Programación con Octave/Octave-UPM (II)

Mario Bermejo

CLUB DE INFORMÁTICA CAMINOS

18 diciembre 2014

Índice

Características Matlab/Octave

Características Matlab/Octave

- Tipo de lenguaje: interpretado
- Tipo de lenguaje: alto nivel
- Tipo de lenguaje: científico y/o ingenieril
- Tipo de lenguaje: estructurado
- No es necesario declarar variables

Matlab

- Propietario
- Mejor interfaz gráfica

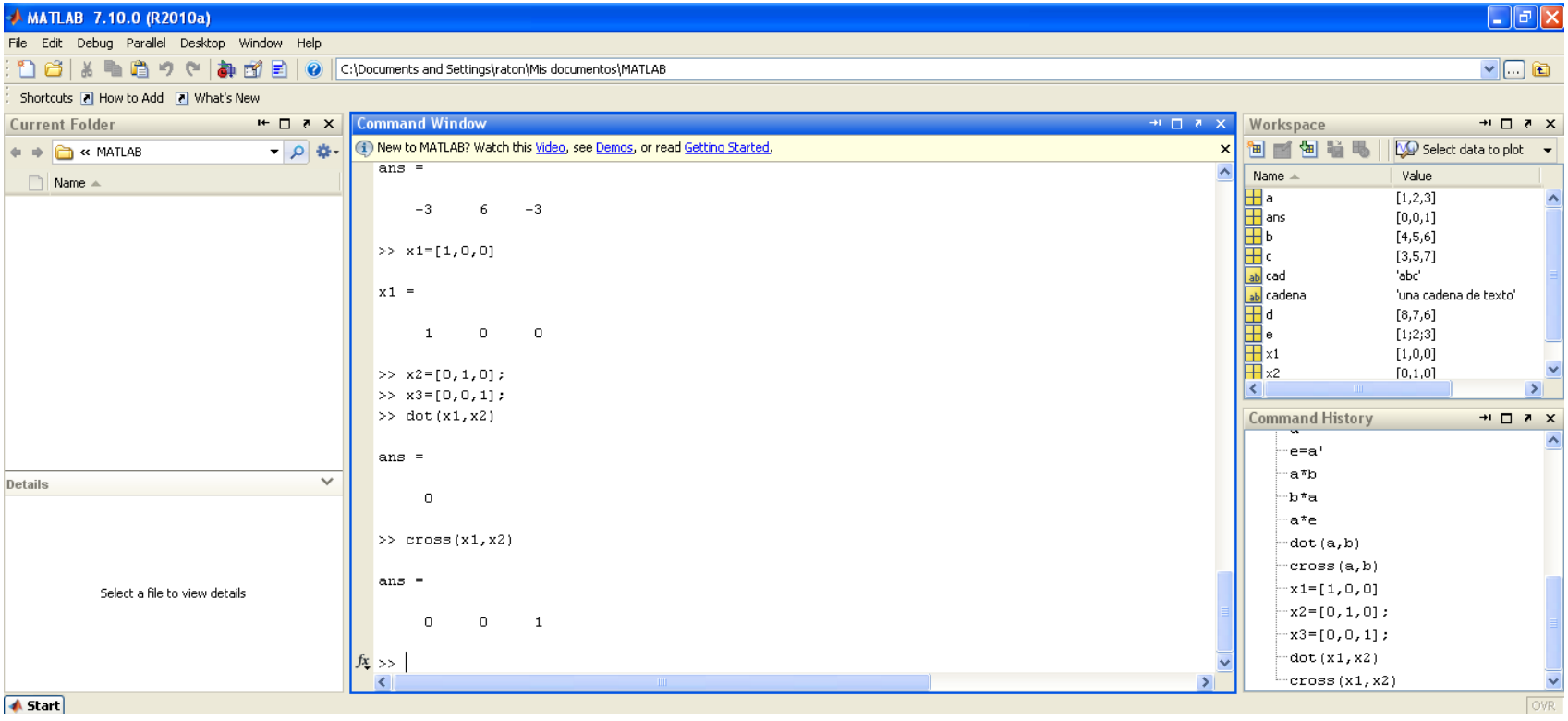
Octave

- Libre (GPL)
- Interfaces gráficas en desarrollo: qtOctave, UPM-Octave

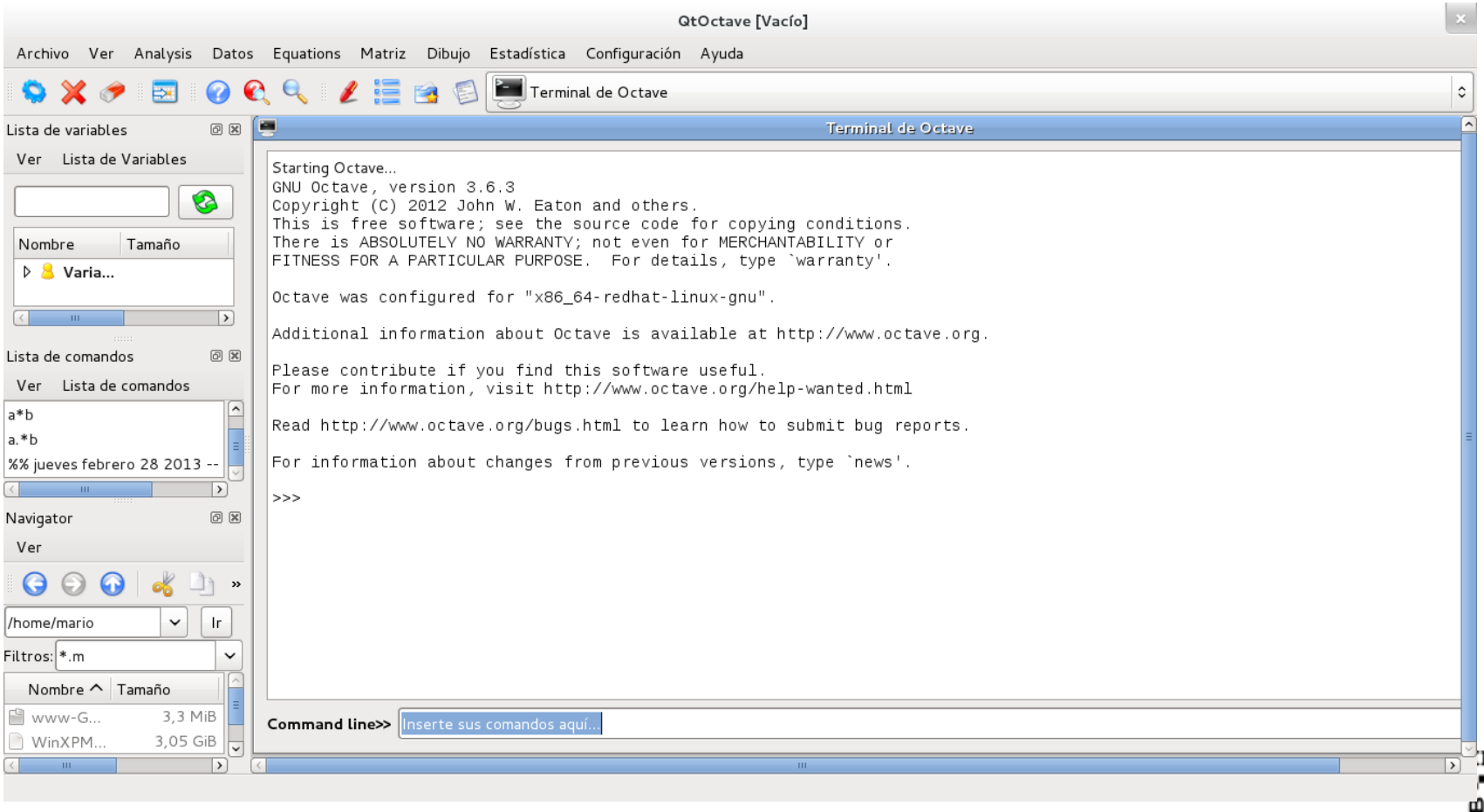
Octave en consola

```
GNU Octave, version 3.6.4
Copyright (C) 2013 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "x86_64-redhat-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type `news'.

octave:1> a = [1 0 0];
octave:2> b = [0 1 0];
octave:3> cross(a, b);
octave:4> cross(a, b)
ans =

   0   0   1

octave:5>
```

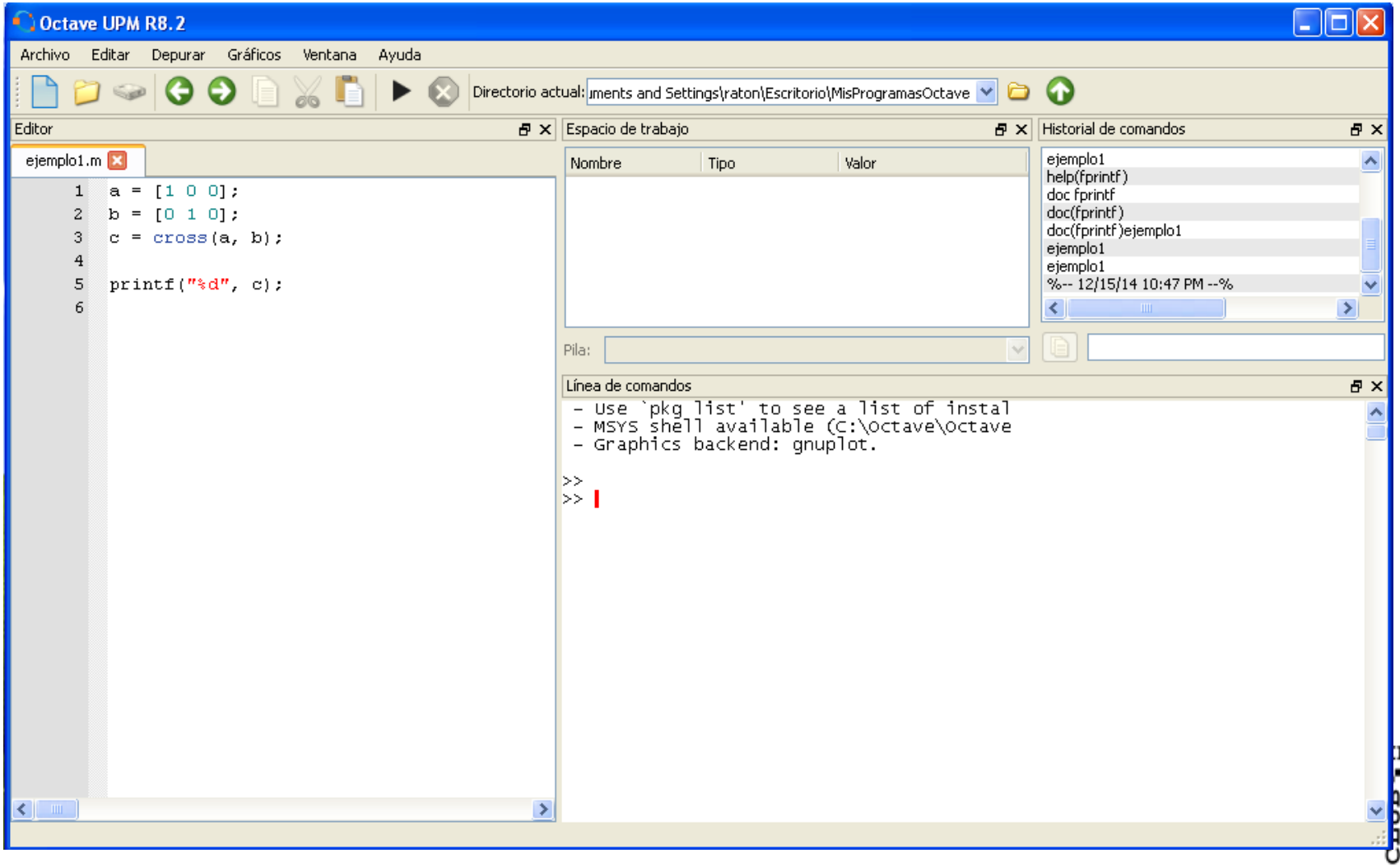
Matlab


MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
C:\Documents and Settings\raton\Mis documentos\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB
Name
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
ans =
-3 6 -3
>> x1=[1,0,0]
x1 =
1 0 0
>> x2=[0,1,0];
>> x3=[0,0,1];
>> dot(x1,x2)
ans =
0
>> cross(x1,x2)
ans =
0 0 1
>>
Workspace
Select data to plot
Name Value
a [1,2,3]
ans [0,0,1]
b [4,5,6]
c [3,5,7]
cad 'abc'
cadena 'una cadena de texto'
d [8,7,6]
e [1;2;3]
x1 [1,0,0]
x2 [0,1,0]
Command History
e=a'
a*b
b*a
a*e
dot(a,b)
cross(a,b)
x1=[1,0,0]
x2=[0,1,0];
x3=[0,0,1];
dot(x1,x2)
cross(x1,x2)
Start
OVR

qtOctave


QtOctave [Vacío]
Archivo Ver Analysis Datos Equations Matriz Dibujo Estadística Configuración Ayuda
Terminal de Octave
Lista de variables
Ver Lista de Variables
Nombre Tamaño
Varia...
Lista de comandos
Ver Lista de comandos
a*b
a.*b
%% jueves febrero 28 2013 --
Navigator
Ver
/home/mario
Ir
Filtros: *.m
Nombre Tamaño
www-G... 3,3 MiB
WinXPM... 3,05 GiB
Terminal de Octave
Starting Octave...
GNU Octave, version 3.6.3
Copyright (C) 2012 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type `warranty'.
Octave was configured for "x86_64-redhat-linux-gnu".
Additional information about Octave is available at http://www.octave.org.
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html
Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type `news'.
>>>
Command line>>
Inserte sus comandos aquí...

Octave UPM


Octave UPM R8.2
Archivo Editar Depurar Gráficos Ventana Ayuda
Directorio actual: uments and Settings\raton\Escritorio\MisProgramasOctave
Editor
ejemplo1.m
1 a = [1 0 0];
2 b = [0 1 0];
3 c = cross(a, b);
4
5 printf("%d", c);
6
Espacio de trabajo
Nombre Tipo Valor
Pila:
Historial de comandos
ejemplo1
help(fprintf)
doc fprintf
doc(fprintf)
doc(fprintf)ejemplo1
ejemplo1
ejemplo1
%-- 12/15/14 10:47 PM --%
Línea de comandos
- Use `pkg list' to see a list of instal
- MSYS shell available (C:\Octave\Octave
- Graphics backend: gnuplot.
>>
>>

Ventanas de la intefaz de Octave UPM

- Editor
- Linea de comandos
- Historial de comandos
- Espacio de trabajo (variables)
- Directorio actual (oculta por defecto)

Enteros

Ejemplo

```
>> 4+5

ans =

     9
```

Ejemplo

```
>> -6*2

ans =

   -12
```

Diferencia entre acabar con ; o no

Ejemplo

```
>> 4+5;
>>
```

No muestra el resultado

Asignación

Ejemplo

```
>> a=4

a =

     4
```

Ejemplo

```
b=5;
```

Ejemplo

```
>> a+b

ans =

     9
```

Números reales

Ejemplo

```
>> 7.8543
ans =
    7.8543
```

Infinito

Es capaz de trabajar con infinito

```
>> 5/0
ans =
   Inf
```

Not a number: NaN

```
>> 0/0
ans =
   NaN
```

Números reales: número de decimales

Ejemplo

```
>> 5.7899734545544
ans =
    5.7900
>> format long
>> 5.7899734545544
ans =
   5.78997345545400
>> format short
5.7899734545544
ans =
    5.7900
```

Números complejos

Ejemplo

```
>> c=4+5*i
c =
  4.000000000000000 + 5.000000000000000i

>> d=3-4*i
d =
  3.000000000000000 - 4.000000000000000i

>> c+d
ans =
  7.000000000000000 + 1.000000000000000i
```

Cadenas

Ejemplo

Hay que usar comillas simples, la que va en la tecla del signo de interrogación final.

```
>> cad='una cadena'
cad =
una cadena
```

Booleano

Ejemplo

```
>> true
ans =     1
>> false
ans =     0
>> true & true
ans =     1
>> true & false
ans =     0
>> false & false
ans =     0
>> true | true
ans =     1
>> true | false
ans =     1
>> false | false
ans =     0
```

Vectores

Ejemplo

```
>> v1=[1,2,3,4]
v1 =
     1     2     3     4

>> v1=[1 2 3 4]
v1 =
     1     2     3     4

>> v2=[1 ;2; 3; 4]
v2 =
     1
     2
     3
     4
```

Vectores

Ejemplo

```
>> v1=[1,2,3,4]
v1 =
     1     2     3     4

>> v1=[1 2 3 4]
v1 =
     1     2     3     4

>> v2=[1 ;2; 3; 4]
v2 =
     1
     2
     3
     4
```

Operar vectores

Ejemplo

```
>> v1*v2
ans =
    30
```

Ejemplo

```
>> v1=[1 2 3 4];
>> v3=[4 5 5 5];
>> v1+v3
ans =
     5     7     8     9
>> v1.*v3
ans =
     4    10    15    20
```

Matrices

Ejemplo

```
>> mat1=[1 2 ; 3 4]
mat1 =
     1     2
     3     4
>> mat2= [2 1 ;1 2]
mat2 =
     2     1
     1     2
>> mat1+mat2
ans =
     3     3
     4     6
>> mat1*mat2
ans =
     4     5
    10    11
>> mat1.*mat2
ans =
     2     2
     3     8
```

Tipos de matrices predefinidos

Ejemplo

```
>> eye(2)
ans =
     1     0
     0     1
>> zeros(2,3)
ans =
     0     0     0
     0     0     0
>> ones(2)
ans =
     1     1
     1     1
>> linspace(2,3,4)
ans =
    2.0000    2.3333    2.6667    3.0000
```

Trasposición de vectores y matrices

Ejemplo

```
>> v1=[1,2,3]
v1 =
     1     2     3

>> v1'
ans =
     1
     2
     3

>> m1=[1 2; 3 4]
m1 =
     1     2
     3     4
>> m1'
ans =
     1     3
     2     4
```

El operador dos puntos :

Permite crear rangos. Se trata de una de las formas de definir vectores y matrices más usada y más fácil de utilizar.

Ejemplo

```
>> 1:5
ans =
     1     2     3     4     5

>> x=1:1:10
x =
     1     2     3     4     5     6     7     8     9    10

x2=1:0.1:1.5
x2 =
    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000
```

Direccionamiento de vectores y matrices (I)

Tomar un elemento o rango de una matriz o vector

Ejemplo

```
>> mat=rand(4,5)
mat =
    0.8147    0.6324    0.9575    0.9572    0.4218
    0.9058    0.0975    0.9649    0.4854    0.9157
    0.1270    0.2785    0.1576    0.8003    0.7922
    0.9134    0.5469    0.9706    0.1419    0.9595

>> mat(1,2)
ans =
    0.6324

>> mat(:,3)
ans =
    0.9575
    0.9649
    0.1576
    0.9706
mat(1,:)
ans =
    0.8147    0.6324    0.9575    0.9572    0.4218
```

Direccionamiento de vectores y matrices (II)

Tomar un elemento o rango de una matriz o vector

Ejemplo

```
>> mat=rand(4,5)
mat =
    0.8147    0.6324    0.9575    0.9572    0.4218
    0.9058    0.0975    0.9649    0.4854    0.9157
    0.1270    0.2785    0.1576    0.8003    0.7922
    0.9134    0.5469    0.9706    0.1419    0.9595

>> mat(2:4,3:5)
ans =
    0.9649    0.4854    0.9157
    0.1576    0.8003    0.7922
    0.9706    0.1419    0.9595

>> mat(2:end,3:end)
ans =
    0.9649    0.4854    0.9157
    0.1576    0.8003    0.7922
    0.9706    0.1419    0.9595
```

Direccionamiento de vectores y matrices (III)

Tomar un elemento o rango de una matriz o vector

Ejemplo

```
>> mat=rand(4,5)
mat =
    0.8147    0.6324    0.9575    0.9572    0.4218
    0.9058    0.0975    0.9649    0.4854    0.9157
    0.1270    0.2785    0.1576    0.8003    0.7922
    0.9134    0.5469    0.9706    0.1419    0.9595

>> mat([2 4],[1 3 5])
ans =
    0.9058    0.9649    0.9157
    0.9134    0.9706    0.9595

>> mat([2 4],1:2:5)
ans =
    0.9058    0.9649    0.9157
    0.9134    0.9706    0.9595
```

Sentencias condicionales: Sentencia IF

Esta sentencia nos sirve para hacer bifurcaciones, podemos hacer 3 usos diferentes de ella:

Una sola sentencia que utilizamos si es verdadera y sino no hacemos nada

```
if (condition)
    then-body
end

>> if (4<5)
  disp('cumple')
end
cumple

>> if (4<5)
  a=5
end

a =
     5
```

Sentencias condicionales: Sentencia IF

Utilizando la expresión else con la que conseguiremos hacer uso de una expresión u otra dependiendo si es true o false.

```
if (condition)
    then-body
else
    else-body
end
```

```
>> if (4>5)
disp('cumple')
else
disp('no cumple')
end
no cumple
```

```
>> if (true)
disp('cumple')
else
disp('no cumple')
end
cumple
```

Sentencias condicionales: Sentencia IF

Utilizando la expression elseif con la que se pueden anidar bifurcaciones (aunque es mejor usar la sentencia switch)

```
if (condition)
    then-body
elseif (condition)
    elseif-body
else
    else-body
end

>> a=6;
>> if (a==5)
   disp('igual a 5')
elseif(a>5)
   disp('mayor que 5')
else
   disp('menor que 5')
end

mayor que 5
```

Sentencias condicionales: Sentencia SWITCH

Función similar a la concatenación de sentencias elseif, de manera que simplificada.

switch

```
switch expression
case label
   command_list
case label
   command_list
...
otherwise
   command_list
end
```

```
>> j=0;
>> switch j
    case -1
        disp('uno negativo');
    case 0
        disp('cero');
    case 1
        disp('uno positivo');
    otherwise
        disp('otro valor');
end
cero
```

Bucle: FOR

Repite una serie de sentencias un número determinado de veces, sin importar los procesos que ocurran dentro. Cuando var llega al valor expression el bucle se detiene.

switch

```
for var = expression
   command_list
endfor

>> for i=1:4
i+10
end

ans =    11
ans =    12
ans =    13
ans =    14
```

Bucle: DO-UNTIL (sólo OCTAVE)

Repite una serie de sentencias hasta que la condición until se hace true, momento en el que se detiene la ejecución.

switch

```
do
   command_list
until (condition)

do
> k++
> k
> until k==4
ans = 0
k =  1
ans =  1
k =  2
ans =  2
k =  3
ans =  3
k =  4
```

Bucle: WHILE

Similar a DO-UNTIL salvo que la comprobación de la condición se hace antes de la ejecución de la iteración.

switch

```
while (condition)
body
endwhile

>> k=0;
>> while k < 5
disp ('k es menor que 5 ya que vale')
disp (k)
k = k + 1;
end

k es menor que 5 ya que vale
     0
k es menor que 5 ya que vale
     1
k es menor que 5 ya que vale
     2
k es menor que 5 ya que vale
     3
k es menor que 5 ya que vale
     4
```

Bucles y condicionales anidados. Sentencia BREAK

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

break

```
>> for i=1:4
i+10
if (i+10)
>> for i=1:10
i+10
if (i+10 == 15)
  break
end
end

ans =    11
ans =    12
ans =    13
ans =    14
ans =    15
```

Bucles y condicionales anidados. Sentencia CONTINUE

La sentencia continue hace que automáticamente se pare la ejecución de la iteración actual, por lo que vuelve al principio del bucle (sólo sirve para el bucle FOR).

continue

```
>> for i=1:10
if (i+10 == 15)
continue
end
i+10
end

ans =    11
ans =    12
ans =    13
ans =    14
ans =    16
ans =    17
ans =    18
ans =    19
ans =    20
```

Comandos: Funciones matemáticas modo escalar

- sin(x) : seno
- cos(x) : coseno
- tan(x) : tangente
- asin(x) : arco seno
- acos(x) : arco coseno
- atan(x) : arco tangente (devuelve un ángulo entre -90 y 90)
- sinh(x) : seno hiperbólico
- cosh(x) : coseno hiperbólico
- tanh(x) : tangente hiperbólica
- asinh(x) : arco seno hiperbólico
- acosh(x) : arco coseno hiperbólico
- atanh(x) : arco tangente hiperbólica

Comandos: Funciones matemáticas escalares

- log(x) : logaritmo natural
- log10(x) : logaritmo decimal
- exp(x) : función exponencial
- sqrt(x) : raíz cuadrada
- round(x) : redondeo hacia el entero más próximo
- fix(x) : redondea hacia el entero más próximo a 0
- floor(x) : valor entero más próximo hacia abajo
- ceil(x) : valor entero más próximo hacia arriba
- gcd(x) : máximo común divisor
- lcm(x) : mínimo común múltiplo
- real(x) : partes reales
- imag(x) : partes imaginarias
- abs(x) : valores absolutos
- angle(x) : ángulos de fase

Constantes matemáticas

- `pi`
- `e`
- `i , I`
- `inf`
- `NaN`

Comandos de control de variables

- **diary** Guarda en un archivo los comando ejecutados
- **clear** Sin argumentos, clear elimina todas las variables creadas previamente (excepto las variables globales).
- **clear A, b** Borra las variables indicadas.
- **clc** Limpia la pantalla de comandos
- **who**: muestra las variables utilizadas.
- **whos**: muestra las variables utilizadas y su valor.
- Guardar y cargar espacio de trabajo (a través de la interfaz)

Bibliografía

- *GNU Octave*. (Manual en pdf)
- *Matlab y sus aplicaciones en las ciencias y la ingeniería*. César Pérez. Ed Pearson Prentice Hall. 2003
- *Cálculo científico con Mathlab y Octave*. A.Quarteroni, F.Saleri. Ed Springer. 2006.
- *Mástering Matlab*. Duane Hanselman, Bruce Littlefield. Ed Pearson Prentice Hall. 2005