CAPÍTULO 1: INTRODUCCIÓN

La utilización de los ordenadores ha crecido enormemente en los últimos tiempos. Cada día existen nuevos programas que facilitan en mayor o menor medida el trabajo rutinario, de simulación o de cálculo que se necesite. En el caso particular del trabajo de los Ingenieros, este incremento de la informatización se ha constatado debido al gran número de nuevos programas que salen a diario al mercado. Algunas empresas utilizan un software propio para gestionar el trabajo que desarrollan sus trabajadores, pero la mayoría optan por algún programa que existe en el mercado y que tiene las características que se puedan necesitar.

En este Proyecto Fin de Carrera se han utilizado dos paquetes o programas informáticos que resultan especialmente útiles en diferentes tareas que puede llevar a cabo un Ingeniero: Matlab y Visual Basic. La utilización del primero de ellos en tareas de formación está suficientemente probado viendo las publicaciones en revistas especializas que se pueden encontrar. Además, en muchas asignaturas de Ingeniería la realización de cálculos con un programa de computación como Matlab sustituye la programación más tradicional. También Matlab se considera en algunos sectores como uno de los lenguajes de cálculo técnico que utilizan empresas líderes de Ingeniería y Ciencia.

En cuanto al lenguaje de alto nivel, Visual Basic, es ampliamente utilizado en muchas empresas que desarrollan aplicaciones de acceso a datos, librerías dinámicas (DLL), programas que permiten la conectividad entre diferentes controles o programas de Windows, etc.

Con el presente proyecto se pretende facilitar el aprendizaje de Matlab a programadores noveles. Para ello se ha desarrollado una aplicación con Visual Basic, que puede ser utilizada por cualquier estudiante de Ingeniería o Ciencias sin ningún tipo de conocimiento previo de programación y le permitirá adquirir un conocimiento básico de Matlab en un tiempo mínimo.

Para desarrollar esta aplicación se ha utilizado el programa Visual Basic ya que dispone

de una interfaz gráfica fácil de utilizar y muy intuitiva, que permite incluir todo tipo de objetos necesarios en una aplicación visual, lo que resulta más atractivo para el usuario final.

La herramienta desarrollada en este proyecto aporta una visión nueva del entorno de Matlab, ya que está basada en la experiencia de una estudiante de la E.T.S.I.I. de Béjar, que conoce perfectamente las necesidades que un estudiante de Ingeniería Industrial se encuentra en la realización de sus estudios. Teniendo en cuenta dichas necesidades, se considera oportuno generar esta aplicación e incluir en ella las asignaturas comunes en todos los planes de estudio de la Escuela: Aritmética, Álgebra, Cálculo, Estadística, Programación y otros módulos adicionales como los de Gráficos e Imágenes, Aplicaciones y el de Ayuda. En la mayoría de estos módulos además de resolver los ejemplos básicos, el estudiante puede introducir otros nuevos, ya que dispone de las herramientas necesarias para calcular cualquier ejercicio en Matlab si conoce las instrucciones que los resuelven.

1.1 Objetivos del Proyecto

Este Proyecto Fin de Carrera surge del deseo de proporcionar una herramienta de aprendizaje e introducción a Matlab en la formación de Ingenieros y estudiantes de Ciencias en general. En asignaturas como Automática, Vibraciones, Simulación y Modelado, etc. se utiliza el programa Matlab como herramienta de apoyo durante el curso. La aplicación diseñada y desarrollada en este proyecto contribuirá al aprendizaje de Matlab y ha sido desarrollada mediante una interfaz sencilla y básica con la que el estudiante podrá interactuar y desarrollar sus destrezas tanto en programación, como en matemáticas o aspectos básicos específicos de mecánica o electricidad.

Con este tipo de aplicaciones se impulsa el propio aprendizaje del estudiante, haciéndolo autodirigido y autogestionado, permitiéndole desarrollar habilidades cognitivas. El estudiante asume la responsabilidad y compromiso de su propio aprendizaje, toma la iniciativa en el diagnóstico de sus necesidades educativas, eligiendo y poniendo en práctica estrategias de estudio adecuadas con vistas a la posterior evaluación de las mismas.

Los objetivos que se pretenden alcanzar con la realización de este trabajo se pueden resumir en:

1. Crear una aplicación que ayude al estudiante con las asignaturas básicas de su plan de estudios.
2. Diseñar una aplicación con la que el estudiante pueda gestionar y dirigir su estudio y que a la vez sea atractiva a los usuarios a los que va dirigido.
3. Desarrollar una aplicación que permita a programadores noveles aprender a utilizar Matlab.
4. Contribuir con dicha aplicación a la comunidad académica y más concretamente, para la utilización en el Departamento de Matemática Aplicada, así como para los estudiantes de primer y segundo curso del Grado en Ingeniería Industrial.

1.2 Descripción del Proyecto

Para el desarrollo de la herramienta para el apoyo a programadores noveles en el programa Matlab que se ha realizado en este Proyecto Fin de Carrera, se han utilizado las funcionalidades con las que cuenta Visual Basic, consiguiendo una aplicación que resulta atractiva para el usuario final, puesto que incluye diferentes formularios y controles. El ejecutable realizado puede ser utilizado en todo tipo de ordenadores, independientemente de su sistema operativo y de disponer del software de Visual Basic, siendo fundamental para su funcionamiento el programa Matlab.

Los menús que parecen más interesantes, a la vista de los potenciales estudiantes a los que puede dirigirse la herramienta desarrollada son los siguientes:

1. Aritmética Básica
2. Álgebra
3. Cálculo
4. Estadística
5. Gráficos e Imágenes

6. Programación
7. Aplicaciones
8. Ayuda

Cada uno de estos menús consta a su vez de otros submenús donde se pueden encontrar todas las nociones básicas, que un estudiante de Ingeniería o Ciencias puede necesitar para desarrollar sus estudios con éxito.

En los capítulos siguientes de esta memoria se detallará cada uno de los aspectos desarrollados en la aplicación, así como las funciones y comandos de Matlab necesarios para los cálculos realizados.

La idea inicial de la herramienta desarrollada era realizar un entorno de trabajo en Visual Basic que trabajara con instrucciones de Matlab, de forma que el usuario de la aplicación pudiera seguir las instrucciones del programa y de esa forma aprender Matlab. Las dificultades iniciales en el aprendizaje de un lenguaje de programación de alto nivel son las mismas que se pueden encontrar al tratar de utilizar todas las potencialidades de un programa como Matlab. A pesar de que éste último no se considera un lenguaje de programación, sí que incluye las mismas instrucciones y estructuras de control, con su sintaxis correspondiente. Así, por ejemplo, si se trata de utilizar una función, un bucle *for* o una sentencia *if*, habrá que disponer de los conocimientos de programación necesarios para entenderlos. Con la aplicación desarrollada aquí se pretende que el estudiante solvente, en la medida de lo posible, las dificultades que puede encontrarse cuando se enfrenta por primera vez a un entorno de programación, mostrándole unas ventanas suficientemente intuitivas y con la información necesaria para poder ejecutar paso a paso cada una de las operaciones y los ejemplos que se incluyen como parte esencial del programa.

El proyecto final consta de una aplicación o ejecutable desarrollado en Visual Basic con Matlab integrado, de forma que la apariencia externa es de visual Basic e internamente trabaja con Matlab. Se incluye también un Manual de Usuario completo, en el que se incluyen todas las funcionalidades implementadas. Finalmente, se han desarrollado unos ficheros de ayuda, similares a los que se pueden encontrar en cualquier programa de

Windows y que permiten acceder a la ayuda del programa a través de la tecla de función F1 o del menú correspondiente.

CAPÍTULO 2: MATLAB

2.1 Introducción

Matlab es un paquete de software orientado hacia el cálculo numérico científico en Ingeniería, con aplicaciones como: cálculo numérico, utilización de algoritmos, resolución de problemas con formulación matricial, cálculos de estadística, optimización, etc. Su nombre significa Laboratorio de Matrices y se basó en el cálculo matricial de LINPACK y EISPACK. Posteriormente se han añadido librerías, denominadas *Toolboxes*, especializadas en diferentes áreas científicas. De entre ellas podemos destacar:

- *Simulink Toolbox:* proporciona un entorno gráfico interactivo y un conjunto de librerías de bloques que le permiten diseñar, simular, implementar y probar muchos sistemas variables en el tiempo, incluidas las comunicaciones, los controles, procesamiento de señales, procesamiento de vídeo y procesamiento de imágenes.
- *Control System Toolbox:* proporciona algoritmos estándares en la industria y herramientas para analizar, diseñar y ajustar sistemas de control lineales (función de transferencia, sistema de espacio de estados, polos, ceros y ganancia, o modelo de respuesta en frecuencia).
- *System Identification Toolbox:* sistema de identificación de herramientas para construir modelos dinámicos lineales y no lineales del sistema de medición de entrada y salida de datos.
- *Robust Control Toolbox:* proporciona las herramientas para el análisis y ajuste de forma automática de los sistemas de control de rendimiento y robustez (diseño de controladores robustos para plantas de incertidumbre)**.**
- *Signal Processing Toolbox:* proporciona algoritmos estándares para el procesamiento de señales analógicas y digitales.
- *Filter Design Toolbox:* proporciona las herramientas necesarias para disponer de técnicas avanzadas para diseñar, simular y analizar los filtros digitales.

- *Symbolic Math Toolbox:* proporciona herramientas para la resolución de problemas, la manipulación de expresiones simbólicas matemáticas y la utilización de la variable de precisión aritmética.

Matlab ha evolucionado y crecido con las aportaciones de muchos usuarios. En entornos universitarios se ha convertido, junto con Mathematica, Maple, Derive u Octave en una herramienta de formación básica para asignaturas de matemáticas aplicadas así como para asignaturas de otras áreas, tal y como se indicó en la Introducción. En entornos industriales se utiliza para investigar y resolver problemas prácticos y cálculos de ingeniería. Es de destacar la aplicación en el estudio, simulación y diseño de los sistemas dinámicos y de control.

2.2 Características principales

Matlab es capaz de procesar de modo secuencial una serie de instrucciones previamente definidas, obteniendo los resultados de forma inmediata. Las instrucciones pueden estar ya definidas en el propio Matlab o ser definidas por el usuario. Para ejecutar las instrucciones se ha de escribir la lista de ellas en la ventana de instrucciones o en un fichero con extensión .m constituyendo entonces un programa. Tras ejecutar ENTER, Matlab procesa la información. Por último Matlab muestra los resultados en la misma ventana de instrucciones.

Los elementos básicos del escritorio de Matlab, tal como se muestra en la Figura 1, son:

- Ventana de instrucciones (*Command Windows*): Es la ventana principal, que se utiliza para introducir variables y ejecutar programas.
- Histórico de Instrucciones (*Command History*): Almacena y visualiza las instrucciones ejecutadas en la ventana de instrucciones. Permite recuperar y ejecutar cualquiera de las instrucciones anteriores sin más que hacer doble click sobre ella.
- Directorio de trabajo actual (*Current directory*): muestra los ficheros que hay en el directorio de trabajo actual.
- Espacio de trabajo (*Workspace*): Proporciona información sobre las variables que se han definido en la sesión actual, junto con sus dimensiones. En la Figura 1 no se muestra dicha ventana abierta, para ello se debe pinchar en el botón *Workspace* que se observa en la misma.

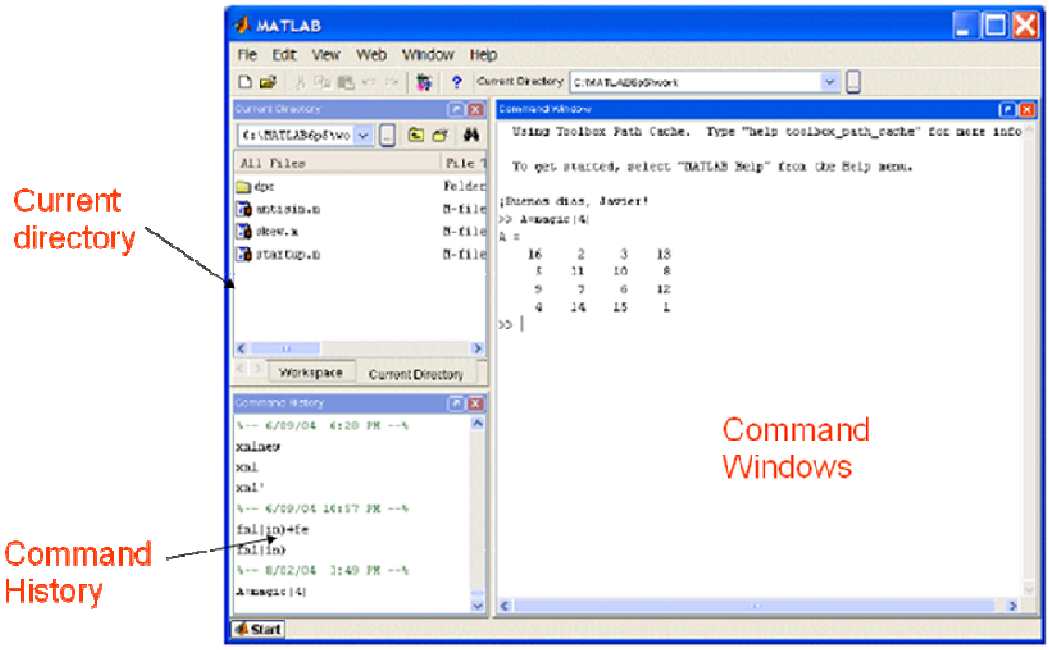


Figura 1: Escritorio de Matlab

Las instrucciones `demo, help, who, whos, dir, diary, load, save, clear, exit o quit` resultan muy útiles para el usuario en el desarrollo de su actividad ya que informan de lo siguiente:

- `Demo` muestra, de modo interactivo, un amplio abanico de ejemplos de aplicación de Matlab y es de gran ayuda durante los inicios con el programa.
- `Help función_deseada` muestra en la pantalla un texto explicando cómo se utiliza.
- `Who y whos` dan una lista de las variables que están actualmente en la memoria (*workspace* de Matlab).
- `Dir` lista el directorio actual.
- `diary` sirve para que todo lo que vamos tecleando y los resultados obtenidos (incluidos los errores) se almacenen en un archivo.
- `Load` carga variables de un archivo en el espacio de trabajo.
- `Save` guarda variables en un fichero.
- `Clear` borra de la memoria todas las variables.
- `Exit o quit` sirven para salir de Matlab.

El lenguaje de programación de Matlab dispone de los siguientes **operadores relacionales** que se detallan en la Tabla I:

Tabla I: Operadores relacionales de Matlab

Operador	Descripción
<	menor que
>	mayor que
<=	menor o igual que
>=	mayor o igual que
==	igual que
~=	distinto que

Los **operadores lógicos** de MATLAB se detallan en la Tabla II:

Tabla II: Operadores lógicos

Operador	Descripción
&	**and** (función equivalente: and(A,B)**):** Se evalúan siempre ambos operandos y el resultado es verdadero solo si ambos son ciertos.
&&	**and breve:** si el primer operando es falso ya no se evalúa el segundo, pues el resultado final ya no puede ser más que falso.
|	**or** (función equivalente: or(A,B)): Se evalúan siempre ambos operandos y el resultado es falso, sólo si ambos son falsos.
||	**or breve:** si el primer operando es verdadero ya no se evalúa el segundo, pues el resultado final no puede ser más que verdadero.
~	**negación lógica** (función equivalente: not(A)): Se niega el operando introducido dentro del paréntesis, mediante dicho operador.
xor(A,B)	realiza un "**or exclusivo**": es decir, devuelve 0 en el caso en que ambos sean 1 ó ambos sean 0.

Se tendrá en cuenta que Matlab distingue entre mayúsculas y minúsculas. Un carácter especial que se utiliza para escribir comentarios es el signo de tanto por ciento, que se utiliza delante del comentario que se quiere realizar (%). El punto y coma al final de la línea ordena a Matlab que evalúe la línea pero no muestre el resultado.

2.3 Funciones matemáticas y cálculo numérico

Para realizar operaciones aritméticas básicas en Matlab, se definen primero las variables y luego se opera con ellas para realizar sumas, restas, multiplicaciones y divisiones de la siguiente forma:

```
>> a = 3;
>> b = 4;
>> s = a+b; r = a-b; p = a*b; d = a/b;
```

Para el cálculo de **funciones trigonométricas** en Matlab, en primer lugar hay que definir un valor cualquiera como parámetro de entrada (a), después se escribe la instrucción correspondiente a la función trigonométrica (ver Tabla III) que se quiera calcular.

Tabla III: Funciones trigonométricas

Función Trigonométrica	Instrucciones Matlab
Seno	`sin(a)`
Coseno	`cos(a)`
Tangente	`tan(a)`
Cotangente	`cot(a)`
Secante	`sec(a)`
Cosecante	`csc(a)`
Función Trigonométrica Inversa	**Instrucciones Matlab**
Arcoseno	`asin(a)`
Arcocoseno	`acos(a)`
Arcotangente	`atan(a)`
Arcocotangente	`acot(a)`
Arcosecante	`asec(a)`
Arcocosecante	`acsc(a)`
Función Trigonométrica Hiperbólica	**Instrucciones Matlab**
Seno hiperbólico	`sinh(a)`
Coseno hiperbólico	`cosh(a)`
Tangente hiperbólica	`tanh(a)`

Cosecante hiperbólica	`asinh(a)`
Secante hiperbólica	`acosh(a)`
Tangente hiperbólica	`atanh(a)`

A continuación y como ejemplo se calculará el coseno de π/6, utilizando las instrucciones de Matlab vistas anteriormente.

```
>> a = pi/6;
>> cos(a)
ans =
    0.8660
```

Los ángulos utilizados y obtenidos en todas las operaciones trigonométricas vienen dados en radianes.

Matlab se caracteriza por su manejo de las matrices y vectores.

- **Vectores**

Un **vector** fila se define introduciendo los componentes separados por espacios o por comas entre corchetes: T = [1, 2, 3];

Para definir un vector columna, se separan los componentes por punto y coma:

```
>> T = [1; 2; 3]
T =
  1.0000
  2.0000
  3.0000
```

Podemos cambiar filas por columnas transponiendo el vector con el apóstrofe o definiendo el vector separado por comas de la siguiente manera:

```
>> T'
ans =

    1.0000   2.0000   3.0000

>> T = [1,2,3]
ans =

    1.0000   2.0000   3.0000
```

Las operaciones matemáticas más elementales (suma, resta, multiplicación y división) se pueden utilizar con vectores:

```
>> M = [1;2;3]
>> N = [4;5;6]
>> M+N
ans =
     5
     7
     9
```

```
>> M-N
ans =
    -3
    -3
    -3
```

Para realizar la multiplicación de vectores, el número de columnas debe ser igual al número de filas porque se realiza elemento a elemento, para ello se utiliza el operador * por el traspuesto del otro vector, como se detalla a continuación.

```
>> M*N'
ans =
     4     5     6
     8    10    12
    12    15    18
```

Matlab no permite la división M/N, pero sí realiza la división de vectores elemento a elemento, para ello se utiliza un punto delante del operador /.

```
>> M./N
ans =
    0.2500
    0.4000
    0.5000
```

Para crear un vector de componentes equiespaciados se emplean los dos puntos. La instrucción es: *x* = incio:paso:fin, indicando que los componentes de *x* van desde inicio hasta fin con un paso, tal como se muestra en el ejemplo siguiente:

```
>> x = 4:2:10

x =

  4  6  8  10
```

- **Matrices**

Para introducir **matrices**, se separa cada fila con un punto y coma y los elementos de la fila con un espacio o una coma:

```
>> M = [1 2 3 ; 4 5 6 ; 7 8 9]

M =

  1  2  3

  4  5  6

  7  8  9
```

Para referirse a un elemento de la matriz se indica el número de la fila y el de la columna separados por una coma:

```
>> M(3,1)

ans =

     7
```

Para referirse a toda una fila o a toda una columna se emplean los dos puntos, en el ejemplo se muestra la columna 2 de la matriz M, (v1).

```
>> v1 = M(:,2)

v1 =

     2

     5

     8
```

Con las matrices también se pueden realizar operaciones elementales como elevar a una potencia:

```
>> M^2

ans =

    30   36   42

    66   81   96

   102  126  150
```

Si se quiere operar en los elementos de la matriz, uno por uno, se pone un punto antes del operador. Por ejemplo si se quiere elevar al cuadrado cada uno de los elementos de M:

```
>> M.^2

ans =

     1   4   9

    16  25  36

    49  64  81
```

Una vez conocidas las diferentes formas de introducir y manejar matrices se pueden realizar operaciones matemáticas básicas con ellas (sumas, restas, multiplicaciones y divisiones), así como calcular la traspuesta, el determinante, el rango o la inversa de una matriz, tal y como se observa en la Tabla IV.

Tabla IV: Funciones básicas de las matrices

Función	Instrucciones Matlab
Traspuesta	`A'`
Determinante	`det(A)`
Rango	`rank(A)`
Inversa	`inv(A)`

Otras funciones elementales de Matlab que se utilizan, son las que se detallan en la Tabla V y para ello se define un valor cualquiera como parámetro de entrada *a*:

Tabla V: Funciones Elementales de Matlab

Funciones Elementales	Instrucciones Matlab
Exponencial	`exp(a)`
Logaritmo natural (base e)	`log(a)`
Logaritmo en base 2	`log2(a)`
Logaritmo en base 10	`log10(a)`
Raíz cuadrada	`sqrt(a)`

- **Polinomios**

En Matlab los **polinomios** se representan por vectores cuyas componentes son los coeficientes del polinomio, colocándolos de mayor a menor orden indicando siempre su

signo negativo, si lo hubiese. Si tenemos el polinomio $P(x)= x^2-3x+2$ se representará de la siguiente forma:

```
>> p = [1 -3 2]

p =

   1   -3   2
```

Para hallar sus raíces se utiliza la instrucción `roots`.

```
>> roots(p)

ans =

     2

     1
```

Si se quiere hallar el valor numérico de $P(x)$ para un determinado valor de x se usa la instrucción `polyval(p, nº)`, por ejemplo, para $x = 0$:

```
>> polyval(p,0)

ans =

     2
```

- **Autovalores y autovectores**

Si se quieren obtener los **autovalores y autovectores** de una matriz diagonal se utiliza la instrucción `eig`. Por ejemplo, para calcular los autovectores de la Matriz A= [4 2;3 3]:

```
>> A = [4 2;3 3]
>> [Q,D] = eig(A)
Q =
    0.7071   -0.5547
    0.7071    0.8321
D =
     6     0
     0     1
```

Donde Q es la matriz que tiene por columnas los autovectores y D es la matriz diagonal de los autovalores.

- **Intervalos de confianza**

Para calcular el **intervalo de confianza** para la media poblacional μ con un determinado nivel de confianza, se utiliza la instrucción `normfit`, que devuelve el valor de la media muestral (`muhat`), desviación estándar (`sigmahat`) y el intervalo de confianza para la media poblacional (`muci`), por ejemplo para una muestra x= [1.25, 1.36, 1.22, 1.19, 1.33, 1.12, 1.27, 1.27, 1.31, 1.26], si se quiere obtener el intervalo de confianza con un nivel de confianza del 95% (1-alpha = 0,95) o con un nivel de significación alpha = 0,05 se haría:

```
>> x = [1.25, 1.36, 1.22, 1.19, 1.33, 1.12,1.27, 1.27,
1.31, 1.26 ]
>> alpha = 0.05
>> [muhat, sigmahat, muci] = normfit(x,alpha)
muhat =
    1.2580
sigmahat =
    0.0697
muci =
    1.2081
    1.3079
```

- **Ecuaciones diferenciales**

Para resolver de forma **exacta** una o varias **ecuaciones diferenciales**, se dispone en Matlab de la orden `dsolve`, por ejemplo, para resolver la ecuación diferencial de valor inicial:

$$\left.\begin{array}{l} u' = 1/2\ u \\ u(0) = 1/4 \end{array}\right]$$

```
>> u = dsolve('Du = u/2','u(0) = 1/4')

u =

1/4*exp(1/2*t)
```

Para calcular soluciones **numéricas de ecuaciones diferenciales ordinarias** se utiliza la instrucción `ode45`, esta instrucción proporciona la solución, que se calcula por el método Runge-Kuta de cuarto y quinto orden. Por ejemplo, para resolver una ecuación diferencial ordinaria de valor inicial:

$$\left.\begin{array}{l} [t0,tf] = [0,20] \\ y0 = 0.1 \end{array}\right]$$

Se define con anterioridad el archivo fun.m de la siguiente forma:

```
function du = fun(t,y)
du1 = 0.3*y*(2-y);
du= [du1];
return
```

```
>> [t,y] = ode45('fun',[0,20],0.1)
>> M = [t,y]
M =
         0    0.1000
    0.0881    0.1051
    0.1763    0.1105
    0.2644    0.1162
      .         .
      .         .
      .         .
   17.9619    1.9992
   18.4715    1.9994
   18.9810    1.9996
   19.4905    1.9997
   20.0000    1.9998
```

2.4 Gráficos disponibles

Matlab incluye herramientas gráficas que permiten representar los datos y las operaciones de una manera cómoda para su interpretación. Para representar gráficas con Matlab, primero se genera una tabla de valores para, a continuación, representar la función.

Por ejemplo, si se quiere representar la parábola de ecuación $y = 2x^2+3x-1$ en el intervalo [-3, 3].

Se definen dos vectores de igual tamaño *n*. Se define el vector *x*:

```
>> x = -3:0.1:3;
```

El vector *y*, de *n* componentes, se define según la fórmula:

$$y_i = 2x_i^2 + 3x_i - 1 \ , \ i = 1.....n$$

```
>> y=2*x.^2+3*x-1;
```

Aclaración: el `.' antes del exponente evita que el término *x*^2, al ser *x* un vector, se calcule como el producto escalar de *x* por sí mismo, no componente a componente. Finalmente, definidos los vectores, se representa la gráfica y se etiqueta con las siguientes instrucciones:

```
>> plot(x,y); % Instrucción que ejecuta el gráfico
>> grid; % Cuadrícula sobre el gráfico
>> xlabel('Eje x'); % Etiqueta eje x
>> ylabel('Eje y'); % Etiqueta eje y
```

Este gráfico se representa en la Figura 2:

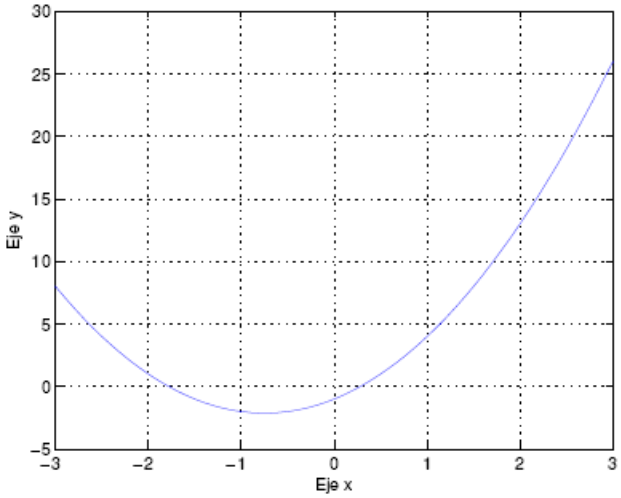


Figura 2: Representación de la parábola

El aspecto del gráfico se puede modificar utilizando alguna de las instrucciones enumeradas en la Tabla VI.

Tabla VI: Elementos que se pueden modificar en un gráfico

Elementos del Gráfico	Instrucciones Matlab
Color(rojo) y trazo(asteriscos)	`plot(x,y,'r*')`
Variar ejes	`axis`
Zoom	`zoom on /off`
Varios gráficos en la misma figura	`hold on/off`
Título del gráfico	`title('texto')`
Leyenda del gráfico	`legend`
Texto en el gráfico	`gtext('texto')`
Conocer coordenadas de un punto	`ginput`
Curvas en el plano	`comet`
Vectores velocidad	`quiver`
Generar un mallado	`meshgrid`
Efecto de sombreado	`surf`
Rotación de gráficos en 3d	`rotate3d`
Curvas de nivel	`contour`
Imprimir gráficos	`print`
Guardar gráficos	`save`

Exportar gráficos	`export`
Cierra todas las figuras	`close all`

Hay varias instrucciones en Matlab que ya vienen programadas y que permiten generar los gráficos de superficies en $\mathbb{R}^3$.

Esfera: Se genera utilizando la instrucción `>>sphere(n)`, donde `n` es el número de puntos en los que queda dividido el ecuador de la esfera. Poniendo solo `>>sphere`, el valor que tomará `n` será 20, por defecto.

