

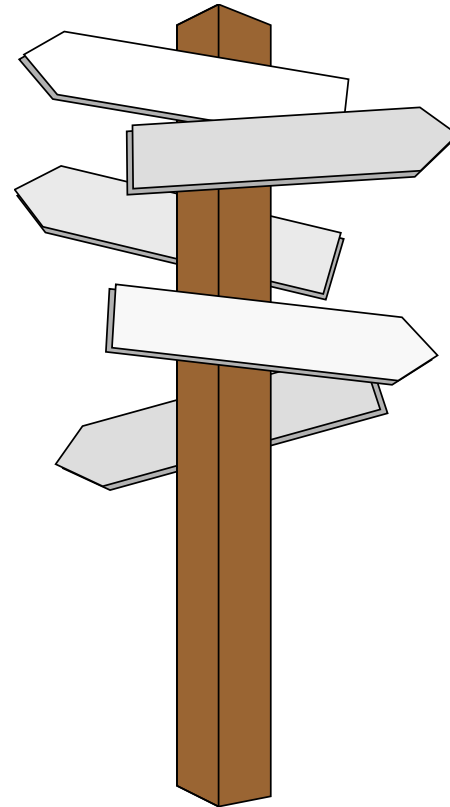
MATLAB



**... una
introducción**

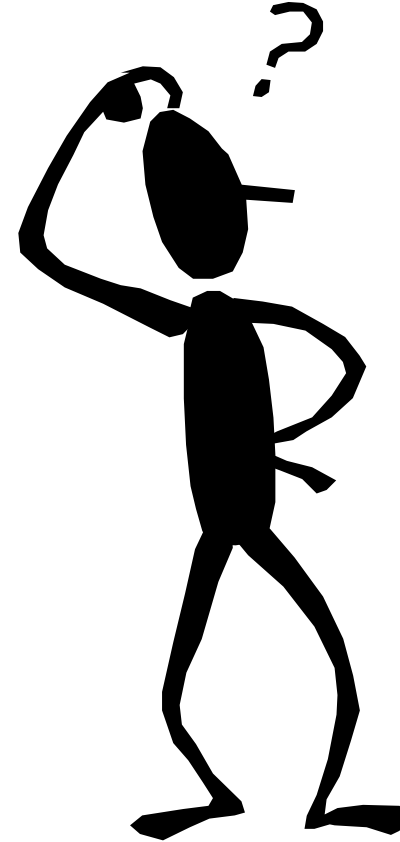
Guión

- ¿Por Qué Matlab?
- Comandos básicos
- Constantes. Operaciones.
- Variables.
 - Asignar. Eliminar.
 - Guardar. Recuperar.
- Funciones.
- Vectores. Matrices.
- Gráficas.



¿Por qué MATLAB?

- Calidad científica
- Potencia
- Flexibilidad
- Facilidad de uso
- Interactividad
- Transparencia
- Gráficos



Comandos básicos



- `help`, `help topic`
- `dir`
- `diary fichero`
- Comentarios: `%`
- Edición de líneas de comando
- Funciones de edición de Windows

Constantes

- Matrices de números complejos

- ▶ $A = [2+3i, 4; 5-i, 2i]$

- ▶ Vectores, escalares, números reales.

