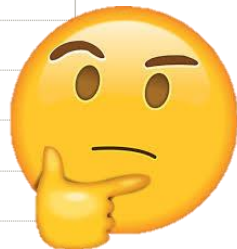
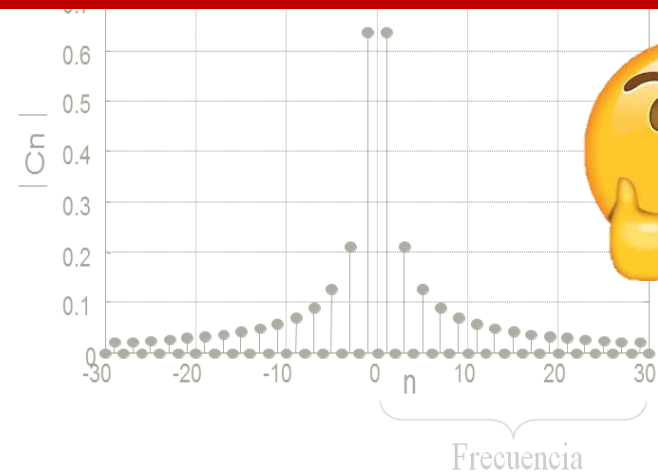
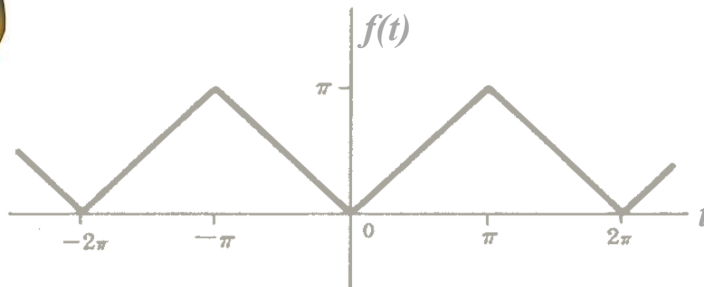


Actividad Práctica

● Introducción a MatLab/Octave ●



El principio de la historia: FORTRAN (1957!!)

Abreviación de IBM Mathematical FORmula TRANslating System

Es un lenguaje de programación de alto nivel de propósito general, procedimental e imperativo, que está especialmente adaptado al cálculo numérico y a la computación científica:

- *Predicción numérica de fenómenos climáticos*
- *Análisis por elementos finitos*
- *Dinámica de fluidos computacional (CFD)*
- *Física computacional y química computacional*

Es uno de los lenguajes más populares en el área de la computación de alto rendimiento y es utilizado en algoritmos que evalúan el desempeño (benchmark) y el ranking de los supercomputadoras más rápidas del mundo

The MATrix LABoratory

MATLAB es un lenguaje de *alto nivel* de computación numérica, especializado en cálculos matriciales. Utiliza el “**Lenguaje M**”, creado en 1970 *para manipular vectores y matrices sin necesidad de utilizar Fortran*

Permite el *análisis de datos*, *desarrollo de algoritmos* y la *creación de modelos* y *aplicaciones*:

- Procesamiento de señales e imágenes
- Sistemas de control
- Estadística Matemática financiera
- Biología computacional
- Testeo y medición en tiempo real
- Desarrollo de Aplicaciones



The MATrix LABoratory

- Trabaja con memoria dinámica, ***no se requiere la declaración de variables***
- Se pueden ***programar funciones***, cuyas variables resultan locales
- La ***representación numérica*** se efectúa a partir de ***doble precisión***: ***16 cifras decimales*** ($2^{-52} = 2,220 \times 10^{-16}$ entre dos números próximos)
- Distingue entre ***mayúsculas*** y ***minúsculas***
- Admite ***números complejos***
- Reconoce ***constantes numéricas***: π , e , inf



Plataformas Similares a MatLab de uso libre:

1. GNU Octave:

El proyecto fue creado alrededor del año 1988, con la finalidad de ser utilizado en un curso de diseño de reactores químicos. Posteriormente, en el año 1992, se decidió extenderlo y comenzó su desarrollo a cargo de J. W. Eaton. Su primera versión (alpha) fue lanzada el 4 de enero de 1993.

