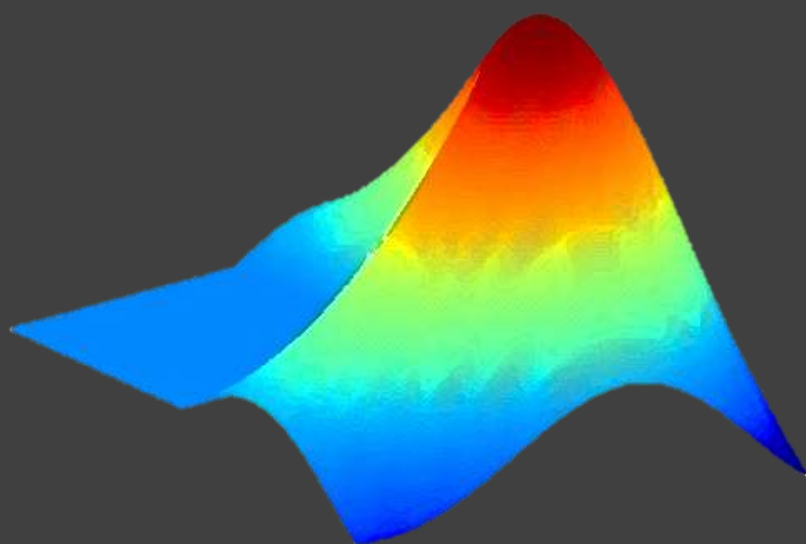


# Ejercicios resueltos de programación en MATLAB



Pedro Jorge De Los Santos



Versión 0.1

# Ejercicios resueltos de programación en MATLAB

Pedro Jorge De Los Santos

10 de febrero de 2016

# Índice general

<b>1. Programación básica e intermedia</b>	<b>5</b>
1.1. Número par / impar . . . . .	5
1.2. Lista de números pares . . . . .	5
1.3. Números primos . . . . .	6
1.4. Área y perímetro de un círculo . . . . .	6
1.5. Conversión grados – radianes . . . . .	7
1.6. Promedio de una lista de valores . . . . .	7
1.7. Sucesión de Fibonacci . . . . .	8
1.8. Múltiplos de un número . . . . .	8
1.9. Ecuación cuadrática . . . . .	8
1.10. Tiempo de vida . . . . .	9
1.11. Adivinar el número . . . . .	9
1.12. Imprimir rombo . . . . .	10
1.13. Factorial de un número . . . . .	10
1.14. Imprimir triángulo . . . . .	11
1.15. Sumatoria de n múltiplos de 5 . . . . .	11
1.16. Tabla de funciones trigonométricas . . . . .	11
1.17. Tabla de multiplicar . . . . .	12
<b>2. Matrices y vectores</b>	<b>13</b>
2.1. Valor máximo de un vector . . . . .	13
2.2. Matriz identidad . . . . .	13
2.3. Elementos diferentes en un vector . . . . .	13
2.4. Ordenar elementos de un vector . . . . .	14
2.5. Adjunta de una matriz de 3x3 . . . . .	14
2.6. Inversa de una matriz 2x2 . . . . .	15
<b>3. Cadenas de caracteres</b>	<b>16</b>
3.1. Invertir una cadena de texto . . . . .	16
3.2. Contar palabras . . . . .	16
3.3. Contar vocales en una cadena de texto . . . . .	16
3.4. Ordenar palabras . . . . .	17
3.5. Cifrado básico . . . . .	17
<b>4. Gráficas</b>	<b>18</b>
4.1. Gráfica de una función a trozos . . . . .	18
4.2. Gráfica de una desigualdad (inecuación) . . . . .	18
4.3. Gráfica de una esfera . . . . .	19
<b>5. Matemáticas</b>	<b>20</b>
5.1. Ecuación de la recta que pasa por dos puntos . . . . .	20

<b>6. Interfaces gráficas de usuario</b>	<b>21</b>
6.1. El hola mundo en GUI . . . . .	21
6.2. Ventana multicolor . . . . .	21
6.3. Lista de archivos M. . . . .	22
6.4. Mini Calculadora . . . . .	22
6.5. Visor de imágenes (Nivel I) . . . . .	24
<b>7. POO</b>	<b>25</b>
7.1. La clase Persona . . . . .	25
7.2. Clase Math (Métodos estáticos) . . . . .	25

## Acerca de...

Este pequeño *libro* se ha escrito con la finalidad de aportar una herramienta auxiliar a la comunidad hispanohablante de MATLAB. Orientado, sobre todo, a quienes dan los primeros pasos en la programación con este entorno de desarrollo.

