Matlab/Octave: Bestiario de comandos y sentencias

A lo largo de este documento, x e y serán vectores fila o columna, z un número complejo, A y B matrices.

Espacio de trabajo: básicos

`Ctrl + C`	Aborta la operación o sentencia actual en la línea de comandos
`clc`	Limpia la ventana de comandos
`clear`	Borra todas las variables
`clear x A`	Borra las variables x y A
`diary 'fichero.txt'`	Registra en un fichero lo que se hace en la ventana de comandos
`diary off`	Para el registro
`diary on`	Reanuda el registro
`save fichero`	Guarda las variables definidas en el fichero
`save fichero x A`	Guarda las variables x y A en el fichero
`load fichero`	Carga las variables almacenadas en el fichero
`pwd`	Muestra la ubicación del directorio de trabajo actual
`dir`	Muestra el contenido del directorio de trabajo actual
`cd carpeta`	Permite acceder a una carpeta del directorio de trabajo actual
`help comando`	Abre la documentación del comando
`lookfor 'texto'`	Busca el texto en la documentación de comandos
`...`	Conecta una misma sentencia escrita en dos lineas seguidas de código
`comando;`	La "`;`" suprime la salida del comando
`nombreprograma`	Ejecuta *nombreprograma.m*
`tic, sentencias, toc`	Devuelve el tiempo de ejecución total de las sentencias

Espacio de trabajo: formato y salida

`format short`	Muestra n^{os} con 4 decimales
`format short e`	Muestra n^{os} con 4 decimales en notación exponencial
`format long`	Muestra n^{os} con 15 decimales
`format long e`	Muestra n^{os} con 15 decimales en notación exponencial
`format rat`	Muestra n^{os} en formato racional
`% Esto es un comentario`	Comentarios
`disp('texto')`	Muestra el texto
`disp(x)`	Muestra el contenido de la variable x

Constantes numéricas

`pi`	$\pi \simeq 3.1415926535897$
`i` ó `j`	Unidad imaginaria $\sqrt{-1}$
`Inf`	Infinito
`NaN`	"No es un número" (p.ej., $0/0$)
`eps`	Precisión relativa de máquina en doble precisión (por defecto, $2.2204 \cdot 10^{-16}$)
`realmax`	Nº positivo más grande en doble precisión, $1.7977 \cdot 10^{308}$
`realmin`	Nº positivo más pequeño en doble precisión $2.2251 \cdot 10^{-308}$

Operaciones aritméticas y funciones básicas

`1.349`	Los decimales de un real se definen CON EL PUNTO "`.`" NO con comas o tildes
`3+4`, `7*4`, `2-6`	Suma, producto y resta
`8/3`, `3\8`	División por la derecha y por la izquierda
`3^7`	Calcula la potencia 3^7
`rem(17,3)`	Resto de la división de 17 entre 3
`sqrt(5)`	Calcula la raíz cuadrada $\sqrt{5}$
`log(3)`	Calcula el logaritmo neperiano $\ln(3)$
`log10(100)`	Calcula el logaritmo $\log_{10}(100)$
`abs(-5)`	Calcula el valor absoluto $\lvert -5 \rvert$
`sin(5*pi/3)`	Calcula el seno $\sin(5\pi/3)$
`cos(-pi/3)`	Calcula el coseno $\cos(-\pi/3)$
`exp(3)`	Calcula la exponencial e^3

Números complejos

`z=1-2*i` ó `z=complex(1,-2)`	Crea el número complejo $z = 1 - 2i$ (a partir de las partes real e imaginaria)
`abs(z)`	Módulo (valor absoluto) de z
`angle(z)`	Argumento de z
`z'`	Conjugado de z
`real(z)`	Parte real de z
`imag(z)`	Parte imaginaria de z
`isreal(z)`	Devuelve 1 si z es real, 0 si no

Definiendo variables básicas

`a = 3`	Define la variable a como 3
`b = 4.321`	Define la variable b como 4.321
`c = 'texto'`	Define en c una cadena de caracteres con el texto
`cond = logical(1)`	Define en $cond$ el valor lógico 1 (*true*)

