Introducción a MATLAB

Departamento Matemática Aplicada

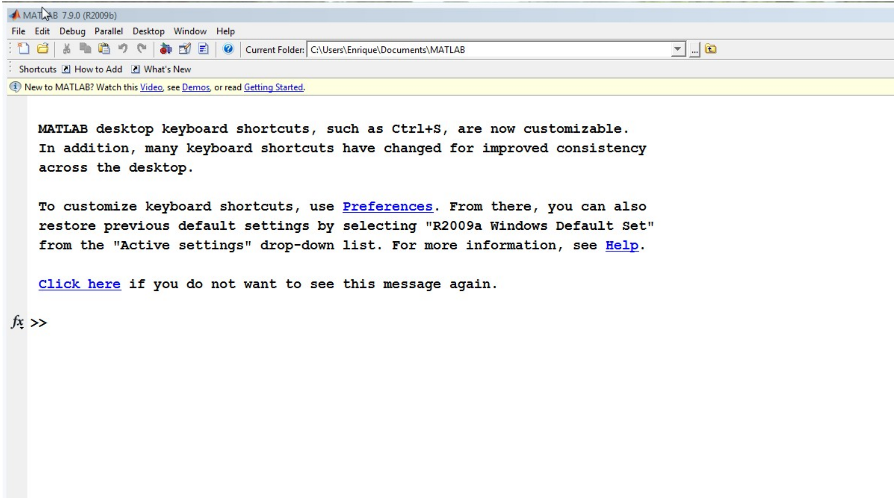
Universidad de Málaga

Curso 2015-2016

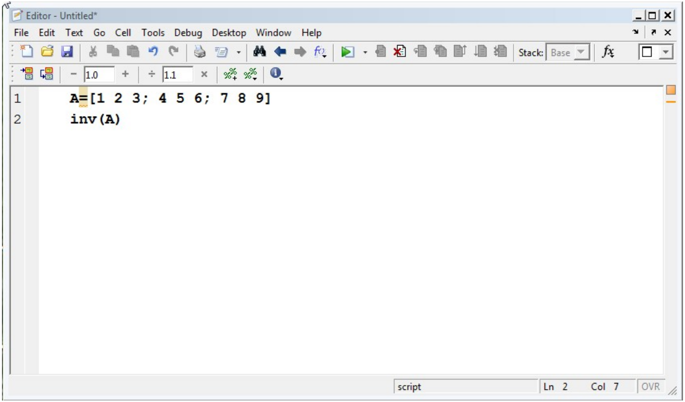
Fundamentos

- MATLAB es un lenguaje de programación implementado por *The MathWorks, Inc.* y disponible en multitud de entornos: Windows, Unix,...
- MATLAB está especializado en *Cálculo Científico*. Integra análisis numérico, computación matricial, procesamiento de señales y un entorno gráfico que permite expresar de forma matemática multitud de problemas.
- MATLAB proviene de *MATrix LABoratory*. Se desarrolla para proporcionar fácil acceso a matrices en los proyectos EISPACK y LINPACK.