Los problemas se han clasificado en diversas temáticas, para facilitar el estudio de los mismos. Todos los códigos de este libro están disponibles en el siguiente repositorio de GitHub: [ERPM](#).

El autor agradece cualquier tipo de comentario, observación u opinión, y puede remitirla a la dirección de correo electrónico que se adjunta posteriormente, o bien, contactándonos a través de las diversas plataformas cuyos enlaces se incluyen en la parte inferior.

Pedro Jorge De Los Santos  
delossantosmfq@gmail.com



# 1. Programación básica e intermedia

## 1.1 Número par / impar

*Desarrolle un programa que determine si un número es par o impar.*

**Solución.**

Este ejercicio es uno de los más comunes en los cursos básicos de programación. Para resolverlo hay que tener en cuenta algunas nociones básicas de matemáticas: es sabido que cualquier número par es divisible por 2, lo cual nos lleva al supuesto de que si un número es par necesariamente el resto de la división entre 2 debe ser igual a cero. La función `rem` de MATLAB devuelve 0 si el residuo de la división entera es nulo, y devuelve 1 si es una cantidad igual o mayor a la unidad. Basándonos en lo anterior podemos diseñar un programa que ejecuta la comprobación y que muestre en pantalla si el número es par o impar.

```
num=input('Numero: ');
if rem(num,2)==0
    disp('Numero par');
else
    disp('Numero impar');
end
```

El programa anterior utiliza la sentencia compuesta `if-else` para llevar a cabo dicha tarea. Es posible también usar la sentencia `switch` tal como se muestra enseguida, los resultados, desde luego, son iguales.

```
num=input('Numero: ');
switch rem(num,2)
    case 0
        disp('Numero par');
    otherwise
        disp('Numero impar');
end
```

## 1.2 Lista de números pares

*Desarrolle un programa que proporcione los primeros números pares que se indiquen:*

**Solución.**

Para desarrollar la solución a este problema se hará uso del operador *dos puntos*, el cual permite crear una lista o vector de valores numéricos con un cierto patrón, con la sintaxis:

```
v=a:h:b;
```

Donde  $a$  es el valor inicial,  $h$  el incremento y  $b$  el valor final. En este caso necesitamos una lista de valores que comience con el primer número par (2), con incrementos de 2, y cuyo valor final dependa de la cantidad de número pares requeridos. Por ejemplo, si se requieren 3 números pares, se esperaría que el último valor sea 6, que es equivalente a tener  $2 \times 3$ , así, de forma generalizada, el valor final será  $2n$ , donde  $n$  es la cantidad de números pares.

```
n=input('Cuántos numeros pares necesita? ');
pares=2:2:2*n;
disp(pares);
```

### 1.3 Números primos

*Escriba una función que le permita determinar si un número entero pasado como argumento es primo, en caso de serlo devolverá un valor lógico **true** y un valor **false** en caso contrario. (La función **isprime** de MATLAB realiza la misma operación).*

**Solución.**

Primero, sin tantos formalismos, un número primo es aquel cuyos únicos divisores son el uno y él mismo. Así, un programa básico puede estructurarse haciendo la comprobación, uno a uno, si cada número en el intervalo  $[1, n]$  es divisor del número en cuestión. Si la cantidad de divisores es mayor a 2, entonces el número no es primo. Véase el código siguiente propuesto como solución:

```
function r = esprimo(n)
L=1:n;
if nnz(rem(n,L)==0)==2
    r=true;
else
    r=false;
end
end
```

En resumen: sea define un vector  $L$  de  $n$  elementos que contiene todos los números en el intervalo  $[1, n]$ , enseguida se aplica la función **rem** a todos los elementos del vector  $L$  y a esto se aplica la condición de ser igual a cero, resultando un vector de tipo lógico, donde los unos corresponden a aquellos números donde el residuo de la división entera es cero y por tanto divisores de  $n$ ; finalmente, mediante la función **nnz** se cuentan los elementos que no son ceros, si estos son igual a dos, entonces se tiene un número primo.

### 1.4 Área y perímetro de un círculo

*Escriba un programa cuya entrada sea el radio del círculo e imprima el valor del área y perímetro correspondiente.*

**Solución.**

Para resolver este problema, lo primero es remitirnos a los cursos de geometría básica y recordar que el área y perímetro (circunferencia) de un círculo vienen dados por:

$$A = \pi r^2$$

$$P = 2\pi r$$

Con lo anterior se plantea una solución:

```
radio=input('Radio del círculo: ');
area=pi*radio^2;
perimetro=2*pi*radio;
fprintf('Área: %g \n',area);
fprintf('Perímetro: %g \n',perimetro);
```

## 1.5 Conversión grados – radianes

Utilice cualquier sentencia de selección múltiple para desarrollar un programa que le permita hacer la conversión de grados sexagesimales a radianes y viceversa.

