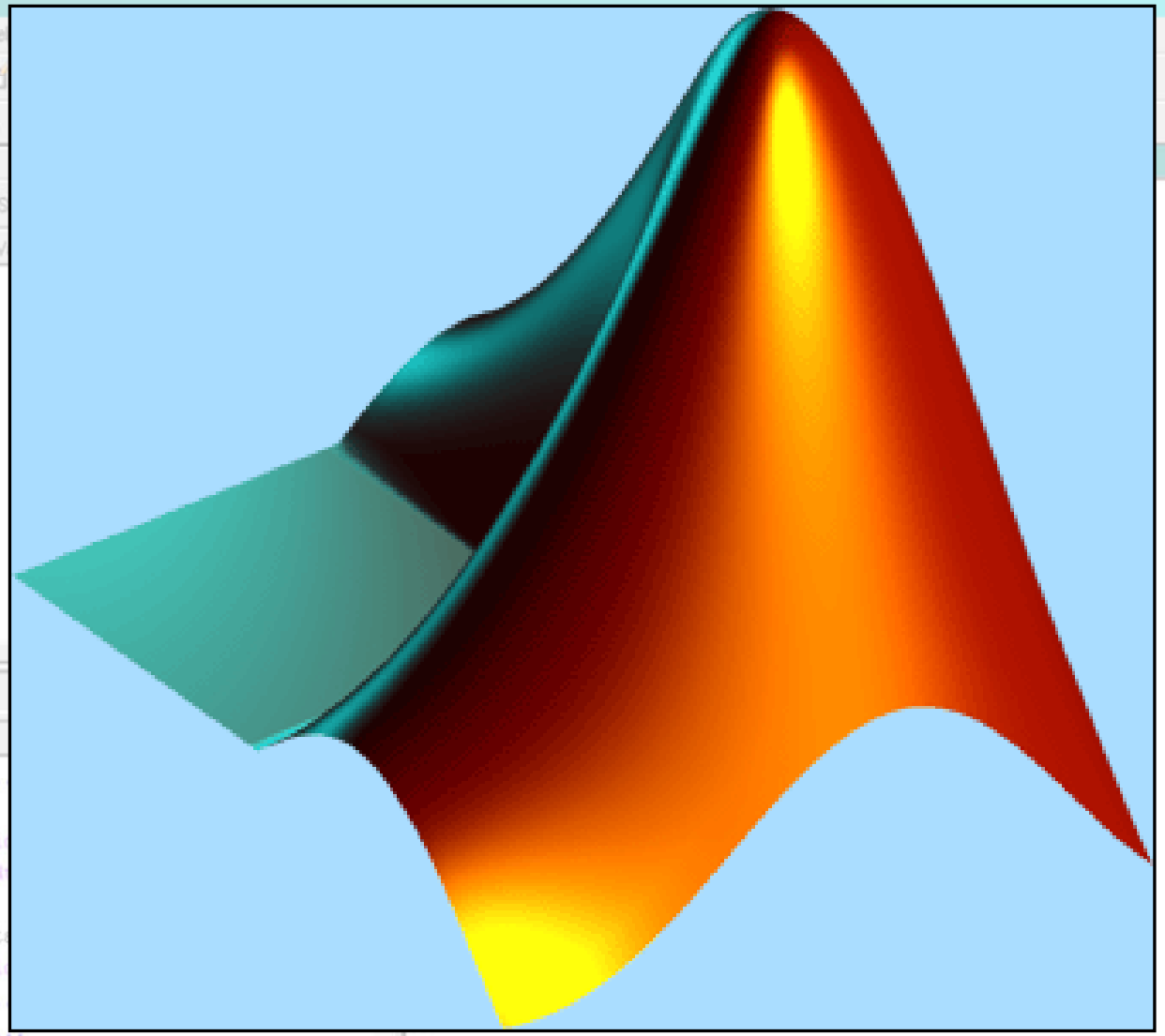
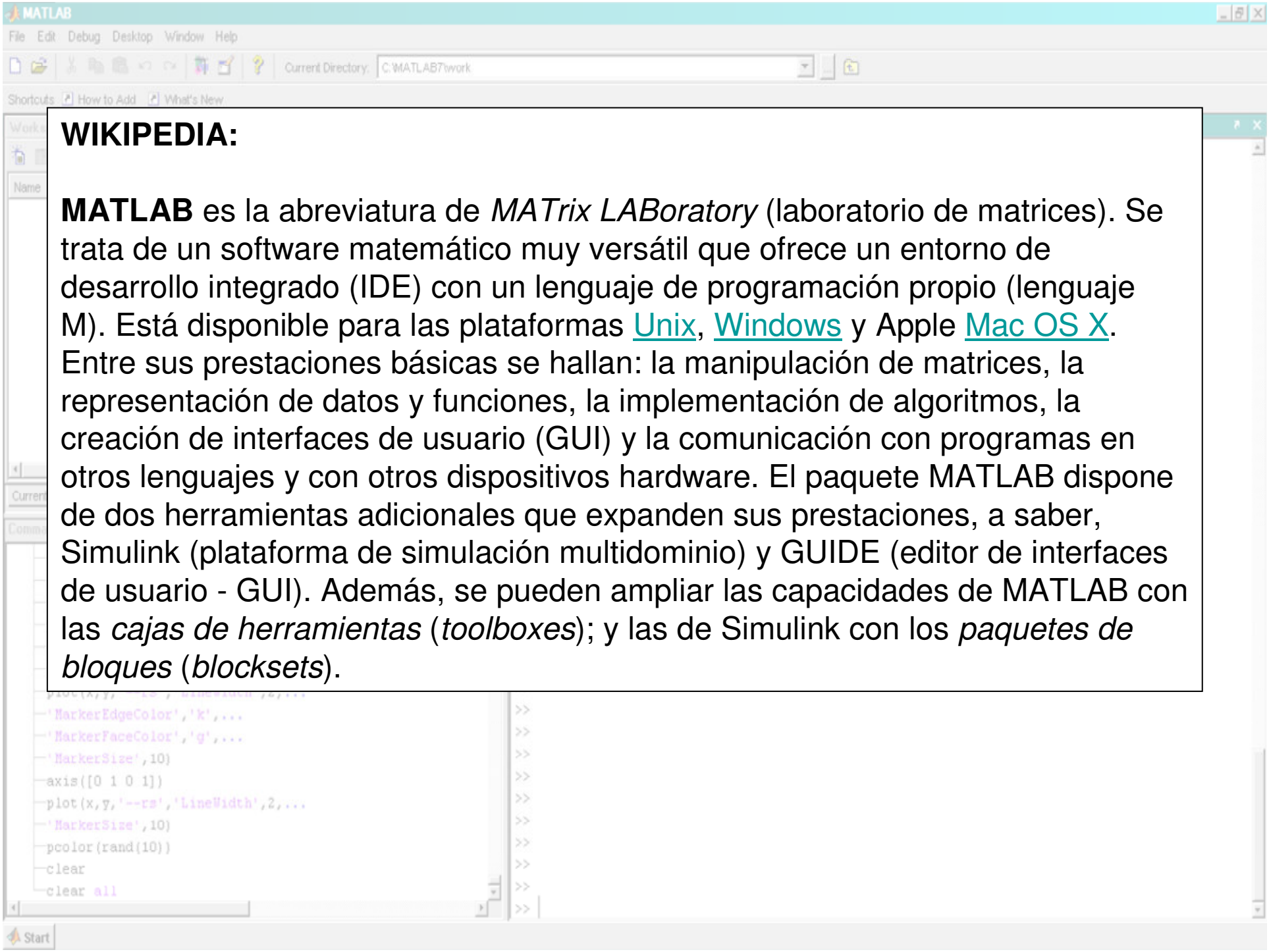