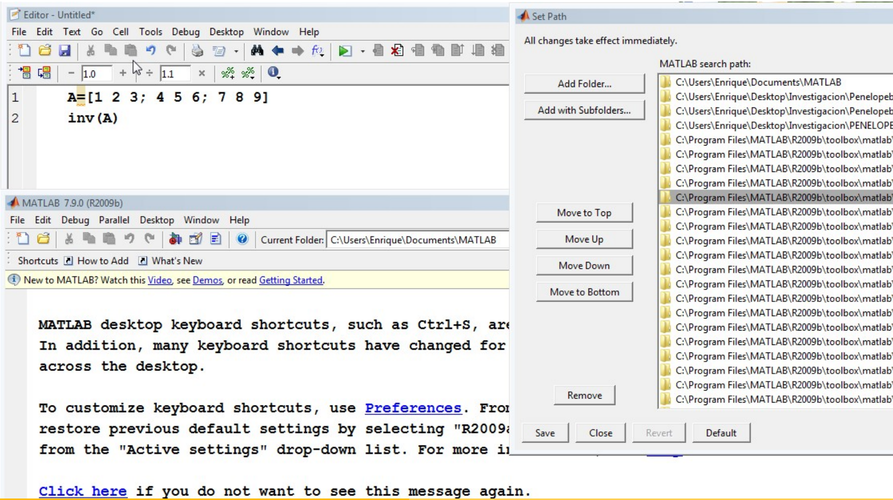
Ventana de Comandos



Editor de textos



Ventanas


Editor - Untitled*
File Edit Text Go Cell Tools Debug Desktop Window Help
A=[1 2 3; 4 5 6; 7 8 9]
inv(A)
MATLAB 7.9.0 (R2009b)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\Enrique\Documents\MATLAB
Shortcuts How to Add What's New
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
MATLAB desktop keyboard shortcuts, such as Ctrl+S, are
In addition, many keyboard shortcuts have changed for
across the desktop.
To customize keyboard shortcuts, use Preferences. Fro
restore previous default settings by selecting "R2009
from the "Active settings" drop-down list. For more i
Click here if you do not want to see this message again.
Set Path
All changes take effect immediately.
MATLAB search path:
Add Folder...
Add with Subfolders...
C:\Users\Enrique\Documents\MATLAB
Move to Top
Move Up
Move Down
Move to Bottom
Remove
Save
Close
Revert
Default

Características

MATLAB proporciona al usuario:

- Gestión dinámica de la estructura de datos matriz rectangular.
- Un conjunto de comandos, funciones y rutinas gráficas muy fáciles de usar.
- La extensibilidad es una de las características más importantes. MATLAB crece constantemente gracias a multitud de matemáticos, ingenieros,... que contribuyen a ampliar las capacidades del lenguaje.
- Herramientas que permiten desarrollar y personalizar las rutinas numéricas.

Existen otros lenguajes similares con software libre: SCILAB, OCTAVE, MAXIMA,. . .
También debemos destacar: MATHEMATICA, R, GAUSS, DERIVE,...

Variables

La variable básica de MATLAB es la matriz compleja, aunque existe la variable cadena (a='qwerty') y dispone de funciones para estas variables. Un número real será una matriz 1×1.

En cuanto a almacenamiento interno, las matrices MATLAB, pueden ser de 3 tipos: enteras, reales o complejas. *La conversión entre tipos es transparente al usuario.*

Los cálculos siempre se hacen con la máxima precisión (16 dígitos), pero la salida por pantalla puede ser:

- formato corto (por defecto) *format short*
- formato largo (long) *format long*
- formato racional (rat) *format rat*
- formato científico corto (short e)
- científico largo (long e)

>>**format compact** evita lineas en blanco.

>>**format long** pone formato largo.

Variables 2

MATLAB distingue entre mayúsculas y minúsculas: Total y TOTAL son variables distintas.
MATLAB tiene las siguientes variables predefinidas:

- **ans**: Variable que almacena el último resultado.
- **eps**: Épsilon de la máquina, o cota superior del error relativo de redondeo al expresar un número real en aritmética de punto flotante.
- **pi**: El número π.
- **inf, NaN**: Infinito y Not a Number (indeterminación).
- **i, j**: Número imaginario $\sqrt{-1}$. CUIDADO: Puede cambiarse.

Las variables pueden ser borradas con el comando **clear**:

- **clear** (borra todas las variables del espacio de trabajo).
- **clear variable1,variable2,. . .** (borra las variables indicadas).
- **clear all** (borra variable, funciones, breakpoints y otros.

Complejos

Una matriz será compleja si lo es algún elemento:

Ejemplo

Introducir $A = [1, 3, 5] + i * [2, 4, 6]$, *o bien*
$A = [1 + 2 * i, 3 + 4 * i, 5 + 6 * i]$

Automáticamente una matriz será compleja si el resultado de un cálculo da un número complejo:

Ejemplo

Introducir $B = [log(-4), 2, sqrt(-2)]$

Características básicas

Ayudas en MATLAB:

- **help**: Lista todas las toolboxes existentes.
- **help sort**: Ayuda sobre el comando sort (ordenar).
- **who**: Lista las variables definidas.
- **whos**: Lista las variables definidas, espacio ocupado, tipo, ...
- **quit**: Salir de MATLAB.
- **info, computer, ver, version** dan información sobre MATLAB, el ordenador en el que estás trabajando, y las versiones de MATLAB y las toolboxes que estás ejecutando.
- **clc**: Limpia la ventana de comandos.

Gestión del entorno

El sistema operativo

- **dir, type, cd, . . .** igual significado que en MS-DOS.
- **quit** abandonar MATLAB.

PATH: La búsqueda de los comandos se hace en el directorio actual y los instalados, pero se puede cambiar esa senda (path) mediante la secuencia de ventanas
"*File*" + "*SetPath*" + "*AddFolder*" .
También puede hacerse mediante comando:
$>>$ **p** $=$ **path**; **path**(**p**,′ **a** :′); aunque no se recomienda.

Diarios

Podemos tener un registro de las órdenes ejecutadas en una sesión de MATLAB y las salidas efectuadas por pantalla.