**Solución.**

Sea  $\theta_r$  un ángulo dado en radianes y  $\theta_g$  en grados sexagesimales, las *formulas* de conversión resultan en:

$$\theta_r = \theta_g \left( \frac{\pi}{180} \right)$$

$$\theta_g = \theta_r \left( \frac{180}{\pi} \right)$$

```
opcion=input(['Seleccione una opción: \n 1. Grados a radianes'...
'\n 2. Radianes a grados \n']);
theta=input('Inserte la magnitud del angulo: ');
switch opcion
    case 1
        thetaR=theta*pi/180;
        fprintf('%g ° = %g rad \n',theta,thetaR);
    case 2
        thetaG=theta*180/pi;
        fprintf('%g rad = %g ° \n',theta,thetaG);
    otherwise
        disp('Opcion incorrecta');
end
```

## 1.6 Promedio de una lista de valores

Realice un programa que pida la cantidad de elementos o datos a introducir y que posteriormente sean ingresados manualmente, con los datos anteriores calcular el promedio de los mismos.

**Solución.**

```
n=input('Cantidad de datos a introducir: ');
for i=1:n
    D(i)=input(['Dato ',num2str(i),': ']);
end
fprintf('El promedio de los datos introducidos es %g\n',mean(D));
```



## 1.7 Sucesión de Fibonacci

*Escriba un programa que imprima en pantalla los primeros  $n$  términos de la sucesión de Fibonacci.*

**Solución.**

La sucesión de Fibonacci es una sucesión infinita de números naturales, cuyos términos resultan de la suma de los dos términos anteriores. Los primeros dos términos son 1 y 1, y enseguida el resto siguen el patrón descrito con anterioridad, así, algunos de los primeros términos se muestran enseguida:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Luego, cada término de la sucesión se calcula bajo la siguiente especificación:  $f_i = f_{i-1} + f_{i-2}$ . La implementación en MATLAB resulta muy sencilla, se definen inicialmente los dos primeros términos y enseguida se concatenan los valores de la sucesión que siguen el patrón mencionado, mediante un bucle. Un código de ejemplo se muestra a continuación:

```
n=input('Numero de terminos a mostrar: ');
F=[1 1];
for i=3:n
    F(i)=F(i-1)+F(i-2);
end
disp(F);
```

## 1.8 Múltiplos de un número

*Desarrolle una función que devuelva los primeros  $k$  múltiplos de un determinado número.*

**Solución.**

Un múltiplo de un número es aquel que lo contiene un número entero de veces. Luego, los primeros  $k$  múltiplos de un número  $N$  cualesquiera, vienen dado por  $< 1, 2, \dots, k-1, k > *N$ . La implementación en MATLAB es muy sencilla, utilizando la notación de dos puntos para crear un vector cuyos elementos sean los enteros desde 1 hasta  $k$ .

```
function x = multiplos(num,k)
% Devuelve los primeros k multiplos de num.
%
x=(1:k)*num;
end
```

## 1.9 Ecuación cuadrática

*Escriba una función que le permita resolver una ecuación de segundo grado de la forma  $ax^2 + bx + c = 0$ , siendo los coeficientes los argumentos de entrada y las raíces de la ecuación los valores de salida.*

**Solución.**

La solución general de una ecuación de segundo grado de la forma  $ax^2 + bx + c = 0$  viene dada por la tan conocida **formula general**.

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
function [x1,x2]=ecuadratica(a,b,c)
% Resuelve una ecuacion de segundo orden, siendo
% los argumentos de entrada los coeficientes de
% la misma.
x1=(-b+sqrt(b^2-4*a*c))/(2*a);
x2=(-b-sqrt(b^2-4*a*c))/(2*a);
end
```

## 1.10 Tiempo de vida

*Desarrolle un script que le permita calcular el número de años, meses, días y horas que ha vivido un individuo.*

**Solución.**

