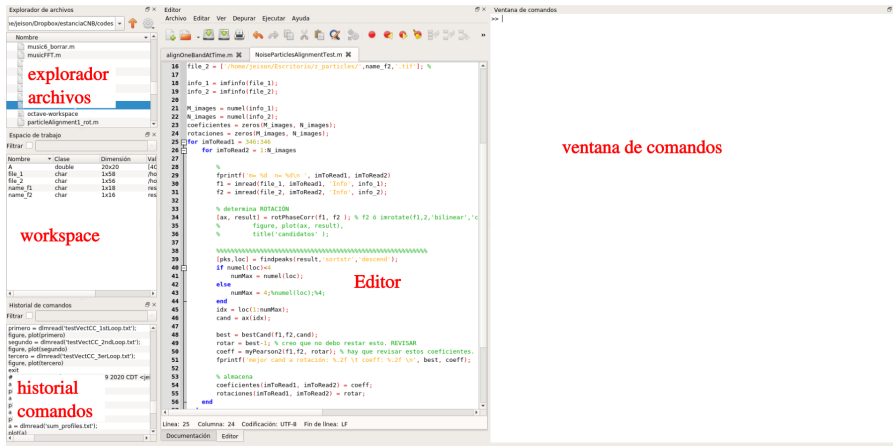


# Conceptos básicos de MATLAB

- MATLAB (MATrix LABoratory) es un programa para hacer cálculos numéricos con vectores y matrices (valor real y complejo), cadenas de caracteres y otras estructuras.
- Permite construir herramientas reutilizables (funciones y *M-files*) las cuales también pueden ser agrupadas en librerías.
- Además de permitir hacer cálculo matricial y álgebra lineal, también permite manejar polinomios, funciones, ecuaciones diferenciales ordinarias, gráficos ...



Variable	Definición	Valor
<b>ans</b>	almacena el último resultado	
<b>pi</b>	razón de circunferencia diámetro	3.1416
<b>eps</b>	$\epsilon \rightarrow$ valor pequeño a sumar (división por cero)	2.2204e-016
<b>inf</b>	infinito	Inf
<b>nan</b>	no numérico	Nan
<b>i y j</b>	$\sqrt{-1}$	$0 + 1.0000i$
<b>realmin</b>	real más pequeño utilizable	2.2251e-308
<b>realmax</b>	real más grande utilizable	1.7977e+308

Para consultar el funcionamiento de un determinado comando se puede usar:

- digitar en la ventana de comandos  
>>> **help** <comando a consultar>
- O simplemente **help**
- Ventana de ayuda usando F1

Función	operación	$x = 5.92$
<b>ceil</b> (x)	redondea al infinito	6
<b>fix</b> (x)	redondea hacia cero	5
<b>floor</b> (x)	redondea hacia menos infinito	5
<b>round</b> (x)	hacia el entero más próximo	6

# Funciones matemáticas

Función	Operación
$\dots(x)$	función trigonométrica con el ángulo expresado en radianes
$\sin(x)$ $\cos(x)$ $\tan(x)$ $\csc(x)$ $\sec(x)$ $\cot(x)$	funciones trigonométricas para seno, coseno, tangente ... ... $x$ (valor en radianes)
$\dots d(x)$	ángulo expresado en grados. e.g., $\text{sin}d(x)$
$\dots h(x)$	función hiperbólica (radianes). e.g., $\text{sinh}(x)$
$a\dots(x)$	inversa de la función trigonométrica (resultado en radianes). e.g., $\text{asin}(x)$
$a\dots d(x)$	inversa de función (resultado en grados). e.g., $\text{asind}(x)$
$a\dots h(x)$	inversa de función hiperbólica (resultado en radianes). e.g., $\text{asinh}(x)$

# Funciones matemáticas

Función	Operación
$\text{abs}(x)$	valor absoluto o magnitud de un número complejo
$\text{sign}(x)$	signo del argumento si $x$ es un valor real (-1 si es negativo, 0 si es cero, 1 si es positivo)
$\text{exp}(x)$	exponencial
$\text{gcd}(m,n)$	máximo común divisor
$\text{lcm}(m,n)$	mínimo común múltiplo
$\log(x)$	logaritmo neperiano
$\log_2(x)$	logaritmo base 2
$\log_{10}(x)$	logaritmo base 10
$\text{mod}(x,y)$	operación módulo
$\text{rem}(x,y)$	resto de la división entera
$\text{sqrt}(x)$	raíz cuadrada
$\text{nthroot}(x,n)$	raíz $n$ -ésima de $x$



