputdlg(prompt,name,numlines,defaultanswer,options);
if ~isempty(answer)
    g=str2double(answer(1));
    rho=str2double(answer(2));mu=str2double(answer(3));
    if ~isnan(g)&&~isnan(rho)&&~isnan(mu)
        handles.g=g; handles.rho=rho; handles.mu=mu;
        guidata(hObject,handles);
        representar_Callback(hObject,0,handles);
    end
end

function dosd_Callback(hObject, eventdata, handles)
if strcmp(get(hObject,'string'),'Cambiar a gráfica en 2-D')==0
    set(hObject,'string','Cambiar a gráfica en 2-D');
    cla(handles.axes2,'reset');set(handles.axes2,'visible','off');
    set(handles.uipanel5,'visible','on');
    set(handles.popupmenu1,'visible','off');
    set(handles.popupmenu2,'visible','off');
else
    set(hObject,'string','Cambiar a gráfica en 3-D');
    set(handles.axes2,'visible','on');
    set(handles.uipanel5,'visible','off');
    set(handles.popupmenu1,'visible','on');
    set(handles.popupmenu2,'visible','on');
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);

function ngiro_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function ngiro_rpm_Callback(hObject, eventdata, handles)
set(handles.ngiro_rpm,'Checked','on');
set(handles.ngiro_rad,'Checked','off');
```

```
handles.vgiro='\Omega (RPM)'; guidata(hObject,handles);
if strcmp(get(handles.uipanel5,'visible'),'on');
    representar_Callback(hObject,0,handles);
end

% --------------------------------------------------------------------
function ngiro_rad_Callback(hObject, eventdata, handles)
set(handles.ngiro_rpm,'Checked','off');
set(handles.ngiro_rad,'Checked','on');
handles.vgiro='\Omega (Rad/s)';guidata(hObject,handles);
if strcmp(get(handles.uipanel5,'visible'),'on');
    representar_Callback(hObject,0,handles);
end

% --- Ventana emergente con datos cargados
function verdatoscurvac Callback(hObject, eventdata, handles)
datos=handles.datos;
dialgtable(datos)

% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)

% Ventana emergente con datos referentes a la bomba
function cte_bomba_Callback(hObject, eventdata, handles)
answer=dialginputimagen(handles.dz*100,handles.Di*100,handles.Da*100,hand
les.Dr*100);
if strcmp(answer,'Yes')
    return
else
    Dr=str2double(answer(4))/100;Da=str2double(answer(3))/100;
    Di=str2double(answer(2))/100;dz=str2double(answer(1))/100;
    if ~isnan(Da)&&~isnan(Di)&&~isnan(dz)&&~isnan(Dr)
        handles.Da=Da;handles.Di=Di;handles.dz=dz;handles.Dr=Dr;
        set(handles.text11,'String',[num2str(handles.Dr*1000),' mm']);
    end
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --- Exportar graficos
function guardargrafica_Callback(hObject, eventdata, handles)
h=figure; a=handles.mapacolor; colormap(a)
if strcmp(get(handles.dosd,'string'),'Cambiar a gráfica en 2-
D')&&strcmp(get(handles.curvas_dimensional,'fontweight'),'bold')
    copyobj(handles.axes1,h);
else
    eleccion=questdlg('¿Que grafica desea exportar?', ...
        'Exportar gráfica...','Superior','Inferior','Ambas','Ambas');
    switch eleccion
        case 'Superior'
            copyobj(handles.axes1,h);
        case 'Inferior'
            copyobj(handles.axes2,h);
        case 'Ambas'
            copyobj(handles.axes1,h); copyobj(handles.axes2,h);
    end
    if isempty(eleccion),close Figure 1,return,end

end
formatos = {'*.jpg','JPEG (*.jpg)';'*.tif','TIFF (*.tif)';'*.bmp',...
    'BMP (*.bmp)';'*.fig','FIG (*.fig)';'*.pbm','PBM (*.pbm)';...
```

```
    '*.pdf','PDF (*.pdf)';'*.png','PNG (*.png)';'*.ppm','PPM (*.ppm)'};
[nomb,ruta] = uiputfile(formatos,'GUARDAR GRAFICA');
if nomb==0,close Figure 1, return, end
fName = fullfile(ruta,nomb);
saveas(h,fName)
close Figure 1

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in representar.
function representar_Callback(hObject, eventdata, handles)
g=handles.g;rho=handles.rho;Da=handles.Da;Di=handles.Di;dz=handles.dz;
Ucaudal=handles.Ucaudal;Upresion=handles.Upresion;Dr=handles.Dr;
RPM=handles.RPM;RPMo=handles.RPMo;datos=handles.datos;mu=handles.mu;
simbolo=handles.simbolo;
tlinea=handles.tlinea;
color=handles.color;
if strcmp(Ucaudal,'l/s'),qx=1/60;end, if strcmp(Ucaudal,'l/min'),qx=1;end
if strcmp(Ucaudal,'m^3/h'),qx=60/1000;end,
if strcmp(Ucaudal,'gpm U.S.A.'),qx=1/4.546;end
if strcmp(Ucaudal,'gpm U.K.'),qx=1/3.785;end
if strcmp(Upresion,'H_m (ft)'),px=1/0.3048;end,
if strcmp(Upresion,'P (Bar)'),px=rho*g*10^-5;end
if strcmp(Upresion,'H_m (m)'),px=1;end,
if strcmp(Upresion,'P (Pascal)'),px=rho*g;end
if strcmp(Upresion,'P (psi)'),px=rho*g/(6.894757*10^3);end
if strcmp(handles.Upotencia,'W (CV)'),wx=1/736;end,
if strcmp(handles.Upotencia,'W (kW)'),wx=10^-3;end
if strcmp(handles.Upotencia,'W (Vatios)'),wx=1;end
if strcmp(handles.vgiro,'\Omega (RPM)'),vx=1;end,
if strcmp(handles.vgiro,'\Omega (Rad/s)'),vx=pi/30;end

%============================================= Curvas Dimensionales
if strcmp(get(handles.curvas_dimensional,'fontweight'),'bold')
    if strcmp(get(handles.dosd,'string'),'Cambiar a gráfica en 2-D')==0
           cla(handles.axes1,'reset'),axes(handles.axes1);
           guidata(hObject,handles);egrid=handles.egrid;
           feval(@grid,egrid)
           hold on
            if
strcmp(get(handles.estadoisorendimiento,'Checked'),'on')&&length(RPM(RPM~
=0))>2; %__________Inicio Isorendimiento
                q=0;
                for i=1:1:length(RPMo)
                    clear qv hv Wuv Wbv nv
                    qv=datos(datos(:,1)==RPMo(i),5);
                    hv=(datos(datos(:,1)==RPMo(i),3)-datos(datos...
                         (:,1)==RPMo(i),2))/(g*rho*10^-5)...
                        +dz+(((4/(pi*Di^2)-4/(pi*Da^2))...
                        *((1/60000)*qv)).^2)/(2*g);
                    Wuv=g*rho*(1/60000)*qv.*hv;
```

```
Wbv=datos(datos(:,1)==RPMo(i),4).*datos(datos(:,1)==RPMo(i),1)*(2*pi/60);
                    if length(qv)>3,
hv=polyval(polyfit(qv,hv,handles.ordenh),linspace(min(qv),max(qv),100));

Wuv=polyval(polyfit(qv,Wuv,handles.ordenwu),linspace(min(qv),max(qv),100)
);

Wbv=polyval(polyfit(qv,Wbv,handles.ordenwb),linspace(min(qv),max(qv),100)
);
                    nv=100*Wuv./Wbv;
                    qv=linspace(min(qv),max(qv),100);l=length(q);
                    for j=1:1:length(qv)
                        q(l+j)=qv(j);h(l+j)=hv(j);n(l+j)=nv(j);
                    end
                    end
                end
                q=q*qx;h=h*px; q=q(h~=0); n=n(h~=0); h=h(h~=0);
                ejeq=linspace(min(q),max(q),101);
                ejeh=linspace(min(h),max(h),100);
                [QQ,HH]=meshgrid(ejeq,ejeh);
                nqh=gridfit(q,h,n,ejeq,ejeh);
                indnqh=isnan(griddata(q,h,n,QQ,HH,'linear'));
                nqh(indnqh==1)=NaN;
                if strcmp(get(handles.colorisorendimiento,
'Checked'),'on');
                    if strcmp(get(handles.autonumeroisolineas,
'Checked'),'on');
                        [C,h] = contourf(QQ,HH,nqh);
                    else
                        [C,h] =
contourf(QQ,HH,nqh,handles.numeroisolineas);
                    end
                else
                    if strcmp(get(handles.autonumeroisolineas,
'Checked'),'on');
                        [C,h] = contour(QQ,HH,nqh);
                    else
                        [C,h] =
contour(QQ,HH,nqh,handles.numeroisolineas);
                    end
                end
                a=handles.mapacolor;
                colormap(a);

                if
strcmp(get(handles.Manuetiqisorendimiento,'Checked'),'on');
                    y=msgbox('Haga clic en los puntos de las curvas de
isorendimiento en los que quiera que figure una etiqueta. Para terminar
de colocar etiquetas pulse la tecla Intro',' Etiquetado manual');
                    waitfor(y);
                    text_handle = clabel(C,h,'manual');
                    set(text_handle,'BackgroundColor',[1 1 1]);
                end
                if
strcmp(get(handles.autoetiqisorendimiento,'Checked'),'on');
                    text_handle =
clabel(C,h,'FontSize',12,'Color','k','Rotation',0,'LabelSpacing',200);
                    set(text_handle,'BackgroundColor',[1 1 1]);
                end %%_____________________ Fin isorendimiento
            end
```

```
            if
strcmp(get(handles.estadoisoconsumo,'Checked'),'on')&&length(RPM(RPM~=0))
>2; %_____________________________________________Inicio Isoconsumo
                q=0;clear h
                for i=1:1:length(RPMo)
                    clear qv hv Wbv ejeq ejeh
                    qv=datos(datos(:,1)==RPMo(i),5);
                    hv=(datos(datos(:,1)==RPMo(i),3)-datos(datos(:,1)...
                        ==RPMo(i),2))/(g*rho*10^-5)...
                        +dz+(((4/(pi*Di^2)-4/(pi*Da^2))*((1/60000...
                        )*qv)).^2)/(2*g);

Wbv=datos(datos(:,1)==RPMo(i),4).*datos(datos(:,1)==RPMo(i),1)*(2*pi/60);
                    if length(qv)>3,
hv=polyval(polyfit(qv,hv,handles.ordenh),linspace(min(qv),max(qv),100));

