

Curso de Matlab. Nivel Básico

Guillem Borrell i Nogueras

9 de marzo de 2009

A set of navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

Yo.

- Guillem Borrell i Nogueras.
- Ingeniero Aeronáutico (aunque no me gustan los aviones).
- Becario del Grupo de Investigación de Mecánica de Fluidos Computacional de la Universidad Politécnica de Madrid.
- Consultor Senior de Englobe Technologies.
- *I Have Become Comfortably Numb*,
<http://guillemborrell.es>
- *Introducción Informal a Matlab y Octave*,
<http://iimyo.forja.rediris.es/>

A set of navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

¿Qué es Matlab?

- Un lenguaje de programación
- Un lenguaje de programación *interpretado*
- Un lenguaje de programación *interactivo*

Usar Matlab == Programar en Matlab



¿Qué no es Matlab

- Una hoja de cálculo
- Un programa de cálculo simbólico. Matlab puede hacer $\int_0^1 \text{erf}(x) dx = 0.486$ pero no $\int \text{erf}(x) dx = x \text{erf}(x) + \frac{e^{-x^2}}{\sqrt{\pi}}$
- La solución a todos nuestros problemas.



¿Qué significa interpretado?

- Un intérprete es un programa.
- Es como un actor que hace todo lo que le dice un *guión*
- Muy parecido a la una calculadora.
- Es interactivo.

1

```
>>
```

Os presento a la consola de Matlab

Algunas mentiras

- Para ser ingeniero aeronáutico no es necesario saber programar.
- Programar es difícil.
- Programar *bien* es fácil.
- Los ingenieros programan bien
- En la vida basta un lenguaje de programación mientras se domine.

Un autoengaño

Si en la escuela sólo me dan seis créditos de informática es porque no es importante.



En Arquitectura nadie enseña Autocad.



Problema:

Representar $I(y)$, la integral de la función de Bessel

$$I(y) = \int_0^y J_{2.5}(x) dx$$

con $y \in [1, 5]$

- ¿Cómo se haría en Fortran?
- ¿Cómo se haría en Excel?

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

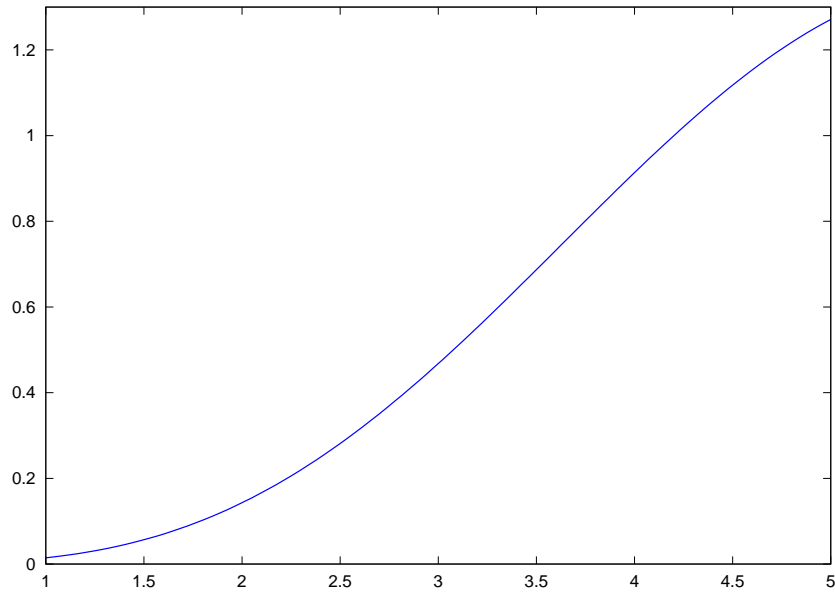
En Matlab son 6 líneas

```
1 x=linspace(1,5,100);  
2 intbessel=@(y) quad(@(x) besselj(2.5,x),0,y);  
3 for i=1:100  
4     z(i)=intbessel(x(i));  
5 end  
6 plot(x,z);
```

No os preocupéis si no entendéis nada. Esto es Matlab avanzado.

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

El resultado



A set of navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

¿Una calculadora programable?

```
1 >> 2+2
2 ans = 4
3 >> mean([1,2,3,4,5,6,7,8,9])
4 ans = 5
5 >> abs(3+4i)
6 ans = 5
```

◀ ◻ ▶ ◀ ◻ ◻ ▶ ◀ ≡ ≡ ▶ ◀ ≡ ≡ ▶ ≡ ≡ ≡ ↺ 🔍 ↻

Todo esto es muy bonito pero...

