Tema 1
Introducción a Matlab

Dra. Paula Triguero Navarro

Máster en Ingeniería Matemática y Computación
Escuela Superior en Ingeniería y Tecnología

LA UNIVERSIDAD
EN INTERNET

Contenido

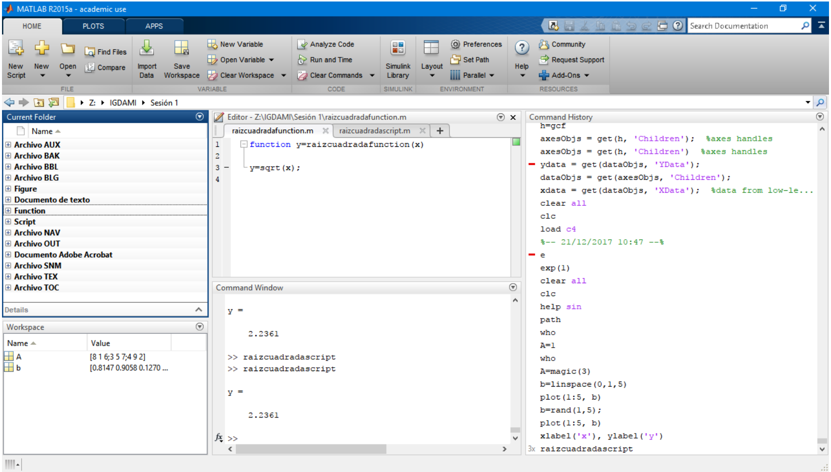
1

¿Qué es Matlab?