# Funciones matemáticas

Función	Operación
<code>abs(x)</code>	magnitud del número complejo $x$
<code>angle(x)</code>	ángulo (en radianes) del complejo $x$
<code>complex(y,z)</code>	genera el complejo $y + i*z$
<code>conj(x)</code>	conjugado del número complejo $x$
<code>imag(x)</code>	parte imaginaria del número complejo $x$
<code>real(x)</code>	parte real del número complejo $x$
<code>sign(x)</code>	divide el complejo $x$ por su magnitud, devuelve un número complejo con el mismo ángulo de fase pero con magnitud 1
<code>isreal(x)</code>	devuelve 1 si es real, y 0 si es complejo

# Vectores y matrices

# construcción de vectores y matrices

- $x = [5 \ 7 \ -2 \ 4 \ -6]$
- $y = [2,1,3,7]$
- $z = [0 \ 1 \ 2,3 \ 4,5]$
- $A = [1 \ 2 \ 3; 4 \ 5 \ 6]$

Se puede acceder al  $n$ -ésimo elemento del vector usando  $x(n)$ .

**Los índices inician en 1 !!**

En el caso de matrices  $A(m,n) \rightarrow$  elemento en la fila  $m$ , columna  $n$

Los vectores y matrices se pueden concatenar (teniendo cuidado con las dimensiones que éstos tengan); p.e.,  $m = [x \ , \ y]$   
(concatenación por filas, en este caso funciona), pero  $m = [x \ ; \ y]$   
(concatenación por columnas, en este caso falla)

# construcción “abreviada” de vectores y matrices

- **(a:b)** crea un vector que comienza en el valor a y acaba en el valor b aumentando de 1 en 1.
- **(a:c:b)** crea un vector que comienza en el valor a y acaba en el valor b aumentando una cantidad c en cada paso.
- **linspace (a,b,c)** genera un vector linealmente espaciado entre los valores a y b con c elementos.
- **linspace (a,b)** genera un vector linealmente espaciado entre los valores a y b con 100 elementos.
- **logspace (a,b,c)** genera un vector logarítmicamente espaciado entre los valores  $10^a$  y  $10^b$  con c elementos.

# Algunas matrices predefinidas

- **zeros( $n$ )** crea una matriz cuadrada  $n \times n$  de ceros.
- **zeros( $m,n$ )** crea una matriz  $m \times n$  de ceros.
- **ones( $n$ )** crea una matriz cuadrada  $n \times n$  de unos.
- **ones( $m,n$ )** crea una matriz  $m \times n$  de unos.
- **rand( $n$ )** crea una matriz cuadrada  $n \times n$  de números aleatorios con distribución uniforme  $(0,1)$ .
- **rand( $m,n$ )** crea una matriz  $m \times n$  de números aleatorios con distribución uniforme  $(0,1)$ .
- **randn( $n$ )** crea una matriz cuadrada  $n \times n$  de números aleatorios con distribución normal  $(0,1)$ .
- **eye( $n$ )** crea una matriz cuadrada  $n \times n$  de unos en la diagonal y ceros el resto.
- **magic( $n$ )** crea una matriz cuadrada  $n \times n$  de enteros de modo que suman lo mismo las filas y las columnas.

# Operaciones con matrices

Símbolo	Expresión	Operación
+	$A + B$	Suma de matrices
-	$A - B$	Resta de matrices
*	$A * B$	Multiplicación de matrices
.*	$A .* B$	Multiplicación elemento a elemento de matrices $a_{ij} * b_{ij}$
/	$A / B$	División de matrices por la derecha $A * \text{inv}(B)$
./	$A ./ B$	División elemento a elemento de matrices por la derecha $a_{ij} * \frac{1}{b_{ij}}$
\	$A \setminus B$	División de matrices por la izquierda $\text{inv}(A) * B$
.\	$A \setminus B$	División elemento a elemento de matrices por la izquierda
^	$A ^ n$	Potenciación (n debe ser un número, no una matriz)
.^	$A .^ B$	Potenciación elemento a elemento de matrices
'	$A '$	Trasposición compleja conjugada
.'	$A .'$	Trasposición de matrices