- ¿Es una herramienta realmente útil?
- ¿Se usa masivamente en la industria?
- ¿Por qué?
- ¿Cuánto cuesta Matlab?
- ¿Es la única solución?

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Octave

- Implementación libre y gratuita del lenguaje Matlab
- `http://www.octave.org`
- Programa muy utilizado en GNU/Linux
- Versiones para Windows y Mac
- QtOctave
- *Libre y gratuito*

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

El lenguaje Matlab

- Caracteres especiales
- Funciones y scripts
- Tipos
- Variables
- Operadores
- Sentencias
- Contenedores
 - *Function handles*

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Caracteres especiales

```
1 >> % Este comando sera ignorado
2 >> 'hola' % 'Hola,Matlab!'
3 ans = hola
4
5 >> 'hola';
6 >> 'hola', 'que_tal'
7 ans = hola
8 ans = que tal
9
10 >> 'hola', ...
11 'que_tal'
12 ans = hola
13 ans = que tal
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

El directorio de trabajo

- Matlab puede ejecutar archivos con código
- Matlab puede cargar archivos de datos
- La biblioteca de funciones está formada por archivos con código.
- Matlab busca en sus directorios de sistema más el *directorio de trabajo*
- Variable `path`



Funciones. Sintaxis

```
1 function [sal1,sal2,...] = nombre(ent1,ent2,...)
2     sentencias ejecutables
3     sal1 = ...
4     sal2 = ...
```

Lo guardaremos todo en *el directorio de trabajo*
en un archivo llamado *nombre.m*



Scripts

- Un script es un programa
- Un programa es una secuencia de instrucciones ejecutables
- Un programa no depende de variables externas
- También se guarda en un archivo `.m` en el *directorio de trabajo*
- Se ejecuta escribiendo el nombre del archivo en la consola o pulsando `F5` en el editor.



Nuestra primera función

Abrimos un archivo nuevo en el editor

```
1 function y = aprsin(x)
2     y=x-(x.^3)/6
```

Y lo guardamos en el directorio de trabajo como `aprsin.m`.



Nuestro primer script

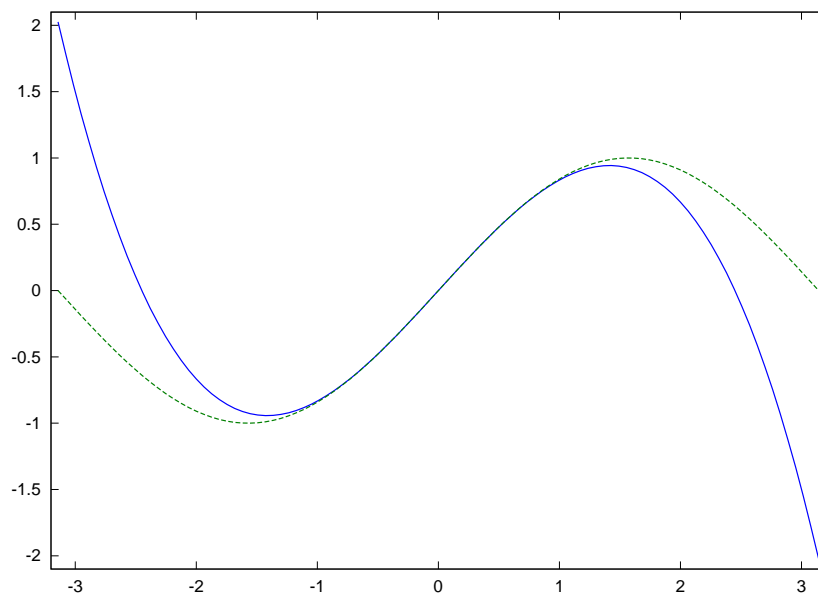
En un archivo nuevo del editor

```
1 x=linspace(-pi,pi,100);  
2 for i = 1:100  
3     y(i)=aprsin(x(i));  
4 end  
5 plot(x,[y;sin(x)])
```

Lo guardamos con el nombre `comparar.m` *en el directorio de trabajo*. Luego pulsamos F5

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

El resultado



◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Ayuda. Función `help`

- En Matlab todo es una función
- Cada función contiene una pequeña ayuda
- Para consultar la ayuda existe la función `help`