# Tutorial de MATLAB

## Curso Intensivo sobre Asimilación de Datos

### Buenos Aires 2008



**WIKIPEDIA:**

**MATLAB** es la abreviatura de *MATrix LABoratory* (laboratorio de matrices). Se trata de un software matemático muy versátil que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las *cajas de herramientas* (*toolboxes*); y las de Simulink con los *paquetes de bloques* (*blocksets*).

Matlab tiene muchas capacidades y posibilidades. En esta breve introducción queremos resumir los comandos básicos que vamos a utilizar en las aplicaciones del curso.

## Definir variables numericas:

| *definir E* | *definir u* | *modificar E* | *multiplicar Eu* |
|---|---|---|---|
| $E = \text{eye}(3)$ | $u = E(:,1)$ | $E(3,1)=5$ | $v = E*u$ |
| $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 0 & 1 \end{bmatrix}$ | $\begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix}$ |

eye es una función de matlab que calcula la matriz identidad del tamaño n (donde n es el número que va entre paréntesis). Algunas funciones similares son:

```
>>rand(n,m)    %Crea una matriz de numeros aleatorios entre 0 y 1 de tamaño nxm
>>ones(n,m)    %Idem pero una matriz de 1.
>>zeros(n,m)   %Matriz de ceros
>>NaN(n,m)     %Matriz de NaN (not a number).
```

## Cómo introducir un vector de filas o de columnas

$u$ = [2 4 5] tiene una fila con tres elementos (matriz de 1 x 3).

$v$ = [2; 4; 5] tiene tres filas separadas por puntos y comas (matriz de 3 x 1).

$v$ = [2 4 5]' o $v = u'$ transpone $u$ para generar la misma $v$.

$w$ = 2:5 define el vector de filas $w$ = [2 3 4 5] mediante valores que aumentan sucesivamente en una unidad.

$u$ = 1:2:7 asigna valores que aumentan en dos unidades para obtener $u$ = [1 3 5 7]

## Cómo definir una matriz (introduciendo las filas una por una)

$A$ = [1 2 3; 4 5 6] tiene dos filas (el punto y coma siempre separa unas filas de otras).

$A$ = [12 3

4 5 6] también genera la matriz $A$, pero es más difícil de escribir.

$B$ = [1 2 3; 4 5 6]' es la **transpuesta** de $A$. Así pues, $A^T$ es A' en MATLAB.

Matlab tiene integradas muchas funciones programadas en forma eficiente y que agilizan la programación de algoritmos más complicados. Un ejemplo de esto es la inversión de matrices que permite la resolución de sistemas lineales.

| *definir A* | *definir b* | *invertir A* | *Resolver Ax=b* |
|---|---|---|---|
| $A = \text{ones}(3) + \text{eye}(3)$ | $b = A(:,3)$ | $C = \text{inv}(A)$ | $x = A\backslash b$ o $x = C*b$ |
| $\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$ | $\begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$ | $\begin{bmatrix} .75 & -.25 & -.25 \\ -.25 & .75 & -.25 \\ -.25 & -.25 & .75 \end{bmatrix}$ | $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ |

En este caso, si queremos calcular la inversa de A no tenemos que programarlo sinó que podemos usar la función que viene con el programa.