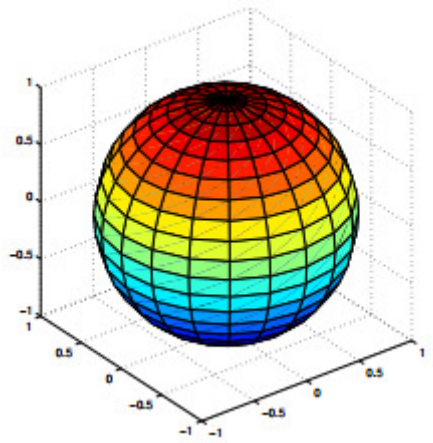


Figura 3: Esfera

Cilindro: La instrucción `>>cylinder(R,n)` genera automáticamente un cilindro de revolución de radio `R`, donde `n` es el número de puntos de la circunferencia de la base del cilindro. Como en el caso de la esfera, si se usa solo `>>cylinder(R)`, el número `n` es, por defecto, 20.

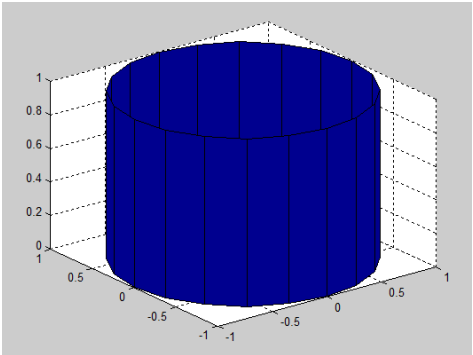


Figura 4: Cilindro

Otras superficies de revolución: La instrucción `>>makevase` hace aparecer una ventana interactiva que permite representar gráficos de superficies de revolución en las que la generatriz es una poligonal cuyos vértices se señalan con el ratón sobre la propia representación.

2.5 Programación

La programación en Matlab se realiza básicamente sobre archivos M, o M-Files. Se los denomina de esta forma debido a su extensión .m. Estos archivos son simples archivos ASCII o scripts y como tales, pueden definirse y modificarse desde cualquier editor de texto común tal como, el Bloc de Notas. Matlab incluye un editor de archivos M, orientado a la programación sobre este software. Si se opta por otro editor, se debe vigilar siempre que los archivos escritos se guarden con esta extensión.

Según cómo se definan, estos archivos pueden separarse en dos tipos:

- Archivos de instrucciones o scripts
- Funciones

Un **script** es una secuencia de instrucciones de Matlab guardada en un archivo con extensión .m, como es el caso del ejem_script.m mostrado en la Figura 5.

```
%Script de ejemplo

%% Inicio
a=magic(4);
fprintf('Inicio cálculos\n');

%% Traza
traza=sum(diag(a));

%% Resultado
fprintf('La traza vale: %f\n',traza)
```

Figura 5: ejem_script.m

Se ejecuta escribiendo su nombre:

```
>> ejem_script
```

Tanto las **funciones** como los **script** se escriben en archivos .m que deben encontrarse en el directorio actual (o en un directorio definido en la variable *path*).

La principal diferencia entre estas dos aplicaciones radica en que, en el primer caso, las variables empleadas son globales (se definen en el *Workspace*); mientras que para las funciones, las variables son de tipo local, debiendo declararse una lista de argumentos

de entrada a la función y sus valores de retorno.

Una función posee la siguiente estructura en su primera línea:

function[Argumentos_salida] = Nombre[Argumentos_entrada]

- **Nombre:** es el nombre de la función.
- **Argumentos_entrada:** son los valores de entrada a la función. Las variables independientes.
- **Argumentos_salida:** es el valor de salida de la función. Las variables dependientes.

El nombre de la función debe coincidir con el nombre del archivo .m con el cual se ha guardado. Si no es así podrían existir errores de directorio y/o ejecución.

Dentro del cuerpo de la función, se puede salir de la función mediante la instrucción `return`. Esta instrucción detiene la ejecución del programa y devuelve el valor actual de las variables de salida a la función, procedimiento o rutina que lo llamó.

Matlab tiene un lenguaje de programación propio, de tipo intérprete. Es decir, es capaz de interpretar una lista de instrucciones contenidas en un fichero .m.

Igual que otros lenguajes de programación, dispone de las estructuras de programación clásicas: `if-then-else, for, while y switch.`

La estructura básica para la bifurcación es el `if…then…else…end.` En Matlab, se escribe:

```
if <condición>
<acciones a realizar si se verifica la condición>
else
<acciones a realizar en caso de no verificarse la condición>
end
```

En donde <condición> es un valor, expresión, o función booleana.

Una forma de ampliar esta estructura es haciendo uso del `elseif`. Esta instrucción permite hacer varias evaluaciones y, por lo tanto, obtener más de dos posibles alternativas. A continuación se muestra la sintaxis para el uso del `elseif.`

```
if <condición 1>
   <acciones a realizar para condición 1>
elseif <condición 2>
   < acciones a realizar para condición 2>
else
   < acciones a realizar si ninguna condición se cumple>
end
```

La definición de una estructura `for` es:

```
for <variable> = <total de elementos>
    <sentencias>
end
```

En los casos particulares para los cuales se usará esta instrucción, el número total de elementos no es más que un vector de una fila y N columnas; por lo tanto, la variable sólo tomará valores escalares, como en el ejemplo que sigue:

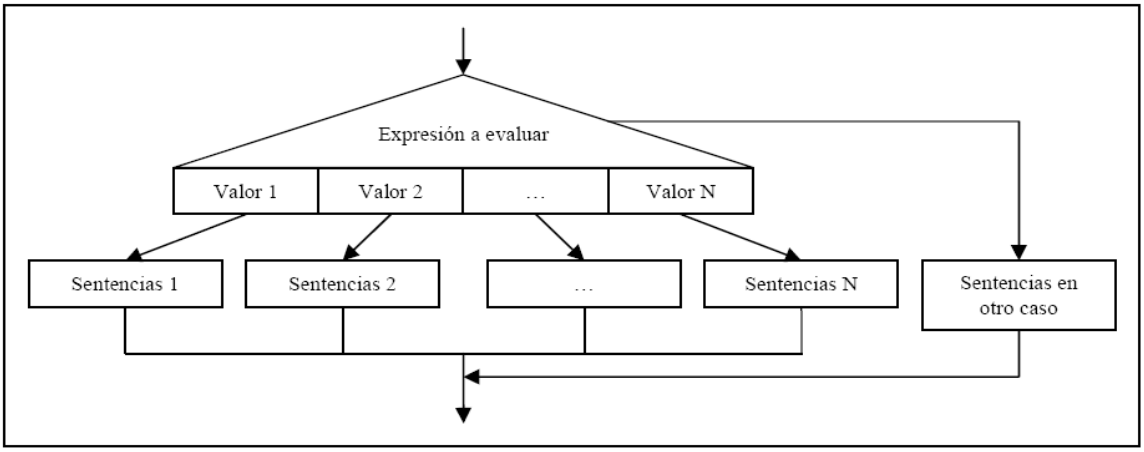
```
for I = 1:N
<sentencias>
end
```

Aquí, todos los elementos están representados por el vector definido por 1: N, es decir, por una matriz de la forma [1, 2, 3, …, N].

Para definir una estructura `while` se usa la siguiente sintaxis:

```
while <condición>
      <sentencias>
end
```

La sentencia `switch` se usa para crear una estructura similar al `case` de control. Esta estructura puede entenderse como varios `elseif` anidados, con la diferencia de que en la estructura `case` no se controlan expresiones booleanas, sino valores de una determinada expresión.




```
switch <expresión a evaluar>
    case <valor 1>
        <sentencias 1>
    case <valor 2>
        <sentencias 2>
    case <…..>
        <….>
    case <valor N>
        <sentencias N>
    otherwise
        <sentencias en otro caso>
end
```

2.6 Simulink: Herramienta de simulación

Simulink es una herramienta de gran utilidad para la simulación de sistemas dinámicos. Principalmente, se trata de un entorno de trabajo gráfico, en el que se especifican las partes de un sistema y su interconexión en forma de diagramas de bloques. Es una herramienta que se complementa con numerosos elementos opcionales.

Además de las capacidades de simulación de las que está dotado Simulink, conviene destacar que contiene cómodas utilidades de visualización y almacenamiento de resultados de simulaciones.

Se comienza llamando a la aplicación, para ello escribiremos:

`>>simulink` en la línea de instrucciones de Matlab, o abriendo desde el Explorador de Windows cualquier fichero con extensión .mdl.

Esta ventana inicial no está destinada a crear modelos de simulación; su función principal consiste en navegar por la enorme librería de bloques disponibles para el modelado.

En ella se distinguen dos partes: la izquierda contiene una visión en forma de árbol de todos los *Toolboxes* instalados que contienen bloques Simulink. La amplitud de este árbol dependerá de las opciones que hayamos activado al seleccionar Matlab.

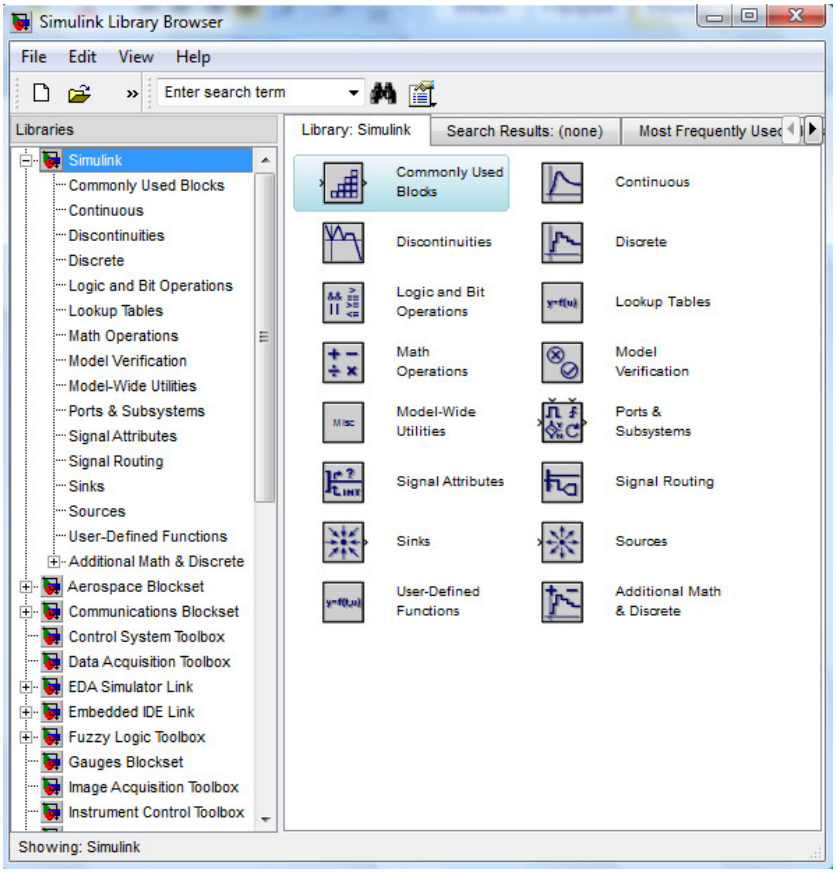


Figura 6: Librería Simulink

De todos los nodos del árbol se utilizarán aquí *Simulink y Control System Toolbox* porque es donde se encuentran los bloques que se necesitan para diseñar el modelo de simulación que se detalla más adelante. Destacan los bloques *Real Time Workshop* destinados a generar automáticamente el código de control para determinadas plataformas hardware comerciales.

La parte derecha de la ventana muestra los bloques Simulink contenidos en el *Toolbox* o nodo de la parte izquierda de la ventana. Estos bloques se deben arrastrar sobre el espacio de trabajo de Simulink para la creación del modelo que vamos a simular.

Por último, en la parte superior de la ventana de inicio de Simulink hay varias herramientas como la búsqueda de un bloque determinado a partir de su nombre, que pueden resultar bastante útiles.

Al pulsar en el menú: File → New (ver Figura 6), se abre una ventana en blanco sobre la que se inicia la creación de un modelo de simulación, insertando los bloques correspondientes.

- **Ejemplo**

Realizar la simulación del sistema de control realimentado que a continuación se muestra:

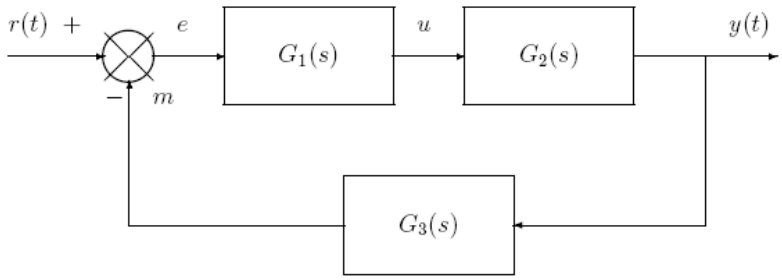


Figura 7: Modelo de simulación

Dicho sistema está formado por tres funciones de transferencia independientes con ganancias:

$$G_1(s)=\frac{1}{s+0,5};\ G_2(s)=\frac{3}{s^2+2s+1};\ G_3(s)=\frac{40}{s^2+5s+40};$$

En primer lugar, se deben insertar tres bloques de tipo Función de Transferencia en el modelo. *Transfer Fcn*, que se encuentra dentro del nodo Simulink → Continuous

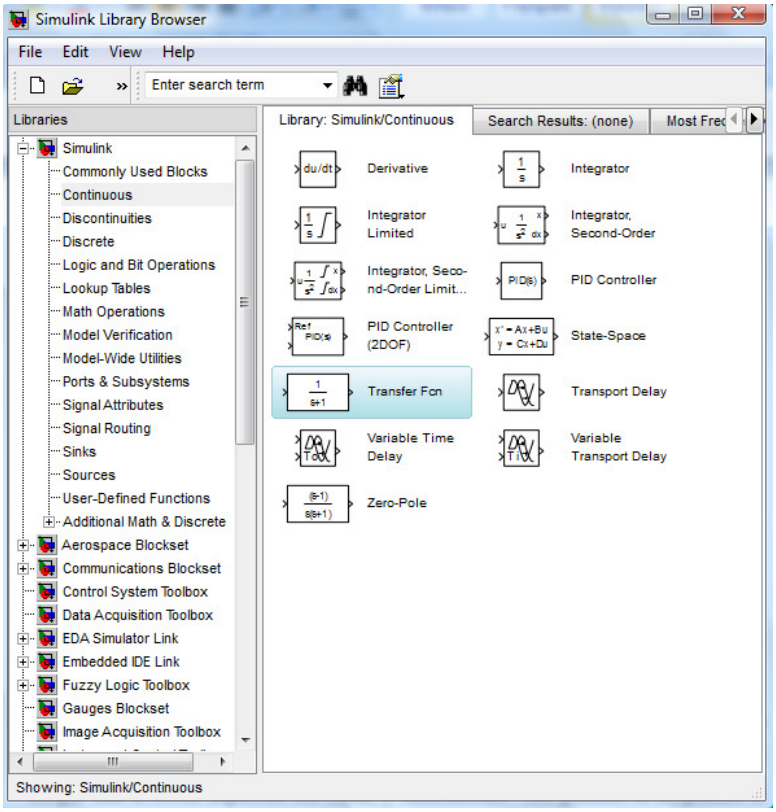


Figura 8: Bloque Transfer Fcn

Una vez localizado el bloque *Transfer Fcn* se arrastra dicho bloque hacia el espacio de trabajo de Simulink (Figura 9). El arrastre de bloques se realiza seleccionando el icono del bloque con el botón izquierdo del ratón y manteniendo éste pulsado se desplaza el cursor hasta la ventana del modelo. Se repite esta operación tres veces, una para cada uno de los bloques del ejemplo.

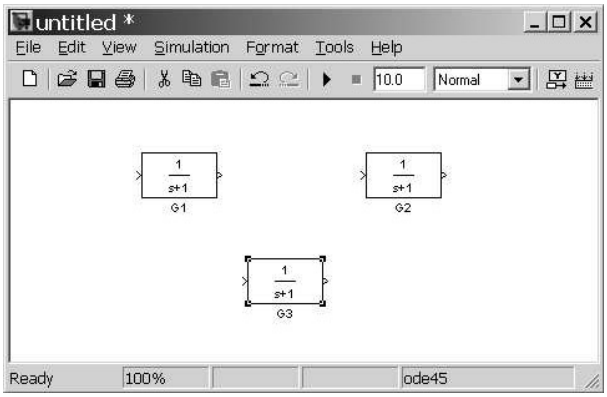


Figura 9: Entorno Simulink

Una vez insertados los bloques de las funciones de transferencia, se les asigna nombres específicos (G1, G2 y G3) para ello se edita el texto, haciendo click al pie de cada icono

y se establecen los valores de dichas funciones, para que coincidan con los parámetros de las funciones G1(s), G2(s) y G3(s) definidas anteriormente.

Con este fin, se hará doble click sobre cada bloque de función de transferencia y en la ventana que se abre en cada caso, se introducen los vectores de coeficientes de los polinomios numerador y denominador de cada función de transferencia (ver Figura 10 para G1(s)).

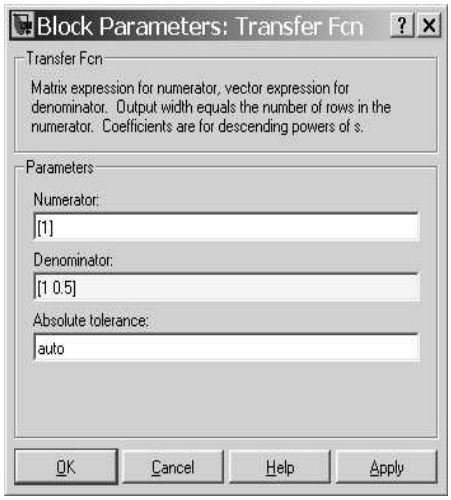


Figura 10: G1(s) = 1/(s + 0,5)

Una vez configuradas las tres funciones de transferencia, se conectarán entre sí con arreglo a la estructura de interconexión de bloques. Para ello se emplean las siguientes operaciones que se observan en la Tabla VII.

Tabla VII: Operaciones en Simulink

Operación	Procedimiento
Conectar bloques (I)	Para conectar las salidas de un bloque a la entrada de otro, hacer click con el botón izquierdo del ratón en el bloque origen. Pulsar y mantener la tecla CTRL y hacer de nuevo click sobre el bloque destino.
Conectar bloques (II)	También se puede extraer un cable de señal haciendo click en el saliente derecho del bloque origen y prolongar la señal (pulsando y manteniendo el botón izquierdo del ratón) hasta llegar a la parte izquierda del bloque destino.
Bifurcar cables	Un cable de señal (que lleva la salida de un bloque hacia otro bloque), puede bifurcarse para distribuir la señal a

	varios bloques pulsando con el botón derecho en cualquier punto del cable.
Sumar o restar señales	Las señales procedentes de salidas de los bloques se pueden sumar o restar entre sí mediante el bloque sumador, que se ubica fácilmente tecleando `Sum` en la ventana de navegación de Simulink.

Tras una serie de operaciones de los tipos indicados en la tabla anterior, se construye la estructura de realimentación (ver Figura 11).

Los bloques de suma y resta de señales y los de funciones de transferencia funcionan como procesadores de señal. Sin embargo, en las simulaciones han de existir fuentes de señal externas, pues lo que se pretende es ver cómo responden determinados sistemas a estímulos exteriores.

En este ejemplo se necesita una señal externa para generar una referencia a seguir por el sistema controlado. Esta referencia debe ser, lógicamente, cambiante con el tiempo. Por lo que se emplea una señal de tipo escalón, que se implementa, con sus parámetros específicos, mediante el bloque *Step*. Bloques como éste, que sólo tienen salidas y ninguna entrada, se localizan en el árbol de navegación de Simulink en el nodo *Simulink/Sources* (Figura 8).

Por otro lado, existen bloques con entradas y sin ninguna salida: nodos sumidero. Uno de ellos es el empleado en este modelo para visualizar la salida del sistema: *Scope*. Los bloques de este tipo se ubican en el árbol de navegación de Simulink en el nodo *Simulink/Sinks* (Figura 8).

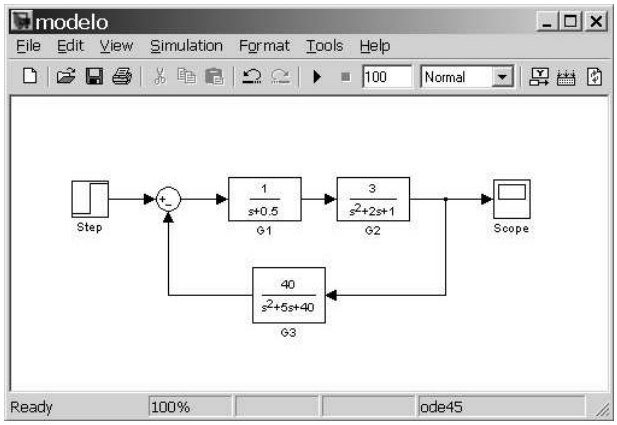


Figura 11: Estructura de realimentación

CAPÍTULO 3: VISUAL BASIC

3.1 Introducción

El sistema de programación Visual Basic, edición de aplicaciones (*Applications Edition*), se incluye en Microsoft Excel, Microsoft Access y muchas otras aplicaciones de Windows que utilizan el mismo lenguaje. El Visual Basic Script (VBScript) es un lenguaje script muy utilizado, principalmente en entornos web y es un subconjunto del lenguaje Visual Basic. El esfuerzo realizado para aprender Visual Basic será de utilidad en estas otras áreas mencionadas anteriormente.

La parte "Visual" se refiere al método utilizado para la Interfaz Gráfica del Usuario (GUI). En lugar de escribir grandes cantidades de código para describir la apariencia y posición de los elementos de la interfaz, como se haría en el lenguaje Java, por ejemplo, aquí simplemente se agregan objetos preconstruidos y se sitúan en la pantalla o formulario deseado.

La parte "Basic", se refiere al lenguaje BASIC (*Beginners All-Purpose Symbolic Instruction Code*), un lenguaje utilizado por más programadores que cualquier otro lenguaje en la historia de la computación. Visual Basic ha evolucionado desde el lenguaje Basic original y ahora contiene cientos de sentencias, funciones y palabras reservadas, muchas de las cuales se relacionan directamente con la GUI de Windows. Los programadores noveles pueden crear aplicaciones útiles sin más que aprender un pequeño número de palabras reservadas. Además, el alcance del lenguaje permite a los profesionales hacer casi cualquier cosa que se pueda hacer utilizando cualquier otro lenguaje de programación.

Visual Basic dispone de las siguientes herramientas:

- Conexión a bases de datos: Las características de acceso a datos permiten crear bases de datos y aplicaciones *front-end* (el usuario interactúa con su PC, interfaz del usuario, aplicación de escritorio, etc.) para muchas bases de datos utilizadas habitualmente como SQL Server y otras bases de datos de nivel corporativo.
- La tecnología ActiveX: Permite utilizar la funcionalidad de otras aplicaciones, como es el procesador de palabras de Microsoft Word, la hoja de cálculo de Microsoft Excel y otras aplicaciones Windows.

- Las características de internet: Hacen fácil acceder a documentos y aplicaciones a través de Internet o de una Intranet desde su aplicación; de igual forma se pueden crear aplicaciones servidor de Internet.
- La aplicación final: es un verdadero archivo .exe que utiliza una máquina virtual Visual Basic que se distribuye libremente.

Para familiarizarnos con Visual Basic (VB) definiremos algunos conceptos:

- **Sistema**: Conjunto de métodos, recursos y dispositivos para realizar un fin específico.
- **Método:** Acción que puede realizar un objeto. Por ejemplo: el método *Refresh o Move*.
- **Evento:** Acción que puede realizar un objeto. Los eventos normalmente se ejecutan como respuesta a una acción del usuario como presionar un botón. Los eventos también pueden dispararse por el sistema (como el Timer) o por una llamada en el código (a partir de ahora se citarán los eventos indicando su nombre entre comillas).
- **Objeto:** Entidad con características y acciones definidas de cierto tipo o clase.
- **Contenedor:** Objeto que existe para contener otros objetos y además puede realizar alguna acción.
- **Clase:** Define las características (propiedades) y acciones de un tipo de objeto. A las clases también se les llama **Moldes**.
- **Propiedad:** Son las características que definen su aspecto gráfico y se pueden modificar en tiempo de diseño y en tiempo de ejecución del programa. Ejemplo de propiedades son: ancho, alto, color, etc.
- **Función:** Conjunto de instrucciones agrupadas en un procedimiento y que devuelve un valor.
- **Ejecución asíncrona:** Algunas funciones pueden ser ejecutadas de manera asíncrona, es decir, la aplicación llama a la función y después de un mínimo procesamiento, devuelve el control a la aplicación. Entonces la aplicación puede llamar a otras funciones mientras la primera función continúa ejecutándose.

- **API:** (*Application Programming Interface*). Son las funciones, mensajes, estructuras de datos, tipos de datos y sentencias que se pueden utilizar para crear aplicaciones. El API de Windows consiste en un conjunto de librerías DLL's (*Dynamic-Link Library*) que contienen los procedimientos relacionados con el sistema. Para llamar a estos procedimientos desde Visual Basic, primero se deben declarar utilizando la sentencia Declare. Entonces se podrán llamar como cualquier otro procedimiento.
- **OLE** (*Object Linking and Embedded*): Permite compartir objetos generados por dos o más aplicaciones, con lo que se puede tener en un mismo formulario una hoja de Excel y un documento de Word.

Las ventajas de Visual Basic son:

- La estructura del lenguaje de programación BASIC es muy simple, en su código ejecutable.
- No es solo un lenguaje sino un entorno de desarrollo interactivo totalmente integrado.
- Ha sido altamente optimizado para soportar el desarrollo rápido de aplicaciones. Es fácil desarrollar una GUI y conectarla a funciones de manejo sencillo y permitidas por la aplicación.
- La GUI brinda una pantalla intuitiva y de vista atractiva para el manejo de la estructura del programa en general así como para las clases, módulos, procedimientos, formas, etc.
- Proporciona un sistema de ayuda fácil de comprender y sensible al contexto con pequeños iconos.
- Al editar programas, se despliega una ventana que informa al usuario del tipo de instrucciones posibles en la posición en la que se encuentra el cursor.
- Es un lenguaje de integración de componentes.
- Los componentes (COM) pueden escribirse en distintos lenguajes y después ser integrados utilizando Visual Basic.
- Es sencillo desarrollar aplicaciones para su distribución.

3.2 Entorno de desarrollo.

El entorno de trabajo en Visual Basic se denomina frecuentemente entorno integrado de desarrollo o IDE (*Integrated Development Environment*); integra funciones diferentes como el diseño, modificación, compilación y depuración en un entorno común. En las herramientas de desarrollo más tradicionales, cada una de esas funciones funcionaría como un programa diferente, con su propia interfaz.

Al ejecutar Visual Basic se observará un IDE, semejante al que se muestra a continuación:

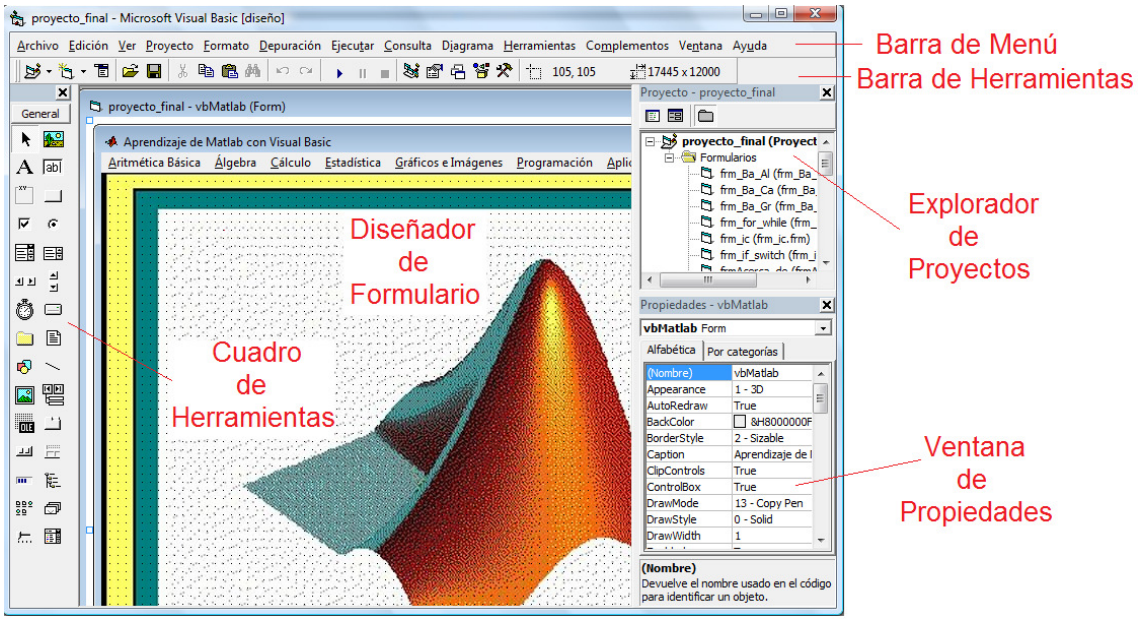


Figura 12: Entorno de trabajo Visual Basic

Este IDE se compone de los siguientes elementos:

- Barra de menú (*Menu Bar*): Presenta las instrucciones que se usan para trabajar con Visual Basic. Además de los menús estándar como Archivo, Edición, Ver, Ventana y Ayuda, se proporcionan otros menús para tener acceso a funciones específicas de programación, tales como Proyecto, Formato o Depuración, etc.