2. SCILAB:

Comienza en los años 80, con Blaise, desarrollado principalmente por F. Delebecque y Serge Steer con el objetivo de proporcionar una herramienta en control automático para investigadores. Fue inspirado por el software Matlab Fortran desarrollado por Cleve Moler quien más tarde cofundó con John Little la compañía "MathWorks".

Análisis Numérico: MatLab/Octave

Introducción

Análisis de Señales y Sistemas R2041

Entornos de Trabajo: MatLab

The screenshot shows the MATLAB R2013a environment. The interface is divided into several panes, each labeled with a blue box:

- CONTROL DE LA PLATAFORMA**: Located in the top right corner, above the Workspace pane.
- EDITOR**: The central pane containing the MATLAB script. The script defines parameters for a Windkessel model, including peripheral resistance (RP), arterial compliance (CA), and time step (dt). It uses the `ode45` function to solve the differential equations and `subplot` to plot the results.
- CARPETA DE TRABAJO**: The Current Folder pane on the left, showing the file structure of the current project.
- VENTANA DE COMANDO**: The Command Window at the bottom left, where MATLAB commands are entered.
- WORKSPACE**: The Workspace pane on the right, displaying the variables currently in memory, such as CA, PAO, PAO0, RP, Tfinal, dt, t, and tspan.

The script in the Editor pane is as follows:

```
18 %Resistencia Periferica RP [mmHg.s^2/cm^3]
19 %Compliance Arterial CA [cm^3/mmHg.s]
20 global RP CA
21 RP=1; CA=1;
22 %Incremento temporal
23 dt=0.01;
24
25 %Respuesta temporal Modelo WK2-----
26 %Condiciones inicales de la EDO
27 PAO0=80;
28 %Tiempo de análisis
29 Tfinal=8;
30 tspan=0:dt:Tfinal;
31 %Resolución de la EDO
32 [t PAO]=ode45('ODE_WK2',tspan,PAO0);
33
34 %Visualización Entrada - Salida-----
35 subplot(211),plot(t,FUN_Q(tspan),'r');grid on;
```

