Práctica 1-2. Programación en Matlab

Matemáticas 2. Ingeniería Informática

Entorno de programación

Introducción. Ficheros M

- MATLAB tiene un poderoso lenguaje de programación
- MATLAB está basado en scripts y funciones: ambos son **códigos (ficheros M) que aceptan entradas (input) de los usuarios y obtienen salidas (output)**. Las funciones no guardan las variables en el workspace
- También se puede introducir comandos en linea de comandos

Entorno de programación

Introducción. Ficheros M

- Para crear un script o una función (**fichero M**) se utiliza el menú *New*
- Puedes usar el editor de MATLAB o cualquier otro para crear **Ficheros M**
- Puedes llamar a los **Ficheros M** como cualquier otro comando e MATLAB. Tecleando su nombre en linea de comandos
- Scripts y Funciones permiten hacer eficiente la computación

Scripts

Script. Definición

- Un script es un conjunto de comandos en un fichero M
- Se ejecutan secuencialmente
- No aceptan argumentos de entrada, ni devuelven argumentos de salida. Operan con datos en el workspace

Scripts

Script. Instrucciones de entrada y salida de datos

- Instrucción para ingreso de datos: $v = input('Mensaje')$. Se muestra mensaje al usuario para que introduzca el dato por teclado
- La instrucción *input* cambia el valor de la variable ans si no se asigna a otra variable
- Instrucción para salida de datos: $disp(v)$. Visualiza el valor de la variable v por pantalla

Scripts

Script. Ejemplo

Crea un script en el que se calcule el área de triángulo dado sus tres lados que deben introducirse por teclado

Primeramente $New -> Script$ y después se teclea

$a = input('Primer\ lado: ');$

$b = input('Segundo\ lado: ');$

$c = input('Tercer\ lado: ');$

$t = (a + b + c)/2;$

$s = sqrt(t * (t - a) * (t - b) * (t - c));$

$disp('La\ superficie\ es: ');$

$disp(s)$

Scripts

Script. Comandos de entrada/salida

- El comando *sprintf* se utiliza para visualizar salidas texto y datos en pantalla
- Si se quiere almacenar datos en un archivo se utiliza el comando *fprintf* (si no se le indica el nombre del archivo se muestra por pantalla)
- A diferencia de *disp*, la salida puede tener un formato preestablecido
- Es útil en la visualización de salidas, pero esto hace que tenga una sintaxis larga en algunos casos
- Permite visualizar mensajes de texto, números y cadenas en la salida, dar formato a la visualización
- El resultado de *sprintf* es un string que se le asigna a una variable (*ans* por omisión)

Scripts

Script. Comandos de entrada/salida

La sintaxis del comando *sprintf* es

sprintf('Mensaje en forma de cadena')

Para controlar la salida hay una serie de caracteres

- \n salto de línea
- \r retorno de carro
- \t tabulador horizontal
- \v tabulador vertical
- \b retroceder un espacio

Nota: el comando *sprintf* no salta automáticamente de línea.

Scripts

Script. Comandos de entrada/salida

Para visualizar texto y datos (valores de variables) juntos, el comando *sprintf* debe utilizarse siguiendo la sintaxis

sprintf('texto $\%-5.2f$ *texto adicional* \n', *var)*

% indica el lugar donde se inserta *var*

$-5.2f$ son los elementos del formato

var es la variable a mostrar

Scripts

Script. Comandos de entrada/salida

En cuanto a los elemento de formato, son:

- El flag o bandera: puede ser $-$ para indicar que justifique a la izquierda, $+$ para que visualice el signo, espacio en blanco para insertar un espacio antes del valor o 0 para que añada ceros.
- Los números especifican el mínimo número de dígitos a imprimir y la precisión del campo.

Scripts

Script. Comandos de entrada/salida

En cuanto a los elemento de formato, son

- la letra es el carácter de conversión y puede valer
  - **c** Carácter simple
  - **s** String de caracteres
  - **i** o **d** Entero en base 10 con signo
  - **u** Entero en base 10 sin signo
  - **o** Entero en base 8 sin signo
  - **x** Entero en base 16 sin signo
  - **e** Notación exponencial en minúsculas (ej. 1.709098e+001)
  - **E** Notación exponencial en mayúsculas (ej. 1.709098E+001)
  - **f** Notación de punto fijo (ej. 17.090980)
  - **g** Formato corto de las notaciones e o f
  - **G** Formato corto de las notaciones E o f

Scripts

Script. Comandos de entrada/salida. Ejemplo

```
>> A = 46 * ones(1,4);
>> text = sprintf('%d %f %e %x\n', A);
text =
46 46.000000 4.600000e+01 2E
```