- **diary nombre_de_fichero** hace que toda la información que aparezca en la pantalla, se envíe al fichero.
- **diary off/on** permitirá enviar o no los resultados.

Ejemplo

Crear un diario con las órdenes y los resultados de un ejercicio.

```
>> diary ejer1.txt,   % Envía a ejer1.txt' en directorio actual.
>> ..., ...,   % órdenes que irán al diario llamado ejer1.txt
>> diary off % las órdenes entre diary off
>> ..., ...,
>> diary on   % y diary on no aparecerán en el diario
>> diary c : \ examen\ejer2.txt
>> ...,   % órdenes que irán al diario llamado ejer2.txt
>> diary off
```

Introducción de una matriz

1) Introducción explícita de sus elementos:

- El comienzo y final de la matriz se indica mediante corchetes.
- Los datos se introducen por filas, separadas por espacios en blanco (uno o varios) o comas.
- Las filas se separan mediante punto y coma (;) o mediante la tecla INTRO.
- Para líneas largas se usan los puntos suspensivos(...), indicando que la línea siguiente es continuación.

Ejemplo

Para introducir la matriz $A = \begin{pmatrix} 1 & 0 & -1 \\ 2 & -1 & 1 \\ 1 & 2 & 0 \end{pmatrix}$ *daremos en* MATLAB *el siguiente comando:*

$>> A = [1, 0, -1; 2, -1, 1; 1, 2, 0]$

Sentencias de asignación

Una instrucción de este tipo es de la forma: **variable=expresión**
Si terminamos la instrucción con el símbolo ';' el resultado se calculará pero no será mostrado en pantalla, lo que acelerará muchos los cálculos.
Varias instrucciones pueden situarse en la misma línea separadas por comas o por puntos y comas.

Ejemplo

A=3*B; C=2*A+3*B,D=A+B-C;D+C
Se calcularán A, C y D pero sólo se mostrará el resultado de C y de la suma D+C (no asignada a ninguna variable).

Operaciones con matrices

- **A+B, A-B, A*B, p*A (p un escalar)** son la suma, resta, producto de matrices y matriz por escalar. Devuelve un error si las matrices no tiene dimensión compatible.
- **A $\backslash$ b, A/b** realizan la división directa y división inversa respectivamente de matrices, esto es, $A^{-1} * b$ y $A * b^{-1}$. Es importante que **x** $=$ **A** $\backslash$ **b** nos da la 'solución' del sistema: A*x=b, mientras que **X=A/B** nos da $X = A * B^{-1}$ solución a *X*B=A*. Cuidado: Si el sistema es imposible da la solución mínimos cuadrados.
- **Aˆ p** (*p* un escalar) devuelve la potencia p-ésima de A. Si *p* es natural mayor que *1* se calcula por productos sucesivos. Si no se calcula por autovalores y autovectores.

Ejemplo

Ejemplo

Dados $A = \begin{pmatrix} 1.7 & -1.1 & 4.5 \\ 1.2 & 1.2 & 0 \\ -1 & 7 & -3 \end{pmatrix}$ *y* $\vec{b} = \begin{pmatrix} 2 \\ 1.1 \\ -0.7 \end{pmatrix}$, *hallar:*
A^2, A^3, $Y = 7I + 2A + 3A^2 + A^3$, A^{-1}, $A^{-1}\vec{b}$ *y la solución* $\vec{x}$ *de* $A\vec{x} = \vec{b}$

Ejemplo

Con la misma matriz A y vector $\vec{b}$, *ejecutar:*
D=diag(diag(A)),L=tril(A)-D,R=triu(A)-D, BJ=-inv(D)(L+R), CJ=inv(D)*b, aut=eig(A), [v,d]=eig(A)*
realizando un diario de la ejecución.

Submatrices

Podemos referirnos a un elemento:

Ejemplo

A(2,3)=5, b=A(3,1)

o a una submatriz:

Ejemplo

B=A([1,2],[2,3]), A([2,3],[2,3])=[1 3; 3 1]

Las matrices pueden crearse mediante submatrices:

Ejemplo

$$>> B = 3 * A + A\hat{\ }2, \qquad B = -inv(D) * (L + R)$$
$$C = [A \quad B; eye(A) \quad zeros(A)]$$
$$>> AA = [A \quad eye(A)]$$

Introducción de una matriz desde fichero

Puede cargarse desde un fichero de texto de nombre 'matriz.m' que contiene la línea: $A = [1, 0, -1; 2, -1, 1; 1, 2, 0]$.
La introducción de la instrucción:
$>>$ **matriz**
produce el mismo resultado que si la lo hacemos directamente en el intérprete MATLAB.