```
1 help eig
```

Tipos

Es importante diferenciar los conceptos de

Tipo Cualquier elemento de un código tiene un tipo: caracteres, números, matrices...

Variable Identificador asignado a un tipo o a un contenedor

Argumento Variable de entrada o salida de una unidad de programa

Tipos numéricos

- Tipo por defecto: arrays n-dimensionales de doble precisión
- Simple precisión
- Enteros de varios bits

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

Mira qué curioso

```
1 >> a = pi
2 a = 3.1416
3 >> a(1)
4 ans = 3.1416
5 >> a(1,1)
6 ans = 3.1416
7 >> a(1,1,1)
8 ans = 3.1416
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

Escribir matrices

- El espacio o la coma separan elementos de la misma fila
- El retorno de carro o el punto y coma separa filas

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ↺ 🔍 ↻

Ejercicio 1

1 `M=[1, 2, 3; 4, 5, 6; 7, 8, 9];`

Escribir

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

de otros 3 modos posibles.

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ↺ 🔍 ↻

Subíndices

- En Matlab el primer índice cuenta elementos en la columna
- El segundo índice cuenta elementos en la fila
- *Pero* un vector es siempre fila a no ser que se diga lo contrario.
- El truco es que no nos preocupen las filas y las columnas, sólo los índices

$$M_{ij} = M(i, j)$$

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ↺ 🔍 ↻

No hay quien te entienda

```

1 >> v(4)=2
2 v =
3
4      0      0      0      2
5
6 >> w(4,1)=2
7 w =
8
9      0
10     0
11     0
12     2

```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺ ↻

Un vector...

$$v = (1, 2, 3, 4, 5)$$

```
1 >> v=[1,2,3,4,5];  
2 >> v(4)  
3 ans = 4
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ 🔍 ↺

Una matriz...

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
1 >> M=[1,2,3;4,5,6;7,8,9];  
2 >> M(2,3)  
3 ans = 6
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ 🔍 ↺

Podemos indexar con vectores

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
1 >> M([1,2],[2,3])
2 ans =
3
4     2     3
5     5     6
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

O con índices mudos

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
1 >> M(2,:)
2 ans =
3
4     4     5     6
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Secuencias

- Es una abreviatura común para escribir un vector fila
- La sintaxis es `inicio:incremento:final`

```
1 >> 0:2:10
2 ans =
3     0     2     4     6     8    10
4
5 >> 0:5
6 ans =
7     0     1     2     3     4     5
```



Ejercicio 2

Crear la matriz siguiente y extraer de ella la submatriz marcada en rojo.

$$\begin{pmatrix} 11 & 12 & 13 & 14 & 15 \\ 21 & 22 & 23 & 24 & 25 \\ 31 & 32 & 33 & 34 & 35 \\ 41 & 42 & 43 & 44 & 45 \\ 51 & 52 & 53 & 54 & 55 \end{pmatrix}$$



Otros tipos

- La unidad imaginaria es i , j , I o J
- Las cadenas de texto se introducen entre comillas simples
- Los tipos lógicos son *true* y *false*. *true* es $\neq 0$ y *false* es $\equiv 0$



Operadores

- Operadores matriciales $+$, $-$, $*$, $/$, $^$
- Operadores escalares $.*$, $./$, $.^$
- Operadores lógicos matriciales $\&$, $|$, $!$
- Relaciones de comparación $<$, $>$, $==$, $<=$, $>=$, \neq
- Relaciones lógicas $\&\&$, $||$



El error más común de Matlab

```
1 >> a=rand(3,3);
2 >> a=rand(3,3);b=rand(3,3);
3 >> a*b
4 ans =
5     1.0297     0.9105     0.3293
6     0.9663     0.8267     0.4211
7     0.5355     0.4318     0.3279
8 >> a.*b
9 ans =
10     0.1824     0.3253     0.0563
11     0.5500     0.6003     0.1897
12     0.0458     0.0017     0.1822
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

El error más común de Matlab

```
1 >> a=[1,2,3;4,5,6;7,8,9];
2 >> a.^pi
3 ans =
4     1.0000     8.8250    31.5443
5    77.8802   156.9925   278.3776
6   451.8079   687.2913   995.0416
7 >> a^pi
8 ans =
9    1.0e+03 *
10    0.69 - 0.0004i  0.85 - 0.0001i  1.01 + 0.0002i
11    1.57 - 0.0000i  1.93 - 0.0000i  2.29 + 0.0000i
12    2.45 + 0.0003i  3.01 + 0.0001i  3.57 - 0.0002i
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

Ejercicio 3



Control de flujo

- Las sentencias son palabras clave necesarias para programar
- Son comunes a la mayoría de lenguajes de programación
- Control de flujo es el uso de bucles, condicionales, casos...
- El control de flujo implica el encapsulamiento de una tarea.



Condicionales o `if`

