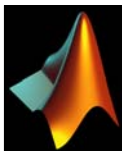


INTRODUCCIÓN AL MATLAB

1



Los componentes individuales

The screenshot shows the MATLAB desktop environment with several windows and components labeled:

- Plataforma de lanzamiento**: Points to the MATLAB Start button in the top-left corner.
- Espacio de trabajo**: Points to the Workspace window, which displays a list of variables: `aspaocal(5)`, `aspaocal(5,2)`, and `aspaocal(5,0)`.
- Ventana histórica de comandos**: Points to the Command History window, which shows a list of executed commands: `a7b/2`, `aspaocal(5)`, `aspaocal(5,2)`, and `aspaocal(5,0)`.
- Directorio Actual**: Points to the Current Directory window, which shows the current directory path: `F:\matlab\12\work`.
- Ventana de Comandos**: Points to the Command Window, which displays the results of MATLAB commands: `a7b =` followed by a 5x5 matrix, `ans =` followed by a 5x1 vector, `aspaocal =` followed by a 5x3 matrix, and `ans =` followed by a 5x1 vector.

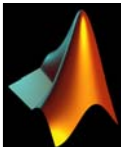
2



Índice

- Introducción a la programación en *Matlab*
 - Variables
 - Operadores
 - Control de Flujo
 - Funciones
- Cómo utilizar *Matlab*
 - Entorno
 - *Debugger*
- Ejercicios I
- *Matlab* como herramienta de cálculo numérico
- Ejercicios II

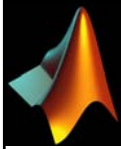
3



Programación en MATLAB

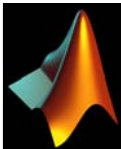
- Lenguaje
 - Un lenguaje intuitivo de alto-nivel
 - Un gran manejo de matrices **MATLAB (Matrix Laboratory)**
- Programación:
 - Interfaz de **Comandos interactivo**
 - Conjunto de comandos en un fichero **.m (Script)**
 - Conjunto de comandos encapsulado en una **función** en un fichero **.m**

4



Variables en MATLAB

- Todas las variables están almacenadas en 32 bit
- No distingue entre variables reales y enteras
- Distingue mayúsculas, en *Matlab* $A \neq a$
- Los nombres tienen que empezar por una letra no obstante se puede usar además de letras nombres y símbolos
- Todo el cálculo se hace en doble precisión `format` puede ser usado para cambiarlo.
- Un “;” al final de la línea no muestra el resultado de la operación
- Permite la definición de variables `globales` y `locales`
- Lista de variables `who`, Borrarlas `clear`, más información `whos`



Vectores MATLAB

vector columna

$$a = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}$$

```
>>a=[1;2;3];
>>a
a =
1
2
3
```

Punto y coma para separar líneas

En general en MATLAB los *arrays* no tienen una dimensión fija

```
>> a = []
```

Vectores fila

```
>>a=[1,2,3];
>>a
a =
1 2 3
```

Coma para separar columnas

```
>> a(0)
```

6



Matrices MATLAB

matriz bi-dim

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
>>a=[1,2,3;4,5,6];
```

```
>>a
```

```
a =
     1     2     3
     4     5     6
```

```
a =
1.0000 + 5.0000i 2.0000    3.0000
4.0000          5.0000    6.0000
```

Asignación

```
>> a(3,3)
```

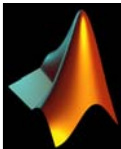
```
??? Index exceeds matrix dimensions.
```

```
a =
     1     2     3
     4     9     6
```

Resultado?

```
>>a(3,3); a(1,1)=1+i*5;
```

7



Operaciones básicas

Creación de matrices

- Creación de matrices mediante el teclado
- Las matrices están definidas por filas
- Los elementos se encierran con corchetes
- Los elementos de una fila se separan por espacios o comas (,)
- Los elementos de una columna se separan por *enter* o punto y coma (;)

```
>> A = [8 9 4, 2 1 5, 0 2 6];
```

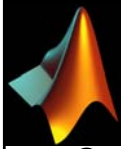
```
A = ( 8, 9, 4, 2, 1, 5, 0, 2, 6)
```

```
>> B = [8 9 4; 2 1 5; 0 2 6];
```

```
B =  $\begin{bmatrix} 8 & 9 & 4 \\ 2 & 1 & 5 \\ 0 & 2 & 6 \end{bmatrix}$  C =  $\begin{bmatrix} 8 & 2 & 0 \\ 9 & 1 & 2 \\ 4 & 5 & 6 \end{bmatrix}$ 
```