- Barra de herramientas (*Toolbar*): Proporciona un rápido acceso a las instrucciones utilizadas normalmente en el entorno de programación. De forma predeterminada, al iniciar Visual Basic se presenta la barra de herramientas estándar. Se pueden activar o desactivar otras barras de herramientas adicionales, para modificar o diseñar formularios, desde la Barra de herramientas del menú Ver. Las barras de herramientas se pueden acoplar debajo de la barra de menús o pueden "flotar" si se selecciona la barra vertical del borde izquierdo y se arrastra fuera de la barra de menús.
- Cuadro o caja de herramientas (*Toolbox)*: Proporciona un conjunto de herramientas que se pueden utilizar durante el diseño para situar controles en un formulario. Además del diseño del cuadro de herramientas predeterminado, se puede crear un diseño personalizado al seleccionar Agregar ficha en el menú contextual y se agregan controles a la ficha resultante.
- Área de trabajo: Es el área central del IDE, donde se realiza el diseño y la programación propiamente dicha. En este área se dispone de:
    - Diseñador de formularios (*Form Designer*): Funciona como una ventana en la que se personaliza el diseño de la interfaz de la aplicación que se vaya a realizar. Se agregan controles, gráficos e imágenes a un formulario para crear la apariencia que se desea. Cada formulario de la aplicación tiene su propia ventana Diseñador de formulario.
    - Editor de código (*Code Editor*): Funciona como un editor para escribir el código de la aplicación. Se crea una ventana editor de código diferente para cada formulario o módulo del código de la aplicación. Para mostrar dicha ventana se pulsa en la primera pestaña de la parte superior del explorador de proyectos.
- Explorador de proyectos (*Project Explorer Window*): Ventana donde se muestran los formularios y módulos del proyecto actual. Un proyecto es la colección de archivos cuya finalidad es la de generar una aplicación ejecutable.
- Ventana de propiedades: Ventana con los valores de las propiedades del control o formulario que se selecciona.

También se dispone de:

- Ventana de posición del formulario *(Form Layout Window)*: Permite colocar los formularios de la aplicación utilizando una pequeña representación gráfica de la pantalla. Se puede mostrar en el formulario haciendo click en el menú Ver / Ventana de posición del formulario.

Figura 13: Posición Formulario

- Examinador de objetos (*Object Browser*): Enumera los objetos disponibles que se pueden usar en el proyecto y proporciona una manera rápida de desplazarse a través del código. Se puede usar el examinador de objetos para explorar objetos en Visual Basic y otras aplicaciones, ver qué métodos y propiedades están disponibles para esos objetos y pegar códigos de procedimientos en la aplicación. Esta ventana se puede mostrar en pantalla de la misma forma que la anterior.
- Las ventanas de inmediato, locales e inspección (*Inmediate, Locals and Watch Windows*): Se utilizan para la depuración de la aplicación. Solo están disponibles cuando se ejecuta la aplicación dentro del IDE. También se pueden agregar características a la interfaz de Visual Basic mediante los complementos. Los complementos, disponibles en Microsoft y otros sistemas de desarrollo pueden proporcionar características como el control de código fuente, que permite mantener proyectos de desarrollo en grupo. Como ya se ha comentado anteriormente se puede mostrar esta ventana desde el menú Ver/ventana de inspección.

3.3 Controles básicos.

Como todos los programas desarrollados bajo el entorno de Windows, Visual Basic está pensado para facilitar la programación con un entorno de trabajo flexible y sencillo. Ofrece las opciones de menú estándar con las que cuentan la mayoría de los programas:

Archivo, Edición, Ver, Formato, Herramientas, Ventana, Ayuda; además de opciones como: Proyecto, Depuración, Ejecutar, Diagrama, Complementos, Consulta.

Los objetos más utilizados en VB son los **formularios:** ventanas que sirven de fondo para los controles y para los gráficos situados en las mismas. Se pueden utilizar tantos formularios como se necesiten y dependiendo de la utilidad que se les dé, serán de diferentes tipos. Así, se puede crear un formulario para que contenga un gráfico, para visualizar información o para aceptar datos. Sobre un formulario se añaden controles con el fin de aceptar o visualizar los datos. Para dibujar los controles se utiliza la herramienta *Toolbox* de VB.

Los **controles**, tales como cajas de texto, botones, marcos, listas o temporizadores, son objetos gráficos que permiten introducir o extraer datos. El formulario y los controles forman la interfaz del programa.

Cada uno de estos controles se tratan como objetos en Visual Basic, por lo que no se debe olvidar que tienen su propio conjunto de propiedades, métodos y eventos. Para la entrada de datos se utilizan principalmente dos controles muy relacionados: las etiquetas y los cuadros de texto.

La **etiqueta** (*Label*), permite mostrar texto en los formularios y tiene la particularidad de que el usuario no puede modificar su contenido.

El **cuadro de texto** (*TextBox*), es el control estándar de entrada de datos en Visual Basic. Permite al usuario de la aplicación introducir información. Algunas propiedades del control de etiqueta y del cuadro de texto se muestran en la Tabla VIII.

Tabla VIII: Propiedades de las etiquetas y cuadros de texto

Propiedad	Definición
Alignment	Establece la alineación del texto de la etiqueta
Autosize	Si su valor es verdadero, el tamaño de la etiqueta se adapta automáticamente a su contenido
Caption	Texto que se visualiza en la etiqueta
Enabled	Permite o no interactuar con la etiqueta
Font	Establece la fuente, tamaño y htmlecto del texto

Multiline	Permite introducir más de una línea de texto en el cuadro
PasswordChar	Estable el carácter que se muestra al realizar una entrada en el objeto. Sólo se mostrará dicho carácter
ScrollBars	Muestra barras de desplazamiento
TabIndex	Establece el orden de tabulación fijado para el objeto
TabStop	Indica si al utilizar el tabulador se puede desplazar entre los controles del formulario
Text	Texto que se visualiza en el control

Los cuadros de texto son semejantes a las etiquetas, pero con la diferencia de que el usuario puede modificar su contenido. El texto que se introduce puede ser tanto numérico como alfanumérico. A diferencia de la etiqueta, el tamaño del texto es fijo al no permitir la propiedad *Autosize*.

En las ventanas de entrada de datos es habitual tener la combinación de etiqueta y cuadro de texto, ya que si se quiere permitir un acceso rápido a cierto cuadro de texto, el uso de una etiqueta asociada es la forma más sencilla de hacerlo. Se utiliza el carácter & en la propiedad *Caption* de la etiqueta asociada y se creará la tecla de acceso para el cuadro de texto.

El **Control marco** (*frame*) se emplea para estructurar el formulario en varias secciones, agrupando en éstas los controles para que la lectura sea más sencilla. La propiedad más interesante de un marco es el título, que se refiere al texto que se presenta en la parte superior izquierda del control. Este control actúa como contenedor de otros controles, así que las propiedades izquierda, derecha, abajo y arriba de un objeto contenido en un marco se establecen en relación al objeto contenedor. Al mover un marco también se mueven los controles que contenga, para ello se debe añadir primero el marco y después los controles que se quieran contener.

El **Botón de instrucciones** (*CommandButton*), es muy sencillo, por lo que no se tienen que establecer muchas propiedades. Con el título se establece el texto que aparecerá en el botón. También se pueden crear teclas de acceso rápido al botón. El uso principal de los botones es realizar acciones en la aplicación.

La **Casilla de verificación** (*CheckBox*), permite elegir entre distintas opciones. Para ello se establecen opciones que no son excluyentes entre sí, es decir, se pueden seleccionar una o más de una. Una casilla de verificación puede estar activada (*checked*), o desactivada (*unchecked*). Además, la casilla puede estar atenuada (*grayed*), que indica que el objeto no está disponible. La propiedad que tienen estos valores de activada o desactivada es *Value*. El evento más utilizado en este tipo de objetos es el momento en el que se pulsa en la casilla, éste es el evento "CheckedChanged", con el que el usuario quiere indicar que desea activar o desactivar la casilla, dependiendo del valor que tenga en ese momento.

El **Botón de opción** (*OptionButton*), también permite presentar opciones al usuario, pero con la particularidad de que solo se puede seleccionar una de las opciones cada vez. Si se quiere que en un mismo formulario se pueda seleccionar más de un botón de opción, se deben añadir distintos conjuntos de ellos. Solo se pueden tener dos valores en la propiedad *Value*: verdadero si está activado y falso si está desactivado.

El **cuadro de lista** (*ListBox)*, es el control en el que se muestran los elementos verticalmente en una columna, aunque también se pueden establecer en más de una columna. Si el número de elementos excede a los que se pueden presentar en el cuadro de lista, aparecen automáticamente barras de desplazamiento en el control.

Entre las propiedades del cuadro de lista se destacan: *ListIndex*, que indica el índice de elementos seleccionados, teniendo valor 0 el primer elemento de la lista. *ListCount* es la propiedad que indica el número de elementos existentes en la lista en todo momento. Estas propiedades se utilizan en tiempo de ejecución, es decir, cuando se interactúa con la lista. Para indicar que la lista está ordenada se utiliza la propiedad *Sorted*, con el valor verdadero.
Para agregar elementos a la lista utilizamos el método *AddItem*. Para borrar elementos de la lista se utiliza el método *RemoveItem*.

A continuación, se muestran estos controles y su representación en la barra de herramientas, en la Tabla IX:

Tabla IX: Controles y su representación

Control	Representación
Etiqueta	
Cuadro de texto	
Marco	
Botón de instrucciones	
Casilla de verificación	
Botón de opción	
Cuadro de lista	

Cuando se desarrolla una aplicación en VB, existe un único archivo de proyecto, con extensión .vbp, que administra los diferentes archivos que se crean como componentes de la misma. El propio programa crea una carpeta, en la que se guarda el proyecto o aplicación, desde la que se pueden manipular todos sus componentes (módulos, formularios, etc.). Una aplicación puede contener formularios (.frm), módulos (.bas) y también controles personalizados (.vbx). Un formulario, a su vez, contiene controles, junto con el código asociado a ellos y el asociado al propio formulario. Un módulo solo contiene código.

En VB se pueden construir distintos tipos de aplicaciones. La ventana Nuevo Proyecto permite seleccionar el tipo de proyecto que se desea crear:

- EXE estándar**:** Este tipo de aplicación se construye a partir de uno o más formularios, módulos y clases.
- Control ActiveX: Crea un control ActiveX. Los controles no son simplemente código, sino que tienen componentes visuales como los formularios, aunque a diferencia de éstos, no pueden existir sin algún tipo de contenedor.
- Aplicación IIS *(Internet Information Server)*: Se trata de una aplicación hecha para residir en un servidor Web y responder a peticiones enviadas por un explorador.

- Aplicación DHTML: Se trata de una o más páginas de código HTML que utilizan código de Visual Basic y el modelo de objetos HTML dinámico para responder instantáneamente a las acciones que se produzcan en dichas páginas.
- Asistente para aplicaciones de VB: Genera una aplicación nueva completamente funcional desde la cual se puede generar una aplicación más compleja.
- Todos los proyectos con las extensiones: .vbp (archivos de proyecto), .vbz (archivos de asistente) o .vbg (grupo de proyectos) están en el directorio de proyectos.

Figura 14: Tipos de proyectos.

3.4 Programación.

Una **variable** es una ubicación temporal de memoria en la que se almacenan datos que interesa retener durante la ejecución de la aplicación. Las variables pueden contener texto, valores numéricos, fechas o propiedades de cierto objeto**.** Se caracterizan por el nombre que las identifica y por el tipo de datos, que establece el conjunto de valores posibles que pueden tomar las operaciones en las que pueden participar. El valor de una variable puede cambiar a lo largo de la aplicación, pero no el tipo.

En Visual Basic no es obligatorio declarar las variables, pero si recomendable para poder utilizarlas en el programa. Al declarar una variable se reserva memoria para ella.

Para establecer la declaración de una variable antes de ser utilizada en el código, se puede hacer en el módulo Declaraciones de la sección General, Visual Basic introduce la instrucción Option Explicit en la sección de declaraciones de cada módulo nuevo que se cree, no de los ya existentes donde se tiene que introducir manualmente. Esta instrucción obliga a declarar las variables antes de utilizarlas. El uso de variables puede hacer que la aplicación sea más rápida, por ejemplo, si se usa muchas veces un determinado valor, se aconseja guardarlo en una variable y utilizarla cuando se necesite. Es más rápido el acceso a las variables que a una propiedad de un objeto, por lo que también es aconsejable guardar el valor de una propiedad en una variable.

La forma de declarar una variable es a través de la instrucción Dim [Nombre de la variable] As [Tipo variable]. Mediante esta instrucción se especifica el tipo de datos, como se observa en el siguiente ejemplo, la variable, *sfichero*, se define como una cadena de caracteres:

```
Dim sfichero As String
```

El tipo *Variant* es un tipo especial de datos que puede contener cualquier clase de datos excepto cadenas de longitud fija y tipos definidos por el usuario. Si se utilizan variables de este tipo, no hay que preocuparse de efectuar conversiones entre tipos para utilizarlas en distintos contextos. Las variables de tipo *Variant* son muy flexibles, pero ocupan mucha memoria y disminuyen la velocidad.

De forma predeterminada, a no ser que se indique el tipo de datos, establece el tipo *Variant* para todas las variables. Así, al utilizar una instrucción como Dim NombreVariable, se especifica implícitamente el tipo *Variant* para dicha variable.

Se dispone de los siguientes tipos de datos (Tabla X):

Tabla X: Clasificación de los datos

Tipo de datos	Tamaño
Entero (Integer)	2 bytes
Entero Largo (Long)	4 bytes
Simple (Single)	4 bytes
Doble (Double)	8 bytes
Moneda (Currency)	8 bytes
Cadena de caracteres (String)	1 byte por carácter
Byte	1 byte
Boleano (Boolean)	2 bytes

Fecha (Date)	8 bytes
Objecto (Object)	4 bytes
Variant	16 bytes + 1 byte por cada carácter

Para declarar las variables se debe usar el tipo de datos en inglés.

Las **constantes** como su propio nombre indican, no pueden cambiar su valor a lo largo del programa. Para utilizar una constante hay que declararla previamente. La forma de declararla es a través de la instrucción: Const NombreConstante = Expresión, donde Expresión está compuesta por operadores de cadena o de números.

En Visual Basic existen muchos operadores que se pueden utilizar para crear fórmulas. Los operadores más utilizados en una aplicación de Visual Basic son los mostrados en la siguientes Tabla XI:

Tabla XI: Operadores de Visual Basic

Operador	Operación que realiza
+	Suma / Concatenación de cadenas de caracteres
-	Resta
*	Multiplicación
/	División
\	División entera
Mod	Resto de la división entera
^	Elevar a una potencia
&	Concatenación de cadenas de caracteres

A continuación se presentan brevemente los **eventos** más usados en esta aplicación, se comenzará con los más generales y se concluirá con los relacionados con el ratón.

Para inicializar las variables definidas a nivel de módulo se suele utilizar el evento "Initialize", que tiene lugar antes que "Load". El evento "Load" se activa al cargar un formulario. Con el formulario principal esto sucede al arrancar la ejecución de un programa, cuando se carga desde cualquier procedimiento o al hacer referencia a alguna propiedad o control de un formulario que no esté cargado. Al descargar un formulario se produce el evento "Unload", para ello es necesario cerrar la ventana con el botón de cerrar o llamarlo explícitamente. Si se detiene el programa desde el botón *Stop* o con un *End,* no se pasa por el evento "Unload".

Uno de los eventos más utilizados es "Click" que se activa cuando el usuario pulsa y suelta rápidamente uno de los botones del ratón. También puede activarse desde código

(sin tocar el ratón) variando la propiedad Value de alguno de los controles. En el caso de un formulario, este evento se activa cuando el usuario hace click sobre una zona del formulario en la que no haya ningún control o sobre un control que en ese momento esté inhabilitado (propiedad Enabled = Falsa). En el caso de un control, el evento se activa cuando el usuario realiza una de las siguientes operaciones:

- Hacer click sobre un control con el botón derecho o izquierdo del ratón. En el caso de un botón de instrucciones, de un botón de selección o de un botón de opción, el evento sucede solamente al hacer click con el botón izquierdo.
- Seleccionar un registro de algún tipo de listas desplegables de las que dispone Visual Basic.
- Pulsar la barra espaciadora cuando el foco está en un botón de instrucciones, en un botón de selección o en un botón de opción.
- Pulsar la tecla Return cuando en un formulario hay un botón que tiene su propiedad Default = Verdadera. Pulsar la tecla Esc cuando en un formulario hay un botón que tiene su propiedad Cancel = Verdadera.
- Pulsar una combinación de teclas aceleradoras (Alt + otra tecla, como por ejemplo cuando se despliega el menú File de Word con Alt+F) definidas para activar un determinado control de un formulario.

También se puede activar el evento "Click" desde código realizando una de las siguientes operaciones:

- Hacer que la propiedad Value de un botón de instrucciones sea Verdadera.
- Hacer que la propiedad Value de un botón de opción sea Verdadera.
- Modificar la propiedad Value de un botón de selección.

El evento "DobleClick" ocurre al hacer click dos veces seguidas sobre un control o formulario con el botón izquierdo del ratón.

El evento "MouseDown" ocurre cuando el usuario pulsa cualquiera de los botones del ratón, mientras que el evento "MouseUp" sucede al soltar un botón que había sido pulsado. El evento "MouseMove" ocurre al mover el ratón sobre un control o formulario.

El evento "DragOver" sucede mientras se está arrastrando un objeto sobre un control.

Suele utilizarse para variar la forma del cursor que se mueve con el ratón, dependiendo de si el objeto sobre el que se encuentra el cursor en ese momento es válido para soltar o no. El evento "DragDrop" ocurre al concluir una operación de arrastrar y soltar. El evento "OleDragOver" es igual que el "DragOver" y requiere de los siguientes argumentos:

```
Private Sub TreeView1_OLEDragOver( _
  Data As MSComctlLib.DataObject, _
  Effect As Long, _
  Button As Integer, _
  Shift As Integer, _
  x As Single, y As Single, _
  State As Integer)
```

Donde Data, Effect, Button, Shift son los objetos que están siendo arrastrados, x e y indican la posición del objeto arrastrado dentro del sistema de coordenadas del objeto sobre el que se está arrastrando y State, que vale 0, 1 ó 2 según esté entrando, saliendo o permaneciendo dentro del mismo objeto, respectivamente.

La propiedad DragMode, puede tomar dos valores (vbManual y vbAutomatic). Esta constante determina cómo comienza una operación de arrastre de un objeto. En modo manual se debe comenzar llamando al método Drag para el objeto a arrastrar. En modo automático basta con hacer click sobre el objeto a arrastrar.

CAPÍTULO 4: APRENDIZAJE DE MATLAB CON VISUAL BASIC

4.1. Introducción

En este capítulo se detallará la parte central de este Proyecto fin de carrera. Como ya se mencionó, se trata de realizar una aplicación desarrollada en Visual Basic que utilice las instrucciones y funciones de Matlab.

En los siguientes apartados se detallarán los desarrollos de VB y Matlab que han permitido compilar el ejecutable, resultando una aplicación atractiva para el usuario con la que adquirir de una forma rápida y sencilla los conocimientos básicos para realizar sus estudios de Ingeniería.

4.2. Aspectos generales del Proyecto Fin de Carrera

Se comienza generando un nuevo proyecto cuya extensión será .vbp, para ello se debe iniciar VB (tal como se ve en la Figura 14) y se selecciona el tipo de proyecto (Exe estándar). Una vez elegido se abre un formulario (vbMatlab.frm), que constituye el formulario principal, desde el cual se pueden configurar las propiedades del proyecto, entre ellas su nombre (proyecto.vbp). En este formulario se añaden los controles y se editan las propiedades según el uso que se quiera dar a la aplicación. Para facilitar la entrada de datos y evitar errores, se deben definir las variables antes de usarlas en el módulo. La definición de variables se realiza en la sección Declaraciones del área general, como se muestra a continuación:

```
Dim datos As String
Dim sfichero As String
```

Para poder ejecutar Matlab desde una aplicación de Visual Basic lo primero que se necesita es definir una variable de tipo objeto, denotada en Matlab de la siguiente forma:

```
Dim Matlab As Object
```

A continuación, en el evento "Load" del formulario, es decir, en el momento en que éste se carga, se crea el objeto correspondiente al programa Matlab, como se refleja en la

Figura 15, lo que permitirá la conexión VB-Matlab en el momento de su ejecución. Estas instrucciones abren Matlab ejecutado desde la línea de instrucciones, que es la forma más simple de interactuar con este software, permite introducir instrucciones y ejecutar instrucciones propias de Matlab obteniendo el resultado de sus operaciones.

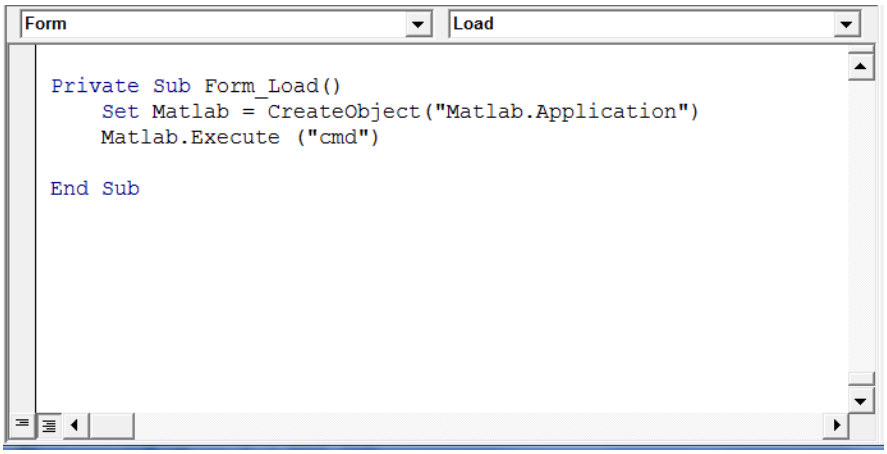


Figura 15: Evento load

En el formulario principal se utiliza el editor de menús, tal como se muestra en la Figura 16, definiendo los diferentes submenús y elementos que forman la aplicación que permitirá, de una forma muy intuitiva y visual, aprender las instrucciones básicas y las instrucciones avanzadas de Matlab.

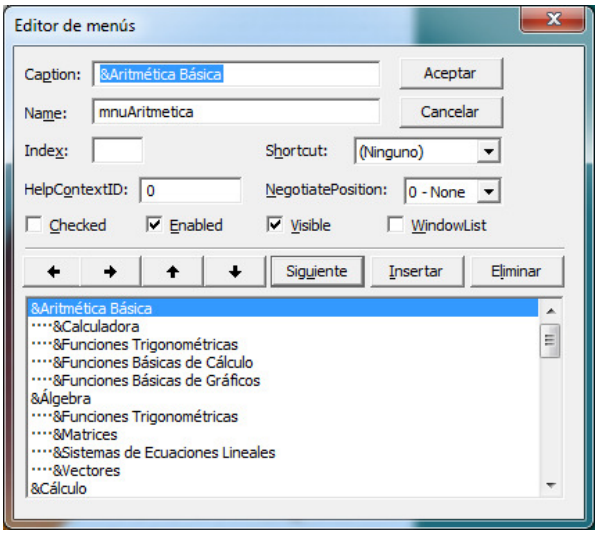


Figura 16: Cuadro de diálogo editor de menús

Este formulario está compuesto por ocho menús y a su vez cada menú contiene diferentes submenús; todos ellos se recogen en la Tabla XII:

Tabla XII: Componentes del formulario principal

Menú	Submenú	Formulario
Aritmética básica	• Calculadora • Funciones trigonométricas • Funciones básicas de cálculo • Funciones básicas de gráficos	• frmCal • frm_Ba_Al • frm_Ba_Ca • frm_Ba_Gr
Álgebra	• Funciones trigonométricas • Vectores • Matrices • Sistemas de ecuaciones lineales	• frm_Ba_Al • frmVectores • frmMa • frmSistema
Cálculo	• Funciones básicas de cálculo • Límites • Derivadas • Integrales • Ecuaciones diferenciales ordinarias (EDO's) • Ecuaciones en derivadas parciales (EDP's)	• frm_Ba_Ca • frmLimites • frmDerivada • frmIntegrales • frmEdos • frmEdps
Estadística	• Estadística descriptiva • Gráficos estadísticos - Box plot - Diagrama de barras - Diagrama de sectores - Histograma • Intervalos de confianza • Regresión lineal	• frmEstadistica • frmbp • frmBarras • frmSector • frmHisto • frmIc • frmRegresion
Gráficos e Imágenes	• Funciones básicas de gráficos • Histograma • Representación de funciones • Ajuste de curvas	• frm_Ba_Gr • Histograma • frmRep_Fun • frmAjuste

Programación	• Bucles (for, while) • Condicionales (if, switch) • Funciones	• frm_for_while • frm_if_switch • frmFunción
Aplicaciones	• Circuitos eléctricos • Vibraciones mecánicas I • Vibraciones mecánicas II	• frmCirc • frmVibrac • frmVic_Mec
Ayuda	• Acerca de • Ficheros de ayuda • Salir	• frmAcerca_de

El cuadro de diálogo del editor de menús de Visual Basic posee los siguientes campos:

- Título (*Caption*): es el nombre que se quiere mostrar en el menú (como en el caso que se puede ver en la Figura 16: Aritmética Básica). Este campo es obligatorio. El símbolo & define al carácter siguiente como tecla de acceso, para acceder a este menú, en tiempo de ejecución, se debe pulsar alt + tecla de acceso.
- Nombre (*Name*): es el nombre con el que se identifica al menú en el código, para el caso del ejemplo (Figura 16) es mnuAritmetica, se trata de otro campo obligatorio.
- Índice (*Index*): se utiliza para introducir un número y formar vectores o arreglos de menús, para esta aplicación no fue necesaria su utilización.
- *HelpContextid*: se introduce un número para asociarlo a este formulario y este número se utiliza para determinar la ayuda interactiva asociada a este formulario, no se utilizó porque era prescindible en este caso.
- Acceso directo (*Shortcut)*: se puede introducir una tecla de acceso rápido para que se despliegue el menú automáticamente, en esta aplicación no es necesario.
- Comprobado (*Checked)*: se utiliza cuando se quiere que aparezca en el menú el símbolo de verificación a la izquierda, como en el caso anterior, tampoco es necesario utilizarlo.
- Habilitado (*Enabled*): es una propiedad booleana que determina si el menú está o no habilitado, por defecto está siempre activado.

- *Visible*: es una propiedad booleana que determina si el menú está o no visible.
- Lista de ventanas *(WindowList)*: se utiliza cuando se quiere desplegar un menú con la lista de ventanas abiertas.

Una vez definido el menú principal, cada uno de los submenús se utilizará para mostrar el formulario deseado mediante el método "*Show*". Se muestra en la Figura 17 el código VB con el que mostrar el formulario de funciones trigonométricas:

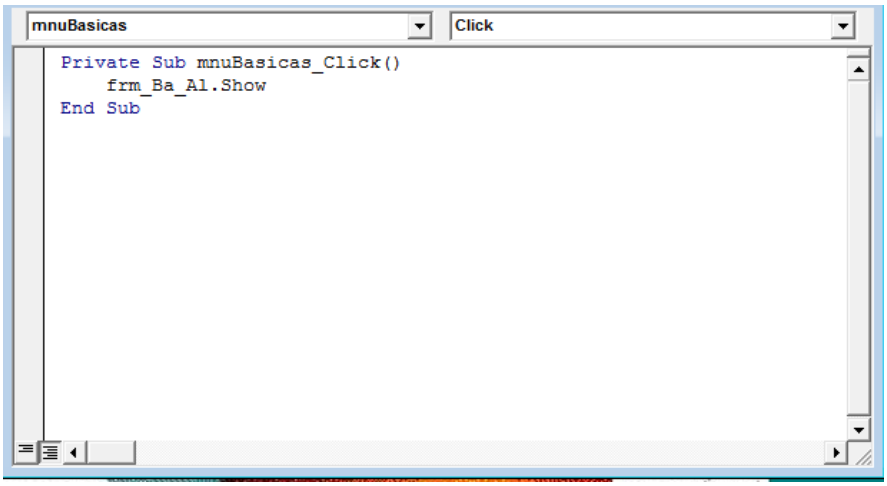


Figura 17: Evento click del menú funciones trigonométricas

En cada uno de los formularios que aparecen en este Proyecto Fin de Carrera se ha añadido un botón Salir, tal y como se muestra en la Figura 18.