Algunas operaciones más:

A’ calcula la transpuesta de A.

A*B es el producto matricial entre A y B (los tamaños de A y B deben satisfacer las condiciones para que este producto se pueda calcular).

A.*B es el producto interno o miembro a miembro entre A y B, para esto A y B deben tener la misma dimensión.

A(:,1) me da la primera columna de A.

A(1,:) me da la primera fila de A.

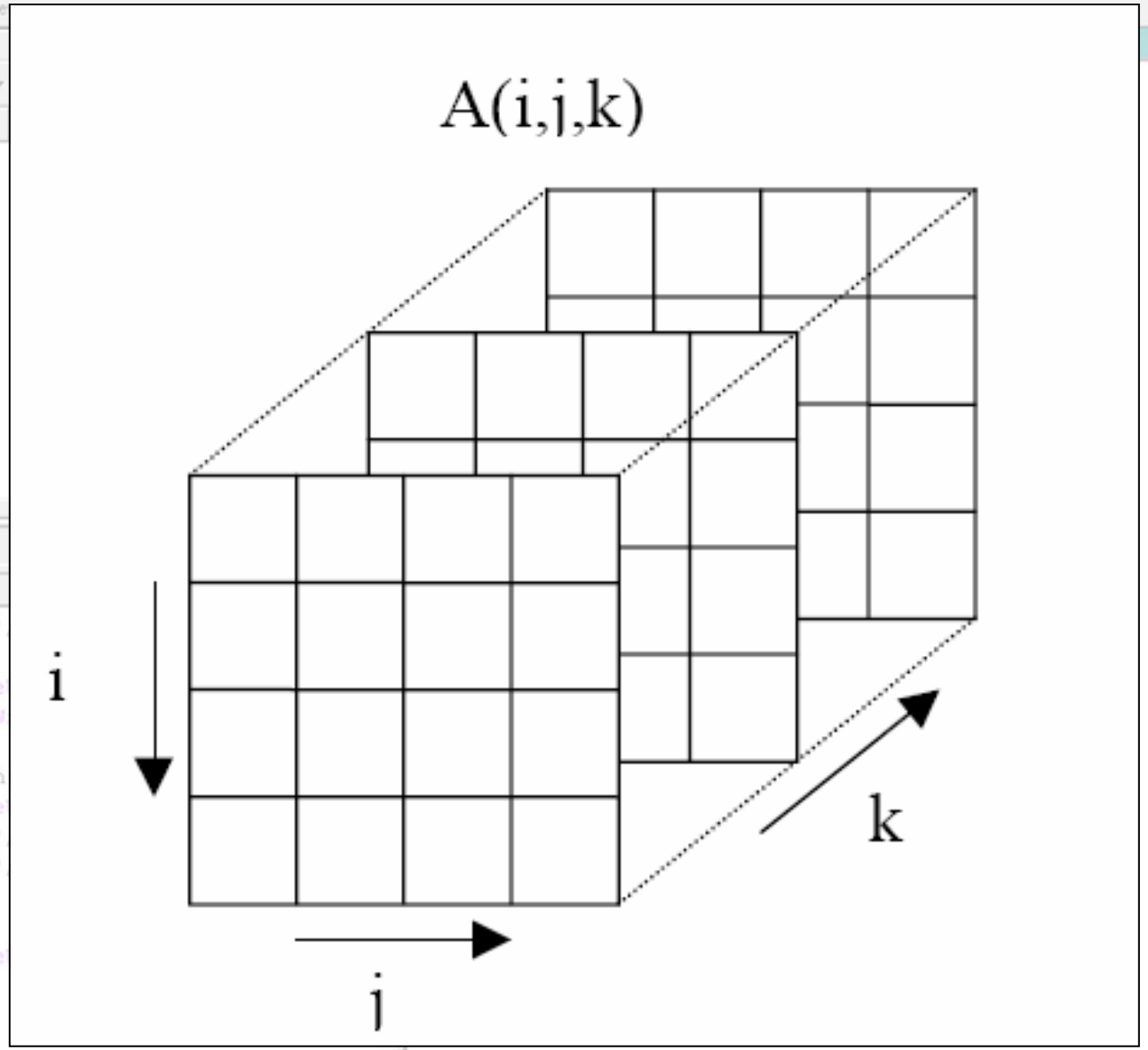
A(1,2:4) me da las columans 2, 3 y 4 de A.

A(1,[2 4]) me da las columnas 2 y 4 de A.

A puede tener más de 3 dimensiones al igual que en fortran.

size(A) me devuelve un vector con el tamaño de cada una de las dimensiones de A.

Ejemplo de matriz tridimensional.



Además de variables numéricas (matrices) el matlab permite definir variables CHARACTER y variables LOGICAS.