Valores lógicos y operaciones lógicas

```
a = 10;     % Asignamos a "a" el valor de 10
a == 5      % Comprobamos si a es igual a 5
    false
~(a == 5)   % Comprobamos la negación de lo anterior
    true
a == 10     % Comprobamos si a es igual a 10
    true
a >= 5      % Comprobamos si a es mayor o igual a 5
    true
a < 11      % Comprobamos si a es menor que 11
    true
a ~= 4      % Comprobamos si a es no igual a  4
    true
a > 1 && a ~= 10 % Comprobamos si a es mayor que 1 Y
    false        % no igual a 10
a > 1 || a ~= 10 % Comprobamos si a es mayor que 1 Ó
    true         % no igual a 10
xor(a == 10, a < 100) % Si a es 10 Ó (exclusivo)
    false             % menor que 100
```

Vectores y matrices: generación y acceso

`x = [1, 2, 3]` ó `x = [1 2 3]`	Define x como el vector fila $[1, 2, 3]$
`x = [1; 2; 3]` ó `x = [1, 2, 3]'`	Define x como el vector columna $[1, 2, 3]^t$
`7:15`	Vector fila con $7, 8, \ldots, 14, 15$
`1.1:0.2:3.3`	Vector fila con $1.1, 1.3, \ldots, 3.3$
`linspace(2, 6.5, 100)`	Genera un vector fila con 100 componentes equiespaciadas entre el 2 y el 6.3
`A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]`	Define A como una matriz 3×4
`x(2:12)`	Del 2º al 12º elemento de x
`x(2:end)`	Del 2º al último elemento de x
`x(1:2:end)`	El 1er, 3er, 5º,...hasta el último elemento de x.
`A(3,4)`	El elemento de la 3^{a} fila y la 4^{a} columna de A
`A(3,:)`	La 3^{a} fila de A
`A(:,4)`	La 4^{a} columna de A
`A(2, 1:5)`	Del 1º al 5º elemento de la 2^{a} fila
`A([1,3],4)`	Los elementos de la 1^{a} y 3^{a} fila que se encuentran en la 4^{a} columna
`A(:)`	La matriz A vista como vector columna (con los elementos en orden columna)

Autor: Juan Viu Sos • ✉ • 🏠 • git • Dpto. de Matemáticas e Informática aplicadas a las Ing. Civil y Naval – UPM

Actualizado: 28 de noviembre de 2022

Matlab/Octave: Bestiario de comandos y sentencias

Vectores y matrices: composición y borrado

`[A ; B]`	Matriz compuesta por las filas de A sobre las filas de B (con mismo nº de columnas)
`[A , B]`	Matriz compuesta por las columnas de A seguidas de las de B (con mismo nº de filas)
`[3,v; c d]`	Matriz compuesta por 3 seguido de la fila de v, sobre las columnas de c seguidas de las de d
`A = []`	Borra todos los elementos de A
`x(4) = []`	Elimina la 4ª componente de x
`A(3,:) = []`	Elimina la 3ª fila A

Operaciones de vectores y matrices

`3 * x`	Multiplica cada elemento de x por 3
`x + 2`	Suma 2 a cada elemento de x
`x + y`	Suma elemento a elemento los vectores x e y
`A * y`	Producto de una matriz y un vector
`A * B`	Producto (matricial) de dos matrices $A \cdot B$
`A .* B`	Producto (elemento a elemento) de dos matrices
`A ^ 3`	La matriz (cuadrada) A elevada a la 3ª potencia
`A .^ 3`	La matriz con los elementos de A elevados al cubo
`A'`	Traspuesta de A
`inv(A)`	Inversa de A
`A / 3`	Divide cada elemento de A por 3
`3 ./ A`	Devuelve la matriz donde cada elemento es 3 dividido por el correspondiente de A
`A / B`	Devuelve $A \cdot B^{-1}$
`A \ B`	Devuelve $A^{-1} \cdot B$
`A ./ B`	División (elemento a elemento) de dos matrices

Funciones auxiliares de vectores y matrices (I)

`length(x)`	Nº de componentes de x
`[m,n]=size(A)`	Tamaño de A. Asigna a `m` el nº de filas y a `n` el nº de columnas de A
`size(A,1)`	Nº de filas de A
`size(A,2)`	Nº de columnas de A
`sum(x)`	Suma todos los elementos de x
`sum(A)`	Vector de sumas de cada columna de A
`prod(x)`	Multiplica todos los elementos de x
`prod(A)`	Vector de productos de cada columna de A
`sort(x)`	Ordena ascendentemente los elementos de x
`sort(A)`	Ordena ascendentemente de forma independiente cada columna de A
`max(x)`	Valor máximo de x
`max(A)`	Vector con el máximo de cada columna de A
`min(x)`	Valor mínimo de x
`min(A)`	Vector con el mínimo de cada columna de A

estructura básica de un programa

```
% CALCULO DEL AREA DE UN CIRCULO

% Entrada de datos
r = input('Introduce el radio del ciculo: ');

% Algoritmo
A = pi*r^2;

% Salida de datos
fprintf('El area del circulo es %.3f \n', A);
```

