Tema 4: Empezando a trabajar con ficheros .m

1. Introducción

Como ya se comentó en el punto 3 del tema1, en Matlab tienen especial importancia los ficheros–M de extensión .m. Contienen conjuntos de comandos a ejecutar o definición de funciones y se ejecutan al teclear su nombre en la línea de comandos y pulsar intro (si se encuentra en el Current Directory) o al pinchar sobre él en Current Directory con el botón derecho del ratón y elegir run. Representan el centro de la programación en Matlab.

Un fichero .m puede llamar a otros ficheros .m y ficheros de comandos pueden ser llamados desde ficheros de funciones. En estos casos es importante tener en cuenta la definición de las variables a utilizar, en la línea de que tengan un tratamiento local o global. Por defecto, Matlab considera las variables locales, es decir, aunque varias funciones tengan la variable x, ésta es diferente en cada caso a no ser que haya sido definida como global.

Son ficheros de texto sin formato y que pueden crearse a partir de un editor de textos, no obstante, lo mejor es utilizar el editor del propio programa al que se accede por defecto al abrir un nuevo fichero.

2. Editor

Para crear un nuevo fichero .m elegimos new M-File del menú File o elegimos el icono correspondiente. Aparece entonces la pantalla del editor/Debugger donde podemos ir escribiendo las sentencias.

En el caso de la figura 13 el fichero contiene comentario, dos sentencias y otro comentario. Las sentencias seguidas de (;) no se mostrarán en pantalla al ejecutar el fichero.

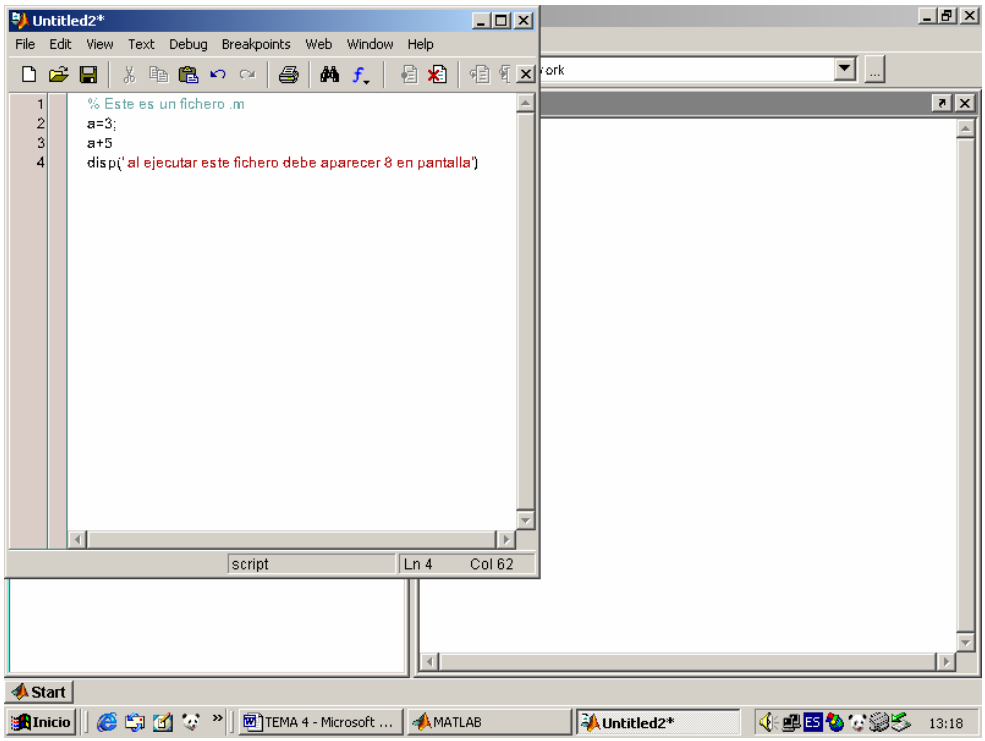

Figura 13

El editor muestra en diferentes colores los diferentes tipos de comandos:

- Verde para los comentarios.
- Rojo para las cadenas de caracteres.
- Negro para las sentencias.

Una posibilidad de ejecutar el fichero es elegir Run del menú Debug (primeramente debe guardarse con un nombre). Es posible ejecutar el mismo por partes incluyendo breakpoints lo que puede hacerse con el icono de los puntos rojos. Si se tienen estos puntos de parada se continua de uno a otro con la opción de Continue. Es posible visualizar el valor que van tomando los distintos elementos del fichero posándonos con el ratón sobre ellos (figura 14).