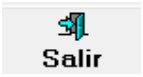


Figura 18: Botón salir

Este botón ejecuta el código programado en su evento "Click":

```
Private Sub cmdSalir_Click()
    Unload Me
End Sub
```

Se descarga de esta forma el formulario actual y se vuelve al principal.

Otras propiedades que se destacan en cada formulario son MaxButton y MinButton, que permiten visualizar o no los botones de maximizar y minimizar respectivamente y que en esta aplicación tiene el valor de false en tiempo de diseño, con lo que no se muestran dichos botones y queda la ventana con el tamaño predeterminado. Para cambiar estas propiedades en tiempo de ejecución se utilizarían las constantes **Ws_maximizebox** o

Ws_minimizebox para MaxButton y MinButton, respectivamente.

Además se les puede asignar a todos los formularios un icono en tiempo de diseño, que se mostrará en la parte superior izquierda del mismo. Para introducir dicho icono se debe seleccionar la propiedad Icon y elegir un icono cualquiera como icono de la ventana.

4.3. Aritmética básica

Para aprender a utilizar Matlab se comenzará por la parte más sencilla: la aritmética básica.

Las operaciones aritméticas que se realizarán en este apartado serán las más elementales. Para ello se dispone de un menú desplegable con las opciones detalladas en la Tabla XII y tal como se muestra en la Figura 19:

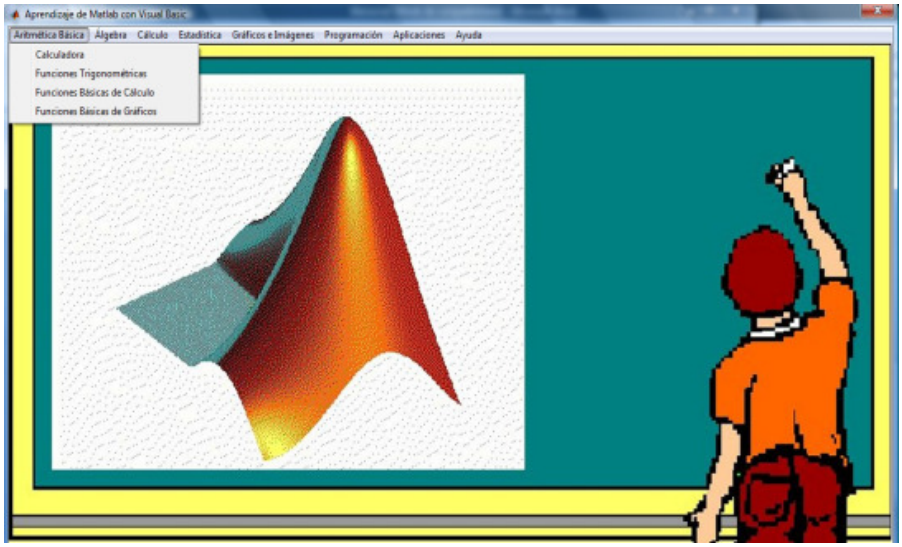


Figura 19: Menú Aritmética básica

4.3.1. Calculadora

Para desarrollar una Calculadora, se agrega al proyecto un objeto nuevo de tipo formulario y se le asigna un nombre, en este caso frmCal. Dentro de este formulario se añaden cuatro controles de tipo *frame* (detallados en la Sección §3.3), que contienen cada uno de los grupos de botones que se detallan a continuación:

- En el primer *frame* se dispone de una etiqueta de nombre lblDisplay donde se mostrarán los dígitos introducidos y el resultado de la operación seleccionada.

- En el segundo *frame* se añaden los botones de memoria con los que el usuario podrá almacenar datos en la memoria de la calculadora.
- El tercer *frame* lo componen los botones numerados del 0 al 9 con los que se operará en dicha calculadora, además del botón con el punto decimal y el signo de igualdad para obtener el resultado de las operaciones.
- En el último *frame* se incluyen los botones de las operaciones a ejecutar en este formulario: sumar, restar, multiplicar, dividir y la raíz cuadrada. Otros botones que también se muestran son: Back, Ce y +/-.

Tal como se detalla en el Manual de Usuario (MU) que acompaña a esta memoria, se comienza la ejecución de este formulario introduciendo un valor en la etiqueta lblDisplay, a continuación se selecciona la operación que se desea realizar, pulsando el botón correspondiente, por ejemplo, en el caso de querer calcular la raíz cuadrada de 81, se introducirá el 81 y al pulsar el botón √ se obtendrá el resultado, en este caso 9, tal y como se observa en la Figura 20.

Su ejecución en Matlab con los datos introducidos sería:

```
>> sqrt(81) % se ejecuta la instrucción en Matlab con el valor introducido.
   ans =
         9
```

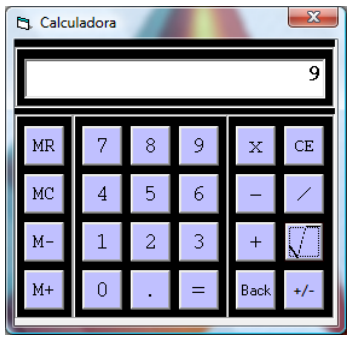


Figura 20: Ejemplo Calculadora

4.3.2. Funciones trigonométricas

Para conocer las funciones trigonométricas en Matlab, se ha añadido al proyecto de VB otro objeto de tipo formulario, asignándole el nombre frm_Ba_Al. Dentro de este formulario (ver Figura 21) se añaden los botones correspondientes a cada una de las operaciones trigonométricas que se quieren obtener: Seno, Coseno, Tangente, Seno

hiperbólico, Coseno hiperbólico, etc. de un ángulo dado, que se denota por *x*.

En la parte central del formulario se añaden otros componentes sin más que arrastrarlos desde el cuadro de herramientas:

- Un objeto de tipo *frame* en el que se incluyen las etiquetas informativas de que los ángulos van en radianes y que se debe introducir un ángulo del que se calcularán las razones trigonométricas.
- También se añade al mismo *frame* una caja de texto (txtPrimer) donde introducir dicho ángulo en radianes.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido en Matlab.
- Otra etiqueta donde se muestra la instrucción de Matlab que obtiene la razón trigonométrica buscada.
- En la parte inferior del formulario se agrega un *picturebox* que inicialmente solo muestra un dibujo, donde posteriormente se representará la gráfica correspondiente a la función obtenida.

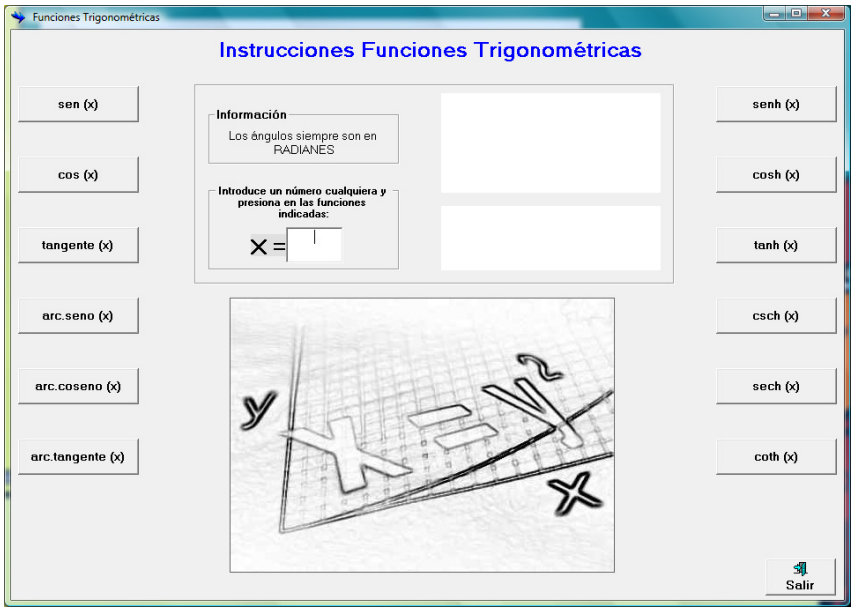


Figura 21: Formulario funciones trigonométricas

Para comenzar la ejecución de esta pantalla se define un valor cualquiera para la variable *x*, introduciendo un número en la caja de texto habilitada para ello, a continuación se seleccionará la función que se desea evaluar, pulsando el botón

correspondiente. En el caso de querer calcular, por ejemplo, el coseno de 45, se introducirá, *x* = pi/4 (en radianes, tal como se ha comentado en la Sección §2.3) y al pulsar el botón cos(x) se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdcos_Click()
    ' Se define la variable con formato de Matlab
     datos = "X = " & TxtPrimer.Text & ""
     lblInfo.Caption = "Coseno" & Chr(13) & Chr(13) & "Su código en Matlab es:" &
     Chr(13) & "cos(X)"
    ' Se ejecuta la definición de la variable en Matlab
     Matlab.Execute (datos)
    ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute("cos(X)")
    ' Se asigna el título a la etiqueta
     LblResultado.Caption = sfichero
     pcture_algebra.Picture=LoadPicture("C:\proyecto\cos.jpg")
End Sub
```

A continuación se muestran los datos que llegan a Matlab y su ejecución, a partir del código programado:

```
>> X = pi/4 % valor introducido en la variable.
>> cos(pi/4) % instrucción que se ejecuta en Matlab.
```

La siguiente ejecución en VB muestra el resultado de dicha operación en la etiqueta lblResultado, así como el código de Matlab en la etiqueta lblInfo. La gráfica se dibuja en el *picturebox* situado en la parte inferior, tal y como se observa en la Figura 22.

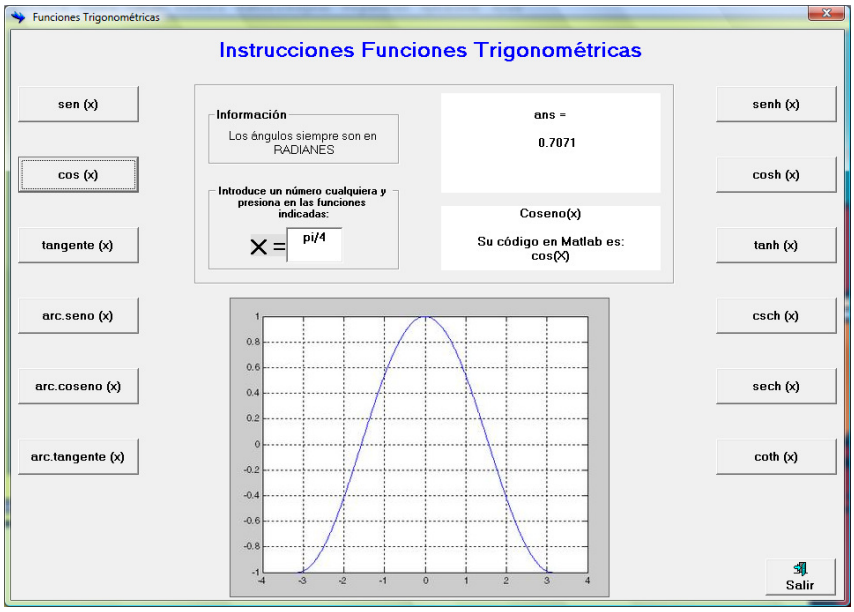


Figura 22: Ejemplo de funciones trigonométricas

4.3.3. Funciones básicas de cálculo

Se agrega un objeto nuevo de tipo formulario con el nombre frm_Ba_Ca. Dentro de este formulario se añaden los botones correspondientes a cada una de las funciones básicas que se quieren aprender: Parte real, Imaginaria, Conjugado, Valor absoluto, Raíz cuadrada, Logaritmos,…. de un valor dado, que denotaremos por A.

Se dispone de otros botones como son Límites, Derivadas e Integrales, que mediante el método "*Show*", se mostrarán en otro formulario. Se detallarán en la Sección §4.5.

En la parte central del formulario se añaden componentes sin más que arrastrarlos desde el cuadro de herramientas:

- Un objeto de tipo *frame* en el que se incluye una etiqueta para saber dónde se debe introducir un valor cualquiera para calcular las funciones deseadas.
- También se añade al mismo *frame* una caja de texto (TxtPrimer) donde se introducirá el valor comentado anteriormente.
- Una etiqueta de nombre LblResultado donde se muestra el resultado obtenido en Matlab.
- Otra etiqueta donde se muestra la instrucción de Matlab que obtiene el valor de la función buscada.
- En la parte inferior del formulario se agrega un *picturebox* que solo contiene un dibujo.

Se comienza la ejecución de este formulario de la misma forma que en el formulario anterior de funciones trigonométricas, definiendo un valor cualquiera para la variable *A*. Se introducirá un número en la casilla habilitada para ello (caja de texto TxtPrimer), a continuación se seleccionará la operación que se desea realizar, pulsando el botón correspondiente, en el caso de querer calcular el valor absoluto de -5, por ejemplo, se introduce *A* = -5 y al pulsar el botón abs se ejecuta el código programado en el evento click del botón:

```
Private Sub cmd_abs_Click()
    ' Se define la variable con formato de Matlab
     datos = "A = " & TxtPrimer.Text & ""
     lblInfo.Caption = "Valor absoluto o magnitud compleja" & Chr(13) & Chr(13) &
     "Su código en Matlab es:" & Chr(13) & "abs(A)"
```

```
    ' Se ejecuta la definición de la variable en Matlab
      Matlab.Execute (datos)
    ' Se define la variable con la instrucción de Matlab
      sfichero = Matlab.Execute("abs(A)")
    ' Se asigna el título a la etiqueta
      LblResultado.Caption = sfichero
End Sub
```

De la misma forma que en el formulario frm_Ba_Al, los datos que llegan a Matlab para su ejecución son:

```
>> A = -5 % valor introducido en la variable.
>> abs(-5) % instrucción que se ejecuta en Matlab.
```

La ejecución final muestra el resultado de dicha operación en la etiqueta lblResultado y el código de Matlab se indica en la etiqueta lblInfo que permite aprender dichos instrucciones, como se observa en la Figura 23.

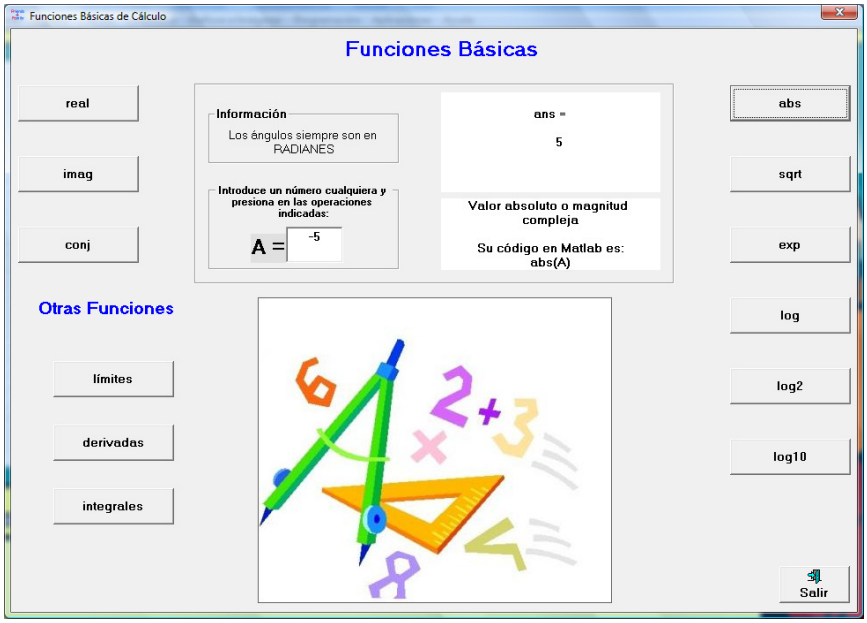


Figura 23: Ejemplo funciones de cálculo

4.3.4. Funciones básicas de gráficos

Se agrega un objeto nuevo al proyecto, de tipo formulario y se le llama frm_Ba_Gr.

Dentro de este formulario (ver Figura 25) se añaden los controles siguientes:

- Dos objetos de tipo *frame* con los títulos "Gráficos en 2D" y "Gráficos en 3D" respectivamente, donde añadimos los botones correspondientes a cada una de las

instrucciones: plot, linspace, grid, axis, hold, text, quiver, mesgrid, plot3, surf y contour con las que vamos a representar dichos gráficos.

- También se añaden en estos *frames* unas etiquetas, una por cada uno de los botones, conteniendo la información sobre las instrucciones de Matlab que se ejecutan en cada click de botón.
- Otra etiqueta lblCodigo que contiene las instrucciones que definen el rango de la función y la función a representar.
- En la parte superior derecha del formulario se agrega un *picturebox* que solo muestra inicialmente una imagen, en el que se mostrará la gráfica correspondiente a la instrucción que se ejecute.

Para desarrollar este formulario, se define un fichero adicional (ver Figura 24) en el que se guarda el código correspondiente a la ejecución del instrucciones seleccionado en el botón que se pulse.

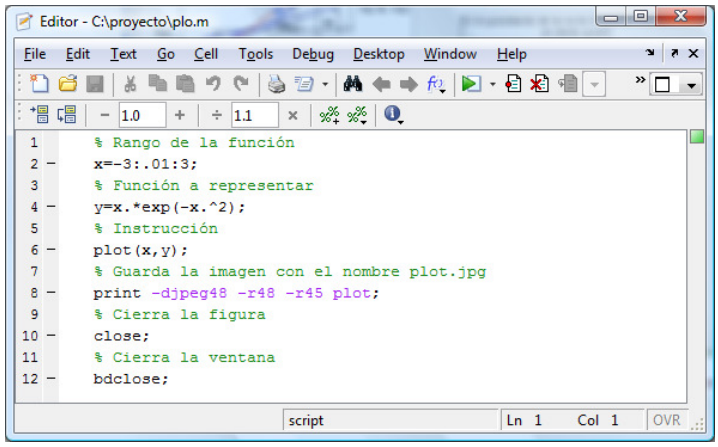


Figura 24: Archivo plo.m

Por ejemplo, en el caso de presionar el botón plot, se ejecuta el código programado en el evento "Click" de dicho botón, en el que se incluye la llamada al archivo plo.m.

```
Private Sub cmd_plot_Click()
    lblInfo.Caption = "Dibuja gráficas de la forma y=f(x)"
    lblCodigo.Caption = "Para representar una función en Matlab definimos:" &
    Chr(13) & Chr(13) & "Rango: x=-3:.01:3;" & Chr(13) & "Función a Representar:
    y=x.*exp(-x.^2);" & Chr(13) & Chr(13) & "Instrucciones: plot(x,y);"
  ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto")
  ' Se define la variable con la instrucción de Matlab
    sfichero = Matlab.Execute("plo")
```

```
    ' Se asigna la ruta para cargar la imagen
      Picture1.Picture=LoadPicture("C:\proyecto\plot.jpg")
End Sub
```

En algunas de las etiquetas utilizadas se rellena la propiedad Caption en tiempo de diseño, pero otras veces, como el caso de la etiqueta de código de este formulario (lblCodigo), se hace en tiempo de ejecución, puesto que de otro modo no tendrían valor las variables utilizadas.

Esta ejecución está programada para que muestre en la etiqueta lblInfo una pequeña explicación de lo que realiza el instrucciones de Matlab que se ejecuta, en la etiqueta lblCodigo se recoge la información correspondiente a la gráfica dibujada.

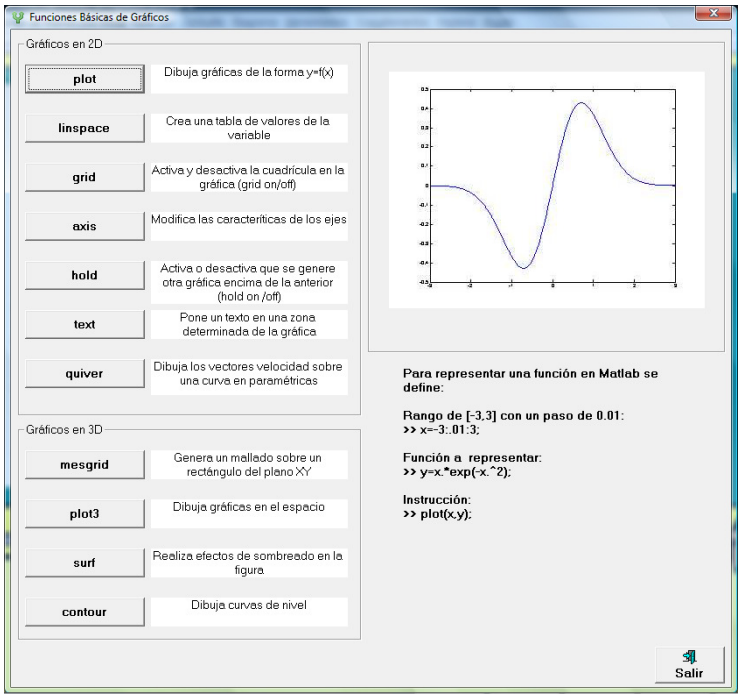


Figura 25: Formulario de funciones básicas de gráficos

4.4. Álgebra

En la parte correspondiente al aprendizaje de Álgebra se trabaja además de con la parte trigonométrica ya vista en la Sección §4.3.2, con vectores, matrices y resolución de sistemas de ecuaciones lineales, es decir, el menú que se definió en la sección §4.2 como Álgebra.

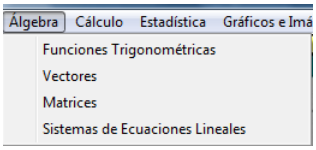


Figura 26: Menú álgebra

4.4.1. Vectores

Se agrega otro formulario frmVectores, dentro del que se añaden los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen las etiquetas informativas de cómo se introducen los vectores, con o sin corchetes, separados por comas, espacios o punto y coma, para poder operar con ellos, o que hay que introducir un valor para estos vectores con los que se calculará seleccionando el botón correspondiente, que también contiene este *frame,* su Suma, Resta, Multiplicación o División (+, -, *, /).
- También se añaden al mismo *frame* dos cajas de texto (txt_V1 y txt_V2) donde introducir dichos vectores.
- Otro objeto de tipo *frame* que contiene una etiqueta de nombre LblResultado donde se muestra el resultado obtenido de Matlab.
- En este último *frame* se incluye también otra etiqueta donde se mostrará la instrucción de Matlab con la que se obtienen las operaciones seleccionadas.

Para comenzar la ejecución en este formulario se define un valor cualquiera para los vectores, introduciendo dichos valores en las casilla habilitadas para ello (cajas de texto txt_V1 y txt_V2), a continuación se selecciona la operación que se desea realizar, pulsando el botón correspondiente, por ejemplo, en el caso de querer calcular la suma de dos vectores cualesquiera, se introduce el primer vector (a = [1 2 3]) y luego el segundo vector (b = [2 4 6]) y al pulsar el botón Suma se ejecuta el código programado en el botón:

```
Private Sub cmd_Suma_Click()
    ' Se definen las variables con el contenido de las cajas de texto en formato Matlab
     datos = "a=[" & txt_V1.Text & "]"
     datos1 = "b=[" & txt_V2.Text & "]"
     lblDetalle.Caption = "Su código en Matlab es:" & Chr(13) & "a+b" & Chr(13) &
     Chr(13) & "Donde a es el primer vector y b el segundo vector"
    ' Se ejecuta la definición de las variables en Matlab
```

```
        Matlab.Execute (datos)
        Matlab.Execute (datos1)
    ' Se define la variable con la instrucción de Matlab
        sfichero = Matlab.Execute("a+b")
    ' Se asigna el título a la etiqueta
        LblResultado = sfichero
End Sub
```

Después de introducir los datos, se procede a su ejecución en Matlab de la siguiente forma:

```
>> a = [1 2 3] % valor introducido en la variable a.
>> b = [2 4 6] % valor introducido en la variable b.
>> a+b % instrucción que se ejecuta en Matlab.
```

La ejecución obtiene el resultado de dicha operación que se muestra en la etiqueta LblResultado, el código de Matlab junto con una pequeña explicación de su instrucción se detalla en la etiqueta lblDetalle, tal y como se aprecia en la Figura 27.

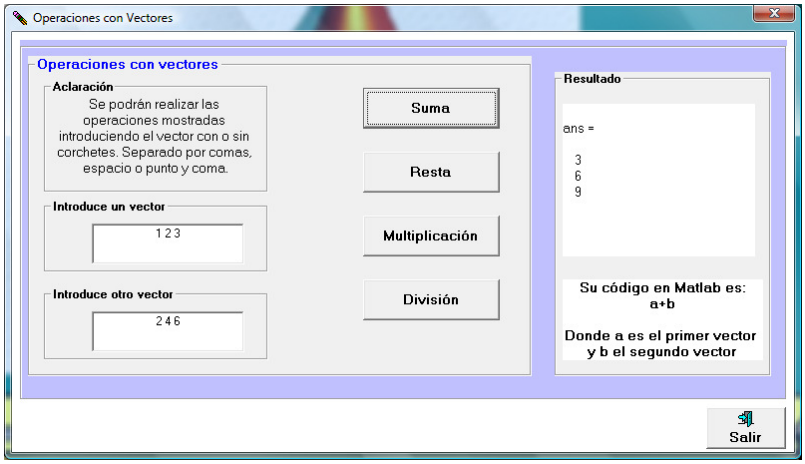


Figura 27: Ejemplo suma de vectores

4.4.2. Matrices

Las matrices son una parte importante en Matlab, de hecho, tal como se comentó en la introducción (Sección §2.1) el nombre de este programa proviene de *Matrix Laboratory*. Para operar con ellas, se agrega al proyecto un objeto nuevo de tipo formulario con el nombre frmMa.

Dentro de este formulario se añaden los siguientes controles:

- Un objeto de tipo *frame* en el que se incluyen los botones correspondientes a cada una de las operaciones que se pueden realizar con matrices: Determinante, Rango, Matriz inversa y Matriz traspuesta.

- Cajas de texto donde se introducirán los valores de la matriz con la que se desea operar.
- Una etiqueta de nombre lblResultado donde se muestra el resultado que devuelve Matlab.
- Otra etiqueta donde se muestra la instrucción de Matlab que calcula la operación indicada.

Para comenzar la ejecución de este formulario se define un valor cualquiera para la matriz *A*, se introducen los números en las casillas habilitadas para ello (cajas de texto txt_a, txt_b,…), a continuación se selecciona la operación que se desea realizar, pulsando el botón correspondiente. Por ejemplo, en el caso de querer calcular el determinante de la matriz $A = \begin{pmatrix} 3 & 2 & 1 \\ 0 & 2 & -5 \\ -2 & 1 & 4 \end{pmatrix}$, se introducen los datos de la misma forma que se comentó en la Sección §2.3 y se pulsa el botón Determinante, que ejecuta el código programado en el evento ”Click” del botón:

```
Private Sub cmdDet_Click()
    lblDetalle.Caption = "Aplicamos la regla de Sarrus para el cálculo del
    determinante de una matriz 3 × 3." & Chr(13) & Chr(13) & "Su código en Matlab
    es:" & Chr(13) & "Determinante = det (A)"
  ' Se define la variable con formato de Matlab
    datos = "A = ["
    & txt_a.Text & Chr(32) & txt_d.Text & Chr(32) & txt_g.Text & Chr(32) & ";" &
    txt_b.Text & Chr(32) & txt_e.Text & Chr(32) & txt_h.Text & Chr(32) & ";" &
    txt_c.Text & Chr(32) & txt_f.Text & Chr(32) & txt_i.Text & "]"
  ' Se ejecuta la definición de la variable en Matlab
    Matlab.Execute (datos)
  ' Se define la variable con la instrucción de Matlab
    sfichero = Matlab.Execute("Determinante = det(A)")
  ' Se asigna el título a la etiqueta
    lblResultado.Caption=sfichero
End Sub
```

Una vez introducidos los datos en el formulario, su ejecución en Matlab es la siguiente:

```
>> A = [3 2 1; 0 2 -5; -2, 1 4] % valor introducido en la variable.
>> det(A) % instrucción que se ejecuta en Matlab.
```

La ejecución muestra el resultado de dicha operación en la etiqueta lblResultado. A su vez el código de Matlab que resuelve cada operación aparece en la etiqueta lblDetalle, como se observa en la Figura 28.

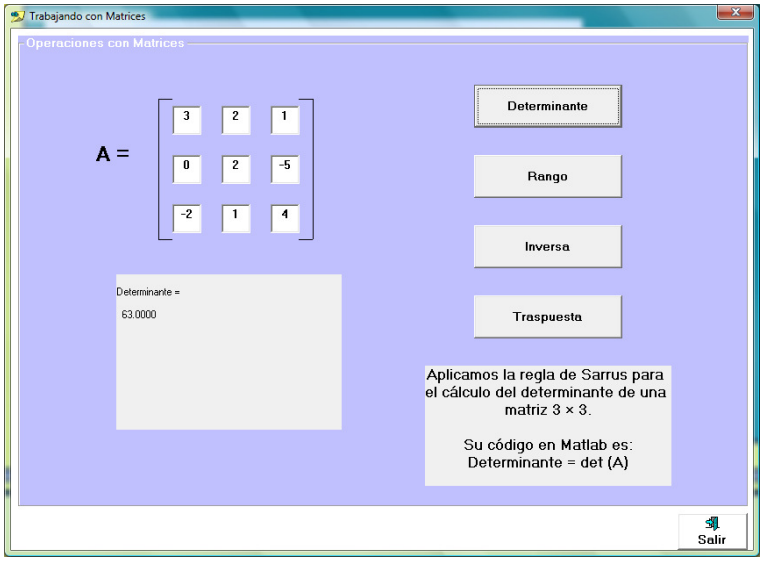


Figura 28: Operaciones con matrices

4.4.3. Sistemas de ecuaciones lineales

Se agrega un nuevo formulario, frmSistema.

Dentro de este formulario se tienen los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluye una etiqueta que da título al formulario, además de un *picturebox* que explica el tipo de sistemas de ecuaciones que se resolverán en el mismo.
- Otro objeto de tipo *frame* que engloba una etiqueta que da título a esa parte del formulario, un *picturebox* que contiene el sistema de ecuaciones a resolver.
- También se incluye dentro de este *frame* otro nuevo *frame* que contiene las cajas de texto donde se introducirán los valores de la matriz con la que se desea trabajar.
- Un botón cuyo nombre es cmdEvaluar que resuelve el sistema mediante su instrucción en Matlab.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido de Matlab.

- Otra etiqueta donde se muestra la instrucción de Matlab que resuelve el sistema propuesto.

Para comenzar la ejecución en este formulario se introducen los valores del sistema de ecuaciones mostrado en pantalla, introduciendo cada número en la casilla habilitada para ello (cajas de texto txt_a, txt_b, txt_c,…), a continuación se evalúa el sistema pulsando el botón correspondiente, por ejemplo, para resolver el siguiente sistema: $\left.\begin{array}{l} 3x_1+2x_2-4x_3=2 \\ 4x_1+3x_2-7x_3=-1 \\ -x_1+6x_2+5x_3=4 \end{array}\right.$ se introducirá la matriz $A=\begin{pmatrix} 3 & 2 & -4 \\ 4 & 3 & -7 \\ -1 & 6 & 5 \end{pmatrix}$, la matriz de términos independientes $B=\begin{pmatrix} 2 \\ -1 \\ 4 \end{pmatrix}$ y al pulsar el botón Resultado se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se definen las variables con formato de Matlab
      datos = "A = [" & txt_a.Text & Chr(32) & txt_d.Text & Chr(32) & txt_g.Text &
      Chr(32) & ";" & txt_b.Text & Chr(32) & txt_e.Text & Chr(32) & txt_h.Text &
      Chr(32) & ";" & txt_c.Text & Chr(32) & txt_f.Text & Chr(32) & txt_i.Text & "]"
      datos1 = "B = [" & txt_j.Text & Chr(32) & ";" & txt_k.Text & Chr(32) & ";" &
      txt_l & Chr(32) & "]"
      lblDetalle.Caption = "Su código en Matlab es:" & Chr(13) & Chr(13) & "X=A\B o
      X=inv(A)*B"
    ' Se ejecuta la definición de las variables en Matlab
      Matlab.Execute (datos)
      Matlab.Execute (datos1)
    ' Se define la variable con la instrucción de Matlab
      sfichero = Matlab.Execute("x= A\B")
    ' Se asigna el título a la etiqueta
      LblResultado.Caption = sfichero
End Sub
```

Después de introducir los datos, se procede con su ejecución en Matlab de la siguiente forma:

```
>> A = [3 2 -4; 4 3 -7; -1, 6 5] % valor introducido en la variable A.
>> B = [2;-1; 4] % valor introducido en la variable B.
>> x=A\B % instrucción que se ejecuta en Matlab.
```

La ejecución muestra el resultado de dicha operación en la etiqueta LblResultado, así como en la etiqueta lblDetalle se indica la instrucción de Matlab que se ejecuta, tal y como se observa en la Figura 29.