```
1 if cond
2   sentencias
3 elseif cond
4   sentencias
5 else
6   sentencias
7 end
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Bucles o `for`

```
1 for var=contador
2   sentencias
3 end
```

- `contador` puede ser un vector o una secuencia
- Para cada paso `i` avanza de valor en `contador`
- Las sentencias pueden depender o no de `i`

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Más control de flujo

case Control de casos finitos

while Bucle controlado por condición lógica

try Control de excepciones

break Salida de bloques

continue Idem

return vuelta al programa principal

Ejercicio 4

Contenedores

- Estructuras de datos. Forma de árbol
- Cell arrays. Forma de matriz
- Function handles. Contenedor para una función.
 - Funciones anónimas



Estructuras de datos

```
1 >> ed.num=1.234;
2 >> ed.str='hola';
3 >> ed.logic.true=1;
4 >> ed.logic.false=0;
5 >> ed
6
7 ed =
8
9         str: 'hola'
10        num: 1.2340
11        logic: [1x1 struct]
```



Cell arrays

```
1 >> celda={1.234,'hola';true,false}
2 celda =
3     [1.2340]     'hola'
4     [         1]     [         0]
5
6 >> celda{1,1}
7 ans = 1.2340
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Function Handles

Es capaz de contener una función.

```
1 >> fhsin = @sin
2 fhsin =
3
4     @sin
5
6 >> fhsin(pi/2)
7 ans = 1
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

A set of navigation controls at the bottom of the slide, including arrows for previous/next slide, a square icon, a stack of squares icon, a list icon, a search icon, and a refresh icon.

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

Conclusiones

- El lenguaje Matlab es muy limitado
- Es sencillor su sintaxis es clara
- Sus estructuras son muy matemáticas
- Está basdo en funcionesy no conocemos ninguna
- La biblioteca de funciones de Matlab es tan grande como quieras pagarla.



Creación de matrices

eye matriz de ceros con unos en la diagonal

linspace Vector de elementos equiespaciados

logspace Vector de elementos con el exponente equiespaciado

meshgrid Matrices de elementos equiespaciados en 2D

ones Matriz de unos

zeros Matriz de ceros

rand Matriz de números aleatorios



Manipulación de matrices

reshape Cambia la forma de la matriz conservando el número de elementos

transpose Traspuesta. Equivale a '.

ctranspose Matriz conjugada. Equivale a '

rot90 Gira la matriz 90 grados hacia la izquierda



Ejercicio 6



Resolución de SEL

Para resolver sistemas de ecuaciones lineales contamos con un operador universal

```
1 >> A=[1,0;2,1];y=[2;4];  
2 >> x=A\y  
3 x =  
4  
5     2  
6     0
```

Cálculo Simbólico

- Podéis hacer operaciones simbólicas con Matlab

Cálculo Simbólico

- Podéis hacer operaciones simbólicas con Matlab
- Que pueda hacerse no significa que tenga que hacerse

Cálculo Simbólico

- Podéis hacer operaciones simbólicas con Matlab
- Que pueda hacerse no significa que tenga que hacerse
- También podéis depilaros las cejas con una sierra mecánica

Integración numérica

quad Integración numérica. 3 argumentos de entrada

quadl Algoritmo de integración mejorado

dblquad Integración bidimensional de funciones de dos variables

trapz Regla del trapecio

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ ↺ 🔍 ↻

Ejercicio 7

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ ↺ 🔍 ↻

Desarrollos en serie de funciones

- Los coeficientes de un desarrollo en serie son un vector

$x^2 + x$ es $1x^2 + 1x + 0$, es decir $[1 \ 1 \ 0]$

```
1 >> p = [1 1 0];
2 >> polyval(p,1)
3 ans = 2
4 >> roots(p)
5 ans =
6     -1
7      0
```



Polinomios

polyval Obtiene el valor en un punto

roots Obtiene las raíces del polinomio

polyder Deriva un polinomio

polyinteg Integra un polinomio

conv Multiplica dos polinomios

residue Desarrollo en fracciones parciales.