Es interesante ir conociendo los demás botones y menús del editor que nos permite eliminar los breakpoints, terminar la ejecución,.... Resulta muy útil para detectar errores y corregirlos y en general para programar

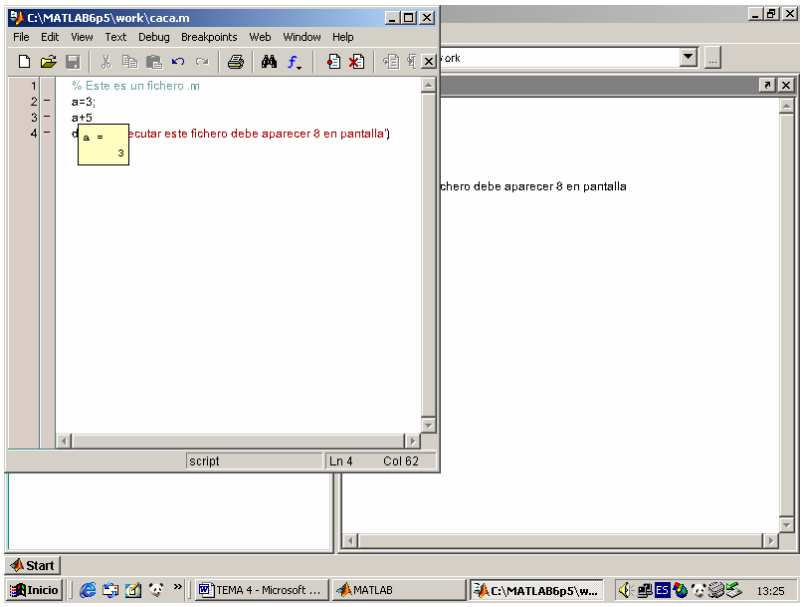

Figura 14

Si se desea modificar o visualizar un fichero .m ya creado, es suficiente con abrirlo desde Open en el menú File o hacer doble cick en su nombre desde el Current Direcrory. Ambas opciones dan acceso al editor y al fichero.

Ya se ha introducido que existen dos tipos de ficheros .m. Se comentarán con más detalle cada uno de ellos.

3. Ficheros de comandos. Programas propios del usuario

Estos ficheros, también llamados scripts, contienen listas de comandos que se ejecutan sucesivamente cuando se ejecuta dicho fichero, es decir cuando se escribe el nombre del fichero en la línea de comandos y se pulsa intro. Es importante destacar que en estos ficheros, las variables que se crean pertenecen al espacio base del Matlab, algo que no ocurre en los ficheros de función donde las variables pertenecen sólo al espacio de trabajo de esa función. Puede comprobarse esto ejecutando el fichero del la figura 14.

Se ampliará el tratamiento de este tipo de ficheros en el capítulo de programación.

4. Definición de funciones

En el tema 1 vimos que Matlab tiene definidas sus propias funciones, por ejemplo Sin, Cos,... En general, el programa tiene un gran número de funciones incorporadas, bien se trata de funciones intrínsecas, es decir, del propio código ejecutable lo que las hace muy rápidas y eficientes, o bien se trata de funciones definidas en ficheros, normalmente .m, que vienen con el programa. A todo ello se

unirán las funciones creadas por el propio usuario. La importante diferencia de estas últimas es que, para que el programa pueda trabajar con ellas, el correspondiente fichero .m debe estar en el directorio actual o en el path.

El trabajo con funciones y el propio concepto de función en Matlab es parecido al que se tiene en C y en otros lenguajes de programación. Toda función tendrá:

- Un nombre. Por ejemplo: f.
- Unos argumentos. Van a continuación del nombre y entre paréntesis, separados por comas si son más de uno. Por ejemplo: f(x) ó f(x,y).
- Unas salidas o retornos que son el resultado de la función. La ventaja de Matlab es que pueden ser valores matriciales múltiples que se recogerán en diversas variables que se agruparán entre corchetes. Por ejemplo: p=f(x) ó [p,q]=f(x).

4.1. Funciones de librería

Ya conocemos, si no puede consultarse la ayuda, los diversos tipos de funciones que tiene el programa. Se clasifican en:

- Funciones matemáticas elementales.
- Funciones especiales.
- Funciones matriciales elementales.
- Funciones matriciales específicas.
- Funciones para la descomposición y/o factorización de matrices.
- Funciones para análisis estadístico de datos.
- Funciones para análisis de polinomios
- Funciones para integración de ecuaciones diferenciales ordinarias.
- Resolución de ecuaciones no-lineales y optimización.
- Integración numérica.
- Funciones para procesamiento de señales.