```
D=input('Día de nacimiento: ');
M=input('Mes de nacimiento: ');
A=input('Año de nacimiento: ');
actual=now;
nac=datetime(A,M,D);
fprintf('\nCuanto has vivido?\n');
fprintf('Años = %g \n', (actual-nac)/365);
fprintf('Meses = %g \n', (actual-nac)/30);
fprintf('Días = %g \n', (actual-nac));
fprintf('Horas = %g \n', (actual-nac)*24);
```

## 1.11 Adivinar el número

*Escriba un programa que genere un número entero aleatorio entre 1 y 100, y posteriormente le pida al usuario tratar de adivinar el número en cuestión, para cada intento el programa debe mostrar si el número que se trata de adivinar es mayor o menor al propuesto por el jugador. El programa termina cuando el jugador adivina el número, imprimiéndose en pantalla la cantidad de intentos realizados.*

**Solución.**

Para generar el número entero aleatorio en el rango [1, 100] se utilizará la función `randi`, que devuelve un entero aleatorio en el intervalo que se especifique. Dentro de un bucle `while` se hacen las comprobaciones de equivalencia del número aleatorio con el número propuesto por el jugador. Además se utiliza un contador que se incrementa en la unidad cada vez que se ejecuta un ciclo. El ciclo se *rompe* cuando el jugador adivina el número mediante la instrucción `break`.

```
n=randi([1 100]);
k=0;
while 1
    np=input('Numero?: ');
    k=k+1;
```

```

    if n<np
        disp('Es menor');
    elseif n>np
        disp('Es mayor');
    else
        break;
    end
end
fprintf('Lo has conseguido en %g intentos \n',k);

```

### 1.12 Imprimir rombo

Desarrolle un programa que imprima un rombo de asteriscos en pantalla para un valor dado de  $n$ , donde  $n$  es el número de filas y/o columnas totales del rombo, debe notar que  $n$  será siempre un entero impar mayor o igual 3. Para una mejor comprensión se muestran a continuación los resultados para  $n=3$  y  $n=7$ .



Solución.

```

n=input('Numero de filas del rombo: ');
R='';
for i=1:n
    if i<=ceil(n/2)
        R(i,[ceil(n/2)+(1-i) ceil(n/2)+(i-1)])='*';
    else
        R(i,[ceil(n/2)-(n-i) ceil(n/2)+(n-i)])='*';
    end
end
disp(R);

```

### 1.13 Factorial de un número

El factorial de un entero positivo  $n$  viene dado por  $n! = (1)(2)\dots(n-1)(n)$ , es decir, el producto de todos los enteros desde 1 hasta  $n$ . Realice un programa que devuelva el factorial de un número introducido (Recuerde que por definición el factorial de 0 es igual a la unidad).

Solución.

```

N=input('Introduzca un numero entero: ');
k=1;
fact=1;

```

```

while k<=N
    fact=k*fact;
    k=k+1;
end
fprintf('\n %d! = %d\n\n',N,fact);

```

### 1.14 Imprimir triángulo

Realice un script que imprima en pantalla un triángulo de asteriscos, el programa debe tener como entrada el número de filas del triángulo.

Solución.

```

n=input('Numero de filas: ');
A='';
for i=1:n
    A(i,1:2:2*i)='*';
end
disp(A)

```

### 1.15 Sumatoria de n múltiplos de 5

Escriba un programa que devuelva la suma de los primeros  $n$  múltiplos de 5, que además cumplan la condición de no ser múltiplos de 10.

```

N=input('Multiplos a sumar: ');
suma=0;
mult=5;
k=0;
while k<N
    if rem(mult,10)
        suma=suma+mult;
        k=k+1;
    end
    mult=mult+5;
end
disp(suma);

```

### 1.16 Tabla de funciones trigonométricas

Desarrolle un programa que imprima una tabla de valores para las funciones trigonométricas más comunes en el rango  $0^\circ$  a  $45^\circ$  (seno, coseno y tangente).

```

fprintf('=====\n');
fprintf(' °   sin(x)   cos(x)   tan(x)\n');
fprintf('=====\n');
for x=0:pi/180:pi/4
    fprintf('%0.0f \t %0.4f \t %0.4f \t %0.4f\n',...
        x*(180/pi),sin(x),cos(x),tan(x));
end

```

```
end
```

### 1.17 Tabla de multiplicar

*Escriba un programa que pida al usuario un entero positivo del cual se desarrollará e imprimirá en pantalla su tabla de multiplicación del 1 al 10, por ejemplo:*

```
Tabla del: 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

Solución.

```
N = input('Tabla del: ');
for k = 1:10
    fprintf('%g x %g = %g\n',N,k,N*k);
end
```

## 2. Matrices y vectores

### 2.1 Valor máximo de un vector

*MATLAB* utiliza la función `max` para determinar el máximo valor de un vector, escriba un programa que lleve a cabo la misma tarea, evitando hacer uso de la función `max`.

Solución.