Entorno de Trabajo: MatLab/Octave

- ***Command Window***

Permite escribir y ejecutar comandos

- ***Workspace/Variable browser***

Visualiza las variables en memoria

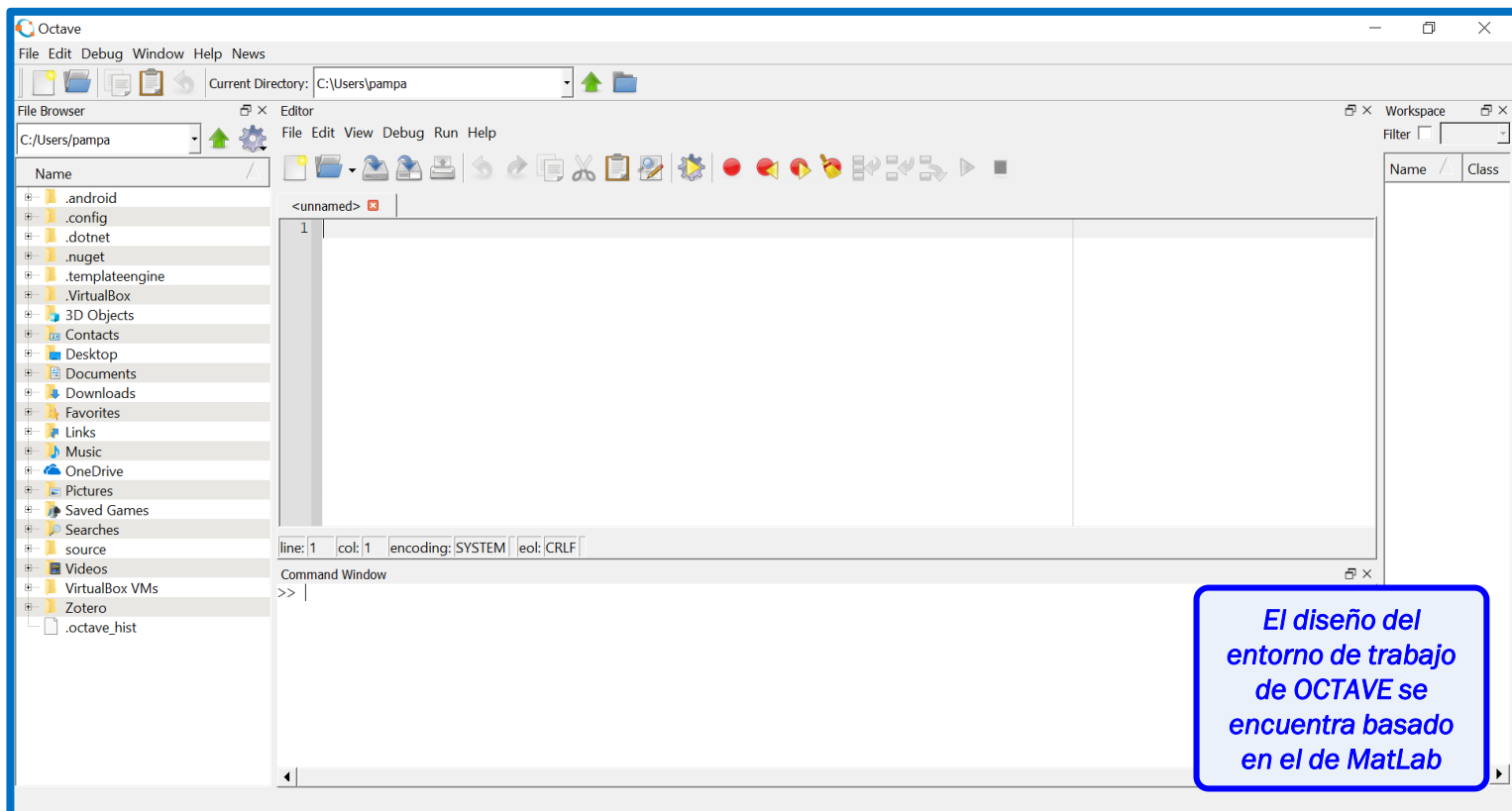
- ***Current Folder / File Browser***

Lista las carpetas y archivos en la carpeta de trabajo

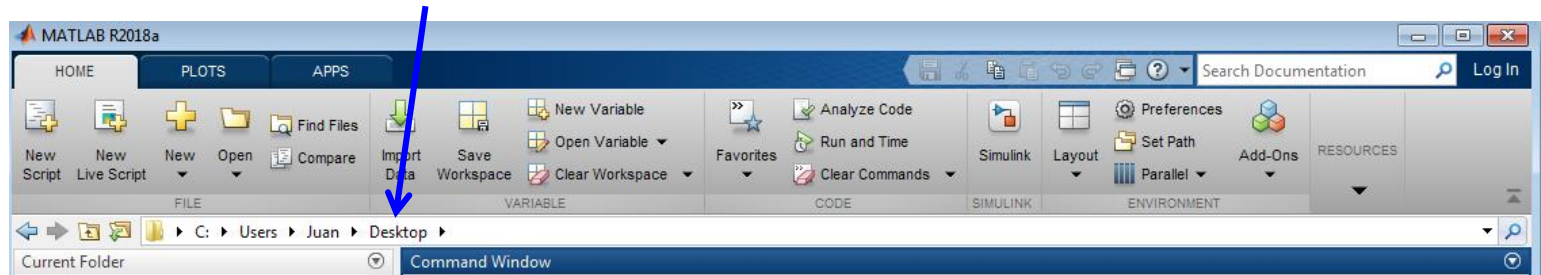
- ***Editor***

Permite generar scripts o funciones, para luego ser llamados por el command window