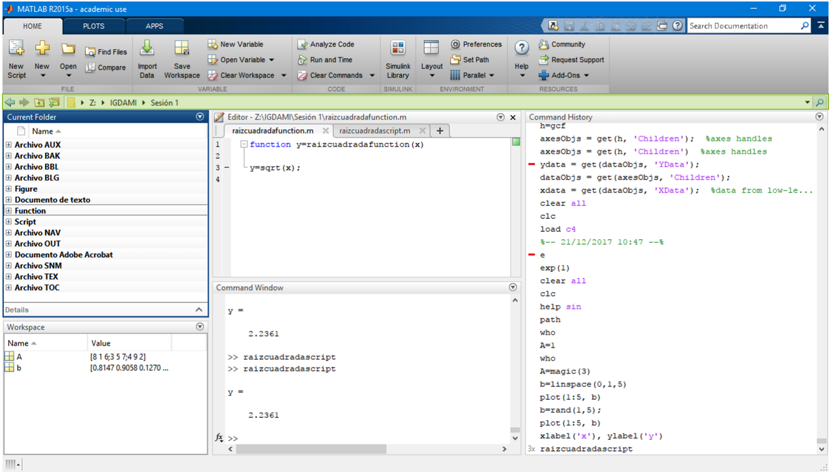
2

La interfaz gráfica

Partes de la interfaz gráfica

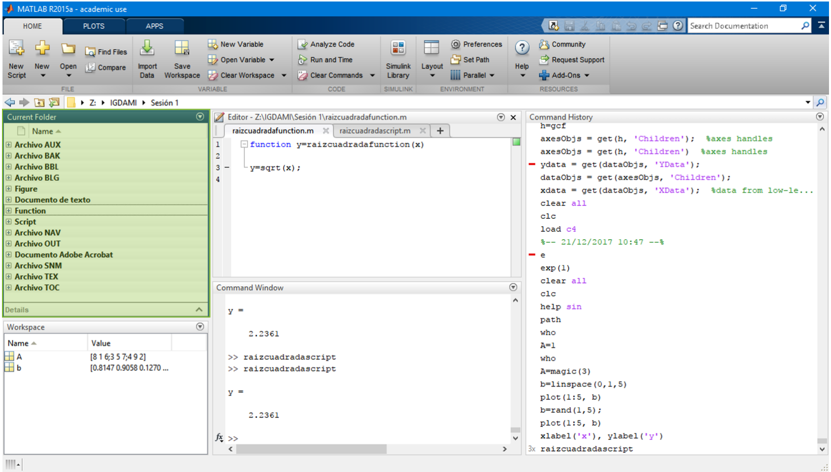


Establecer el directorio de trabajo

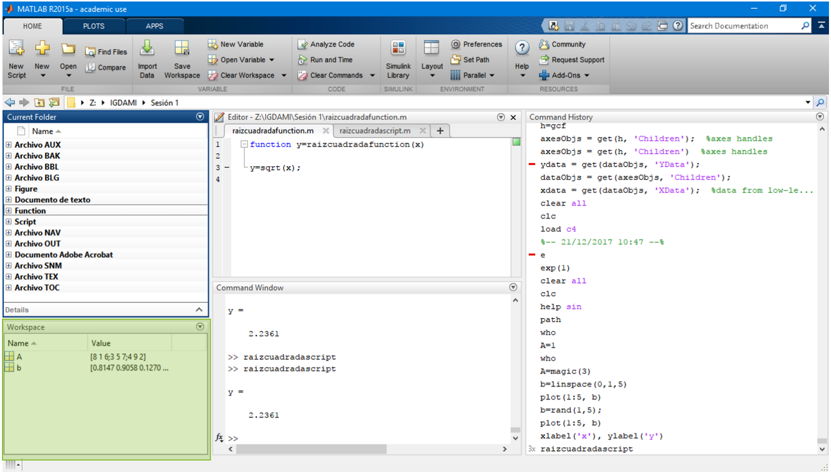


La interfaz gráfica

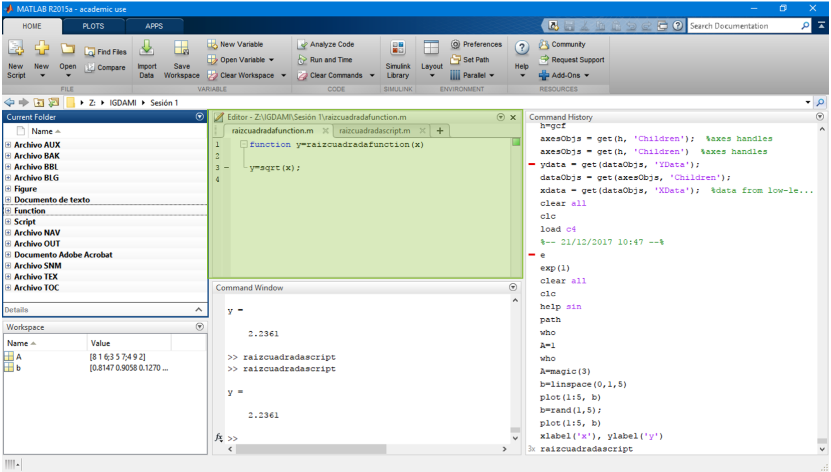
Archivos de la carpeta actual



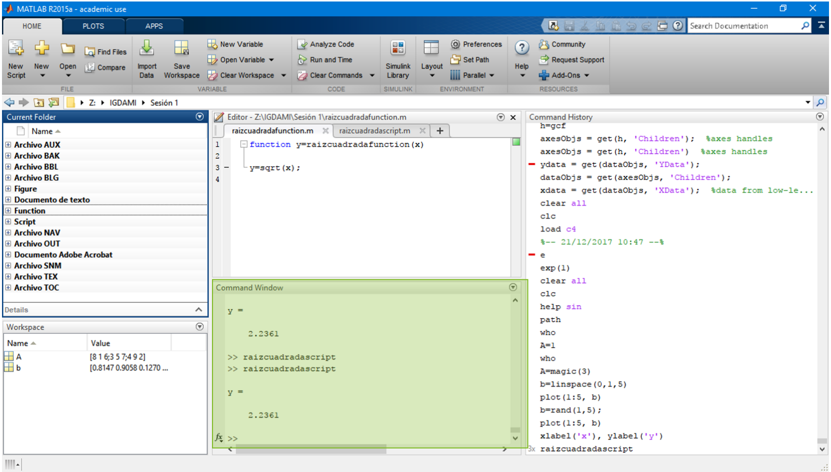
Workspace


MATLAB R2015a - academic use
HOME
PLOTS
APPS
Current Folder
Archivo AUX
Archivo BAK
Archivo BBL
Archivo BLG
Figure
Documento de texto
Function
Script
Archivo NAV
Archivo OUT
Documento Adobe Acrobat
Archivo SNM
Archivo TEX
Archivo TOC
Workspace
Name
Value
A
[8 1 6;3 5 7;4 9 2]
b
[0.8147 0.9058 0.1270 ...
Editor - Z:\IGDAMI\Sesión 1\raizcuadradafunction.m
function y=raizcuadradafunction(x)
y=sqrt(x);
Command Window
y =
2.2361
>> raizcuadradascript
>> raizcuadradascript
y =
2.2361
Command History
h=gcf
clear all
clc
load c4
%-- 21/12/2017 10:47 --%
e
exp(1)
clear all
clc
help sin
path
who
A=1
who
A=magic(3)
b=linspace(0,1,5)
plot(1:5, b)
b=rand(1,5);
plot(1:5, b)
xlabel('x'), ylabel('y')
raizcuadradascript

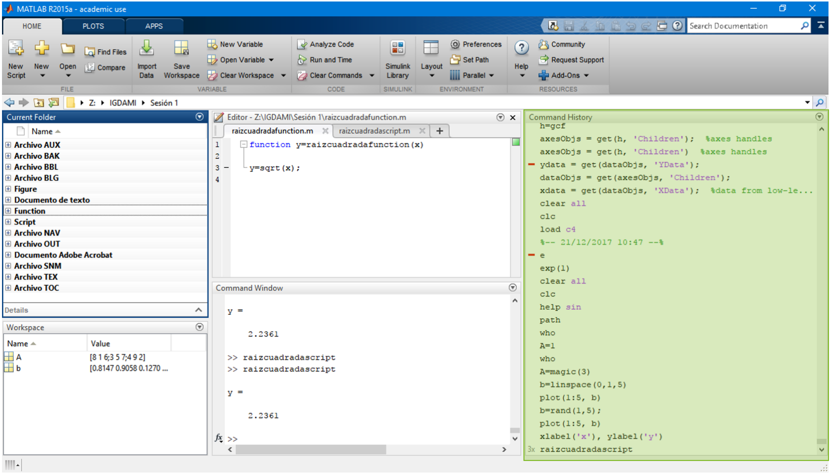
Editor de funciones, scripts, texto, ...