Existen funciones (las matemáticas trascendentes y algunas básicas) que actúan sobre escalares o sobre cada elemento de una matriz. Por ejemplo:

```
>> sin(0)
ans =
    0
>> A=[1 2 3]
A =
    1     2     3
>> sin(A)
```

```
ans =
   0.8415    0.9093    0.1411
>>
```

Existen otras que sólo actúan sobre vectores, no sobre escalares ni matrices. Por ejemplo, max(x), min(x) devuelve el elemento máximo y mínimo de entre los elementos del vector x. También nos dan la posición donde se encuentra.

```
>> A=[1 2 3]
A =
    1     2     3
>> max(A)
ans =
    3
>> [p,q]=max(A)
p =
    3
q =
    3
>>
```

De igual forma, otras funciones sólo se aplican sobre matrices. Son las que se encuentran en el grupo de funciones matriciales elementales, funciones matriciales especiales y funciones de factorización y/0 descomposición matricial. Por ejemplo trace(A) que nos da la traza de la matriz A, [p,q]=eig(A) nos da los valores propios y vectores propios asociados a la matriz A.

4.2. Funciones creadas por el usuario

La palabra *function* escrita al comienzo de un fichero .m nos permite definir una función constituyendo una de las aplicaciones más importantes del programa. Su sintaxis es la siguiente:

function parámetros de retorno=nonbre de la función (argumentos)
cuerpo de la función

Es muy importante incidir en que los argumentos o variables de la función son de carácter local, es decir, no interfieren con otras variables del mismo nombre que hayan sido definidas en otra parte del programa, otros ficheros .m o en la ventana de trabajo. Para que una función tenga acceso a variables que no se hayan definido como parte de sus argumentos, éstas deben definirse como globales tanto en el programa

principal como en los ficheros .m dónde se quiera tener en cuenta. El comando es global x,y,.....

El siguiente ejemplo, figura 15, define una función f que calcula el cuadrado del valor que se desee:

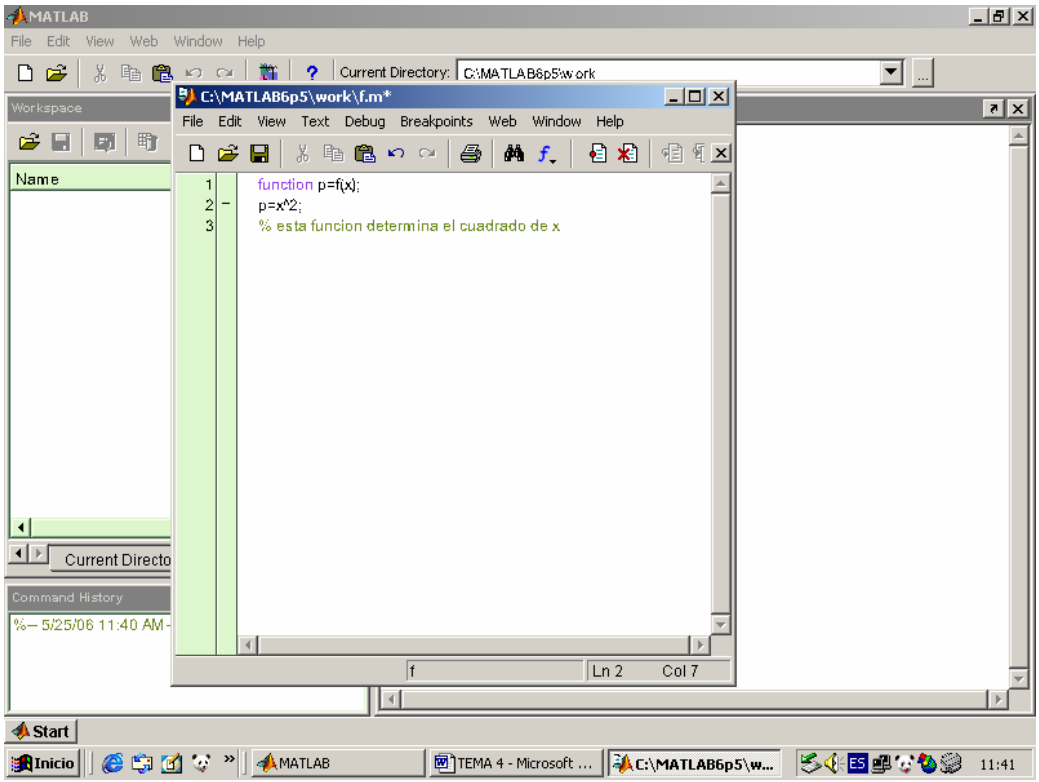

Figura 15

Es importante destacar:

- En este caso existe un argumento o variable de entrada: x, y un retorno o salida: p.
- Si no ponemos los (;) después de la definición de función y el retorno, al ejecutar desde la línea de comandos, el resultado saldría por duplicado o triplicado, el valor de f(x), el de p y el de ans.
- Para finalizar debemos guardar el fichero. Al hacerlo, el programa por defecto nos sugiere como nombre el de la propia función. Es aconsejable usarlo para evitar confusiones entre los ficheros y las propias funciones. También debemos acordarnos de guardarlo en el directorio donde trabajamos para tener acceso inmediato a la función. Si no lo hacemos así, para utilizarla deberemos antes situarnos en el directorio donde la hayamos grabado.

- Es posible poner todos los comentarios necesarios para, en un futuro recordar lo que hace esa función. Si tecleamos en la línea de comandos help seguido del nombre de la función, nos aparecerán los comentarios que en su día escribimos en dicho fichero (figura 16).

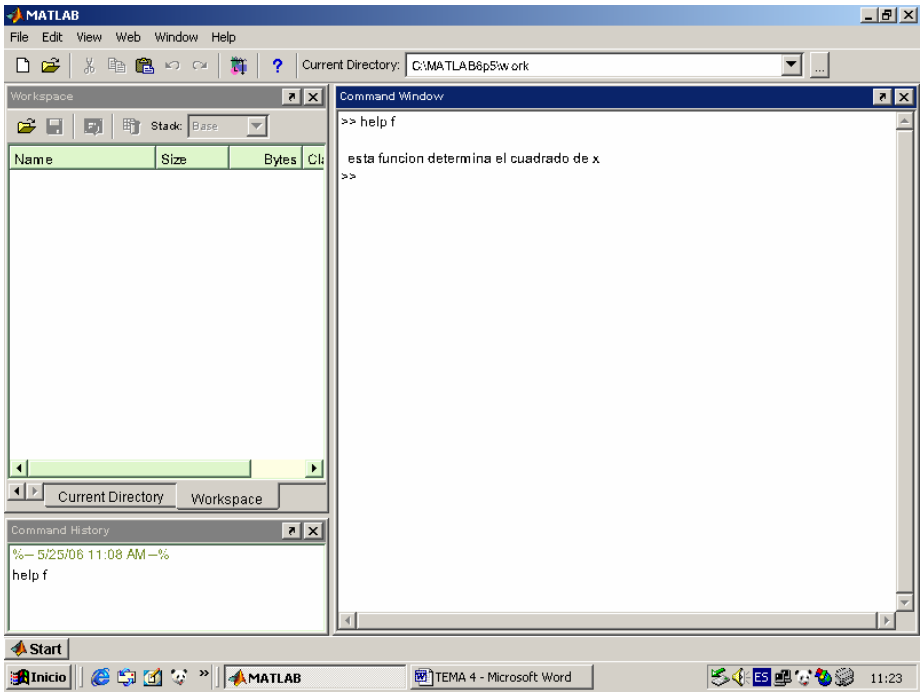

Figura 16

- Destacar también que esta función se aplica sobre cualquier tipo de entrada para la que esté definida la operación. En este caso si lo aplicamos sobre un vector o una matriz no cuadrada nos dará un mensaje de error. Sí se aplica sobre matrices cuadradas calcula el producto de dicha matriz por ella misma. Si queremos que la función f se aplique sobre todo tipo de matrices calculando el cuadrado de cada elemento es suficiente con definir la operación con el punto delante:

```
function p=f(x);
p=x.^2
```

4.2.1. Comandos eval y feval

El comando feval:

La evaluación de una función también puede hacerse a través del comando feval cuya sintaxis es:

feval (‘nombre de la función’, valor del argumento 1, valor del argumento 2, ...)

