Estructuras de control

Instrucciones de entrada (lectura) y de salida (escritura)

Se verá a continuación una forma sencilla de leer variables desde teclado y escribir mensajes en la pantalla del PC. Más adelante se considerarán otros modos más generales y complejos de hacerlo.

- Función input

  Esta permite imprimir un mensaje en la línea de comandos de MATLAB y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario.

  >> x = input('texto')

  Entre comillas debe escribirse el texto que queremos que lea.

```
u = input('var ')
var   7
u =
     7
```

```
u = input('var ')
var   [ 1 2 3 ]
u =
     1     2     3
```

```
u = input('var ')
var   'esto'
u =
esto
```

```
>> x = input('texto‘, ‘s’)      →   u = input('var ‘,’s’)
                                    var   esto es
                                    u =
                                    esto es
```

El texto tecleado se lee y se almacena como texto sin evaluar

Estructuras de control

Instrucciones de entrada (lectura) y de salida (escritura) (cont.)

- Función disp

  Esta permite imprimir en pantalla un mensaje de texto o el valor de una variable, pero sin imprimir su nombre

  >> disp(‘el programa ha terminado’)

  >> a = rand(4,4);

  >> disp(a);

0.9501	0.8913	0.8214	0.9218
0.2311	0.7621	0.4447	0.7382
0.6068	0.4565	0.6154	0.1763
0.4860	0.0185	0.7919	0.4057

Resolución de problemas

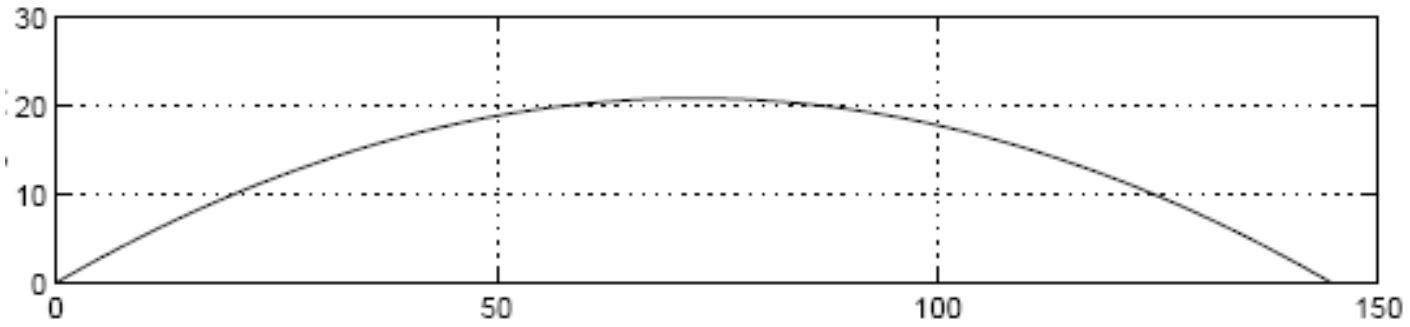
El proceso de resolución de un problema tiene las etapas siguientes:

- Definición del problema.
- Creación del modelo matemático.
- Desarrollo de un modelo computacional para la resolución del problema. (Analizar y dividir el modelo matemático en una serie de pasos que conduzcan a la solución, es decir diseñar un algoritmo)
- Implementación del modelo computacional.
- Realización de pruebas verificando que funciona en todos los casos previstos en el diseño.
- Determinación de la solución del problema

Resolución de problemas

Ejemplo:

Un objeto pequeño se lanza desde el piso a una velocidad de 50 millas por hora con un ángulo de 30 grados respecto al piso. Determinar el tiempo de vuelo y la distancia recorrida cuando el objeto alcanza de nuevo el piso.