Wbv=polyval(polyfit(qv,Wbv,handles.ordenwb),linspace(min(qv),max(qv),100)
);
                        qv=linspace(min(qv),max(qv),100);l=length(q);
                        for j=1:1:length(qv),
                            q(l+j)=qv(j);h(l+j)=hv(j);Wb(l+j)=Wbv(j);
                        end
                    end
                end
                q=q*qx; h=h*px; Wb=Wb*wx; q=q(h~=0); Wb=Wb(h~=0);
                h=h(h~=0);
                ejeq=linspace(min(q),max(q),101);
ejeh=linspace(min(h),max(h),100);
                [QQ,HH]=meshgrid(ejeq,ejeh);
Wbqh=gridfit(q,h,Wb,ejeq,ejeh);

indWbqh=isnan(griddata(q,h,Wb,QQ,HH,'linear'));Wbqh(indWbqh==1)=NaN;
                if strcmp(get(handles.colorisoconsumo, 'Checked'),'on');
                    if strcmp(get(handles.autonumisoliconsumo,
'Checked'),'on');
                        [C,h] = contourf(QQ,HH,Wbqh);
                    else
                        [C,h] =
contourf(QQ,HH,Wbqh,handles.numeroisoconsumo);
                    end
                else
                    if strcmp(get(handles.autonumisoliconsumo,
'Checked'),'on');
                        [C,h] = contour(QQ,HH,Wbqh);
                    else
                        [C,h] =
contour(QQ,HH,Wbqh,handles.numeroisoconsumo);
                    end
                end
                a=handles.mapacolor;
                colormap(a);

                if
strcmp(get(handles.Manuetiqisoconsumo,'Checked'),'on');
                    y=msgbox('Haga clic en los puntos de las curvas de
isoconsumo en los que quiera que figure una etiqueta. Para terminar de
colocar etiquetas pulse la tecla Intro',' Etiquetado manual');
                    waitfor(y);
                    text_handle = clabel(C,h,'manual');
                    set(text_handle,'BackgroundColor',[1 1 1]);
                end
```

```
                if
strcmp(get(handles.autoetiqisoconsumo,'Checked'),'on');
                    text_handle =
clabel(C,h,'FontSize',12,'Color','k','Rotation',0,'LabelSpacing',200);
                    set(text_handle,'BackgroundColor',[1 1 1]);
                end %%_____________________Fin isoconsumo
            end
            for i=1:1:length(RPM)%%          Inicio Gráfica Hm(Q)
                clear qv hv Wuv Wbv nv tv
                q=datos(datos(:,1)==RPM(i),5);
                h=(datos(datos(:,1)==RPM(i),3)-datos(datos...
                (:,1)==RPM(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)-...
                4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);

                if isempty(datos(datos(:,1)==RPM(i),1))&&RPM(i)~=0 %
                    [v,p]=min(abs(datos(:,1)-RPM(i))); RPMref=datos(p,1);
                    q=datos(datos(:,1)==RPMref,5);
                    h=(datos(datos(:,1)==RPMref,3)-datos(datos...
                    (:,1)==RPMref,2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)...
                    -4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
                end

                q=q*qx;h=h*px;
                if
~strcmp(char(simbolo(i)),'N')&&~isempty(datos(datos(:,1)==RPM(i),1))
                    plot(q,h,char(simbolo(i)),'color',color(i,:))
                end
                if length(q)>3,pol_h=polyfit(q,h,handles.ordenh);
                    if isempty(datos(datos(:,1)==RPM(i),1))&&RPM(i)~=0

pol_h=pol_h*((RPM(i)/RPMref)^2);q=q*(RPM(i)/RPMref);
                    end
                    if tlinea(i)==0

plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),'color',color(i,:))
                    end
                    if tlinea(i)==1

plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),'--','color',color(i,:))
                    end
                    if tlinea(i)==2

plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),':','color',color(i,:))
                    end
                    if tlinea(i)==3

plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),'-.','color',color(i,:))
                    end
                end

            end%%_____________________________________ Fin Gráfica Hm(Q)

            ylabel(Upresion);%%_________ Inicio Gráfica Wu(Q),Wb(Q),...
            hold off
            cla(handles.axes2,'reset')
            axes(handles.axes2);
            xlabel(['Q (',Ucaudal,')'])
```

```
            guidata(hObject,handles);egrid=handles.egrid;
            feval(@grid,egrid)
            hold on
            clear pol_h
            pol_h=0;
            if
get(handles.popupmenu1,'value')~=4&&get(handles.popupmenu2,'value')==1||g
et(handles.popupmenu1,'value')==4&&get(handles.popupmenu2,'value')~=1
            for i=1:1:length(RPM)
                q=datos(datos(:,1)==RPM(i),5);
                h=(datos(datos(:,1)==RPM(i),3)-datos(datos(:,1)...
                   ==RPM(i),2))/(g*rho*10^-5)...
                   +dz+(((4/(pi*Di^2)-4/(pi*Da^2))*...
                   ((1/60000)*q)).^2)/(2*g);

Wb=datos(datos(:,1)==RPM(i),4).*datos(datos(:,1)==RPM(i),1)*(2*pi/60);
                t=datos(datos(:,1)==RPM(i),4);
                if isempty(datos(datos(:,1)==RPM(i),1))&&RPM(i)~=0 %
                    [v,p]=min(abs(datos(:,1)-RPM(i))); RPMref=datos(p,1);
                    q=datos(datos(:,1)==RPMref,5);
                    h=(datos(datos(:,1)==RPMref,3)-datos(datos(:,1)==...
                       RPMref,2))/(g*rho*10^-5)+dz...
                       +(((4/(pi*Di^2)-4/(pi*Da^2))...
                       *((1/60000)*q)).^2)/(2*g);

Wb=datos(datos(:,1)==RPMref,4).*datos(datos(:,1)==RPMref,1)*(2*pi/60);
                    t=datos(datos(:,1)==RPMref,4);
                end
                Wu=g*rho*(1/60000)*q.*h;
                if length(q)>3,pol_Wu=polyfit(q*qx,Wu*wx,handles.ordenwu);
                    if isempty(datos(datos(:,1)==RPM(i),1)),
                       pol_Wu=pol_Wu*((RPM(i)/RPMref)^3);
                    end
                end

                if length(q)>3,pol_Wb=polyfit(q*qx,Wb*wx,handles.ordenwb);
                    if isempty(datos(datos(:,1)==RPM(i),1)),
                       pol_Wb=pol_Wb*((RPM(i)/RPMref)^3);
                    end
                end

                if ~isempty(datos(datos(:,1)==RPM(i),1)),
                    n=100*Wu./Wb;
                    if length(q)>3,
                       pol_n=polyfit(q*qx,n,handles.ordenN);
                    end
                end



                if length(q)>3,pol_t=polyfit(q*qx,t,3);
                    if isempty(datos(datos(:,1)==RPM(i),1)),
                       pol_t=pol_t*((RPM(i)/RPMref)^3);
                    end
                end
                if
get(handles.popupmenu1,'value')==1||get(handles.popupmenu1,'value')==3
                    q=q*qx;h=Wu*wx; ylabel(handles.Upotencia);
                    if length(q)>3,pol_h=pol_Wu;end
                end
                if get(handles.popupmenu1,'value')==2,q=q*qx;h=Wb*wx;
                    ylabel(handles.Upotencia);
```

```
                    if length(q)>3,pol_h=pol_Wb;end
                end
                if
get(handles.popupmenu2,'value')==2&&~isempty(datos(datos(:,1)==RPM(i),1))
                    q=q*qx;h=n;ylabel('\eta (%)');
                    if length(q)>3,pol_h=pol_n;end
                end
                if get(handles.popupmenu2,'value')==3,
                    q=q*qx;h=t;ylabel('T (N·m)');
                    if length(q)>3,pol_h=pol_t;end
                end
                if
~strcmp(char(simbolo(i)),'N')&&~isempty(datos(datos(:,1)==RPM(i),1))
                    plot(q,h,char(simbolo(i)),'color',color(i,:))
                    if get(handles.popupmenu1,'value')==3 %Usado para
imprimir Wu y Wb a la vez.
                        if
length(q)>3,plot(q,Wb*wx,char(simbolo(i)),'color',color(i,:)),end
                    end
                end
                if pol_h==0,q=[];end
                if isempty(datos(datos(:,1)==RPM(i),1))&&RPM(i)~=0,
                    q=q*(RPM(i)/RPMref);
                end
                if length(q)>3
                    if tlinea(i)==0
plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),'color',color(i,:))
                        if get(handles.popupmenu1,'value')==3 %Usado para
imprimir Wu y Wb a la vez.

plot(linspace(min(q),max(q),200),polyval(pol_Wb,linspace(min(q),max(q),20
0)),'color',color(i,:))
                        end
                    end
                    if tlinea(i)==1

plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),'--','color',color(i,:))
                        if get(handles.popupmenu1,'value')==3 %Usado para
imprimir Wu y Wb a la vez.

plot(linspace(min(q),max(q),200),polyval(pol_Wb,linspace(min(q),max(q),20
0)),'--','color',color(i,:))
                        end
                    end
                    if tlinea(i)==2

plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),':','color',color(i,:))
                        if get(handles.popupmenu1,'value')==3 %Usado para
imprimir Wu y Wb a la vez.

plot(linspace(min(q),max(q),200),polyval(pol_Wb,linspace(min(q),max(q),20
0)),':','color',color(i,:))
                        end
                    end
                    if tlinea(i)==3

plot(linspace(min(q),max(q),200),polyval(pol_h,linspace(min(q),max(q),200
)),'-.','color',color(i,:))
```

```
                        if get(handles.popupmenu1,'value')==3 %Usado para
imprimir Wu y Wb a la vez.

plot(linspace(min(q),max(q),200),polyval(pol_Wb,linspace(min(q),max(q),20
0)),'-.','color',color(i,:))
                        end
                    end
                end
            end

end%_____________________________________________________________
_____________________________________________ Fin Gráfica Wu(Q),Wb(Q),...
            if
get(handles.popupmenu1,'value')~=4&&get(handles.popupmenu2,'value')~=1&&l
ength(RPM(RPM~=0))>=1 %____________________Inicio Gráfica con dos ejes
verticales

a=zeros(100,length(RPM(RPM~=0)));b=zeros(100,length(RPM(RPM~=0)));c=zeros
(100,length(RPM(RPM~=0)));

d=zeros(100,length(RPM(RPM~=0)));e=zeros(100,length(RPM(RPM~=0)));
                RPMl=RPM(RPM~=0);
                for i=1:1:length(RPMl)
                    clear q
                    q=datos(datos(:,1)==RPMl(i),5);
                    h=(datos(datos(:,1)==RPMl(i),3)-datos(datos(:,1)...
                       ==RPMl(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)-...
                       4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
                    Wu=g*rho*(1/60000)*q.*h;
                    if
length(q)>3,Wu_int=polyval(polyfit(q*qx,Wu*wx,handles.ordenwu),linspace(m
in(q),max(q),100)*qx);end