# Análisis de matrices

Función	Operación
$\det(A)$	determinante
$\text{diag}(v)$	crea una matriz diagonal con el vector $v$ sobre la diagonal
$\text{diag}(A)$	extrae la diagonal de la matriz $A$ como un vector columna
$\text{eig}(A)$	valores propios
$\text{inv}(A)$	matriz inversa
$\text{norm}(A)$	norma de $A$
$\text{orth}(A)$	ortogonalización
$\text{pinv}(A)$	pseudoinversa
$\text{rref}(A)$	reducción mediante la eliminación de Gauss de una matriz
$\text{size}(A)$	dimensiones
$\text{tril}(A)$	$A$ como triangular inferior
$\text{triu}(A)$	$A$ como triangular superior

# Otras operaciones importantes

Función	Operación
<code>find(A)</code>	índices de las entradas de A que son mayores que 0
<code>[VE,VA] = eig(A)</code>	VE son los vectores propios y VA son los valores propios
<code>[L,U] = lu(A)</code>	factorización LU
<code>[Q,R] = qr(A)</code>	factorización QR
<code>[U, S, V] = svd(A)</code>	descomposición de valor singular $A = U \cdot S \cdot V'$



# Estructuras

Permite agrupar datos de diferente naturaleza. Se puede usar el comando **struct** o llenar los campos de forma individual.

```
>> alumno(1) = struct  
( 'nombre', 'Pablo', 'apellido1', 'fernández', 'apellido2', 'García', 'edad', 15 );  
  
>> alumno(2) = struct  
( 'nombre', 'Fermín', 'apellido1', 'Martínez', 'apellido2', 'Gil', 'edad', 16 );
```

Función	Operación
fieldnames(E)	devuelve el nombre de los campos de la estructura E
isfield(E, 'c')	devuelve 1 si 'c' es un campo de la estructura E y 0 si no lo es
isstruct(E)	devuelve 1 si E es una estructura y 0 si no lo es
rmfield(E, 'c')	elimina el campo 'c' de la estructura E

# Vectores y matrices de celdas

Esta estructura de datos permite elementos de diferentes tipos  
(vectores, matrices, cadenas de caracteres, estructuras, ... )

```
>> celda(1) = {[0 1 2]};  
>> celda(2) = {'cadena de caracteres'};  
>> celda(3) = {eye(2)};  
>> celda(4) = {-7};  
>> celda  
celda =  
[1x3 double] [1x20 char] [2x2 double] [-7]
```

```
>> cel{1} = [0 1 2];  
>> cel{2} = 'cadena de caracteres';  
>> cel{3} = eye (2);  
>> cel{4} = -7;  
>> cel  
cel =  
[1x3 double] [1x20 char] [2x2 double] [-7]
```

# Algunas operaciones

Para **m** y **n** números naturales, **c** celdas y **x** vector o matriz

Función	Operación
<b>cell(m,n)</b>	crea una matriz de celdas con <b>m</b> filas y <b>n</b> columnas
<b>celldisp(c)</b>	muestra el contenido de todas las celdas de <b>c</b>
<b>cellplot(c)</b>	muestra la representación gráfica de las celdas de <b>c</b>
<b>iscell(c)</b>	devuelve 1 si es una matriz de celdas y 0 si no lo es
<b>num2cell(x)</b>	convierte el vector o matriz numérica en celdas

# Programación en MATLAB

## Sentencia **FOR**

```
for x = 1:5
    disp('valor de x es:')
    disp(x)
end
```

## Sentencia **WHILE**

```
while a < 5
    disp('valor de a es:  ')
    disp(a)
    a = a + 1;
end
```

## Sentencia **Condicionales**

```
if b == 0
    disp('b vale 0')
elseif b == 1
    disp('b vale 1')
elseif b == 2
    disp('b vale 2')
elseif b == 3
    disp('b vale 3')
else
    disp('b no vale ni 0 ni 1 ni 2 ni 3')
end
```

En el caso de usar **OCTAVE** las sentencias terminan con **endfor**, **endwhile** y **endif**, respectivamente

# M-files y funciones



# M-files

Un M-file es un fichero de comandos que se ejecutan sucesivamente cuando se teclea el nombre del fichero en la línea de comandos de Matlab o se incluye dicho nombre en otro fichero \*.m.

p.e., se escribe en el editor y se guarda como 'operaciones.m':

```
x = 4
y = 3
suma = x + y
resta = x - y
producto = x*y
```

Al ejecutar en la ventana de comandos la instrucción 'operaciones' la salida es como se muestra:

```
>> operaciones
x = 4
y = 3
suma = 7
resta = 1
producto = 12
```

# Funciones

Se pueden construir funciones propias y almacenarlas en M-files individuales (aunque pueden estar también en el script que se quiere ejecutar).