Scripts

Script. Comandos de entrada/salida. Ejemplo

```
valor = 3.1;
sprintf('El valor es %8.2f metros \n',valor)
sprintf('El valor es %8.2e metros \n',valor)
sprintf('El valor es %i metros \n',fix(valor))
```

Scripts

Script. Comandos de entrada/salida. Ejemplo

Dado un numero entero de dos cifras, mostrarlo con las cifras en orden opuesto.

```
n = input('Numero de dos cifras: ');
d = fix(n/10);
u = mod(n,10);
r = 10*u + d;
fprintf('El número invertido es %u \n',r)
```

Decisiones y bucles

Decisiones

Las decisiones son operaciones que permiten condicionar la ejecución de instrucciones dependiendo del resultado de una expresión cuyo resultado únicamente puede ser **verdadero o falso**.

Decisiones y bucles

Decisiones. La instrucción *if - else*

Se puede condicionar la ejecución de instrucciones dependiendo de si el resultado es verdadero o falso.

if Condicion
    Instrucciones
else
    Instrucciones
end

Para escribir la *Condición* se pueden usar operadores relacionales y conectores lógicos. Para que una expresión pueda ser usada como una condición, las variables incluidas deben tener asignado algún valor, en caso contrario la *Condición* no podrá evaluarse.

Decisiones y bucles

Decisiones. Ejemplo

Dado un numero entero determinar si es par o impar

```
x = input('Numero: ');
r = mod(x,2);
if r == 0
        disp('Par')
else
        disp('Impar')
end
```

Decisiones y bucles

Decisiones. La instrucción *switch*

Cuando hay decisiones múltiples, una estructura alternativa es la instrucción *switch*. Esta permite realizar una o varias instrucciones agrupadas en casos, dependiendo de una variable de control

switch variable de control
    case valor,
        Instrucciones
    case valor,
        Instrucciones
    case valor,
        Instrucciones
    ...
    otherwise,
        Instrucciones
end

Decisiones y bucles

Decisiones. La instrucción *switch*

Esta instrucción en MATLAB da error si se comparan tipos distintos. En Octave en cambio se evalúa a falso. La sección *otherwise* es opcional y se ejecuta cuando no se activa ninguno de los casos incluidos en la instrucción *switch*. Se pueden omitir las comas luego del valor de cada caso si se escriben las instrucciones en la siguiente linea después de *case*

Decisiones y bucles

Decisiones. Ejemplo

Dado un numero entero entre 1 y 7 mostrar el día de la semana

```
d = input('Numero del día: ');
switch d
        case 1, disp('Lunes')
        case 2, disp('Martes')
        case 3, disp('Miércoles')
        case 4, disp('Jueves')
        case 5, disp('Viernes')
        case {6,7}, disp('Fin de Semana')
        otherwise disp('No es el número de un día de la semana')
end
```

Decisiones y bucles

Bucles. La instrucción *for*

En la asignación a la variable del bucle *for* se pone un vector Se especifica el número de veces que se repite el bloque, indicando el valor inicial, el incremento y el valor final. Al entrar y ejecutarse la estructura, el valor de la variable que cuenta el numero de ciclos cambia siguiendo la especificación establecida en el incremento. Mientras el valor de la variable no exceda del valor final, el bloque sera nuevamente ejecutado. Cuando la variable excede al valor final, la ejecución continuar después del bloque. Se pueden anidar

for $i = a : b : c$

Bloque de instrucciones que se repite

end

i = contador, a = valor inicial, b = incremento y c = valor final

Decisiones y bucles

Bucles. Ejemplo

Lista los números naturales impares entre 1 y n, siendo n un dato a introducir por teclado

```
n = input('Valor final: ');
for i = 1 : 2 : n
        disp(i);
end
```

Decisiones y bucles

Bucles. Ejemplo

```
for i = [2, pi, 3 : 5, exp(1)]
        disp(i);
end
```

o

```
for i = ['a','Caracteres']
        disp(i);
end
```

Decisiones y bucles

Bucles. La instrucción *while*

Al entrar a esta estructura, se evalúa la condición. Si el resultado es verdadero (V) se ejecuta el bloque y regresa nuevamente a evaluar la condición. Mientras la condición mantenga el valor verdadero, el bloque de instrucciones sigue ejecutándose. Esto significa que es necesario que en algún ciclo la condición tenga el resultado falso (F) para salir de la estructura y continuar la ejecución después del bloque.

while *Condicion*
    Bloque instrucciones
end

Decisiones y bucles

Bucles. Ejemplo

Simular lanzamientos de un dado. Muestre el resultado en cada intento hasta que salga el 5

```
x = 0;
while x ~= 5
        x = fix(rand * 6) + 1;
        disp(x);
end
```