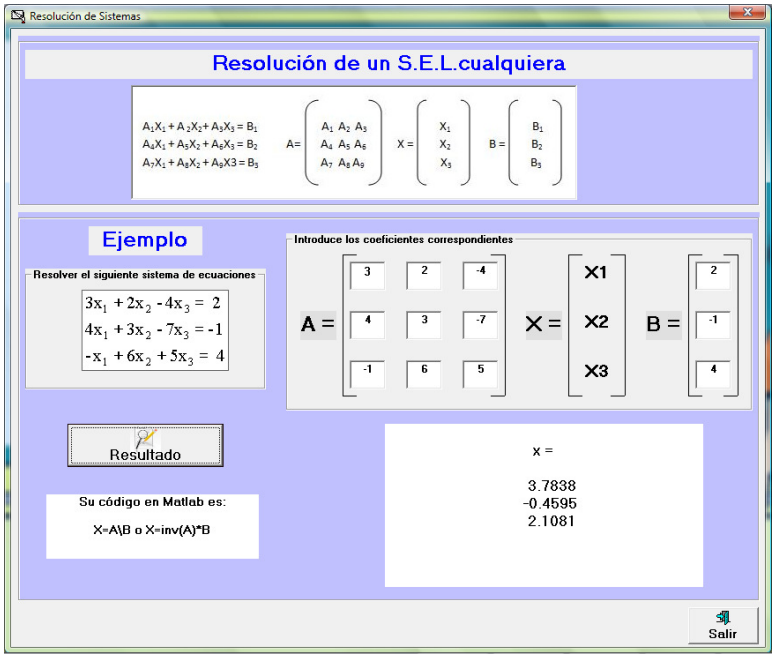


Figura 29: Resolución S.E.L

4.5. Cálculo

Dentro de este menú se recogen varias funciones que se utilizan en la mayoría de asignaturas que componen los estudios de Ingeniería y cuyo aprendizaje es fundamental para el desarrollo de sus competencias en su futuro laboral. Estas funciones, según se muestran en la Figura 30, son:

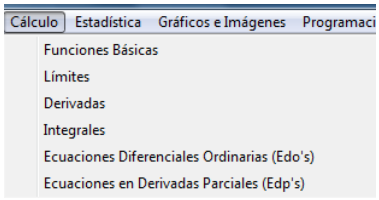


Figura 30: Menú Cálculo

4.5.1. Funciones básicas de cálculo

Que se explicaron en la Sección §4.3.3.

4.5.2. Límites

Se agrega un nuevo formulario con el nombre frmLimites. En este formulario se añaden los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define la función que se va a tratar en este formulario y un *picturebox* que muestra un dibujo estándar sobre esta función.
- En este mismo *frame* se añade un *picturebox* con un ejemplo de un límite y una etiqueta con su resolución en Matlab.
- Otro objeto de tipo *frame* que contiene una etiqueta que muestra el límite a calcular, además de una caja de texto donde se muestra el código de Matlab, al pulsar el botón Mostrar código con el que resolver dicho límite, en esta caja de texto se podrá modificar el código para calcular el límite de cualquier función.
- También se añade al mismo *frame* un botón cuyo nombre es cmdEvaluar que resuelve el sistema mediante su instrucción en Matlab.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido en Matlab.

Se comienza la ejecución de este formulario pulsando el botón Mostrar código, con lo que se muestra el código en la caja de texto, a continuación se pulsa el botón Resultado y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se define la variable con formato de Matlab
     datos = "" & txt_fun.Text & ""
    ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute(datos)
    ' Se asigna el título a la etiqueta
     lblresul.Caption = sfichero
End Sub
```

Su ejecución muestra el resultado de dicha operación en la etiqueta lblResultado, tal y como se observa en la Figura 31.

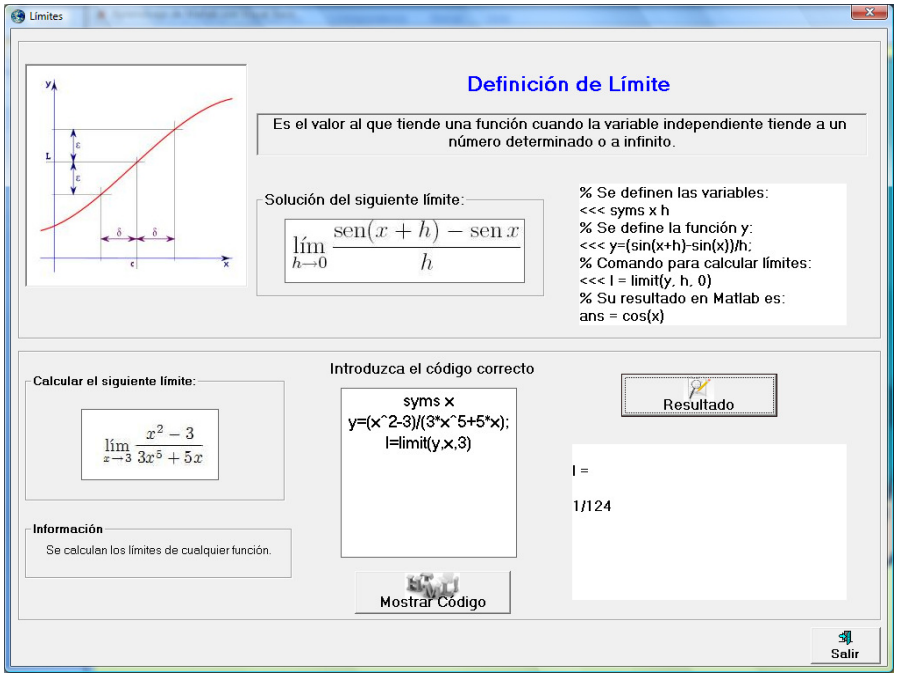


Figura 31: Cálculo de límite

4.5.3. Derivadas

Se agrega un objeto nuevo de tipo formulario y le damos el nombre relacionado con las funciones que vamos a calcular, en este caso frmDerivada. Este formulario está compuesto por los siguientes controles:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define la función que se va a tratar en este formulario y un *picturebox* que muestra un dibujo estándar sobre esta función.
- Otro objeto de tipo *frame* que contiene una etiqueta que muestra la función a derivar, además de una caja de texto donde se muestra el código de Matlab con el que calcular la derivada seleccionada, en esta caja de texto se podrá modificar el código para calcular la derivada de cualquier función.
- También se añaden al mismo *frame* tres botones de selección con los que se podrá elegir entre calcular la Derivada Primera, Segunda o Tercera y una etiqueta que informa sobre la opción de calcular cualquier derivada de otra función.
- Un botón cuyo nombre es cmdEvaluar que calcula la derivada de la función mediante su instrucción en Matlab.

- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido de Matlab.

Para comenzar la ejecución en este formulario se seleccionará el tipo de derivada a calcular que puede ser primera, segunda o tercera. A continuación se mostrará el código en la caja de texto, por ejemplo, si se quiere calcular la derivada primera de la función propuesta, se seleccionará la primera opción y al pulsar el botón Resultado se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se define la variable con formato de Matlab
     datos = "" & txt_fun.Text & ""
    ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute(datos)
   ' Se asigna el título a la etiqueta
     lblResultado.Caption = sfichero
End Sub
```

Los datos que se introducen por código para su ejecución en Matlab, se muestran de la siguiente forma:

```
% código introducido en la variable datos, que se ejecuta en Matlab.
>> "syms x; y=4*x*sin(x); d1f=diff(y); pretty (d1f)"
```

La ejecución muestra el resultado de dicha operación en la etiqueta lblResultado, tal y como se observa en la Figura 32.

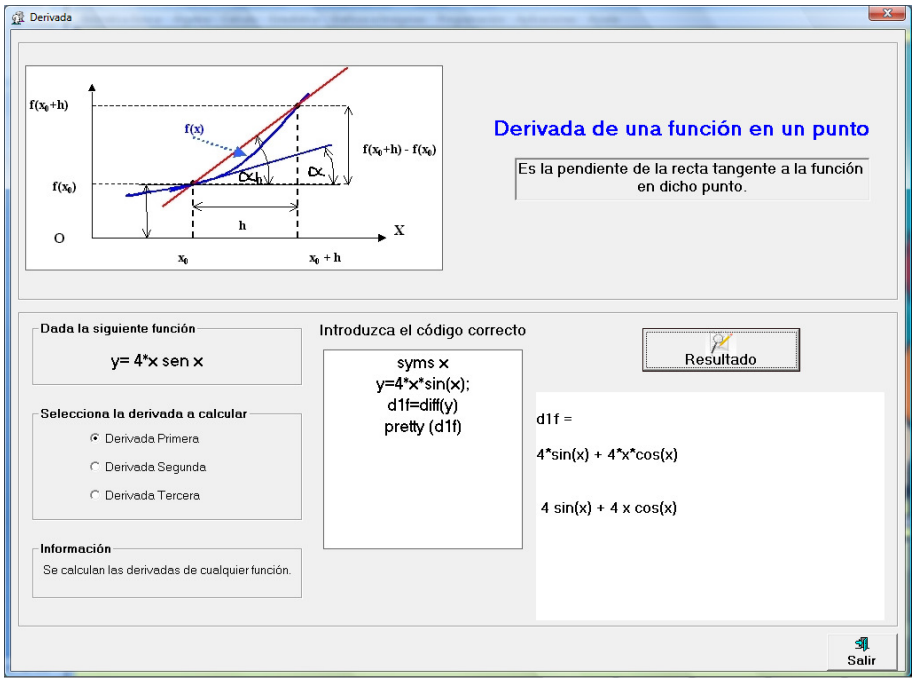


Figura 32: Cálculo de derivada primera

4.5.4. Integrales

Se agrega un objeto nuevo de tipo formulario y se le da el nombre frmIntegrales. Dentro de este formulario se añaden los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define la función que se va a tratar en este formulario y un *picturebox* que muestra un dibujo estándar sobre esta función.
- Otro objeto de tipo *frame* que contiene una etiqueta que muestra la integral a calcular, además de una caja de texto donde se muestra el código de Matlab con el que resolver el tipo de integral seleccionada, en esta caja de texto se podrá modificar el código para calcular la integral de cualquier función integrable.
- También se añaden al mismo *frame* dos botones de selección con los que se podrá elegir entre calcular la integral definida e indefinida y una etiqueta que informa sobre la opción de calcular cualquier integral de otra función integrable.
- Un botón cuyo nombre es cmdEvaluar que calcula la integral mediante su instrucción en Matlab.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido de Matlab.

Para comenzar la ejecución en este formulario se selecciona el tipo de integral a calcular, a continuación se muestra el código en la caja de texto, por ejemplo, si se quiere calcular la integral definida (similar para indefinida) de la función propuesta, se selecciona la primera opción y al pulsar el botón Resultado se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se define la variable con formato de Matlab
     datos = "" & txt_fun.Text & ""
    ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute(datos)
    ' Se asigna el título a la etiqueta con el resultado
     lblresul.Caption = sfichero
End Sub
```

La ejecución muestra el resultado de dicha operación en la etiqueta lblResultado, tal y como se observa en la Figura 33.

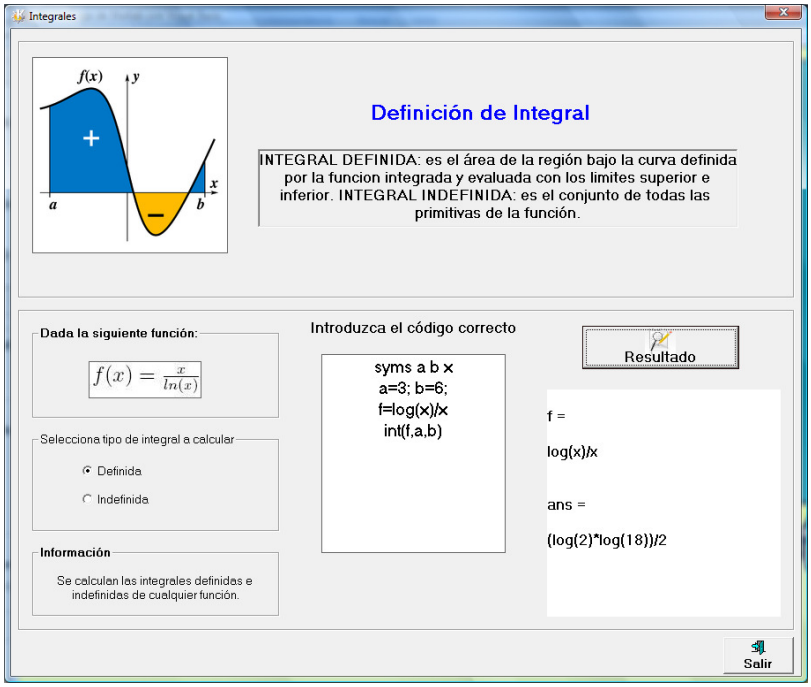


Figura 33: Cálculo de integrales definidas

4.5.5. Ecuaciones diferenciales ordinarias

Se agrega un objeto nuevo de tipo formulario frmEdos. Se añaden a este formulario los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define el tipo de sistemas de Edos que se van a resolver en este formulario y un *picturebox* que muestra un dibujo estándar de estos sistemas, donde se dibujará la gráfica correspondiente a la función obtenida.
- Otro objeto de tipo *frame* que contiene dos *picturebox* que muestran el sistema de ecuaciones diferenciales a calcular y sus condiciones iniciales, además de una etiqueta informativa de que solo se puede resolver el ejercicio propuesto.
- También se añaden al mismo *frame* tres botones correspondientes a cada una de las opciones que se pueden realizar en este formulario: Mostrar código, Solución exacta y Solución numérica, que el usuario podrá seleccionar según su elección.
- Otra etiqueta donde se muestra el método utilizado para resolver el sistema de Edos.

- En la parte inferior del formulario se agrega un *picturebox* que muestra las instrucciones de Matlab con las que se obtienen las soluciones exacta y numérica del sistema de Edos dado.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido de Matlab.

Se comienza la ejecución de este formulario pulsando el botón Mostrar código, que muestra el código que se ejecuta en Matlab y que aparece en el *picturebox*, a continuación se pulsa el botón Solución exacta con el que se calculan los valores de u y v. Por último para calcular la solución aproximada del sistema propuesto, se pulsa el botón Solución numérica y se ejecuta el código programado en el evento "Click" del botón, que se detalla más adelante.

Como en la Sección §4.3.4, es necesario definir con anterioridad un archivo con el nombre gredo.m, tal y como se observa en la Figura 34.

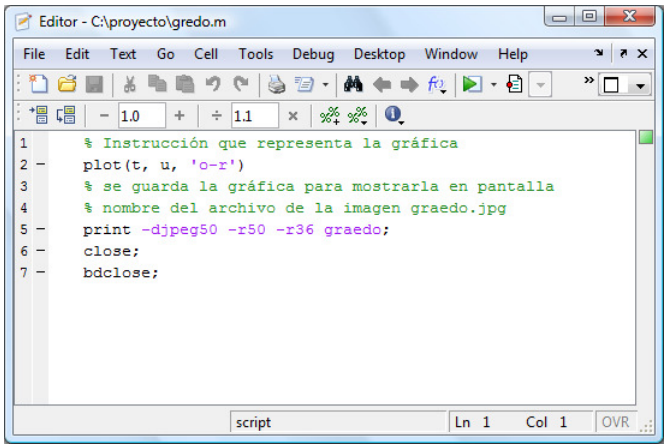


Figura 34: Archivo gredo.m

El código implementado en el botón Solución numérica, que utiliza el archivo gredo.m, se muestra a continuación:

```
Private Sub cmdnumerica_Click()
    lblInfo.Caption = "Método utilizado para su resolución:" & Chr(13) & Chr(13) &
    "Runge-Kutta"
  ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto")
  ' Se definen las variable con las instrucciones de Matlab
    sfichero = Matlab.Execute("[t,u] = ode45('f',[0 1],[0.2; 0.05]);")
    sfichero2 = Matlab.Execute("M= [t u]")
    sfichero3 = Matlab.Execute("gredo;")
  ' Se asigna el título a la etiqueta
```

```
        lblResultado.Caption = sfichero2
        pcture_gra.Picture=LoadPicture("C:\proyecto\graedo.jpg")
    End Sub
```

Los datos se introducen por código para su ejecución en Matlab:

```
% instrucción que se ejecuta en Matlab.
>> "[t,u] = ode45('f',[0 1],[0.2; 0.05]"
% instrucción que se ejecuta en Matlab para mostrar el resultado anterior.
>> "M = [t,u]"
% se ejecutan las instrucciones del archivo gredo.m en Matlab.
>> gredo
```

El resultado de ejecutar dicha operación se muestra en la etiqueta lblResultado. En la etiqueta lblInfo aparece el método utilizado para resolver el sistema de ecuaciones diferenciales dado y se dibuja la gráfica en el *picturebox* situado en la parte superior donde inicialmente teníamos el dibujo estándar, tal y como se observa en la Figura 35.

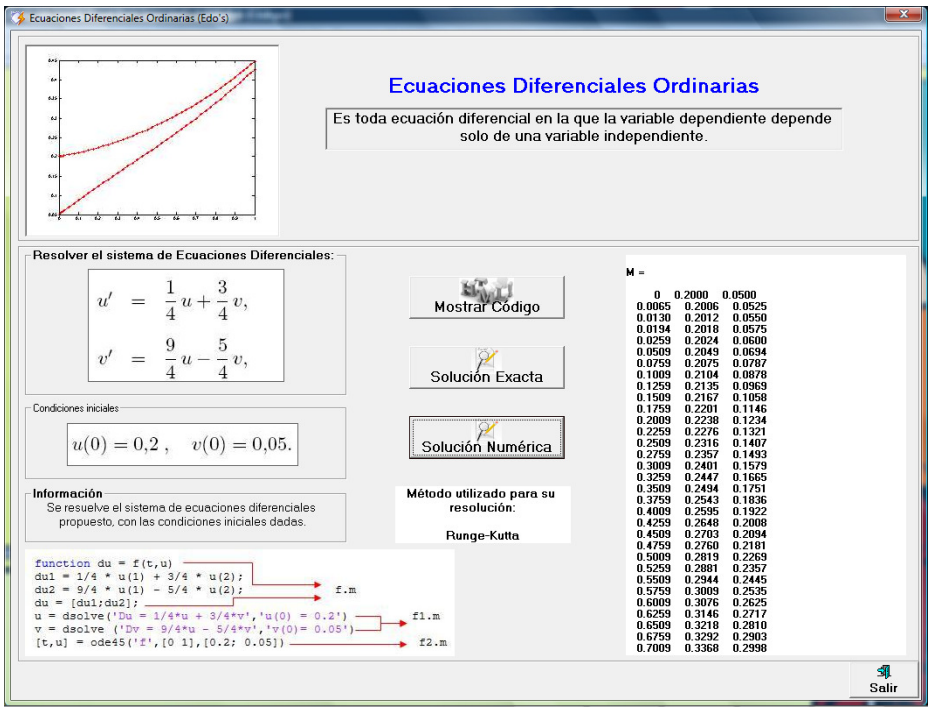


Figura 35: Solución numérica de EDO's

4.5.6. Ecuaciones en derivadas parciales

Se agrega un formulario y se le asigna el nombre frmEdps. Dentro de este formulario se tienen los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define la EDP que se va a analizar en este formulario y un *picturebox* que muestra un dibujo estándar sobre las Edp's.
- Otro objeto de tipo *frame* que contiene un *picturebox* que muestra la ecuación a calcular, además de una caja de texto donde se muestra el código de Matlab con el que resolver dicha ecuación, en esta caja de texto se podrá modificar el código para resolver cualquier Edp's que cumpla los mismos requisitos.
- También se añaden al mismo *frame* dos botones correspondientes a cada una de las opciones que se pueden realizar en este formulario: Mostrar código y calcular la Solución exacta que el usuario podrá seleccionar según su elección.
- Un *picturebox* que muestra el código que aparece también en la caja de texto que ya hemos comentado anteriormente.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido en Matlab.

Se comienza la ejecución de este formulario pulsando el botón Mostrar código, con lo que se muestra el código en la caja de texto, a continuación se pulsa el botón Solución exacta y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdExacta_Click()
    ' Se define la variable con formato de Matlab
     datos = "" & txt_fun.Text & ""
    ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute(datos)
    ' Se asigna el título a la etiqueta
     lblResultado.Caption = sfichero
End Sub
```

De nuevo los datos que se introducen por código para su ejecución, se muestran en Matlab de la siguiente forma:

```
% código introducido en la variable datos, que se ejecuta en Matlab.
>> "syms z x a y c; dsolve ('D2z + c*y', 'x');"
```

La ejecución de esta operación muestra el resultado de la misma en la etiqueta lblResultado, tal y como se observa en la Figura 36.

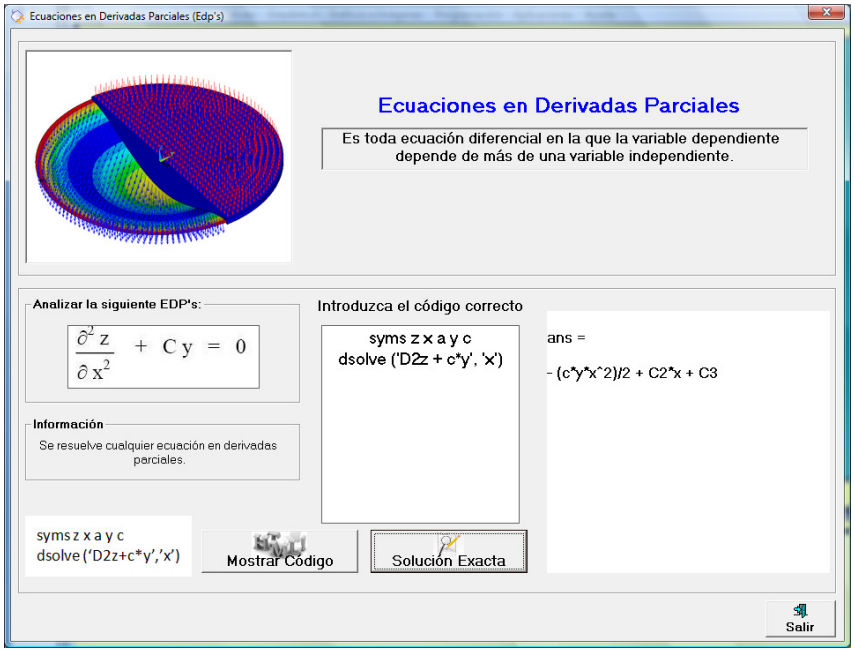


Figura 36: Solución exacta de EDP's

4.6. Estadística

Dentro de este menú se estudian diferentes contenidos de estadística descriptiva e inferencia estadística, que son las dos ramas en que se divide la estadística aplicada. Estos contenidos, según se muestran en la Figura 37, son:

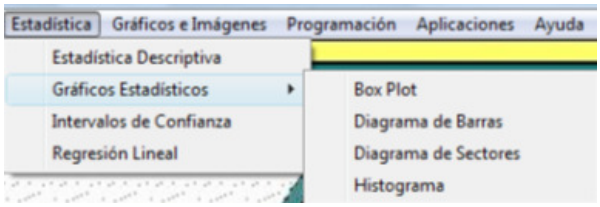


Figura 37: Menú Estadística

4.6.1. Estadística descriptiva

Se agrega un objeto nuevo al proyecto, de tipo formulario y se le asigna el nombre frmEstadistica.

Dentro de este formulario se definen los siguientes controles:

- Un objeto de tipo *frame* en el que se incluyen dos etiquetas informativas que indican dónde se deben introducir los datos con los que queremos calcular las funciones que contiene este formulario.

- También se añaden al mismo *frame* dos cajas de texto (txtDatos1 y txtDatos2) donde introducir los datos.
- A continuación se añaden los botones correspondientes a cada una de las instrucciones: length, mean, median, range, iqr, prctile, max, min, cov y corrcoef con las que operaremos.
- También se añaden cajas de texto, una por cada uno de los botones, donde aparece el resultado de la operación seleccionada al ejecutar el click del botón.
- Otra etiqueta lblInfo donde se muestra información sobre lo que realiza la instrucción seleccionada.

Para comenzar la ejecución en este formulario se da un valor cualquiera a las dos muestras, las variables *x* e *y*, introduciendo los datos de ambas muestras en las cajas de texto txtDatos1 y txtDatos2, habilitadas para ello. A continuación se selecciona la instrucción que se desea calcular, pulsando el botón correspondiente, por ejemplo, en el caso de querer calcular la mediana de la muestra [1, 4, 8, 10, 12], se introduce x = [1, 4, 8, 10, 12] y al pulsar el botón median(x) se ejecuta el código correspondiente:

```
Private Sub cmdMediana_Click()
    ' Se define la variable con formato de Matlab
      datos1 = "x = [" & txtDatos1.Text & "]"
      lblInfo.Caption = "Mediana"
    ' Se ejecuta la definición de la variable en Matlab
      Matlab.Execute (datos1)
    ' Se define la variable con la instrucción de Matlab
      sMediana1 = Matlab.Execute("median(x)")
    ' Se asigna el título a la etiqueta
      txtMediana1.Text=LTrim(Mid(sMediana1,InStr(1,sMediana1,"=",vbTextCompare
      )+ 2))
End Sub
```

En el código anterior encontramos las siguientes instrucciones, a continuación se detalla su uso dentro del mismo:

LTrim: elimina los espacios vacíos de la parte izquierda de la cadena.

