Práctica 1-1. Introducción a MATLAB

Matemáticas 2. Ingeniería Informática

Introducción

¿Qué es MATLAB?

MATLAB es una de las muchas herramientas de computación disponibles para resolver problemas matemáticas, tales como Maple, Mathematica, MathCad, Maxima, OpenAxiom, etc. Todas tienen fortalezas y debilidades y cada una permite efectuar cálculos matemáticos básicos, pero difieren en el modo como manejan los cálculos simbólicos y procesos matemáticos más complicados como la manipulación de matrices grandes.

Introducción

¿Porqué MATLAB?

Lenguaje de alto nivel para la codificación de **objetos matemáticos** tales como funciones o matrices, así como **operar** con ellos en un entorno amigable.

MATLAB es el nombre abreviado de MATrix LABoratory y fue ideado para proporcionar un fácil manejo de matrices. Hoy en día, MATLAB es una potente herramienta para todo tipo de problemas.

Algunas características son

- *Calculadora numérica:* Evalúa, numéricamente, expresiones complejas
- *Calculadora Simbólica:* Puede manipular expresiones simbólicas (por ejemplo calculo de limites o derivadas) y resolver ecuaciones

Introducción

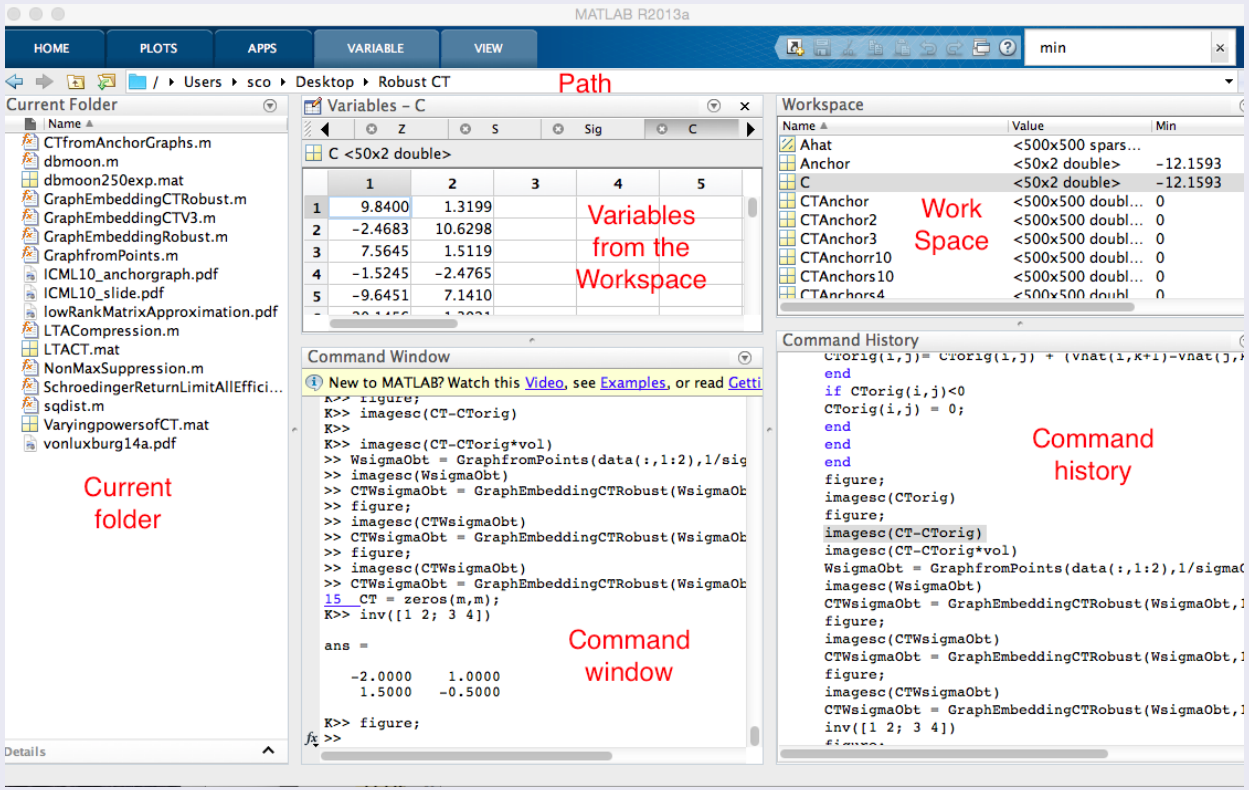
¿Porqué MATLAB?

Más....

- *Programación:* Es un Lenguaje de Programación procedimental con un entorno de depuración
- *Datos de tipo estructurados:* Provee objetos predefinidos así como objetos definidos por el usuario
- *Recursos Gráficos*: Posee herramientas para el análisis gráfico
- *Toolboxes:* Posee librerías especializadas agrupadas en Toolboxes

Introducción

Entorno de Trabajo



Introducción

Entorno de Trabajo

MATLAB utiliza varias subventanas de despliegue.

- Ventana de comandos (Command window)
- Directorio actual (Current folder)
- Área de trabajo (Workspace)
- Historia de comandos (Command history)

Cuando se necesita, se abren automáticamente ventanas como la de gráficos (Figure) o edición (Editor).
Command window y Workspace son las subventanas principales.
En la subventana de comandos (Command window) se escriben las órdenes que se desea ejecutar, a la derecha del símbolo >>

Introducción

Comandos de entorno

Para comenzar a usar MATLAB, sólo se necesita prestar atención a la ventana de comandos. En ella puede realizar cálculos en forma similar a como se hace en una calculadora científica.

Introducción

Comandos de entorno

- Para guardar la sesión de trabajo en un archivo
  >> *diary nombre archivo*
- Este archivo aparece en el directorio actual (Current Folder) y se puede abrir con el Editor
- Para finalizar la sesión de trabajo
  >> *diary off*
- Los comandos utilizados se guardan en la ventana Command history

Introducción

Comandos de entorno

- Para teclear un comando el cursor tiene que situarse a la derecha de >>
- Una vez tecleado el comando se pulsa la tecla *intro*
- Para incluir comentarios se utiliza el símbolo %
- Para introducir varias órdenes, se separan por comas. Las ordenes se ejecutan de izquierda a derecha
- Para no mostrar el resultado en pantalla se finaliza con ;
- Las flechas arriba y abajo recuperan líneas ya escritas
- Las flechas izquierda y derecha sirven para moverse en la misma linea

Introducción

Comandos de entorno

Más....