En el caso de guardar la función en un M-file independiente se recomienda usar el mismo nombre para la función y para el archivo.

Para definir una función se usa:

**function [a,b,c]=nombre\_función(x,y,z)**

donde **a**, **b**, **c** son los argumentos de salida de la función y **x**, **y**, **z** son los de entrada.

En el caso de usar un M-file independiente para la función, la anterior debe ser la primera línea de código (luego de los comentarios y la descripción de la función en caso de incluirla).

En el caso de usar **OCTAVE** terminar la función usando **endfunction**

# Funciones

Existe flexibilidad en el número de argumentos de entrada y salida usando **varargin** y/o **varargout**.

La información sobre el número de entradas y salidas están en **nargin** y **nargout**.

```
function varargout=atan3(varargin)
% atan3 puede recibir 1 o 2 argumentos:  llama atan y atan2 respectivamente
% valores de retorno:  el ángulo en radianes o ángulo en radianes y en grados.
if nargin==1
    rad = atan(varargin{1});
elseif nargin==2
    rad = atan2(varargin{1},varargin{2});
else
    disp('Error:  más de dos argumentos')
    return
end
varargout{1}=rad;
if nargout>1
    varargout{2}=rad*180/pi;
end
```

# Exploración de datos

# Análisis estadístico

MATLAB permite hacer análisis estadístico sobre conjuntos de datos (matrices con orientación por columnas).

Función	Operación
<code>corrcoef(X)</code>	coeficientes de correlación
<code>cov(X)</code>	matriz de covarianzas
<code>cumprod(X)</code>	producto acumulativo de columnas
<code>cumsum(X)</code>	suma acumulativa de columnas
<code>diff(X)</code>	diferencias entre elementos adyacentes de X
<code>hist(X)</code>	histograma o diagrama de barras
<code>iqr(X)</code>	rango intercuartílico de la muestra
<code>max(X)</code>	máximo de cada columna
<code>mean(X)</code>	promedio por columnas
<code>median(X)</code>	mediana por columnas
<code>min(X)</code>	mínimo de cada columna
<code>range(X)</code>	rango ( $V_{\{max\}} - V_{\{min\}}$ ) por columna
<code>sort(X)</code>	orden ascendente por columna
<code>std(X)</code>	desviación estándar por columna
<code>sum(X)</code>	suma de elementos de cada columna
<code>var(X)</code>	varianza de los elementos por columna

# Polinomios

# Raíces y otras operaciones entre polinomios

Los polinomios se representan a través de sus coeficientes en un vector fila. Por ejemplo, para indicar el polinomio  $5x^4 + 3x^2 - x + 5$  se escribe  $[5,0,3,-1,5]$ .

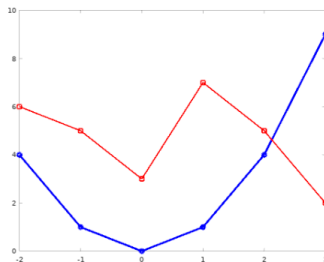
Función	Operación
roots(p)	raíces del polinomio
conv(p,q)	multiplica los dos polinomios p y q
deconv(c,q)	divide el polinomio c entre q
polyder(p)	calcula la derivada del polinomio p
polyder(p,q)	calcula la derivada del producto de los polinomios p y q
polyval(p,x)	evalúa el polinomio p en cada valor de x. $p(x) = p_1x^n + p_2x^{n-1} + \cdots + p_{n-1}$

# Representación gráfica



La instrucción **plot** genera representaciones gráficas 2D de vectores de vectores con **igual número de elementos**

```
>> x = [-2 -1 0 1 2 3]; y = [4 1 0 1 4 9]; z = [6 5 3 7 5 2];  
>> plot (x,y, '-ob', 'linewidth', 3, x,z, '-sr', 'linewidth', 2)
```