**Ejemplo de variable character**

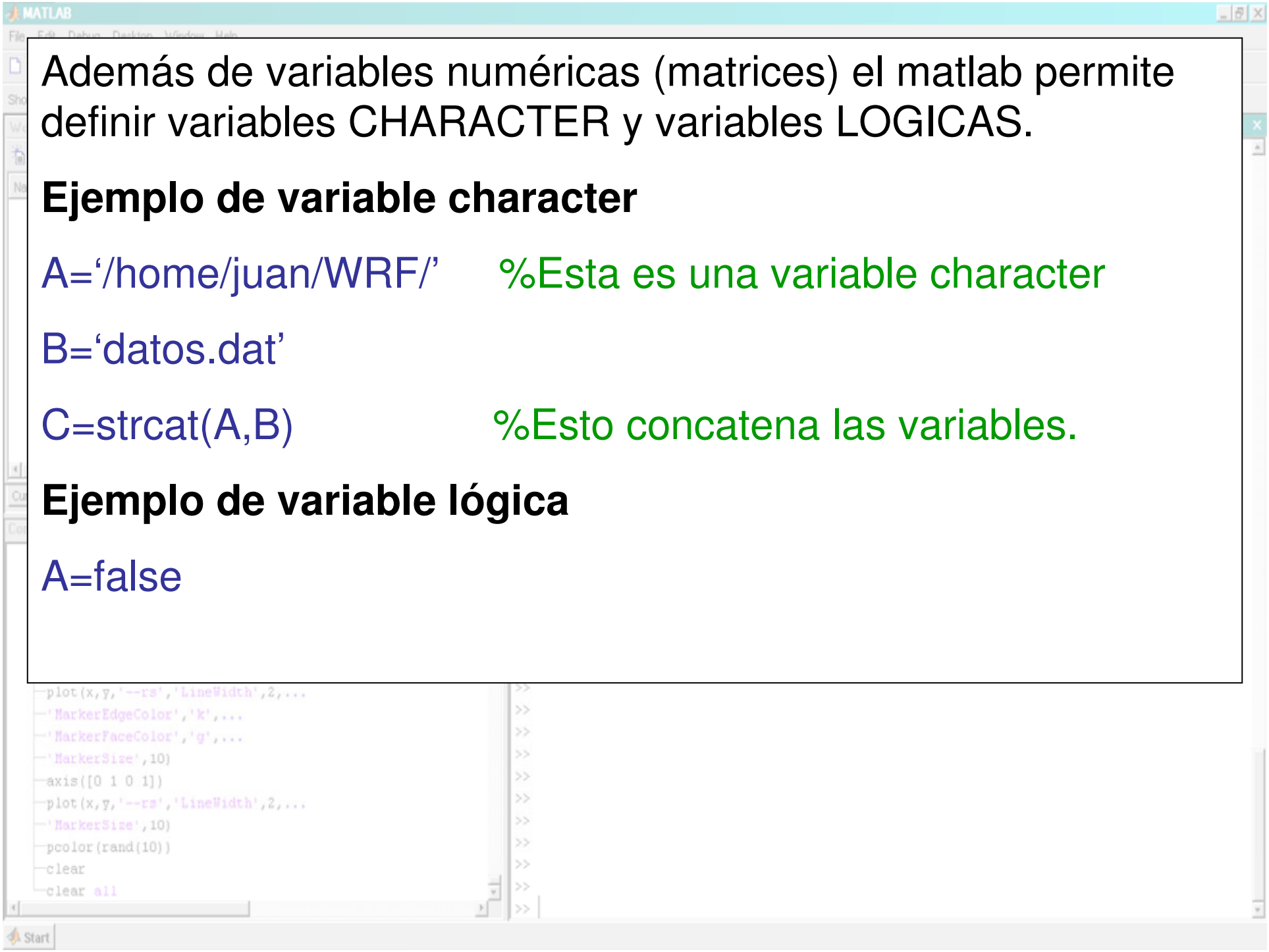
A='/home/juan/WRF/' %Esta es una variable character

B='datos.dat'

C=strcat(A,B) %Esto concatena las variables.

**Ejemplo de variable lógica**

A=false



# Funciones mátemáticas disponibles….

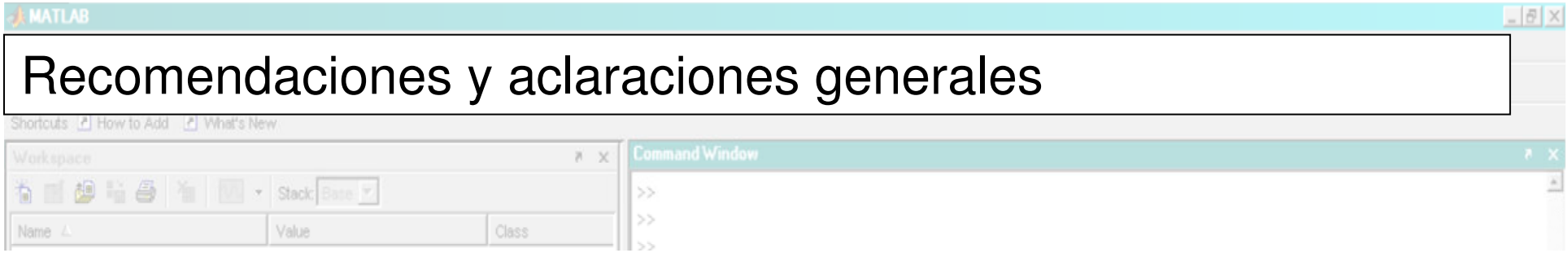
| | |
|---|---|
| sin(x) | seno |
| cos(x) | coseno |
| tan(x) | tangente |
| asin(x) | arco seno |
| acos(x) | arco coseno |
| atan(x) | arco tangente (devuelve un ángulo entre $-\pi/2$ y $+\pi/2$) |
| atan2(x) | arco tangente (devuelve un ángulo entre $-\pi$ y $+\pi$); se le pasan 2 argumentos, proporcionales al seno y al coseno |
| 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 |
| log(x) | logaritmo natural |
| log10(x) | logaritmo decimal |
| exp(x) | función exponencial |
| sqrt(x) | raíz cuadrada |
| sign(x) | devuelve -1 si <0, 0 si =0 y 1 si >0. Aplicada a un número complejo, devuelve un vector unitario en la misma dirección |
| rem(x,y) | resto de la división (2 argumentos que no tienen que ser enteros) |
| mod(x,y) | similar a ***rem*** (Ver diferencias con el ***Help***) |
| 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 $-\infty$ |
| ceil(x) | valor entero más próximo hacia $+\infty$ |
| 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 |