- Si un comando es demasiado grande para una línea (límite 4096 caracteres) se puede dividir tecleando tres puntos seguidos y pulsando *intro*. Luego se continúa la escritura del comando en la línea siguiente
- Cuando se sale de MATLAB, o cuando se ejecuta el comando *clc*, la ventana de comandos se limpia

$>> B = 15, H = 3$ %B es la base y H la altura
$>> 30/5 + 12/6 + 15/3 + 125/5 + 40/8 + 36/2 + 45/9$ ...
$-25 + 24/12;$

Introducción

Command History

La ventana Command History es valiosa puesto que

- Permite revisar sesiones anteriores de MATLAB
- Se puede usar para transferir comandos a la ventana de comandos haciendo doble clic o arrastrándolos

Conforme se ejecuten cálculos más complicados se irá viendo la utilidad de esta ventana.

Introducción

Workspace

La ventana Workspace informa de las variables que están definidas.
Esta ventana puede dar más información sobre las variables al hacer clic con el botón derecho sobre la barra con las etiquetas de las columnas.
El comando *clear* borra todas las variables guardadas.
Si se cierra el Workspace se puede saber las variables definidas mediante el comando *whos*.

Introducción

Current Folder

La ventana Current Folder contiene los archivos del directorio de trabajo.
Cuando MATLAB guarda información, usa este directorio.
La ubicación aparece en la parte superior de la ventana principal.
El Current Folder se puede cambiar.

Introducción

Otras subventanas

Hacer doble clic sobre cualquier variable de la ventana del área de trabajo se abre una subventana llamada *Variables* que contiene el editor de variables.
La ventana Graphics se abre automáticamente cuando realiza un gráfico.
La ventana Editor se abre al al crear un archivo nuevo.

Introducción

Ayudar

- Menú contextual que se abre *Help on Selection*
- Tecleando F1
- Tecleando la orden *help*
- Para obtener información específica sobre alguna función *help nombre función*
- Para que la ayuda aparezca en una ventana independiente *helpwin plot* (en Octave *doc*)
- Si se teclean las primeras letras del nombre de un comando y pulsamos la tecla del tabulador aparece un menú de opciones

Introducción

Toolboxes

Los comandos que guardan relación temática, se agrupan en librerías específicas llamadas Toolboxes.
Estás librerías pueden estar instaladas o no.
Con el comando *ver* se pueden ver las *Toolbox* instaladas

Calculadora

Línea de Comandos

- Si se teclea $>> 2+3$
- Se obtiene

  $ans =$

  5
- *ans* es la respuesta actual. Si se teclea $>>$ *ans* se obtiene

  $ans =$

  5
- $>> a = 3+2$; el resultado se almacena en la variable *a*

Calculadora

Operadores aritméticos

- $(+)$: suma
- $(-)$: resta
- $(*)$: multiplicación
- $(/)$: división
- $(\wedge)$: potencia

El orden de prioridad de las operaciones es el mismo que en las calculadoras. Si se desea alterar este orden se deben introducir paréntesis.

Cuando los operadores van precedidos de un punto deben entenderse en el sentido de que la operación se efectúa elemento a elemento

Calculadora

Operadores lógicos

- (>): mayor que
- (<): menor que
- (>=): mayor o igual que
- (<=): menor o igual que
- (==): igual a
- (~=): diferente de

Se pueden agrupar con los operadores AND (& o &&) y OR (| o ||)

Calculadora

Ejemplo

```
>> 8 + 4/2 - 1
ans =
9
>> (8 + 4)/2 - 1
ans =
5
>> ((8 + 4) * (2 - 6) - 10)/(3 - 1)
ans =
-29
```

Calculadora

Ejemplo

```
>> 3*4^2-1
ans =
47
>> (3*4)^2-1
ans =
143
```

Introducción

Funciones elementales

`sqrt(x)`	raiz cuadrada	`sin(x)`	seno (radianes)
`abs(x)`	módulo	`cos(x)`	coseno (radianes)
`conj(z)`	complejo conjugado	`tan(z)`	tangente (radianes)
`real(z)`	parte real	`cotg(x)`	cotangente (radianes)
`imag(z)`	parte imaginaria	`asin(x)`	arcoseno
`exp(x)`	exponencial	`acos(x)`	arcocoseno
`log(x)`	logaritmo natural	`atan(x)`	arcotangente
`log10(x)`	logaritmo decimal	`cosh(x)`	cos. hiperbólico
`rat(x)`	aprox. racional	`sinh(x)`	seno hiperbólico
`mod(x,y)` `rem(x,y)`	resto de dividir x por y . Iguales si x,y>0 . Ver help para definición exacta	`tanh(x)`	tangente hiperbólica
`fix(x)`	Redondeo hacia 0	`acosh(x)`	arcocoseno hiperb.
`ceil(x)`	Redondeo hacia + infinito	`asinh(x)`	arcoseno hiperb.
`floor(x)`	Redondeo hacia - infinito	`atanh(x)`	arcotangente hiperb.
`round(x)`	Redondeo al entero más próximo		

Calculadora

Funciones de cálculo del resto

- *mod*: resto de división entera sin signo
  $>>$ *mod*$(-1{:}5, 3)$ devuelve 2 0 1 2 0 1 2
- *rem*: resto de división entera con signo
  $>>$ *rem*$(-1{:}5, -3)$ devuelve -1 0 1 2 0 1 2
- *sign*: devuelve $1, 0$ o -1 según el signo sea positivo, cero o negativo
  $>>$ *sign*(5), *sign*(0), *sign*(-5), devuelve -1 0 o 1 respectivamente

Calculadora

Formatos de Representación

Los formatos de representación numérica en pantalla pueden ser

1. *format short*: 3 cifras (como máximo) en la parte entera y 4 en la decimal
2. *format rat*: número racional
3. *format long*: 2 cifras en la parte entera y 15 en la decimal
4. *format short e*: 1 cifra en la parte entera, 4 en la decimal y 3 en el exponente
5. *format long e*: 1 cifra en la parte entera, 15 en la decimal y 3 en el exponente

Aunque cambie el formato, solo es apariencia.