Resolución de problemas

Definición del problema. Del enunciado se requiere información adicional:

- Propiedades del objeto y del medio pudieran afectar la trayectoria de vuelo. Por ejemplo si el objeto es liviano en peso, tiene una superficie relativamente grande, y viaja en el aire, entonces la resistencia del aire pudiera afectar su vuelo. Si el aire se mueve (vientos fuertes) también el vuelo del objeto puede verse afectado. Si esta información no esta disponible, será necesario asumir que el aire no afecta la trayectoria de vuelo.
- La aceleración de la gravedad también afecta el vuelo. Si no hay información adicional disponible, es razonable asumir que la aceleración de la gravedad en la superficie es la que aplica.
- La precisión de la velocidad inicial y el ángulo de salida son necesarias para determinar las cantidades a ser determinadas.
- Unidades de conversión podrían ser requeridas. Por ejemplo, millas a pies, horas a segundo, grados a radianes, etc.
- Las unidades de los resultados no fueron especificadas, pero podemos tomar segundos para el tiempo y pies para la distancia.
- La teoría a ser aplicada es el movimiento balístico en dos dimensiones.

Resolución de problemas

Modelo matemático.

Definición de la notación:

- Tiempo: t (seg.) con t=0 cuando el objeto es lanzado.
- Velocidad inicial: v = 50 millas por hora
- Ángulo inicial: θ = 30 grados
- Posición horizontal del objeto: x(t) (pies)
- Posición vertical del objeto: y(t) (pies)
- Aceleración de la gravedad: g = 32.2 pies/seg2, dirigida en la dirección de y negativo.

1 milla es 5280 pies
1 hora es 3600 segundos

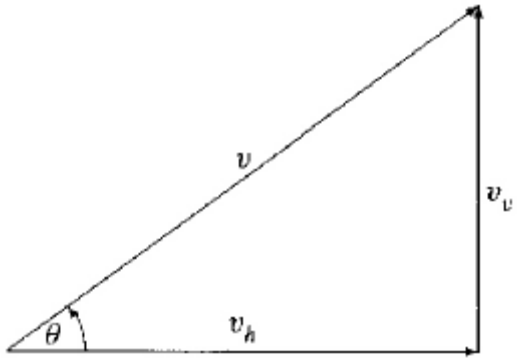
La clave del desarrollo del modelo matemático es descomponer la trayectoria en sus componentes horizontal y vertical.

Resolución de problemas

Modelo matemático (cont.).

La velocidad inicial *v* también puede ser descompuesta de esta manera,

$$v_h = v\cos\theta$$
$$v_v = v\sin\theta$$



La posición vertical y horizontal puede ser determinada en función de t. Como no hay fuerzas externas que retarden el movimiento horizontal, el objeto se mueve a velocidad constante en la dirección horizontal:

$$x(t) = v\,t\cos\theta$$

En la dirección vertical, el movimiento del objeto esta afectado por la gravedad y su posición se calcula como:

$$y(t) = v\,t\sin\theta - \frac{1}{2}g\,t^2$$

Resolución de problemas

Modelo computacional.

Usando el modelo anterior, el objeto alcanza el piso cuando su posición vertical es cero, es decir,

$$y(t) = v\,t\sin\theta - \frac{1}{2}\,g\,t^2 = 0$$

de donde, se tienen 2 soluciones

$$t = 0 \quad y \quad t = \frac{2v\sin\theta}{g}$$

La segunda indica que el tiempo de vuelo es

$$t_p = \frac{2v\sin\theta}{g}$$

y la posición horizontal a este tiempo es

$$x(t_p) = v\,t_p\cos\theta$$

Resolución de problemas

Implementación del modelo computacional.