```
>> C = B';
```

8



Operaciones básicas

Creación de matrices

- Otras formas de definir matrices

- Matrices predefinidas

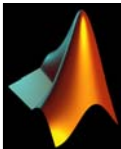
```
>> eye (3)
>> zeros (2)
>> ones (1,5)
>> rand (2, 4)
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$(1 \ 1 \ 1 \ 1 \ 1)$$

$$\begin{bmatrix} 0.9501 & 0.6068 & 0.8913 & 0.4565 \\ 0.2311 & 0.4860 & 0.7621 & 0.0185 \end{bmatrix}$$

9



Operaciones básicas

Creación de matrices

- Otras formas de definir matrices

- Creación de matrices a partir de otras

```
>> B = zeros (size (A))
>> C = diag (x)
>> D = [B, C; h]
>> E = D (1, 4)
>> F = 1: 5
```

$$A = \begin{bmatrix} 1 & 4 \\ 6 & 8 \end{bmatrix}$$

$$x = (4 \ 3)$$

$$h = (4 \ 1 \ 5 \ 2)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

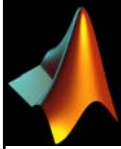
$$C = \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 3 \\ 4 & 1 & 5 & 2 \end{bmatrix}$$

$$E = 0$$

$$F = (1 \ 2 \ 3 \ 4 \ 5)$$

10



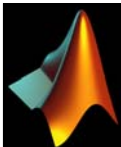
Cadenas de caracteres MATLAB

string

- Texto entre dos simples comillas.
- La variable se almacena como una cadena de caracteres:

```
>>cadena='hola mundo';  
>> cadena(1:4)  
ans =  
hola
```

11



Variables complejas MATLAB

Las variables “i” o “j” se usan típicamente para representar una variable **compleja**; a menos que se hayan usado previamente

Qué pasa en el caso ?

```
>>i=3;  
>> z=23+i*56;  
>>z  
z =
```

```
>> real(z)
```

Qué pasa en el caso ?

```
>>a=sqrt(-1);  
>>z=23+a*56;  
>>z  
z =
```

```
>> imag(z)
```

$$y = \sqrt{x}$$

Para $x < 0$, y es compleja. Matlab asume la introducción de un número complejo. Si no es el caso hay que pasar por una comprobación.

12

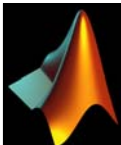


Arrays Multi-dimensionales MATLAB

- En general, en Matlab se suele trabajar con *arrays* de 1 y 2 dimensiones. No obstante, se pueden definir array de más de 2 dimensiones.
- Se pueden construir de diferentes maneras:
 - extensión:
 - `a = [5 7 8 ; 0 1 9 ; 4 3 6];`
 - `a(:, :, 2) = [1 0 4 ; 3 5 6; 9 8 7]`
 - extensión escalar
 - `a(:, :, 3) = 3`

`a(1,1,3) ??? a[:, :, 4]=[1 1] ???`

13



Cells Multi-dimensional

- Cell array son similares a los *arrays* multidimensionales pero los elementos de los cells array pueden tener diferentes tipos.

```
a=cell(1,3);  
a{1,1} = [ 1 2 ; 4 5];  
a{1,2} = 'Name';  
a{1,3} = 2-4i;
```

14



Operadores I - MATLAB

Operadores básicos

Suma	+
potencia	^
resta	-
multiplicación	*
división	
división derecha	/
división izquierda	\

```
>>c=3;b=4;
>>c1=c/b; c1=0.75
>>c2=c\b; c2=1.3333
```

Transpuesta **C=A'**

```
>> a = [1 2]; 3/a
??? Error using ==> /
Matrix dimensions must agree.
```

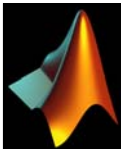
división derecha	./
división izquierda	.\

```
>> a.^2; a^2; ?
```

```
>> 3/0
Warning: Divide by zero.
ans =
    Inf
```

```
>> 0/0
Warning: Divide by zero.
ans =
    NaN
```

15



Operadores II - MATLAB

Operadores condicionales

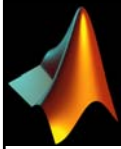
==	igual a
<	Inferior a
>	Superior a
<=	Inferior o igual
>=	Superior o igual
~=	diferente