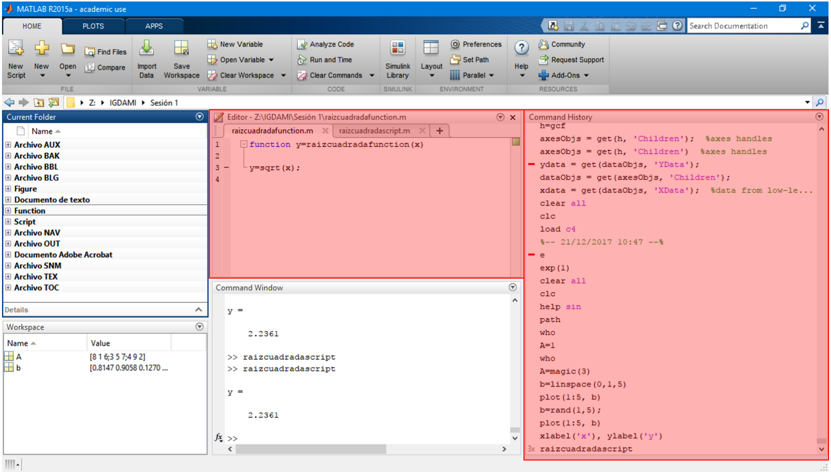
Ventana de comandos



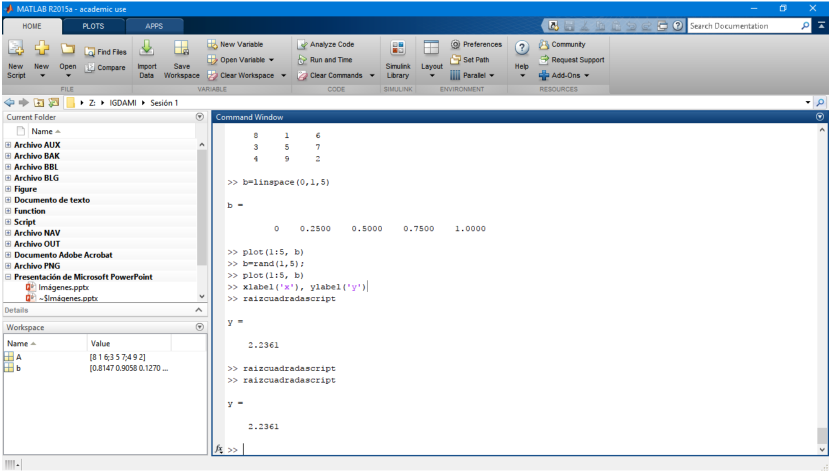
Historial de comandos



Aspecto que le vamos a dar



Aspecto que le vamos a dar



3

Instrucciones básicas

Instrucciones básicas

Comandos

- `help`
- `save, clear, clc, who, load`

Ejemplo 1. Información sobre el comando `save`

```
>> help save
```

Ejemplo 2. Información sobre la función `ode45`

```
>> help ode45
```

Instrucciones básicas

Tipos de archivos

- `.m`, `.mat`, `.fig`, `.mlx`

Ejemplo 3. Generación del archivo `holaMundo.m`

1. Abrir el editor de Matlab
2. Escribir `disp('Hola Mundo');`
3. Guardar como holaMundo.m
4. Ejecutar en la consola: >> `holaMundo`

Ejemplo 4. Generación del archivo `Notas.mat`

1. Introducir en la consola las notas de Modelado y Simulación Numérica: >> `MSN=[5 7 8 4 10]`
2. Introducir en la consola las notas de Métodos Numéricos: >> `MN=[3 7 10 9 8]`
3. Introducir en la consola las notas de Geometría Diferencial Aplicada: >> `GDA=[9 7 8 5 1]`
4. Guardar todas las variables: >> `save Notas`

Instrucciones básicas

Operadores

- `par1+par2`
- `par1-par2`
- `par1*par2`
- `par1/par2`
- `par1^par2`

Para operar con vectores y matrices elemento a elemento:

- `par1.*par2`
- `par1./par2`
- `par1.^par2`

Ejemplo 5. Producto de $u = [1, 2, 3]$ y $v = [4, 5, 6]$

```
>> u*v'
```

$$u \cdot v^t = [1, 2, 3] \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = 32$$

```
>> u.*v
```

$$\begin{bmatrix} 1 \cdot 4 \\ 2 \cdot 5 \\ 3 \cdot 6 \end{bmatrix} = \begin{bmatrix} 4 \\ 10 \\ 18 \end{bmatrix}$$

Funciones

- `sin, cos, log, log10, log2, atan, atan2`

Ejemplo 6. Cálculo del $\cos(\pi/4)$

```
>> cos(pi/4)
```

Ejemplo 7. Cálculo del $\log(0)$

```
>> log10(0)
```

Otros

- `ans`
- $\frac{0}{0}$: `NaN`
- π: `pi`
- e: `exp(1)`
- i: `1i`
- $\cdots$

Ejemplo 8. Cálculo del último elemento elevado al cuadrado

```
>> ans^2
```

Ejemplo 9. Cálculo de $e^{-i\frac{\pi}{2}}$

```
>> exp(-1i*pi/2)
```

Instrucciones básicas

Comando	Resultado
`format short`	4 dígitos tras el punto decimal
`format long`	15 dígitos tras el punto decimal
`format shortE`	4 dígitos tras el punto decimal con notación científica
`format longE`	15 dígitos tras el punto decimal con notación científica
`format shortG`	Ajusta automáticamente la presentación
`format bank`	2 dígitos tras el punto decimal
`format rat`	Fracción entre dos enteros

Tabla: Algunos formatos numéricos

Tipos numéricos

- Booleanos: `logical`
- Reales: `double`

Ejemplo 10. Formatos numéricos de salida de π

- `format short`: `pi=3.1416`
- `format long`: `pi=3.141592653589793`
- `format shortE`: `pi = 3.1416e+000`
- `format longE`: `pi = 3.141592653589793e+000`

4

Operaciones con vectores y matrices

Contenidos

Vectores

¿Cómo se introducen los vectores?

- Vector fila: entre corchetes, con comas o espacios entre los elementos
- Vector columna: entre corchetes, con punto y coma entre los elementos
- Vector traspuesto: ’

Vectores

¿Cómo se introducen los vectores?

- Vector fila: entre corchetes, con comas o espacios entre los elementos
- Vector columna: entre corchetes, con punto y coma entre los elementos
- Vector traspuesto: '

Ejemplo 11. Introduce el vector $w = \begin{bmatrix} 4 & \pi & -1 & 0 \end{bmatrix}$ y $v = w^t$.

```
>> w=[4 pi -1 0]
>> v=[4; pi; -1; 0]
```

Vectores

¿Cómo se introducen los vectores?

- Vector fila: entre corchetes, con comas o espacios entre los elementos
- Vector columna: entre corchetes, con punto y coma entre los elementos
- Vector traspuesto: '

Ejemplo 11. Introduce el vector $w = \left[\begin{array}{cccc} 4 & \pi & -1 & 0 \end{array}\right]$ y $v = w^t$.

```
>> w=[4 pi -1 0]
>> v=[4; pi; -1; 0]
```

Extracción de una componente

- Entre paréntesis la componente del elemento que se quiere extraer

Vectores

¿Cómo se introducen los vectores?

- Vector fila: entre corchetes, con comas o espacios entre los elementos
- Vector columna: entre corchetes, con punto y coma entre los elementos
- Vector traspuesto: '

Ejemplo 11. Introduce el vector $w = \left[\begin{array}{cccc} 4 & \pi & -1 & 0 \end{array}\right]$ y $v = w^t$.

```
>> w=[4 pi -1 0]
>> v=[4; pi; -1; 0]
```

Extracción de una componente

- Entre paréntesis la componente del elemento que se quiere extraer

Ejemplo 12. Obtén la última componente del vector w

```
>> w(end)
```

Vectores

¿Cómo se introducen los vectores?

- Vector fila: entre corchetes, con comas o espacios entre los elementos
- Vector columna: entre corchetes, con punto y coma entre los elementos
- Vector traspuesto: '

Ejemplo 11. Introduce el vector $w = \begin{bmatrix} 4 & \pi & -1 & 0 \end{bmatrix}$ y $v = w^t$.

```
>> w=[4 pi -1 0]
>> v=[4; pi; -1; 0]
```

Extracción de una componente

- Entre paréntesis la componente del elemento que se quiere extraer

Ejemplo 12. Obtén la última componente del vector w

```
>> w(end)
```

Ejemplo 13. Obtén las componentes primera y tercera del vector w

```
>> w([1 3])
```

Vectores

Generación de vectores con componentes equiespaciadas

- `a:b` genera un vector de componentes espaciadas una unidad desde `a` hasta `b`.
- `a:h:b` genera un vector de componente espaciadas `h` desde `a` hasta `b`.
- `linspace(a,b)` genera un vector de 100 componentes equiespaciadas entre `a` y `b`.
- `linspace(a,b,n)` genera un vector de `n` componentes equiespaciadas entre `a` y `b`.