Mid: sirve para extraer partes de una cadena y cuenta con los siguientes parámetros: inicio y longitud.

InStr: se utiliza para buscar una cadena o parte de una cadena dentro de otra cadena.

La ejecución muestra el resultado de dicha operación, en la caja de texto txtMediana1, el código de Matlab en la etiqueta lblInfo, tal y como se aprecia en la Figura 38.

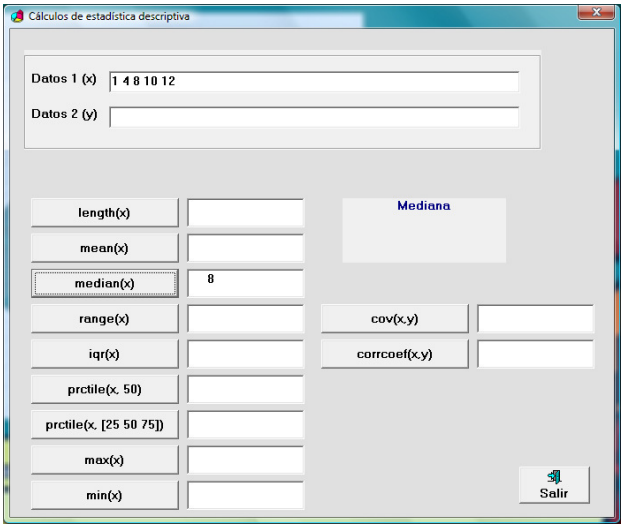


Figura 38: Cálculo de parámetros en estadística descriptiva

4.6.2. Gráficos estadísticos

Permiten representar gráficamente los datos recogidos de una muestra. A continuación, se detallan los gráficos con los que cuenta esta aplicación.

4.6.2.1. Box plot

Se agrega un objeto nuevo de tipo formulario y se le asigna el nombre relacionado con el tipo de gráficos que vamos a representar, en este caso frmbp. Dentro de este formulario se añaden los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define el gráfico que se va a representar en este formulario y un control de imagen que muestra un dibujo estándar sobre este gráfico.
- También se añade un etiqueta de nombre lblCodigo donde se muestra la instrucción en Matlab para representar este tipo de gráficos.
- Otro objeto de tipo *frame* que contiene una etiqueta que muestra un conjunto de datos a representar en un box plot.
- Se añaden los botones correspondientes a cada una de las operaciones que nos interesan: Primer cuartil, Mediana y Tercer cuartil, junto con sus etiquetas, que contienen el resultado de la operación seleccionada.

- Una etiqueta donde se muestra la instrucción de Matlab que calcula el primer cuartil, mediana y tercer cuartil así como la definición de los mismos.
- Un botón cuyo nombre es cmdGrafica que representa el gráfico mediante su instrucción en Matlab.
- Un *picturebox* donde se dibuja el gráfico box plot correspondiente al conjunto de datos dados.

Para comenzar la ejecución en este formulario se selecciona cada uno de los tres botones que se muestran en pantalla, Primer cuartil, Mediana y Tercer cuartil, que son necesarios para calcular el gráfico Box plot. Una vez obtenidos estos valores, se define el archivo e2.m, tal y como se detalla en la Figura 39.

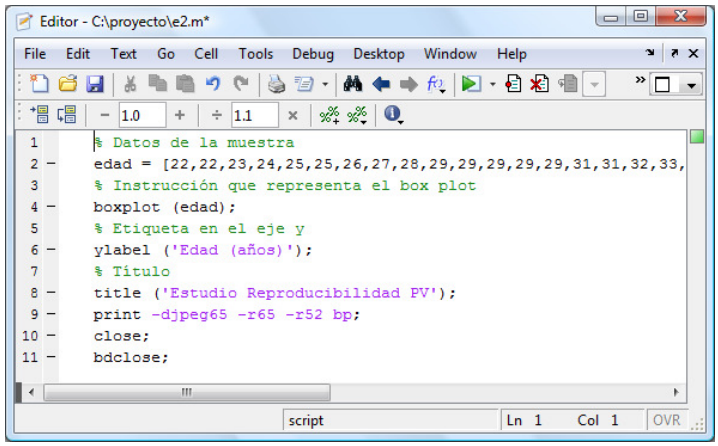


Figura 39: Archivo e2.m

Se prosigue pulsando el botón Rep. Gráfica y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdGrafica_Click()
    lblCodigo = "Su instrucción en Matlab es:" & Chr(13) & "boxplot(edad)"
    ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto")
    ' Se define la variable con la instrucción de Matlab
    sfichero = Matlab.Execute("e2;")
    pcturebox.Picture = LoadPicture("C:\proyecto\bp.jpg")
End Sub
```

Esta ejecución representa la gráfica en la caja de control situada en la parte inferior del formulario, tal y como se muestra en la Figura 40, el código de su instrucción en Matlab se muestra en la etiqueta lblCodigo.

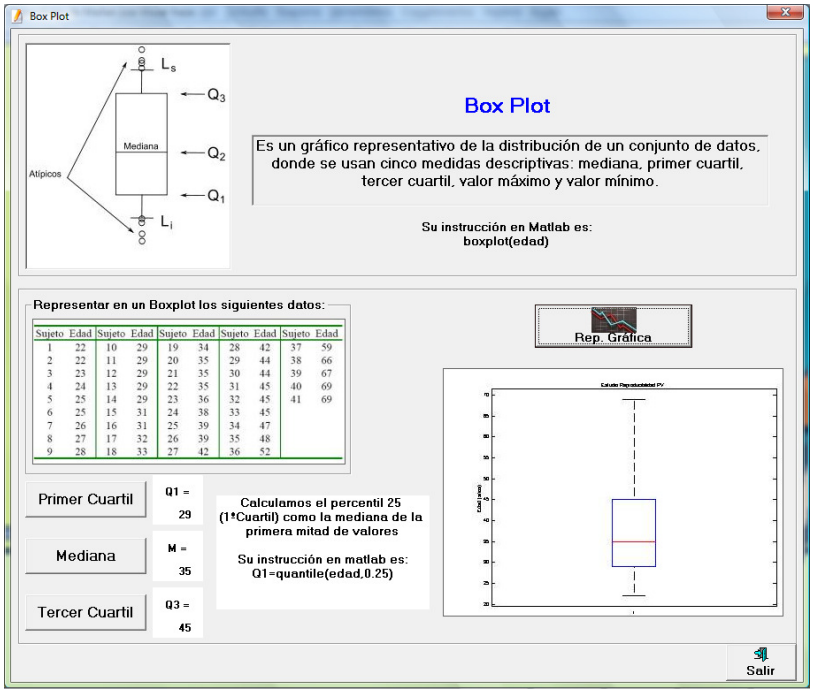


Figura 40: Gráfico box plot

4.6.2.2. Diagrama de barras

Se agrega un nuevo formulario con el nombre frmBarras. En este formulario se añaden los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define el gráfico que se va a representar en este formulario y un *picturebox* que muestra un dibujo estándar sobre este tipo de gráficos.
- También se añade una etiqueta de nombre lblCodigo donde se muestra la instrucción en Matlab para representar este tipo de gráficos.
- Otro objeto de tipo *frame* que contiene una etiqueta y un *picturebox* con el enunciado y datos que se pretenden representar, además de una caja de texto donde se muestra la instrucción que representa dicho diagrama, esta instrucción se podrá modificar para calcular cualquier diagrama de barras.
- También se añaden al mismo *frame* dos botones con los que se selecciona la opción que se quiera realizar: Mostrar Código o Rep. Gráfica.
- En la parte inferior del formulario se agrega un *picturebox* donde se representa el diagrama de barras correspondiente a los datos propuestos.

Se comienza la ejecución de este formulario pulsando en el botón Mostrar Código, con lo que se muestra el código en la caja de texto, a continuación se pulsa el botón Rep. Gráfica y se ejecuta el código programado en el evento "Click" del botón. Para desarrollar el código es necesario definir con anterioridad el archivo barras.m, tal y como se observa en la Figura 41.

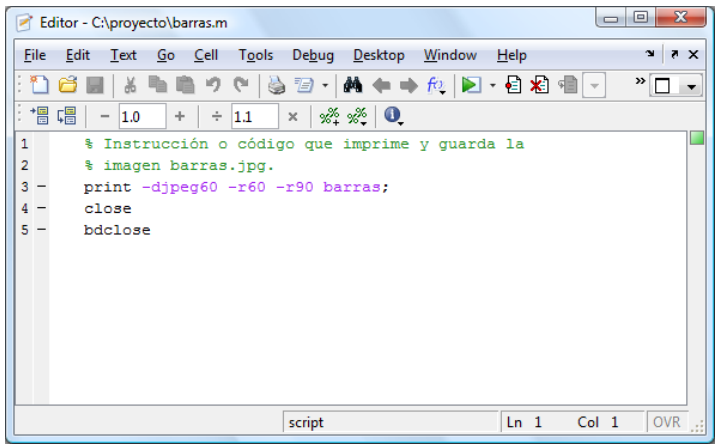


Figura 41: Archivo barras.m

El código asociado al botón Rep. Gráfica, que utiliza el archivo barras.m, se muestra a continuación:

```
Private Sub cmdGrafica_Click()
    lblCodigo = "Su instrucción en Matlab es:" & Chr(13) & " bar(X)"
  ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto ")
  ' Se define la variable con formato de Matlab
    datos = "" & txt_fun.Text & ""
  ' Se ejecuta la definición de la variable en Matlab
    sfichero = Matlab.Execute (datos)
  ' Se define la variable con la instrucción de Matlab
    sfichero2 = Matlab.Execute("barras;")
    pcturebox.Picture = LoadPicture("C:\proyecto\barras.jpg")
End Sub
```

Una vez que se introducen los datos, su ejecución en Matlab es la siguiente:

```
% valor introducido en la variable datos.
>> "x = [13 13 8 5 1]; subplot(2,3,1),bar(x);"
% se ejecutan las instrucciones del archivo barras.m en Matlab.
>> barras
```

La ejecución representa la gráfica en el *picturebox* situado en la parte inferior del formulario, tal y como se observa en la Figura 42, el código de su instrucción en Matlab se muestra en la etiqueta lblCodigo.

Figura 42: Diagrama de barras

4.6.2.3. Diagrama de sectores

Se agrega un objeto nuevo a este proyecto, frmSector. En este formulario se añaden los mismos componentes que en el diagrama de barras que se explicó en la Sección §4.6.2.2:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define el gráfico que se va a representar en este formulario y un *picturebox* que muestra un dibujo estándar sobre este tipo de gráficos.
- También se añade una etiqueta de nombre lblCodigo donde se muestra la instrucción en Matlab para representar este tipo de gráficos.
- Otro objeto de tipo *frame* que contiene una etiqueta y un *picturebox* con el enunciado y datos que se pretenden representar, además de una caja de texto donde se muestra la instrucción que representa dicho diagrama, esta instrucción se podrá modificar para calcular cualquier diagrama de este tipo.
- También se añaden al mismo *frame* dos botones con los que se selecciona la opción que se quiera realizar: Mostrar Código o Rep. Gráfica.

- En la parte inferior del formulario se agrega un *picturebox* donde se representa el diagrama de barras correspondiente a los datos propuestos.

Se comienza la ejecución de este formulario pulsando en el botón Mostrar Código, con lo que se muestra el código en la caja de texto. Para que el código funcione es necesario definir con anterioridad el archivo sec.m, como se explicó en la Sección §4.6.2.2. A continuación se pulsa el botón Rep. Gráfica y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdGrafica_Click()
    lblCodigo = "Su instrucción en Matlab es:" & Chr(13) & " pie(X)"
  ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto ")
  ' Se define la variable con formato de Matlab
    datos = "" & txt_fun.Text & ""
  ' Se ejecuta la definición de la variable en Matlab
    sfichero = Matlab.Execute (datos)
  ' Se define la variable con la instrucción de Matlab
    sfichero2 = Matlab.Execute("sec;")
    pcturebox.Picture = LoadPicture("C:\proyecto\secto.jpg")
End Sub
```

La ejecución representa la gráfica en el *picturebox* situado en la parte inferior del formulario, el código de su instrucción en Matlab se muestra en la etiqueta lblCodigo, tal y como se aprecia en la Figura 43.

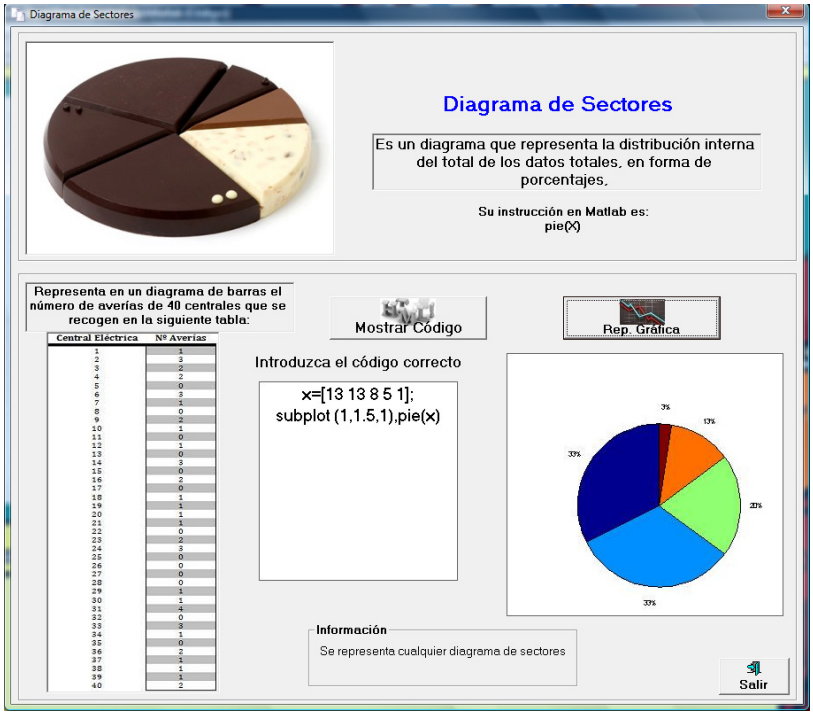


Figura 43: Diagrama de sectores

4.6.2.4. Histograma

Se agrega un nuevo formulario con el nombre frmHisto. En esta ventana se añaden los mismos controles que aparecen en los formularios de la Sección §4.6.2.2 y §4.6.2.3:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define el gráfico que se va a representar en este formulario y un *picturebox* que muestra un dibujo estándar sobre este tipo de gráficos.
- También se añade una etiqueta de nombre lblCodigo donde se muestra la instrucción en Matlab para representar este tipo de gráficos.
- Otro objeto de tipo *frame* que contiene una etiqueta y un *picturebox* con el enunciado y datos que se pretenden representar, además de una caja de texto donde se muestra la instrucción que representa dicho diagrama, esta instrucción se podrá modificar para calcular cualquier diagrama de este tipo.
- También se añaden al mismo *frame* dos botones con los que se selecciona la opción que se quiera realizar: Mostrar Código o Rep. Gráfica.
- En la parte inferior del formulario se agrega un *picturebox* donde se representa el diagrama de barras correspondiente a los datos propuestos.

Se comienza la ejecución de este formulario pulsando en el botón Mostrar Código, con lo que se muestra el código en la caja de texto. Para que el código funcione es necesario definir con anterioridad el archivo his.m, como se detalló en la Sección §4.6.2.2. A continuación se pulsa el botón Rep. Gráfica y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdGrafica_Click()
    lblCodigo = "Su instrucción en Matlab es:" & Chr(13) & " hist(x,6)"
  ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto ")
  ' Se define la variable con formato de Matlab
    datos = "" & txt_fun.Text & ""
  ' Se ejecuta la definición de la variable en Matlab
    sfichero = Matlab.Execute (datos)
  ' Se define la variable con la instrucción de Matlab
    sfichero2 = Matlab.Execute("his;")
    pcturebox.Picture = LoadPicture("C:\proyecto\his.jpg")
End Sub
```

La ejecución representa la gráfica en el control de imagen situado en la parte inferior del formulario, tal y como se observa en la Figura 44, su instrucción en Matlab se muestra en la etiqueta lblCodigo.

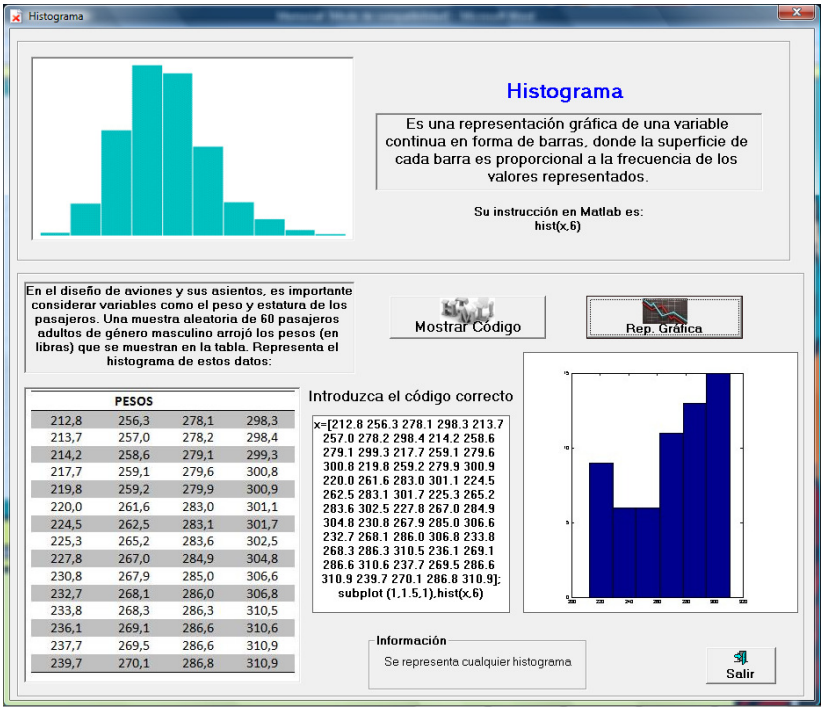


Figura 44: Histograma

4.6.3. Intervalos de confianza

Se agrega un objeto nuevo de tipo formulario y se le asigna el nombre relacionado con las estimaciones que vamos a realizar, frmIc. Para este formulario se añaden los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define el estimador que se va a analizar en este formulario y un *picturebox* que muestra un dibujo estándar sobre este estimador.
- También se añade una etiqueta de nombre lblCodigo donde se muestra la instrucción en Matlab para calcular un intervalo de confianza determinado.
- Otro objeto de tipo *frame* que contiene una etiqueta informativa con los datos cuyos parámetros se quieren estimar, además de dos cajas de texto donde se introducen los valores de la muestra del enunciado y el nivel de significación exigido, en estas cajas de texto se podrán modificar los valores de la muestra y el nivel de significación.

- A continuación se añaden a este mismo *frame* dos etiquetas informativas, una indicando cómo se deben introducir los datos de las muestras y la otra definiendo que el nivel de confianza es 1-α, donde α es el nivel de significación.
- También se añade un botón cuyo nombre es cmdEvaluar que calcula el intervalo de confianza mediante su instrucción en Matlab.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido en Matlab.

Para comenzar la ejecución en este formulario se introducen los datos de la muestra en la caja de texto (txtMuestra) y el nivel de significación en la caja de texto (txtNs) habilitadas para tal efecto. Se define el archivo inter.m (de la misma forma que en la Sección §4.3.4), que se encuentra implementado en el código. A continuación se pulsa el botón Resultado y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se ejecuta Matlab desde la ubicación definida
      Matlab.Execute ("cd C:\proyecto")
     ' Se definen las variables con formato de Matlab
      datos = "x=[" & txtMuestra.Text & "]"
      datos1 = "y=[" & txtNs.Text & "]"
    ‘ Se ejecutan las definiciones de las variables en Matlab
      Matlab.Execute(datos)
      Matlab.Execute(datos1)
     ' Se definen las variables con las instrucciones de Matlab
      sfichero = Matlab.Execute("inter;")
      sfichero2=Matlab.Execute("Media=[muhat],DesviacionEstandar=[sigmahat],
      IntervaloConfianzaMediaPoblacional=[muci]")
     ' Se asigna el título a la etiqueta
      lblResultado.Caption = sfichero2
End Sub
```

Los datos se introducen por código para su ejecución en Matlab y son los siguientes:

```
% valor introducido en la variable x
>> x = [21 19 23 19 23]
% valor introducido en la variable y
>> y = [0.01]
% se ejecutan las instrucciones del archivo inter.m en Matlab.
>> inter
% instrucciones que se ejecutan en Matlab para mostrar el resultado
anterior.
>> "Media=[muhat],DesviacionEstandar=[sigmahat],Intervalo
ConfianzaMediaPoblacional=[muci]"
```

La ejecución muestra el resultado de dicha operación en la etiqueta lblResultado y su instrucción en Matlab, en la etiqueta lblCodigo, tal y como se observa en la Figura 45.

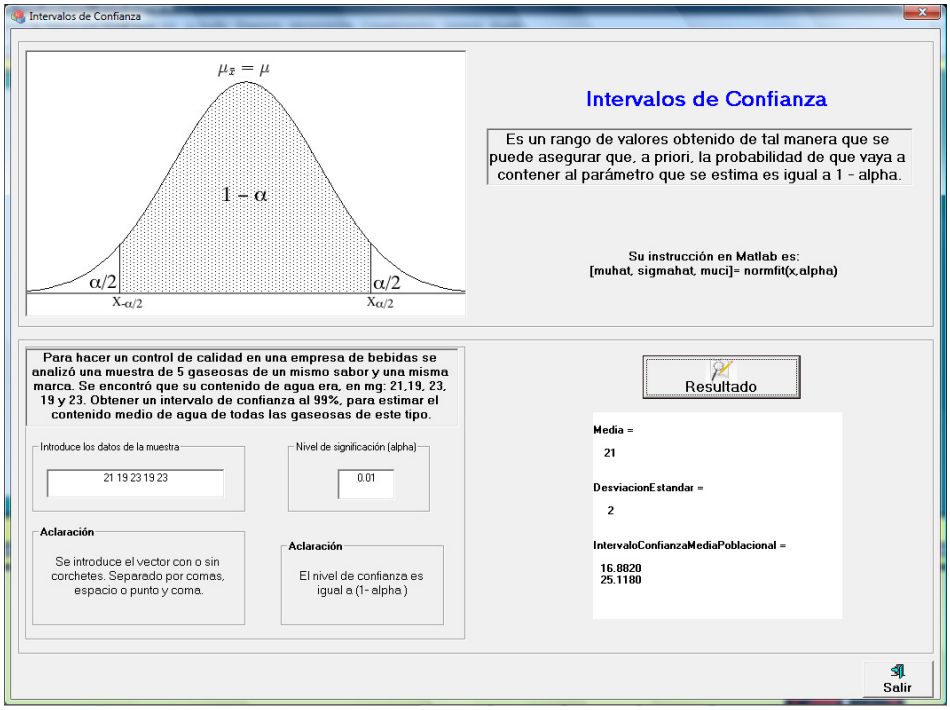


Figura 45: Intervalos de confianza

4.6.4. Regresión Lineal

Se agrega al proyecto un objeto nuevo de tipo formulario con el nombre frmRegresion. En este formulario se tienen los siguientes elementos:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define el método que se va a aplicar en este formulario y un control de imagen que muestra un dibujo estándar sobre el mismo.
- Otro objeto de tipo *frame* que contiene un *picturebox* que muestra una tabla de datos para su ajuste, además de una caja de texto donde se muestra el código de Matlab, al pulsar el botón Mostrar código con el que resolver el ajuste solicitado, en esta caja de texto se podrá modificar el código para calcular cualquier otro ajuste.
- Un botón cuyo nombre es cmdEvaluar que resuelve el sistema mediante su instrucción en Matlab.

- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido en Matlab.

Se comienza la ejecución de este formulario pulsando el botón Mostrar código, con lo que se muestra el código en la caja de texto, a continuación se pulsa el botón Resultado y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se define la variable con formato de Matlab
     datos = "" & txt_fun.Text & ""
  ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute(datos)
   ' Se asigna el título a la etiqueta
     lblresul.Caption = sfichero
End Sub
```

La ejecución muestra el resultado de dicha operación en la etiqueta lblResultado, tal y como se observa en la Figura 46.

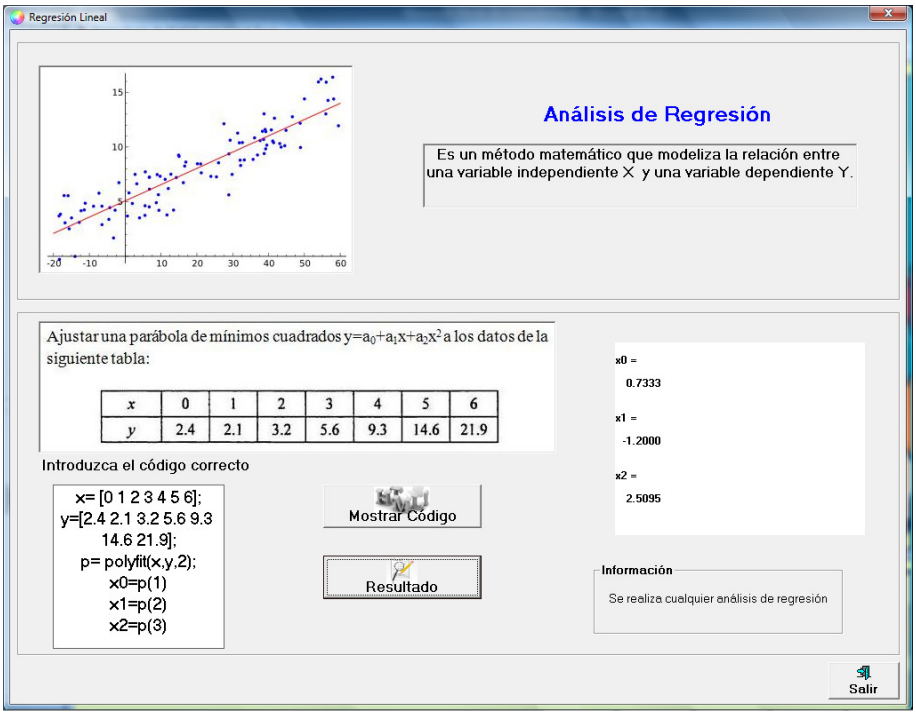


Figura 46: Regresión

4.7. Gráficos e Imágenes

En este menú se conocerán las instrucciones necesarias para trabajar con la representación de rectas, superficies, el tratamiento de imágenes y las funciones básicas en 2D y 3D. Dicho formulario está compuesto por:

- Funciones básicas (que se explicaron en la Sección §4.3.4)
- Histograma
- Representación de Funciones
- Ajuste de curvas

4.7.1. Histograma

Se agrega un nuevo objeto y se le asigna el nombre Histograma.frm. Este formulario está formado por los siguientes componentes:

- Cinco *picturebox* donde se muestran en una de ellas el dibujo del cual vamos a evaluar la posición y color del pixel, una vez seleccionado el pixel se muestra en un control de imagen y en los otros controles se representan los histogramas del pixel seleccionado de los tres colores (rojo, verde y azul) marcados en el formulario.
- Seis etiquetas informativas, una de ellas da título al formulario y las otras cinco indican donde se van a mostrar la posición y color del pixel seleccionado con el ratón.
- Cinco cajas de texto, tres de ellas muestran los valores RGB del color y las otras dos las coordenadas de *x* e *y* del pixel elegido.
- Un botón de nombre cmdHisto que representa los gráficos de cada color en su *picturebox* correspondiente.
- Una barra de progreso que permite medir el porcentaje de tiempo que tarda en calcular los histogramas del pixel evaluado para los colores considerados.
- Una etiqueta donde se muestra las instrucciones a seguir para evaluar cualquier pixel del dibujo.

Se comienza la ejecución en este formulario marcando un punto con el ratón en el dibujo, automáticamente aparecen en las cajas de texto los valores de la posición y color del pixel elegido. A continuación se pulsa el botón Histograma y se muestra en pantalla los histogramas azul, verde y rojo del pixel seleccionado, tal y como se observa en la Figura 47.

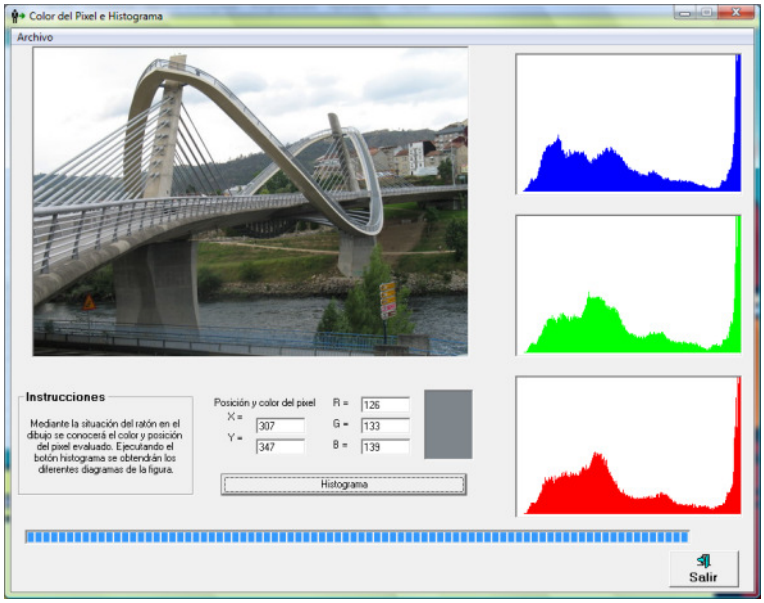


Figura 47: Histograma

4.7.2. Representación de Funciones

Se agrega un nuevo formulario y se le llama frmRep_Fun. Dentro de este formulario se añaden los siguientes controles:

- Un objeto de tipo *frame* que contiene varias etiquetas que muestran el enunciado de la función que se quiere representar y el código necesario para representar dicha función en un *picturebox.*
- También se añaden al mismo *frame* dos botones correspondientes a cada una de las opciones que se pueden realizar en este formulario: Mostrar código y Rep. gráfica que el usuario podrá utilizar para su aprendizaje.
- Otro objeto de tipo *frame* que contiene los mismos elementos que el anterior, solo que hace referencia a la representación de superficies.
- Una caja de texto donde se muestra el código de Matlab con el que representar curvas planas o superficies dependiendo del botón seleccionado, en esta caja de texto se podrá modificar el código para representar cualquier curva o superficie.
- Un *picturebox* donde se muestra la representación de la gráfica obtenida en Matlab.

Se comienza la ejecución de este formulario pulsando el botón Mostrar código, con el que aparece el código en la caja de texto habilitada para ello. Como ya hemos visto en la

Sección §4.3.4, en primer lugar, se define el archivo fsuper.m que forma parte del código del botón. A continuación se pulsa el botón Rep. Superficie y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdSuper_Click()
   ' Se ejecuta Matlab desde la ubicación definida
     Matlab.Execute ("cd C:\proyecto")
    ' Se define la variable con formato de Matlab
     datos = "" & txt_fun.Text & ""
   ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute(datos)
     sfichero2 = Matlab.Execute("fsuper;")
     pcturebox.Picture=LoadPicture("C:\proyecto\funsuper.jpg")
End Sub
```

Los datos se introducen por código para su ejecución en Matlab y son los siguientes:

```
% código introducido en la variable datos
>> "[x,y]=meshgrid(-4:0.2:4); z=y.^2/16 - x.^2/9; surf(x,y,z)"
% se ejecutan las instrucciones del archivo fsuper.m en Matlab.
>> fsuper
```

La ejecución muestra la representación del gráfico en el picturebox, tal y como se observa en la Figura 48.