Operadores logicos

&	and
	or
~	Not
any	True if any element of vector is nonzero
all	True if all elements of vector are nonzero

```
>> a = [1 2 3 4 5 0];
>> any(a) = 1;
>> all(a) = 1;
```

16



Control de Flujo - MATLAB "if"

```

if expression
    statements
end
if expression
    statements1
else
    statements2
end

```

Matlab evalua las *expression* como "Verdadero" o "falso"

"falso" equivale a cero

"verdadero" equivale a un numero no nulo

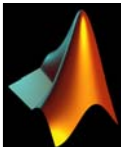
statements: cualquier comando valido de *Matlab*

```

a=5;b=5;
if a+b "verdadero"
if a-b "falso"
>> s=1;
>> if(a<=5)&(b<=-5)
s=2;
end

```

17



Control de Flujo - MATLAB "for"

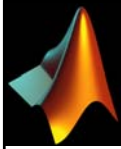
```

for index = start : [increment :] end
    statements
end

```

- *index*, *start*, *increment*, y *end* no tienen que ser enteros
- *increment* es opcional, si no se especifica el *increment*, el valor 1 se toma por defecto
- El valor de *index* puede aumentar (*increment* > 0) ó disminuir (*increment* < 0)
- Los bucles finalizan cuando *index* > *end* (o *index* < *end*)

18



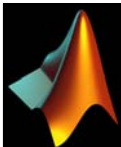
Control de Flujo MATLAB “for/while”

```
F = [];  
for j=1:100  
    if j<=10  
        F(j) = j;  
    else  
        break  
    end  
end  
end
```

```
G = [];  
j=1;  
while(j<=10)  
    G(j) = j;  
    j=j+1;  
end
```

BREAK termina la ejecución del bucle **FOR** y el **WHILE**.
En bucles anidados, **BREAK** sale de el bucle interno =>
RETURN

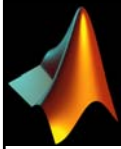
19



Funciones - MATLAB

- Funciones matemáticas comunes
 - Es imposible conocer todos los ficheros .m
- Base que ofrece Matlab
 - Help, helpwin, help browser and
 - <http://www.mathworks.com/>
- Av • Se puede crear su propia función
 - • m-files
 - • Script

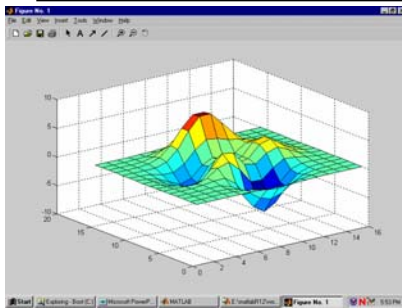
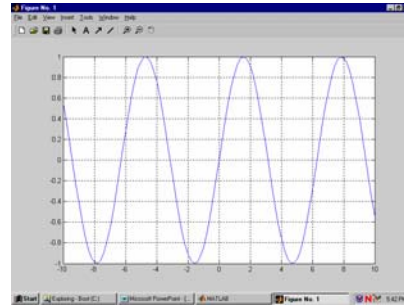
20



Funciones gráficas

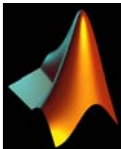
Ejemplos de gráficos

```
>> x=[-10:0.2:10];
>> y=sin(x);
>> close
>> plot(x,y)
>> grid
```



```
>> x=[-3:0.4:3];y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=test3d(X,Y);
>> surf(Z)
```

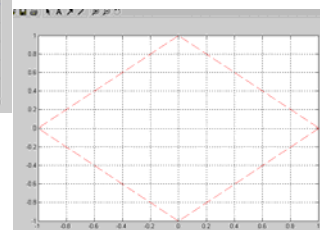
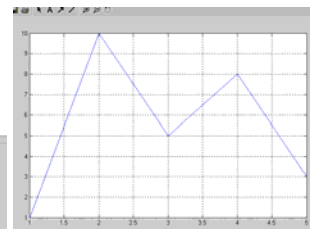
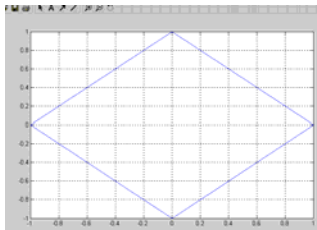
21