Podemos guardar la matriz mediante
$>>$ **save matriz**
y posteriormente recuperarla mediante:
$>>$ **load matriz**

Matrices especiales

- **ones(m,n)**: Matriz de unos de m filas y n columnas.
- **zeros(m,n)**: Matriz de ceros de m filas y n columnas.
- **rand(m,n)**: Matriz $m \times n$ de números aleatorios distribuidos uniformemente en (0,1).
- **randi([Imin,Imax],m,n)**: Matriz $m \times n$ de números aleatorios distribuidos uniformemente en los enteros entre Imin e Imax.
- **randn(m,n)**: Matriz $m \times n$ de números aleatorios distribuidos según la normal de media 0 y desviación típica 1.
- **eye(n)**: Matriz identidad de orden n.
- **size(A)**: Un vector [m,n] con las filas y las columnas de A.
- **length(v)**: Longitud de un vector.
- **A'** : Matriz traspuesta (conjugada).
- **A.'** : Matriz traspuesta (sin conjugar los elementos).

Operaciones y funciones matriciales

- **inv(A)** calcula A^{-1}.
- **det(A)** calcula el determinante.
- **trace(A)** calcula la traza de A.
- **norm(A)** calcula la norma de la matriz.
- **poly(A)** proporciona el polinomio característico de una matriz.
- **expm(A)** calcula e^A (función matricial).
- **sqrtm(A)** calcula $\sqrt{A}$.
- **logm(A)** calcula la matriz logaritmo neperiano.
- **[V,D]=eig(A)** calcula los autovalores y autovectores de A.

Otras funciones matriciales

Sea p un vector y A una matriz.

- **rot90(A)** gira la matriz A.
- **fliplr(A)** pasa la primera columna al final.
- **flipud(A)** pasa la primera fila a final.
- **reshape** redimensiona matrices.
- **vander(p)** matriz de Vandermonde con penúltima columna p.
- **diag(p,k)** da una matriz que tiene por diagonal k-esima el vector p.
- **diag(A,K)** da un vector, que es la diagonal k-ésima de A.
- **diag(diag(A))** produce una matriz con todos los elementos cero excepto los de la diagonal que son los de A.
- **tril(A)** pone a ceros los elementos por encima de la diagonal.
- **triu(A)** análogo al anterior pero con ceros bajo la diagonal.
- **lu(A)** produce 3 matrices L, U, P tales que LU=PA.

Vectores

Son matrices 1xn, o bien, nx1. Se crean:

- Por enumeración de sus elementos.
- Usando la notación rango: *Valor Inicial* : *paso* : *Valor Final*
- A partir de otros u otra operación que produzca un vector: $y = 3 * x + sin(2 * pi/5)$

Ejemplo

$>> \mathbf{x} = [\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}, \mathbf{10}]$
$>> \mathbf{y} = \mathbf{1} : \mathbf{10}$
$>> \mathbf{z} = \mathbf{1} : \mathbf{2} : \mathbf{10}$
$>> \mathbf{u} = \mathbf{x} + \mathbf{sin}(\mathbf{y})$
$>> \mathbf{v} = \mathbf{linspace}(\mathbf{0}, \mathbf{5}, \mathbf{12})$
$>> \mathbf{w} = \mathbf{diff}(\mathbf{x})$ *(diferencia entre elementos)*

Gráficas de funciones

Ejemplo

Representar gráficamente las funciones $y = sen(\pi x) + 2cos(\pi x)$ *e* $y = x^2$ *en el intervalo* $[-3, 3]$

xx=-3:0.01:3; $y1 = sin(pi * x) + 2 * cos(pi * x)$; $y2 = x.\hat{}2$;
plot(x,y1,x,y2),grid

Operadores lógicos, predicados

Las constante lógicas vienen dadas por *0* (que indica falso) y *1* (que indica cierto).

Los operadores relacionales $<, <=, >, >=, ==, \sim=$ operan elemento a elemento en el caso de matrices, así como las conectivas lógicas &, $|$, y $\sim$, respectivamente 'and', 'or' (ALT+1) y 'not' (ALT+4).