Funciones

Funciones. Concepto de Función

- Las funciones definidas por los usuarios son similares a las funciones pre-definidas
- Toman cierto numero de entradas, realizan ciertas operaciones, y dan una o varias salidas
- Como las funciones incorporadas de MATLAB, necesitamos conocer que hace la función, y si lo hace correctamente

Funciones

Funciones. Concepto de Función

- Una Función (subrutina, método, procedure, o sub-programa) es un trozo de código, que realiza una tarea específica
- Es como una caja negra, toma entradas y genera salidas
- Tiene variables locales a la Función (invisible en el workspace)

Funciones

Funciones. Funciones anónimas

Es una forma muy flexible de crear funciones sobre la marcha, bien en línea de comandos, bien en una línea cualquiera de una función o script.
La forma general de las funciones *anónimas* es
handle = @(argumentos) expresión
Por ejemplo
$>> mifun1 = @(x)sin(2 * x)$
$>> mifun2 = @(x, t)sin(2 * x * t)$
$>> mifun2(pi/4, 1)$
ans =
1
Una funciones *anónima*, se puede usar como argumento de entrada en otras funciones

Funciones

Funciones. Funciones inline

Otra forma muy flexible de crear funciones sencillas sobre la marcha, es utilizar las funciones *inline* cuyo formato es

nombre-funcion = inline('expresión', 'arg1','arg2',...)

Por ejemplo

$>> g = inline('sin(2 * x * t)','x','t')$

Para evaluar esta función en un punto $g(pi/4, 1)$

Una funciones *inline*, también se puede usar como argumento de entrada en otras funciones

Funciones

Funciones. Tipos de Funciones

- Funciones definidas por los usuarios
- Funciones incorporadas $\rightarrow$ *sin*, *exp*...No se puede visualizar el código
- Funciones proporcionadas por MATLAB $\rightarrow$ linspace, mean... Se puede visualizar el código con el comando **type**

Cada toolbox tiene una lista de funciones especiales que puedes usar

Funciones

Funciones. Sintaxis de las Funciones

- Las Funciones se escriben en un fichero M
- El nombre de la función y el del fichero M deben coincidir obligatoriamente
- La sintaxis es

$$function\ y = myfunction(x)$$

donde x es la entrada de myfunction e y es la salida
- La llamada de la Función definida se hace invocando directamente a la Función: **myfunction(x)**

Funciones

Funciones. Notas sobre las funciones

- Para nombrar funciones se utilizan las mismas reglas que para las variables
- Es importante que se le asignen nombres que signifiquen algo
- Los comentarios son muy importantes en las funciones

Funciones

Funciones. Ejemplos

```
function volumen = volumesphere(radio)
volumen = (4/3) * pi * (radio^3);
```

Funciones

Funciones. Ejemplos

function perímetro = perimetersquare(side)
perimetro = $4 * side$;

Funciones

Funciones. Ejemplos

function raíz = squareroot(x)
root = x^(1/2);

Funciones

Funciones. Uso de las funciones

- A las funciones definidas por los usuarios se accede de la misma forma que las funciones incorporadas
- El fichero M y la función deben tener el mismo nombre y debe estar en el directorio actual
- Se pueden usar tecleando directamente $\rightarrow$ perimetersquare(4)

Funciones

Funciones. Tipos de funciones dependiendo de las entradas y salidas

- Funciones con argumentos de entrada y de salida
- Funciones con argumentos de entrada y sin argumentos de salida
- Funciones con argumentos de salida y sin argumentos de entrada
- Funciones sin argumentos

Funciones

Funciones. Funciones con múltiples entradas

Las Funciones definidas por los usuarios pueden tener varios parámetros de entrada

Ejemplo

function $x = \text{displacement}(x_0, v_0, a, t)$

$x = x_0 + v_0 * t + (1/2) * a * t\text{^}2;$

Funciones

Funciones. Funciones con múltiples entradas y salidas

Ejemplo

function $[a, F] = \text{acelerationcalculation}(v_2, v_1, t_2, t_1, m)$

$a = (v_2 + v_1)/(t_2 - t_1);$

$F = m * a;$

Si tiene dos salidas se deben igualar a dos variables cuando se invoca la función.