Calculadora

Representación numérica

MATLAB almacena internamente los números en doble precisión (ocho bytes). Eso significa que solo se representan números reales en un determinado rango: entre 10^{-308} y $2 * 10^{308}$.
A cualquier número menor que el menor número representado (*realmin*) el programa lo considera como cero y a cualquier número de una magnitud mayor que el mayor número representado (*realmax*) el programa lo considera de magnitud infinita (Inf o inf).

Calculadora

Ejemplo.

a) $>> x = 0 : 0.01 : 2 * pi$; % discretiza del intervalo $[0, 2\pi]$ cada 0,01 unidades

b) $>> y1 = sin(x)$; % almacena en $y1$ un vector con los valores de sin(x)

c) $>> size(x, 2)$, $size(y1, 2)$ %devuelve el tamaño de los vectores $x, y1$

d) $>> y2 = cos(x)$; % almacena en $y2$ un vector con los valores de cos(x)

e) $>> plot(x, y1)$ % realiza un gráfico de los puntos $(x, y1)$

f) $>> hold\ \ on$; % mantiene el gráfico en la misma ventana

g) $>> plot(x, y2, 'color', 'r')$ % dibuja los puntos $(x, y2)$ en color rojo

Tipos de datos y variables

Variables

- Una variable es una etiqueta que identifica un trozo de memoria
- Para asignar el nombre de una variable a un ente (número, vector, matriz o carácter) se debe utilizar el operador de asignación $=$
- No es necesario declarar los nombres de la variables ni sus tipos
- El nombre de una variable debe empezar por una letra y puede contener letras, números y guión bajo
- No se permiten espacios y como máximo puede contener 63 caracteres
- Distingue mayúsculas y minúsculas

Tipos de datos y variables

Variables Especiales (no se deben usar)

- *eps*: es el epsilon de la máquina ($2.2204e-16$)
- *pi*: es el número pi ($3.14159\ldots$)
- *i, j*: unidad imaginaria
- *inf*: infinito
- *NaN*: no es número
- *exp(1)*: es el número e
- *date*: representa la fecha
- *nargin*: número de argumentos de entrada a una función
- *nargout*: número de argumentos de salida de una función

Tipos de datos y variables

Ejemplo. Volumen de una esfera

a) $>>$ *clear*;

b) $>> r = 2$

c) $>> vol = (4/3) * pi * r\hat{}3$

Tipos de datos y variables

Cadenas de caracteres

MATLAB puede definir variables que contengan cadenas de caracteres. Las cadenas de texto van entre apostrofes y se almacenan en un vector, con un carácter por elemento.

Por ejemplo

$>>$ $fahr = 70, grd = (fahr - 32)/1.8;$

$>>$ title(['Temp. Amb: ', num2str(grd), 'grados centigrados'])

Tipos de datos y variables

Tipos de variables

Para recordar que variables se han definido: >> *who*
Si tecleamos >> *whos* se incluyen también el tipo de variable.
Para saber si ya esta definida una variable >> *exist*('var')
Para borrar una variable *clear* seguido del nombre de la variable (o variables, separados por espacios en blanco).
El comando *clear* borra todas las variables del área de trabajo (workspace).

Introducción

Funciones que generan vectores

`linspace(a,b,n)`	Si **a** y **b** son números reales y **n** un número entero, genera una partición regular del intervalo **[a,b]** con **n** nodos (**n**-1 subintervalos)
`linspace(a,b)`	Como el anterior, pero se asume **n**=100
`logspace(e,f,n)`	Vector con **n** elementos logarítmicamente espaciados desde 10^**e** hasta 10^**f**, es decir, cuyos logaritmos están regularmente espaciados. (`logspace(e,f,n) = 10.^linspace(e,f,n)`)
`logspace(e,f)`	Como el anterior, pero se asume **n**=50

Introducción

Funciones que generan vectores

`diag(v)`	Si **v** es un vector, **diag(v)** es una matriz cuadrada de ceros con diagonal principal = **v**
`diag(A)`	Si **A** es una matriz, **diag(A)** es un vector = diagonal principal de **A**
`diag(A,k)`	Si **A** es una matriz y **k** es un entero, **diag(A,k)** es un vector = **k**-ésima sub o super diagonal de **A** (según sea **k**<0 ó **k**>0)
`blkdiag(A,...,K)`	Construye una matriz diagonal por bloques con las matrices **A**, ... **K**
`triu(A)` `tril(A)`	Extrae la parte triangular superior (inferior) de la matriz **A**
`triu(A,k)` `tril(A,k)`	Extrae la parte superior (inferior) de la matriz **A**, desde la **k**-ésima diagonal hacia arriba (abajo)
`zeros(n,m)`	matriz **n**x**m** con todas sus componentes iguales a cero.
`ones(n,m)`	matriz **n**x**m** con todas sus componentes iguales a uno
`eye(n,m)`	matriz unidad: matriz **n**x**m** con diagonal principal =1 y el resto de las componentes =0
`reshape(A,n,m)`	Re-dimensiona una matriz: si **A** es una matriz **h**x**k**, **reshape(A,n,m)** es otra matriz con los mismos elementos que **A**, pero de dimensiones **n**x**m** (tiene que ser **h***k=**n***m)

Introducción

Funciones que generan vectores

`rand(n)` `rand(n,m)`	Construye una matriz **nxn** ó **nxm** con números aleatorios con distribución uniforme
`randn(n)` `randn(n,m)`	Ídem con distribución normal
`compan(p)`	Es la matriz compañera (ó de Frobenius) del vector **p**, es decir, que tiene como autovalores las raíces de **p**
`magic(n)`	Devuelve una matriz **nxn** que es un cuadrado mágico
`pascal(n)`	matriz de Pascal de dimensión **nxn**
`vander(v)`	matriz de Vandermonde asociada al vector v: **a(i,j) = v(i)^(n-j)**

Tipos de datos y variables

Variables simbólicas

Hay cálculos que se realizan habitualmente en Matemáticas y que no son posibles con las órdenes de MATLAB estudiadas hasta el momento.
Por ejemplo si efectuamos

$$(a+b)(a-b)$$

obtenemos un mensaje de error debido a que las variables a, b no tienen valores asignados.