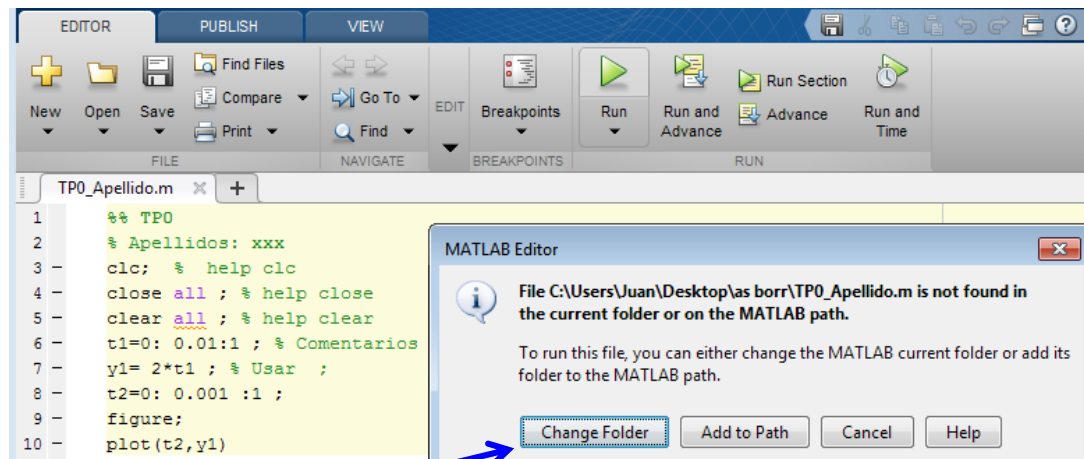
Entornos de Trabajo: GNU Octave



MatLab/Octave: Path Actual (carpeta de trabajo)



Cambio de Path



MatLab/Octave: Herramientas de Trabajo (ToolBoxes)

Tanto en MatLab como en Octave se dispone de **conjuntos de herramientas denominadas “Toolboxes”**, las cuales proporcionan funciones específicas **según la disciplina de trabajo**, cuya amplitud comprende desde el **procesamiento de señales e imágenes** hasta la **biología computacional**.

Algunas funciones tales como **mean**, **max**, **sum** o **sort**, entre otras, están directamente **incorporadas al kernel de la plataforma**.

Cualquier función perteneciente a un TOOLBOX es ejecutable desde la ventana de comandos, independientemente de la carpeta en la que se esté trabajando. A efectuar un llamado, la plataforma detecta la ubicación de dicha función según una “ruta de búsqueda del sistema” (listado de carpetas existentes denominado “SEARCH PATH”)

MatLab/Octave: La función “help”

La función help **proporciona información** acerca de cualquier función existente en la plataforma. Ejemplo: **help plot** (ayuda sobre función “plot”) o **help find** (ayuda sobre la función “find”)

```
Command Window

>> help plot
plot - 2-D line plot

This MATLAB function creates a 2-D line
corresponding values in X.

plot(X,Y)
plot(X,Y,LineSpec)
plot(X1,Y1,...,Xn,Yn)
```