- Cadenas de caracteres

- ▶ `'Esto es una cadena'`

- ▶ `Esto no es una cadena`

Operaciones

- Suma y resta: $+$ $-$

- Producto, cociente y potencia: $*$ $/$ $^$

- ▶ Uso del punto: $.*$ $./$ $.^$

- Precisión aparente

- ▶ `format long`

- ▶ `format short`

$$1+1=3$$

Variables

- Asignar

- ▶ `a=3, b=4`

- Listar

- ▶ `ans`

- ▶ `who`

- ▶ `whos`

- Eliminar

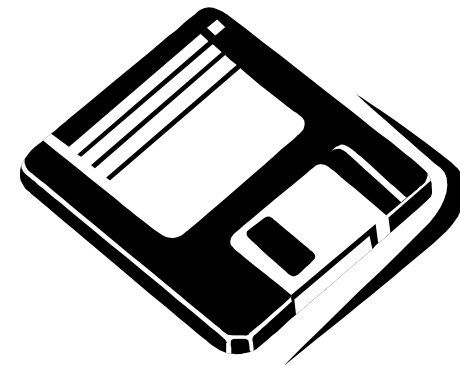
- ▶ `clear b`

- Guardar

- ▶ `save fichero`

- Recuperar

- ▶ `load fichero`



Funciones

● `help elfun`

● `sin`

● `cos`

● `tan`

● `exp`

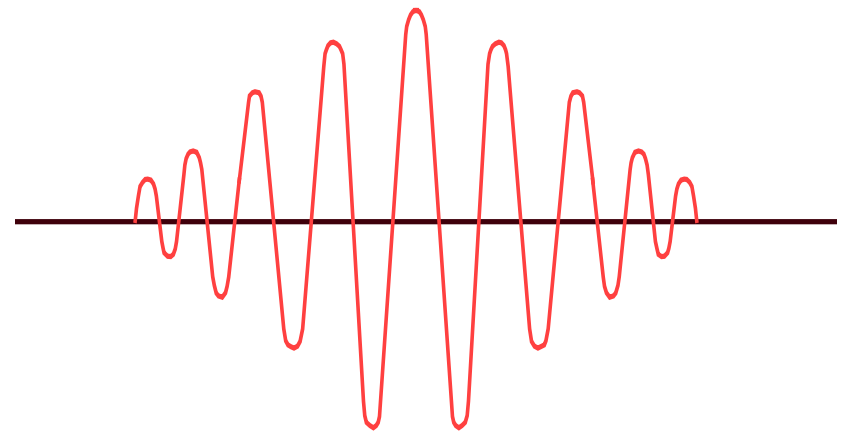
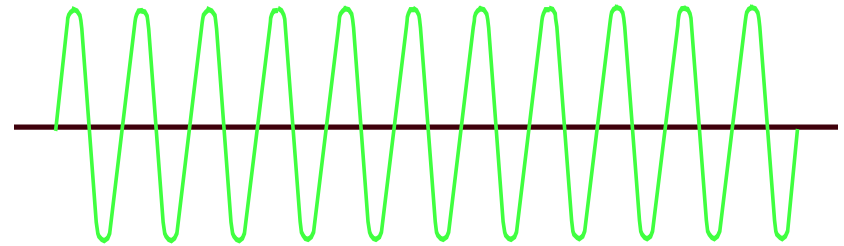
● `ezplot sin(x)`