# Funciones que actúan sobre vectores / matrices.

Las siguientes funciones ***sólo actúan sobre vectores*** (no sobre matrices, ni sobre escalares):

| | |
|---|---|
| [xm,im]=max(x) | máximo elemento de un vector. Devuelve el valor máximo **xm** y la posición que ocupa **im** |
| min(x) | mínimo elemento de un vector. Devuelve el valor mínimo y la posición que ocupa |
| sum(x) | suma de los elementos de un vector |
| cumsum(x) | devuelve el vector suma acumulativa de los elementos de un vector (cada elemento del resultado es una suma de elementos del original) |
| mean(x) | valor medio de los elementos de un vector |
| std(x) | desviación típica |
| prod(x) | producto de los elementos de un vector |
| cumprod(x) | devuelve el vector producto acumulativo de los elementos de un vector |
| [y,i]=sort(x) | ordenación de menor a mayor de los elementos de un vector **x**. Devuelve el vector ordenado **y**, y un vector **i** con las posiciones iniciales en **x** de los elementos en el vector ordenado **y**. |

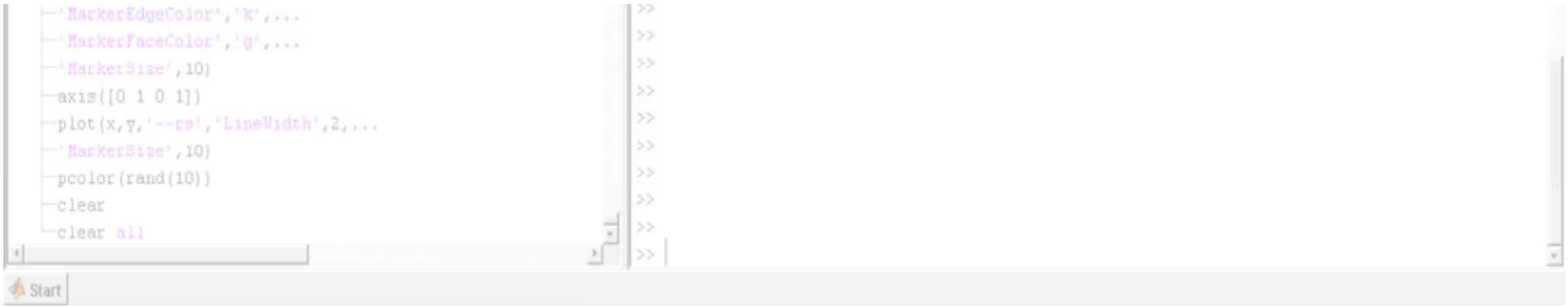
# Recomendaciones y aclaraciones generales




- Los símbolos *a* y *A* son **diferentes**: MATLAB distingue por defecto entre unos casos y otros.
- Escribir *help slash* para obtener una explicación del modo de utilizar el símbolo de la barra inversa. La palabra *help* (ayuda) puede ir seguida de un símbolo o del nombre de un comando o de un archivo (de extensión .m) de MATLAB.

Nota: El nombre del comando aparece con una mayúscula inicial en la explicación que da *help*, pero debe escribirse en minúsculas al utilizarlo. La barra inversa *A\b* actúa de forma distinta cuando *A* no es cuadrada.

- Para ver los números con 16 dígitos, escribir *format long* (formato largo). El formato normal, *format short* (formato corto), muestra 4 dígitos decimales.
- Si se pone un punto y coma tras un comando, el programa no mostrará su resultado. *A* = ones(3); no mostrará la matriz identidad de 3 x 3.
- Utilizar la flecha del desplazamiento hacia arriba del cursor para volver a comandos anteriores.



# Matlab I/O

Matlab puede leer y escribir información en muchos formatos.

A continuación damos algunos ejemplos simples.

```
>>A=rand(3,3)    %Defino una matriz A
>>save    mi_archivo.mat  A        %Guarda la matriz a en el archivo mi_archivo.mat
>>save   mi_archivo.mat     %Guarda todas las variables definidas hasta el momento.
```

El archivo mi_archivo.mat está en un formato definido por matlab.

Para recuperar el valor de Guardado en el archivo, usamos el comando load.

```
>>load mi_archivo.mat    %recupera las variables guardadas en el archivo.
```

Es importante tener en cuenta que si tenemos una variable A y hacemos load la variable A va a ser borrada y sus valores y dimensiones se cambiaran a los valores y dimensiones de la variable A guardada en el archivo.

# Matlab y Netcdf

Las funciones que permiten leer y escribir archivos en formato netcdf utilizando Matlab vienen en un paquete aparte y se instalan por separado.