Entrada y salida de datos

`a=input('Introduce dato: ')`	Saca en pantalla el texto de entrada de dato y se lo asigna a a al presionar **Enter**
`c=input('¿Nombre?','s')`	Asigna la cadena de caracteres introducida por usuario
`disp(A)` ó `disp('texto')`	Muestra A o texto
`fprintf('Es a=%f \n', a)`	Escribe en pantalla el texto combinado con el dato de a
`c=sprintf('Es a=%f \n', a)`	Almacena en c la cadena de caracteres del texto combinado con el dato de a

Formato de salida de datos

`%f`	Formato en coma flotante (escribe con 6 decimales)
`%d`	Formato como enteros, lógicos,...
`%s`	Formato de cadenas de caracteres
`%-6.3f`	Salida de datos en coma flotante, justificado a izqda. (con `-`), con 6 caracteres mínimos reservados para escritura, escrito con 3 decimales
`%4.2d`	Salida de datos de enteros etc, justificado a dcha. (sin `-`), con 4 caracteres mínimos reservados para escritura, con un mínimo de 2 dígitos
`%7s`	Salida de cadenas de caracteres, justificado a dcha. (sin `-`) con mínimo de anchura de 7 caracteres

Redondeo de números

`fix(3.2)`	Elimina la parte decimal de 3.2 y devuelve el entero 3
`floor(3.2)`	Mayor entero por debajo de 3.2, es decir 3
`floor(-3.2)`	Mayor entero por debajo de -3.2, es decir -4
`ceil(3.2)`	Menor entero por encima de 3.2, es decir 4
`ceil(-3.2)`	Menor entero por encima de -3.2, es decir -3
`round(3.2)`	Entero más cercano a 3.2, es decir 3
`round(3.7)`	Entero más cercano a 3.7, es decir 4

Generación de elementos aleatorios

`rand()`	Genera nº aleatorio equiprobable en el intervalo ABIERTO $(0, 1)$
`rand(3)`	Genera una matriz 3×3 de nºs aleatorios equiprobables en $(0, 1)$
`rand(4,2)`	Genera una matriz 4×2 de reales aleatorios equiprobables en $(0, 1)$
`2+13*rand()`	Genera nº aleatorio equiprobable en el intervalo ABIERTO $(2, 15)$
`randi([2,14])`	Genera nº ENTERO aleatorio equiprobable en el intervalo CERRADO $[2, 14]$

Sentencias condicionales

```
n=input('Introduce un entero: ');
if n<0                                  % caso n negativo
    disp('el número es negativo')
elseif  n==0                            % caso n cero
    disp('el número es cero')
else                                    % caso restante
    disp('el número es positivo')
end
```

Bucle while (I)

```
% Bucle while como un for
i = 0;
while i < 7
    disp(i);
    i = i + 1;
end
```

Bucle while (II)

```
% Generamos los cubos de num naturales <100
i=1; c=1;
while c<100
    disp(c);
    i=i+1; c=i^3;
end
```

Bucle while para condición input

```
% Debes introducir un numero entero no vacio...
n=input('Introduce un número entero: ');
% Mientras que el dato sea vacio o no sea entero...
while isempty(n) || fix(n)~=n
    n=input('Debes introducir un número entero: ');
end
fprintf('Has introducido el %d \n ',n)
```

Matlab/Octave: Bestiario de comandos

Bucle for (I)

```
for i = 1:10
    disp(i);
end
```

Bucle for (III)

```
% Bucle con paso no trivial
for i=1.5:0.1:2
    disp(i)
end
```

Bucle for (III)

```
% Bucle sobre un vector predefinido
for i=[4,1,1,-2,0.4]
  disp(i)
end
```

Bucle for anidado I

```
% Bucle anidado independiente
for i=1:2
    for j=1:4
        A(i,j)=1/(i+j-1);
    end
end
```