# algunos parámetros para modificar gráficas 2D












`xlabel('texto')`  
`ylabel('texto')`  
`title('texto')`  
`text(x,y, 'texto')`  
`gtext('texto')`

etiqueta sobre el eje X de la gráfica actual  
etiqueta sobre el eje Y de la gráfica actual  
título en la cabecera de la gráfica actual  
texto en el lugar especificado por las coordenadas  
texto que se ubica en la coordenada que indiquemos con el ratón

`grid`  
`axis( [xmin xmax ymin ymax] )`  
`axis equal`  
`axis square`  
`axis normal`

dibujar una rejilla  
fija valores máximo y mínimo de los ejes  
fija que la escala en los ejes sea igual  
fija que la gráfica sea un cuadrado  
desactiva `axis equal` y `axis square`

# Otras gráficas

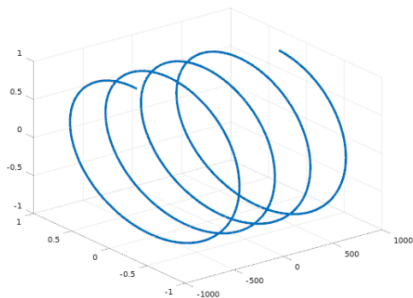
Orden	¿Qué hace?	Imagen
<b>area</b>	colorea el area bajo la gráfica	
<b>bar</b>	diagrama de barras (verticales)	
<b>barh</b>	diagrama de barras (horizontales)	
<b>hist</b>	histograma	
<b>pie</b>	sectores	
<b>rose</b>	histograma polar	
<b>stairs</b>	gráfico de escalera	
<b>stem</b>	secuencia de datos discretos	
<b>loglog</b>	como plot pero con escala logarítmica en ambos ejes	
<b>semilogx</b>	como plot pero escala logarítmica en el eje x	
<b>semilogy</b>	como plot pero escala logarítmica en el eje y	

# 3D

En este caso se usa la instrucción **plot3**. Los vectores deben tener el mismo número de elementos.

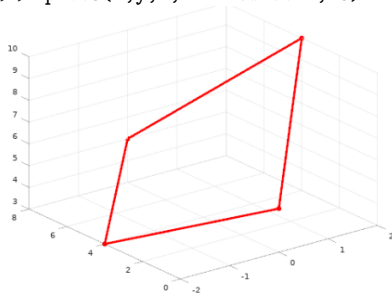
```
>> x = -720:720; y = sind (x); z =  
cosd (x);
```

```
>> plot3(x,y,z, 'linewidth', 3)
```



```
>> x = [-2 0 2 0 -2]; y = [4 8 4 0 4];  
z = [3 5 10 5 3];
```

```
>> plot3(x,y,z, 'linewidth', 3)
```

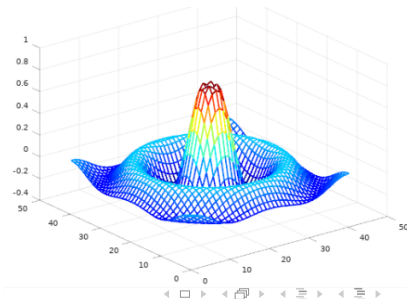
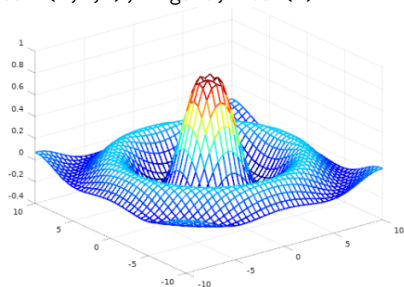


# superficies de malla

La orden `[X,Y]=meshgrid(x,y)` crea una matriz **X** cuyas filas son copias del vector **x** y una matriz **Y** cuyas columnas son copias del vector **y**.

Para generar la gráfica de malla se usa la orden `mesh(X,Y,Z)`. También puede tomar una matriz simple como argumento: `mesh(Z)`.

```
x = -10:0.5:10; y = -10:0.5:10;  
[X,Y] = meshgrid (x,y);  
Z = sin (sqrt (X .^2 + Y .^2)) ./ sqrt (X .^ 2 + Y .^ 2 + 0.1);  
mesh (X,Y,Z); figure, mesh(Z)
```

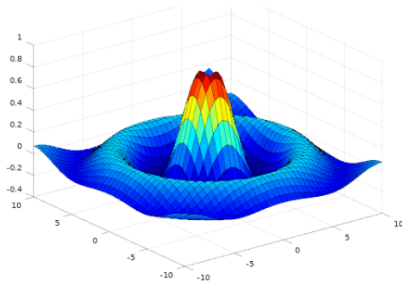


# superficies con color

La instrucción **surf(X,Y,Z)** permite generar una superficie de forma similar a la instrucción **mesh(X,Y,Z)** pero adicionalmente “colorea” los espacios entre líneas del mallado de superficie.