Wb=datos(datos(:,1)==RPMl(i),4).*datos(datos(:,1)==RPMl(i),1)*(2*pi/60);
                    if
length(q)>3,Wb_int=polyval(polyfit(q*qx,Wb*wx,handles.ordenwb),linspace(m
in(q),max(q),100)*qx);end
                    n=100*Wu./Wb;
                    if
length(q)>3,n_int=polyval(polyfit(q*qx,n,handles.ordenN),linspace(min(q),
max(q),100)*qx);end
                    t=datos(datos(:,1)==RPMl(i),4);
                    if
length(q)>3,t_int=polyval(polyfit(q*qx,t,handles.ordenT),linspace(min(q),
max(q),100)*qx);end
                    if length(q)>3,q=linspace(min(q),max(q),100)*qx;end
                    for j=1:1:length(q)

a(j,i)=q(j);b(j,i)=Wu_int(j);c(j,i)=n_int(j);d(j,i)=t_int(j);e(j,i)=Wb_in
t(j);
                    end
                end
                if
get(handles.popupmenu1,'value')==1&&get(handles.popupmenu2,'value')==2
                [hAx,hLine1,hLine2]=plotyy(a,b,a,c);
                set(hLine1,'LineStyle','-');
                set(hLine2,'LineStyle','--');
                xlabel(['Q (',Ucaudal,')']);
                ylabel(hAx(1),handles.Upotencia);
                ylabel(hAx(2),'\eta (%)');
                end
```

```
                if
get(handles.popupmenu1,'value')==1&&get(handles.popupmenu2,'value')==3
                [hAx,hLine1,hLine2]=plotyy(a,b,a,d);
                set(hLine1,'LineStyle','-');
                set(hLine2,'LineStyle','--');
                xlabel(['Q (',Ucaudal,')']);
ylabel(hAx(1),handles.Upotencia); ylabel(hAx(2),'T (N·m)');
                end
                if
get(handles.popupmenu1,'value')==2&&get(handles.popupmenu2,'value')==2
                [hAx,hLine1,hLine2]=plotyy(a,e,a,c);
                set(hLine1,'LineStyle','-');
                set(hLine2,'LineStyle','--');
                xlabel(['Q (',Ucaudal,')']);
                ylabel(hAx(1),handles.Upotencia);
                ylabel(hAx(2),'\eta (%)');
                end
                if
get(handles.popupmenu1,'value')==2&&get(handles.popupmenu2,'value')==3
                [hAx,hLine1,hLine2]=plotyy(a,e,a,d);
                set(hLine1,'LineStyle','-');
                set(hLine2,'LineStyle','--');
                xlabel(['Q (',Ucaudal,')']);
                ylabel(hAx(1),handles.Upotencia);
                ylabel(hAx(2),'T (N·m)');
                end
                if
get(handles.popupmenu1,'value')==3&&get(handles.popupmenu2,'value')==3
                [hAx,hLine1,hLine2]=plotyy(horzcat(a,a),horzcat(b,e),a,d);
                set(hLine1,'LineStyle','-');
                set(hLine2,'LineStyle','--');
                xlabel(['Q (',Ucaudal,')']);
                ylabel(hAx(1),handles.Upotencia);
                ylabel(hAx(2),'T (N·m)');
                end
                if
get(handles.popupmenu1,'value')==3&&get(handles.popupmenu2,'value')==2
                [hAx,hLine1,hLine2]=plotyy(horzcat(a,a),horzcat(b,e),a,c);
                set(hLine1,'LineStyle','-');
                set(hLine2,'LineStyle','--');
                xlabel(['Q (',Ucaudal,')']);
                ylabel(hAx(1),handles.Upotencia);
                ylabel(hAx(2),'\eta (%)');
                end
            end %_____________Fin gráfica con dos ejes verticales
    end
     if strcmp(get(handles.dosd,'string'),'Cambiar a gráfica en 3-D')==0;
%                           Inicio Gráficas 3D
         if handles.z==0&&length(find(RPM))>=3;
          q=0;
            for i=1:1:length(RPM)
                if RPM(i)~=0
                    clear qv hv Wuv Wbv nv tv
                    qv=datos(datos(:,1)==RPM(i),5);
                    vgirov=RPM(i)*ones(length(qv));
                    hv=(datos(datos(:,1)==RPM(i),3)-datos(datos(:,1)...
                       ==RPM(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)- ...
                       4/(pi*Da^2))*((1/60000)*qv)).^2)/(2*g);
                    Wuv=g*rho*(1/60000)*qv.*hv;
                    Wbv=datos(datos(:,1)==RPM(i),4).*datos...
                        (datos(:,1)==RPM(i),1)*(2*pi/60);
```

```
                    tv=datos(datos(:,1)==RPM(i),4);
                    l=length(q);nv=100*Wuv./Wbv;
                    for j=1:1:length(qv)
                        q(l+j)=qv(j); h(l+j)=hv(j); n(l+j)=nv(j);
                        t(l+j)=tv(j); Wu(l+j)=Wuv(j); Wb(l+j)=Wbv(j);
                        vgiro(l+j)=vgirov(j);
                    end
                end
            end
        end
        if handles.z==1&&length(find(RPM))>=3;
            q=0;
            for i=1:1:length(RPM)
                if RPM(i)~=0
                clear qv hv Wuv Wbv nv tv
                qv=datos(datos(:,1)==RPM(i),5);
                vgirov=RPM(i)*ones(length(qv));
                hv=(datos(datos(:,1)==RPM(i),3)-datos(datos...
                   (:,1)==RPM(i),2))/(g*rho*10^-5)+dz+(((4/...
                   (pi*Di^2)-4/(pi*Da^2))*( ...
                   (1/60000)*qv)).^2)/(2*g);
                Wuv=g*rho*(1/60000)*qv.*hv;
                Wbv=datos(datos(:,1)==RPM(i),4).* ...
                     datos(datos(:,1)==RPM(i),1)*(2*pi/60);
                tv=datos(datos(:,1)==RPM(i),4);
                if length(qv)>3,
hv=polyval(polyfit(qv,hv,handles.ordenh),linspace(min(qv),max(qv),100));
                end
                if length(qv)>3,
Wuv=polyval(polyfit(qv,Wuv,handles.ordenwu),linspace(min(qv),max(qv),100)
);
                end
                if length(qv)>3,
Wbv=polyval(polyfit(qv,Wbv,handles.ordenwb),linspace(min(qv),max(qv),100)
);
                end
                if length(qv)>3,
tv=polyval(polyfit(qv,tv,handles.ordenT),linspace(min(qv),max(qv),100));
                end
                nv=100*Wuv./Wbv;
                qv=linspace(min(qv),max(qv),100);l=length(q);
                for j=1:1:length(qv)
                    q(l+j)=qv(j); h(l+j)=hv(j); n(l+j)=nv(j);
                    t(l+j)=tv(j); Wu(l+j)=Wuv(j); Wb(l+j)=Wbv(j);
                    vgiro(l+j)=vgirov(j);
                end
                end
            end
        end

        if
handles.z==0&&length(find(RPM))>=3||handles.z==1&&length(find(RPM))>=3;

            if get(handles.popupmenu51,'value')==2,n=Wb*wx;end
            if get(handles.popupmenu51,'value')==3,n=Wu*wx;end
            if get(handles.popupmenu51,'value')==4,n=t;end
            q=q*qx; q=q(h~=0); n=n(h~=0);
            h=h(h~=0);h=h*px;vgiro=vgiro(h~=0);
            if get(handles.popupmenu49,'value')==1,x=q;end
            if get(handles.popupmenu49,'value')==2,x=h;end
            if get(handles.popupmenu49,'value')==3,x=vgiro*vx;end
```

```
            if get(handles.popupmenu50,'value')==1,y=h;end
            if get(handles.popupmenu50,'value')==2,y=q;end
            if get(handles.popupmenu50,'value')==3,y=vgiro*vx;end
            ejeq=linspace(min(x),max(x),100);
            ejeh=linspace(min(y),max(y),100);
            [QQ,HH]=meshgrid(ejeq,ejeh); nqh=gridfit(x,y,n,ejeq,ejeh);
            indnqh=isnan(griddata(x,y,n,QQ,HH));nqh(indnqh==1)=NaN;
            axes(handles.axes1);
            surf(QQ,HH,nqh);
            a=handles.mapacolor;
            colormap(a);
            camlight right; lighting phong;
            shading interp;egrid=handles.egrid;feval(@grid,egrid)

            if get(handles.popupmenu51,'value')==1,zlabel('\eta (%)');end
            if get(handles.popupmenu51,'value')==2,
                zlabel(handles.Upotencia);
            end
            if get(handles.popupmenu51,'value')==3,
                zlabel(handles.Upotencia);
            end
            if get(handles.popupmenu51,'value')==4,zlabel('T(N·m)');end
            if get(handles.popupmenu49,'value')==1,
                xlabel(['Q (',Ucaudal,')']);
            end
            if get(handles.popupmenu49,'value')==2,
                xlabel(Upresion);
            end
            if get(handles.popupmenu49,'value')==3,
                xlabel(handles.vgiro);
            end
            if get(handles.popupmenu50,'value')==1,ylabel(Upresion);end
            if get(handles.popupmenu50,'value')==2,
                ylabel(['Q (',Ucaudal,')']);
            end
            if get(handles.popupmenu50,'value')==3,
                ylabel(handles.vgiro);
            end
        end
        if length(find(RPM))<3
            warndlg('Para poder representar una superficie debe
seleccionar al menos tres valores de RPM','Nota:');
        end
    end
end
if strcmp(get(handles.curvasadimensionales,'fontweight'),'bold')
%================================================== Curvas Adimensionales
    cla(handles.axes1,'reset'),axes(handles.axes1);
    guidata(hObject,handles);egrid=handles.egrid;feval(@grid,egrid)
    hold on
    for i=1:1:length(RPM)%%_________ Inicio Gráfica 1
        q=datos(datos(:,1)==RPM(i),5);
        h=(datos(datos(:,1)==RPM(i),3)-datos(datos(:,1)==...
           RPM(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)-...
           4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
        P_q=(q/60000)/((RPM(i)*pi/30)*((Dr)^3));x=P_q;

        if get(handles.popupmenu52,'value')==1,
            P_ghm=(g*h)/(((RPM(i)*pi/30)^2)*((Dr)^2)); y=P_ghm;
            ylabel('\Pi_ _g_·_H_m');orden=handles.ordenh;
        end
```