■ `asin`

■ `acos`

■ `atan`

■ `log`



Números Complejos

Forma binómica

▶ $z = 3 + 4i$

Parte real e imaginaria

▶ $\text{real}(z) \rightarrow 3$

▶ $\text{imag}(z) \rightarrow 4$

Complejo conjugado

▶ $\text{conj}(z), z'$

Módulo y argumento

▶ $\text{abs}(z) \rightarrow 5$

▶ $\text{angle}(z)$

Representación gráfica

▶ $\text{compass}(z)$

Vectores

▶ `v = [1 9 9 8]`

Valores de funciones

```
x = 0 : 0.01 : 1
```

```
y = sin(2*pi*x)
```

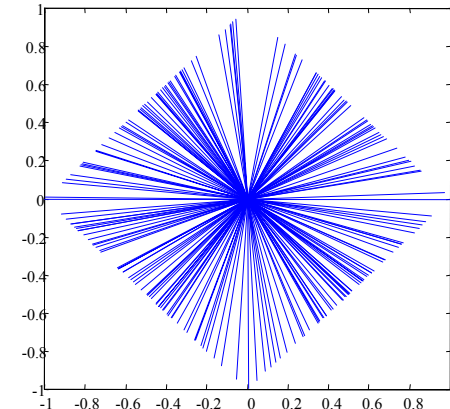
```
plot(x,y)
```

Normas

▶ `norm(v, 2)`

▶ `norm(v, 1)`

▶ `norm(v, inf)`



Operaciones

▶ `n=5;`

▶ `x=1:n`

Transpuesta: `x'`

▶ `2*x`

¡Ojo!: `x*x`

Prod. escalar: `x*x'`

Cuadrado: `x2=x.*x`

▶ `sum(x2)`

Matriz rango 1: `x'*x`

Gráficos

▶ `plot(x,x2)`

▶ `plot(x2,x)`

▶ `plot(x,x2,'*')`

Voltear

▶ `fliplr(x)`

▶ `flipud(x')`

Ejercicio: $1+2+\dots+n$

Lenguaje programación

```
n=5;  
suma=0;  
for k=1:n  
    suma=suma+k;  
end  
suma
```

MATLAB

```
n=5;  
x=1:n;  
suma=sum(x)
```

Ejercicio:

$$1^p+2^p+\dots+n^p$$

Polinomios

$$p(x) = x^4 - x^3 + 5x^2 - 1$$

- $p = [1 \ -1 \ 5 \ 0 \ -1]$ ¡De mayor a menor grado!
- Valor de p en x :
`polyval(p, x)`
- Multiplicación: `conv(p, q)`
- División con resto: `[q, r] = deconv(p, d)`

Matrices



▶ `A = [1, 9; 9, 8]`

▶ `A'`

▶ `eye(2)`

▶ `fliplr(A)`

▶ `eye(size(A))`

▶ `flipud(A)`

▶ `zeros(3,4)`

▶ `det(A)`

▶ `ones(1,10)`

▶ `inv(A)`

▶ `rand(1,10)`

▶ `rank(A)`

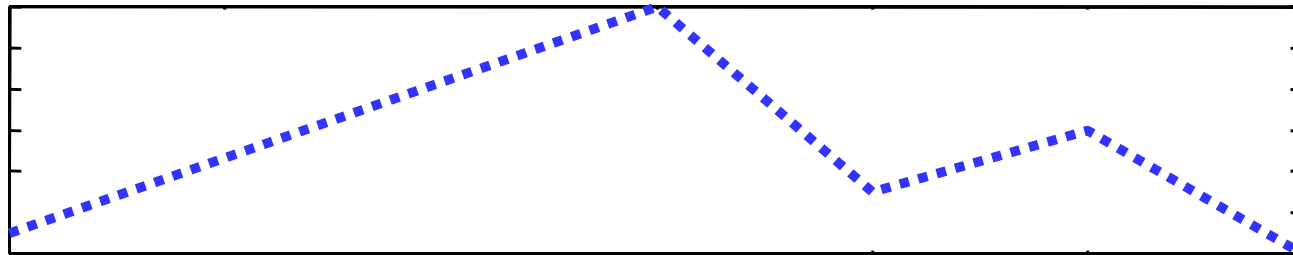
Gráficos



- Vectores
- Funciones de una variable
 - Coordenadas cartesianas
 - Coordenadas polares
 - Ecuaciones paramétricas
- Matrices
- Funciones de dos variables

Gráficos de vectores

- ▶ `x = [11 14 15 16 17];`
- ▶ `y = [695 750 705 720 690];`
- ▶ `plot(x,y,'r*:')`,
`title('Indice general de la Bolsa de Madrid')`,
`xlabel('Septiembre 1998')`