Ejemplo

Si x=[2, 3, 4, 1, 1,-2, 3] e y=[2, 2, 0, 2, 1, 1, -1]

$>>$ $\mathbf{z} = (\mathbf{x} < \mathbf{3})$ *produce z=[1,0,0,1,1,1,0]*

$>>$ $\mathbf{u} = (\mathbf{x} == \mathbf{y})$ *produce u=[1,0,0,0,1,0,0]*

$>>$ $\mathbf{w} = (\mathbf{z}\&\mathbf{u}) \mid (\sim \mathbf{z}\& \sim \mathbf{u})$ *produce w=[1,1,1,0,1,0,1]*

$>>$ $\mathbf{C} = (\mathbf{A} \sim= \mathbf{B})$ *produce una matriz C con '1' donde sean distintos* a_{ij} *y* b_{ij}*.*

Tabla verdad

Ejercicio

Crear la tabla de verdad de las proposiciones lógicas: $(p \vee \bar{q})$, $(\bar{r} \vee q)$ *y* $(p \vee \cup \bar{q}) \wedge (\bar{r} \vee q)$

p=[zeros(1,4) ones(1,4)];
q=[zeros(1,2) ones(1,2) zeros(1,2) ones(1,2)];
x=[0,1]; r=[x x x x];
$a = p \mid (\sim p); b = (\sim r) \mid q; c = a\&b;$
TABLA=[p' q' r' a' b' c']

Operaciones elemento a elemento

$$X.Operador\ Y \qquad \textbf{(X e Y matrices)}$$

Realizará la operación indicada de los elementos que se encuentren en la misma posición.

Ejemplo

$>> A = X.*Y$, *calcula* $A_{ij} = X_{ij} * Y_{ij}$.

Ejemplo

Si $M = \begin{pmatrix} 2 & 3 \\ 2 & 2 \end{pmatrix}$ *y* $N = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ *entonces:*

$$M * N = \begin{pmatrix} 5 & 3 \\ 4 & 2 \end{pmatrix}, \qquad M.*N = \begin{pmatrix} 2 & 0 \\ 2 & 2 \end{pmatrix}$$

Operadores punto

$$X.Operador\ Y \qquad (Producto,\ División,\ Potencia)$$

Las funciones de MATLAB operan elemento a elemento:

- *exp(A)* calcula una matriz con los elementos $e^{A_{ij}}$
- *sqrt(A)* calcula una matriz con los elementos $\sqrt{A_{ij}}$
- *log(A)* calcula el logaritmo neperiano elemento a elemento.
- *abs(A), sin(A), cos(A)...*

Polinomios

Para representar un polinomio en MATLAB usaremos vectores. El polinomio

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x_1 + a_0$$

se representa en MATLAB como

$$p = [a_n, a_{n-1}, \ldots a_1, a_0]$$

Por ejemplo: si queremos representar en MATLAB el polinomio: $P(x) = 3x^2 + 4x + 5$ lo almacenaríamos de la siguiente forma:
$>> p = [3, 4, 5]$.
Y para $q = x^4 - 5x^3$ lo haremos mediante
$>> q = [1, -5, 0, 0, 0]$.

Operaciones con polinomios

Las operaciones típicas para manipular polinomios son:

- $P(x)+Q(x)$, $P(x)-Q(x)$: Se ponen como $>>$ $\mathbf{p}+\mathbf{q}$, $>>$ $\mathbf{p}-\mathbf{q}$, pero hay que tener mucho cuidado en usar vectores de igual longitud.
- $P(x)\cdot Q(x)$: Se calcula mediante $>>$ $\mathbf{conv}(\mathbf{p},\mathbf{q})$
- $\frac{P(x)}{Q(x)}$: Con $>>$ $[\mathbf{c},\mathbf{r}]=\mathbf{deconv}(\mathbf{p},\mathbf{q})$
- $P'(x)$: En MATLAB como $>>$ $\mathbf{polyder}(\mathbf{p})$
- $\displaystyle\int P(x)dx$: En MATLAB como $>>$ $\mathbf{polyint}(\mathbf{p})$
- Evaluar $P(x)$: En MATLAB como $>>$ $\mathbf{polyval}(\mathbf{p},\mathbf{x})$
- Las raíces de $P(x)=0$: En MATLAB como $>>$ $\mathbf{roots}(\mathbf{p})$
- Polinomio con raíces dadas: $>>$ $\mathbf{poly}(\mathbf{x})$

Ejercicios

Poner las instrucciones MATLAB para:

- Expresar los polinomios: $P(X) = -x^4 + x^2 - 1$ y $Q(x) = (1 - x^2)^2$.
- Hallar los puntos en que $P(x) = 2$.
- Evaluar $P(3)$, $P(x)$ siendo x=1,1.1, 1.2,. . . ,3.
- Hallar los puntos en que se cortan, es decir, resolver $D(x) = P(x) - Q(x) = 0$.
- Hallar su producto $C(x) = P(x) * Q(x)$.
- Hallar el cociente y resto de P entre Q.
- Resolver la ecuación algebraica: $P(x) * Q(x) + x^2 - 2 = 0$.
- Calcular $P(5)$, $P'(5)$ y $P''(5)$.
- ¿Cómo obtenemos $IP = \int P(x)dx$ y que forma tomará.
- Calcular $I = \int_2^4 P(x)dx$