Bucle for anidado II

```
% Bucle anidado dependiente
for i=1:4
    for j=1:i
      suma = i+j;
      fprintf('a i=%d,\t sumo j=%d: da %d\n',i,j,suma)
    end
end
```

Interrupciones de bucles y programas/funciones

break	Interrumpe el menor bucle que lo contiene y continua con el programa
continue	Pasa automáticamente a la siguiente iteración del menor bucle que lo contiene
return	Termina automáticamente el progreso del programa o función

Algoritmo del intercambio

```
a=input('Introduce un dato...');
b=input('Introduce otro dato...');

disp('a es: '), disp(a);
disp('y b es: '), disp(b);

aux=a;      % var. auxiliar con el dato de "a"
a=b;        % el dato de "b" a "a"
b=aux;      % el dato original de "a" (en "aux") a "b"

disp('Ahora, a es: '), disp(a);
disp('y b es: '), disp(b);
```

Algoritmo de la suma

```
v=input('Introduce un vector...');

%Variable para la suma parcial, "S"
S=0;   %inicializacion (0 es neutro para la suma)
for i=1:length(v)
    S=S+v(i);   %suma parcial con elem. del vector
end

fprintf('La suma es %f \n', S)
```

Algoritmo del producto

```
v=input('Introduce un vector...');

%Variable para el producto parcial, "P"
P=1;   %inicializacion (1 es neutro para el producto)
for i=1:length(v)
    P=P*v(i);   %producto parcial con elem. del vector
end

fprintf('El producto es %f \n', P)
```

Algoritmo del máximo

```
v=input('Introduce un vector...');

%Variable para el CANDIDATO a maximo, "M"
M=v(1);              %inicializacion (1er elem.)
for i=2:length(v)    % seguimos desde 2a componente
    if v(i)>M        % Si supera al candidato....
        M=v(i);      % ...actualizamos el candidato
    end
end

fprintf('El maximo es %f \n', M)
```

Funciones

```
% suma
% IMPORTANTE: en el fichero suma.m
function w = suma(x, y)
    w = x + y;
end %opcional

>> suma(10, -5)
      5
```

Funciones sin argumentos

```
% FICHERO: escribodato.m
function escribodato(n)
  fprintf('El valor del dato es %f\n',n);

% FICHERO: errores.m
function errores()
  fprintf('DATOS ERRONEOS: fin del programa\n');

>> escribodato(12.5)
      El valor del dato es 2.500000

>> errores()
      DATOS ERRONEOS: fin del programa
```

Funciones anónimas (I)

```
>> fun1= @(x) sin(x)*cos(x)+1;

>> fun1(pi/4)
    1.5000

>> f = @(x) sin(x.^2)./(5*x);   %elmto a elemto

>> f(pi/2)
    0.0795
>> f([-pi/2, 0, pi/2])
    -0.0795 NaN 0.0795

>> g= @(x,y) x*cos(y^2);        %varias variables

>> g(1,0)
    1
```

Funciones anónimas (II)

```
>> h=inline('x*y*z-x^3+y^2*z');

>> h(1,1,0)
    -1
```

Matlab/Octave: Bestiario de comandos

Funciones auxiliares de vectores y matrices (II)

`norm(x)`	Norma $\|x\|$
`dot(x,y)`	Producto escalar $x \cdot y$
`cross(x,y)`	Producto vectorial $x \times y$
`det(A)`	Determinante de A
`trace(A)`	Traza de A
`eig(A)`	Vector de autovalores de A

Generando matrices y vectores

`zeros(5)`	Crea una matriz 5×5 de 0's
`zeros(12, 5)`	Crea una matriz 12×5 de 0's
`ones(5)`	Crea una matriz 5×5 de 1's
`ones(12, 5)`	Crea una matriz 12×5 de 1's
`eye(5)`	Crea una matriz identidad de 5×5
`eye(12, 5)`	Crea una matriz 12×5 con 1's en la diagonal
`repmat(A,3,2)`	Crea una matriz por bloques compuesta por 3 filas y 2 columnas de matrices A
`diag(x)`	Crea una matriz con x en la diagonal y 0's en el resto
`diag(A)`	Devuelve un vector fila conteniendo la diagonal de A
`diag(diag(A))`	Crea una matriz manteniendo la diagonal de A y con 0's en el resto
`blkdiag(A,B)`	Crea una matriz por bloques compuesta por A y B como bloques diagonales
`triu(A)`	Matriz triangular superior de A
`fliplr(A)`	Matriz formada por intercambiar las columnas de A con respecto al eje vertical medio
`fliplr(A)`	Matriz formada por intercambiar las filas de A con respecto al eje horizontal medio
`reshape(A,[5,2])`	Matriz 5×2 formada por los elementos de A manteniendo el orden