Las ecuaciones definidas en el modelo computacional pueden ser inmediatamente implementadas usando MATLAB:

trayectoria.m

Editor - C:\Users\CSC-UCAB\Desktop\Saul-DATOS\cursos\CO2111\trayectoria.m

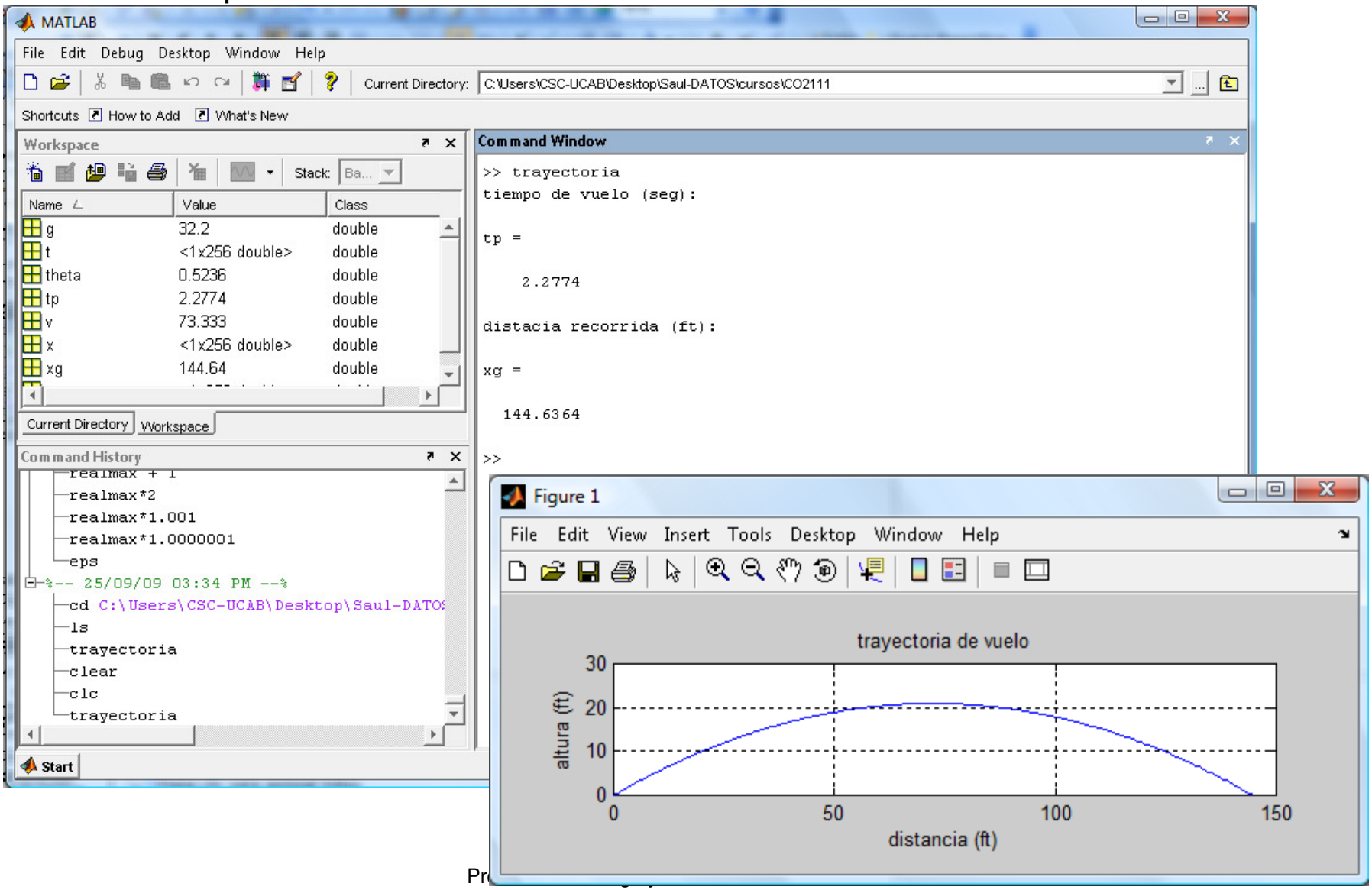
File Edit Text Desktop Window Help

```
1  % Calculo de la trayectoria de vuelo
2  %
3  % valores iniciales
4  g = 32.2;   % gravedad, ft/seg^2
5  v = 50 * 5280/3600;   % velocidad de lanzamiento, ft/seg
6  theta = 30 * pi/180;   % angulo de lanzamiento, radianes
7
8  % Calculo de los resultados
9  disp('tiempo de vuelo (seg):')   % etiqueta para el tiempo de vuelo
10 tp = 2 * v * sin(theta)/g   % tiempo de regreso al piso, seg
11 disp('distacia recorrida (ft):')   % etiqueta para la distancia
12 xg = v * cos(theta) * tp   % distancia recorrida
13
14 % calculo y ploteo de la trayectoria de vuelo
15 t = linspace(0,tp,256);
16 x = v * cos(theta) * t;
17 y = v * sin(theta) * t - g/2 * t.^2;
18 plot(x,y), axis equal, axis([ 0 150 0 30 ]), grid, ...
19 xlabel('distancia (ft)'), ylabel('altura (ft)'), title('trayectoria de vuelo')
```