Tipos de datos y variables

Variables simbólicas

Trabajar con variables simbólicas permite manejar ecuaciones de manera simbólica y resolverlas después reemplazando por un valor cada una de las variables.

Permite resolver, por ejemplo, ecuaciones, integrales definidas o realizar cálculos de primitivas de manera analítica.

Las variables simbólicas son cadenas de caracteres que pueden representar números, funciones, expresiones o variables.

Tipos de datos y variables

Variables simbólicas

Por defecto todas las variables son numéricas.
Para definir un objeto simbólico se utiliza el comando *sym*.
Ejemplo >> $x = sym('x')$ declara x como una variable simbólica.
Para definir variables simbólicas conjuntamente, se debe utilizar *syms*.
Ejemplo >> *syms a b c* declara a, b y c como variables simbólicas.

Tipos de datos y variables

Expresiones simbólicas

Si en una expresión interviene una variable simbólica la expresión es simbólica (aunque también intervengan variables numéricas).
Por ejemplo
$>>$ *syms a b c x*
$>>$ *poli* $= a * x\hat{}2 + b * x + c$
define una expresión simbólica llamada *poli*.
Si se teclea $>>$ *poli* se obtiene la expresión $ax^2 + bx + c$
Para conocer las variables simbólicas de una expresión: *symvar*.

Tipos de datos y variables

Expresiones simbólicas

Si se desea conocer el valor de una expresión simbólica para un valor concreto de la variable se utiliza el comando *subs*.
Por ejemplo $>>$ $subs(poli, x, 1)$ sustituye en la expresión poli el valor de la variable x por 1.
Se pueden hacer substituciones de forma conjunta.
Por ejemplo, si $a = 1$ y $b = 2$, la orden que se utiliza es
$>>$ $subs(poli, [a, b], [1, 2])$
o
$>>$ $subs(poli, \{a, b\}, \{1, 2\})$
proporciona como respuesta $x^2 + 2x + c$

Tipos de datos y variables

Expresiones simbólicas

Para extraer el numerador y denominador de una expresión simbólica racional, se utiliza el comando *numden*.

Por ejemplo

$>> syms\ x; y = (x\text{\^{}}2 - 1)/(2 * (x + 4)\text{\^{}}3)$

$>> numden(y)$

ans =

$x^2 - 1$

$>> [num, den] = numden(y)$

$num =$

$x^2 - 1$

$den = 2 * (x + 4)^3$

Tipos de datos y variables

Expresiones simbólicas

>> *syms x a*

>> *formula1* = 3 * x^2 + 2 * a * x + 15 * x^3 * a^2 + 25 * a

>> *formula2* = (x + a)^3

>> *formula3* = x^3 + x^2 + x + 1

>> *formula4* = (a^3 + 3 * a^2 + 3 * a + 1)/(a^2 + 2 * a + 1)

Para simplificar una expresión simbólica: *simplify*.

>> *simplify*(*formula4*)

Para factorizar una expresión: *factor*.

>> *factor*(*formula3*)

Para desarrollar en sumas: *expand*.

>> *expand*(*formula2*)

Para simular la escritura matemática habitual: *pretty*.

>> *pretty*(*formula1*)

Gráficos 2D

Consideraciones

MATLAB es una potente herramienta para realizar gráficos.
Las gráficas 2D de MATLAB están fundamentalmente orientados a la representación gráfica de vectores (y matrices).
En el caso más sencillo los argumentos básicos del comando *plot* van a ser vectores. Cuando una matriz aparezca como argumento, se considera como un conjunto de vectores columna.
MATLAB utiliza un tipo especial de ventanas (denominada *Figure*) para realizar las operaciones gráficas.

Gráficos 2D

Consideraciones

Ciertos comandos abren una ventana nueva y otros dibujan sobre la ventana activa, bien sustituyendo lo que hubiera en ella, bien añadiendo nuevos elementos gráficos a un dibujo anterior. Otros comandos influyen en la apariencia de los gráficos variando el intervalo de presentación en los ejes cartesianos o incluyendo una cuadricula.

Gráficos 2D

Comando Plot

Si x representa un vector con componentes x_i, entonces el comando *plot(x)* dibuja los pares de puntos (i, x_i) unidos por segmentos

$>> x = [-4 \ -2 \ 0 \ 1 \ 2 \ 3 \ 5];$

$>>$ *plot*(x)

Dibuja los segmentos que unen los puntos

$(1,-4),(2,-2),(3,0),(4,1),(5,2),(6,3),(7,5)$

Si x e y representan vectores con componentes x_i, y_i respectivamente, el comando *plot(x,y)* dibuja los pares de puntos (x_i, y_i) unidos por segmentos

$>> x = [-4 \ -2 \ 0 \ 1 \ 3 \ 5];$

$>> y = [16 \ 4 \ 0 \ 1 \ 9 \ 25];$

$>>$ *plot*(x, y)

Dibuja la poligonal que une los puntos

$(-4,16),(-2,4),(0,0),(1,1),(3,9),(5,25)$

Gráficos 2D

Comando Plot

Dados los puntos $(1,1),(2,-1),(3,7),(-4,4),(5,-2)$, para dibujar la poligonal que los une

$>> x = [1\ 2\ 3\ -4\ 5]; y = [1\ -1\ 7\ 4\ -2]; plot(x,y)$

El comando plot dibuja la poligonal que une puntos del plano generados al utilizar las componentes de dos vectores.

Gráficos 2D

Comando Plot

En la ventana gráfica se puede realizar tareas como

- Se puede añadir un título a la gráfica con *title*
- Se puede incluir etiquetas en los ejes con *xlabel* o *ylabel*
- También se puede introducir texto con *text* en un zona de interés
- En el caso de tener varias curvas, se pueden añadir etiquetas con *legend*

Para todo ello basta acceder al menú de la ventana del gráfico y en *Insert* escoger la opción deseada

Gráficos 2D

Comando Plot

Otra forma de realizar estas tareas es mediante órdenes.

Sintaxis de algunas de estas órdenes

>> *title*('El título')

>> *xlabel*('Eje de abscisas') % con xlabel off se borra.

>> *ylabel*('Eje de ordenadas') % con ylabel off se borra.