Ejercicio resuelto

Ejemplo

Dado el polinomio: $p(x) = x^4 - 3x^3 + 2x - 5$

- *Hallar sus raíces.*
  $>>\ \mathbf{p} = [\mathbf{1}, -\mathbf{3}, \mathbf{0}, \mathbf{2}, -\mathbf{5}], \mathbf{r} = \mathbf{roots}(\mathbf{p})$
- *Representarlo gráficamente en [-5,5].*
  $\mathbf{xx} = -\mathbf{5} : \mathbf{0.01} : \mathbf{5}; \mathbf{y} = \mathbf{polyval}(\mathbf{p}, \mathbf{xx}); \mathbf{plot}(\mathbf{xx}, \mathbf{y}), \mathbf{grid}$
- *Hallar el área delimitada entre el eje OX y las 2 raíces reales que posee.*
  $\mathbf{Ip} = \mathbf{polyint}(\mathbf{p}), \mathbf{Area} = \mathbf{polyval}(\mathbf{Ip}, \mathbf{r}(\mathbf{1})) - \mathbf{polyval}(\mathbf{Ip}, \mathbf{r}(\mathbf{2})).$
  NOTA: A estas alturas sabemos que $r_1 \approx 2.96$ *y* $r_2 \approx -1.2$ *son las raíces reales de* $p(x)$.

Funciones

Como en todos los lenguajes las funciones pueden ser de librería o definidas por el usuario.
Las funciones de librería se cargan al iniciar MATLAB y son las **básicas** del lenguaje y **externas** estando incluidas en las TOOLBOXES instaladas.
Cualquier usuario puede añadir sus propias funciones fácilmente.
Tanto los parámetros de llamada, como los resultados de la función son matrices.
Una misma función **puede devolver uno o varios argumentos**, según se realice la llamada y recibir diferente número de parámetros de entrada.

- **d=eig(A)** devolvería en d los autovalores de A.
- **[v,d]=eig(A)** devuelve los autovectores en v y los autovalores en d.
- **help eig** indica todas las posibles formas de llamar a la función *eig*.

Definición de funciones

La forma es:

function $[\mathbf{y_1}, \ldots, \mathbf{y_m}] = \mathbf{nombre\_funcion}(\mathbf{x_1}, \ldots, \mathbf{x_n})$
sentencias

Ejemplo

Definir la función en MATLAB *que calcule la siguiente función:*

$$alfa(x) = \frac{7x^2 - sen(x)}{2x + 3}.$$

```
function y=alfa(x)
y=(7*x.^2-sin(x))./(2*x+3);
```

Luego mediante

$>> \mathbf{y} = \mathbf{alfa}(\mathbf{7}), \mathbf{y2} = \mathbf{alfa}([\mathbf{123}]), \mathbf{x} = \mathbf{1} : \mathbf{10}; \mathbf{y3} = \mathbf{alfa}(\mathbf{x})$

podremos obtener su valor

Características

- Las *variables* que se declaren dentro de la función son **locales**.
- La única conexión entre el espacio de trabajo y el cuerpo de la función se hace a través de lista_entrada y lista_salida.
- La instrucción **return** provoca la terminación inmediata de la ejecución del fichero.
- **%** se emplea para introducir comentarios. Los comentarios que se escriban hasta la primera línea de código, se devuelven cuando se solicita ayuda mediante **help nombre_función.**
- **nargin** y **nargout**: número de parámetros de entrada y de salida (respectivamente).

funciones INLINE

Conviene ver: **help inline**.
INLINE(EXPR) construye un objeto función 'inline' de la expresión MATLAB contenida en EXPR.
Ejemplos:
g = inline('t^2')
g = inline('sin(2*pi*f + theta)', 'f', 'theta')

Ejemplo

Introduce en MATLAB *la función* $y = e^{-x/4} sen(x)$ *utilizando la orden inline y la función 'fplot' para dibujarla en* $[0, 10]$
$>>$ $\mathbf{F = inline('exp(-x./4).*sin(x)')}$
$>>$ $\mathbf{fplot(F, [0, 10]); grid\ on}$

Scripts

Contienen instrucciones MATLAB y para ejecutarlos basta con poner su nombre. Se crean y modifican con el editor de textos. Se usan para introducir datos iniciales (matrices grandes) y organizar los cálculos (programas).