```
        if get(handles.popupmenu52,'value')==2,
            Wu=g*rho*(1/60000)*q.*h;
            P_Wu=Wu/(rho*((RPM(i)*pi/30)^3)*(Dr^5)); y=P_Wu;
            ylabel('\Pi_ _W_u');orden=handles.ordenwu;
        end
        if get(handles.popupmenu52,'value')==3,
            Wb=datos(datos(:,1)==RPM(i),4).*datos(datos...
            (:,1)==RPM(i),1)*(2*pi/60);
            P_Wb=Wb/(rho*((RPM(i)*pi/30)^3)*(Dr^5)); y=P_Wb;
            ylabel('\Pi_ _W_b');orden=handles.ordenwb;
        end

        if get(handles.popupmenu52,'value')==4,
            Wu=g*rho*(1/60000)*q.*h;
            Wb=datos(datos(:,1)==RPM(i),4).*datos...
                (datos(:,1)==RPM(i),1)*(2*pi/60);
            y=Wu*100./Wb;ylabel('\eta');orden=handles.ordenN;
        end

        if
~strcmp(char(simbolo(i)),'N')&&~isempty(datos(datos(:,1)==RPM(i),1))
            plot(x,y,char(simbolo(i)),'color',color(i,:))
        end
        if length(x)>3&&get(handles.dos,'value')==1,
            pol_y=polyfit(x,y,orden);
            if tlinea(i)==0
                plot(linspace(min(x),max(x),200),polyval(pol_y,...
                    linspace(min(x),max(x),200)),'color',color(i,:))
            end
            if tlinea(i)==1
                plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                    linspace(min(x),max(x),200)),'--','color',color(i,:))
            end
            if tlinea(i)==2
                plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                    linspace(min(x),max(x),200)),':','color',color(i,:))
            end
            if tlinea(i)==3
                plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                    linspace(min(x),max(x),200)),'-.','color',color(i,:))
            end
        end
    end%%_____________________________ Fin Gráfica 1
    cla(handles.axes2,'reset'),axes(handles.axes2);
    guidata(hObject,handles);egrid=handles.egrid;feval(@grid,egrid)
    hold on, xlabel('\Pi_Q');
    for i=1:1:length(RPM)%%_________ Inicio Gráfica 2
        q=datos(datos(:,1)==RPM(i),5);
        h=(datos(datos(:,1)==RPM(i),3)-datos(datos(:,1)==...
           RPM(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)-4/...
           (pi*Da^2))*((1/60000)*q)).^2)/(2*g);
        P_q=(q/60000)/((RPM(i)*pi/30)*((Dr)^3));x=P_q;

        if get(handles.popupmenu53,'value')==1,
            Wu=g*rho*(1/60000)*q.*h;
            P Wu=Wu/(rho*((RPM(i)*pi/30)^3)*(Dr^5)); y=P Wu;
            ylabel('\Pi_ _W_u');orden=handles.ordenwu;
        end
        if get(handles.popupmenu53,'value')==2,
            P_ghm=(g*h)/(((RPM(i)*pi/30)^2)*((Dr)^2)); y=P_ghm;
            ylabel('\Pi_ _g_·_H_m');orden=handles.ordenh;
```

```
        end
        if get(handles.popupmenu53,'value')==3,
            Wb=datos(datos(:,1)==RPM(i),4).*datos...
               (datos(:,1)==RPM(i),1)*(2*pi/60);
            P_Wb=Wb/(rho*((RPM(i)*pi/30)^3)*(Dr^5)); y=P_Wb;
            ylabel('\Pi_ _W_b');orden=handles.ordenwb;
        end

        if get(handles.popupmenu53,'value')==4,
            Wu=g*rho*(1/60000)*q.*h;
            Wb=datos(datos(:,1)==RPM(i),4).*datos(datos(:,1)...
               ==RPM(i),1)*(2*pi/60);
            y=Wu*100./Wb;
            ylabel('\eta');orden=handles.ordenN;
        end

        if
~strcmp(char(simbolo(i)),'N')&&~isempty(datos(datos(:,1)==RPM(i),1))
            plot(x,y,char(simbolo(i)),'color',color(i,:))
        end
        if length(x(x~=0))>3&&get(handles.dos,'value')==1,
            pol_y=polyfit(x,y,orden);
            if tlinea(i)==0
                plot(linspace(min(x),max(x),200),polyval(pol_y,...
                   linspace(min(x),max(x),200)),'color',color(i,:))
            end
            if tlinea(i)==1
                plot(linspace(min(x),max(x),200),polyval(pol_y,...
                   linspace(min(x),max(x),200)),'--','color',color(i,:))
            end
            if tlinea(i)==2
                plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                   linspace(min(x),max(x),200)),':','color',color(i,:))
            end
            if tlinea(i)==3
                plot(linspace(min(x),max(x),200),polyval(pol_y,
                   linspace(min(x),max(x),200)),'-.','color',color(i,:))
            end
        end
    end%%_____________________________________________ Fin Gráfica 2
end
if
strcmp(get(handles.semejanza,'fontweight'),'bold')&&get(handles.popupmenu
57,'value')~=1 %======================= Curvas por semejanza
    cla(handles.axes1,'reset'),axes(handles.axes1);ylabel(Upresion);
    guidata(hObject,handles);egrid=handles.egrid;feval(@grid,egrid)
    hold on;
    simboloe=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
    if get(handles.popupmenu55,'value')==8,
        colore=handles.semejanzacolor1;
    %_____________________________________Curva de Referencia Superior
    else colore=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
        colore=colore(get(handles.popupmenu55,'value'),:);
    end
        i=get(handles.popupmenu57,'value')-1;
    if i~=0
        q=datos(datos(:,1)==RPMo(i),5);x=q*qx;qref=q*qx;
        h=(datos(datos(:,1)==RPMo(i),3)-datos(datos(:,1)...
           ==RPMo(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)-...
           4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
        y=h*px;href=h*px;
```

```
        wu=g*rho*(1/60000)*q.*h*wx;

        if
~strcmp(char(simboloe(get(handles.popupmenu54,'value'))),'N')&&~isempty(d
atos(datos(:,1)==RPMo(i),1))

plot(x,y,char(simboloe(get(handles.popupmenu54,'value'))),'color',colore)
        end
        if length(x)>3&&~isempty(datos(datos(:,1)==RPMo(i),1)),
            pol_y=polyfit(x,y,handles.ordenh);
            if (get(handles.popupmenu56,'value')-1)==0
                plot(linspace(min(x),max(x),200),polyval(pol_y,...
                   linspace(min(x),max(x),200)),'color',colore)
            end
            if (get(handles.popupmenu56,'value')-1)==1
               plot(linspace(min(x),max(x),200),polyval(pol_y,linspace...
                  (min(x),max(x),200)),'--','color',colore)
            end
            if (get(handles.popupmenu56,'value')-1)==2
                plot(linspace(min(x),max(x),200),polyval(pol_y,...
                   linspace(min(x),max(x),200)),':','color',colore)
            end
            if (get(handles.popupmenu56,'value')-1)==3
                plot(linspace(min(x),max(x),200),polyval(pol_y,...
                   linspace(min(x),max(x),200)),'-.','color',colore)
            end
        end
    end
    if get(handles.radiobutton6,'value')
        if get(handles.popupmenu59,'value')==8,
           colore=handles.semejanzacolor2;
%_____________________________________Curva por Semejanza 1 Superior
        else colore=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
             colore=colore(get(handles.popupmenu59,'value'),:);
        end
        if
str2double(get(handles.edit3,'string'))~=0&&i~=0&&str2double(get(handles.
text17,'string'))~=0
            n1=str2double(get(handles.edit3,'string'));
            D1=str2double(get(handles.text17,'string'))/1000;
            x=q*((n1/RPMo(i)))*((D1/Dr)^3)*qx;
            y=h*((n1/RPMo(i))^2)*((D1/Dr)^2)*px;
            if
~strcmp(char(simboloe(get(handles.popupmenu61,'value'))),'N')
                plot(x,y,char(simboloe(get(handles.popupmenu61,...
                  'value'))),'color',colore)
            end
            if length(x)>3,pol_y=polyfit(x,y,handles.ordenh);
                if (get(handles.popupmenu60,'value')-1)==0
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                       linspace(min(x),max(x),200)),'color',colore)
                end
                if (get(handles.popupmenu60,'value')-1)==1
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'--','color',colore)
                end
                if (get(handles.popupmenu60,'value')-1)==2
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),':','color',colore)
                end
                if (get(handles.popupmenu60,'value')-1)==3
```

```
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'-.','color',colore)
                end
            end
        end

        if get(handles.popupmenu62,'value')==8,
           colore=handles.semejanzacolor3;
%_____________________________________Curva por Semejanza 2 Superior
        else colore=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
             colore=colore(get(handles.popupmenu62,'value'),:);
        end
        if
str2double(get(handles.edit5,'string'))~=0&&i~=0&&str2double(get(handles.
edit4,'string'))~=0
            n1=str2double(get(handles.edit5,'string'));
            D1=str2double(get(handles.edit4,'string'))/1000;
            x=q*((n1/RPMo(i)))*((D1/Dr)^3)*qx;
            y=h*((n1/RPMo(i))^2)*((D1/Dr)^2)*px;
            if
~strcmp(char(simboloe(get(handles.popupmenu64,'value'))),'N')
                plot(x,y,char(simboloe(get(handles.popupmenu64,...
                  'value'))),'color',colore)
            end
            if length(x)>3,pol_y=polyfit(x,y,handles.ordenh);
                if (get(handles.popupmenu63,'value')-1)==0
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'color',colore)
                end
                if (get(handles.popupmenu63,'value')-1)==1
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'--','color',colore)
                end
                if (get(handles.popupmenu63,'value')-1)==2
                    plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                      linspace(min(x),max(x),200)),':','color',colore)
                end
                if (get(handles.popupmenu63,'value')-1)==3
                    plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                      linspace(min(x),max(x),200)),'-.','color',colore)
                end
            end
        end
        hold off
    else
        n1=RPMo(get(handles.popupmenu57,'value')-1);
        %RPM=RPM(RPM~=0);
        for i=1:1:length(RPM)%%________ Inicio Gráficas comparativas H_m
            clear q h x y
            q=datos(datos(:,1)==RPM(i),5);
            h=(datos(datos(:,1)==RPM(i),3)-datos(datos(:,1)==...
                RPM(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)-...
                4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
            x=q*qx;y=h*px;
            if
~strcmp(char(simbolo(i)),'N')&&~isempty(datos(datos(:,1)==RPM(i),1))
                plot(x,y,char(simbolo(i)),'color',color(i,:))
            end
            x=qref*(RPM(i)/n1);y=href*((RPM(i)/n1)^2);
            if length(x(x~=0))>3,pol_y=polyfit(x,y,handles.ordenh);
                if tlinea(i)==0
```