Por ejemplo, figura 17:

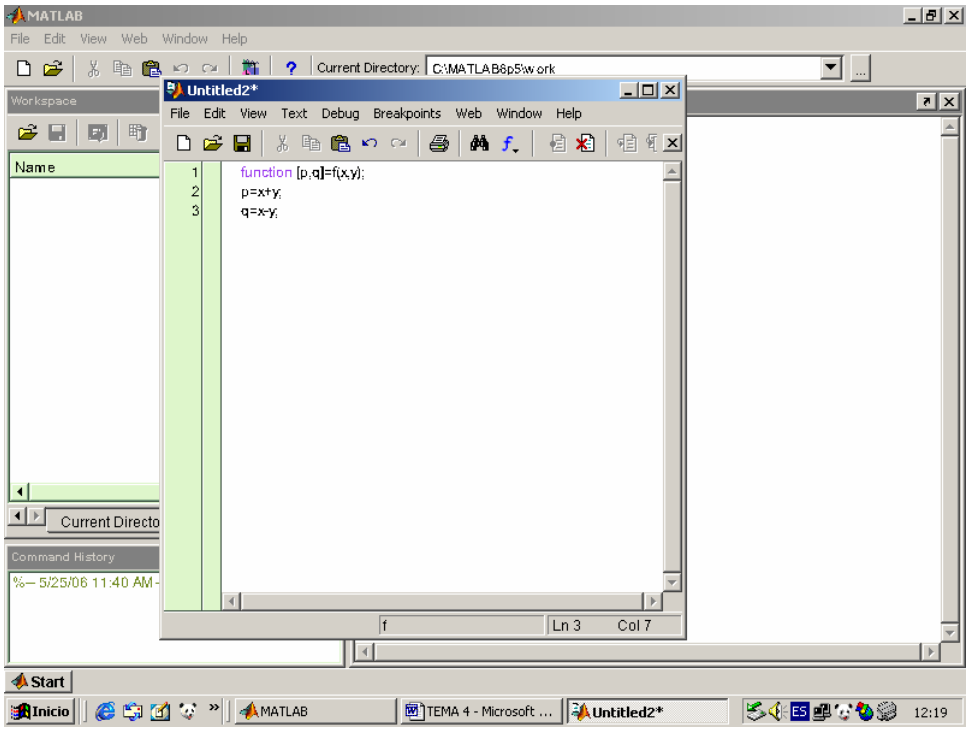

Figura 17

Si ejecutamos en la línea de comandos:

```
>> [a,b]=feval('f',3,4)
a =
     7
b =
    -1
>>
```

El comando eval

Este comando se utiliza para trabajar con cadenas, algo que se comentó en el tema 1.

Supongamos que definimos una función como una cadena de caracteres desde la línea de comandos, para ello se utilizan las comillas:

```
>> f='x^2+5'
f =
x^2+5
```

El comando eval se utiliza como sigue:

```
>> x=3;
```

```
>> eval(f,x)
ans =
   14
>>
```

De forma general se puede hacer:

```
>> x=3;
>> eval('x ^2+5',x )
ans =
   14
>>
```

Existen algunas funciones relacionadas que trabajan de la misma forma. Tal es el caso, por ejemplo de:

diff('f','x'): Calcula la función derivada de f con respecto a x.

diff('f','x',n): Calcula la función derivada enésima de f con respecto a x.

Estos comandos entre muchos otros, forman el trabajo de cálculo simbólico del programa.

Ejemplos:

```
>> diff('x^2','x')
ans =
2*x
```

Si declaramos primeramente la variable x como simbólica:

```
>> syms x;
>> diff('x^2',x,3)
ans =
0
```

Práctica 4: Empezando a trabajar con ficheros .m

1. Construir una función que calcule el seno de cualquier valor más 5. Utilizarla para calcular sen(45)+5, sen(n)+5 con n los 10 primeros números naturales.
2. Repetir el ejercicio anterior para el cálculo del cubo de los valores.
3. Define la función g(x)= $\frac{x^3 - x}{x^2 + 1}$
   a. Calcula g(4).
   b. Calcula g(x) siendo x=(-2, -1.9, -1.8,.....1.8 ,1.9 ,2)
4. Construir una función que calcule el ángulo formado entre dos vectores.
5. Define una función que permute las filas i y j de una matriz.
6. Construir una función ecu(a,b,c) que determine las raíces de la ecuación $ax^2+bx+c=0$.
7. Define como variable global la matriz A= $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$. Define una función que calcule el incremento de todos los elementos de A en un número a elegir.
8. Construir una función tal que, dados dos vectores u, v, devuelva un vector con los elementos de u excepto el primero, y los elementos de v excepto el último.