Para las matrices X, Y, Z definidas antes.

```
>> surf (X,Y,Z)
```

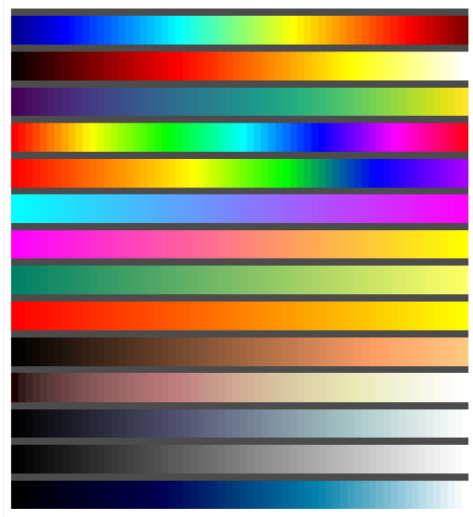


Se puede cambiar el mapa de colores usando la instrucción **colormap** **<mapa a usar>**. Para ver los mapas disponibles usar **help colormap**

```
cellColor={'jet','hot','viridis','hsv','rainbow','cool',...  
'spring','summer','autumn','copper','pink','bone','gray','ocean'};
```

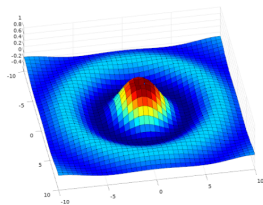
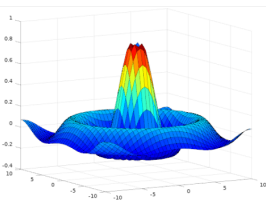
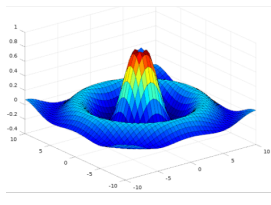
Generar una matriz de  $64 \times 3$  (64 colores diferentes)

```
col = colormap('hot');
```



# cambiar “punto de vista”

Se puede cambiar usando la instrucción **view(azimuth,elevation)** o **view([x,y,z])**





# Ficheros de entrada y salida

# Lectura

```
1 fid = fopen('testFile.txt','r');
2
3 data = textscan(fid,'%s');
4 fclose(fid);
5 stringData = data{:};
6
7 nCol = 3;
8 linesToSkip=2;
9 nFil = numel(stringData)/nCol;
10
11 matHeader = {};
12 matData = zeros(nFil-linesToSkip,nCol);
13
14 for ii=1:linesToSkip
15     for jj=1:nCol
16         idx= (ii-1)*nCol+jj;
17         matHeader{ii,jj} = stringData{idx};
18     endfor
19 endfor
20
21
22 for ii=linesToSkip+1:nFil
23     for jj=1:nCol
24         idx= (ii-1)*nCol+jj;
25         matData(ii-linesToSkip,jj) = str2double(stringData{idx});
26     endfor
27 endfor
28
```

```
uno dos tres
cuatro cinco seis
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
```

```
>> matHeader
matHeader =
{
    [1,1] = uno
    [2,1] = cuatro
    [1,2] = dos
    [2,2] = cinco
    [1,3] = tres
    [2,3] = seis
}
```

```
>> matData
matData =

     1     2     3
     4     5     6
     7     8     9
    10    11    12
    13    14    15
```

```
fid=fopen('header.txt','wt');      uno,dos,tres
[rows,cols]=size(matHeader)      cuatro,cinco,seis
for i=1:rows
    fprintf(fid,'%s ',matHeader{i,1:end-1})
    fprintf(fid,'%s\n',matHeader{i,end})
end
fclose(fid);
dlmwrite('data.txt',matData)
```

1,2,3  
4,5,6  
7,8,9  
10,11,12  
13,14,15