>> *text*(a,b,'este es el punto (a,b)')

>> *legend*('texto identificativo de una linea')

La siguiente orden sirve para introducir texto con el ratón

>> *gtext*('este texto va aquí')

Gráficos 2D

Comando Plot

MATLAB utiliza ciertas opciones por defecto sobre la apariencia de un gráfico.
Algunas se pueden modificar, por ejemplo
Igual escala en ambos ejes
>> *axis equal*
Representar una curva en el rectángulo $[a, b] \times [c, d]$
>> *axis*([*a b c d*])
Restaura el rango por defecto
>> *axis auto*
Introducir una cuadrícula
>> *grid on* (grid off la desactiva).

Gráficos 2D

Dibujar conjuntamente varias gráficas

Se puede incluir en el comando *plot* tantos pares de vectores como gráficos a representar.

$>> x = 0 : pi/100 : 3 * pi; y = cos(x); z = cos(3 * x);$

$>> plot(x, y, x, z);$

También se puede utilizar

$>> x = 0 : pi/100 : pi/4; y = tan(x);$

$>> plot(x, y)$

$>> hold\ on$

$>> z = sin(2 * x);$

$>> plot(x, z)$

Se pueden superponer dos gráficas mediante el comando *plotyy*

$>> x = -1 : 0.01 : 12; y = 300 * exp(-0.01 * x); z = 0.2 * exp(-0.1 * x);$

$>> plotyy(x, y, x, z)$

Gráficos 2D

Color

Se puede incluir el color como una opción en el comando plot (entre apóstrofes)

- b: azul (blue)
- c: verde claro (cyan)
- g: verde oscuro (green)
- k: negro (black)
- m: rojo oscuro (magenta)
- r: rojo (red)
- w: blanco (white)
- y: amarillo (yellow)

$>> x = 0 : pi/100 : 3 * pi; y = cos(x);\ \ plot(x, y, 'r')$

Gráficos 2D

Trazo

Se puede incluir el trazo como una opción en el comando plot (entre apóstrofes)

- -: trazado lineal continuo
- $--$: trazado lineal discontinuo
- -.: trazado lineal discontinuo intercalando punto y linea
- : : trazado con linea de puntos
- *: marca *
- +: marca +
- o: círculo
- x: marca con un aspa
- .: un punto

$>> x = 0 : pi/100 : 3 * pi; y = cos(3 * x);\ plot(x, y, \text{'g- -'})$

Gráficos 2D

Subplot

Para ver varias gráficas en una misma ventana subdividiendo ésta en subventanas se utiliza $>>$ *subplot*(m, n, i)
La sintaxis indica que la ventana de gráficos activa se divide en m filas y n columnas. El tercer argumento indica la subdivisión activa (numerada de izquierda a derecha y de arriba abajo).
Veamos esto con un ejemplo:

$>>$ $x = -2 * pi : pi/25 : 2 * pi; y = sin(x); z = abs(x);$
$>>$ $t = log(x + 10); w = exp(x - 1);$
$>>$ $subplot(2, 2, 1); plot(x, y, \text{'r-'})$
$>>$ $subplot(2, 2, 2); plot(x, z, \text{'g-.'})$
$>>$ $subplot(2, 2, 3); plot(x, t, \text{'b o'})$
$>>$ $subplot(2, 2, 4); plot(x, w, \text{'k*'})$

Gráficos 2D

Gráficas de funciones

MATLAB dispone del comando *fplot* que evitan las operaciones sobre cada componente de un vector. Un ejemplo de utilización es

$>>$ *fplot*(*f*, [*Xmin Xmax*]) %dibuja f en el intervalo $[Xmin, Xmax]$

Por ejemplo

$>>$ *fplot*(@(*x*)*sin*(*x*), [0 4 $*$ *pi*],'g-.')

o

$>>$ *fun* = @(*x*)*sin*(*x*) %función anónima con parámetro x

$>>$ *fplot*(*fun*, [0 4 $*$ *pi*],'g-.')

Gráficos 2D

Gráficas de funciones simbólicas

La orden *ezplot* es una alternativa para generar de forma sencilla la gráfica de una función simbólica. Su sintaxis es

Por ejemplo

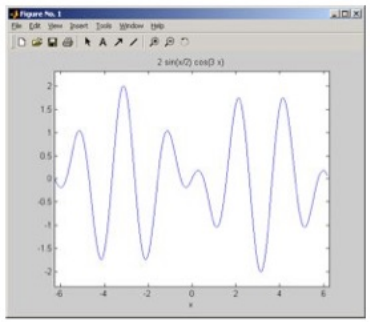
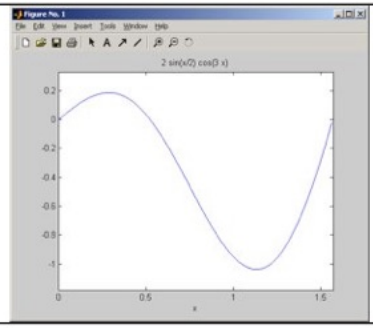
```
>> syms x;    %define la variable simbólica
>> f = x^2 - 1; ezplot(f)    %define la expresión simbólica
```

o

```
>> ezplot('x^2 - 1')    %dibuja directamente la función simbólica
```

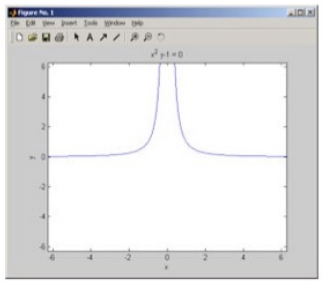
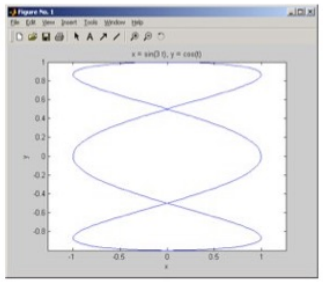
Gráficos 2D

Curvas planas: easy-to-use

`ezplot(f)`	donde • **f** es una cadena de caracteres conteniendo la expresión de una función **y=f(x)** dibuja la función **y=f(x)** para x en el intervalo **[-2π,2π]** **Ejemplo:** `>> ezplot('2*sin(x/2)*cos(3*x)')`	
`ezplot(f,[a,b])`	lo mismo que la anterior para **x** variando en el intervalo **[a,b]** **Ejemplo:** `>>ezplot('2*sin(x/2)*cos(3*x)',[0,pi/2])`	