Existen, claro, múltiples formas de proceder para resolver este problema, por ejemplo, se podrían ordenar los elementos del vector en forma descendente y entonces tomar el primer elemento que correspondería al de mayor valor. Pero, para este caso vamos a proceder de la forma más *simplista*, utilizando un bucle `for` para recorrer el vector e ir comprobando, elemento a elemento, si un elemento anterior es menor al actual. Suponemos, en principio, que el valor máximo es el primer elemento del vector y enseguida hacemos la comprobación para el resto de elementos, tal como se muestra.

```
v=input('Vector: ');
maxv=v(1);
for i=2:length(v)
    if v(i)>maxv
        maxv=v(i);
    end
end
fprintf('El valor maximo es %g\n\n',maxv);
```

### 2.2 Matriz identidad

La función `eye` crea una matriz identidad; programe una función llamada `idmat` que reciba como argumento la dimensión de la matriz (cuadrada) y devuelva una matriz identidad.

Solución.

```
function M = idmat(n)
M(1:n,1:n)=0;
for i=1:n
    M(i,i)=1;
end
end
```

### 2.3 Elementos diferentes en un vector

Escriba un programa que determine el número de elementos diferentes en un vector.

Solución.

```
v=input('Vector: ');
v=sort(v);
k=1;
for i=2:length(v)
    if v(i)~=v(i-1)
        k=k+1;
    end
end
fprintf('Hay %d elementos diferentes\n\n',k);
```

## 2.4 Ordenar elementos de un vector

La función `sort` de MATLAB ordena los elementos de un vector en forma ascendente o descendente, escriba una función llamada `ordenarasc` cuyo argumento de entrada sea un vector y devuelva como salida el mismo vector ordenado en forma ascendente (de menor a mayor).

Solución.

```
function xo = ordenarasc(x)
% Ordena un vector en forma ascendente
%
xo=zeros(size(x));
i=1;
while i<=length(xo)
    ne=length(x(x==min(x)));
    xo(i:i+ne-1)=min(x);
    x(x==xo(i))=[];
    i=i+ne;
end
end
```

## 2.5 Adjunta de una matriz de 3x3

Desarrolle una función que reciba como parámetro de entrada una matriz de 3x3 y que devuelva como salida la adjunta de dicha matriz.

Solución.

```
function X = adjunta3(A)
% Calcula la adjunta de una matriz de 3x3
% Entrada:
%     A - Matriz de 3x3
% Salida:
%     X - Matriz adjunta de A
%
if size(A,1)~=3 || size(A,2)~=3
    error('Introduzca una matriz de 3x3');
end
```

```

MC=zeros(size(A));
idx=1:3;
for i=1:size(A,1)
    for j=1:size(A,2)
        idxf=idx(idx~=i);
        idxc=idx(idx~=j);
        cof=A(idxf,idxc);
        if rem(i+j,2)
            MC(i,j)=-det(cof);
        else
            MC(i,j)=det(cof);
        end
    end
end
X=MC';
end

```

## 2.6 Inversa de una matriz 2x2

Desarrolle una función llamada *inversa22* que calcule la inversa de una matriz de 2x2. Evite usar la función *inv* de MATLAB.

Solución.

Una matriz de 2x2 tiene la forma general:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Del álgebra lineal sabemos que la inversa de una matriz de esta forma viene dada por:

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -c \\ -b & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -c \\ -b & a \end{bmatrix}$$

Implementando lo anterior en MATLAB:



## 3. Cadenas de caracteres

### 3.1 Invertir una cadena de texto

Escriba un programa que le permita invertir una palabra ingresada, por ejemplo, si usted introduce MATLAB deberá devolverle BALTAM.

Solución.

En esencia lo que se debe hacer es *invertir* los índices del vector tipo char en el cual se guarda la cadena de texto.

```
cad=input('Introduzca una palabra: ','s');  
disp(cad((end:-1:1)));
```

### 3.2 Contar palabras

Desarrolle un script que reciba como entrada una cadena de caracteres y que devuelva la cantidad de palabras que la componen. Para este caso se asumirá que el texto pasado como dato de entrada tiene una estructura coherente y libre de cualquier secuencia extraña de signos de puntuación u otro tipo de caracteres diferentes a los alfanuméricos.

Solución.

```
txt = input('Inserte un texto: ','s');  
resto = txt;  
k = 0; % Inicializa contador  
while true  
    [palabra, resto] = strtok(resto, ' ');  
    if isempty(palabra), break; end  
    k = k + 1;  
end  
fprintf('No. de palabras encontradas: %d\n\n',k);
```

### 3.3 Contar vocales en una cadena de texto

Escriba un programa que reciba como entrada una frase o cadena de texto y que muestre como salida el número de vocales que contiene dicha frase.