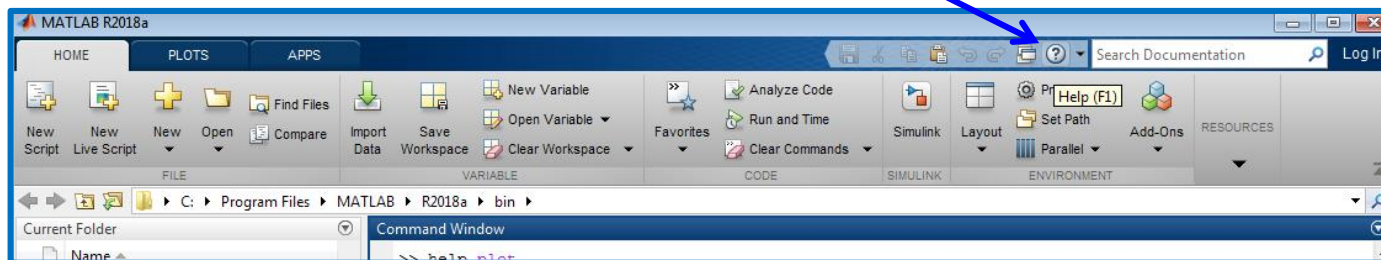
```
>> help find
find - Find indices and values of

This MATLAB function returns
nonzero element in array X.

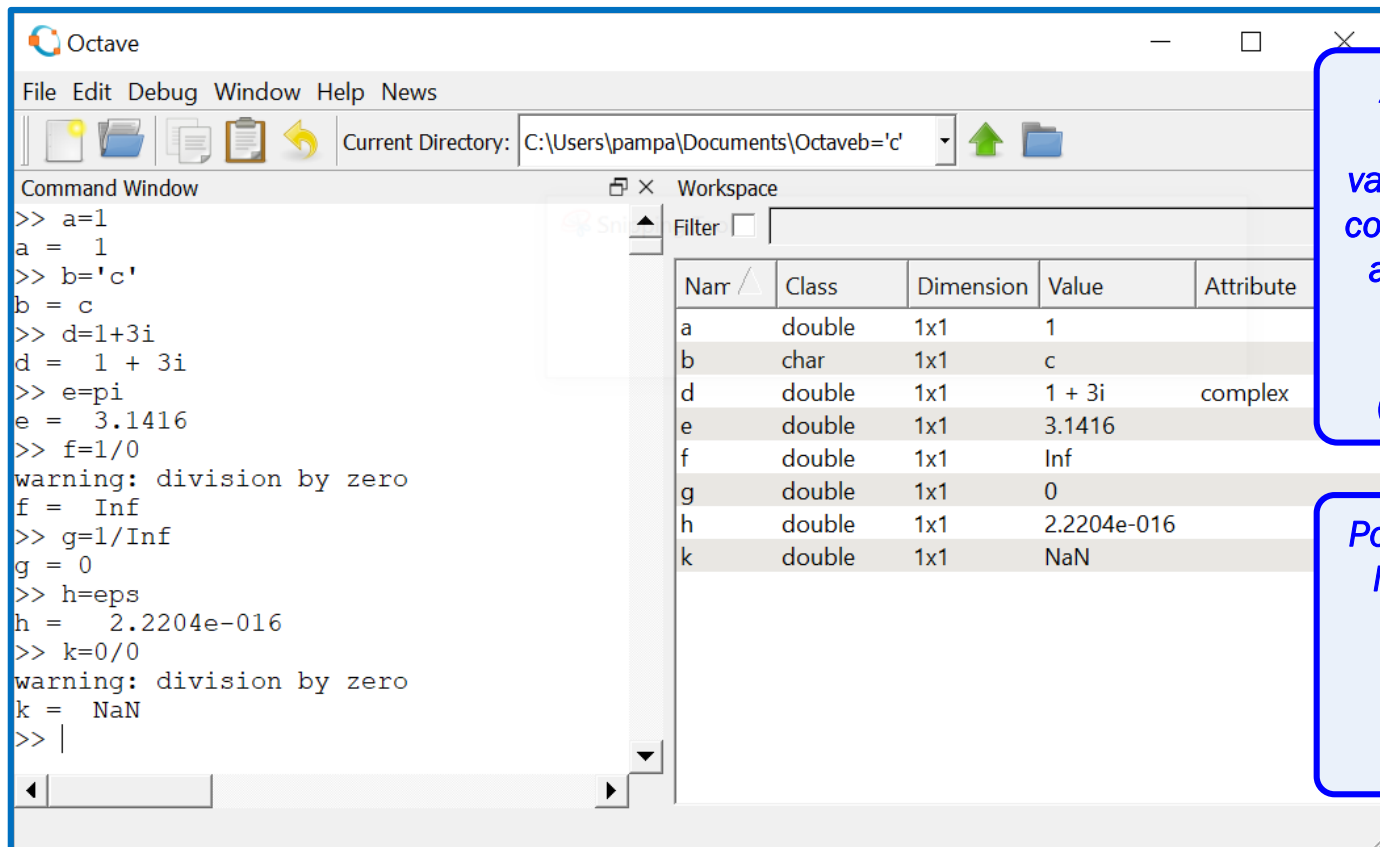
k = find(X)
k = find(X,n)
k = find(X,n,direction)
[row,col] = find(____)
```