$>> [acceleration, force] = accelerationcalculation(v_2, v_1, t_2, t_1, m).$

Funciones

Funciones. Funciones sin entrada

Si se necesita acceder a algunos valores constantes un número determinado de veces, se puede codificar en una función sin entradas. Cuando la constante se necesita se accede vía la función

Ejemplo

function masofearth = moe()

masofearth = $5.976E24$;

>> *sprintf*('La masa de la Tierra es: % e \n',moe)

Funciones

Funciones. Funciones sin salida

Las funciones sin salida se pueden usar para muchos propósitos, tales como dibujar figuras de un conjunto de datos de entrada o imprimir información de salida en el command windows

Ejemplo

function [] = star()

theta = $pi/2 : 0.8 * pi : 4.8 * pi$;

$r = [1\ 1\ 1\ 1\ 1\ 1]$;

$polar(theta, r)$;

$>>$ *star*

Funciones

Funciones. Funciones sin salida

La función incorporada **tic** no tiene salida. Arranca un "timer" para ser usado con otra función incorporada **toc**

Ejemplo

$>>$ *tic*;

$>>$ *pause*(2);

$>>$ *toc*;

Otro ejemplo

$>>$ *timer* = *tic*;

$>>$ *pause*;

$>>$ *toc*(*timer*)

Funciones

Funciones. SubFunciones

- Las SubFunciones se almacenan en el mismo fichero que la función principal
- Se pueden llamar solo desde ese fichero
- Están restringidas al fichero en el cual están definidas

Funciones

Funciones. Ejemplo de subFunciones

function [avg,med] = mystat(u)

$n = length(u);\ avg = mymean(u, n);\ med = mymedian(u, n);\ \ end$

function a = mymean(v,n);

$a = sum(v)/n;\ \ end$

function m = mymedian(v,n);

$w = sort(v);$

$if\ \ rem(n, 2) == 1$

$\quad m = w((n + 1)/2);$

$else$

$\quad m = (w(n/2) + w(n/2 + 1))/2;$

end

end

$>> datos = fix(rand(1, 10) * 100));$

$>> [media, mediana] = mystat(datos)$

Funciones

Funciones. Notas sobre Variables

- Las variables que se crean dentro de una función definida por el usuario, solo puede ser accedida dentro de esa función. Se referencian como variables locales
- Una vez que la función realiza sus operaciones, las variables locales se borran de memoria
- La única variable que aparece en el workspace es la salida de la función (si tiene)

Funciones

Funciones. El comando Type

- El comando *type* seguido del nombre del fichero M imprime el contenido del fichero M en el Command Window
- Esto incluye cualquier función definida por el usuario o función incorporada, así como scripts disponibles en MATLAB

Ejercicios

Ejercicio #1

Crea un script en el que se calcule el área y el volumen de un cono dados su radio de la base y altura que se deben introducir por teclado

Ejercicios

Ejercicio #2

Crea un script que escriba un número natural en una base dada. La nueva base y el número se introducirán por teclado.

Ejercicios

Ejercicio #3

Crea un csript que diga si un número es primo. El número se debe introducir por teclado.

Ejercicios

Ejercicio #4

El precio de una pizza depende de su tamaño según: Tamaño 1 - 5 euros; tamaño 2 - 8 euros y tamaño 3 - 12 euros. Cada ingrediente adicional cuesta 1.5 euros. Lee el tamaño de la pizza y el numero de ingredientes y calcula el precio a pagar (Utiliza la instrucción *if - else -end*)

Ejercicios

Ejercicio #5

Resuelve el problema anterior con la instrucción switch

Ejercicios

Ejercicio #6

Crea una función anónima para la función $f = a + b * x + c * x^2$. Dibuja la gráfica de f para $a = 2; b = 4; c = 3$ utilizando el comando *ezplot*

Ejercicios

Ejercicio #7

Crea una función anónima para la función
$g = a * y^2 + b * x * y + c * x^2$.
Dibuja la gráfica de g para $a = 17; b = 9; c = 11$

Ejercicios

Ejercicio #8

Lista todas las ternas (a, b, c) de numeros enteros entre 1 y 20 que cumplen la propiedad Pitagorica: $a^2 + b^2 = c^2$ pero sin ternas repetidas

Ejercicios

Ejercicio #9

Crea una función que sume los cuadrados de los n primeros números naturales. El número n se le pasa como parámetro (utiliza la instrucción *while*)

Ejercicios

Ejercicio #10

Las ecuaciones que describen el movimiento de caída de los cuerpos son: $v = v_0 + g * t$ y $x = x_0 + v_0 t + \frac{1}{2} g t^2$.
Donde v_0 y x_0 son la velocidad inicial y la posición inicial respectivamente y $g = -9{,}8 m/s^2$ es la aceleración constante de la gravedad.
Escribe una función que admita como parámetro de entrada el tiempo t y devuelva la posición x y la velocidad v de un móvil que se lanza verticalmente hacia arriba desde un acantilado de 300 m de altura con una velocidad inicial de $40 m/s$

Ejercicios

Ejercicio #11

Escribir una función que calcule las dos raíces de una ecuación de segundo grado

$$ax^2 + bx + c = 0,$$

donde a, b**, y** c **son los parámetros de entrada**

Ejercicios

Ejercicio #12

Escribe una función que calcule em máximo común divisor mediante el algoritmo de Euclides. Los dos números naturales se le pasarán como parámetro a la función.