Operadores e índices lógicos

```
>> x = 1:9
    1  2  3  4  5  6  7  8  9
>> (x>2)&(x<6)          %condicion logica sobre vector
    0  0  1  1  1  0  0  0  0
>> x(x>7)               %acceso usando indices logicos
    7 8 9
>> x(x>7) = 100         %asignacion con indices logicos
    1  2  3  4  5  6  100  100  100
>> v=[1 2 3]; w=[-1 0 4];
>> v>w                  %tambien elemento a elemento
    1  1  0
```

Funciones para búsqueda de condiciones lógicas

```
>> b=[-1,0,1,2];
>> any(b>0)       %alguno es >0 ?
    1
>> all(b>0)       %todos son >0 ?
    0
>> find(b>0)      %donde es >0 ?
    3
    4
%Para matrices, "any" y "all" funcionan por COLUMNAS
>> A = [-2:2; linspace(0,1,5)]
   -2.0000  -1.0000         0   1.0000   2.0000
         0   0.2500   0.5000   0.7500   1.0000
>> A>0
    0  0  0  1  1
    0  1  1  1  1

>> any(A>0)       %alguno en la columna es >0 ?
    0  1  1  1  1

>> all(A>0)       %todos en la columna son >0 ?
    0  0  0  1  1

>> find(A>0)      %dónde en A(:) (vect. colum.) son >0 ?
    4
    6
    7
    8
    9
   10
```

Gráficos 2D

`plot([2,4],[-1,3])`	Dibuja el segmento entre puntos $(2,-1)$ y $(4,3)$
`plot( [1 6 5 2 1], [2 0 4 3 2] )`	Dibuja el polígono de puntos (1^{er} y último punto iguales: p.ej. $(1,2)$)
`plot(x,y)`	Dibuja el gráfico de puntos con ordenadas y con respecto a las abscisas x (es decir, los puntos $(x(i), y(i))$)
`axis equal`	Fuerza a tener la misma escala en el eje x y en el y
`axis([xmin, xmax, ymin, ymax])`	Fija los límites del gráfico en valores particulares de los ejes
`title('Un Título')`	Añade un título al gráfico
`xlabel('etiqueta x')`	Añade una etiqueta al eje x
`ylabel('etiqueta y')`	Añade una etiqueta al eje y
`legend('esta','este')`	Etiqueta dos curvas en el gráfico
`grid`	Añade una cuadrícula al gráfico
`hold on,..., hold off`	Superpone gráficos
`figure`	Comienza un nuevo gráfico
`clf`	Limpia la ventana de gráficos

Modificadores de formato del gráfico

`plot(x,y,'--*r')`	Gráfico de con linea de trazos discontinuos (`'--'`), puntos estrellados (`'*'`), en color rojo (`'r'`)
`. * x o + -`	Tipos de puntos para rellenar el gráfico
`- -- -. :`	Tipos de linea uniendo puntos
`y g m b w r k`	Colores: amarillo, verde, magenta, azul, blanco, rojo, negro
`'Linewidth',0.5`	Cambia el grosor de línea a la mitad
`'Markersize',2`	Cambia el grosor del punto al doble