Funciones que devuelven polinomios

poly Obtiene el polinomio característico de una matriz.

polyfit Modelo polinómico de una serie de datos.



Representación gráfica

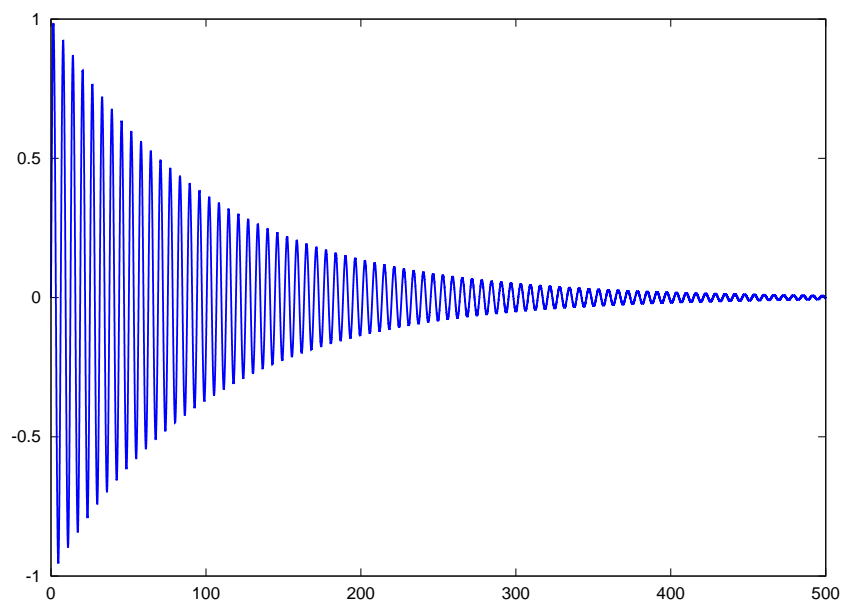
- Representar datos es sencillo e intuitivo
- No hay que emocionarse con la representación gráfica
- Sólo veremos curvas en el plano
- ¿Necesitamos más?



Plot

Representa curvas en el plano. $e^{-x/100} \sin x$ para $x \in [0, 500]$

```
1 >> x=linspace(0,500,100000);
2 >> plot(x,exp(-x/100).*sin(x))
```