script Ln 9 Col 68 OVR

Resolución de problemas

Solución del problema.



Estructuras de control

Selectivas

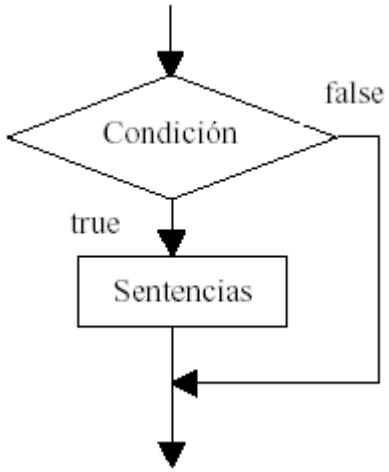
Permiten realizar una u otra operación según se cumpla o no una determinada condición.

- Selección simple

  if (condición)

  sentencia o bloque de instrucciones

  end



- Selección múltiple

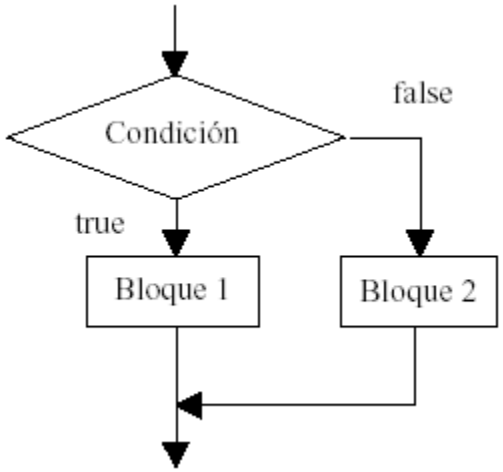
  if (condición)

  sentencia 1 o bloque de instrucciones 1

  else

  sentencia 2 o bloque de instrucciones 2

  end



Estructuras de control

Ejemplo.

```
if interval < 1
   xinc = interval/10;
else
   xinc = 0.1;
end
```

Ejemplo. Cálculo del factorial de un número entero positivo

```
1  n = input('ingrese un numero  ');
2
3  % Verifica si el parámetro n puede ser procesado
4  if (round(n) ~= n) | (n < 0)
5      facto = -1;
6      disp('Se esperaba un número entero mayor o igual que 0');
7  else
8      if (n == 0) | (n == 1)
9         facto = 1;
10     else
11        facto = prod(2:n);
12     end
13     disp('el factorial del numero es');
14     disp(facto);
15 end
```