Funciones gráficas

Gráficos bidimensionales

- Función **plot**
- **plot(x)**
- **plot(x,y)**
- **plot(x,y,s)**



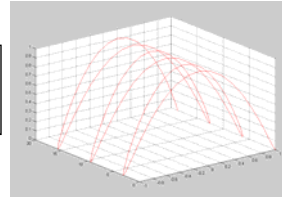


Funciones gráficas

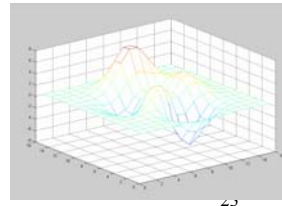
Gráficos tridimensionales

- Función **plot3**

```
>> x=[0:pi/20:6*pi];
>> plot3((cos(x)).^1, x, (sin(x)).^2, 'r'),grid
```

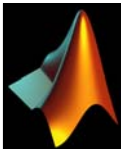


```
>> x=[-3:0.4:3]; y=x;
>> [X,Y]=meshgrid(x,y);
>> function z=test3d(x,y)
z=3*(1-x).^2.*exp(-(x.^2)-(y+1).^2)...
-10*(x/5-x.^3-y.^5).*exp(-x.^2-y.^2)...
-1/3*exp(-(x+1).^2-y.^2);
```



- Función **mesh**

```
>> Z=test3d(X,Y);
>> mesh(Z);
```

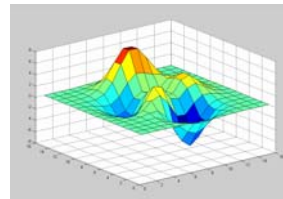


Funciones gráficas

Gráficos tridimensionales

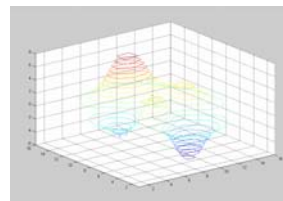
- Función **surf**

```
>> Z=test3d(X,Y);
>> surf(Z);
```



- Función **contour3**

```
>> Z=test3d(X,Y);
>> contour3(Z,20);
```



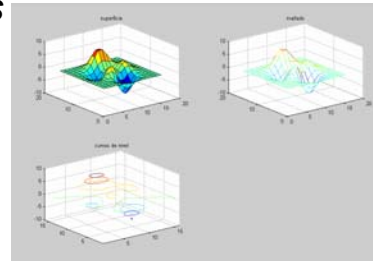
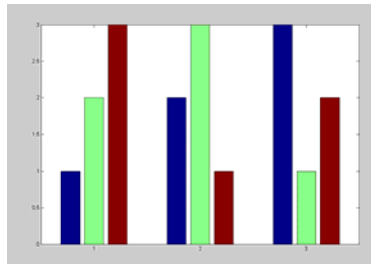


Funciones gráficas

Otras funciones de gráficos

- función **subplot**

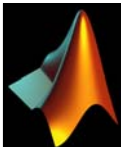
```
>> subplot(2,2,1), surf(Z);  
>> subplot(2,2,2), mesh(Z);  
>> subplot(2,2,3), contour3(Z,20);
```



- función **bar**

```
>> a=[1 2 3; 2 3 1; 3 1 2];  
>> bar (a);
```

25

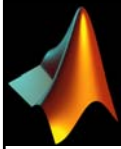


Funciones gráficas

Exportación de figuras

- Se guardan como figuras de MATLAB en ficheros *.fig*.
- Se exportan como *.bmp*, *.tif*, *.jpg*, etc., desde Archivo/Exportar.
- Se copian, por ejemplo, para pegar en *Word* o en un editor de gráficos, desde Edición/Copiar Figura.

26



Script y Funciones - Matlab

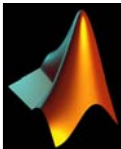
– Script M-files:

- No acepta argumentos de entrada y de salida
- Maneja los datos del *workspace*
- Ventajoso para automatizar una serie de etapas

– Función M-files

- Acepta argumento(s) de entrada y devuelve salida(s)
- Las variables internas son, por defecto, locales a la función; pero pueden ser declaradas globales
- Ventajoso para extender códigos

27

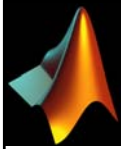


Función I - Matlab

- Colección de comandos que juntos representan una función
- Permiten multi-entrada/multi-salida y cero salida

Salida entre corchetes, []

```
function [A] = inversa(B)
% This function provide the inverse
% when B is square full rank
[m,n] = size(B);
if (m ~= n)
>> nargin('inversa')
ans =
1
>> help inversa
% This function provide the inverse
% when B is square full rank
else
disp ('the inverse of your matrix is: ')
A = inv(B)
end
>> nargout('inversa')
ans =
1
```



script

clear all

clc

format short

% the matrix B can be a global variable or ca Your variables are: **workspace**

B = [1 2 ; 3 4];

fun = input('inv or inversa: ');

switch fun

case 'inversa'

disp('This is our programmed function')