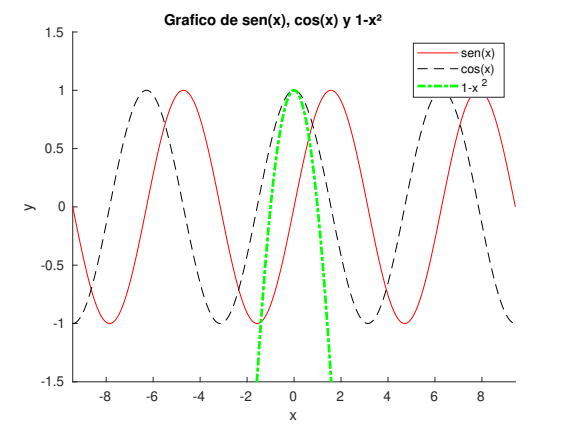
Gráficos 2D

```
%Dominio: [-3pi,3pi] (subdiv. en 1000 puntos)
x = linspace(-3*pi, 3*pi, 1000);
y1 = sin(x); y2 = cos(x);

f=@(x) 1-x.^2; %podemos usar funciones anonimas
y3 = f(x);

hold on             %Para añadir varias curvas
plot(x, y1, 'r-');  %sen(x) linea continua roja
plot(x, y2, 'k--'); %cos(x) linea discontinua negra
plot(x, y3, 'g-.', 'Linewidth', 2); %1-x^2 linea verde
                                    %ptos&trazos doble
hold off

% Fijamos los limites de los ejes
axis([-3*pi, 3*pi, -1.5, 1.5])
% Etiquetas de ejes
xlabel('x'); ylabel('y');
% Titulo
title('Grafico de sen(x), cos(x) y 1-x^2');
% Leyenda detallando las curvas
legend('sen(x)', 'cos(x)', '1-x^2');
```



Matlab/Octave: Bestiario de comandos

Operaciones de ficheros

`fid=fopen('datos.txt','r')`	Asocia la var. **fid** al fichero **'datos.txt'** y permiso **'r'**. Devuelve **-1** si hay un error
`close(fid);`	Cierra el fichero asignado al nombre interno **fid** y lo desvincula de Matlab/Octave. Devuelve **-1** si hay un error
`frewind(fid)`	"Rebobina" el fichero: cursor de lectoescritura al inicio
`feof(fid)`	Vale 1 si estamos en el final del fichero asociado a **fid**
`fprintf(fid,'a es%.2f', a)`	Escribe en el fichero de nombre interno **fid** el texto combinado con el dato de a
`c=fgetl(fid)`	Lee línea (sin incluir '**\n**') del fichero **fid** y la almacena en **c** como cadena. **-1** si llega a final de fichero
`c=fgets(fid)`	Lee línea (incluyendo '**\n**') del fichero **fid** y la almacena en **c** como cadena. **-1** si llega a final de fichero
`c=fgets(fid,5)`	Similar, pero lee 5 caracteres siguientes a la última posición, salvo final de linea. **-1** si llega a final de fichero
`[A,cont]=fscanf(fid,'%f')`	Lee todos los datos con formato **'%f'** en **fid** hasta que encuentra algo que no lo es o EOF. Los almacena en **A** (en orden columna), y en **cont** el nº de datos leídos
`A=fscanf(fid,'%f',2)`	Igual, pero **A** columna 1×2. Para de leer cuando rellena **A**
`A=fscanf(fid,'%f',[3,2])`	Igual, pero en **A** siendo 3×2, añade **0**'s hasta completar
`A=fscanf(fid,'%f',[2,inf])`	Igual, pero en **A** con 2 filas

Tipos de permisos para apertura de ficheros

`'r'`	Abre un fichero **existente** para **solo lectura**
`'r+'`	Abre un fichero **existente** para **lectura y escritura**
`'w'`	Crea un fichero **nuevo** (borrándolo si ya existe) para **solo escritura**
`'w+'`	Crea un fichero **nuevo** (borrándolo si ya existe) para **lectura y escritura**
`'a'`	Abre un fichero para escritura. Si ya existe, **comienza a escribir al final de lo ya escrito**

Comprobación de apertura de fichero

```
% Si el fichero no se encuentra en la carpeta
% de trabajo, se ha de dar la RUTA COMPLETA
% de su ubicacion en la maquina
fid=fopen('C:\ejercicios\puntos.txt','r');

if fid==-1  % En caso de error, fid toma el valor -1
    disp('ERROR: El fichero no se encuentra');
    return;
end
```

Escritura de resultados en fichero (I)

```
% Ejemplo secillo de escritura de una frase con dato
% en fichero 'resultados.txt' usando fprintf.
% Los datos se mostrarán con 3 decimales cada uno.

%Entrada: dato numérico
x = input('Dame un dato: ');

%Apertura (nuevo, escritura)
fid = fopen('resultado.txt','w');

%Escritura del resultado
y=sqrt(x+3);
fprintf(fid,'Con dato %.3f, resultado: %.3f.\n',x,y);

%Cierre
fclose(fid);
```