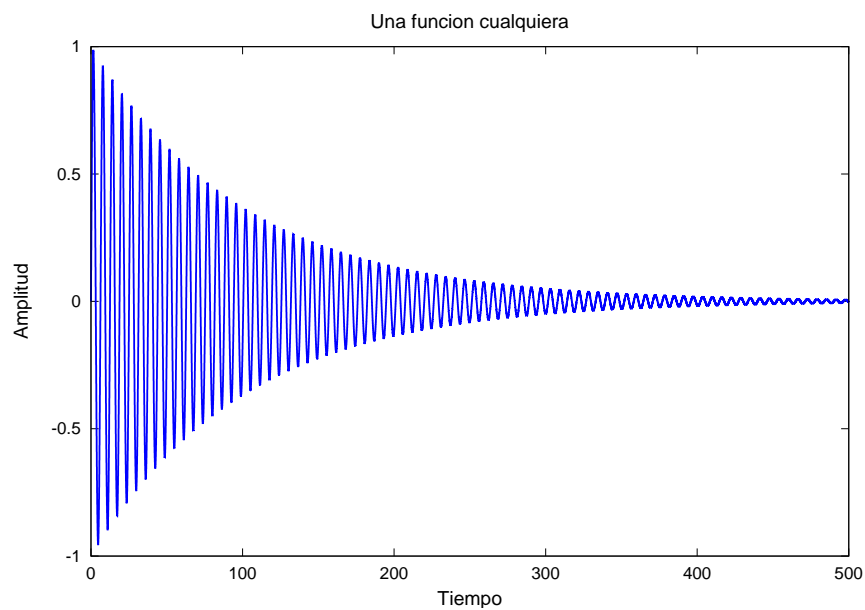
El resultado

Etiquetas

```
1 >> title('Una_funcion_cualquiera')
2 >> xlabel('Tiempo')
3 >> ylabel('Amplitud')
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ ↺ 🔍 ↻

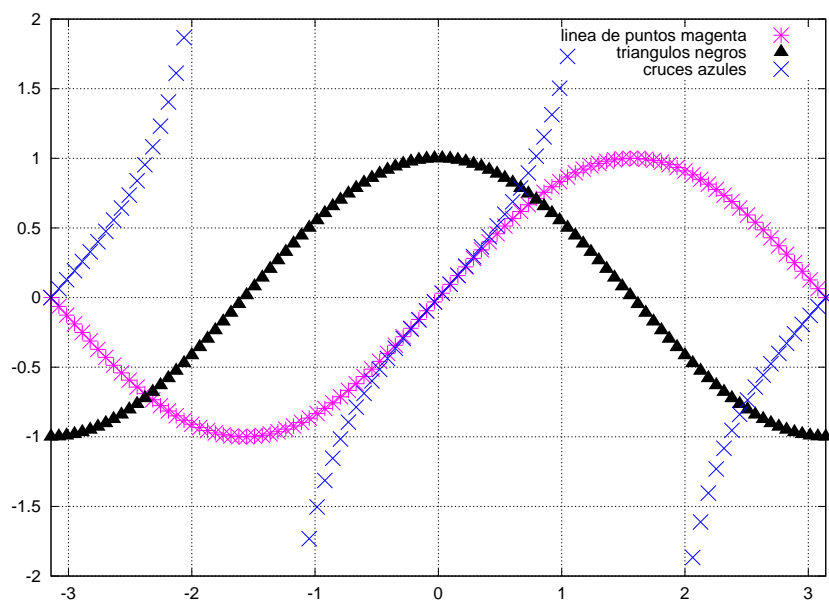
El resultado



◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ ↺ 🔍 ↻

Estilos

```
1 >> x=linspace(-pi,pi,100);
2 >> plot(x,sin(x),'m:',...
3 x,cos(x),'k^',x,tan(x),'bx')
4 >> axis([-pi,pi,-2,2])
5 >> grid on
6 >> legend('linea_de_puntos_magenta',...
7           'triangulos_negros',...
8           'cruces_azules')
```



El resultado

hold

- La ventana gráfica se borra automáticamente cada vez que dibujamos algo
- Para cambiar el comportamiento anterior se usa la función *hold*
 - `hold on` mantiene todo lo dibujado en la pantalla
 - `hold off` vuelve al comportamiento inicial
- Para borrar la ventana gráfica usamos `clf`

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

figure

- Las ventanas gráficas se manipulan con la función `figure`
- Cada ventana gráfica tiene asociada un número entero
 - `figure` se llama con un número que corresponde al de la ventana
 - Si utilizamos un número que no corresponde a ninguna ventana existente crearemos una nueva con este número asociado
 - Si utilizamos un número existente activaremos la ventana correspondiente

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

subplot

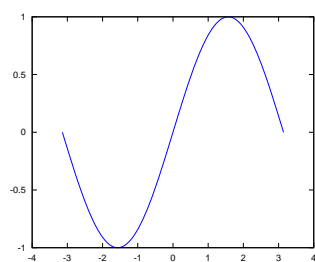
Es el comando que permite poner varios ejes en una misma figura

```
1 >> x= linspace(-pi,pi,100);  
2 >> subplot(2,2,1)  
3 >> plot(x,sin(x))
```

Primero de los cuatro sectores.

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

El resultado

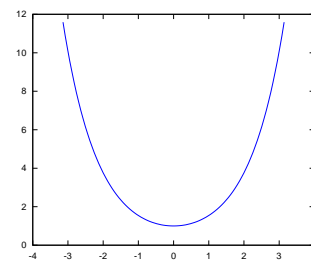
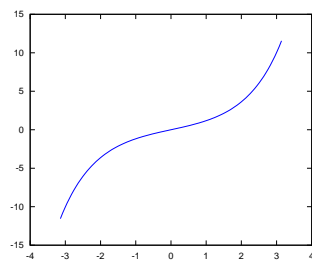
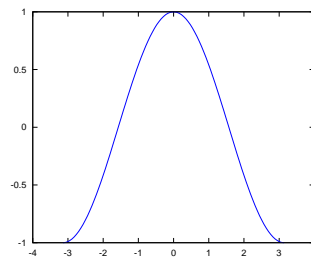
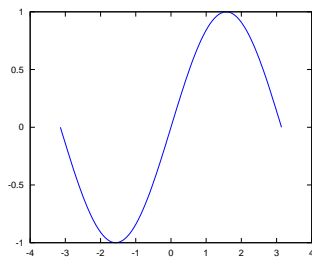


◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

subplot

Ahora completamos los cuatro cuadrantes.