A = inversa(B)

case 'inv'

disp('This is the matlab function')

A = inv(B)

otherwise

disp('you must choose between "inv" and "inversa" ')

end

save results.mat A

colección de comando

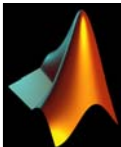
```
>> clear all
>> load results.mat
>> who
```

Your variables are:

```
A
>> A
A =
-2.0000  1.0000
 1.5000 -0.5000
```

[DEMO == callinversa.m](#)

29

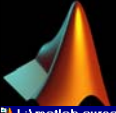


Entrada/Salida

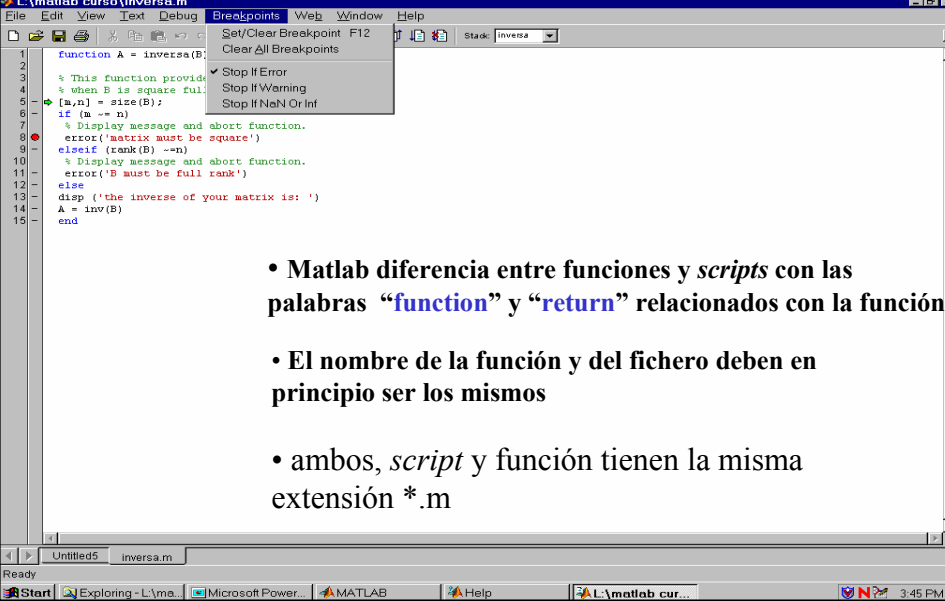
– I/O: parecido al lenguaje C

- fopen,
- fclose,
- fread (binary),
- fwrite (binary),
- fscanf (formatted read),
- fprintf (format write),
- fgetl (read line),
- fgets (read line keep new line character)
- sscanf (string read),
- sprintf (string write)

30



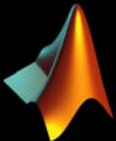
Debugger de Matlab



```

1 function A = inversa(B)
2
3 % This function provides the inverse of a square matrix
4 % when B is square full rank
5 [m,n] = size(B);
6 if (m ~= n)
7     % Display message and abort function.
8     error('matrix must be square')
9 elseif (rank(B) ~= n)
10    % Display message and abort function.
11    error('B must be full rank')
12 else
13    disp('The inverse of your matrix is: ')
14    A = inv(B)
15 end
  
```

- Matlab diferencia entre funciones y *scripts* con las palabras “function” y “return” relacionados con la función
- El nombre de la función y del fichero deben en principio ser los mismos
- ambos, *script* y función tienen la misma extensión *.m



Ejemplo-0: creación de matrices

- Crear una matriz A de tamaño 3*3 donde todos los elementos sean iguales a cero
- Asignar a todos los elementos de la fila 1 un valor de 1 y guardarla como C
- Crear otra matriz B = matriz identidad de 3

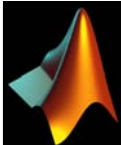
32



Ejemplo-1: sum de matrices

- Crear un función m-file para sumar dos matrices
 - si A y B son del mismo tamaño,
 $C(j,k)=A(j,k)+B(j,k)$
 - si A es un escalar, $C(j,k)=A+B(j,k)$
 - no olvide de manejar las excepciones
- suma A y B