Ejemplo

Se crea un fichero de texto de nombre 'ej1.m' con las órdenes:

```
x=1:10;
y=x.^2-2;
plot(x,y)
grid on
```

Lo llamaremos mediante >> **ej1** *y dibujará la gráfica.*

Instrucciones de Entrada/Salida

- **echo** visualiza el comando en ejecución.
- **disp** visualiza texto o números en la pantalla.
- **input** permite visualizar un texto y obtener datos desde teclado.
- **keyboard** permite leer y modificar variables desde teclado.
- **pause** detiene la ejecución un tiempo determinado o hasta pulsar una tecla.
- **return** provoca la terminación de la ejecución del fichero y devolución del control a la instrucción llamante.
- **menu** permite generar un menú de opciones.

Funciones de orden superior y manejo de excepciones

Para pasar funciones como argumento a otras funciones.

- **feval** evalúa la función cuyo nombre se ha pasado como primer parámetro con los argumentos que se han pasado como parámetros adicionales.
- **eval** permite interpretar cadenas de texto que contienen expresiones MATLAB válidas.
- **error** sirve para dar un mensaje de error desde un fichero, y muestra su argumento en pantalla.

Ejemplo

```
>>x=[4 8 16];
>>fun1='sqrt';
>>feval(fun1,x)
>>ans= 2.0000 2.8284 4.0000
```

Funciones matemáticas

Las funciones matemáticas en MATLAB, están dirigidas a vectores, así: $>>$ **x** $=$ **[0 pi/2 pi]**, **sin(x)**, produce la salida:
$>>$ **ans** $=$ **0 1 0**
Una lista de funciones elementales puede verse mediante 'help elfun'

- Funciones **trigonométricas** elementales incluidas son:
  sin, cos, tan, asin, acos, atan, asinh, acosh, atanh, . . .
- Otras funciones elementales son:
  abs, angle, sqrt, exp, log, log2, log10, round, fix, ceil, floor, rem, sign . . . .
- Funciones con salida un vector: **diff, linspace, find, sort . . .**
- Funciones **estadísticas:** Entre ellas están **max, min, sum, prod, sort, cumsum, cumprod, mean, median, std, cumsum, cumprod, hist, corrcoef, cov, var, corr, skewness, kurtosis, rand, randn, randi, geomean, harmmean, mad, prctile, boxplot, . . .**

Predicados

- **any(x)** devuelve cierto si algún elemento del vector *x* no es cero.
- **all(z)** devuelve cierto solo si los elementos de *x* son distintos de 0.
- **find(x)** devuelve las posiciones de los elementos distintos de cero del vector *x*.
- **exist(var)** devuelve cierto si la variable existe.
- **isnan(A)** devuelve cierto donde vale *NaN* y cero donde no.
- **finite(A)** devuelve cierto en los valores finitos y 0 donde no lo sea.
- **isempty(A)** devuelve cierto si es una matriz vacía.

Sentencias de control de flujo

Sentencia IF ... THEN ... ELSE:

```
if condición,
   sentencia-1
elseif condición2
   sentencia-2
else
   sentencia-3
end
```

Cuidado: La condición debe dar un valor, no un vector o matriz.
Estaría mal:

x=[2 3 4], y=[1 3 2], if x==y, z=1, else z=x-y, end

que podría hacerse mediante:

if sum(abs(x-y))==0, z=1, else z=x-y, end

Case

Sentencias CASE, SWITCH

switch Variable,
case 1, Sentencias para Variable=1.
case 2, Sentencias para Variable=2.
. . .
case 8,10,12, Sentencias cuando Variable vale 8, 10 ó 12.
otherwise
error('No esta contemplado el caso.')
end

Si la variable de conmutación vale 1, ejecuta la 1ª, si vale 2 la 2ª, etc.

Bucles

Sentencia FOR:

```
for matriz
   sentencias
end
```

Sentencia WHILE:

```
while condición
   sentencias
end
```

La instrucción *break* provoca la salida del bucle, o del último bucle si están anidados.