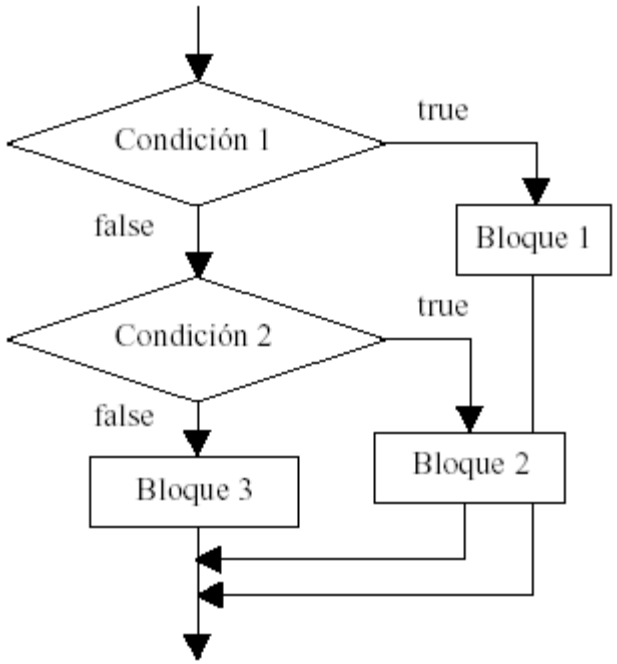

archivo : factorial_numero.m

Estructuras de control

Selectivas

- Selección múltiple (cont.)

```
if (condición 1)
        sentencia 1 o bloque de instrucciones 1
elseif (condición 2)
        sentencia 2 o bloque de instrucciones 2
else
        sentencia 3 o bloque de instrucciones 3
end
```



Obs. Permiten realizar una u otra operación según se cumpla o no una determinada condición.

archivo : notas_usb.m

Obs: La condición del "if" puede ser una condición matricial.

archivo : Condicion_Matricial.m

Estructuras de control

Ejemplo: decidir el funcionamiento de un equipo

```
if temperature > 100
   disp('Too hot - equipment malfunctioning.')
elseif temperature > 90
   disp('Normal operating range.')
elseif temperature > 50
   disp('Below desired operating range.')
else
   disp('Too cold - turn off equipment.')
end
```

Ejemplo: decidir si una variable almacena un escalar, un vector o una matriz

```
1  x = input('variable ');
2  [m,n] = size(x);
3  if(m==n & n==1)
4      disp('es un escalar')
5  elseif(m == 1  | n== 1)
6      disp('es un vector');
7  else
8      disp('es una matriz')
9  end
10
```

```
>> a=2; b=[2 3]; c=[4 5; 6 7];
```

archivo : tipo.m

Estructuras de control

Selectivas

- Switch

  Esta realiza una función análoga a un conjunto de “if … elseif …” concatenados.

```
switch  expresión
case  valor1
      sentencia 1 o bloque de
      instrucciones 1
case { valor2 , valor3 , valor4 }
      sentencia 2 o bloque de
      instrucciones 2
otherwise
      sentencia 3 o bloque de
      instrucciones 3
end
```

Obs. No acepta el uso de matrices, sólo acepta escalares

Ejemplo:

```
switch n
case  -1
      disp(‘número -1');
case  0
      disp(‘cero');
case  1
      disp(‘número +1');
otherwise
      disp('otro valor');
end
```

Estructuras de control

Ejemplos.

Conversión de unidades (meter a inches, feet, meter, centimeter, milimeter)

```
 1  % Conversión de unidades (meter a inches, feet,
 2  %   meter, centimeter, milimeter)
 3  x = input('valor en metros ');
 4  units = input('a que unidades desea transformar ');
 5  switch units
 6      case{'inch','in'}
 7              y = x*39.3696;
 8      case{'feet','ft'}
 9              y = x*3.2808;
10      case{'meter', 'm'}
11              y = x;
12      case{'centimeter', 'cm'}
13              y = x*100;
14      case{'milimeter', 'mm'}
15              y = x*1000;
16      otherwise
17              disp(['desconocido  ' units])
18              y = NaN;
19  end
20  disp(y)
```

archivo : unidades.m

Estructuras de control

Repetitivas

Permiten repetir las mismas operaciones o análogas sobre datos distintos.