Generación de vectores con componentes equiespaciadas

- `a:b` genera un vector de componentes espaciadas una unidad desde `a` hasta `b`.
- `a:h:b` genera un vector de componente espaciadas `h` desde `a` hasta `b`.
- `linspace(a,b)` genera un vector de 100 componentes equiespaciadas entre `a` y `b`.
- `linspace(a,b,n)` genera un vector de `n` componentes equiespaciadas entre `a` y `b`.

Ejemplo 14. Vectores con nodos equiespaciados

- $v = \left[\begin{array}{ccccc} 2 & 4 & 6 & 8 & 10 \end{array}\right]$
```
>> v=2:2:10;
```
- $v = \left[\begin{array}{cccccc} 0 & 0.01 & 0.02 & \cdots & 0.98 & 0.99 \end{array}\right]$
```
>> v=linspace(0,0.99);
```
- $v = \left[\begin{array}{cccccc} 0 & 0.01 & 0.02 & \cdots & 0.99 & 1 \end{array}\right]$
```
>> v=linspace(0,1,101);
```

Vectores

Generación de vectores con componentes equiespaciadas

- `a:b` genera un vector de componentes espaciadas una unidad desde `a` hasta `b`.
- `a:h:b` genera un vector de componente espaciadas `h` desde `a` hasta `b`.
- `linspace(a,b)` genera un vector de 100 componentes equiespaciadas entre `a` y `b`.
- `linspace(a,b,n)` genera un vector de `n` componentes equiespaciadas entre `a` y `b`.

Ejemplo 14. Vectores con nodos equiespaciados

- $v = \begin{bmatrix} 2 & 4 & 6 & 8 & 10 \end{bmatrix}$
  ```
  >> v=2:2:10;
  ```
- $v = \begin{bmatrix} 0 & 0.01 & 0.02 & \cdots & 0.98 & 0.99 \end{bmatrix}$
  ```
  >> v=linspace(0,0.99);
  ```
- $v = \begin{bmatrix} 0 & 0.01 & 0.02 & \cdots & 0.99 & 1 \end{bmatrix}$
  ```
  >> v=linspace(0,1,101);
  ```

Ejemplo 15. Calcula $3^4 + 4^4 + 5^4 + 6^4 + 7^4$.

```
>> x=3:7;
>> y=x.^4;
>> z=sum(y);
```

Vectores

Operaciones con vectores

- Mismo número de componentes
- `u+v, u-v, u*v, u.*v, u^2, u.^2, u.\v`
- `cross(u,v), dot(u,v), norm(u), length(u)`

Ejemplo 16. Sean $w = \begin{bmatrix} 4 & \pi & -1 & 0 \end{bmatrix}$ y $t = \begin{bmatrix} i & 0 & 2 & 1-i \end{bmatrix}^T$, calcula

- $w + t^T$: `w+t.'`
- $w \cdot t$: `w*t`
- $z = \begin{bmatrix} w_1t_1 & w_2t_2 & w_3t_3 & w_4t_4 \end{bmatrix}$: `w.*t.'`

Contenidos

¿Cómo se introducen las matrices?

- Entre corchetes
- Las filas se introducen como vectores, es decir, con comas o espacios entre los elementos de una fila
- Para separar cada una de las filas se introduce un punto y coma

¿Cómo se introducen las matrices?

- Entre corchetes
- Las filas se introducen como vectores, es decir, con comas o espacios entre los elementos de una fila
- Para separar cada una de las filas se introduce un punto y coma

Ejemplo 17. Introduce la matriz $B = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & -1 \\ 1 & 5 & -1 \end{bmatrix}$

```
>> B=[1 0 0;1 2 -1;1 5 1]
```

Matrices

Extracción de elementos de una matriz

`>> A=[1,-1,3; 0,2,-1; 1,5,3];`

- Una componente: `A(2,3)=-1`
- Varias componentes: `A([2,3],2)=2 5`
- Una fila: `A(3,:)=1 5 3`
- Una columna: `A(:,2)=-1 2 5`

Matrices

Extracción de elementos de una matriz

```
>> A=[1,-1,3; 0,2,-1; 1,5,3];
```

- Una componente: `A(2,3)=-1`
- Varias componentes: `A([2,3],2)=2 5`
- Una fila: `A(3,:)=1 5 3`
- Una columna: `A(:,2)=-1 2 5`

Ejemplo 18. Obtén la primera fila de la matriz B

```
>> B(1,:)
```

Extracción de elementos de una matriz

```
>> A=[1,-1,3; 0,2,-1; 1,5,3];
```

- Una componente: `A(2,3)=-1`
- Varias componentes: `A([2,3],2)=2 5`
- Una fila: `A(3,:)=1 5 3`
- Una columna: `A(:,2)=-1 2 5`

Ejemplo 18. Obtén la primera fila de la matriz B

```
>> B(1,:)
```

Ejemplo 19. Obtén un vector b compuesto por los elementos de la diagonal principal de B

```
>> b=[B(1,1) B(2,2) B(3,3)]
```

Operaciones con matrices

- Respetando las dimensiones de las operaciones
- `A+B`, `A*B`, `A.*B`, `A\B`
- `det`, `inv`, `size`, `rank`, `rref`

Matrices

Operaciones con matrices

- Respetando las dimensiones de las operaciones
- `A+B`, `A*B`, `A.*B`, `A\B`
- `det`, `inv`, `size`, `rank`, `rref`

Matrices especiales

- Matriz de ceros: `zeros(n)`, `zeros(m,n)`
- Matriz de unos: `ones(n)`, `ones(m,n)`
- Matriz identidad de tamaño n: `eye(n)`
- Matriz aleatoria en $(0, 1)$: `rand(n)`, `rand(m,n)`

Matrices

Operaciones con matrices

- Respetando las dimensiones de las operaciones
- `A+B`, `A*B`, `A.*B`, `A\B`
- `det`, `inv`, `size`, `rank`, `rref`