Solución.

```
cad=input('Introduzca una cadena de texto: ','s');  
k=0;
```

```

for i=1:length(cad)
    switch cad(i)
        case {'A','a','E','e','I','i','O','o','U','u'}
            k=k+1;
        otherwise
            % ...
    end
end
fprintf('Numero de vocales: %g\n\n',k);

```

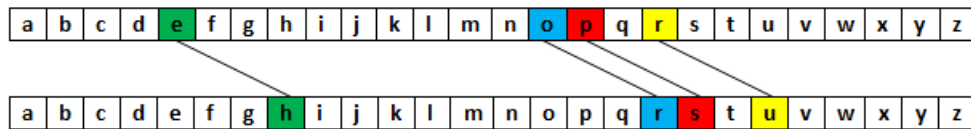
### 3.4 Ordenar palabras

Escriba un programa que ordene las palabras contenidas en una cadena de caracteres, imprimiéndolas en pantalla.

Solución.

### 3.5 Cifrado básico

El método de cifrado más simple es el cifrado por desplazamiento, que consiste en remplazar un caracter por otro ubicado **n** posiciones a la derecha en la tabla de código ASCII. Por ejemplo, si tenemos la palabra **perro**, y queremos cifrarla desplazando tres lugares obtendríamos: **shuur**. En la figura siguiente se muestra un esquema de lo anterior.



Desarrolle un programa que tenga como entrada una cadena de caracteres y devuelva en pantalla la cadena cifrada mediante el desplazamiento por tres posiciones.

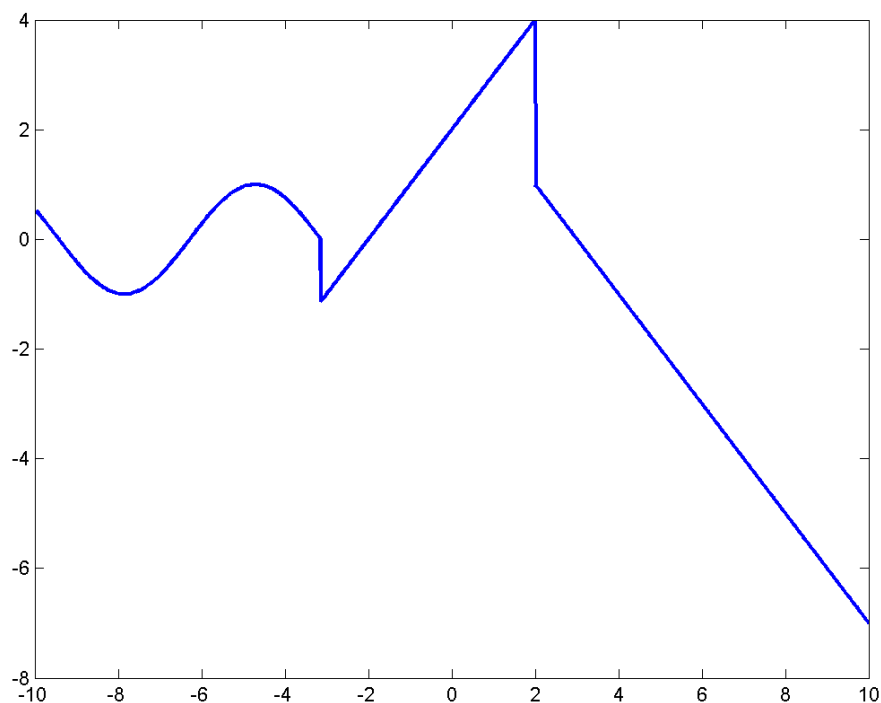
## 4. Gráficas

### 4.1 Gráfica de una función a trozos

Escriba un programa que grafique la siguiente función a trozos:

$$f(x) = \begin{cases} \sin(x) & x < -\pi \\ x + 2 & -\pi \leq x \leq 2 \\ -x + 3 & x > 2 \end{cases}$$

```
x=-10:0.01:10;  
y1=sin(x(x<-pi));  
y2=x(x>=-pi & x<=2)+2;  
y3=-(x(x>2))+3;  
plot(x,[y1 y2 y3], 'linewidth',2);
```



### 4.2 Gráfica de una desigualdad (inecuación)

MATLAB no proporciona de manera nativa una función para trazar gráficas de inecuaciones,

por ello el objetivo de este ejercicio es desarrollar una función llamada *inecgraf*, cuyos argumentos de entrada sean: la inecuación dada como una cadena de caracteres y el intervalo en el cual se graficará (vector de dos elementos).