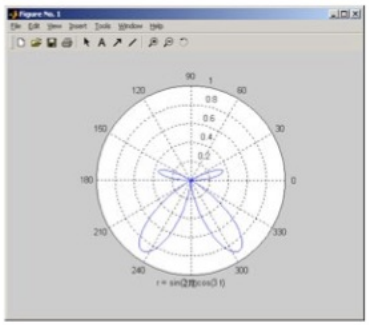
Gráficos 2D

Curvas planas: easy-to-use

`ezplot(f)` `ezplot(f,[a,b])`	si **f** es una expresión de **(x,y)**, dibuja la curva implícitamente definida por **f(x,y)=0**, para **x** e **y** variando en el intervalo **[-2π,2π]** en el primer caso y para **x** e **y** variando en el intervalo **[a,b]** en el segundo caso. **Ejemplo:** `>> ezplot('x^2*y-1')`	
`ezplot(x,y)` `ezplot(x,y,[a,b])`	donde • **x** e **y** son dos cadenas de caracteres conteniendo las expresiones de dos funciones **x(t)** e **y(t)** dibuja la curva de ecuaciones paramétricas **x=x(t) y=y(t)** para **t** en el intervalo **[0,2π]**, en el primer caso y para **t** en el intervalo **[a,b]** en el segundo **Ejemplo:** `>> ezplot('sin(3*t)','cos(t)')`	

Gráficos 2D

Curvas planas: easy-to-use

`ezpolar(f)` `ezpolar(f,[a,b])`	donde • **f** es una cadena de caracteres conteniendo la expresión de una función **f(θ)** dibuja la curva definida en coordenadas polares por **ρ= f(θ)** para θ variando en el intervalo **[0,2π]**, en el primer caso y en el intervalo **[a,b]** en el segundo **Ejemplo:** `>> ezpolar('sin(2*t)*cos(3*t)',[0,pi])`	

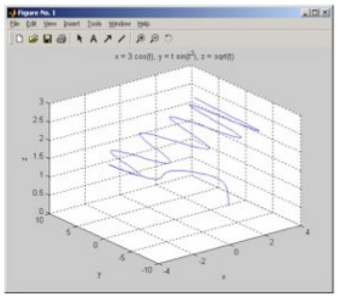
Gráficos 3D

Curvas en el espacio: easy-to-use

Para dibujar curvas en el espacio tridimensional de manera sencilla, MATLAB dispone del comando *ezplot3*

Gráficos 3D

Curvas en el espacio: easy-to-use

`ezplot3(x,y,z)` `ezplot3(x,y,z,[a,b])`	donde • **x** , **y** , **z** son tres cadenas de caracteres conteniendo las expresiones de tres funciones **x(t)** , **y(t)** , **z(t)** dibuja la curva de ecuaciones paramétricas **x=x(t) y=y(t) z=z(t)** para **t** en el intervalo **[0,2π]**, en el primer caso y para **t** en el intervalo **[a,b]** en el segundo **Ejemplo:** `>>ezplot3('3*cos(t)','t*sin(t^2)','sqrt(t)')`	

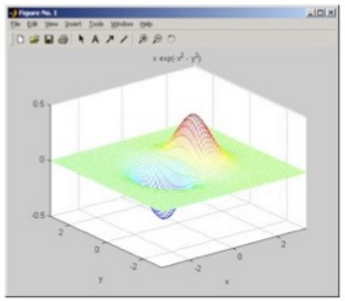
Gráficos 3D

Superficies en el espacio: easy-to-use

Para dibujar superficies en el espacio tridimensional de manera sencilla, MATLAB dispone del comando *ezmesh*

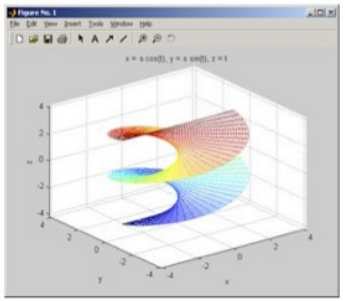
Gráficos 3D

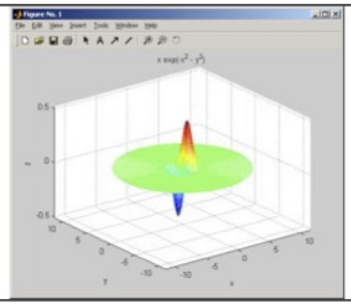
Superficies en el espacio: easy-to-use

`ezmesh(f)`
`ezmesh(f,[a,b])`
`ezmesh(f,[a,b,c,d])`
	donde
	• **f** es una expresión de dos variables
	dibuja la superficie **z=f(x,y)** para **(x,y)** variando en el cuadrado **[-π,π]x[-π,π]** en el primer caso, en el cuadrado **[a,b]x[a,b]** en el segundo, y en el rectángulo **[a,b]x[c,d]** en el tercer caso. El método de dibujo es una malla con segmentos coloreados, en función de los valores en los extremos.
	
	Ejemplo:
	
	`>> ezmesh('x*exp(-x^2 - y^2)')`
		

Gráficos 3D

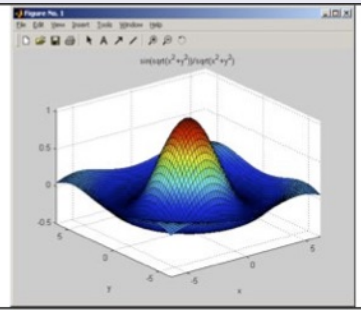
Superficies en el espacio: easy-to-use

`ezmesh(x,y,z)`
`ezmesh(x,y,z,[a,b])`
`ezmesh(x,y,z,[a,b,c,d])`
	donde
	• **x**, **y**, **z** son expresiones de funciones de dos variables
	dibuja la superficie de coordenadas paramétricas **x=x(s,t) y=y(s,t) z=z(s,t)** para **(s,t)** variando en el cuadrado **[-2π,2π]x[-2π,2π]** en el primer caso, en el cuadrado **[a,b]x[a,b]** en el segundo, y en el rectángulo **[a,b]x[c,d]** en el tercer caso
	Ejemplo:
	`>> ezmesh('s*cos(t)','s*sin(t)','t',[-pi,pi])`
		