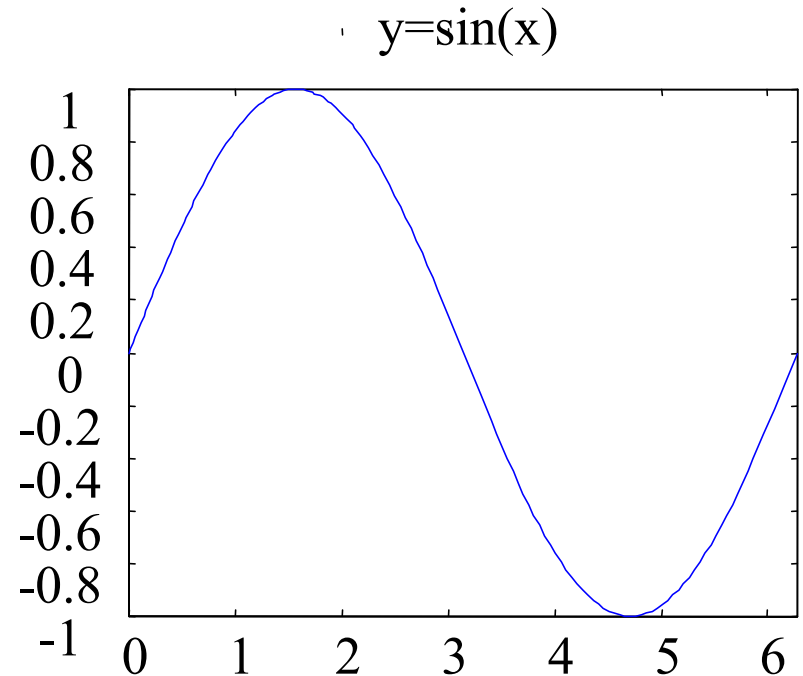
Coordenadas cartesianas

Tabla de valores

- ▶ `x = 0:0.1:2*pi;`
- ▶ `y = sin(x);`
- ▶ `plot(x,y)`
- ▶ `label('y=sin(x)')`

Orden de MATLAB

- ▶ `fplot('sin(x)', [0 2*pi])`



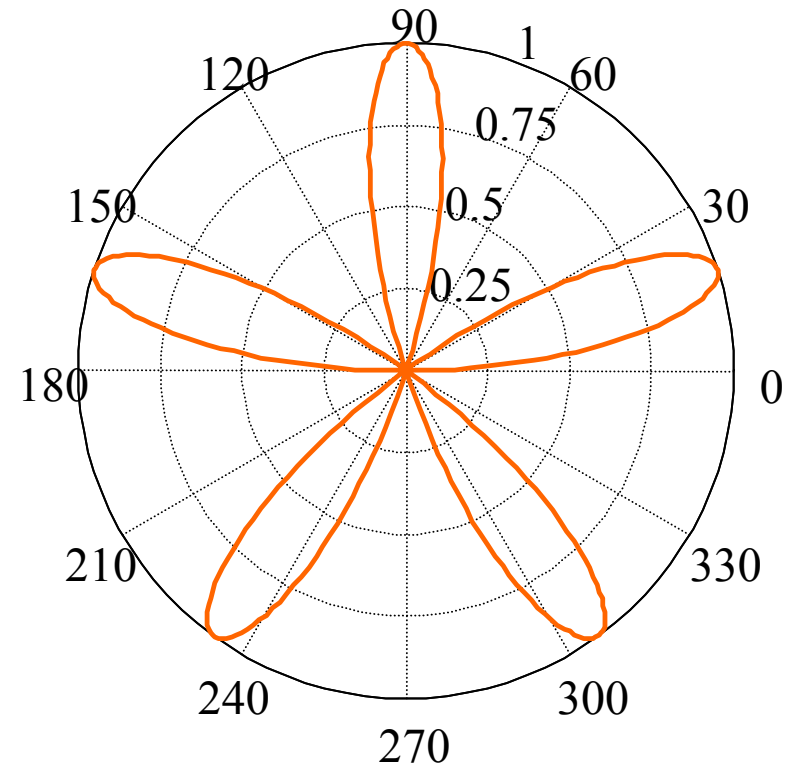
Coordenadas polares

Diagramas de radiación de antenas

▶ `z = 0:0.1:2*pi;`

▶ `r = sin(5*z);`

▶ `polar(z,r)`



Ecuaciones paramétricas

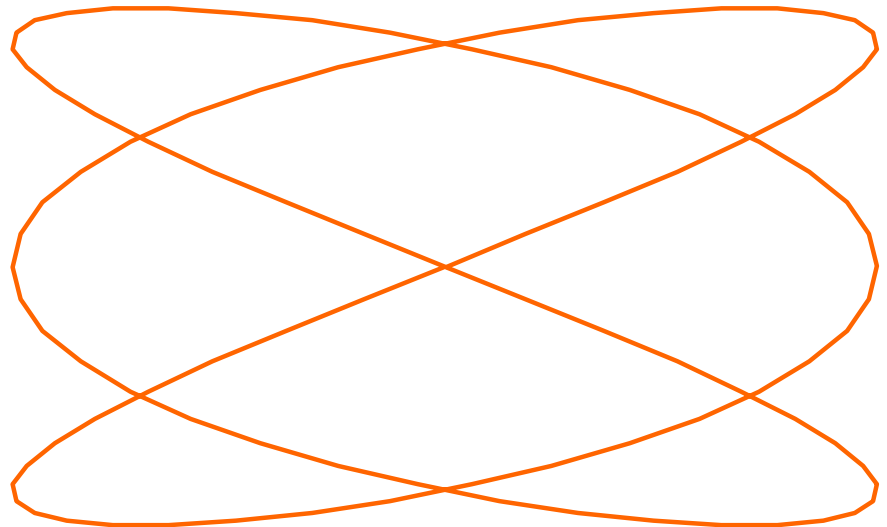
Curvas de Lisajoux

▶ `t = 0:2*pi/100:2*pi;`

▶ `x = sin(2*t);`

▶ `y = sin(3*t);`

▶ `plot(y,x)`

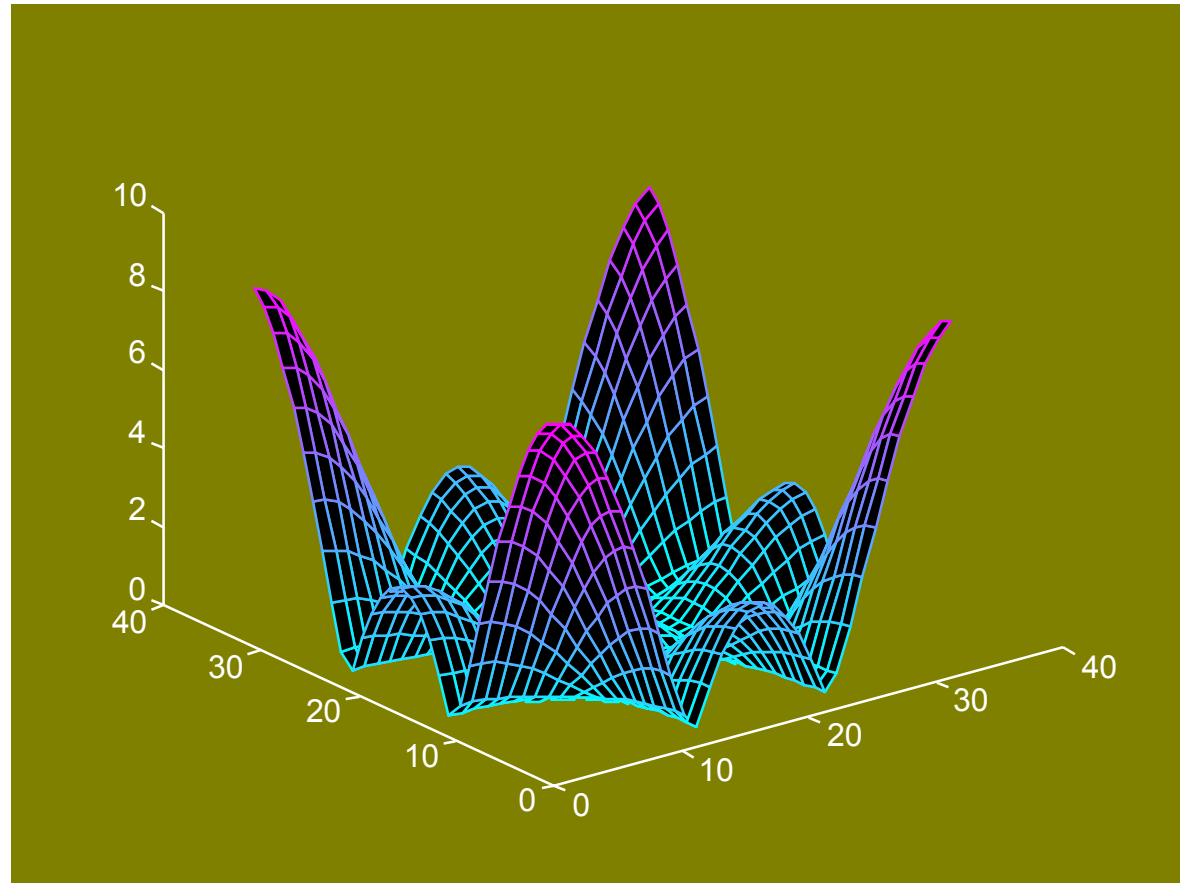


Gráficos de matrices

▶ `mesh (A)`

▶ `surf (A)`

▶ `contour (A)`



Funciones de dos variables

- ▶ `help grafxyz`
- ▶ `x = -1:0.1:1; y = x;`
- ▶ `[X,Y] = meshgrid(x,y);`
- ▶ `Z = X .* Y;`
- ▶ `surf(Z)`
- ▶ `mesh, contour, surfc`

Archivos.m



- Contienen órdenes de MATLAB
- Se invocan desde la ventana de órdenes, o desde otro archivo.m
- Se editan y graban como ficheros ASCII.

Archivos.m de Función



- Empiezan por