Las funciones que permiten la lectura y escritura de archivos son las siguientes:

ncload('mi_archivo.nc','variable1','variable2',…) %Permite leer del archivo netcdf las variables variable1 y variable2 (si no se especifica carga todas las variables contenidas en el archivo). Esta instrucción permite leer las variables, pero no los atributos de las mismas (nombres, dimensiones, unidades, etc).

ncsave('mi_archivo.nc','variable1','variable2',…) %Reemplaza el valor de las variables en el archivo netcdf por el de las variables variable1… variable2. Esto permite modificar los valores, pero nuevamente no podemos cambiar las dimensiones ni los atributos de las variables o del archivo.

Para modificar variables, dimensiones y atributos existen otros comandos que no vamos a utilizar en esta materia.

## Lectura y escritura en formato ASCII.

Si el formato del archivo presenta solo números, y está ordenado de forma tal que tiene el mismo número de filas y columnas, entonces podemos leerlo directamente usando el comando LOAD con la opción –ascii

```
>>load –ascii mi_archivo.txt
```

De la misma manera podemos guardar una matriz en un archivo

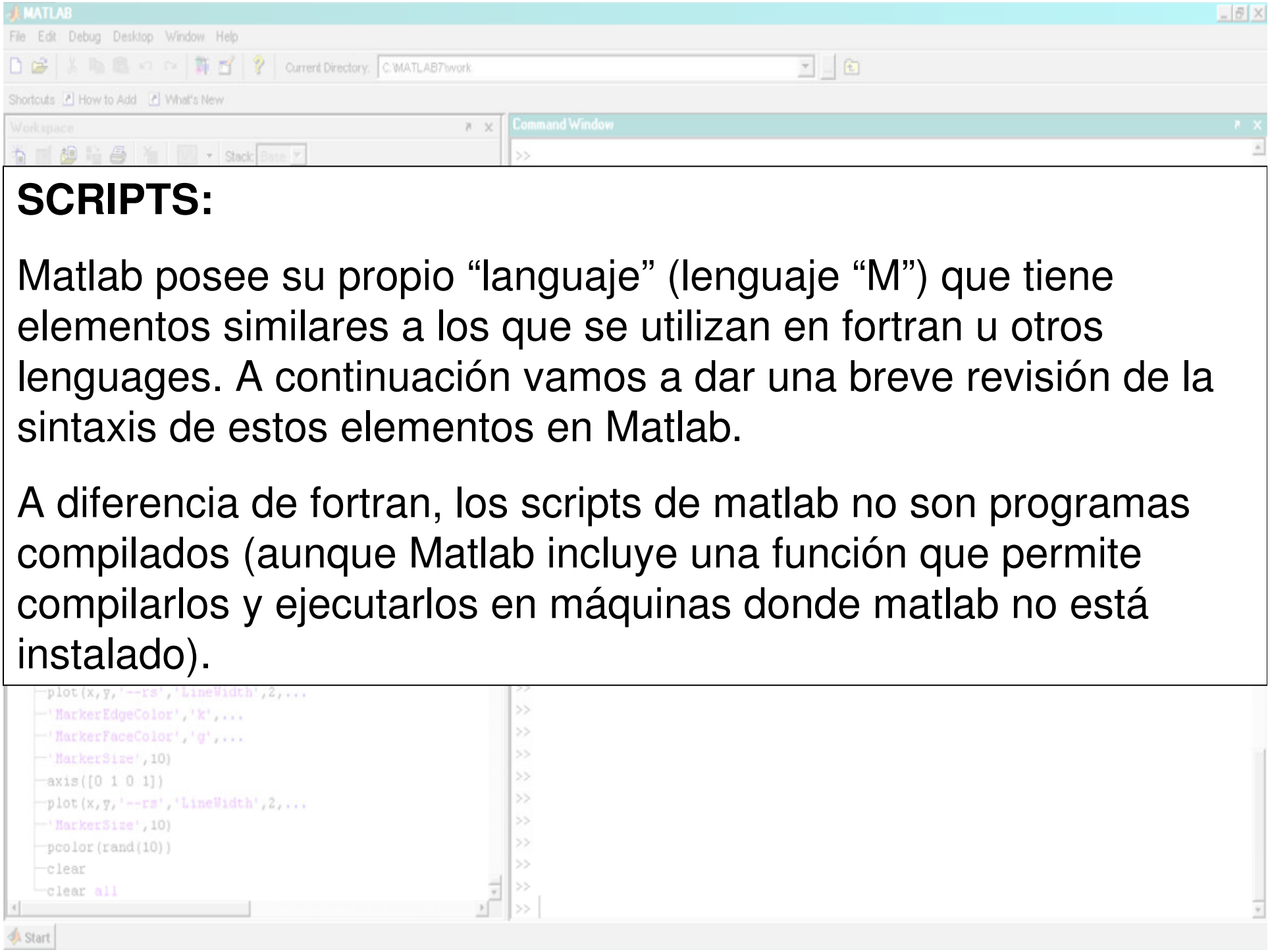
```
>>A=rand(10,10);        %Defino una variable

>>save –ascii mi_archivo.txt A     %Guardo la variable A en forma de texto.
```

Se puede leer archivos con formatos más complicados especificando el formato de lectura. Para eso existen los comandos

fscanf , fprintf

También se pueden leer archivos en formato binario como los generados por un programa fortran utilizando los comandos

fread y fwrite




## SCRIPTS:

Matlab posee su propio “languaje” (lenguaje “M”) que tiene elementos similares a los que se utilizan en fortran u otros lenguages. A continuación vamos a dar una breve revisión de la sintaxis de estos elementos en Matlab.

A diferencia de fortran, los scripts de matlab no son programas compilados (aunque Matlab incluye una función que permite compilarlos y ejecutarlos en máquinas donde matlab no está instalado).