`ezmesh(... , 'circ')`	en cualquiera de los usos anteriores, dibuja la función correspondiente sobre un círculo centrado en el origen **Ejemplo:** `>> ezmesh('x*exp(-x^2 - y^2)','circ')`	

Gráficos 3D

Superficies en el espacio: easy-to-use

`ezsurf(f)`	dibuja una superficie coloreada **z=f(x,y)**. Sus argumentos son como en **ezmesh** **Ejemplo:** `>> ezsurf('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)')`	

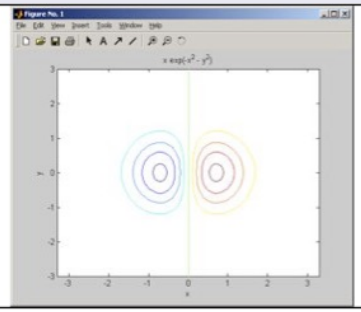
Gráficos 3D

Curvas de nivel: easy-to-use

Para dibujar superficies de nivel, MATLAB dispone del comando *ezcontour*

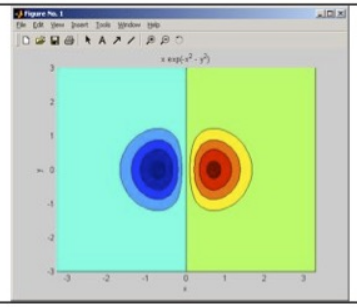
Gráficos 3D

Curvas de nivel: easy-to-use

`ezcontour(f)`
`ezcontour(f,[a,b])`
`ezcontour(f,[a,b,c,d])`
	dibuja las líneas de nivel (isovalores) de la función z=f(x,y)
	
	Ejemplo:
	
	`>> ezcontour('x*exp(-x^2 - y^2)')`
		

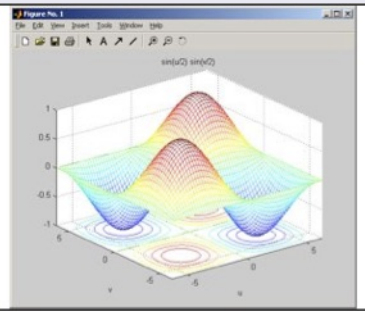
Gráficos 3D

Curvas de nivel: easy-to-use

`ezcontourf(. . . )`	hace lo mismo que **ezcontour**, pero rellenando con un color sólido las distintas zonas determinadas por las líneas de nivel **Ejemplo:** `>> ezcontourf('x*exp(-x^2 - y^2)')`	

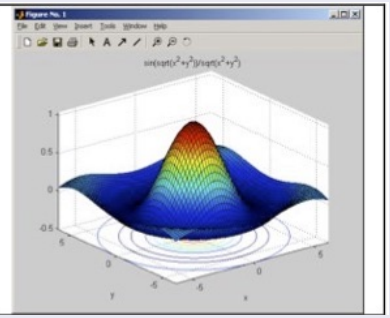
Gráficos 3D

Superficies en el espacio: easy-to-use

`ezmeshc(f)`	con los mismos argumentos que **ezmesh**, dibuja simultáneamente las líneas de nivel y la superficie **Ejemplo:** `>> ezmeshc('sin(u/2)*sin(v/2)')`	

Gráficos 3D

Superficies y curvas de nivel: easy-to-use

`ezsurfc(f)`	**Ejemplo:** `>>ezsurfc('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)')`	

Gráficos 3D

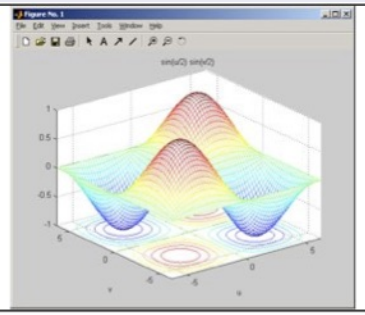
Gráfico easy-to-use con funciones "handle"

En los comandos vistos en las transparencias anteriores aparecen los argumentos f, x, y o z que representan las funciones a dibujar. En todos los ejemplos se ha escrito directamente la expresión en la llamada a la función *ez** correspondiente, como una cadena de caracteres.
Sin embargo, también se pueden poner dichas funciones como

Gráficos 3D

Gráfico easy-to-use con funciones "handle"

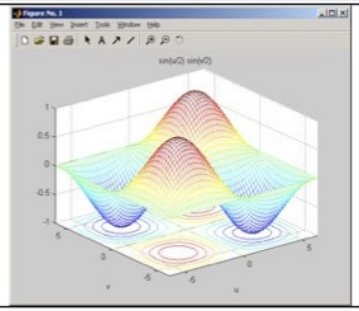
Una función "handle"

`ezmeshc(f)`	en vez de
	`>> ezmeshc('sin(u/2)*sin(v/2)')`
	se podría haber escrito:
	
	`>> fun=@(u,v) sin(u/2)*sin(v/2)`
	`>> ezmeshc(fun)`
		

Gráficos 3D

Gráfico easy-to-use con funciones "handle." [en] ficheros M

O mediante un "handle" de una M-función escrita en un fichero

`ezmeshc(f)`	`>> ezmeshc(@fun)`
	
	siendo **fun** la M-función, almacenada en el fichero **fun.m**, siguiente:
	
	`function [z]=fun(u,v)`
	`z= sin(u/2)*sin(v/2);`
		

El paquete Simbólico

Sustitución

Dado una variable simbólica

$>>$ *syms* x

y una expresión simbólica

$>> f = x^2 - x + 1$

para obtener el valor de la función en un punto se utiliza el comando *subs*

$>>$ *subs*$(f, -3)$ % valor de la función simbólica f cuando la variable es -3

ans

11

El paquete Simbólico

Haciendo derivadas

- *Declaración de las variables:*
  $>>$ *syms x y* %ahora x e y son **objetos especiales**
- *Definición de la función:*
  $>> f(x,y) = x\hat{}2 + y\hat{}3$ %$f(x,y)$ es un **objeto especial**, una función.
  Los objetos especiales también se almacenan en **workspace**
- *Derivar:*
  $>>$ *diff*(f,x) %comando para derivar
  $ans(x,y) =$
  $2 * x$

El paquete simbólico

Ejemplos

- *Primera derivada:*
  $>> diff(f, y)$
  $ans(x, y) =$
  $3 * y^2$
- Segunda derivada: $>> diff(diff(f, x))$ o $diff(f, 2)$
  $ans(x, y) =$
  2

El paquete simbólico

Ejemplo

a) Crea dos variables simbólicas x y n

b) Crea la expresión simbólica, $p(x) = 2x^n + 5x^{n-1}$

c) Calcula $diff(p), diff(p, 2), diff(p, 3), \ldots$. Analiza el nuevo grado n

d) Con $>> n = 5$; se produce algún cambio en la derivada de $p(x)$?

e) Ejecuta $>> q = symfun(2 * x\hat{}n + 5 * x\hat{}(n-1), [x])$ y calcula $diff(q, 2)$. Depende el resultado de $n = 5$?