- For

  Repite un conjunto de sentencias un número predeterminado de veces.

```
for  i = 1 : n
      sentencia  o bloque de
      instrucciones
end
```

```
for  i = n : -0.1 : 1
      sentencia  o bloque de
      instrucciones
end
```

```
for  i = vector de valores
      sentencia  o bloque de
      instrucciones
end
```

```
for  i = 1 : n
      for  j = 1 : m
            sentencia  o bloque
            de instrucciones
      end
end
```

"for" anidado

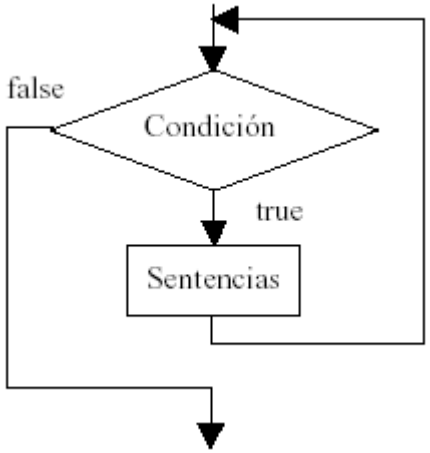
Estructuras de control

Repetitivas

- While

  Repite un conjunto de sentencias mientras haya elementos distintos de cero en condición, es decir, mientras haya algún o algunos elementos “true”. El bucle se termina cuando todos los elementos de condición son “false” (es decir, cero).

```
while  condición
        sentencia  o bloque de
        instrucciones
end
```



Observación:

Break : hace que se termine la ejecución del bucle más interno de los que comprenden a dicha sentencia.

Continue : hace que se pase inmediatamente a la siguiente iteración del bucle for o while, saltando todas las sentencias que hay entre el continue y el fin del bucle en la iteración actual.

Estructuras de control

Ejemplo

Calculo del valor de la variable “eps” de MATLAB:
eps es el menor número que se puede sumar a 1 tal que el resultado es mas grande que 1 usando precisión finita

```
% Calculo del epsilon (eps) de la maquina
format long;
format compact;
num = 0;
EPS = 1;
while (1+EPS>1)
    EPS = EPS/2;
    num = num + 1;
end
disp('epsilon de la maquina');
EPS = 2 * EPS
disp('exponente correspondiente')
num = num - 1; disp(num);
format short;
```

```
num =
    52
EPS =
  2.2204e-016
```

Obs. MATLAB trabaja en precisión doble

archivo : Calc_eps.m (def lamina 29), también ver Calc_RealMax.m

Estructuras de control

Ejemplo: producto de matrices, uso del “for” y del “if”

```
1  clear;  clc;
2  a = input('primera matriz  ');
3  b = input('segunda matriz  ');
4  [m1,n1] = size(a);
5  [m2,n2] = size(b);
6  if ( n1 == m2 )
7      for i = 1:m1
8          for k = 1:n2
9              sum = 0;
10             for j = 1:n1
11                 sum = sum + a(i,j)*b(j,k);
12             end
13             c(i,k) = sum;
14         end
15     end
16     disp('la matriz producto es')
17     disp(c);
18 else
19     disp(['dimensiones de las matrices ' ...
20       'no compatibles para el producto']);
21 end
```

```
a =
    0.4451    0.4186    0.2026    0.0196
    0.9318    0.8462    0.6721    0.6813
    0.4660    0.5252    0.8381    0.3795

b =
    0.8318    0.3046    0.3028    0.3784    0.4966
    0.5028    0.1897    0.5417    0.8600    0.8998
    0.7095    0.1934    0.1509    0.8537    0.8216
    0.4289    0.6822    0.6979    0.5936    0.6449

a * b =
    0.7329    0.2676    0.4058    0.7131    0.7769
    1.9696    1.0391    1.3174    2.0585    2.2157
    1.4090    0.6626    0.8168    1.5687    1.6373
```