```
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'color',color(i,:))
                end
                if tlinea(i)==1
                    plot(linspace(min(x),max(x),200),...
                      polyval(pol_y,linspace(min(x),max(x),200)...
                      ),'--','color',color(i,:))
                end
                if tlinea(i)==2
                    plot(linspace(min(x),max(x),200),...
                      polyval(pol_y,linspace(min(x),max(x),...
                      200)),':','color',color(i,:))
                end
                if tlinea(i)==3

plot(linspace(min(x),max(x),200),polyval(pol_y,linspace(min(x),max(x),200
)),'-.','color',color(i,:))
                end
            end

end%%_____________________________________________________________________
___________ FIN Gráficas comparativas H_m
    end

    cla(handles.axes2,'reset'),axes(handles.axes2);
    guidata(hObject,handles);egrid=handles.egrid;feval(@grid,egrid)
    hold on, xlabel(['Q (',Ucaudal,')']);ylabel(handles.Upotencia);
    if
get(handles.popupmenu55,'value')==8,colore=handles.semejanzacolor1;%_____
_____________________________Curva de Referencia Inferior
    else
colore=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];colore=colore(get(hand
les.popupmenu55,'value'),:);
    end

    i=get(handles.popupmenu57,'value')-1;
    if i~=0
        q=datos(datos(:,1)==RPMo(i),5);
        h=(datos(datos(:,1)==RPMo(i),3)-
datos(datos(:,1)==RPMo(i),2))/(g*rho*10^-5)...
            +dz+(((4/(pi*Di^2)-4/(pi*Da^2))*((1/60000)*q)).^2)/(2*g);
        wu=g*rho*(1/60000)*q.*h*wx;wuref=wu;
        x=q*qx;y=wu;
        if
~strcmp(char(simboloe(get(handles.popupmenu54,'value'))),'N')&&~isempty(d
atos(datos(:,1)==RPMo(i),1))
            plot(x,y,char(simboloe(get(...
              handles.popupmenu54,'value'))),'color',colore)
        end
        if length(x)>3&&~isempty(datos(datos(:,1)==RPMo(i),1)),
            pol_y=polyfit(x,y,handles.ordenh);
            if (get(handles.popupmenu56,'value')-1)==0
                plot(linspace(min(x),max(x),200),polyval(pol_y,...
                  linspace(min(x),max(x),200)),'color',colore)
            end
            if (get(handles.popupmenu56,'value')-1)==1
                plot(linspace(min(x),max(x),...
                  200),polyval(pol_y,linspace(min(x),...
                  max(x),200)),'--','color',colore)
            end
            if (get(handles.popupmenu56,'value')-1)==2
```

```
                plot(linspace(min(x),max(x),200),polyval...
                  (pol_y,linspace(min(x),max(x),200)),':','color',colore)
            end
            if (get(handles.popupmenu56,'value')-1)==3
                plot(linspace(min(x),max(x), ...
                  200),polyval(pol_y,linspace(min...
                  (x),max(x),200)),'-.','color',colore)
            end
        end
    end

    if get(handles.radiobutton6,'value')
        if get(handles.popupmenu59,'value')==8,
           colore=handles.semejanzacolor2;
%_____________________________________Curva por Semejanza 1 Inferior
        else colore=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
             colore=colore(get(handles.popupmenu59,'value'),:);
        end

        if
str2double(get(handles.edit3,'string'))~=0&&i~=0&&str2double(get(handles.
text17,'string'))~=0
            n1=str2double(get(handles.edit3,'string'));
            D1=str2double(get(handles.text17,'string'))/1000;
            x=q*((n1/RPMo(i)))*((D1/Dr)^3)*qx;
            y=wu*((n1/RPMo(i))^3)*((D1/Dr)^5);
            if
~strcmp(char(simboloe(get(handles.popupmenu61,'value'))),'N')
                plot(x,y,char(simboloe(get(handles.popupmenu61...
                  ,'value'))),'color',colore)
            end
            if length(x)>3,pol_y=polyfit(x,y,handles.ordenh);
                if (get(handles.popupmenu60,'value')-1)==0
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                       linspace(min(x),max(x),200)),'color',colore)
                end
                if (get(handles.popupmenu60,'value')-1)==1
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'--','color',colore)
                end
                if (get(handles.popupmenu60,'value')-1)==2
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),':','color',colore)
                end
                if (get(handles.popupmenu60,'value')-1)==3
                    plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                      linspace(min(x),max(x),200)),'-.','color',colore)
                end
            end
        end

        if get(handles.popupmenu62,'value')==8,
            colore=handles.semejanzacolor3;
 %_____________________________________Curva por Semejanza 2 Inferior
        else colore=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
            colore=colore(get(handles.popupmenu62,'value'),:);
        end
        if
str2double(get(handles.edit5,'string'))~=0&&i~=0&&str2double(get(handles.
edit4,'string'))~=0
            n1=str2double(get(handles.edit5,'string'));
```

```
            D1=str2double(get(handles.edit4,'string'))/1000;
            x=q*((n1/RPMo(i)))*((D1/Dr)^3)*qx;
            y=wu*((n1/RPMo(i))^3)*((D1/Dr)^5);
            if
~strcmp(char(simboloe(get(handles.popupmenu64,'value'))),'N')
                plot(x,y,char(simboloe(get(handles.popupmenu64,...
                  'value'))),'color',colore)
            end
            if length(x)>3,pol_y=polyfit(x,y,handles.ordenh);
                if (get(handles.popupmenu63,'value')-1)==0
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'color',colore)
                end
                if (get(handles.popupmenu63,'value')-1)==1
                    plot(linspace(min(x),max(x),200),polyval(pol_y,...
                      linspace(min(x),max(x),200)),'--','color',colore)
                end
                if (get(handles.popupmenu63,'value')-1)==2
                    plot(linspace(min(x),max(x),200),polyval(pol_y, ...
                      linspace(min(x),max(x),200)),':','color',colore)
                end
                if (get(handles.popupmenu63,'value')-1)==3
                    plot(linspace(min(x),max(x),200),...
                      polyval(pol_y,linspace(min(x),max(x),...
                      200)),'-.','color',colore)
                end
            end
        end
        hold off
    else
        n1=RPMo(get(handles.popupmenu57,'value')-1);
        %RPM=RPM(RPM~=0);
        for i=1:1:length(RPM)
%%_____________________________________ Inicio Gráficas comparativas W_u
            clear q h x y
            q=datos(datos(:,1)==RPM(i),5);
            h=(datos(datos(:,1)==RPM(i),3)-datos(datos(:,1)==...
               RPM(i),2))/(g*rho*10^-5)+dz+(((4/(pi*Di^2)-4/...
               (pi*Da^2))*((1/60000)*q)).^2)/(2*g);
            wu=g*rho*(1/60000)*q.*h*wx;
            x=q*qx;y=wu;
            if
~strcmp(char(simbolo(i)),'N')&&~isempty(datos(datos(:,1)==RPM(i),1))
                plot(x,y,char(simbolo(i)),'color',color(i,:))
            end
            x=qref*(RPM(i)/n1);y=wuref*((RPM(i)/n1)^3);
            if length(x(x~=0))>3,pol_y=polyfit(x,y,handles.ordenh);
                if tlinea(i)==0
                    plot(linspace(min(x),max(x),200),...
                      polyval(pol_y,linspace(min(x),max(x),...
                      200)),'color',color(i,:))
                end
                if tlinea(i)==1
                    plot(linspace(min(x),max(x),200),...
                      polyval(pol_y,linspace(min(x),max(x),...
                      200)),'--','color',color(i,:))
                end
                if tlinea(i)==2
                    plot(linspace(min(x),max(x),200),...
                      polyval(pol_y,linspace(min(x),max(x),...
                      200)),':','color',color(i,:))
```

```
                end
                if tlinea(i)==3
                    plot(linspace(min(x),max(x),...
                      200),polyval(pol_y,linspace...
                      (min(x),max(x),200)),'-.','color',color(i,:))
                end
            end
        end%%                           FIN Gráficas comparativas W u
    end
end

% --------------------------------------------------------------------
function presion_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function MCfluido Callback(hObject, eventdata, handles)
Upresion='H_m (m)';handles.Upresion=Upresion;guidata(hObject,handles);
set(handles.Piesfluido,'Checked','off');set(handles.bar,'Checked','off');
set(handles.Librapulgada,'Checked','off');
set(handles.Pascal,'Checked','off');set(handles.MCfluido,'Checked','on');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function Piesfluido_Callback(hObject, eventdata, handles)
Upresion='H_m (ft)';handles.Upresion=Upresion;guidata(hObject,handles);
set(handles.Piesfluido,'Checked','on');set(handles.bar,'Checked','off');
set(handles.Librapulgada,'Checked','off');
set(handles.Pascal,'Checked','off');set(handles.MCfluido,'Checked','off')
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function bar_Callback(hObject, eventdata, handles)
Upresion='P (Bar)';handles.Upresion=Upresion;guidata(hObject,handles);
set(handles.Piesfluido,'Checked','off');set(handles.bar,'Checked','on');
set(handles.Librapulgada,'Checked','off');
set(handles.Pascal,'Checked','off');set(handles.MCfluido,'Checked','off')
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function Librapulgada_Callback(hObject, eventdata, handles)
Upresion='P (psi)';handles.Upresion=Upresion;guidata(hObject,handles);
set(handles.Piesfluido,'Checked','off');set(handles.bar,'Checked','off');
set(handles.Librapulgada,'Checked','on');
set(handles.Pascal,'Checked','off');set(handles.MCfluido,'Checked','off')
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function gpmi_Callback(hObject, eventdata, handles)
Ucaudal='gpm U.K.';handles.Ucaudal=Ucaudal;guidata(hObject,handles);
set(handles.U_caudal_ls,'Checked','off');
set(handles.U_caudal_m3h,'Checked','off');
set(handles.U_caudal_lm,'Checked','off');
set(handles.gpmusa,'Checked','off'); set(handles.gpmi,'Checked','on');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function gpmusa_Callback(hObject, eventdata, handles)
Ucaudal='gpm U.S.A.';handles.Ucaudal=Ucaudal;guidata(hObject,handles);
set(handles.U_caudal_ls,'Checked','off');
set(handles.U_caudal_m3h,'Checked','off');
```

```
set(handles.U_caudal_lm,'Checked','off');
set(handles.gpmusa,'Checked','on');set(handles.gpmi,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function Pascal_Callback(hObject, eventdata, handles)
Upresion='P (Pascal)';handles.Upresion=Upresion;guidata(hObject,handles);
set(handles.Piesfluido,'Checked','off');set(handles.bar,'Checked','off');
set(handles.Librapulgada,'Checked','off');
set(handles.Pascal,'Checked','on');set(handles.MCfluido,'Checked','off');
representar_Callback(hObject,0,handles);