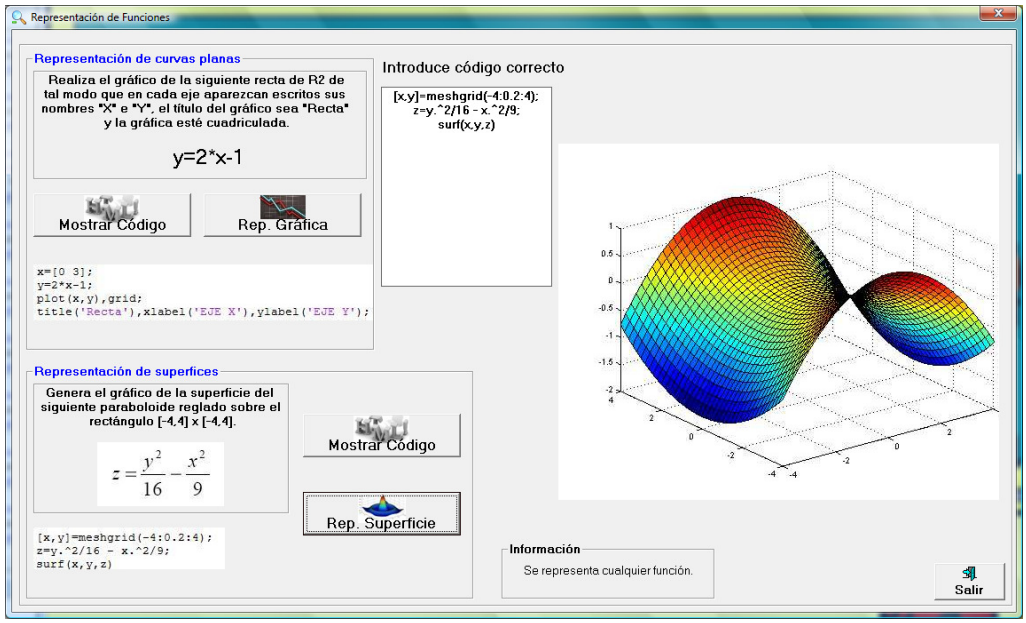


Figura 48: Representación de superficies

4.7.3. Ajuste de curvas

Se comienza agregando un formulario nuevo de nombre frmAjuste. En este formulario se añaden los siguientes componentes, tal y como se detallan en la Figura 49:

- Un objeto de tipo *frame* en el que se incluyen una etiqueta que define la aproximación que se va a utilizar en este formulario y un *picturebox* que muestra un dibujo sobre dicha aproximación.
- Otro objeto de tipo *frame* que contiene una etiqueta que muestra el enunciado, además de dos *picturebox* donde se detallan el conjunto de datos y la curva a la cual se realiza el ajuste.
- También se añade al mismo *frame* un botón cuyo nombre es cmdEvaluar que resuelve el sistema mediante su instrucción en Matlab.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido en Matlab.
- Un botón cuyo nombre es cmdGrafica que representa el gráfico mediante su instrucción en Matlab y un control de imagen que contiene a dicha gráfica.
- En la parte inferior del formulario se agrega un control de imagen que muestra las instrucciones de Matlab con las que se obtienen el resultado y la representación gráfica del ajuste.

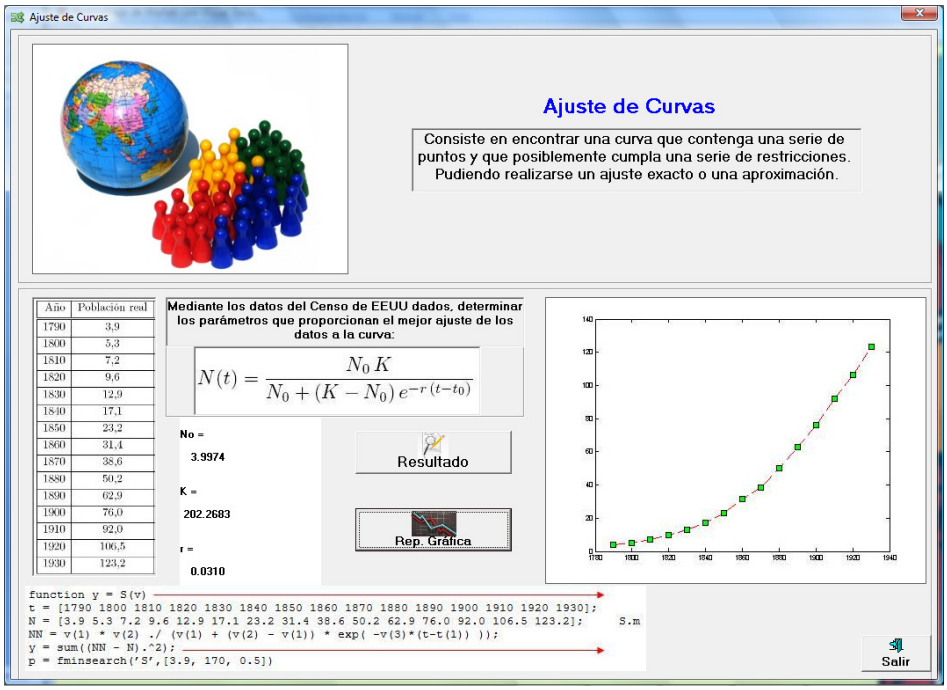


Figura 49: Ajuste de curvas

Se comienza la ejecución en este formulario pulsando el botón Resultado, con el que se obtendrán los valores de las constantes: No, K y r, también se muestra en pantalla la instrucción que calcula este tipo de ajuste. Como ya hemos visto anteriormente, primero se tiene que definir el archivo ajuste.m que se detalla en la Figura 50.

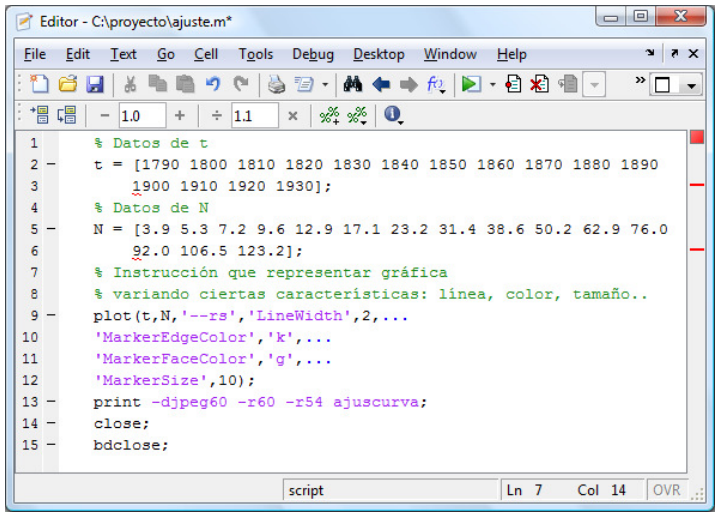


Figura 50: Archivo ajuste.m

A continuación se pulsa el botón Rep. Gráfica y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdGrafica_Click()
    ' Se ejecuta Matlab desde la ubicación definida
      Matlab.Execute ("cd C:\proyecto")
    ' Se define la variable con las instrucciones de Matlab
      sfichero = Matlab.Execute("ajuste")
      pcturebox.Picture=LoadPicture("C:\proyecto\ajuscurva.jpg")
End Sub
```

La ejecución muestra el resultado de dicha operación en la etiqueta lblResultado y la representación del gráfico en el control de imagen pcturebox.

4.8. Programación

Como cualquier lenguaje de programación, Matlab incluye un conjunto de herramientas que permiten a los programadores desarrollar programas informáticos. Utilizando el formulario que se detalla en esta sección, se conocerán las instrucciones que componen los programas que se ejecutan de forma secuencial. Es necesario conocer esas instrucciones y su sintaxis, por lo que este formulario contendrá, como se muestra en la Figura 51, las siguientes ventanas:

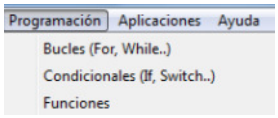


Figura 51: Menú programación

4.8.1. Bucles (For, While)

Se dispone de un nuevo formulario, con el nombre frm_for_while (haciendo alusión a las instrucciones que se utilizan). Dentro de este formulario se añaden los siguientes controles:

- Un objeto de tipo *frame* en el que se incluyen las etiquetas informativas donde se indica la función a programar o las instrucciones para desarrollar un pequeño programa. Además de un *picturebox* que muestra un dibujo estándar.
- Se incluyen en este *frame* dos botones de selección con los que se podrá elegir entre programar la función con `for` o con `while`.
- Otro objeto de tipo *frame* que contiene una lista de elementos y una tabla de contenido con estructura de árbol diferente según se seleccione `for` o `while`, donde se dispondrá del código de Matlab con el que programar la función solicitada.
- Una lista de imágenes que contiene las imágenes que se usan en la tabla de contenido, estas imágenes cambian al colocar los elementos en la posición correcta dentro de la función.
- Una etiqueta de nombre lblFor y lblWhile donde se muestran las indicaciones a seguir para programar una función u otra en Matlab.
- Dos botones de nombre cmdVerificar y cmdVerifica1 que comprueban en Matlab si la función programada es correcta para for o while respectivamente.
- Dos etiquetas de nombre lblResultado y lblResultado1 que muestra el resultado obtenido al verificar la función.

Para comenzar la ejecución en este formulario se selecciona entre `for` o `while`, por ejemplo, si elegimos `for` se muestra el código de la función en la lista de elementos y en la tabla de contenido los campos en forma de nodos, que se pueden ir arrastrando desde la lista de elementos a la tabla de contenido (ver Figura 52).

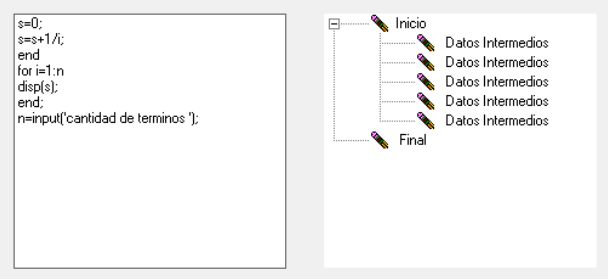


Figura 52: Lista de elementos y tabla de contenido

Si la secuencia es correcta el icono del nodo mostrado cambia, si por el contrario no es correcta aparece un mensaje de error (ver Figura 53), hasta completar las sentencias de dicha función.

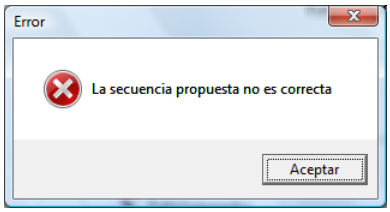


Figura 53: Mensaje de error

Por último, una vez completada la función y definido el archivo fo.m, (como se explicó en la Sección §4.3.4), se comprueba la función pulsando el botón Verificar y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdVerificar_Click()
    ' Se ejecuta Matlab desde la ubicación definida
      Matlab.Execute ("cd C:\proyecto ")
   ' Se define la variable con la instrucción de Matlab
      sfichero = Matlab.Execute("fo")
    ' Se asigna el título a la etiqueta
      lblResultado.Caption = sfichero
End Sub
```

La ejecución muestra el resultado de verificar dicha función en Matlab en la etiqueta lblResultado y las instrucciones para programar en Matlab esta función en la etiqueta lblFor, tal y como se aprecia en la Figura 54.

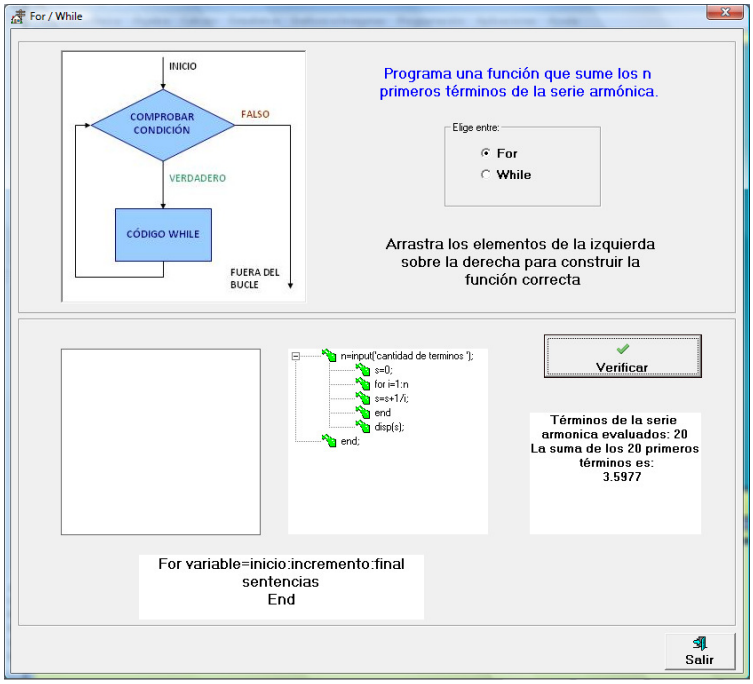


Figura 54: Formulario "for"

4.8.2. Condicionales (If, Switch)

Se agrega un nuevo formulario, con el nombre frm_if_switch. En este formulario, igual que en el anterior, se añaden los siguientes controles:

- Un objeto de tipo *frame* en el que se incluyen las etiquetas informativas donde se indica la función a programar o la forma de mover los elementos disponibles, arrastrándolos de izquierda a derecha para crear la función correcta. Además de un *picturebox* que muestra un dibujo estándar sobre la función a programar.
- Se incluyen en este *frame* dos botones de selección con los que se podrá elegir entre programar la función con `if` o con `switch`.
- Otro objeto de tipo *frame* que contiene una lista de elementos y una tabla de contenido con estructura de árbol diferente según se seleccione `if` o `switch`, donde se dispondrá del código de Matlab con el que programar la función solicitada.
- Una lista de imágenes que contiene las imágenes que se usan en la tabla de contenido, estas imágenes cambian al colocar los elementos en la posición correcta dentro de la función.

- Una etiqueta de nombre lblIf y lblSw donde se muestran las condiciones que debe tener la función si se programa con If o con Switch.
- Un botón de nombre cmdVerificar y cmdVerificar1 que comprueba en Matlab si la función programada es correcta.
- Otra etiqueta de nombre lblResultado y lblResultado1 donde se muestra el resultado obtenido en Matlab.

Para comenzar la ejecución en este formulario se selecciona entre `if` o `switch`, por ejemplo, si elegimos `if` se muestra el código de la función en la lista de elementos y en la tabla de contenido los campos en forma de nodos, donde se pueden ir arrastrando desde la lista de elementos a la tabla de contenido (ver Figura 55).

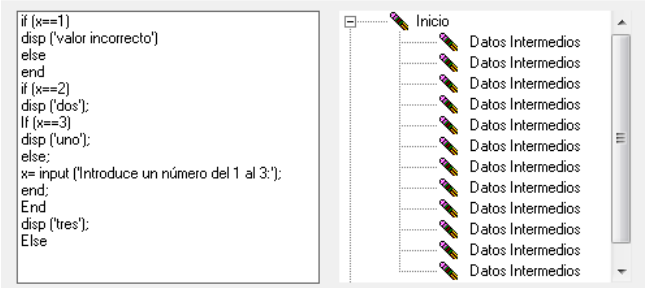


Figura 55: Lista de elementos y tabla de contenido "if"

Si la secuencia es correcta el icono del nodo mostrado cambia, si por el contrario no es correcta aparece un mensaje de error, hasta completar las sentencias de dicha función, mostrando el siguiente mensaje, tal y como se muestra en la Figura 56.

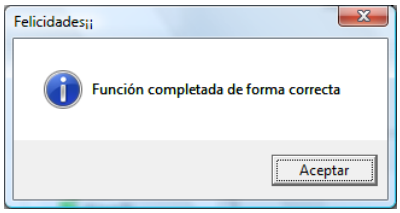


Figura 56: Mensaje función correcta

Una vez completada la función y definido el archivo fi.m (como se detalló en la Sección §4.3.4), se pulsa el botón Verificar y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdVerificar_Click()
    ' Se ejecuta Matlab desde la ubicación definida
      Matlab.Execute ("cd C:\proyecto ")
   ' Se define la variable con la instrucción de Matlab
```

```
        sfichero = Matlab.Execute("fi")
    ' Se asigna el título a la etiqueta
        lblResultado.Caption = sfichero
End Sub
```

La ejecución muestra el resultado de comprobar la función en la etiqueta lblResultado y las instrucciones para programar en Matlab esta función en la etiqueta lblIf, tal y como se observa en la Figura 57.

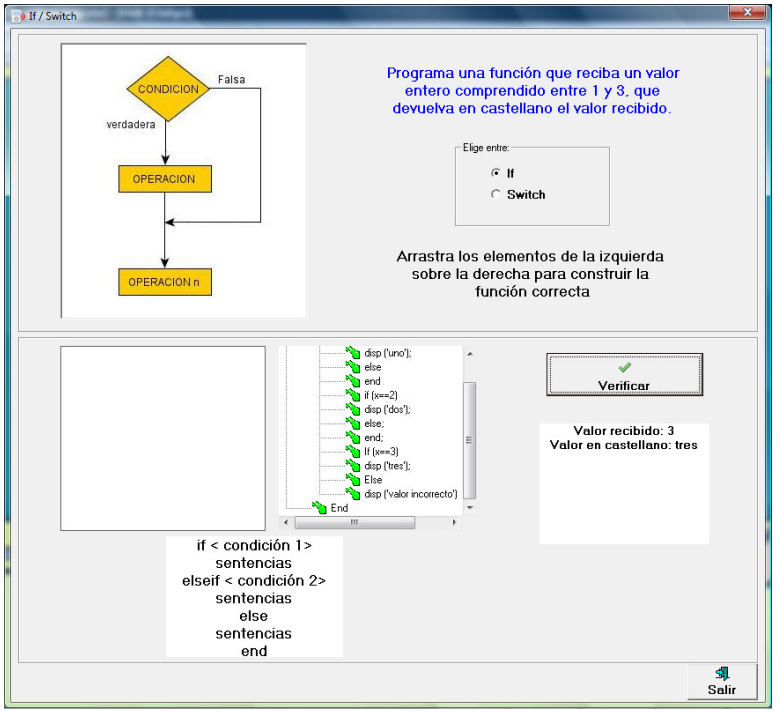


Figura 57: Formulario "if"

4.8.3. Funciones

En el formulario añadido a continuación, frmFuncion, se mostrará cómo programar funciones de Matlab de forma sencilla. Dentro de este formulario se añaden los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluye una etiqueta que indica cómo se programa una función en Matlab y un *picturebox* que muestra el icono de Matlab.
- Otro objeto de tipo *frame* que contiene una etiqueta que muestra la función a programar, además de una caja de texto donde se muestra el código de Matlab con el que se programa dicha función, en esta caja de texto se podrá modificar el código para programar cualquier función.

- Se añade al mismo *frame* una caja de texto donde se introducirá un parámetro cualquiera con el que comprobar que la función es correcta, mostrando en pantalla los valores pedidos en el enunciado.
- También se añaden al mismo *frame* dos botones correspondientes a cada una de las opciones que se pueden realizar en este formulario: Mostrar código y Resultado que resultan útiles para el aprendizaje por parte del usuario de las instrucciones necesarias en Matlab.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido de Matlab.

Se comienza la ejecución de este formulario pulsando el botón Mostrar código, con lo que se muestra el código en la caja de texto, a continuación se introduce un parámetro en la caja de texto habilitada para tal efecto, por ejemplo, parámetro = 8, se pulsa el botón Resultado y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se ejecuta Matlab desde la ubicación definida
      Matlab.Execute ("cd C:\proyecto")
    ' Se define la variable con formato de Matlab
      datos = "" & txt_fun.Text & ""
      sparam=txtParametro.Text
    ' Se define la variable con la instrucción de Matlab
      sfichero = "par("&sparam&")"
      sfichero2 = Matlab.Execute(sfichero)
    ' Se asigna el título a la etiqueta
      lblResultado.Caption = sfichero2
End Sub
```

Los datos para su ejecución en Matlab, se muestran de la siguiente forma:

```
% código introducido en la variable datos
>> "function par(x); if x<10000; for i=1:x; if mod(i,2)==0; disp
(i);end; end; end; return"
% valor introducido en la variable parámetro.
>> 8
% se ejecuta la función par con el parámetro 8 en Matlab.
>> par(8)
```

Esta ejecución muestra el resultado de dicha operación en la etiqueta lblResultado y el código de la función programada en Matlab en el control de imagen, tal y como se observa en la Figura 58.

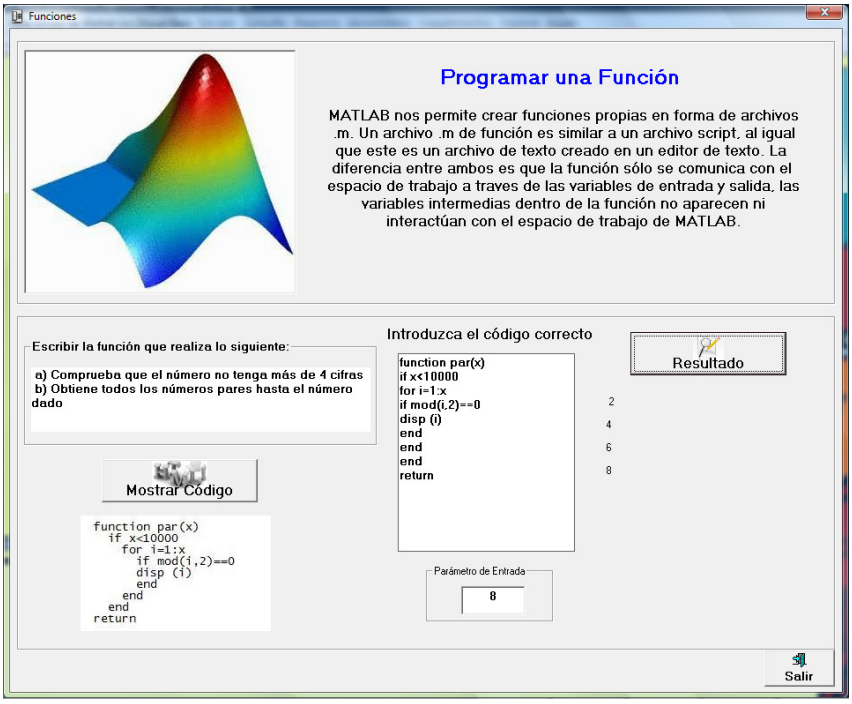


Figura 58: Programar funciones

4.9. Aplicaciones

En este menú se resolverán algunas aplicaciones tipo relacionadas con las asignaturas que se cursan en los estudios de ingeniería y que serán de ayuda a los usuarios que utilicen dicha aplicación para su aprendizaje. Para ello se tienen las ventanas que se muestran en la Figura 59:

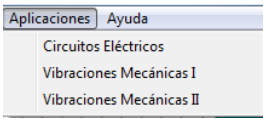


Figura 59: Menú aplicaciones

4.9.1. Circuitos eléctricos

Se agrega un nuevo objeto al proyecto, de tipo formulario y le damos el nombre relacionado con las aplicaciones que vamos a resolver, en este caso frmCirc.

Dentro de este formulario se tienen los siguientes componentes (ver Figura 60):

- Un objeto de tipo *frame* en el que se incluye una etiqueta con el enunciado de la aplicación que se va a resolver, además de un *picturebox* que contiene el circuito mencionado en el enunciado y del cual se tienen que calcular sus intensidades.

- Otro objeto de tipo *frame* que contiene una etiqueta informativa de cómo hay que introducir los valores de las resistencias y de las fuentes de tensión.
- También se incluyen dos cajas de texto (txtResis y txtFuentes) donde introducir los valores de las resistencias y fuentes de tensión.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido en Matlab.
- Un botón cuyo nombre es cmdCodigo que muestra el código con el que se resuelve el circuito dado mediante su instrucción en Matlab.
- Otro botón cuyo nombre es cmdEvaluar que calcula el valor de las intensidades mediante su instrucción en Matlab.
- Un *picturebox* donde se muestra la instrucción de Matlab que resuelve el circuito propuesto.

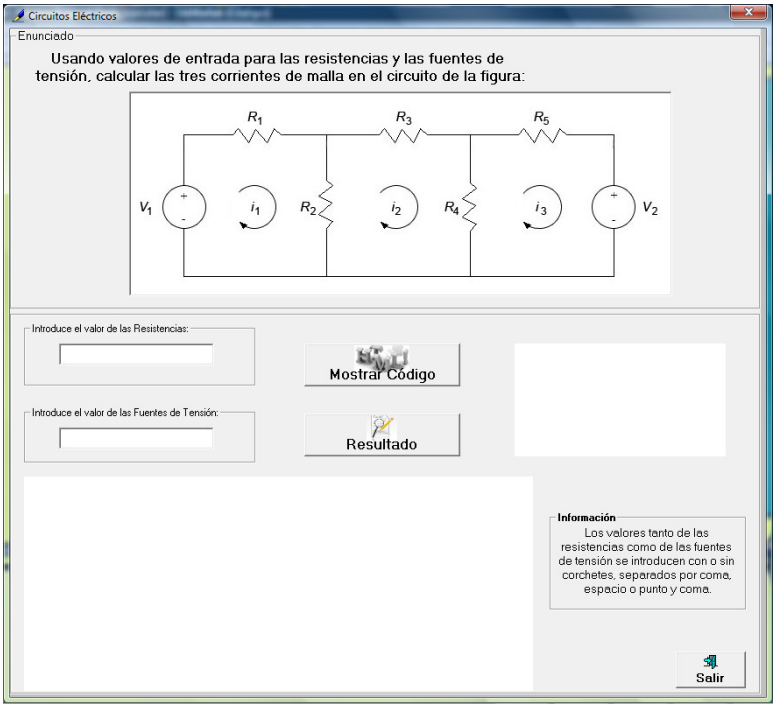


Figura 60: Formulario de circuitos eléctricos

Para comenzar la ejecución en este formulario se introducen los valores de las resistencias y fuentes de tensión. En primer lugar se tiene que definir el archivo circuito.m, que va implementado en el código del botón, tal y como se detalla en la Figura 61.