Matrices especiales

- Matriz de ceros: `zeros(n)`, `zeros(m,n)`
- Matriz de unos: `ones(n)`, `ones(m,n)`
- Matriz identidad de tamaño n: `eye(n)`
- Matriz aleatoria en $(0,1)$: `rand(n)`, `rand(m,n)`

Ejemplo 20. Sean las matrices $C = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & -1 \\ 1 & 5 & -1 \end{bmatrix}$ y $D = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$, obtén:

- $C + D$: `C+D`
- F tal que $f_{ij} = c_{ij} \cdot d_{ij}$: `F=C.*D`
- $|D|$: `det(D)`

Ejemplo 21. Resolución de sistemas de ecuaciones lineales

Vamos a resolver el sistema

$$\left.\begin{array}{rcc} x_1 + x_2 - 2x_3 & = & 4 \\ x_1 - x_2 & = & 1 \\ x_1 + 3x_3 & = & 0 \end{array}\right\},$$

por lo que tenemos un sistema $Ax = b$, donde

$$A = \begin{bmatrix} 1 & 1 & -2 \\ 1 & -1 & 0 \\ 1 & 0 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad b = \begin{bmatrix} 4 \\ 1 \\ 0 \end{bmatrix}.$$

Entonces:

$$Ax = b \leftrightarrow A^{-1}Ax = A^{-1}b \leftrightarrow x = A^{-1}b.$$

- Utilizando matrices inversas (no recomendado):
  ```
  >> x=inv(A)*b;
  ```
- Sin utilizar ninguna inversa (menor coste computacional):
  ```
  >> x=A\b;
  ```

5

Funciones anónimas y cálculo simbólico

Funciones anónimas

- Se almacenan en el `Workspace`
- Su definición comienza con una arroba `@` seguida de la variable entre paréntesis, poniendo fuera del paréntesis la expresión de $f(x)$.
- Para ejecutar la función $f(x)$, se escribe el nombre de la función introduciendo entre paréntesis el escalar o el vectorial sobre el que se quiere evaluar.

Funciones anónimas

- Se almacenan en el `Workspace`
- Su definición comienza con una arroba `@` seguida de la variable entre paréntesis, poniendo fuera del paréntesis la expresión de $f(x)$.
- Para ejecutar la función $f(x)$, se escribe el nombre de la función introduciendo entre paréntesis el escalar o el vectorial sobre el que se quiere evaluar.

Ejemplo 22. $f(t) = \sqrt{t} - \sin(t^2)$

- Introducimos la función:

```
>> f= @(t) sqrt(t)-sin(t.^2);
```

- Evaluamos la función sobre los valores de la matriz

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 5 \\ 1 & 4 & 7 \end{bmatrix}$$

```
>> A=[1 2 3; 1 3 5; 1 4 7];
>> f(A)
```

Funciones anónimas

- Se almacenan en el `Workspace`
- Su definición comienza con una arroba `@` seguida de la variable entre paréntesis, poniendo fuera del paréntesis la expresión de $f(x)$.
- Para ejecutar la función $f(x)$, se escribe el nombre de la función introduciendo entre paréntesis el escalar o el vectorial sobre el que se quiere evaluar.

Ejemplo 22. $f(t) = \sqrt{t} - \sin(t^2)$

- Introducimos la función:

```
>> f= @(t) sqrt(t)-sin(t.^2);
```

- Evaluamos la función sobre los valores de la matriz

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 5 \\ 1 & 4 & 7 \end{bmatrix}$$

```
>> A=[1 2 3; 1 3 5; 1 4 7];
>> f(A)
```

Ejemplo 23. Definimos $f(x,y) = e^{xy}\sin(x) - x^3y + \cos(y)$

```
>> f=@(x,y) exp(x*y).*sin(x)-x.^3.*y+cos(y);
```

Cálculo simbólico

Matlab también puede trabajar con cálculo simbólico, aunque no es tan potente como Mathematica

Cálculo simbólico

Matlab también puede trabajar con cálculo simbólico, aunque no es tan potente como Mathematica

- Es necesario definir qué variables serán simbólicas:

```
>> syms x
```

Cálculo simbólico

Matlab también puede trabajar con cálculo simbólico, aunque no es tan potente como Mathematica

- Es necesario definir qué variables serán simbólicas:
  ```
  >> syms x
  ```
- Una vez obtengamos el resultado, este será simbólico. Para poder recuperar el valor numérico, utilizaremos el comando `double`.

Cálculo simbólico

Matlab también puede trabajar con cálculo simbólico, aunque no es tan potente como Mathematica

- Es necesario definir qué variables serán simbólicas:
  `>> syms x`
- Una vez obtengamos el resultado, este será simbólico. Para poder recuperar el valor numérico, utilizaremos el comando `double`.

Matlab

```
>> syms x c
>> fx=x^2+c
fx =
x^2 + c
>> v=x+fx
v =
x^2 + x + c
>> fv=v^2+c
fv =
c + (x^2 + x + c)^2
>> M=simplify(x-fx^2/(fv-fx))
M =
(x^3 + x^2 + c*x - c)/(x^2 + 2*x + c)
>> solve(M==x,x)
ans =
  (-c)^(1/2)
 -(-c)^(1/2)
```

Mathematica

```
f[x_, c_] = x^2 + c
```

$c + x^2$

```
v[x_, c_] = x + f[x, c]
```

$c + x + x^2$

```
M[x_, c_] = x - f[x, c]^2 / (f[v, c] - f[x, c]) // Simplify
```

$$x - \frac{\left(c + x^2\right)^2}{v^2 - x^2}$$

```
Solve[M[x, c] == x, x]
```

$$\left\{\left\{x \to -i\sqrt{c}\right\}, \left\{x \to -i\sqrt{c}\right\}, \left\{x \to i\sqrt{c}\right\}, \left\{x \to i\sqrt{c}\right\}\right\}$$

Cálculo simbólico

Ejemplo 24. Operaciones utilizando variables simbólicas

Definimos la función $f(x) = x^2 - 1$:

```
>> syms x;
>> f=x^2-1;
```

- Raíces:
```
>> sol=solve(f==0);
```
- Derivada:
```
>> df=diff(f,x);
```
- Integral:
```
>> F=int(f,x);
```

Ejemplo 25. Operaciones con varias variables

Definimos la función $g(x, y) = x^2 y - 2x$:

```
>> syms x y;
>> g=x^2*y-2*x;
```