```
1 >> subplot(2,2,2)
2 >> plot(x,cos(x))
3 >> subplot(2,2,3)
4 >> plot(x,sinh(x))
5 >> subplot(2,2,4)
6 >> plot(x,cosh(x))
```



El resultado

Otros comandos

semilogx Dibuja una curva con el eje x en escala logarítmica

semilogy Dibuja una curva con el eje y en escala logarítmica

loglog Dibuja una curva en escala logarítmica

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ ↺ 🔍 ↻

Ejercicio 8

Representar en una misma ventana y dos frames (uno superior y otro inferior) la función

$$\sqrt{x} \sin(1/x) \quad x \in [0.001, 1]$$

en escala normal y en escala semilogarítmica en el eje x.

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ ↺ 🔍 ↻

Comandos interesantes

get, set Cambia los atributos de un plot handle

text Ponete texto en la figura

contour Isolíneas de una matriz de datos en 3D

griddata Interpola para el contour



Desarrollos de datos

interp1 Interpolación de una serie de puntos

interp2 Interpolación de una nube de puntos

`polyfit` Coeficientes del polinomio de grado n
con mínimo error cuadrático

`fft` Realiza la transformada de Fourier



interp1

```
1 >> x=[1 2 3 4 5 6 7 8];
2 >> y=[1 4 2 5 7 4 2 7];
3 >> interp1(x,y,7.234,'spline')
4 ans = 2.3437
5 >> test=@(x,y,z) interp1(x,y,z,'spline');
6 >> test(x,y,7.234)
7 ans = 2.3437
```

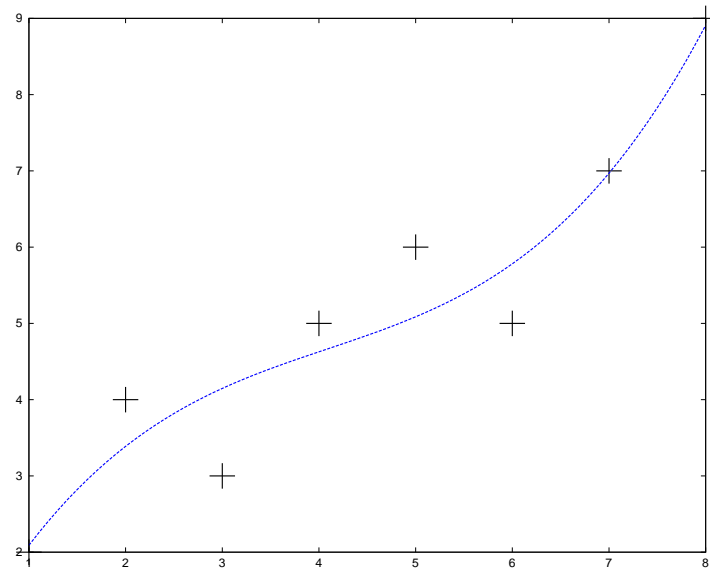
◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

polyfit

```
1 >> x=[1 2 3 4 5 6 7 8];
2 >> y=[2 4 3 5 6 5 7 9];
3 >> coeff=polyfit(x,y,3);
4 >> plot(x,y,'k+',1:0.1:8,...
5 polyval(coeff,1:0.1:8),'b-')
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

El resultado



A set of navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

Estadística descriptiva

mean Media

std Desviación típica

median Mediana

sort Orena los elementos de menor a mayor

center Elimina la media de una muestra

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ◻ ◻ ◻

EDO

- Es una de las aplicaciones más importantes del Cálculo Numérico
- Los problemas más comunes son los problemas de Cauchy no lineales
- En ese caso la solución numérica es esencial
- Lo más importante es saber si nuestro problema es stiff



Stiff

- Un problema es *stiff* cuando el paso temporal viene determinado por la estabilidad del esquema.
- Suelen relacionarse con problemas no lineales o condiciones de contorno muy exigentes
- Requieren esquemas de integración temporal implícitos



Funciones

ode45 Runge-Kutta de paso variable orden 4-5

ode113 Adams multipaso

ode23s Esquema implícito Rosenbrock

lsode Octave



Van der Pol

Un caso típico es la ecuación de Van der Pol

$$x'' + x + \mu(x'^2 - 1)x = 0$$

Dependiendo del valor de μ el problema va a ser *stiff* o no.