Solución.

```
function inecgraf(I,R)
% Grafica una desigualdad (inecuacion) en un rango
% especificado.
%
% Argumentos de entrada:
%         I   -   Inecuacion
%         R   -   Rango en el cual se trazara la
%                 grafica.
%
set(gca,'NextPlot','add');
axis([R(1) R(2) R(1) R(2)]);
dd=(R(2)-R(1))/50;
[x,y]=meshgrid(R(1):dd:R(2));
[f,c]=find(eval(I));
h=zeros(1,length(f));
for i=1:length(f)
    h(i)=plot(x(f(i)),c(i)),y(f(i),c(i)),'b*','MarkerSize',2);
end
end
```

### 4.3 Gráfica de una esfera

Desarrolle una función *esfera* cuyos parámetros de entrada sean: el radio, y las coordenadas  $x, y, z$  del centro. Con los datos anteriores trace una esfera, apoyándose en la función *sphere* nativa de MATLAB.

Solución.

# 5. Matemáticas

## 5.1 Ecuación de la recta que pasa por dos puntos

Desarrolle un programa cuyos valores de entrada sean las coordenadas  $(x_1, y_1)$  y  $(x_2, y_2)$  de dos puntos y que devuelva en pantalla la ecuación de la recta que pasa por estos puntos.

```
x1=input('x1: ');
y1=input('y1: ');
x2=input('x2: ');
y2=input('y2: ');
m=(y2-y1)/(x2-x1);
b=(-x1*m)+y1;
if b<0
    signo='- ';
elseif b>0
    signo='+ ';
else
    signo='';
    b='';
end
if m==1
    m='';
end
y=['y = ',num2str(m),' x ',signo,num2str(abs(b))];
fprintf('\nEcuacion de la recta: %s\n\n',y);
```

## 6. Interfaces gráficas de usuario

Comunmente se denomina interfaz gráfica de usuario (GUI) a un conjunto de objetos gráficos mostrados en una ventana, que el usuario de un ordenador utiliza para interactuar con el sistema operativo o el software en ejecución. En MATLAB las GUI normalmente incluyen botones, campos de texto, check boxes, menús desplegables, list boxes, tablas, entre otros que facilitan la interacción del usuario. Cada uno de estos elementos gráficos es programado para responder a una determinada acción realizada por el usuario.

### 6.1 El hola mundo en GUI

*Desarrolle una interfaz gráfica que contenga un botón, el cual al ser presionado por el usuario deberá responder mostrando un cuadro de dialogo (msgbox) con la clásica cadena hola mundo.*

Solución.

```
function HolaMundo
figure('MenuBar','None',...
    'NumberTitle','off',...
    'Name','Hola Mundo');

uicontrol('style','push',...
    'String','Botón',...
    'Callback','msgbox(''Hola mundo'')');
end
```

### 6.2 Ventana multicolor

*Programa una interfaz gráfica que no contenga control gráfico alguno, y que solamente cambie de color a cada cierto tiempo, esto hasta que el usuario cierre la ventana correspondiente.*

Solución.

```
function CambiaColor
f = figure('MenuBar','none',...
    'NumberTitle','off',...
    'Name','Cambia Color');

while ishandle(f)
    set(f,'Color',rand(1,3));
    pause(0.5);
    drawnow;
end
```

```
end
```

### 6.3 Lista de archivos M.

*Desarrolle una GUI con un List Box, el cual debe contener una lista de los nombres de archivos \*.m ubicados en el mismo directorio. Al presionar cada uno de los elementos de la lista debe abrir el archivo seleccionado en el editor de MATLAB.*

**Solución.**

```
function ListaArchivosM
figure('MenuBar','none',...
    'NumberTitle','off',...
    'Name','Lista Archivos',...
    'Position',[0 0 200 300]);
centerfig();

archivos_m = dir('*.m');
archivos_m = struct2cell(archivos_m);
uicontrol('style','listbox',...
    'String',archivos_m(1,:),...
    'Units','Normalized',...
    'Position',[0.02 0.02 0.96 0.96],...
    'Callback',@edit_call);

function edit_call(src,~)
    str = get(src,'String');
    k = get(src,'Value');
    edit(str{k});
end
end
```

### 6.4 Mini Calculadora

*Diseñe y desarrolle una interfaz gráfica de usuario que asemeje el comportamiento de una calculadora muy sencilla. Debe contener cuatro botones correspondientes a los operadores aritméticos básicos, dos campos editables que permitan insertar los datos de entrada y otro campo estático o editable que permita mostrar el resultado de la operación realizada.*