archivo : multiplicacion.m

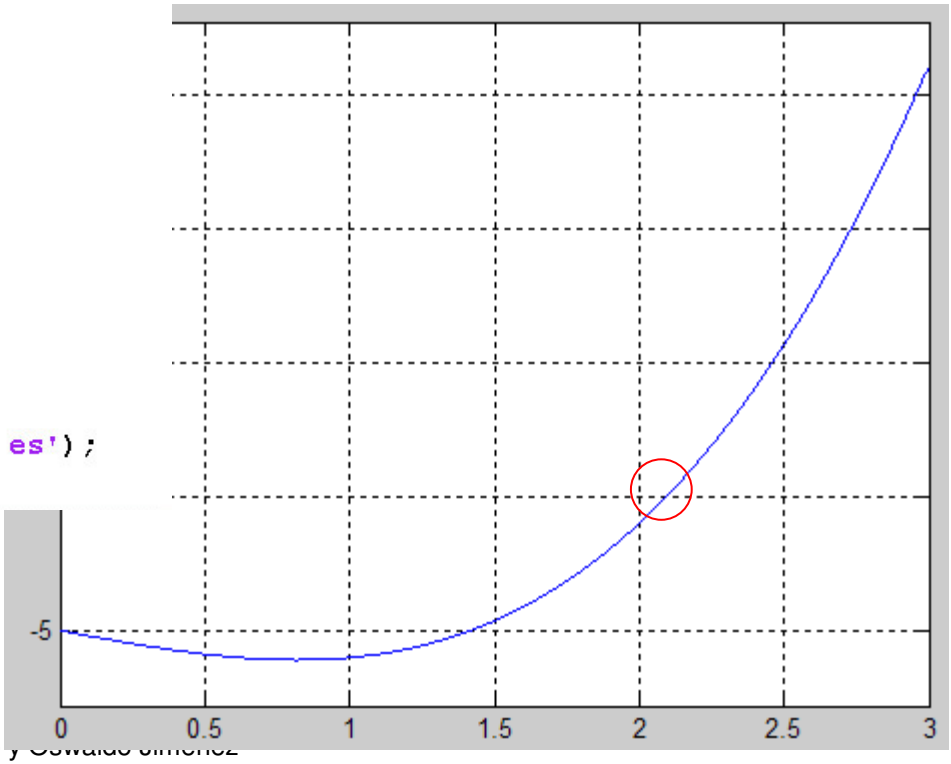
Estructuras de control

Ejemplo: calculo de la raíz de $x^3 - 2x - 5$ en [0,3], usando el método de bisección

```
1  % Calculo de una raiz del polinomio  x^3-2*x-5 en
2  % el intervalo [0,3]
3
4  tol = 1.e-8;
5
6  a = 0; fa = a^3-2*a-5;
7  b = 3; fb = b^3-2*b-5;
8
9  while b-a > tol*b
10     x = (a+b) / 2;
11     fx = x^3-2*x-5;
12     if sign(fx) == sign(fa)
13         a = x;
14         fa = fx;
15     else
16         b = x;
17         fb = fx;
18     end
19 end
20
21 disp(' ');
22 disp('La raiz del polinomio x^3-2*x-5 en [0,3] es');
23 disp(x);
```

archivo: raiz_polinomio.m

x = 2.09455148154233



Estructuras de control

Ejemplo: Calcular el MCD de 2 números naturales

Dados dos números naturales x e y, el proceso para determinar el máximo común divisor (MCD) se define recursivamente por

$$x_{n+1} = y_n$$

$$y_{n+1} = x_n - y_n \left[\frac{x_n}{y_n} \right]$$

para $n = 0,1,2,\cdots$

donde $x_0 = x, \; y_0 = y$

El proceso se detiene cuando $y_n = 0$, en cuyo caso x_n es el divisor o factor requerido.

```
% MCD de 2 numeros naturales
clear
a=input('Introduce el primer numero natural (>0) ');
b=input('Introduce el segundo numero natural (>0) ');
if(a<=0 | b<=0)
  if(a==0 & b==0)
     disp('el MCD no esta definido cuando ambos numeros son cero')
  else
     disp('hay numeros menores o iguales a cero')
  end
else
  x = a;
  y = b;
  while (y ~= 0)
     z = x - y * floor(x/y);
     x = y;
     y = z;
  end
  disp('el MCD es');
  disp(x);
end

```

archivo: maximo_comun_divisor.m

Si x = 24, y = 28

x1 = 28, y1 = 24 – 28 [24/28] = 24

x2 = 24, y2 = 28 – 24 [28/24] = 4

x3 = 4, y3 = 24 – 4 [24/4] = 0

MCD(24,28) = 4

También ver minimo_comun_multiplo.m

Estructuras de control

Ejemplo:
Determinar si una matriz cuadrada de dimensión > 2 es mágica

```
% Programa que verifica si una matriz dada es o no magica
format compact;
% Lee la matriz por pantalla
A = input('Introduzca la matriz ');
% Verifica si la matriz es cuadrada. De no serlo, le da
% al usuario una segunda oportunidad de lectura
[n,m] = size(A);
if n ~= m
    disp('La matriz no es cuadrada. Intente de nuevo '); disp(' ');
    A = input('Introduzca la matriz ');
    [n,m] = size(A);
    if n ~= m
        disp('La matriz no es cuadrada. Au revoir ...'); return;
    end
end
% Crea el vector sumas que contiene las sumas de los
% elementos de cada fila y cada columna de la matriz
sumas = [];
for k=1:n
    sumas = [sumas sum(A(k,:)) sum(A(:,k))];
end
% se incluye las 2 diagonales de la matriz
sumas = [sumas sum(diag(A)) sum(diag(A(:,n:-1:1)))];
% Usando el comando unique, verifica si todas las
% entradas del vector sumas son iguales
if length(unique(sumas)) == 1
    disp('La matriz dada SI es magica')
else
    disp('La matriz dada NO es magica')
end
```

archivo: esMagica.m

Estructuras de control

Ejemplo: Implementación del algoritmo Criba de Eratóstenes para hallar todos los números primos comprendidos entre 2 y un entero positivo N, mayor que 2

```
% Este programa implementa el algoritmo Criba de Eratóstenes
% para hallar todos los números primos comprendidos entre 2
% y un entero positivo N, mayor que 2, el cual es leído por
% pantalla

% Lee un número entero N > 2 por pantalla
N = input('Dé un entero > 2: ');
[n,m] = size(N);

% Verificación de la consistencia del dato ingresado por el usuario
if round(N) ~= N | ~isnumeric(N) | N <= 2 | n ~= 1 | m ~= 1
    disp('Dato incorrecto, ciao ... ');
    return
end

% Se inicializa el arreglo de números primos en la secuencia
% de números impares entre 3 y N. Esto así porque sabemos que
% los números pares distintos de 2 NO son primos
primos = 3:2:N;
tam_primos = length(primos);

% Se eliminan los múltiplos impares de todos los números
% declarados como primos (ya que sabemos que los múltiplos
% pares de dichos números no están en el arreglo 'primos')
for primo_actual = 3:2:sqrt(N)
    if primos((primo_actual-1)/2)
        primos((primo_actual^2-1)/2 : primo_actual : tam_primos) = 0;
    end
end

% Finalmente, se agrega el 2 al comienzo de la lista de
% números primos
primos = [2 primos(find(primos ~= 0))];

disp('Los numeros primos entre 2 y ');
disp(N);
disp(' son: ');
disp(primos);
```

archivo: criba_de_eratostenes.m