33



Ejemplo-2: *Plotting*

- Crear un vector de tiempo t empezando de cero hasta 10 con un intervalo de tiempo de 1.
- Crear dos vectores x y z
 - $x = \sin(t)$,
 - $z = \cos(t)$
- 2-D vectores: `plot(t, z)`
- 3-D: `plot3(t, x, z)` (curva en el espacio)
- utilizar un *script*

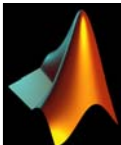
34



Algunos ejemplos de cálculo numérico

- Resolución de sistemas lineales
- Ajuste Polinómico
- Interpolación
- Integración Numérica
- Optimización

35

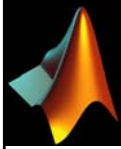


Resolución de un sistema de ecuaciones lineales

$$Ax = y \quad \left\{ \begin{array}{l} A = \begin{bmatrix} 3 & 2 \\ 1 & -1 \end{bmatrix} \\ y = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \end{array} \right.$$

```
A = [3 2; 1 -1];  
y = [-1 1]';  
x = inv(A)*y  
x =  
    0.2000  
   -0.8000
```

36

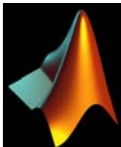
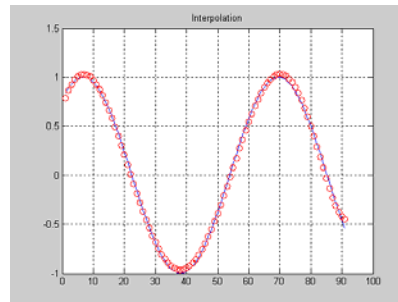


Ajuste polinómico

$$y = 2x^3 + x^2 + 4x + 5$$

```
p = [2 1 4 5];  
x = 1;  
y = polyval(p,x);
```

```
t = 1:0.1:10; x = sin(t);  
y = polyfit(t,x,6)  
j= 1;  
for i = 1:0.1:10  
    x2(j)=polyval(y,i);  
    j = j+1;  
end  
plot(x) , hold on, plot(x2,'or')  
grid on, title('Interpolation')
```



Interpolación

- $X = [0.0 \ 0.25 \ 0.5 \ 0.75 \ 1.0]'$;
- $Y = [0.91 \ 0.81 \ 0.69 \ 0.55 \ 0.40]'$;
- $y = [0.9 \ 0.7 \ 0.6 \ 0.5]'$;
- $x = \text{interp1}(Y,X,y,'linear')$
- $\text{plot}(X,Y,'-sb')$, hold on, $\text{plot}(x,y,'-ok')$,
grid on



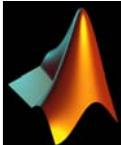
Ecuaciones diferenciales

- La función `ode45` está basada en el método de Runge-Kutta (4,5) explícito de *Matlab*
- Es capaz de resolver numéricamente un conjunto de ecuaciones diferenciales de primer orden
- `[T,Y] = ode45('rigid',TSPAN,Y0,options)`
 - TSPAN = [T0 TFINAL]
 - condiciones iniciales Y0

```
function dy = rigid(t,y)
dy = zeros(3,1);
dy(1) = y(2) * y(3);
dy(2) = -y(1) * y(3);
dy(3) = -0.51 * y(1) * y(2);
```

```
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
```

39



Optimización

- Minimización sin restricciones para resolver un problema bi-dimensional : [min_unc.m](#)

$$\min_x (e^{x_1}) (4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$$

- Minimización con restricciones. Unas desigualdades han sido agregadas al problema anterior: [min_con.m](#)

$$\begin{aligned} 1.5 + x_1x_2 - x_1 - x_2 &< 0 \\ -x_1x_2 - 10 &< 0 \end{aligned}$$

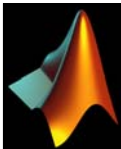
40



MATLAB

- History
 - MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects.
 - First version was released 1984.
 - Current version is version 6 (specifically 6.5 —is available on all platforms)
 - Interactive system whose basic data element is an array that does not require dimensioning
 - UNIX, PC and Mac versions. Similar but differences.

41



Exporting to figures

- `'print -dpsc2 myFig.ps'`
 - Prints current figure window as level 2 postscript file
 - Ready for use in LaTeX: `\epsfig{...}`
 - Other possibilities: `-dpng`, `-dtiff`, `-djpeg`
 - See with `'!gv myFig.ps'`
- Output jpegs -> `mpeg_encode` -> portable movie

42