- Solución:
```
>> solx=solve(g==0,x);
>> soly=solve(g==0,y);
```
- Derivada parciales:

$$\frac{\partial g}{\partial x}$$ `>> dfx=diff(f,x);` $$\frac{\partial g}{\partial y}$$ `>> dfy=diff(f,y);`

- Integrales:

$$\int g(x,y)dx$$ `>> Fx=int(f,x);` $$\int g(x,y)dy$$ `>> Fy=int(f,y);`

6

Funciones y scripts

Funciones y scripts

Ambos ejecutan automática y secuencialmente una serie de líneas de comandos

Funciones

- Tiene parámetros de entrada
- Trabaja con un `Workspace` local
- Útil con pocos y muchos parámetros

Scripts

- No tiene parámetros de entrada
- Actúa con las variables del `Workspace` y las modifica si necesita acceder a ellas
- Útil con pocos parámetros

Funciones y scripts

Ambos ejecutan automática y secuencialmente una serie de líneas de comandos

Funciones

- Tiene parámetros de entrada
- Trabaja con un `Workspace` local
- Útil con pocos y muchos parámetros

raizcuadradafunction.m

```
function y=raizcuadradafunction(x)
  y=sqrt(x);
end
```

Scripts

- No tiene parámetros de entrada
- Actúa con las variables del `Workspace` y las modifica si necesita acceder a ellas
- Útil con pocos parámetros

raizcuadradascript.m

```
x=5;
y=sqrt(x);
```

Ejemplo 26. Genera el archivo `e2g.m` que resuelve polinomios de segundo grado del tipo $ax^2 + bx + c = 0$. Como parámetros de entrada tienes que introducir los coeficientes a, b y c, y como parámetros de salida las soluciones.

1. Abrir el editor de Matlab
2. Escribir la función

```
function [sol]=e2g(a,b,c)
sol1=(-b+sqrt(b^2-4*a*c))/(2*a);
sol2=(-b-sqrt(b^2-4*a*c))/(2*a);
sol=[sol1 sol2];
```

Ejemplo 26. Genera el archivo `e2g.m` que resuelve polinomios de segundo grado del tipo $ax^2 + bx + c = 0$. Como parámetros de entrada tienes que introducir los coeficientes a, b y c, y como parámetros de salida las soluciones.

1. Abrir el editor de Matlab
2. Escribir la función

```
function [sol]=e2g(a,b,c)
sol1=(-b+sqrt(b^2-4*a*c))/(2*a);
sol2=(-b-sqrt(b^2-4*a*c))/(2*a);
sol=[sol1 sol2];
```

Ejemplo 27. Ejecuta el programa e2g para obtener la solución de los siguientes polinomios de segundo grado.

- $2x^2 - 5 = 0$: `>> s1=e2g(2,0,-5)`
- $x^2 + 1 = 0$: `>> s2=e2g(1,0,1)`
- $x^2 - i2x + \pi = 0$: `>> s3=e2g(1,-2*1i,pi)`

7

Estructuras de control

Condiciones

Booleano

Tipo de dato que ocupa 1 byte y tiene como posibles estados falso y verdadero.
En Matlab:

- `logical`
- Posibles estados: Verdadero (1) y falso (0)

Condiciones

Booleano

Tipo de dato que ocupa 1 byte y tiene como posibles estados falso y verdadero.
En Matlab:

- `logical`
- Posibles estados: Verdadero (1) y falso (0)

Condición

- Operación cuya salida es un booleano
- Requieren del uso de operadores relacionales

Operador	Significado
==	Igual
>=	Mayor o igual
>	Estrictamente mayor
<=	Menor o igual
<	Estrictamente menor
~=	Diferente

Condiciones

- Una pareja de elementos booleanos se pueden relacionar a través de las operaciones lógicas. En Matlab:

A	B	and(A,B)	or(A,B)	xor(A,B)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

- Una pareja de elementos booleanos se pueden relacionar a través de las operaciones lógicas. En Matlab:

`A`	`B`	`and(A,B)`	`or(A,B)`	`xor(A,B)`
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Ejemplo 28. `x`: números positivos menores que 20

```
>> and(x>=0,x<20)
```

Ejemplo 29. `x`: números pares o números negativos

```
>> or(rem(x,2)==0,x<0)
```

Contenidos

Condicionales

if

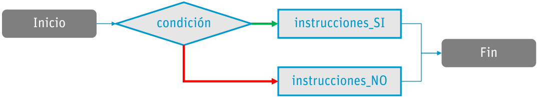
```
if condición
    instrucciones
end
```

Condicionales

if

```
if condición
    instrucciones
end
```

```
if condición
    intrucciones
else
    instrucciones
end
```



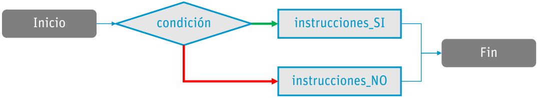
Condicionales

if

```
if condición
    instrucciones
end
```

```
if condición
    intrucciones
else
    instrucciones
end
```

```
if condición
    intrucciones
else if condición
          instrucciones
      else
          instrucciones
      end
end
```



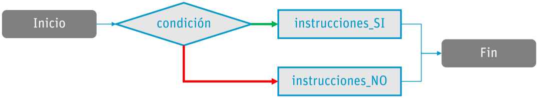
Condicionales

if

```
if condición
    instrucciones
end
```

```
if condición
    intrucciones
else
    instrucciones
end
```

```
if condición
    intrucciones
else if condición
          instrucciones
      else
          instrucciones
      end
end
```



Ejemplo 30. if

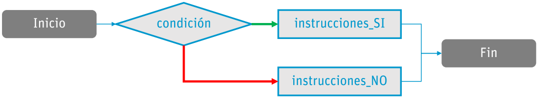
```
if nota<5
 disp('Suspenso')
else if nota<7
          disp('Aprobado')
      else if nota<9
               disp('Notable')
           else
               disp('Sobresaliente')
           end
      end
end
```

if

```
if condición
    instrucciones
end
```

```
if condición
    intrucciones
else
    instrucciones
end
```

```
if condición
    intrucciones
else if condición
          instrucciones
      else
          instrucciones
      end
end
```



Ejemplo 30. if

```
if nota<5
 disp('Suspenso')
else if nota<7
        disp('Aprobado')
     else if nota<9
            disp('Notable')
         else
            disp('Sobresaliente')
         end
     end
end
```