% --- Executes when selected object is changed in uipanel3.
function uipanel3_SelectionChangeFcn(hObject, eventdata, handles)
if get(handles.uno,'value')
    handles.RPM=handles.RPMo;
    RPM=handles.RPMo;
    set(handles.uipanel4,'visible','off')
    simbolo=zeros(1,length(RPM));
    for i=1:1:length(RPM),simbolo(i)='.';end
    handles.simbolo=simbolo; handles.tlinea=zeros(1,length(RPM));
    color=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
    for i=8:1:length(RPM),color(i,:)=rand(1,3);end
    handles.color=color;
else
    handles.RPM=0;
    RPMo=handles.RPMo;
    b='0';
    for i=1:1:length(RPMo),b=[b,'|',num2str(RPMo(i))];end
    set(handles.popupmenu3,'String', b);
    set(handles.popupmenu4,'String', b);
    set(handles.popupmenu5,'String', b);
    set(handles.popupmenu6,'String', b);
    set(handles.popupmenu7,'String', b);
    set(handles.popupmenu8,'String', b);
    set(handles.popupmenu9,'String', b);
    set(handles.popupmenu10,'String', b);
    set(handles.popupmenu11,'String', b);
    set(handles.popupmenu12,'String', b);
    set(handles.uipanel4,'visible','on')
    RPM=handles.RPM;RPMo=handles.RPMo;

    if get(handles.popupmenu3,'value')==1,RPM(1)=0;
    else RPM(1)=RPMo(get(handles.popupmenu3,'value')-1);end
    if get(handles.popupmenu4,'value')==1,RPM(2)=0;
    else RPM(2)=RPMo(get(handles.popupmenu4,'value')-1);end
    if get(handles.popupmenu5,'value')==1,RPM(3)=0;
    else RPM(3)=RPMo(get(handles.popupmenu5,'value')-1);end
    if get(handles.popupmenu6,'value')==1,RPM(4)=0;
    else RPM(4)=RPMo(get(handles.popupmenu6,'value')-1);end
    if get(handles.popupmenu7,'value')==1,RPM(5)=0;
    else RPM(5)=RPMo(get(handles.popupmenu7,'value')-1);end
    if get(handles.popupmenu8,'value')==1,RPM(6)=0;
    else RPM(6)=RPMo(get(handles.popupmenu8,'value')-1);end
    if get(handles.popupmenu9,'value')==1,RPM(7)=0;
    else RPM(7)=RPMo(get(handles.popupmenu9,'value')-1);end
    if get(handles.popupmenu10,'value')==1,RPM(8)=0;
    else RPM(8)=RPMo(get(handles.popupmenu10,'value')-1);end
    if get(handles.popupmenu11,'value')==1,RPM(9)=0;
    else RPM(9)=RPMo(get(handles.popupmenu11,'value')-1);end
    if get(handles.popupmenu12,'value')==1,RPM(10)=0;
```

```
    else RPM(10)=RPMo(get(handles.popupmenu12,'value')-1);end
    if isempty(get(handles.edit1,'string'));RPM(11)=0;
    else RPM(11)=str2double(get(handles.edit1,'string'));end
    handles.RPM=RPM;

    a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
    simbolo=handles.simbolo;
    simbolo(1)=a(get(handles.popupmenu13,'Value'));
    simbolo(2)=a(get(handles.popupmenu14,'Value'));
    simbolo(3)=a(get(handles.popupmenu15,'Value'));
    simbolo(4)=a(get(handles.popupmenu16,'Value'));
    simbolo(5)=a(get(handles.popupmenu17,'Value'));
    simbolo(6)=a(get(handles.popupmenu18,'Value'));
    simbolo(7)=a(get(handles.popupmenu19,'Value'));
    simbolo(8)=a(get(handles.popupmenu20,'Value'));
    simbolo(9)=a(get(handles.popupmenu21,'Value'));
    simbolo(10)=a(get(handles.popupmenu22,'Value'));
    simbolo(11)=a(get(handles.popupmenu23,'Value'));

    k=[0,1,2,3,4];tlinea=handles.tlinea;
    tlinea(1)=k(get(handles.popupmenu35,'Value'));
    tlinea(2)=k(get(handles.popupmenu36,'Value'));
    tlinea(3)=k(get(handles.popupmenu37,'Value'));
    tlinea(4)=k(get(handles.popupmenu38,'Value'));
    tlinea(5)=k(get(handles.popupmenu39,'Value'));
    tlinea(6)=k(get(handles.popupmenu40,'Value'));
    tlinea(7)=k(get(handles.popupmenu41,'Value'));
    tlinea(8)=k(get(handles.popupmenu42,'Value'));
    tlinea(9)=k(get(handles.popupmenu43,'Value'));
    tlinea(10)=k(get(handles.popupmenu44,'Value'));
    tlinea(11)=k(get(handles.popupmenu45,'Value'));
    handles.simbolo=simbolo;handles.tlinea=tlinea;
    handles.color=handles.color0;
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(1)=0;
else
    RPM(1)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

function popupmenu3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(2)=0;
else
    RPM(2)=RPMo(get(hObject,'value')-1);
```

```
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu5.
function popupmenu5_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(3)=0;
else
    RPM(3)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu6.
function popupmenu6_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(4)=0;
else
    RPM(4)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu7.
function popupmenu7_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(5)=0;
else
    RPM(5)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu7_CreateFcn(hObject, eventdata, handles)
```

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

% --- Executes on selection change in popupmenu8.
function popupmenu8_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(6)=0;
else
    RPM(6)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu8_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu9.
function popupmenu9_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(7)=0;
else
    RPM(7)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu9_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu10.
function popupmenu10_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(8)=0;
else
    RPM(8)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu10_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu11.
```

```
function popupmenu11_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(9)=0;
else
    RPM(9)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu11_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu12.
function popupmenu12_Callback(hObject, eventdata, handles)
RPM=handles.RPM;
RPMo=handles.RPMo;
if get(hObject,'value')==1,RPM(10)=0;
else
    RPM(10)=RPMo(get(hObject,'value')-1);
end
handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu12_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit1_Callback(hObject, eventdata, handles)
RPM=handles.RPM;RPM11=get(hObject,'string');
if isempty(RPM11);RPM11=0;end
RPM11=str2double(RPM11);
if isnan(RPM11);set(hObject,'string','');RPM11=0;
waitfor(errordlg('El valor debe ser un valor numérico','ERROR'));
end
if RPM11<500&&RPM11~=0;set(hObject,'string','');RPM11=0;
waitfor(errordlg('El valor debe ser mayor que 500 o igual a 0','ERROR'));
end
if RPM11>6000;set(hObject,'string','');RPM11=0;
    waitfor(errordlg('El valor debe ser menor que 6000','ERROR'));
end
RPM(11)=RPM11;handles.RPM=RPM;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes when uipanel4 is resized.
function uipanel4_ResizeFcn(hObject, eventdata, handles) %#ok<DEFNU>
```

```
% --- Executes during object deletion, before destroying properties.
function uipanel4_DeleteFcn(hObject, eventdata, handles)

% --- Executes on selection change in popupmenu13.
function popupmenu13_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(1)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu13_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu14.
function popupmenu14_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(2)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu14_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu15.
function popupmenu15_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(3)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu15_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu16.
function popupmenu16_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(4)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu16_CreateFcn(hObject, eventdata, handles)
```

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

% --- Executes on selection change in popupmenu17.
function popupmenu17_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(5)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu17_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu18.
function popupmenu18_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(6)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu18_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu19.
function popupmenu19_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(7)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu19_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu20.
function popupmenu20_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(8)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu20_CreateFcn(hObject, eventdata, handles)
```

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

% --- Executes on selection change in popupmenu21.
function popupmenu21_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(9)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu21_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu22.
function popupmenu22_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(10)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu22_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu23.
function popupmenu23_Callback(hObject, eventdata, handles)
a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(11)=a(get(hObject,'Value'));
handles.simbolo=simbolo;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu23_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu24.
function popupmenu24_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(1,:)=a(get(hObject,'Value'),:);
else color(1,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu24_CreateFcn(hObject, eventdata, handles)
```

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

% --- Executes on selection change in popupmenu25.
function popupmenu25_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(2,:)=a(get(hObject,'Value'),:);
else color(2,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu25_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu26.
function popupmenu26_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(3,:)=a(get(hObject,'Value'),:);
else color(3,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu26_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu27.
function popupmenu27_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(4,:)=a(get(hObject,'Value'),:);
else color(4,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu27_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu28.
function popupmenu28_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(5,:)=a(get(hObject,'Value'),:);
else color(5,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu28_CreateFcn(hObject, eventdata, handles)
```

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

% --- Executes on selection change in popupmenu29.
function popupmenu29_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(6,:)=a(get(hObject,'Value'),:);
else color(6,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu29_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu30.
function popupmenu30_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(7,:)=a(get(hObject,'Value'),:);
else color(7,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu30_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu31.
function popupmenu31_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(8,:)=a(get(hObject,'Value'),:);
else color(8,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu31_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu32.
function popupmenu32_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(9,:)=a(get(hObject,'Value'),:);
else color(9,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu32_CreateFcn(hObject, eventdata, handles)
```

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

% --- Executes on selection change in popupmenu33.
function popupmenu33_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(10,:)=a(get(hObject,'Value'),:);
else color(10,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu33_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu34.
function popupmenu34_Callback(hObject, eventdata, handles)
a=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];color=handles.color;
if get(hObject,'Value')<8,color(11,:)=a(get(hObject,'Value'),:);
else color(11,:)=uisetcolor ([0,0,1], 'Seleccione un color');end
handles.color=color;handles.color0=color; guidata(hObject,handles);
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu34_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function uipushtool7_ClickedCallback(hObject, eventdata, handles)%
Imprimir
printpreview

% --------------------------------------------------------------------
function uipushtool6_ClickedCallback(hObject, eventdata, handles)% Icono
Abrir
imp_cc_0_Callback(hObject,0,handles)% Importa datos


% --- Executes on selection change in popupmenu35.
function popupmenu35_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(1)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu35_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in popupmenu36.
function popupmenu36_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(2)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu36_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu37.
function popupmenu37_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(3)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu37_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu38.
function popupmenu38_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(4)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu38_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu39.
function popupmenu39_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(5)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu39_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu40.
function popupmenu40_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
```

```
tlinea(6)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu40_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu41.
function popupmenu41_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(7)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu41_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu42.
function popupmenu42_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(8)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu42_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu43.
function popupmenu43_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(9)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu43_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu44.
function popupmenu44 Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(10)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu44_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu45.
function popupmenu45_Callback(hObject, eventdata, handles)
a=[0,1,2,3,4];tlinea=handles.tlinea;
tlinea(11)=a(get(hObject,'Value'));
handles.tlinea=tlinea;
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu45 CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu1.
function popupmenu47_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function popupmenu47_CreateFcn(hObject, eventdata, handles)

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

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% ------------------------------------------------------------------
function grafica_Callback(hObject, eventdata, handles)

% ------------------------------------------------------------------
function grid_Callback(hObject, eventdata, handles)

% ------------------------------------------------------------------
function gridoff_Callback(hObject, eventdata, handles)
handles.egrid='off';guidata(hObject,handles);
axes(handles.axes1);grid off
axes(handles.axes2);grid off
set(handles.gridon,'Checked','off');
set(handles.gridminor,'Checked','off');
set(handles.gridoff,'Checked','on');
if get(handles.popupmenu1,'value')~=4&&get(handles.popupmenu2,'value')~=1
    representar_Callback(hObject,0,handles);
```