resultado.txt

```
Con dato 10.000, resultado: 3.606.
```

Escritura de resultados en fichero (II)

```
% Datos
x=0:.1:1;
y=[x;exp(x)];

% Apertura (nuevo + escritura)
fid=fopen('resultados.txt','w');

% Escritura
fprintf(fid,'%4s %12s \n', 'x','exp(x)'); %Titulo
fprintf(fid,'%6.2f %12.8f \n', y);        %Datos

% Cierre
fclose(fid);
```

resultados.txt

```
     x       exp(x)
  0.00   1.00000000
  0.10   1.10517092
  0.20   1.22140276
  0.30   1.34985881
  0.40   1.49182470
  0.50   1.64872127
  0.60   1.82211880
  0.70   2.01375271
  0.80   2.22554093
  0.90   2.45960311
  1.00   2.71828183
```

Lectura de líneas con `fgetl` hasta final de fichero

```
% Apertura del fichero (existente + lectura)
fid=fopen('prueba.txt','r');

n=0;
linea=fgetl(fid)        %1a lectura de linea
while linea~=-1         %mientras no llegamos al final
    linea=fgetl(fid);   %lectura de linea
    n = n+1;
end
fprintf('Num. de lineas: %d\n', n);

% Cierre
fclose(fid);
```

Conteo de número de datos en fichero con `feof`

```
fid=fopen('res.txt','r');
n=0;
while ~feof(fid)
    fscanf(fid,'%f',1);
    n=n+1;
end
```

Gráfica a partir de datos de un archivo

```
% Datos de gráfica en un archivo 'misteriosa.txt'
% cada linea contiene una ordenada con su abscisa
fid=fopen('misteriosa.txt','r'); %Apertura (lectura)

aux=fgetl(fid);   %retiramos 1ª linea (cabecera)
%datos en matriz de 2 filas (columnas las necesarias)
datos = fscanf(fid,'%f', [2,inf]);
x = datos(1,:); y = datos(2,:);   %abs & ords.
plot(x,y,'Linewidth',2);          %grafica

fclose(fid);            %Cierre
```

Matlab/Octave: Bestiario de comandos

notas.txt

```
Calificaciones del examen final
Nombre y apellidos                Ej1     Ej2     Ej3
Sandra Joan Segundo               9       4.5     6
Pedro Díaz Sánchez                7       8.1     3.9
... ... ...
```

Media de fichero con notas (I)

```
fid=fopen('notas.txt','r'); %Apertura (lectura)
%retiramos dos 1as lineas
aux=fgetl(fid); aux=fgetl(fid);
n=0;
medias=[ ];
while ~feof(fid)
    fscanf(fid,'%30c',1);    %nombre (30 caracteres)
    notas=fscanf(fid,'%f',[1,3]);
    medias=[medias,sum(notas)/3];
    n=n+1;
end
fprintf('El número de alumnos presentados es %d\n',n);
fprintf('La media de notas es: %f\n',sum(medias)/n);
fclose(fid);
```

Media de fichero con notas (II)

```
fid=fopen('notas.txt','r'); %Apertura (lectura)
%retiramos dos 1as lineas
aux=fgetl(fid); aux=fgetl(fid);

% usamos * en el formato para ignorar los 30 primeros
% caracteres, almacenamos los 3 números siguientes
notas_todos = fscanf(fid,'%*30c %f %f %f', [3,inf]);

% cada columna corresponde a 3 notas de un estudiante
n = size(notas_todos,2);
medias = sum(notas_todos)/3;
fprintf('El número de alumnos presentados es %d\n',n);
fprintf('La media de notas es: %f\n',sum(medias)/n);
fclose(fid);
```

Matriz de 4 columnas numéricas desde fichero

```
fid=fopen('matriz.txt','r'); %Apertura (lectura)

% Leemos la matriz por filas EN ORDEN COLUMNA
[A,cont] = fscanf(f1,'%f',[4,inf]);

% Tomamos la traspuesta para obtener matriz original
A=A';

fclose(fid);
```

Gráfico paramétrico: cardioide

```
% Ec. en polares: rho = 1-cos(theta)
% Dominio del parametro theta: [0,2*pi]
% Discretizamos con 100 puntos
tt=linspace(0,2*pi,100);
% Pasamos de coordenadas polares a cartesianas
x = (1-cos(tt)).*cos(tt);
y = (1-cos(tt)).*sin(tt);
plot(x,y)
```