**Solución.**

```
function MiniCalculadora
figure('MenuBar','None',...
    'NumberTitle','off',...
    'Name','Mini Calculadora',...
    'Resize','off',...
    'Position',[0 0 300 150]);
centerfig();
```

```

% ===== DATOS =====
panel_datos = uipanel('Units','Pixels',...
    'Position',[10 50 280 95]);

uicontrol(panel_datos,'Style','text',...
    'String','# 1',...
    'Units','Normalized',...
    'Position',[0 0.67 0.4 0.25]);
hN1=uicontrol(panel_datos,'Style','edit',...
    'String','',...
    'Units','Normalized',...
    'Position',[0.45 0.72 0.5 0.25]);

uicontrol(panel_datos,'Style','text',...
    'String','# 2',...
    'Units','Normalized',...
    'Position',[0 0.33 0.4 0.25]);
hN2=uicontrol(panel_datos,'Style','edit',...
    'String','',...
    'Units','Normalized',...
    'Position',[0.45 0.38 0.5 0.25]);

uicontrol(panel_datos,'Style','text',...
    'String','Resultado',...
    'Units','Normalized',...
    'Position',[0 0 0.4 0.25]);
hR=uicontrol(panel_datos,'Style','edit',...
    'String','',...
    'Units','Normalized',...
    'Position',[0.45 0.05 0.5 0.25],...
    'BackG',ones(1,3)*0.8);

% ===== BOTONES OPERADORES =====
panel_botones = uipanel('Units','Pixels',...
    'Position',[10 5 280 40]);

OPERADORES = '+-*/';

for k = 1:length(OPERADORES)
    uicontrol(panel_botones,'style','push',...
        'String',OPERADORES(k),...
        'Units','normalized',...
        'Position',[(k-1)*(1/4) 0 1/4 1],...
        'FontSize',16,...
        'FontWeight','bold',...
        'Callback',@calcular);
end

function calcular(src,~)
    n1 = get(hN1,'String'); % Primer número
    n2 = get(hN2,'String'); % Segundo número
    oper = get(src,'String'); % Operador
    set(hR,'String',num2str(eval([n1,oper,n2])));

```



```

    end

end

```

## 6.5 Visor de imágenes (Nivel I)

Desarrolle una GUI que funcione como un visor de imágenes simple, el cual debe contener un menú Archivo y dentro de este dos sub-menús llamados Abrir y Salir. La opción Abrir debe mostrar al usuario un explorador de archivos interactivo (`uigetfile`) que le permita seleccionar una imagen en formato PNG y enseguida mostrarla en un axes ubicado dentro la misma GUI (utilice las funciones `imread` e `imshow` para la manipulación de la imagen). La opción Salir, en este caso, es auto-descriptiva.

Solución.

```

function AbrirImagen
f = figure('MenuBar','None',...
    'NumberTitle','off',...
    'Name','Abrir imagen');

% ===== MENÚ =====
hMenu = uimenu(f,'Label','Archivo');
uimenu(hMenu,'Label','Abrir','Callback',@abrir_img);
uimenu(hMenu,'Label','Salir','Callback','close(gcf)');

% ===== AXES =====
ax = axes('Units','Normalized',...
    'Position',[0 0 1 1],...
    'Visible','off');

function abrir_img(~,~)
    [filename, pathname] = uigetfile('*.PNG', 'Seleccione una imagen');
    if isequal(filename,0) || isequal(pathname,0)
        return;
    else
        X = imread(fullfile(pathname,filename));
        imshow(X);
    end
end
end

```

# 7. POO

## 7.1 La clase Persona

*Defina una clase llamada Persona, cuyos atributos serán nombre y edad, mismos que serán pasados como argumento en el constructor de la clase. Además, la clase debe contener un método crecer que recibirá como argumento un entero positivo que indicará los años que crecerá el objeto, desde luego modificando el atributo edad.*

Solución.

```
classdef Persona < handle

    properties
        nombre;
        edad;
    end

    methods
        function obj = Persona(nombre,edad)
            obj.nombre = nombre;
            obj.edad = edad;
        end

        function crecer(obj,anios)
            obj.edad = obj.edad + anios;
        end
    end
end
```

## 7.2 Clase Math (Métodos estáticos)

*Desarrolle una clase Math que contenga los atributos constantes  $E$  (constante  $e$ ) y  $PI$  (constante  $\pi$ ), y los métodos estáticos sumar, multiplicar, redondear, mayor y menor.*

Solución.

```
classdef Math

    % Clase Math

    properties (Constant = true)
        % Atributos constantes
    end
end
```

```
    PI = pi;
    E = exp(1);
end

methods (Static)
    % Métodos estáticos
    function r = sumar(a,b)
        r = a + b;
    end

    function r = multiplicar(a,b)
        r = a * b;
    end

    function r = redondear(n)
        r = round(n);
    end

    function r = mayor(a,b)
        r = a;
        if b > a
            r = b;
        end
    end

    function r = menor(a,b)
        r = a;
        if b < a
            r = b;
        end
    end
end
end
```