Ejemplo

Ejemplo

1. **for k=1:4, y(k)=x(k+2)-x(k),end**
2. **k=1:4, for k, y(k)=x(k+2)-x(k),end**
3. **k=1, while k**<=4, **y(k)=x(k+2)-x(k), k=k+1, end**

Ejemplo

Hallar el logaritmo en base 4 de los 1.000 primeros naturales. Se puede hacer con los dos métodos siguientes:

1. **for k=1:1000, y(k)=log(k)/log(4);end**
2. **k=1:1000; y=log(k)/log(4);** *% Mucho más eficiente.*

Comandos gráficos

- **plot(x,y)** representa la tabla de puntos (x_i, y_i).
  $>> \ \mathbf{x} = \mathbf{1} : \mathbf{0.5} : \mathbf{10}; \mathbf{y} = \mathbf{sin}(\mathbf{x}); \mathbf{plot}(\mathbf{x}, \mathbf{y}), \mathbf{grid}$
  representa la función $y = sen(x)$.
- **fplot('funcion', [a, b])** dibuja una función.
- **hold on/off** Permite superponer dos o más gráficas.
- **grid on** activa una cuadrícula en la figura.
- **whitebg** cambia el color de fondo.
- **title, xlabel, ylabel, xcolor, ycolor,. . .** consultar *help comando*.
- **shg** permite visualizar el último gráfico en la pantalla.
- **clf, clg** borran la pantalla de gráficos.
- **subplot(m,n,p)** permite crear varias gráficas en una misma figura.
- **figure** abre una nueva ventana de gráficos.

Ejemplo

Ejemplo

Dibujar el seno y el coseno en la misma figura.

$>>$ **fplot('sin(x)', [-2 * pi, 2 * pi])**

$>>$ **hold on, fplot('cos(x)', [-2 * pi, 2 * pi])**

La siguiente orden hace lo mismo

$>>$ **fplot('[sin(x), cos(x)]', [-2 * pi, 2 * pi])**

Representar gráficamente en $[0, 10]$ la función $y = e^{-x/4} sen(x)$ introduciendo en MATLAB la función. Expresar los valores en una tabla.

function y=fun22(x)
y=exp(-x./4).*sin(x);

y luego ejecutar:

$>>$ **x = 0 : 0.01 : 10; y = fun22(x); plot(x, y); grid**

$>>$ **[x', y']**

Ejercicios

1. Crear una matriz 5x10 aleatoria siguiendo una normal de media 3 y desviación típica 2 y calcular la proporción de los que son mayores estrictos que 4.
2. Introducir la matriz $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}$ y la $B = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{pmatrix}$ calculando A*B, B*A, y los autovalores de A, de B y A*B.
3. ¿Son los autovalores de B iguales a los de su matriz traspuesta? ¿Ocurre lo mismo con los autovectores?
4. Dada una matriz cuadrada C cualquiera ¿Coinciden X=C*C' e Y=C'*C? ¿Resultan siempre simétricas X e Y? Poner algún ejemplo.

Ejercicios-2

3. Usando las funciones de álgebra lineal, ¿Cómo podemos obtener la siguiente matriz? $\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \\ 0 & 1 & 5 & 1 & 0 \\ 0 & 0 & 1 & 7 & 1 \\ 0 & 0 & 0 & 1 & 9 \end{pmatrix}$
4. Dado el vector v=[1 2 3], usando las funciones de álgebra, qué instrucción debemos ejecutar para obtener: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$
5. Interpolar 7 elementos en progresión aritmética entre 0 y 20.

Ejercicios-3

1. Representar conjuntamente las gráficas de las funciones $y = f(x) = sen(x) + sen(2x)$ y de $y = g(x) = cos(x) + cos(2x)$. ¿Se cortan en algún punto? ¿Cuántos cortes hay en $[0, 2\pi]$? Acotar el mayor de ellos en un intervalo de amplitud 1 centésima.
2. Hacer la tabla de verdad de $(p \wedge \bar{q}) \vee [(\bar{p} \vee q) \vee \bar{q}]$ ¿Es una tautología (siempre es cierto)?
3. Generar 10000 números aleatorios del 0 al 9. ¿Cuántos han salido de cada clase? ¿Calcular las frecuencias relativas? Pista: n(k)=sum(x==k)
4. Agrupar los datos anteriores como tabla con frecuencias relativas, absolutas acumuladas y relativas acumuladas.

Ejercicios-4

6. Preparar una rutina que dadas las marcas de clase x_i y las frecuencias absolutas n_i, calcule los momentos ordinarios y centrales hasta el cuarto orden, la media, varianza, sesgo y curtosis.
7. Preparar una rutina, tal que, dados los extremos de clase L_i (primera clase $(-\infty, L_1]$ y última (L_n, ∞)) calcule la moda, mediana y cuartiles.
8. Preparar una rutina, tal que, dados los extremos de clase L_i (primera clase $(-\infty, L_1]$ y última (L_n, ∞)) calcule la media, varianza, media armónica y cuadrática.