**Ver parámetros de
entrada y de salida
[row,col] = find(____)**

Otra manera: F1 (Se obtienen ejemplos sumamente detallados)



MatLab/Octave: Utilización de la Ventana de Comandos



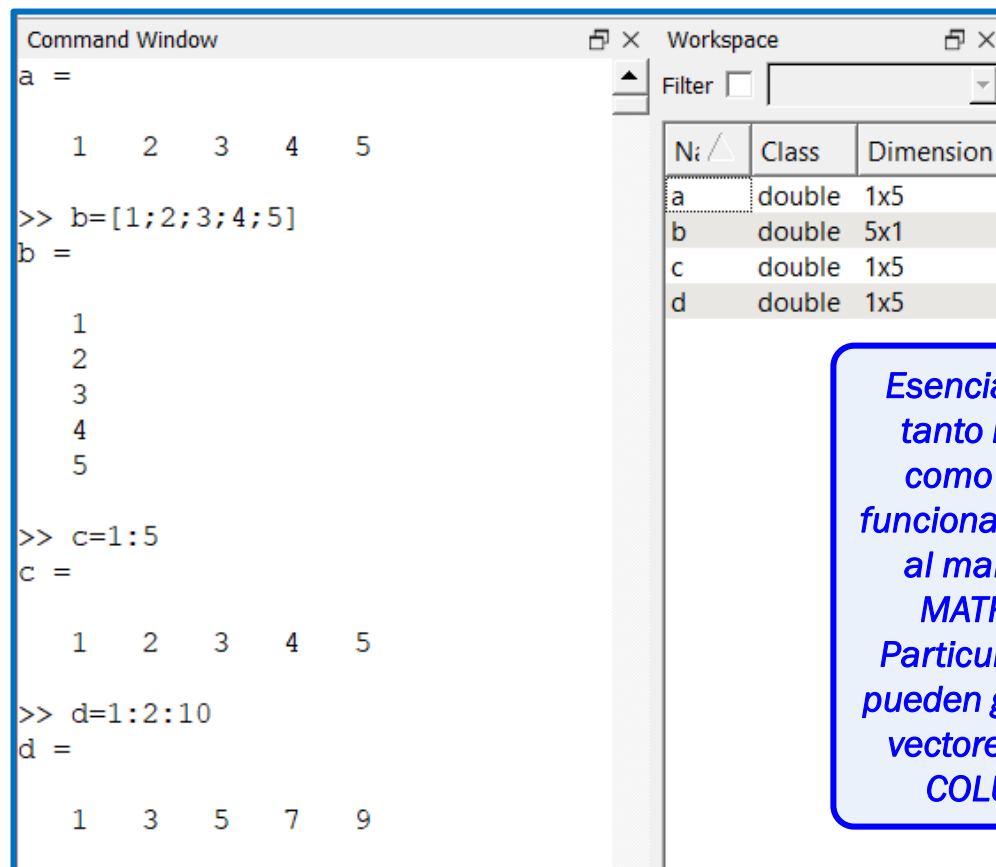
Al efectuar un comando, la variable generada como resultado se almacena en el espacio de variables (WORKSPACE)

Por defecto, todos los valores son DOBLE PRECISIÓN, excepto para caracteres

MatLab/Octave: Utilización de la Ventana de Comandos

Comando	Definición	Ejemplo
<code>;</code>	No visualiza el resultado por consola al ejecutar una instrucción	<code>a=1;</code>
<code>format</code>	Indica el formato numérico de <u>visualización</u> de los resultados (las operaciones siempre se efectúan en doble precisión)	<code>format short</code> : 4 decimales <code>format long</code> : 16 decimales <code>format short e</code> : Notación científica
<code>clc</code>	Limpia la línea de comandos	<code>clc</code>
<code>clear</code>	Elimina variables	<code>clear all</code> (todas las variables) <code>clear a</code> (variable específica)
<code>ans</code>	Al ejecutar una instrucción sin asignar el resultado a una variable específica, se crea la variable “ans”	