IF

```
>>If( a== 1)
>>   comandos si verdadero
>>end
```

Pruebas lógicas:

== (igual), >, <, <=, >=, ~= (distinto).

Por otra parte podemos combinar pruebas lógicas usando & (and) y | (or).

```
If(a==1 & b==2)      o  bien if (a==1 | b==2)
```

También podemos usar el ~ para negar la prueba

```
If ~(a==1)
Comandos si verdadero
end
```

En este caso la condicion resultará cierta si a no es 1.

else y elseif tambíen son comandos relacionados.

MATLAB

Ciclos for

```
for i=1:30
A(i)=2*i;
end
```

La sintaxis es muy similar a la del fortran.

Se puede variar el incremento en el for:

```
for i=1:2:30
A(i)=2*i;
end
```

En este caso i aumenta de 1 a 30, pero saltando de 2 en dos 1 3 5 …

El comando while también puede ser utilizado para repetir una operación mientras se cumpla una condición

```
while condicion
Repeticion
end
```

La condicion se escribe de la misma manera que en el caso del comando if.

Start

Funciones:

Matlab brinda la posibilidad de definir nuestras propias funciones. Esto es algo análogo a lo que serían por ejemplo las subrutinas de un programa fortran.

Ejemplo:

```
%Funcion de calculo de las derivadas espaciales por diversos metodos.
function [derivada]=diff_sh(variable,dx)

nx=length(variable);

%Para los puntos interiores.
for i=2:nx-1
derivada=(variable(i+1)-variable(i-1))/(2*dx);

%Veo que pasa en los bordes. Uso esquema atrasado y adelantado.
derivada(1)=(variable(2)-variable(1))/dx;
derivada(nx)=(variable(nx)-variable(nx-1))/dx;

end
```

Para usar la funcion uso:

```
dudx=diff_sh(u,0.5)
```

# Graficado en Matlab

Graficos de linea (graficado de un vector)

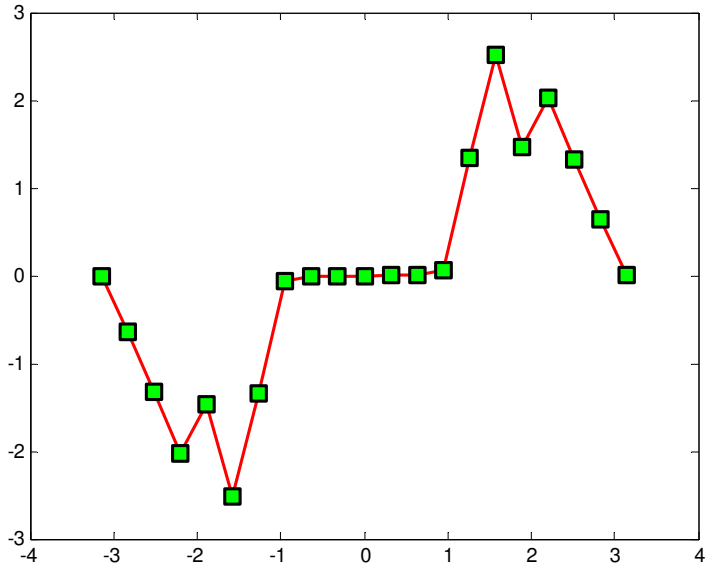
plot(x)  %Plotea el vector X como una serie de valores.

plot(x,y)  %Plotea los valores del vector y como función de los valores de x (x e y deben tener la misma longitud).

plot(X,Y,'LineWidth',2,'Color',[.6 0 0])  %Podemos agregar atributos que controlan el espesor de la línea, el color, el estilo etc. Algunas de estas cosas se pueden abreviar.