El paquete Simbólico

Resolución de ecuaciones

Otra funcionalidad del paquete simbólico es la **resolución de ecuaciones**. Por ejemplo, encontrar las raíces del siguiente polinomio:

$$p(x) = x^3 - x^2 - 14x + 24$$

De forma que, una vez definida x y $p(x)$ llamamos a la función

$$>> solve(p)$$

Y el resultado es $2, 3, -4$ que son las raíces de $p(x)$. Puedes ver esto dibujando la función $p(x)$ en el intervalo $[-5, 5]$.

El paquete Simbólico

Resolución de ecuaciones

Realmente *solve* devuelve un conjunto de valores $\{-4, 2, 3\}$ en el ejemplo o un conjunto vacío si no encuentra solución. Otras veces se obtiene como resultado un conjunto infinito de elementos.
Por ejemplo:

$$>> solve(x\hat{}2 > 2)$$

produce
$Dom :: Interval(2^{(1/2)}, Inf)$
$Dom :: Interval(-Inf, -2^{(1/2)})$

Ejercicios

Ejercicio #1

Calcula

1. $7^2 + 9 - \frac{26}{2}$
2. $\frac{4^3 - 17 + 13}{2 \cdot 3^2}$
3. $2.87 * 9/3.6 * (5 + 4)$
4. $\sqrt{35}$
5. $\sin(\pi)$

Ejercicios

Ejercicio #2

Representa en distintos formatos los siguientes números

1. $\frac{2}{6}$
2. $\frac{20}{6}$
3. $\frac{2000}{6}$
4. $\frac{20000000000}{6}$
5. 0.333

Ejercicios

Ejercicio #3

Utiliza la ayuda de MATLAB para obtener información sobre roots

Ejercicios

Ejercicio #4

Crea dos vectores incrementales (a, b) desde 0 hasta 10 con un incremento de 0.01 y 0.1 respectivamente. Dibuja la gráfica del coseno de los vectores en una misma ventana

Ejercicios

Ejercicio #5

Calcula $a^2 - b^2$ **con** $a = 1.4 \times 10^{154}$ **y** $b = 1.3 \times 10^{154}$

1. ¿Qué ocurre con el resultado?
2. ¿Cómo se puede solucionar el problema?

Ejercicios

Ejercicio #6

Ejecuta las siguientes órdenes y observa las respuestas

1. $>> x = 10/7$
2. $>> y = sym('10/7')$
3. $>> double(y)$
4. $>> y + 5$
5. $>> double(y) + 5$

Ejercicios

Ejercicio #7

Halla la diagonal de un rectángulo cuyos lados miden 12 y 5

Ejercicios

Ejercicio #8

Dado un triángulo con ángulo $\pi/6$ entre dos lados de longitud 5 y 7, utiliza el teorema del coseno para encontrar el tercer lado c

Ejercicios

Ejercicio #9

Obtén las expresiones indicadas

1. El desarrollo de $(5 * x - y)^2$
2. La simplificación de $9 * (x - y)^2 + 12 * (x - y) * (x + y) + 4 * (x + y)^2$
3. La factorización de $9 * (x - y)^2 + 12 * (x - y) * (x + y) + 4 * (x + y)^2$

Ejercicios

Ejercicio #10

Se desea medir la altura que alcanza un cultivo, sabiendo que dicha altura está en función del tiempo. Las mediciones se hacen una vez al día y los resultados obtenidos se resumen en la siguiente tabla

Tiempo (días)	1	2	3	4	5
Altura (cm)	5.2	6.6	7.3	8.6	10.7

1. Obtén la gráfica que explique ese crecimiento
2. Debe incluir un título apropiado
3. Debe incluir etiquetas en los ejes que indiquen cual es la variable correspondiente
4. Varía el trazo y el color

Ejercicios

Ejercicio #11

Dibuja en el intervalo $[-2, 2]$ las curvas $y = x^3$ e $y = x^2 + 1$ conjuntamente, etiquetándolas con un título que indique cuál es cada una de ellas. Hazlo de dos formas distintas

1. Superponiendo las gráficas utilizando los mismos ejes para ambas gráficas
2. En dos subventanas, cada gráfica con sus propios ejes coordenados

Ejercicios

Ejercicio #12

Dibuja

1. La función coseno en el intervalo $[-3\pi, \pi]$ con el comando *ezplot*
2. Dibuja la función seno en ese mismo intervalo, conjuntamente con la función coseno

Ejercicios

Ejercicio #13

Dada la matriz M = [1:3; 4:6], crea una matriz de ceros con el mismo tamaño que M

Ejercicios

Ejercicio #14

Crea 2 vectores de la misma longitud y realiza las siguientes operaciones:

1. Súmalos
2. Réstalos
3. Elévalos al cuadrado
4. Eleva al cuadrado elemento a elemento
5. Multiplícalos elemento a elemento
6. Divídelos elemento a elemento

Ejercicios

Ejercicio #15

Define la función $(x^2 + y^2–1)^3 - x^2 * y^3$ en el intervalo $[-1{,}5\ 1{,}5]$ para las x y $[-1\ 1{,}5]$ para las y. Representala gráficamente

Ejercicios

Ejercicio #16

Define la función $x^3 + y^3 - 6xy$ y representa gráficamente (con subventanas) la función y sus curvas de nivel

Ejercicios

Ejercicio #17

Crea una matriz de dimensión 6x6 cuya diagonal sea un vector de números aleatorios entre 2 y 10

Ejercicios

Ejercicio #18

Utilizando el comando plot y las funciones seno y coseno dibuja la siguiente figura: