

Dpto. de Economía Cuantitativa  
Universidad Complutense de Madrid  
GNU Octave y Matlab

Funciones de calculadora científica y gráficas

Marcos Bujosa

© 2005–2007 Marcos Bujosa [marcos.bujosa@ccee.ucm.es](mailto:marcos.bujosa@ccee.ucm.es)  
Actualizado el: 21 de junio de 2007

Versión 1.2



Algunos derechos reservados. Esta obra está bajo una licencia Reconocimiento-CompartirIgual de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/2.5/es/> o envíe una carta a Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Puede encontrar la última versión de este material en <http://www.ucm.es/info/ecocuan/mbb/index.html#octave>

### Resumen

Clase práctica de uso de MATLAB y GNU Octave para el Master MAEF del Instituto Complutense de Análisis Económico.



#### comandos de ayuda

1

- help
- help help
- help sqrt

También puede consultar:

- <http://octave.sourceforge.net/index/index.html>
- <http://www.gnu.org/software/octave/>
- <http://octave.sourceforge.net/>

Puede encontrar un tutorial completo en <http://www.aims.ac.za/resources/tutorials/octave/>

**EJERCICIO 1.** ¿Para qué sirve el comando diary?



#### calculadora científica

2

- `sqrt(1024)`
- `2*0.25+3*0.15+0.3`
- `platanos=2, manzanas=3, naranjas=1`
- `platanos*0.25+manzanas*0.15+naranjas*0.3`
- `precios=[0.25; 0.15; 0.30]`
- `cesta=[platanos, manzanas, naranjas]`  
(nótese las comas “,” y los puntos y comas “;”)
- `valor=cesta*precios`
- `2^3`  
es decir  $2^3$
- `(3*(cos(2*pi)+log(e)))/sqrt(9)`  
es decir  $\frac{3*(\cos 2\pi + \ln e)}{\sqrt{9}}$

**EJERCICIO 2.** suponga que en un segundo momento, los precios pasan a ser:

`precios2=[0.28; 0.13; 0.35]`

Calcule la tasa de crecimiento del valor de la cesta.

*Pista.*

- asigne a la variable `valor2` el valor de la cesta con los nuevos precios
- calcule la tasa de variación:  $\frac{v_t - v_{t-1}}{v_{t-1}}$

Más funciones aritméticas 3

**Trigonométricas :** `sin (x)`, `cos(x)`,...

**Otras :** `exp(x)`, `log(x)`, `sqrt(x)`, `abs(x)`, `sign(x)`,

**Redondeo :**

- `round(x)` redondea a la parte entera más próxima
- `fix(x)` redondea en dirección a cero:  

$$\text{fix}(3.56)=3; \quad \text{fix}(-3.56) =-3$$
- `floor(x)` redondea a la baja
- `ceil(x)` redondea al alza

Visite <http://www.gnu.org/software/octave/doc/interpreter/Arithmetic.html#Arithmetic>  
o bien: <http://www.gnu.org/software/octave/doc/interpreter/>

Constantes matemáticas: <http://www.gnu.org/software/octave/doc/interpreter/Mathematical-Constants.html#Mathematical-Constants>

**EJERCICIO 3.** ¿cómo redondearía el tercer decimal de un número con tres o más decimales? por ejemplo 1.23456

Leyendo y grabando datos 4

- guardando datos de una variable
  - **en octave:**  
`save -ascii NombreFichero.txt cesta`
  - **en matlab:**  
`save NombreFichero.txt cesta -ascii`
- guardando datos de varias variables
  - **en octave:**  
`save -ascii NombreFichero.txt cesta precios precios2`
  - **en matlab:**  
`save NombreFichero.txt cesta precios precios2 -ascii`
- `load NombreFichero.txt`

**EJERCICIO 4.** Realice las siguientes operaciones

- (a) Lea los datos del fichero `ipi.txt`
- (b) represéntelos gráficamente con el comando `plot`
- (c) guarde los datos en un fichero con otro nombre

↑ Variables 5

- who, whos
- antes hicimos las siguientes operaciones:
  - platanos=2, manzanas=3, naranjas=1
  - cesta=[platanos, manzanas, naranjas]
  - valor=cesta\*precios
- valor, platanos=4, valor
- who, clear manzanas, who, manzanas
- clear, who
- platanos=2, platanos, PlAtAnos

Nombres válidos: P, p1, preCio\_3\_2  
 Nombres NO válidos: 3p, #2, precio-patata

↑ Vectores 6

- Crear un vector fila:
  - VF=[1 -7 3 500]
  - VF=[1, -7, 3, 500]
  - o combinando ambos modos: VF=[1, -7 3 500]
- Crear un vector columna:
  - VC=[1; 0; -1; 5]
  - o bien “saltando de línea” pulsando “intro”  
 FC=[1  
 0  
 -1  
 5]
  - o combinando ambos modos:  
 FC=[1; 0  
 -1; 5]

↑ Matrices 7

- Crear una matriz empleando espacios “ ” y “;”  
 A=[1 2 1; 4 6 5; 7 2 7]
- Crear una matriz empleando comas “,” y saltos de línea  
 B=[1,1,2  
 5,4,2  
 2,4,31]

También podemos generar matrices agregando vectores

- con vectores fila:  
 a1=[1 2 1], a2=[4 6 5], a3=[7 2 7], A2=[a1; a2; a3]
- o vectores columna  
 b1=[1;5;2], b2=[1;4;4], b3=[2;2;3], B2=[b1, b2, b3]

Operaciones con matrices 8

- $C=A+B$
- $C2=C*10$
- $C3=C+5$
- $E=A*A; \quad n=2; \quad E2=A^n$
- $E3=A.*A; \quad E2=A.^n$
- $IB=inv(B); \quad IB*B$
- $A*B$
- $A.*B$
- $Trasp\_A=A'$
- $Vec\_A=A(:)$
- $det(A), \quad trace(A)$

Con `help -i matrix` se nos explican más operaciones (en GNU Octave)

### EJERCICIO 5.

(a) Cree las siguientes matrices:

$$A = \begin{bmatrix} 4 & 3 & 2 \\ 5 & 6 & 3 \\ 3 & 5 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & -1 & 2 & 6 \\ 4 & 4 & 1 & 5 \\ 5 & 2 & 4 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

(b) calcule

- $AB$
- $A^{-1}$
- la matriz  $X$  tal que  $AX = B$
- el vector  $x$  tal que  $Ax = b$ .

Otras formas de crear vectores y matrices 9

- siguiendo el esquema `comienzo:salto:fin`, por ejemplo  
`y= 2 : 2 : 10`  
`A=[4:-1:0;1:2:9]`
- si se omite `salto`, los saltos son de una unidad por defecto  
`y= 2 : 10`
- `linspace(0,8,10)` genera un vector con 10 números igualmente espaciados entre 0 y 8 (ambos incluidos)
- ¿Que hace `logspace (0,6,7)`?

EJERCICIO 6. Cree los siguientes vectores y matrices

(a)

$$[1 \ 2 \ 3 \ 4 \ 5]$$

(b)

$$\begin{bmatrix} 2 \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

(c)

$$\begin{bmatrix} 2 & 4 & 6 \\ 8 & 6 & 4 \end{bmatrix}$$

## EJERCICIO 7.

(a) Sin emplear GNU Octave escriba en el papel los vectores que resultan de las siguientes órdenes:

- `x1 = 0:1:5`
- `x2 = 0:5`
- `x3 = 2:4:14`
- `x4 = 13:-4:2`
- `x5 = pi*[0:1/2:2]`

Verifique posteriormente sus respuestas con GNU Octave.

(b) Genere un vector con todos los múltiplos de 3 menores que 50

(c) Genere un vector como el anterior, pero en orden descendente

*Pista.* el comando `floor` redondea hacia abajo.

(d) Genere un vector con 11 números uniformemente repartidos entre los números 5 y 10

↑	<u>Matrices especiales</u>	10
<ul style="list-style-type: none"> <li>▪ <code>I=eye(4)</code></li> <li>▪ <code>X=zeros(2,4); X2=zeros(3)</code></li> <li>▪ <code>J=ones(3,2); J2=ones(2)</code></li> <li>▪ <code>diag([1:5]); diag([1:5],1)</code></li> </ul> <p>Matrices aleatorias</p> <ul style="list-style-type: none"> <li>▪ datos con distribución Uniforme(0, 1) <code>U=rand(2); U2=rand(2,3)</code></li> <li>▪ datos con distribución N(0, 1) <code>N=randn(2,4); N2=randn(4)</code></li> </ul> <p>Diagonales</p> <ul style="list-style-type: none"> <li>▪ <code>diag(N), diag(N,1), diag(N,-2)</code></li> </ul>		

## EJERCICIO 8.

(a) Genere una matriz de dimensiones  $3 \times 4$  llena de cincos

(b) Genere una matriz identidad de dimensión 4 empleando `diag`

(c) Seleccione el elemento de “esquina” superior derecha de la matriz `N2`, de dimensiones  $4 \times 4$ , creada anteriormente.

(d) Genere un vector columna con 10 datos con distribución Uniforme(0, 1).

(e) Genere un vector fila con 7 datos con distribución N(0, 1).

## EJERCICIO 9.

(a) Genere una matriz de dimensiones  $3 \times 3$  de datos con distribución Uniforme(0, 2).

(b) Genere una matriz de dimensiones  $3 \times 3$  de datos con distribución Uniforme(1, 3).

(c) Genere un vector con 10 datos con distribución N(0, 4).

(d) Genere un vector con 10 datos con distribución N(5, 4).

↑	<u>Generar matrices a partir de otras matrices</u>	11
<p>Antes creamos las siguientes matrices:</p> <ul style="list-style-type: none"> <li>▪ <code>N=randn(2,4)</code></li> <li>▪ <code>X=zeros(2,4)</code></li> </ul> <p>Teclee <code>N,X</code> para verificar que todavía están en memoria (también puede emplear los comandos <code>who</code> o <code>whos</code>).</p> <p>Cree además el vector fila <code>P=1:3:12</code></p> <p>Genere las siguientes matrices:</p> <ul style="list-style-type: none"> <li>▪ <code>F=[N;X]</code></li> <li>▪ <code>K=[P;0 -1 0 -1; [1:4]; eye(4)]</code></li> </ul>		

Seleccionando y operando sobre los elementos de una matriz 12

- `K(1,2)`
- `K(1,2)=88`
- `K(:,2), K(:,3)`
- `K(2:5,1)`
- `K(2,:), K(2,1:3)`
- `K(3:4,1:3)`
- `K([1 3 5 7],[2 4])`
- `K([1 3 5 7],[2 4])=ones(4,2)*99`
- `K([1 3 5 7],[2 4])`
- `K(3,6)=7` observe que K no tiene 6 columnas.
- `K(:,3)=[]`

Más documentación en <http://www.gnu.org/software/octave/doc/interpreter/Matrix-Manipulation.html#Matrix-Manipulation>  
**EJERCICIO 10.**

- (a) Escriba un comando tal que extraiga los elementos con índice par de un vector  $x$  de longitud  $N$
- (b) Análogamente, escriba un comando que extraiga los elementos con índice impar de un vector  $x$  de longitud arbitraria (las funciones `length` o `size` y `max` le pueden ser útiles)
- (c) Genere (con el comando `zeros`) un vector  $x$  con 10 ceros. Reemplace cada uno de los elementos de índice par por el número  $\pi$

Operaciones lógicas 13

Dan como resultado 0 cuando son falsas.

- `2==2`
- `2==3`
- `[0 1 0 1 0]~= [0 1 2 3 4]`
- `[0 1 0 1 0]< [0 1 2 3 4]`
- `[0 1 0 1 0]>= [0 1 2 3 4]`
- `[0 2 4 6 8]==4`
- `[0 2 4 6 8]==4 | [0 2 4 6 8]==8`
- `[0 2 4 6 8]>=2 & [0 2 4 6 8]<8`
- `y=randn(10,1), y(y>0), y(y>0 & y<0.75)`

Funciones estadísticas (análisis de datos)

suponga que  $x$  es una matriz con  $f$  filas y  $n$  columnas:

- `mean(x)` vector con las medias aritméticas de las columnas
- `mean(x')` vector con las medias aritméticas de las filas de  $x$
- `sum(x)` vector con las sumas de los elementos de cada columna
- `prod(x)` vector con los productos de los elementos de cada columna
- `median(x)` vector con las medianas de las columnas
- `std(x)` *idem* con las desviaciones típicas
- `var(x)` *idem* con la varianza
- `max(x)` *idem* con el máximo valor
- `min(x)` *idem* con el mínimo valor
- `cov(x)` matriz de varianzas y covarianzas
- `corrcoef(x)` matriz de correlaciones

- `sort(x)` ordena los elementos de las columnas de `x` de menor a mayor (teclea `help sort` para consultar más opciones)

Más documentación en <http://www.gnu.org/software/octave/doc/interpreter/Statistics.html#Statistics> o en la versión PDF del [Manual de Octave. Sección Statistics](#) página 201.

↑	<u>Funciones gráficas</u>	14
<ul style="list-style-type: none"> <li>▪ <code>y=randn(100,1); z=randn(100,3); x=[-79:20]'; w=[x,[1:100]'];</code></li> <li>▪ <code>plot(y); plot(x,y); plot(x,z); plot(w,y);</code></li> <li>▪ <code>plot(x,y,'g;Normal(0,1)');</code></li> <li>▪ <code>figure(2);</code></li> <li>▪ <code>plot(x,y,'-@33;Normal(0,1)');</code> <code>hold on;</code></li> <li>▪ <code>plot(w(:,2),rand(100,1),'-@15;Uniforme(0,1)');</code> <code>hold off</code></li> <li>▪ <code>figure(3);</code></li> <li>▪ <code>xlabel('x');</code></li> <li>▪ <code>ylabel('y');</code></li> <li>▪ <code>title('Figura de muestra')</code></li> <li>▪ <code>plot(x,y,'-@33;Normal(0,1)'; w(:,2),rand(100,1),'-@15;Uniforme(0,1)');</code></li> </ul>		

↑	<u>Funciones gráficas</u>	15
<ul style="list-style-type: none"> <li>▪ <code>figure(4);</code></li> <li>▪ <code>subplot(2,1,1); plot(x,y);</code></li> <li>▪ <code>subplot(2,1,2); plot(x,z);</code></li> <li>▪ <code>figure(5); stem([1:10],randn(10,1),'*');</code></li> </ul>		

**EJERCICIO 11.** En todos los gráficos ponga etiquetas en los ejes y añada un título adecuado.


- (a) cree una variable `t` con valores de 0 a  $2\pi$  en saltos de 0.001; y también los siguientes vectores:

$$y_1 = \sin(t); \quad y_2 = t^2 + \cos(t) + \frac{e^{t^2}}{10^{16}}; \quad y_3 = \cos(t)$$

- (b) Represente gráficamente  $y_1$  e  $y_2$  frente a  $t$  en figuras separadas
- (c) Represente gráficamente  $y_1$  e  $y_2$  frente a  $t$  en la misma figura;  $y_3$  debe aparecer en verde
- (d) Divida una figura en dos empleando `subplot`.  
Dibuje  $y_1$  e  $y_3$  en la parte de arriba, e  $y_2$  en la parte de abajo (todas frente a  $t$ )
- (e) Cree el vector `n` con los números enteros de -10 a 10; y genere un gráfico con `stem` de  $y_4 = \cos 2\pi \frac{n}{10}$

↑	<u>Funciones gráficas</u>	16
<ul style="list-style-type: none"> <li>▪ <code>hist(y)</code></li> <li>▪ <code>hist(y,30)</code></li> <li>▪ <code>figure(4); stem([1:10],randn(10,1),'*');</code></li> <li>▪ <code>y=randn(100,2); scatter(y(:,1),y(:,2))</code></li> <li>▪ <code>y=randn(100,2); scatter(y(:,1),y(:,2), 0 , 3 )</code></li> <li>▪ <code>w=rand(1000,2);</code></li> <li>▪ <code>w=w(w(:,1)&lt;w(:,2),:);</code></li> <li>▪ <code>x=w(:,1); y=w(:,2);</code></li> <li>▪ <code>scatter(x,y,0, y&gt;1/2)</code></li> <li>▪ <code>sum(y&gt;1/2)/length(w)</code></li> </ul>		

## 1. gráficos tridimensionales (Matlab)



función escalar de dos variables **17**

Queremos representar  $z = f(x, y)$  Para un conjunto de pares de valores  $x, y$ . Dicho conjunto se genera con `meshgrid`:

```
x = -3:3;
y = 1:5;
[X,Y] = meshgrid(x,y)
```

Supongamos que la función es  $z = f(x, y) = (x + y)^2$  y por tanto

```
Z = (X+Y).^2
```


mesh **18**

```
mesh(Z)
colormap(gray)
```

los ejes no corresponden


```
mesh(X,Y,Z)
```

Pruebe


```
hidden off
```

y también

```
hidden on
```


Más ejemplos **19**

```
[X,Y,Z]=sphere(12);
mesh(X,Y,Z)
colormap(hsv)
hidden off
hidden on
```


mapa de colores colormap **20**

**autumn** varies smoothly from red, through orange, to yellow.

**bone** is a grayscale colormap with a higher value for the blue component. This colormap is useful for adding an "electronic" look to grayscale images.


**colorcube** contains as many regularly spaced colors in RGB colorspace as possible, while attempting to provide more steps of gray, pure red, pure green, and pure blue.

**cool** consists of colors that are shades of cyan and magenta. It varies smoothly from cyan to magenta.

**copper** varies smoothly from black to bright copper.

**flag** consists of the colors red, white, blue, and black. This colormap completely changes color with each index increment.


**gray** returns a linear grayscale colormap.


mapa de colores colormap **21**

**hot** varies smoothly from black through shades of red, orange, and yellow, to white.

**hsv** varies the hue component of the hue-saturation-value color model. The colors begin with red, pass through yellow, green, cyan, blue, magenta, and return to red. The colormap is particularly appropriate for displaying peqriodic functions.

**jet** ranges from blue to red, and passes through the colors cyan, yellow, and orange. It is a variation of the hsv colormap. The jet colormap is associated with an astrophysical fluid jet simulation from the National Center for Supercomputer Applications. See the "Examples" section.

 [mapa de colores colormap](#) 22

**lines** produces a colormap of colors specified by the axes ColorOrder property and a shade of gray.

**pink** contains pastel shades of pink. The pink colormap provides sepia tone colorization of grayscale photographs.


**prism** repeats the six colors red, orange, yellow, green, blue, and violet.

**spring** consists of colors that are shades of magenta and yellow.

**summer** consists of colors that are shades of green and yellow.


**white** is an all white monochrome colormap.

**winter** consists of colors that are shades of blue and green.

 [mapa de colores colormap](#) 23

Se pueden crear mapas de colores nuevos.

```
function map = waves(m)
    % save in file called waves.m
    m=128;
    R = zeros(m,3);
    j=1:128;
    r=0.5*(1-sin(2*pi*j/128));
    b=0.5*(1+sin(2*pi*j/128));
    g=0.2*(2+sin(2*pi*j/8));
    R(:,1)=r';
    R(:,2)=g';
    R(:,3)=b';
    map=R;
```

 [image](#) 24

Pruebe lo siguiente

```
load flujet
image(X) % in gray map chosen earlier
```

Cambiando el mapa de colores:

```
colormap(jet) % change map
```

Cambiando a nuestro nuevo mapa de colores

```
colormap(waves) % change map
```

 [surf](#) 25

Crea una superficie en lugar de una rejilla.

```
[X,Y,Z]=peaks(30);
surf(X,Y,Z)
% compare con mesh(X,Y,Z)
colormap(hsv)
shading flat
shading interp
colormap waves
colormap hot
```

↑	<u>Variantes de surf</u>	26
<p>Podemos incluir curvas de nivel</p> <pre>surfc(X,Y,Z)</pre> <p>o luces y sombras</p> <pre>surf1(X,Y,Z) shading interp colormap(pink) % mejor con un color</pre> <p>Pulse en Tools &gt;Rotate3d en el menú de la ventana  Pulse en Tools &gt;Reset View en el menú de la ventana para restaurar la vista original</p> <p>Con</p> <pre>view(-37.5,30.0)</pre> <p>también se puede modificar el punto de vista , donde el primer número es el ángulo respecto de <math>x</math> y el segundo respecto de <math>z</math></p>		

↑	<u>Otros gráficos: contour, contour3, pcolor, contourf</u>	27
<pre>[X,Y,Z]=peaks; contour(X,Y,Z,20); colormap(hsv) contour3(X,Y,Z,20); pcolor(X,Y,Z); shading interp axis square contourf(X,Y,Z,12); C=contour(X,Y,Z,12); clabel(C);</pre>		

↑	<u>Gradientes y campos vectoriales</u>	28
<pre>[X,Y,Z]=peaks(16); [DX,DY]=gradient(Z,0.5,0.5); contour(X,Y,Z,10); colormap(hot) hold on quiver(X,Y,DX,DY); hold off  [X,Y,Z]=peaks(20); [Nx,Ny,Nz]=surfnorm(X,Y,Z); surf(X,Y,Z); hold on quiver3(X,Y,Z,Nx,Ny,Nz); hold off</pre>		

Más gráficos tecleando [demo](#) en Matlab.

### Lista de Transparencias

- 1 comandos de ayuda
- 2 calculadora científica
- 3 Más funciones aritméticas
- 4 Leyendo y grabando datos
- 5 Variables
- 6 Vectores
- 7 Matrices
- 8 Operaciones con matrices
- 9 Otras formas de crear vectores y matrices
- 10 Matrices especiales

- 11 Generar matrices a partir de otras matrices
- 12 Seleccionando y operando sobre los elementos de una matriz
- 13 Operaciones lógicas
- 14 Funciones gráficas
- 15 Funciones gráficas
- 16 Funciones gráficas
- 17 función escalar de dos variables
- 18 `mesh`
- 19 Más ejemplos
- 20 mapa de colores `colormap`
- 21 mapa de colores `colormap`
- 22 mapa de colores `colormap`
- 23 mapa de colores `colormap`
- 24 `image`
- 25 `surf`
- 26 Variantes de `surf`
- 27 Otros gráficos: `contour`, `contour3`, `pcolor`, `contourf`
- 28 Gradientes y campos vectoriales

## Starting Octave

<code>octave</code>	start interactive Octave session
<code>octave file</code>	run Octave on commands in <i>file</i>
<code>octave --help</code>	describe command line options

## Stopping Octave

<code>quit</code> or <code>exit</code>	exit Octave
<code>INTERRUPT</code>	( <i>e.g.</i> <code>C-c</code> ) terminate current command and return to top-level prompt

## Getting Help

<code>help</code>	list all commands and built-in variables
<code>help command</code>	briefly describe <i>command</i>
<code>help -i</code>	use Info to browse Octave manual
<code>help -i command</code>	search for <i>command</i> in Octave manual

## Motion in Info

<code>SPC</code> or <code>C-v</code>	scroll forward one screenful
<code>DEL</code> or <code>M-v</code>	scroll backward one screenful
<code>C-l</code>	redraw the display

## Node Selection in Info

<code>n</code>	select the next node
<code>p</code>	select the previous node
<code>u</code>	select the ‘up’ node
<code>t</code>	select the ‘top’ node
<code>d</code>	select the directory node
<code>&lt;</code>	select the first node in the current file
<code>&gt;</code>	select the last node in the current file
<code>g</code>	reads the name of a node and selects it
<code>C-x k</code>	kills the current node

## Searching in Info

<code>s</code>	search for a string
<code>C-s</code>	search forward incrementally
<code>C-r</code>	search backward incrementally
<code>i</code>	search index & go to corresponding node
<code>,</code>	go to next match from last ‘i’ command

## Command-Line Cursor Motion

<code>C-b</code>	move back one character
<code>C-f</code>	move forward one character
<code>C-a</code>	move the the start of the line
<code>C-e</code>	move to the end of the line
<code>M-f</code>	move forward a word
<code>M-b</code>	move backward a word
<code>C-l</code>	clear screen, reprinting current line at top

## Inserting or Changing Text

<code>M-TAB</code>	insert a tab character
<code>DEL</code>	delete character to the left of the cursor
<code>C-d</code>	delete character under the cursor
<code>C-v</code>	add the next character verbatim
<code>C-t</code>	transpose characters at the point
<code>M-t</code>	transpose words at the point

[ ] surround optional arguments ... show one or more arguments

## Killing and Yanking

<code>C-k</code>	kill to the end of the line
<code>C-y</code>	yank the most recently killed text
<code>M-d</code>	kill to the end of the current word
<code>M-DEL</code>	kill the word behind the cursor
<code>M-y</code>	rotate the kill ring and yank the new top

## Command Completion and History

<code>TAB</code>	complete a command or variable name
<code>M-?</code>	list possible completions
<code>RET</code>	enter the current line
<code>C-p</code>	move ‘up’ through the history list
<code>C-n</code>	move ‘down’ through the history list
<code>M-&lt;</code>	move to the first line in the history
<code>M-&gt;</code>	move to the last line in the history
<code>C-r</code>	search backward in the history list
<code>C-s</code>	search forward in the history list
<code>history [-q] [N]</code>	list <i>N</i> previous history lines, omitting history numbers if <code>-q</code>
<code>history -w [file]</code>	write history to <i>file</i> ( <code>~/octave_hist</code> if no <i>file</i> argument)
<code>history -r [file]</code>	read history from <i>file</i> ( <code>~/octave_hist</code> if no <i>file</i> argument)
<code>edit_history lines</code>	edit and then run previous commands from the history list
<code>run_history lines</code>	run previous commands from the history list
<code>[beg] [end]</code>	Specify the first and last history commands to edit or run.
	If <i>beg</i> is greater than <i>end</i> , reverse the list of commands before editing. If <i>end</i> is omitted, select commands from <i>beg</i> to the end of the history list. If both arguments are omitted, edit the previous item in the history list.

## Shell Commands

<code>cd dir</code>	change working directory to <i>dir</i>
<code>pwd</code>	print working directory
<code>ls [options]</code>	print directory listing
<code>getenv (string)</code>	return value of named environment variable
<code>system (cmd)</code>	execute arbitrary shell command string

## Matrices

Square brackets delimit literal matrices. Commas separate elements on the same row. Semicolons separate rows. Commas may be replaced by spaces, and semicolons may be replaced by one or more newlines. Elements of a matrix may be arbitrary expressions, provided that all the dimensions agree.

<code>[ x, y, ... ]</code>	enter a row vector
<code>[ x; y; ... ]</code>	enter a column vector
<code>[ w, x; y, z ]</code>	enter a 2×2 matrix

## Ranges

*base* : *limit*  
*base* : *incr* : *limit*

Specify a range of values beginning with *base* with no elements greater than *limit*. If it is omitted, the default value of *incr* is 1. Negative increments are permitted.

## Strings and Common Escape Sequences

A *string constant* consists of a sequence of characters enclosed in either double-quote or single-quote marks.

<code>\\</code>	a literal backslash
<code>\"</code>	a literal double-quote character
<code>\'</code>	a literal single-quote character
<code>\n</code>	newline, ASCII code 10
<code>\t</code>	horizontal tab, ASCII code 9

## Index Expressions

<code>var (idx)</code>	select elements of a vector
<code>var (idx1, idx2)</code>	select elements of a matrix
<code>scalar</code>	select row (column) corresponding to <i>scalar</i>
<code>vector</code>	select rows (columns) corresponding to the elements of <i>vector</i>
<code>range</code>	select rows (columns) corresponding to the elements of <i>range</i>
<code>:</code>	select all rows (columns)

## Global Variables

`global var1 ...` Declare variables global.

Global variables may be accessed inside the body of a function without having to be passed in the function parameter list provided they are also declared global within the function.

## Selected Built-in Variables

<code>EDITOR</code>	editor to use with <code>edit_history</code>
<code>Inf, NaN</code>	IEEE infinity, NaN
<code>LOADPATH</code>	path to search for function files
<code>PAGER</code>	program to use to paginate output
<code>ans</code>	last result not explicitly assigned
<code>eps</code>	machine precision
<code>pi</code>	$\pi$
<code>realmax</code>	maximum representable value
<code>realmin</code>	minimum representable value

<code>automatic_replot</code>	automatically redraw plots
<code>do_fortran_indexing</code>	Fortran-style indexing of matrices
<code>implicit_str_to_num_ok</code>	allow strings to become numbers
<code>output_max_field_width</code>	maximum numeric field width
<code>output_precision</code>	min significant figures displayed
<code>page_screen_output</code>	control whether output is paged
<code>prefer_column_vectors</code>	create column vectors by default
<code>resize_on_range_error</code>	automatic resizing of matrices
<code>save_precision</code>	digits stored by <code>save</code> command
<code>silent_functions</code>	suppress output from functions
<code>warn_divide_by_zero</code>	suppress divide by zero errors

`commas_in_literal_matrix`  
control handling of spaces in matrices

`ignore_function_time_stamp`  
ignore changes in function files during session

`ok_to_lose_imaginary_part`  
allow complex to real conversion

`prefer_zero_one_indexing`  
if ambiguous, prefer 0-1 style indexing

## Arithmetic and Increment Operators

<code>x + y</code>	addition
<code>x - y</code>	subtraction
<code>x * y</code>	matrix multiplication
<code>x .* y</code>	element by element multiplication
<code>x / y</code>	right division, conceptually equivalent to $(\text{inverse}(y') * x')$
<code>x ./ y</code>	element by element right division
<code>x \ y</code>	left division, conceptually equivalent to $\text{inverse}(x) * y$
<code>x.\ y</code>	element by element left division
<code>x ^ y</code>	power operator
<code>x .^ y</code>	element by element power operator
<code>- x</code>	negation
<code>+ x</code>	unary plus (a no-op)
<code>x ' </code>	complex conjugate transpose
<code>x .' </code>	transpose
<code>++ x ( -- x)</code>	increment (decrement) <i>x</i> , return <i>new</i> value
<code>x ++ ( x --)</code>	increment (decrement) <i>x</i> , return <i>old</i> value

## Assignment Expressions

<code>var = expr</code>	assign expression to variable
<code>var (idx) = expr</code>	assign expression to indexed variable

## Comparison and Boolean Operators

These operators work on an element-by-element basis. Both arguments are always evaluated.

<code>x &lt; y</code>	true if <i>x</i> is less than <i>y</i>
<code>x &lt;= y</code>	true if <i>x</i> is less than or equal to <i>y</i>
<code>x == y</code>	true if <i>x</i> is equal to <i>y</i>
<code>x &gt;= y</code>	true if <i>x</i> is greater than or equal to <i>y</i>
<code>x &gt; y</code>	true if <i>x</i> is greater than <i>y</i>
<code>x != y</code>	true if <i>x</i> is not equal to <i>y</i>
<code>x &amp; y</code>	true if both <i>x</i> and <i>y</i> are true
<code>x   y</code>	true if at least one of <i>x</i> or <i>y</i> is true
<code>! bool</code>	true if <i>bool</i> is false

## Short-circuit Boolean Operators

Operators evaluate left-to-right, expecting scalar operands. Operands are only evaluated if necessary, stopping once overall truth value can be determined. Operands are converted to scalars by applying the `all` function.

<code>x &amp;&amp; y</code>	true if both <i>x</i> and <i>y</i> are true
<code>x    y</code>	true if at least one of <i>x</i> or <i>y</i> is true

## Operator Precedence

Here is a table of the operators in Octave, in order of increasing precedence.

<code>;</code>	statement separators
<code>=</code>	assignment, groups left to right
<code>   &amp;&amp;</code>	logical “or” and “and”
<code>  &amp;</code>	element-wise “or” and “and”
<code>&lt; &lt;= == &gt;= &gt; !=</code>	relational operators
<code>:</code>	colon
<code>+ -</code>	addition and subtraction
<code>* / \ .* ./ .\</code>	multiplication and division
<code>' .'</code>	transpose
<code>+ - ++ -- !</code>	unary minus, increment, logical “not”
<code>^ .^</code>	exponentiation

## Statements

**for** *identifier* = *expr stmt-list* **endfor**  
Execute *stmt-list* once for each column of *expr*. The variable *identifier* is set to the value of the current column during each iteration.

**while** (*condition*) *stmt-list* **endwhile**  
Execute *stmt-list* while *condition* is true.

**break** exit innermost loop  
**continue** go to beginning of innermost loop  
**return** return to calling function

**if** (*condition*) *if-body* [**else** *else-body*] **endif**  
Execute *if-body* if *condition* is true, otherwise execute *else-body*.

**if** (*condition*) *if-body* [**elseif** (*condition*) *elseif-body*] **endif**  
Execute *if-body* if *condition* is true, otherwise execute the *elseif-body* corresponding to the first **elseif** condition that is true, otherwise execute *else-body*.  
Any number of **elseif** clauses may appear in an **if** statement.

**unwind\_protect** *body* **unwind\_protect\_cleanup** *cleanup* **end**  
Execute *body*. Execute *cleanup* no matter how control exits *body*.

## Defining Functions

**function** [*ret-list*] *function-name* [(*arg-list*)]  
*function-body*  
**endfunction**

*ret-list* may be a single identifier or a comma-separated list of identifiers delimited by square-brackets.

*arg-list* is a comma-separated list of identifiers and may be empty.

## Basic Matrix Manipulations

<b>rows</b> ( <i>a</i> )	return number of rows of <i>a</i>
<b>columns</b> ( <i>a</i> )	return number of columns of <i>a</i>
<b>all</b> ( <i>a</i> )	check if all elements of <i>a</i> nonzero
<b>any</b> ( <i>a</i> )	check if any elements of <i>a</i> nonzero
<b>find</b> ( <i>a</i> )	return indices of nonzero elements
<b>sort</b> ( <i>a</i> )	order elements in each column of <i>a</i>
<b>sum</b> ( <i>a</i> )	sum elements in columns of <i>a</i>
<b>prod</b> ( <i>a</i> )	product of elements in columns of <i>a</i>
<b>min</b> ( <i>args</i> )	find minimum values
<b>max</b> ( <i>args</i> )	find maximum values
<b>rem</b> ( <i>x</i> , <i>y</i> )	find remainder of <i>x/y</i>
<b>reshape</b> ( <i>a</i> , <i>m</i> , <i>n</i> )	reformat <i>a</i> to be <i>m</i> by <i>n</i>

<b>diag</b> ( <i>v</i> , <i>k</i> )	create diagonal matrices
<b>linspace</b> ( <i>b</i> , <i>l</i> , <i>n</i> )	create vector of linearly-spaced elements
<b>logspace</b> ( <i>b</i> , <i>l</i> , <i>n</i> )	create vector of log-spaced elements
<b>eye</b> ( <i>n</i> , <i>m</i> )	create <i>n</i> by <i>m</i> identity matrix
<b>ones</b> ( <i>n</i> , <i>m</i> )	create <i>n</i> by <i>m</i> matrix of ones
<b>zeros</b> ( <i>n</i> , <i>m</i> )	create <i>n</i> by <i>m</i> matrix of zeros
<b>rand</b> ( <i>n</i> , <i>m</i> )	create <i>n</i> by <i>m</i> matrix of random values

## Linear Algebra

<b>chol</b> ( <i>a</i> )	Cholesky factorization
<b>det</b> ( <i>a</i> )	compute the determinant of a matrix
<b>eig</b> ( <i>a</i> )	eigenvalues and eigenvectors
<b>expm</b> ( <i>a</i> )	compute the exponential of a matrix
<b>hess</b> ( <i>a</i> )	compute Hessenberg decomposition
<b>inverse</b> ( <i>a</i> )	invert a square matrix
<b>norm</b> ( <i>a</i> , <i>p</i> )	compute the <i>p</i> -norm of a matrix
<b>pinv</b> ( <i>a</i> )	compute pseudoinverse of <i>a</i>
<b>qr</b> ( <i>a</i> )	compute the QR factorization of a matrix
<b>rank</b> ( <i>a</i> )	matrix rank
<b>schur</b> ( <i>a</i> )	Schur decomposition of a matrix
<b>svd</b> ( <i>a</i> )	singular value decomposition
<b>syl</b> ( <i>a</i> , <i>b</i> , <i>c</i> )	solve the Sylvester equation

## Equations, ODEs, DAEs, Quadrature

<b>*fsolve</b>	solve nonlinear algebraic equations
<b>*lsode</b>	integrate nonlinear ODEs
<b>*dassl</b>	integrate nonlinear DAEs
<b>*quad</b>	integrate nonlinear functions

**perror** (*nm*, *code*) for functions that return numeric codes, print error message for named function and given error code

\* See the on-line or printed manual for the complete list of arguments for these functions.

## Signal Processing

<b>fft</b> ( <i>a</i> )	Fast Fourier Transform using FFTPACK
<b>ifft</b> ( <i>a</i> )	inverse FFT using FFTPACK
<b>freqz</b> ( <i>args</i> )	FIR filter frequency response
<b>sinc</b> ( <i>x</i> )	returns $\sin(\pi x)/(\pi x)$

## Image Processing

<b>colormap</b> ( <i>map</i> )	set the current colormap
<b>gray2ind</b> ( <i>i</i> , <i>n</i> )	convert gray scale to Octave image
<b>image</b> ( <i>img</i> , <i>zoom</i> )	display an Octave image matrix
<b>imagesc</b> ( <i>img</i> , <i>zoom</i> )	display scaled matrix as image
<b>imshow</b> ( <i>img</i> , <i>map</i> )	display Octave image
<b>imshow</b> ( <i>i</i> , <i>n</i> )	display gray scale image
<b>imshow</b> ( <i>r</i> , <i>g</i> , <i>b</i> )	display RGB image
<b>ind2gray</b> ( <i>img</i> , <i>map</i> )	convert Octave image to gray scale
<b>ind2rgb</b> ( <i>img</i> , <i>map</i> )	convert indexed image to RGB
<b>loadimage</b> ( <i>file</i> )	load an image file
<b>rgb2ind</b> ( <i>r</i> , <i>g</i> , <i>b</i> )	convert RGB to Octave image
<b>saveimage</b> ( <i>file</i> , <i>img</i> , <i>fmt</i> , <i>map</i> )	save a matrix to <i>file</i>

## Sets

<b>create_set</b> ( <i>a</i> , <i>b</i> )	create row vector of unique values
<b>complement</b> ( <i>a</i> , <i>b</i> )	elements of <i>b</i> not in <i>a</i>
<b>intersection</b> ( <i>a</i> , <i>b</i> )	intersection of sets <i>a</i> and <i>b</i>
<b>union</b> ( <i>a</i> , <i>b</i> )	union of sets <i>a</i> and <i>b</i>

## Strings

<b>strcmp</b> ( <i>s</i> , <i>t</i> )	compare strings
<b>strcat</b> ( <i>s</i> , <i>t</i> , ...)	concatenate strings

## C-style Input and Output

<code>fopen</code> ( <i>name, mode</i> )	open file <i>name</i>
<code>fclose</code> ( <i>file</i> )	close <i>file</i>
<code>printf</code> ( <i>fmt, ...</i> )	formatted output to <code>stdout</code>
<code>fprintf</code> ( <i>file, fmt, ...</i> )	formatted output to <i>file</i>
<code>sprintf</code> ( <i>fmt, ...</i> )	formatted output to string
<code>scanf</code> ( <i>fmt</i> )	formatted input from <code>stdin</code>
<code>fscanf</code> ( <i>file, fmt</i> )	formatted input from <i>file</i>
<code>sscanf</code> ( <i>str, fmt</i> )	formatted input from <i>string</i>
<code>fgets</code> ( <i>file, len</i> )	read <i>len</i> characters from <i>file</i>
<code>fflush</code> ( <i>file</i> )	flush pending output to <i>file</i>
<code>ftell</code> ( <i>file</i> )	return file pointer position
<code>frewind</code> ( <i>file</i> )	move file pointer to beginning
<code>freport</code>	print a info for open files
<code>fread</code> ( <i>file, size, prec</i> )	read binary data files
<code>fwrite</code> ( <i>file, size, prec</i> )	write binary data files
<code>feof</code> ( <i>file</i> )	determine if pointer is at EOF

A file may be referenced either by name or by the number returned from `fopen`. Three files are preconnected when Octave starts: `stdin`, `stdout`, and `stderr`.

## Other Input and Output functions

<code>save</code> <i>file var ...</i>	save variables in <i>file</i>
<code>load</code> <i>file</i>	load variables from <i>file</i>
<code>disp</code> ( <i>var</i> )	display value of <i>var</i> to screen

## Miscellaneous Functions

<code>eval</code> ( <i>str</i> )	evaluate <i>str</i> as a command
<code>feval</code> ( <i>str, ...</i> )	evaluate function named by <i>str</i> , passing remaining args to called function
<code>error</code> ( <i>message</i> )	print message and return to top level
<code>clear</code> <i>pattern</i>	clear variables matching pattern
<code>exist</code> ( <i>str</i> )	check existence of variable or function
<code>who</code>	list current variables

## Polynomials

<code>compan</code> ( <i>p</i> )	companion matrix
<code>conv</code> ( <i>a, b</i> )	convolution
<code>deconv</code> ( <i>a, b</i> )	deconvolve two vectors
<code>poly</code> ( <i>a</i> )	create polynomial from a matrix
<code>polyderiv</code> ( <i>p</i> )	derivative of polynomial
<code>polyreduce</code> ( <i>p</i> )	integral of polynomial
<code>polyval</code> ( <i>p, x</i> )	value of polynomial at <i>x</i>
<code>polyvalm</code> ( <i>p, x</i> )	value of polynomial at <i>x</i>
<code>roots</code> ( <i>p</i> )	polynomial roots
<code>residue</code> ( <i>a, b</i> )	partial fraction expansion of ratio <i>a/b</i>

## Statistics

<code>corrcoef</code> ( <i>x, y</i> )	correlation coefficient
<code>cov</code> ( <i>x, y</i> )	covariance
<code>mean</code> ( <i>a</i> )	mean value
<code>median</code> ( <i>a</i> )	median value
<code>std</code> ( <i>a</i> )	standard deviation
<code>var</code> ( <i>a</i> )	variance

## Basic Plotting

<code>plot</code> [ <i>ranges</i> ] <i>expr</i> [ <i>using</i> ] [ <i>title</i> ] [ <i>style</i> ]	2D plotting
<code>gplot</code> [ <i>ranges</i> ] <i>expr</i> [ <i>using</i> ] [ <i>title</i> ] [ <i>style</i> ]	3D plotting
<i>ranges</i>	specify data ranges
<i>expr</i>	expression to plot
<i>using</i>	specify columns to plot
<i>title</i>	specify line title for legend
<i>style</i>	specify line style

If *ranges* are supplied, they must come before the expression to plot. The *using*, *title*, and *style* options may appear in any order after *expr*. Multiple expressions may be plotted with a single command by separating them with commas.

<code>set options</code>	set plotting options
<code>show options</code>	show plotting options
<code>replot</code>	redisplay current plot
<code>closeplot</code>	close stream to <code>gnuplot</code> process
<code>purge_tmp_files</code>	clean up temporary plotting files
<code>automatic_replot</code>	built-in variable

## Other Plotting Functions

<code>plot</code> ( <i>args</i> )	2D plot with linear axes
<code>semilogx</code> ( <i>args</i> )	2D plot with logarithmic x-axis
<code>semilogy</code> ( <i>args</i> )	2D plot with logarithmic y-axis
<code>loglog</code> ( <i>args</i> )	2D plot with logarithmic axes
<code>bar</code> ( <i>args</i> )	plot bar charts
<code>stairs</code> ( <i>x, y</i> )	plot stairsteps
<code>hist</code> ( <i>y, x</i> )	plot histograms

`title` (*string*) set plot title

<code>axis</code> ( <i>limits</i> )	set axis ranges
<code>xlabel</code> ( <i>string</i> )	set x-axis label
<code>ylabel</code> ( <i>string</i> )	set y-axis label
<code>grid</code> [ <i>on off</i> ]	set grid state
<code>hold</code> [ <i>on off</i> ]	set hold state
<code>ishold</code>	return 1 if hold is on, 0 otherwise

<code>mesh</code> ( <i>x, y, z</i> )	plot 3D surface
<code>meshdom</code> ( <i>x, y</i> )	create mesh coordinate matrices

---

Edition 1.1 for Octave Version 1.1.1. Copyright 1996, John W. Eaton (jwe@che.utexas.edu). The author assumes no responsibility for any errors on this card.

This card may be freely distributed under the terms of the GNU General Public License.

T<sub>E</sub>X Macros for this card by Roland Pesch (pesch@cygnus.com), originally for the GDB reference card

Octave itself is free software; you are welcome to distribute copies of it under the terms of the GNU General Public License. There is absolutely no warranty for Octave.