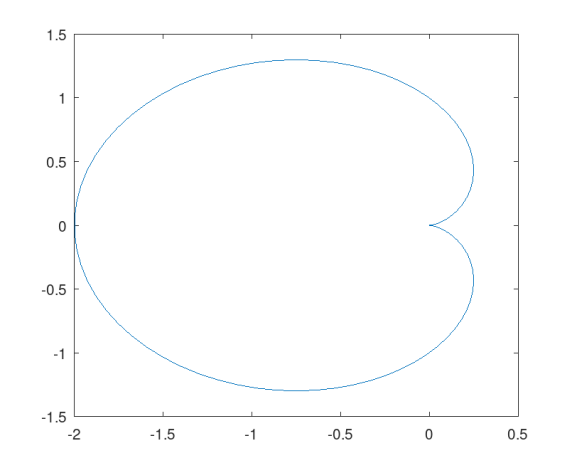
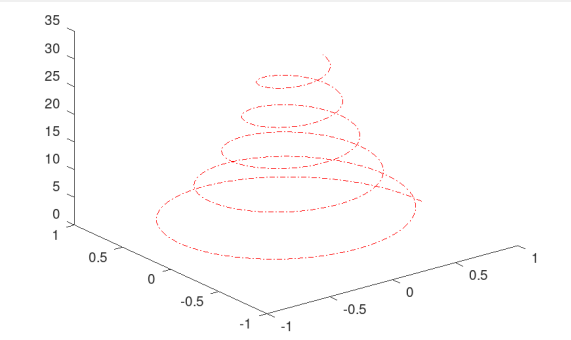


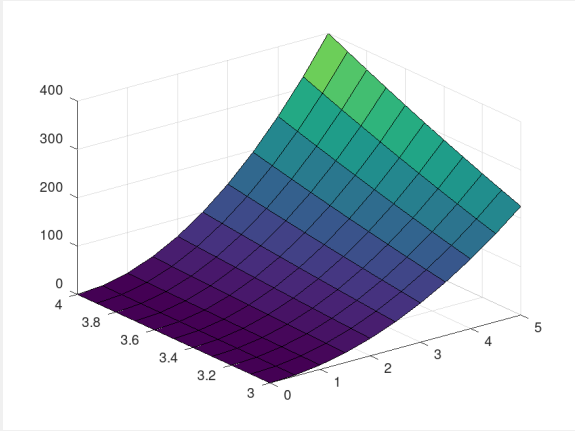
Gráfico paramétrico en 3D con `plot3`

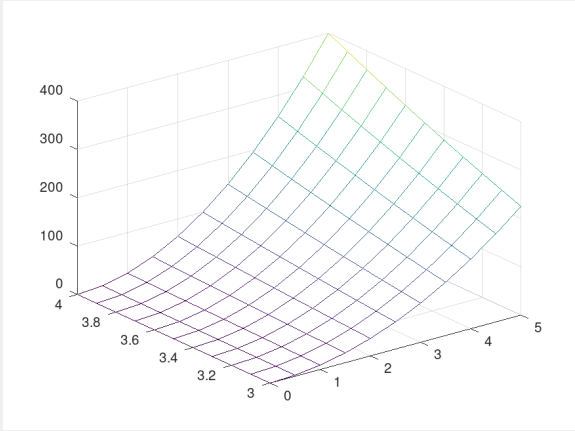
```
t = 0:0.1:10*pi;
x = exp(-t/20).*cos(t);
y = exp(-t/20).*sin(t);
z = t;
plot3(x,y,z,'ro--');
```



Superficies: gráficas de funciones $z = f(x, y)$

```
% GRAFICA DE z = x^2*y^2
% Creamos el mallado para (x,y) en [0,5]x[3,4]
x = 0:0.5:5;
y = 3:0.1:4;
[Mx,My]= meshgrid(x,y);
% Se aplica la función declarada ELEMENTO
% A ELEMENTO A los puntos de la malla
f=@(x,y) (x.^2).*(y.^2);
Mz=f(Mx,My);
% Dibujamos la superficie definida por
% los puntos Mx, My, Mz
figure
surf(Mx,My,Mz)
% Ahora solo el mallado
figure
mesh(Mx,My,Mz)
```





Autor: Juan Viu Sos • ✉ • 🏠 • git • Dpto. de Matemáticas e Informática aplicadas a las Ing. Civil y Naval – UPM

Actualizado: 28 de noviembre de 2022