<u>Alineación en Matlab</u>

$\boxed{\text{Ctrl+A}}+\boxed{\text{Ctrl+I}}$

Condicionales

switch

```
switch expresión
  case valor1
      instrucciones
  case valor2
      instrucciones
end
```

Condicionales

switch

```
switch expresión
  case valor1
      instrucciones
  case valor2
      instrucciones
end
```

```
switch expresión
  case valor1
      instrucciones
  case valor2
      instrucciones
  otherwise
      instrucciones
end
```

Condicionales

switch

```
switch expresión
  case valor1
      instrucciones
  case valor2
      instrucciones
end
```

```
switch expresión
  case valor1
      instrucciones
  case valor2
      instrucciones
  otherwise
      instrucciones
end
```

Ejemplo 31. switch

Raíces de $f(x) = x^2 + c$ para $c \in \{-1, 0, 1\}$

```
switch c
  case -1
      disp('Raíces reales')
  case 0
      disp('Raíz real')
  otherwise
      disp('Raíces complejas')
end
```

Contenidos

Bucles

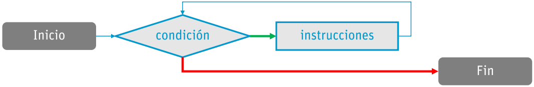
while

```
while condición
    instrucciones
end
```

Bucles

while

```
while condición
    instrucciones
end
```



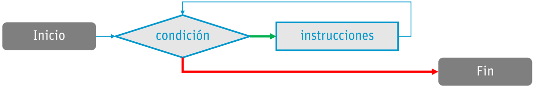
Uso de condiciones

Incrementales

```
n=0;
while n<10
    instrucciones
    n=n+1;
end
```

while

```
while condición
    instrucciones
end
```



Uso de condiciones

Incrementales

```
n=0;
while n<10
    instrucciones
    n=n+1;
end
```

Ejemplo 32. while

```
iter=0;
while iter<=20&&fx>1e-6
    fx=f(x); dfx=df(x);
    x=x-fx/dfx;
    iter=iter+1;
end
```

Bucles

for simple

```
for índices=vector
    instrucciones
end
```

for simple

```
for índices=vector
    instrucciones
end
```

Ejemplo 33. for simple

```
v=[4 7 1 9]; v2=[];
for k=1:length(v)
   v2(k)=v(k)*2;
end
```

Bucles

for simple

```
for índices=vector
    instrucciones
end
```

Ejemplo 33. for simple

```
v=[4 7 1 9]; v2=[];
for k=1:length(v)
    v2(k)=v(k)*2;
end
```

for doble

```
for índices1=vector1
    for índices2=vector2
        instrucciones
    end
end
```

Bucles

for simple

```
for índices=vector
    instrucciones
end
```

Ejemplo 33. for simple

```
v=[4 7 1 9]; v2=[];
for k=1:length(v)
   v2(k)=v(k)*2;
end
```

---

for doble

```
for índices1=vector1
    for índices2=vector2
        instrucciones
    end
end
```

Ejemplo 34. for doble

```
A=rand(3,4); A2=[];
for fil=1:size(A,1)
   for col=1:size(A,2)
     A2(fil,col)=A(fil,col)*2;
   end
end
```

8

Representaciones gráficas

Contenidos

`plot(x,y)`

Representa en abscisas los valores de `x` y en ordenadas los valores de `y`, unidos por líneas. Los vectores tienen que tener las mismas dimensiones.

`plot(x,y)`

Representa en abscisas los valores de `x` y en ordenadas los valores de `y`, unidos por líneas. Los vectores tienen que tener las mismas dimensiones.

Opciones:

- Añadir título a los ejes: `xlabel(texto)`, `ylabel(texto)`
- Añadir título a la figura: `title(texto)`
- Los títulos permiten texto en LaTeX
- Guardar figura como `*.fig`
- Añadir mallado a los ejes: `grid`
- Incluir leyenda: `legend(texto1,texto2)`

`plot(x,y)`

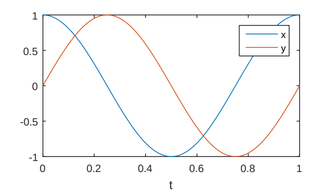
Representa en abscisas los valores de `x` y en ordenadas los valores de `y`, unidos por líneas. Los vectores tienen que tener las mismas dimensiones.

Opciones:

- Añadir título a los ejes: `xlabel(texto)`, `ylabel(texto)`
- Añadir título a la figura: `title(texto)`
- Los títulos permiten texto en LaTeX
- Guardar figura como `*.fig`
- Añadir mallado a los ejes: `grid`
- Incluir leyenda: `legend(texto1,texto2)`

Ejemplo 35. `plot`

```
t=linspace(0,2*pi);
x=cos(t);
y=sin(t);
plot(t/2/pi,[x;y]);
xlabel('t'), legend('x','y')
```



Contenidos

En una función de dos variables $f(x, y)$, se genera un mallado a partir de los vectores `x` e `y` para obtener las matrices `X` e `Y`:

```
>> [X,Y]=meshgrid(x,y);
```

En una función de dos variables $f(x, y)$, se genera un mallado a partir de los vectores `x` e `y` para obtener las matrices `X` e `Y`:

```
>> [X,Y]=meshgrid(x,y);
```

- `plot3(A,B,C)` hace la representación con líneas para cada valor de la variable `A`.
- `mesh(A,B,C)` hace la representación con un mallado de las variables `A` y `B`.
- `surf(A,B,C)` hace la representación con un mallado de las variables `A` y `B`, generando una superficie.

Representación de funciones de dos variables

En una función de dos variables $f(x,y)$, se genera un mallado a partir de los vectores `x` e `y` para obtener las matrices `X` e `Y`:

```
>> [X,Y]=meshgrid(x,y);
```

- `plot3(A,B,C)` hace la representación con líneas para cada valor de la variable `A`.
- `mesh(A,B,C)` hace la representación con un mallado de las variables `A` y `B`.
- `surf(A,B,C)` hace la representación con un mallado de las variables `A` y `B`, generando una superficie.