**Ejemplo**

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
```



axis([0 1 0 1])  %Controla el rango de los ejes xmin xmax ymin ymax

Si luego de hacer un gráfico queremos graficar otra curva, el gráfico orginal desaparece. Para eso debemos usar la instrucción “hold on” que nos permite graficar varias curvas en el mismo gráfico.

Una vez generado el gráfico podemos ajustar muchas cosas manualmente utilizando el menú de la figura.

A diferencia de GrADS, Matlab permite modificar y remodificar varios de los atributos del gráfico luego de que este fue generado.

title(‘Titulo del grafco’)  %El comando title permite agregar titulos a los graficos.

legend(‘datos 1’,’datos 2’) %El comando legend genera un recuadro con los colores correspondientes a las lineas y nos permite nombrar cada una de ellas.

xlabel(‘etiqueta del eje X')  %Permite nombrar al eje x (lo mismo con ylabel para el eje y).

print ('-dpng',’mi_figura.png’)  %El comando print permite generar un png, gif, jpg, pdf, tiff entre otros a partir de la figura que estamos generando.

Las figuras también pueden guardarse en el formato fig de matlab que permite seguir modificándolas.

# Graficado de superficies:

Comandos para graficar superficies en 2D:

pcolor(x,y,variable)  %Plotea la variable "variable" como funcion de x y de y. Variable debe ser una matriz cuadrada, x e y pueden ser matrices o vectores.

Este comando genera bordes negros entre celda y celda que se eliminan usando el comando "shading flat" o "shading interp"

contour(x,y,variable) %idem pero grafico de contornos.

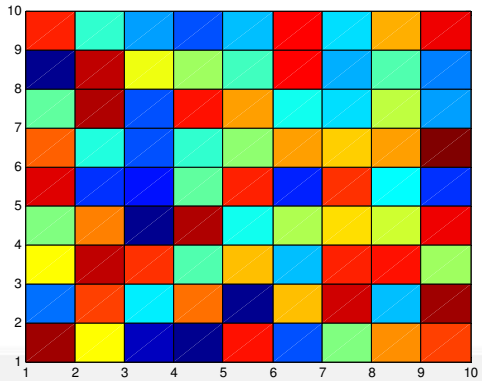
contourf(x,y,variable) %idem pero grafico de contornos con sombreado.

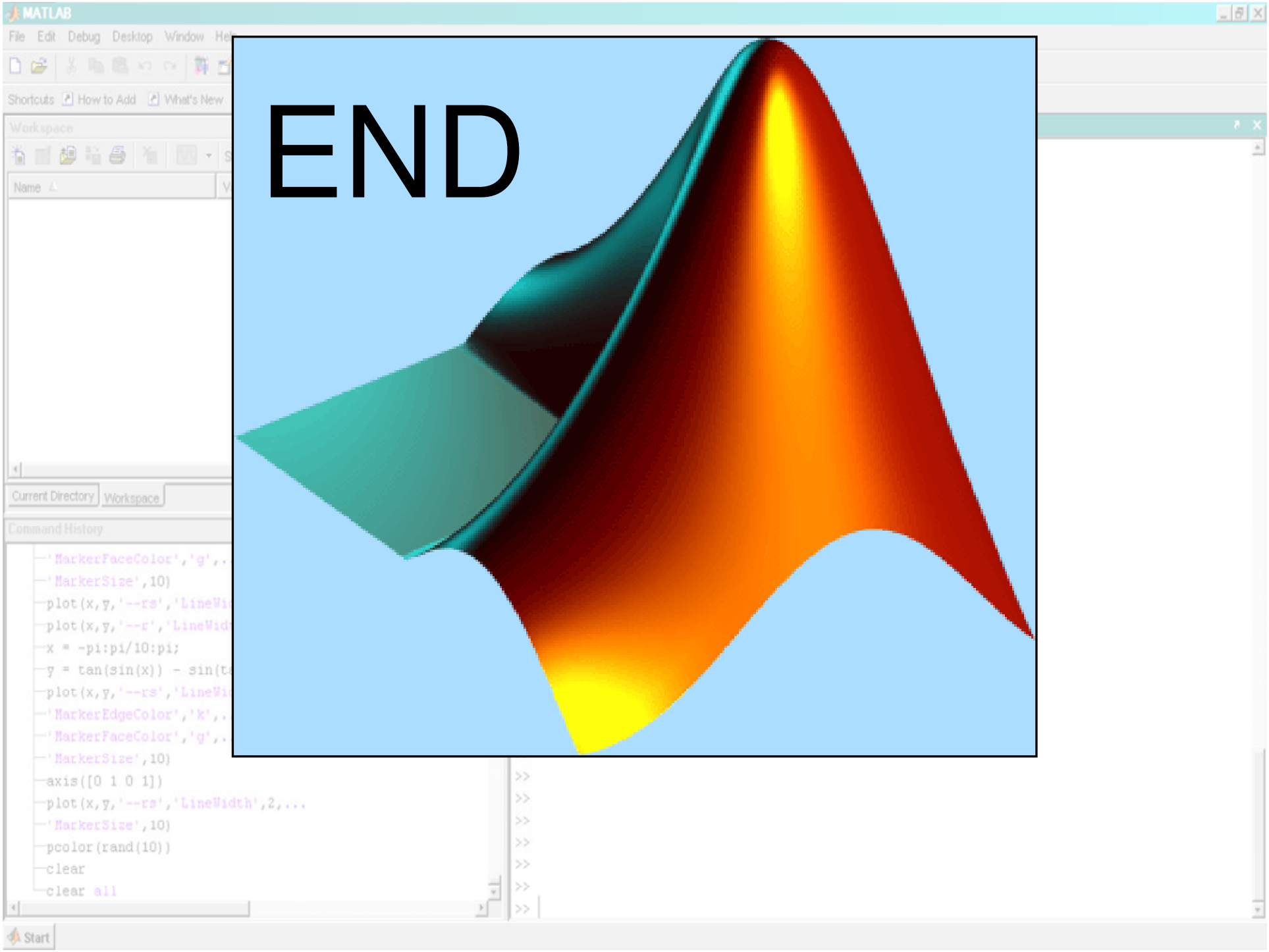
Para graficar una variable georeferenciada debemos tener matrices lat y lon con las coordenadas de cada punto y la matriz "variable" correspondiente a nuestros datos.

pcolor(lon,lat,variable)  %Plotea la matriz como función de la latitud y longitud de nuestros datos.

La función quiver permite plotear vectores.

"run colorbar" despliega la barra de colores una vez que hemos dibujado la matriz de datos.




END
MATLAB
File Edit Debug Desktop Window
Shortcuts How to Add What's New
Workspace
Name
Current Directory Workspace
Command History
'MarkerFaceColor','g',..
'MarkerSize',10)
plot(x,y,'--rs','LineWi
plot(x,y,'--r','LineWid
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(t
plot(x,y,'--rs','LineWi
'MarkerEdgeColor','k',..
'MarkerFaceColor','g',..
'MarkerSize',10)
axis([0 1 0 1])
plot(x,y,'--rs','LineWidth',2,...
'MarkerSize',10)
pcolor(rand(10))
clear
clear all
>>
Start