```
end

% --------------------------------------------------------------------
function gridon_Callback(hObject, eventdata, handles)
handles.egrid='on';guidata(hObject,handles);
axes(handles.axes1);grid off,grid on
axes(handles.axes2);grid off,grid on
set(handles.gridoff,'Checked','off');
set(handles.gridminor,'Checked','off');
set(handles.gridon,'Checked','on');
if get(handles.popupmenu1,'value')~=4&&get(handles.popupmenu2,'value')~=1
    representar_Callback(hObject,0,handles);
end

% --------------------------------------------------------------------
function gridminor Callback(hObject, eventdata, handles)
handles.egrid='minor';guidata(hObject,handles);
axes(handles.axes1);grid minor
axes(handles.axes2);grid minor
set(handles.gridon,'Checked','off');set(handles.gridoff,'Checked','off');
set(handles.gridminor,'Checked','on');
if get(handles.popupmenu1,'value')~=4&&get(handles.popupmenu2,'value')~=1
    representar_Callback(hObject,0,handles);
end

% --------------------------------------------------------------------
function actualizacolor_Callback(hObject, eventdata, handles)
if get(handles.uno,'value')==1
    RPM=handles.RPM; color=[0,0,1;1,1,0;1,0,1;0,1,1;1,0,0;0,1,0;0,0,0];
    for i=8:1:length(RPM),color(i,:)=rand(1,3);end
    handles.color=color;guidata(hObject,handles);
representar_Callback(hObject,0,handles);
end