`function y = nomdefun(x)`
- Las variables definidas no modifican las existentes en el espacio de trabajo.
- Extienden las funciones de MATLAB.
- Permiten el paso de parámetros.

La instrucción WHILE



- Bucle controlado por una condición.

- Sintaxis:

```
while condición  
    instrucciones  
end
```

- Las instrucciones se repiten mientras la condición se verifique.

La instrucción FOR

- Bucle que se repite un número de veces.

- Sintaxis:

```
for x = array  
    instrucciones  
end
```

- Las instrucciones se ejecutan una vez para cada columna en el array.
- Podemos anidar sentencias for.

La instrucción IF



- Bifurcación condicional.

- Sintaxis:

```
if condición  
    instrucciones  
end
```

- Las instrucciones se realizan si la condición se verifica.

Operadores lógicos y relacionales

● Operadores de comparación:

- ▶ Mayor, menor <, >
- ▶ Mayor o igual >=
- ▶ Menor o igual <=
- ▶ Igual ==
- ▶ Distinto !=

● Operadores lógicos:

- ▶ Conjunción &
- ▶ Disyunción |
- ▶ O exclusiva xor
- ▶ Negación ~

Nota: ~ es [Alt] +
126

Archivo pfijo.m

```
function a = pfijo(fun,x0,tol,maxiter)
% Aproxima por el método del punto fijo una raíz de la ecuación fun(x)=x
% cercana a x0, tomando como criterio de parada abs(fun(x)-x)<tol o la cota sobre
% el numero de iteraciones dada por maxiter.
%
% Variables de entrada:
% fun(x): función a iterar, se debe introducir con notación simbólica (eg. 'g')
% x0: estimación inicial para el proceso de iteración
% tol: tolerancia en error absoluto para la raíz
% maxiter: máximo numero de iteraciones permitidas
%
% Variables de salida:
% a: valor aproximado de la raíz
fprintf(1, 'Método del punto fijo \n');
incr=1;iter=1;
while (incr>tol) & (iter<maxiter)
    a = feval(fun,x0);           % Itera la función g
    incr = abs(a-x0);           % Calcula la variación
    fprintf(1, 'iter= %i, a= %x0, incr= %e \n', iter, a, incr);
    iter = iter + 1; x0 =a;     % Cuenta los pasos y actualiza x0

end;                          % Salida
```