MatLab/Octave: Vectores



```
Command Window
a =
    1     2     3     4     5

>> b=[1;2;3;4;5]
b =
     1
     2
     3
     4
     5

>> c=1:5
c =
     1     2     3     4     5

>> d=1:2:10
d =
     1     3     5     7     9

Workspace
Filter [ ]


| Ni | Class  | Dimension |
|----|--------|-----------|
| a  | double | 1x5       |
| b  | double | 5x1       |
| c  | double | 1x5       |
| d  | double | 1x5       |

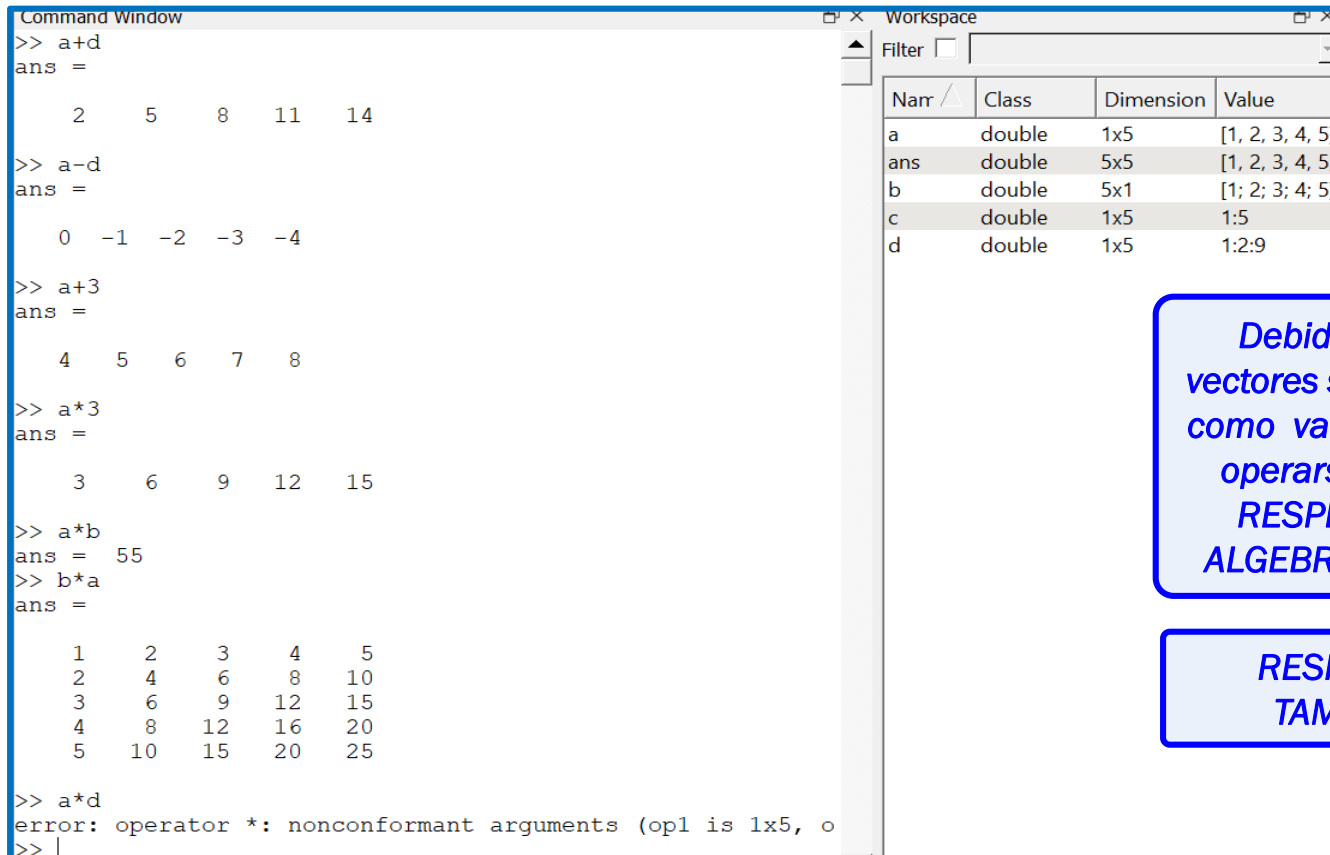