% --------------------------------------------------------------------
function cisorendimiento_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function estadoisorendimiento_Callback(hObject, eventdata, handles)
if strcmp(get(gcbo, 'Checked'),'on')
    set(gcbo, 'Checked', 'off');
    set(handles.estadoisorendimiento,'label','Contornos de Isorendimiento
(Mostrar)')
else
    set(gcbo, 'Checked', 'on');
    set(handles.estadoisorendimiento,'label','Contornos de Isorendimiento
(Ocultar)')
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function colorisorendimiento_Callback(hObject, eventdata, handles)
if strcmp(get(gcbo, 'Checked'),'on');
    set(gcbo, 'Checked', 'off');
else
    set(gcbo, 'Checked', 'on');
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
```

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

% --------------------------------------------------------------------
function noetiqisorendimiento_Callback(hObject, eventdata, handles)
set(handles.noetiqisorendimiento,'Checked','on');
set(handles.Manuetiqisorendimiento,'Checked','off');
set(handles.autoetiqisorendimiento,'Checked','off');
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function autoetiqisorendimiento_Callback(hObject, eventdata, handles)
set(handles.noetiqisorendimiento,'Checked','off');
set(handles.Manuetiqisorendimiento,'Checked','off');
set(handles.autoetiqisorendimiento,'Checked','on');
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function Manuetiqisorendimiento_Callback(hObject, eventdata, handles)
set(handles.noetiqisorendimiento,'Checked','off');
set(handles.Manuetiqisorendimiento,'Checked','on');
set(handles.autoetiqisorendimiento,'Checked','off');
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mapcolor_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function mcpordefecto_Callback(hObject, eventdata, handles)
handles.mapacolor='default';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','on');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');
set(handles.mcspring,'Checked','off');
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');
set(handles.mcgray,'Checked','off');
set(handles.mcbone,'Checked','off');
set(handles.mccopper,'Checked','off');
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mchot_Callback(hObject, eventdata, handles)
handles.mapacolor='hot';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','on');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');
set(handles.mcspring,'Checked','off');
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);
```

```
% --------------------------------------------------------------------
function mcjet_Callback(hObject, eventdata, handles)
handles.mapacolor='jet';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','on');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mchsv_Callback(hObject, eventdata, handles)
handles.mapacolor='hsv';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','on');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mccool_Callback(hObject, eventdata, handles)
handles.mapacolor='cool';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','on');set(handles.mcspring,'Checked','off');
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mcspring_Callback(hObject, eventdata, handles)
handles.mapacolor='spring';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','on');
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
```

```
function mcsummer_Callback(hObject, eventdata, handles)
handles.mapacolor='summer';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','on');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mcautumn_Callback(hObject, eventdata, handles)
handles.mapacolor='autumn';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','on');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mcwinter_Callback(hObject, eventdata, handles)
handles.mapacolor='winter';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','on');set(handles.mcgray,'Checked','off');
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mcgray_Callback(hObject, eventdata, handles)
handles.mapacolor='gray';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','on');
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mcbone_Callback(hObject, eventdata, handles)
```

```
handles.mapacolor='bone';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','on');set(handles.mccopper,'Checked','off');
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mccopper_Callback(hObject, eventdata, handles)
handles.mapacolor='copper';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','on');
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mcpink_Callback(hObject, eventdata, handles)
handles.mapacolor='pink';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','on');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mclines_Callback(hObject, eventdata, handles)
handles.mapacolor='lines';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','on');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mccolorcube_Callback(hObject, eventdata, handles)
handles.mapacolor='colorcube';guidata(hObject,handles);
```

```
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','off');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','on');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function mcflag_Callback(hObject, eventdata, handles)
handles.mapacolor='flag';guidata(hObject,handles);
set(handles.mcpordefecto,'Checked','off');
set(handles.mchot,'Checked','off');
set(handles.mcflag,'Checked','on');set(handles.mchsv,'Checked','off');
set(handles.mccool,'Checked','off');set(handles.mcspring,'Checked','off')
set(handles.mcsummer,'Checked','off');
set(handles.mcautumn,'Checked','off');
set(handles.mcwinter,'Checked','off');set(handles.mcgray,'Checked','off')
set(handles.mcbone,'Checked','off');set(handles.mccopper,'Checked','off')
set(handles.mcpink,'Checked','off');set(handles.mclines,'Checked','off');
set(handles.mccolorcube,'Checked','off');
representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function nivelisorendimiento_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function autonumeroisolineas_Callback(hObject, eventdata, handles)
set(handles.autonumeroisolineas,'Checked','on');set(handles.valornumisoli
neas,'Checked','off');
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function valornumisolineas_Callback(hObject, eventdata, handles)
answer=inputdlg('Introduzca un valor para el número de niveles de
isorendimiento:’,'Número de niveles de isorendimiento');
if isempty(answer);return,end
answer=str2double(answer);
if isnan(answer); errordlg('El valor debe ser un valor
numérico','ERROR'); return,end
if answer<2; errordlg('El valor debe ser mayor que 1','ERROR');
return,end
if answer>50; errordlg('El valor debe ser menor de 50','ERROR');
return,end
if ~not(mod(answer,1)); errordlg('El valor debe ser un número
entero','ERROR'); return,end
handles.numeroisolineas=answer;guidata(hObject,handles);
set(handles.autonumeroisolineas,'Checked','off');
set(handles.valornumisolineas,'Checked','on');
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function ordenajuste_Callback(hObject, eventdata, handles)
prompt={'Orden de ajuste para H_m(q)','Orden de ajuste para Wu(q):',...
    'Orden de ajuste para Wb(q):','Orden de ajuste para N(q):',...
    'Orden de ajuste para T(q):'};
```

```
name='Orden de ajsuste'; numlines=1;
defaultanswer{1}=num2str(handles.ordenh);
defaultanswer{2}=num2str(handles.ordenwu);
defaultanswer{3}=num2str(handles.ordenwb);
defaultanswer{4}=num2str(handles.ordenN);
defaultanswer{5}=num2str(handles.ordenT);
options.Resize='off';
options.WindowStyle='normal';options.Interpreter='tex';
[answer]=inputdlg(prompt,name,numlines,defaultanswer,options);
if isempty(answer);return,end
answer=str2double(answer);
if
isnan(answer(1))||isnan(answer(2))||isnan(answer(3))||isnan(answer(4))||i
snan(answer(5))
    errordlg('Todos los valores tienen que ser numéricos','ERROR');
return
end
if answer(1)<1||answer(2)<1||answer(3)<1||answer(4)<1||answer(5)<1
    errordlg('Ningún valor puede ser menor a 1','ERROR'); return
end
if answer(1)>10||answer(2)>10||answer(3)>10||answer(4)>10||answer(5)>=10
    errordlg('Nigún valor puede ser superior a 10','ERROR'); return
end
if
~not(mod(answer(1),1))||~not(mod(answer(2),1))||~not(mod(answer(3),1))||~
not(mod(answer(4),1))||~not(mod(answer(5),1))
    errordlg('Los valores tienen que ser números enteros','ERROR');
return
end
ordenh=answer(1);ordenwu=answer(2);ordenwb=answer(3);ordenN=answer(4);
ordenT=answer(5);handles.ordenh=ordenh; handles.ordenwu=ordenwu;
handles.ordenwb=ordenwb; handles.ordenN=ordenN;handles.ordenT=ordenT;
guidata(hObject,handles);representar_Callback(hObject,0,handles);


function uipanel5_ButtonDownFcn(hObject, eventdata, handles)

% --- Executes when selected object is changed in uipanel6.
function uipanel6_SelectionChangeFcn(hObject, eventdata, handles)
if(hObject==handles.tres)
    handles.z=0;
else
    handles.z=1;
end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes on selection change in popupmenu49.
function popupmenu49_Callback(hObject, eventdata, handles)
if
get(handles.popupmenu49,'value')==1&&get(handles.popupmenu50,'value')==2
    set(handles.popupmenu50,'value',1)
end
if
get(handles.popupmenu49,'value')==2&&get(handles.popupmenu50,'value')==1
    set(handles.popupmenu50,'value',2)
end
if
get(handles.popupmenu49,'value')==3&&get(handles.popupmenu50,'value')==3
    set(handles.popupmenu50,'value',2)
end
guidata(hObject,handles);representar_Callback(hObject,0,handles);
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu49_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu50.
function popupmenu50_Callback(hObject, eventdata, handles)
if
get(handles.popupmenu50,'value')==2&&get(handles.popupmenu49,'value')==1
    set(handles.popupmenu49,'value',2)
end
if
get(handles.popupmenu50,'value')==1&&get(handles.popupmenu49,'value')==2
    set(handles.popupmenu49,'value',1)
end
if
get(handles.popupmenu50,'value')==3&&get(handles.popupmenu49,'value')==3
    set(handles.popupmenu49,'value',1)
end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu50_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu51.
function popupmenu51_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu51_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function isoconsumo_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function estadoisoconsumo_Callback(hObject, eventdata, handles)
if strcmp(get(gcbo, 'Checked'),'on')
    set(gcbo, 'Checked', 'off');
    set(handles.estadoisoconsumo,'label','Contornos de Isoconsumo
(Mostrar)')
else
    set(gcbo, 'Checked', 'on');
    set(handles.estadoisoconsumo,'label','Contornos de Isoconsumo
(Ocultar)')
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function colorisoconsumo_Callback(hObject, eventdata, handles)
```

```
if strcmp(get(gcbo, 'Checked'),'on');
    set(gcbo, 'Checked', 'off');
else
    set(gcbo, 'Checked', 'on');
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function nivelisoconsumo_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function etiqisoconsumo_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function noetiqisoconsumo_Callback(hObject, eventdata, handles)
set(handles.noetiqisoconsumo,'Checked','on');
set(handles.Manuetiqisoconsumo,'Checked','off');
set(handles.autoetiqisoconsumo,'Checked','off');
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function autoetiqisoconsumo_Callback(hObject, eventdata, handles)
set(handles.noetiqisoconsumo,'Checked','off');
set(handles.Manuetiqisoconsumo,'Checked','off');
set(handles.autoetiqisoconsumo,'Checked','on');
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function Manuetiqisoconsumo_Callback(hObject, eventdata, handles)
set(handles.noetiqisoconsumo,'Checked','off');
set(handles.Manuetiqisoconsumo,'Checked','on');
set(handles.autoetiqisoconsumo,'Checked','off');
guidata(hObject,handles); representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function autonumisoliconsumo_Callback(hObject, eventdata, handles)
set(handles.autonumisoliconsumo,'Checked','on');
set(handles.valornumisoliconsumo,'Checked','off');
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --------------------------------------------------------------------
function valornumisoliconsumo_Callback(hObject, eventdata, handles)
answer=inputdlg('Introduzca un valor para el número de niveles de
isoconsumo:','Número de niveles de isoconsumo');
if isempty(answer);return,end
answer=str2double(answer);
if isnan(answer);
    errordlg('El valor debe ser un valor numérico','ERROR');return,
end
if answer<2; errordlg('El valor debe ser mayor que 1','ERROR');
return,end
if answer>50;
    errordlg('El valor debe ser menor de 50','ERROR'); return
end
if ~not(mod(answer,1));
     errordlg('El valor debe ser un número entero','ERROR'); return
end
handles.numeroisoconsumo=answer;guidata(hObject,handles);
set(handles.autonumisoliconsumo,'Checked','off');
set(handles.valornumisoliconsumo,'Checked','on');
```

```
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes on selection change in popupmenu52.
function popupmenu52_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu52_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu53.
function popupmenu53_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu53_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)

% --- Executes on selection change in popupmenu57.
function popupmenu57_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu57_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu54.
function popupmenu54_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu54_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu55.
function popupmenu55_Callback(hObject, eventdata, handles)
if get(hObject,'value')==8, handles.semejanzacolor1=uisetcolor
(handles.semejanzacolor1, 'Seleccione un color');end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu55_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu56.
function popupmenu56_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu56_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function uipushtool8_ClickedCallback(hObject, eventdata, handles)
guardargrafica_Callback(hObject,0,handles)

% --------------------------------------------------------------------
function uipushtool10_ClickedCallback(hObject, eventdata, handles)
verdatoscurvac_Callback(hObject,0,handles)

% --- Executes on selection change in popupmenu59.
function popupmenu59_Callback(hObject, eventdata, handles)
if get(hObject,'value')==8, handles.semejanzacolor2=uisetcolor
(handles.semejanzacolor2, 'Seleccione un color');end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu59_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu60.
function popupmenu60_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu60_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu61.
function popupmenu61_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu61_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
if isempty(get(hObject,'string'));set(hObject,'string','');end
```

```
a=str2double(get(hObject,'string'));
if isnan(a);set(hObject,'string','');
    waitfor(errordlg('El valor debe ser un valor numérico','ERROR'));
end
if a<500&&a~=0;set(hObject,'string','');
waitfor(errordlg('El valor debe ser mayor que 500 o igual a 0','ERROR'));
end
if a>6000;set(hObject,'string','');
    waitfor(errordlg('El valor debe ser menor que 6000','ERROR'));
end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function text17_Callback(hObject, eventdata, handles)
if isempty(get(hObject,'string'));set(hObject,'string','');end
a=str2double(get(hObject,'string'));
if isnan(a);set(hObject,'string','');
    waitfor(errordlg('El valor debe ser un valor numérico','ERROR'));
end
if a<0;set(hObject,'string','');
    waitfor(errordlg('El valor debe ser mayor o igual a 0','ERROR'));
end
if a>10000;set(hObject,'string','');
    waitfor(errordlg('El valor debe ser menor que 10000','ERROR'));
end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

function edit4_Callback(hObject, eventdata, handles)
if isempty(get(hObject,'string'));set(hObject,'string','');end
a=str2double(get(hObject,'string'));
if isnan(a);set(hObject,'string','');
    waitfor(errordlg('El valor debe ser un valor numérico','ERROR'));
end
if a<0;set(hObject,'string','');
    waitfor(errordlg('El valor debe ser mayor o igual a 0','ERROR'));
end
if a>10000;set(hObject,'string','');
    waitfor(errordlg('El valor debe ser menor que 10000','ERROR'));
end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu62.
function popupmenu62_Callback(hObject, eventdata, handles)
if get(hObject,'value')==8, handles.semejanzacolor3=uisetcolor
(handles.semejanzacolor3, 'Seleccione un color');end
guidata(hObject,handles);representar_Callback(hObject,0,handles);
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu62_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu63.
function popupmenu63_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu63_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu64.
function popupmenu64_Callback(hObject, eventdata, handles)
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function popupmenu64_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
if isempty(get(hObject,'string'));set(hObject,'string','');end
a=str2double(get(hObject,'string'));
if isnan(a);set(hObject,'string','');
    waitfor(errordlg('El valor debe ser un valor numérico','ERROR'));
end
if a<500&&a~=0;set(hObject,'string','');
waitfor(errordlg('El valor debe ser mayor que 500 o igual a 0','ERROR'));
end
if a>6000;set(hObject,'string','');
    waitfor(errordlg('El valor debe ser menor que 6000','ERROR'));
end
guidata(hObject,handles);representar_Callback(hObject,0,handles);

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function uipanel4_ButtonDownFcn(hObject, eventdata, handles)

% --- Executes when selected object is changed in uipanel13.
function uipanel13 SelectionChangeFcn(hObject, eventdata, handles)
if get(handles.radiobutton6,'value')
    set(handles.uipanel4,'visible','off')
    set(handles.uipanel9,'visible','on')
    set(handles.uipanel10,'visible','on')
else
```

```
set(handles.uipanel9,'visible','off')
set(handles.uipanel10,'visible','off')
set(handles.uipanel4,'visible','on')

handles.RPM=0;
RPMo=handles.RPMo;
b='0';
for i=1:1:length(RPMo),b=[b,'|',num2str(RPMo(i))];end
set(handles.popupmenu3,'String', b);
set(handles.popupmenu4,'String', b);
set(handles.popupmenu5,'String', b);
set(handles.popupmenu6,'String', b);
set(handles.popupmenu7,'String', b);
set(handles.popupmenu8,'String', b);
set(handles.popupmenu9,'String', b);
set(handles.popupmenu10,'String', b);
set(handles.popupmenu11,'String', b);
set(handles.popupmenu12,'String', b);
set(handles.uipanel4,'visible','on')
RPM=handles.RPM;
RPMo=handles.RPMo;

if get(handles.popupmenu3,'value')==1,RPM(1)=0;
else RPM(1)=RPMo(get(handles.popupmenu3,'value')-1);end
if get(handles.popupmenu4,'value')==1,RPM(2)=0;
else RPM(2)=RPMo(get(handles.popupmenu4,'value')-1);end
if get(handles.popupmenu5,'value')==1,RPM(3)=0;
else RPM(3)=RPMo(get(handles.popupmenu5,'value')-1);end
if get(handles.popupmenu6,'value')==1,RPM(4)=0;
else RPM(4)=RPMo(get(handles.popupmenu6,'value')-1);end
if get(handles.popupmenu7,'value')==1,RPM(5)=0;
else RPM(5)=RPMo(get(handles.popupmenu7,'value')-1);end
if get(handles.popupmenu8,'value')==1,RPM(6)=0;
else RPM(6)=RPMo(get(handles.popupmenu8,'value')-1);end
if get(handles.popupmenu9,'value')==1,RPM(7)=0;
else RPM(7)=RPMo(get(handles.popupmenu9,'value')-1);end
if get(handles.popupmenu10,'value')==1,RPM(8)=0;
else RPM(8)=RPMo(get(handles.popupmenu10,'value')-1);end
if get(handles.popupmenu11,'value')==1,RPM(9)=0;
else RPM(9)=RPMo(get(handles.popupmenu11,'value')-1);end
if get(handles.popupmenu12,'value')==1,RPM(10)=0;
else RPM(10)=RPMo(get(handles.popupmenu12,'value')-1);end
if isempty(get(handles.edit1,'string'));RPM(11)=0;
else RPM(11)=str2double(get(handles.edit1,'string'));end
handles.RPM=RPM;

a=['.','*','o','+','s','d','v','^','<','>','p','h','N'];
simbolo=handles.simbolo;
simbolo(1)=a(get(handles.popupmenu13,'Value'));
simbolo(2)=a(get(handles.popupmenu14,'Value'));
simbolo(3)=a(get(handles.popupmenu15,'Value'));
simbolo(4)=a(get(handles.popupmenu16,'Value'));
simbolo(5)=a(get(handles.popupmenu17,'Value'));
simbolo(6)=a(get(handles.popupmenu18,'Value'));
simbolo(7)=a(get(handles.popupmenu19,'Value'));
simbolo(8)=a(get(handles.popupmenu20,'Value'));
simbolo(9)=a(get(handles.popupmenu21,'Value'));
simbolo(10)=a(get(handles.popupmenu22,'Value'));
simbolo(11)=a(get(handles.popupmenu23,'Value'));

k=[0,1,2,3,4];tlinea=handles.tlinea;
```

```
    tlinea(1)=k(get(handles.popupmenu35,'Value'));
    tlinea(2)=k(get(handles.popupmenu36,'Value'));
    tlinea(3)=k(get(handles.popupmenu37,'Value'));
    tlinea(4)=k(get(handles.popupmenu38,'Value'));
    tlinea(5)=k(get(handles.popupmenu39,'Value'));
    tlinea(6)=k(get(handles.popupmenu40,'Value'));
    tlinea(7)=k(get(handles.popupmenu41,'Value'));
    tlinea(8)=k(get(handles.popupmenu42,'Value'));
    tlinea(9)=k(get(handles.popupmenu43,'Value'));
    tlinea(10)=k(get(handles.popupmenu44,'Value'));
    tlinea(11)=k(get(handles.popupmenu45,'Value'));
    handles.simbolo=simbolo;handles.tlinea=tlinea;
    handles.color=handles.color0;
end
guidata(hObject,handles); representar_Callback(hObject,0,handles);
```