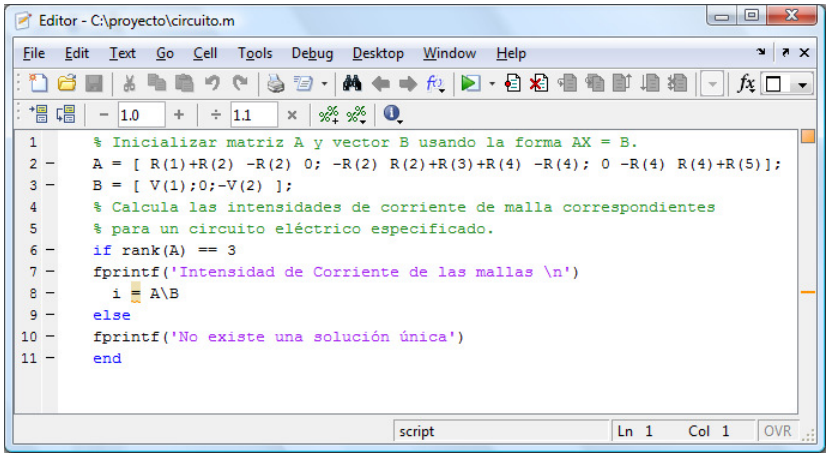


Figura 61: Archivo circuito.m

A continuación se pulsa el botón Mostrar código si se quiere conocer el código que resuelve la aplicación o por el contrario se calcula el valor de las intensidades del circuito pulsando el botón Resultado, seguidamente se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdEvaluar_Click()
    ' Se definen las variables con formato de Matlab
     datos = "R = [" & txtResis.Text & "]"
     datos1 = "V = [" & txtFuentes.Text & "]"
   ' Se ejecutan las definiciones de las variables en Matlab
     Matlab.Execute (datos)
     Matlab.Execute (datos1)
    ' Se define la variable con la instrucción de Matlab
     sfichero = Matlab.Execute("circuito")
   ' Se asigna el título a la etiqueta
     lblResultado.Caption = sfichero
End Sub
```

Los datos que se introducen por código para su ejecución en Matlab son los siguientes:

```
% valor introducido en la variable R
>> R = [1 1 1 1 1]
% valor introducido en la variable V
>> V = [5 -5]
% se ejecutan las instrucciones del archivo circuito.m en Matlab.
>> circuito
```

Esta ejecución muestra el resultado de dicha operación en la etiqueta lblResultado y el código de Matlab en la etiqueta lblInfo, tal y como se observa en la Figura 62.

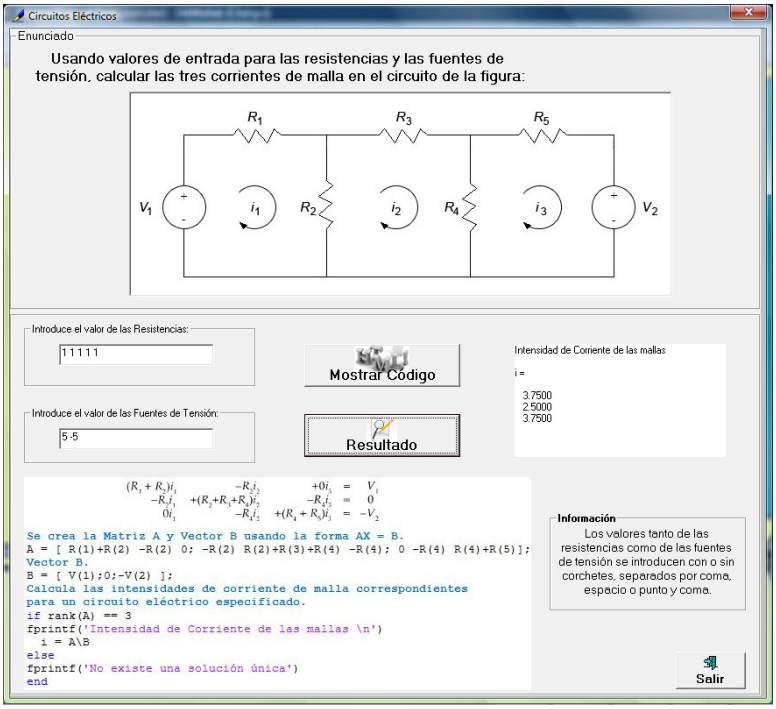


Figura 62: Ejecución Circuitos eléctricos

4.9.2. Vibraciones mecánicas I

Se agrega un nuevo objeto al proyecto, de tipo formulario frmVibrac. En este formulario se añaden los siguientes controles (ver Figura 63):

- Un objeto de tipo *frame* en el que se incluye una etiqueta con el enunciado de la aplicación que se va a resolver, además de un *picturebox* que contiene el sistema mencionado en el enunciado y del cual tenemos que calcular su función de transferencia y respuesta del sistema.
- Se añaden también a este *frame* dos *picturebox* donde se muestra la instrucción de Matlab que resuelve el circuito propuesto.
- Otro objeto de tipo *frame* en el que se incluyen tres cajas de texto donde hay que introducir los valores de M (masa), K (constante del resorte) y B (fricción viscosa) necesarias para calcular el valor de la función de transferencia.
- Los botones correspondientes a cada una de las operaciones que se pueden realizar: Transferencia, Respuesta del sistema y Diagrama de bloques, con los que se obtendrán los resultados pedidos en el enunciado.

- También se incluyen tres *picturebox* que contienen la fórmula, el diagrama de la función de transferencia y un esquema de un diagrama de bloques tipo, este diagrama cambia al pulsar el botón Diagrama de bloques mostrando el de la función solicitada.
- Un control de imagen cuyo nombre es pctureResp que representa el gráfico de la respuesta del sistema.
- Otra etiqueta de nombre lblResultado donde se muestra el resultado de la función de transferencia obtenida en Matlab.

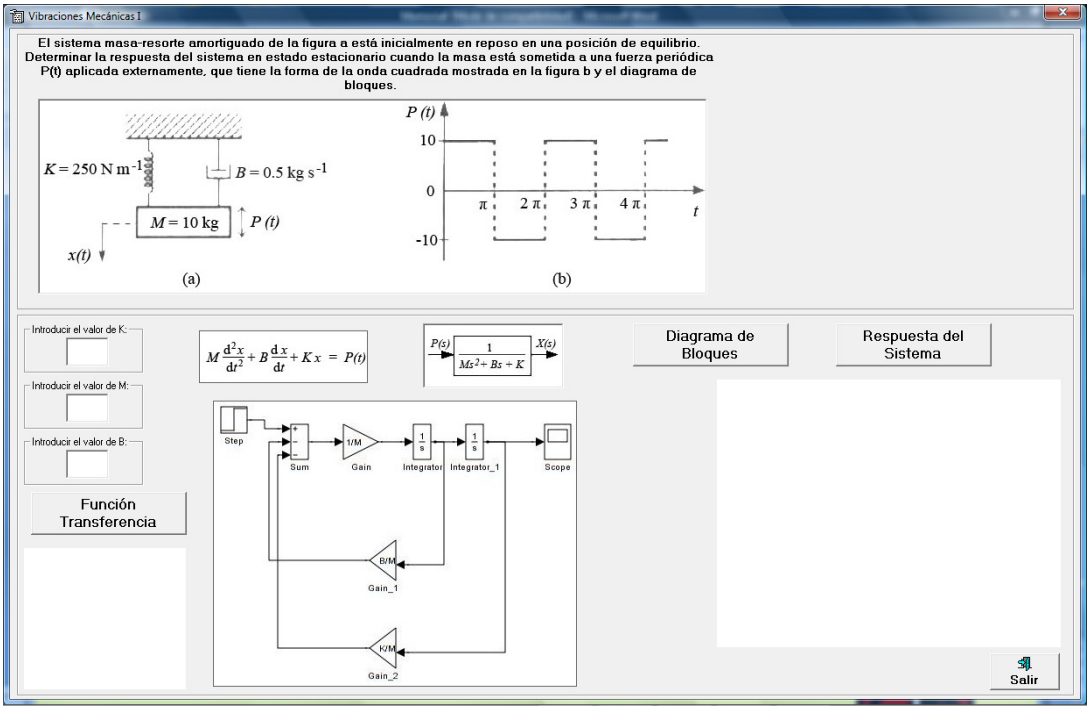


Figura 63: Formulario Vibraciones mecánicas I

Para comenzar la ejecución en este formulario se introducen los valores de M, K y B. Después se pulsa el botón Transferencia para obtener los valores de la función de transferencia, de la misma forma para conocer la gráfica de la respuesta del sistema se pulsa el botón Respuesta del sistema y por último para obtener su diagrama de bloques en Simulink, es necesario definir el archivo test.m que va implementado en el código del botón, tal y como se detalla en la Figura 64.

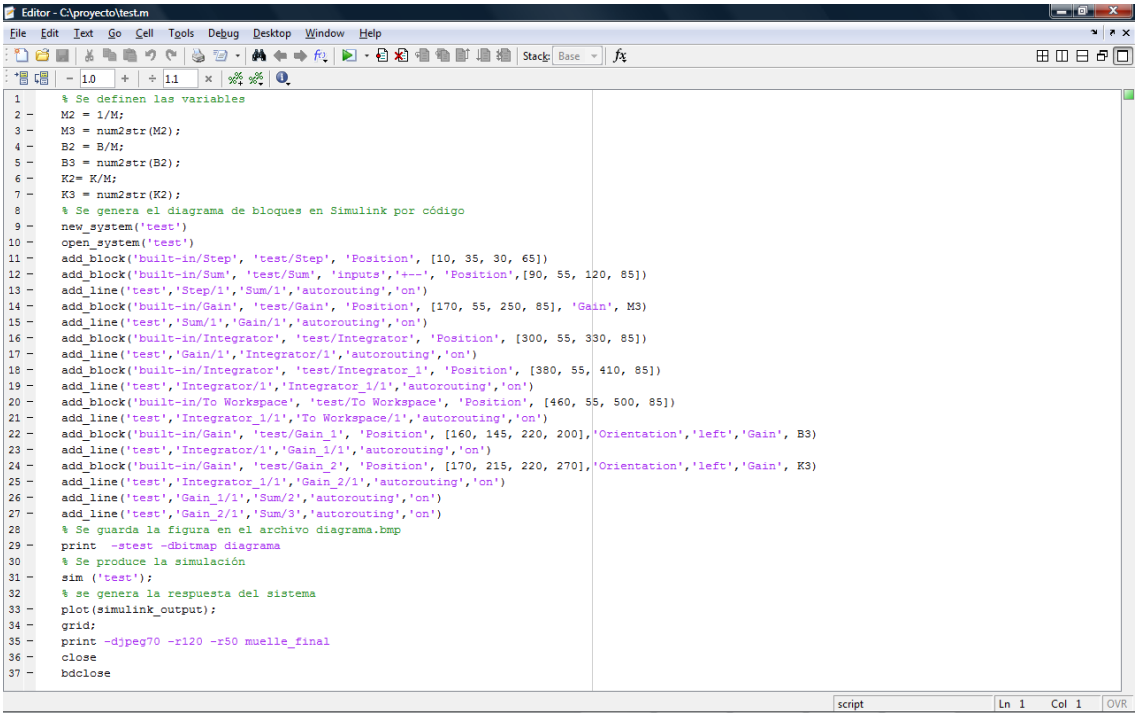


Figura 64: Archivo test.m

A continuación se pulsa el botón Diagrama de bloques y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdBloques_Click()
    pctureBloques.Picture = LoadPicture("C:\proyecto\bloques_codigo.jpg")
    pctureTrans.Visible = False
  ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto")
  ' Se definen las variables con formato de Matlab
    datos = "M = [" & txtvalorm.Text & "]"
    datos1 = "B = [" & txtvalorb.Text & "]"
    datos2 = "K = [" & txtvalork.Text & "]"
  ' Se ejecutan las definiciones de las variables en Matlab
    Matlab.Execute (datos)
    Matlab.Execute (datos1)
    Matlab.Execute (datos2)
  ' Se define la variable con la instrucción de Matlab
    sfichero = Matlab.Execute("test")
    pctureSim.Picture = LoadPicture("C:\proyecto\diagrama.bmp")
End Sub
```

Una vez que se introducen los datos, su ejecución en Matlab es la siguiente:

```
% valor introducido en la variable M.
>> M= [10]
% valor introducido en la variable B.
>> B= [250]
```

```
% valor introducido en la variable K.
>> K= [0.5]
% se ejecutan las instrucciones del archivo test.m en Matlab.
>> test
```

Esta ejecución muestra el resultado de la función de transferencia en la etiqueta lblResultado, el código de Matlab en el pctureBloques, la respuesta del sistema en el control de imagen pctureResp y el diagrama de bloques en el control de imagen pctureSim, tal y como se observa en la Figura 65.

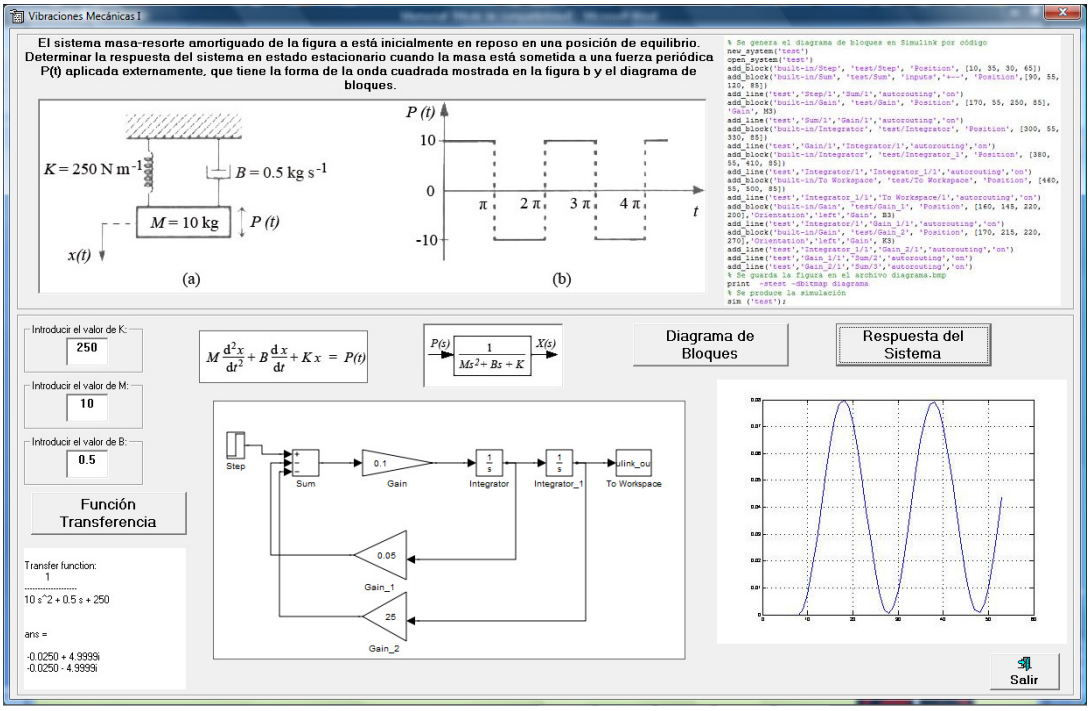


Figura 65: Ejecución Vibraciones

4.9.3. Vibraciones mecánicas II

Se agrega un nuevo formulario con el nombre relacionado frmVic_Mec.

Dentro de este formulario se tienen los siguientes componentes (ver Figura 66):

- Un objeto de tipo *frame* en el que se incluye una etiqueta con el enunciado de la aplicación que se va a resolver, además de un *picturebox* que contiene la representación gráfica del ejercicio que se resuelve.
- Otro objeto de tipo *frame* que contiene cajas de texto donde se introducirán los valores de la matriz M (masa) y K (constante de rigidez) que se desea calcular, para obtener el valor de Kr, valores y vectores propios.

- Tres botones correspondientes a cada una de las operaciones que se pueden realizar: Valor de kr, Valores propios y Vectores propios con los que se obtendrán los resultados solicitados en el enunciado.
- Una etiqueta de nombre lblResultado donde se muestra el resultado obtenido de Matlab.
- Otra etiqueta donde se muestra una explicación del botón seleccionado y la instrucción de Matlab que resuelve cada operación.

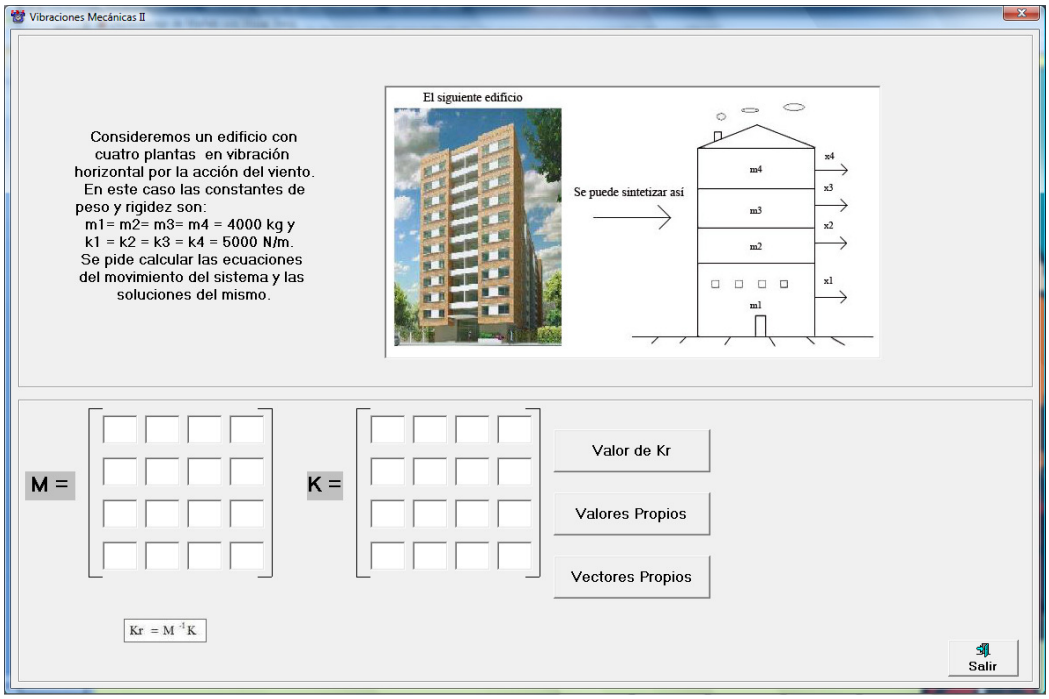


Figura 66: Formulario Vibraciones II

Se comienza la ejecución en este formulario introduciendo los valores de M y K. Se dispone de los botones Valor de Kr, Valores y Vectores propios, con los que se obtienen Kr, los valores propios y vectores propios, respectivamente. Por ejemplo, si el usuario quiere conocer los vectores propios, se pulsa el botón Vectores propios y se ejecuta el código programado en el evento "Click" del botón:

```
Private Sub cmdVec_Click()
    lblDetalle.Caption = "Su instrucción en Matlab es:" & Chr(13) & "[V,D] = eig(Kr)"
  ' Se ejecuta Matlab desde la ubicación definida
    Matlab.Execute ("cd C:\proyecto")
  ' Se definen las variables con formato de Matlab
    datos = "M = [" & txt_a.Text & Chr(32) & txt_e.Text & Chr(32) & txt_i.Text &
    Chr(32) & txt_m.Text & Chr(32) & ";" & txt_b.Text & Chr(32) & txt_f.Text &
    Chr(32) & txt_j.Text & Chr(32) & txt_n.Text & Chr(32) & ";" & txt_c.Text &
```

```
        Chr(32) & txt_g.Text & Chr(32) & txt_k.Text & Chr(32) & txt_o.Text & Chr(32) &
        ";" & txt_d.Text & Chr(32) & txt_h.Text & Chr(32) & txt_l.Text & Chr(32) &
        txt_p.Text & "]"
        datos1 = "K = [" & txt_1.Text & Chr(32) & txt_2.Text & Chr(32) & txt_3.Text &
        Chr(32) & txt_4.Text & Chr(32) & ";" & txt_5.Text & Chr(32) & txt_6.Text &
        Chr(32) & txt_7.Text & Chr(32) & txt_8.Text & Chr(32) & ";" & txt_9.Text &
        Chr(32) & txt_10.Text & Chr(32) & txt_11.Text & Chr(32) & txt_12.Text &
        Chr(32) & ";" & txt_13.Text & Chr(32) & txt_14.Text & Chr(32) & txt_15.Text &
        Chr(32) & txt_16.Text & "]"
    ' Se ejecutan las definiciones de las variables en Matlab
        Matlab.Execute (datos)
        Matlab.Execute (datos1)
    ' Se define la variable con la instrucción de Matlab
        sfichero = Matlab.Execute("[V,D] = eig(Kr)")
    ' Se asigna el título a la etiqueta
        LblResultado.Caption = sfichero
End Sub
```

La ejecución muestra el resultado en la etiqueta lblResultado y el código de Matlab en la etiqueta lblDetalle, tal y como se observa en la Figura 67.

Figura 67: Ejecución vibraciones II

4.10. Ayuda

En este último menú se encuentra toda la información que se necesita tanto para manejar esta aplicación como para ampliar los conocimientos de Matlab y Visual Basic.

Se tienen las siguientes ventanas, como se muestra en la Figura 68:

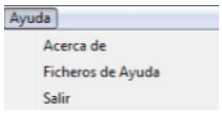


Figura 68: Menú ayuda

4.10.1. Acerca de

Se agrega un nuevo formulario con el nombre frmAcerca_de. En este formulario (ver Figura 69) se dispone de los siguientes componentes:

- Un objeto de tipo *frame* en el que se incluyen las etiquetas informativas sobre el título de la aplicación, su versión y un aviso que informa sobre su legalidad y formas de uso responsables.
- También se añade al mismo *frame* dos *picturebox* que incluyen los logotipos de los dos programas, Matlab y Visual Basic, utilizados para desarrollar esta aplicación.
- Un botón de nombre cmdOk que finaliza el formulario actual y vuelve al principal.

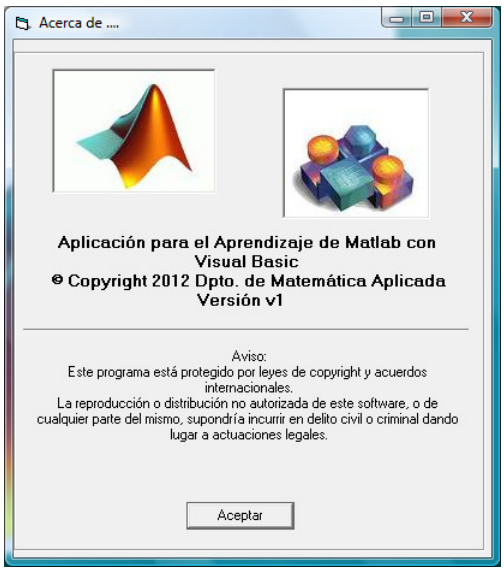


Figura 69: Formulario Acerca de

4.10.2. Ficheros de ayuda

Desde el menú principal (vbMatlab.frm) llamaremos al fichero de ayuda mediante la api ShellExecute, asignando al botón el evento "Click" correspondiente:

```
Private Sub mnuAyuda_Click()
  Dim x
  x=ShellExecute(Screen.ActiveForm.hWnd,"open","C:\proyecto\ayuda.chm",vbNullSt
  ring, vbNullString, 1)
End Sub
```

Además se puede solicitar la ayuda presionando F1, para ello en el evento "Load" del formulario principal se incluye el siguiente código:

```
App.HelpFile = "C:\proyecto\ayuda.chm"
```

Para crear estos archivos se utiliza cualquier programa que permita generar o modificar ficheros .html en los que se incluye los contenidos que se quieren facilitar en la ayuda, en esta aplicación son: instrucciones útiles, manuales de: usuario, Matlab y Visual Basic. A continuación se compila con el programa Html Help Workshop. Con el que se obtiene el fichero .chm que se necesita para enlazar al menú principal.

4.10.3. Salir

Se selecciona este submenú y mediante la sentencia **Unload me** se finaliza la aplicación.

CAPÍTULO 5: CONCLUSIONES

Con este Proyecto Fin de Carrera se ha pretendido generar una herramienta o aplicación que sirva para aprender de forma sencilla el manejo de Matlab, para usuarios con un nivel bajo o mínimo de programación y que puedan utilizarla también como apoyo para sus estudios, ya que se tratan las materias básicas que componen los estudios de Ingeniería y Ciencias.

Los objetivos marcados inicialmente se han cumplido satisfactoriamente, ya que la aplicación reúne las características suficientes para que el usuario final pueda utilizarla para un aprendizaje eficiente, de modo que se pueda utilizar de una forma autodirigida y ser el propio usuario quien decida lo que desea aprender, cómo y cuándo. Sin duda constituirá un material de trabajo muy útil para las asignaturas de matemáticas. Por estos motivos podrá ser útil para los estudiantes de primer y segundo curso de los Grados de Ingeniería.

Personalmente, este proyecto me ha aportado nuevas técnicas para afrontar diferentes actividades de mi vida profesional, he adquirido soltura a la hora de buscar información, conocimientos básicos del lenguaje Visual Basic y el programa Matlab, que son muy importantes y útiles. También me ha permitido mejorar a la hora de redactar memorias y documentos que no siempre es fácil.

Para trabajos futuros se tendrá en cuenta que en este proyecto no se han considerado las validaciones que harían falta en este tipo de proyectos de Visual Basic.

CAPÍTULO 6: REFERENCIAS

[1] Información en línea:
http://www.mathworks.es/
[2] Información en línea:
http://www.elguille.info/
[3] Información en línea:
http://www.esi2.us.es/~fabio/apuntes_matlab.pdf
[4] Información en línea:
http://www.recursosvisualbasic.com.ar/htm/utilidades-codigo-fuente/calculadora-simple.htm
[5] Información en línea:
http://www.mat.ucm.es/~rrdelrio/documentos/rrrescorial2002.pdf
[6] Información en línea:
http://usuarios.multimania.es/explorar/htmlvb/htmlvb.htm
[7] ÁLVAREZ CONTRERAS, S. J. "Estadística aplicada: teoría y problemas", Editorial: CLAG, S.A.
[8] ARNOLD, JESSE C. Y MILTON, J. SUSAN "Probabilidad y estadística con aplicaciones para ingeniería y ciencias computacionales", Editorial: McGRAW-HILL, Cuarta Edición.
[9] BURGOS, J. "Cálculo Infinitesimal de una variable".
[10] BUSTOS, M. T. "Teoría de Fundamentos Matemáticos II. Cálculo infinitesimal", Editorial: REVIDE.
[11] CHAPRA, S. C. Y CANALE, R. P. "Métodos numéricos para Ingenieros". Editorial: McGRAW-HILL.
[12] DE LA VILLA CUENCA, A. "Problemas de Algebra", Editorial: CLAGSA.
[13] ETTER, DELORES M. "Solución de problemas de Ingeniería con Matlab", Editorial: PRENTICE HALL, Segunda Edición.
[14] GARCÍA, A. "Cálculo I. Teoría y Problemas de Análisis Matemático en una variable", Editorial: CLAGSA.
[15] GARCÍA, A. "Cálculo II. Teoría y Problemas de funciones de varias variables", Editorial: CLAGSA.

[16] GARCÍA, A. "Ecuaciones diferenciales ordinarias. Teoría y Problemas", Editorial: CLAGSA.

[17] GARCÍA DE JALÓN, J. RODRÍGUEZ, J. I. Y VIDAL, J. "Aprenda matlab 7.0 como si estuviera en primero", Escuela Técnica Superior de Ingenieros Industriales (Universidad Politécnica de Madrid).

[18] GERALD, C. F. Y WHEATLEY, P. O. "Análisis Numérico con Aplicaciones", Editorial: PRENTICE HALL.

[19] JAMES, GLYN "Matemáticas Avanzadas para Ingenieros", Editorial: PRENTICE HALL, Segunda Edición.

[20] KINCAID, D. CHENEY, W. "Análisis Numérico", Editorial: ADDISON WESLEY IBEROAMERICANA.

[21] MATHEWS, J. H. Y FINK, K. D. "Métodos Numéricos con Matlab", Editorial: PRENTICE HALL.

[22] PEASLEY, R. PRUCHNIAK, W. Y RESELMAN, B. "Visual Basic 6", Editorial: PRENTICE HALL.

[23] PÉREZ, C. "Matlab y sus Aplicaciones en las Ciencias y la Ingeniería", Editorial: PRENTICE HALL.

[24] ROJO, J. "Algebra lineal", Editorial: McGRAW-HILL, Segunda Edición.