- Abscisas: vector o matriz `A`
- Ordenadas: vector o matriz `B`
- Cota: matriz `C`
- La matriz `C` contiene `length(A)` filas y `length(B)` columnas

Opciones:

- Interpolar el resultado: `shading interp`
- Visualizar la barra de colores: `colorbar`
- Cambiar la paleta de colores: `colormap` mapa_de_colores

Opciones:

- Interpolar el resultado: `shading interp`
- Visualizar la barra de colores: `colorbar`
- Cambiar la paleta de colores: `colormap` mapa_de_colores

Ejemplo 36. surf

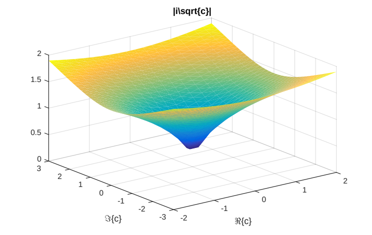
```
>> rc=linspace(-2,2,20);
ic=linspace(-3,3,40);
>>[RC,IC]=meshgrid(rc,ic);
>> C=RC+1i*IC; zC=1i*sqrt(C);
>> surf(rc,ic,abs(zC))
>> shading interp
>> xlabel('\Re\{c\}')
>> ylabel('\Im \{c \}')
```

Opciones:

- Interpolar el resultado: `shading interp`
- Visualizar la barra de colores: `colorbar`
- Cambiar la paleta de colores: `colormap` mapa_de_colores

Ejemplo 36. surf

```
>> rc=linspace(-2,2,20);
ic=linspace(-3,3,40);
>>[RC,IC]=meshgrid(rc,ic);
>> C=RC+1i*IC; zC=1i*sqrt(C);
>> surf(rc,ic,abs(zC))
>> shading interp
>> xlabel('\Re\{c\}')
>> ylabel('\Im \{c \}')
```



Ejemplo 37.

Para representar la función

$$f(x,y) = e^{-x^2-y^2}, \quad (x,y) \in [-2,2] \times [-2,2],$$

vamos a utilizar un vector de 100 puntos para las variables independientes y, a continuación, evaluamos la función sobre dichos puntos:

```
>> x=linspace(-2,2); y=linspace(-2,2);
>> [X,Y]=meshgrid (x,y);
>> f=exp(-X.^2-Y.^2);
>> figure
>> subplot(1 ,3 ,1), plot3(X,Y,f), title('plot3 ')
>> subplot(1 ,3 ,2), mesh(X,Y,f), title('mesh ')
>> subplot(1 ,3 ,3), surf(X,Y,f), title('surf ')
```

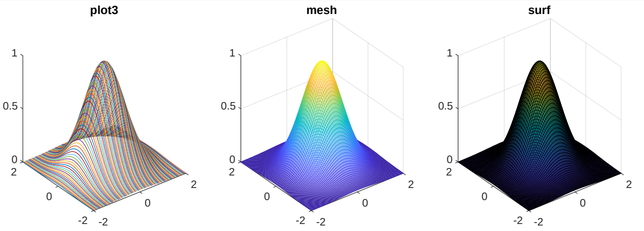
Ejemplo 37.

Para representar la función

$$f(x,y) = e^{-x^2-y^2}, \quad (x,y) \in [-2,2] \times [-2,2],$$

vamos a utilizar un vector de 100 puntos para las variables independientes y, a continuación, evaluamos la función sobre dichos puntos:

```
>> x=linspace(-2,2); y=linspace(-2,2);
>> [X,Y]=meshgrid (x,y);
>> f=exp(-X.^2-Y.^2);
>> figure
>> subplot(1 ,3 ,1), plot3(X,Y,f), title('plot3 ')
>> subplot(1 ,3 ,2), mesh(X,Y,f), title('mesh ')
>> subplot(1 ,3 ,3), surf(X,Y,f), title('surf ')
```

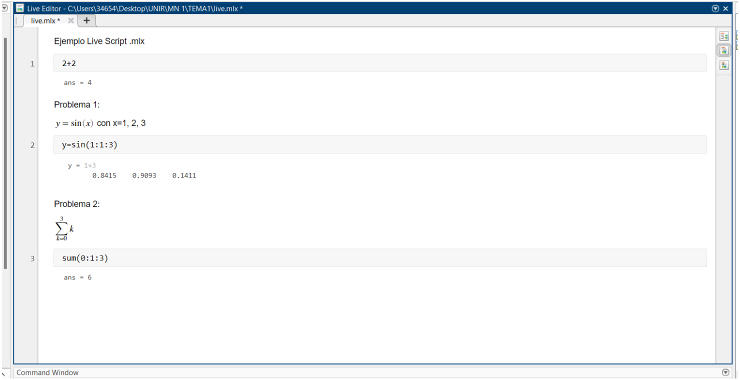


9

Archivos .mlx

Archivos .mlx

Son documentos en los que podemos escribir texto y ecuaciones, además que nos permite ver las salidas de las ejecuciones.



10

Ejercicios

Ejercicio 1.

1. Genera un script que exporte a Excel en formato tabla los resultados de las siguientes funciones en el intervalo $x \in [-5, 5]$, tomando 11 valores en cada intervalo. Muestra por pantalla los gráficos resultantes.
   - $f_1(x) = x^2 - x$
   - $f_2(x) = \cos^2(x)$
   - $f_3(x) = \exp(-x)$
   - $f_4(x) = \sin(x) + \cos(x)$
2. Genera un informe con los valores numéricos y las representaciones gráficas.

Ejercicio 2.

1. Genera un programa (una `function` que muestre por pantalla la derivada analítica y la derivada aproximada de una función. La derivada aproximada viene dada por la expresión
$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$
donde $h = x_i - x_{i-1}$. Como parámetros de entrada tendrás que dar la función y los valores de x.
2. Modifica dicho programa para que exporte los valores numéricos de la derivada analítica y la derivada aproximada para las siguientes funciones y los siguientes intervalos, tomando siempre 11 nodos:
   - $f_1(x) = x^3 - 3x + 2$, $x \in [-1, 1]$,
   - $f_2(x) = \sin^2(x) - \ln(x+5)$, $x \in [-3, 0]$,
   - $f_3(x) = \sqrt{3x} + \exp(-x/2)$, $x \in [-1, 2]$.
3. Genera un informe con los valores numéricos y las representaciones gráficas.

Referencias

- A. Cordero, J. L. Hueso, E. Martínez, J. R. Torregrosa. **Métodos Numéricos con Matlab**. *Universidad Politécnica de Valencia, 2005*. ISBN 84-9705-854-2.

unir

LA UNIVERSIDAD
EN INTERNET