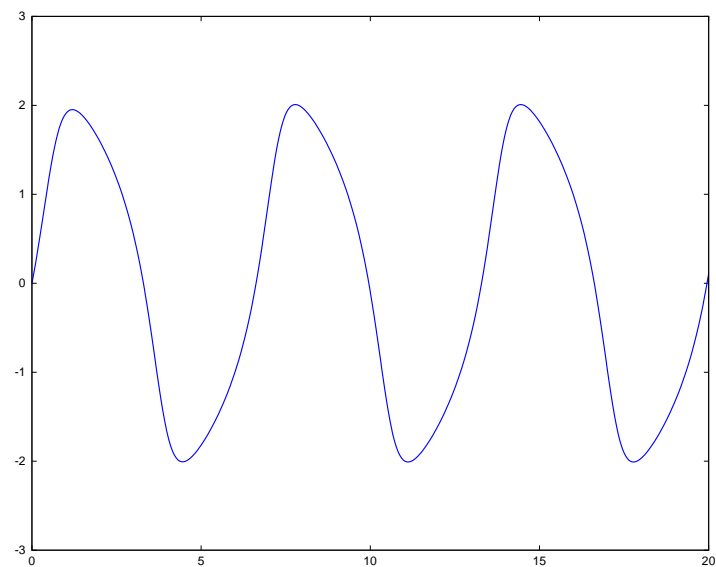


Solución

```
1 >> [tout,xout]=ode45(@vdp1,[0 20],[2 0])  
2 >> plot(tout,xout(:,1))
```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ 🔍 ↺

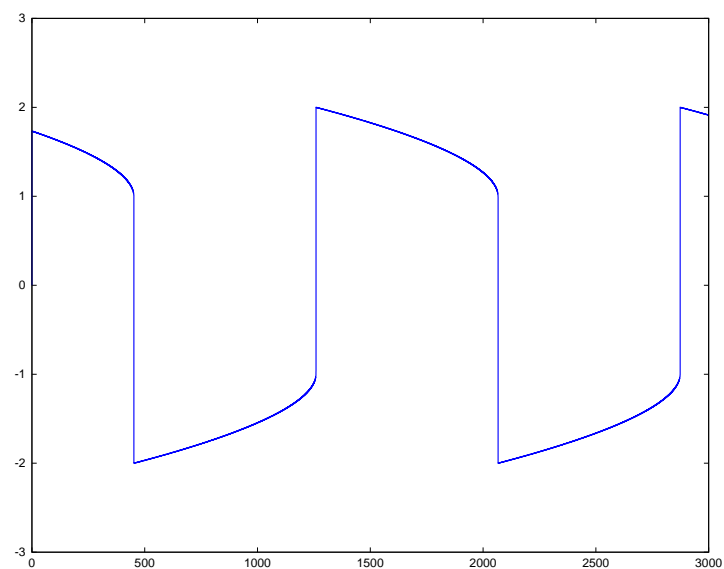
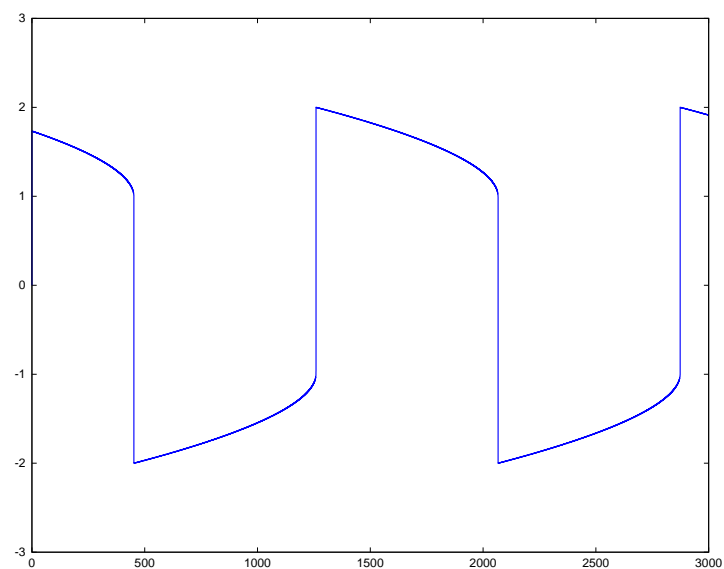
Solución



◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ 🔍 ↺

Solución

```
1 >> [tout,xout]=ode23s(@vdp1000,[0 20],[2 0])
2 >> plot(tout,xout(:,1))
```



Ejercicio 9