```

*Esencialmente,
tanto MatLab
como Octave
funcionan en base
al manejo de
MATRICES.
Particularmente
pueden generarse
vectores FILA o
COLUMNA*

MatLab/Octave: Vectores

Acción	Definición	Ejemplo
Vector Fila	Vector de $1 \times N$	<code>a=[1,2,3,4,5]</code>
Vector Columna	Vector de $N \times 1$	<code>a=[1;2;3;4;5]</code>
Trasposición: <code>'</code>	Permite transponer el vector	<code>b=a'</code>
Vector incremental INI : inc : FIN	Genera un vector que va desde INI, hasta FIN, en incrementos INC	<code>a=0:0.1:10</code>
Indización	Acceso a los elementos del vector	<code>a(1)</code> Primer elemento <code>a(3)</code> Tercer elemento <code>a(end)</code> Último elemento <code>a(2:4)</code> Subvector

MatLab/Octave: Operaciones Básicas con Vectores



The screenshot displays the MatLab/Octave environment. The Command Window on the left shows a series of operations on vectors `a` (1x5), `b` (5x1), and `d` (1x5). The Workspace window on the right lists these variables with their dimensions and values.

```
Command Window
>> a+d
ans =
     2     5     8    11    14

>> a-d
ans =
     0    -1    -2    -3    -4

>> a+3
ans =
     4     5     6     7     8

>> a*3
ans =
     3     6     9    12    15

>> a*b
ans = 55
>> b*a
ans =
     1     2     3     4     5
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
     5    10    15    20    25

>> a*d
error: operator *: nonconformant arguments (op1 is 1x5, o
>> |
```

Name	Class	Dimension	Value
a	double	1x5	[1, 2, 3, 4, 5]
ans	double	5x5	[1, 2, 3, 4, 5; ...]
b	double	5x1	[1; 2; 3; 4; 5]
c	double	1x5	1:5
d	double	1x5	1:2:9

Debido a que los
vectores se constituyen
como variables, puede
operarse con ellos,
**RESPETANDO EL
ALGEBRA MATRICIAL**

**RESPETAR LOS
TAMAÑOS!!!!**

MatLab/Octave: Operaciones elemento a Elemento

```
Command Window
error: operator *: nonconformant arguments (op1 is 1x5, op2 is 1x5)
>> a.*d
ans =
    1     6    15    28    45

>> a./d
ans =
    1.00000    0.66667    0.60000    0.57143    0.55556

>> a.^d
ans =
     1         8        243       16384      1953125
```

El incorporar el operador "." a la operación entre dos vectores se interpreta que dicha operación debe ser ejecutada entre los elementos de igual índice (operación "elemento a elemento"). Si se utilizan escalares, la operación se aplica a cada elemento del vector

MatLab/Octave: Funciones básicas sobre vectores

Tipo	Definición	Ejemplo
length	Devuelve el tamaño del vector	<code>a=[4,2,1,8,10]</code> <code>length(a)</code>
mean	Devuelve el promedio del vector	<code>mean(a)</code>
linspace INI,FIN,N	Permite generar un vector de N elementos, iniciando en INI y terminando en FIN	<code>t=linspace(0,10,101)</code>
max, min	Devuelve el valor máximo (o mínimo) y su posición dentro del vector	<code>[m, pos]=max(a)</code> <code>[m, pos]=min(a)</code>
find	Devuelve la posición de los elementos del vector que cumplen una determinada condición	<code>find(a>3)</code>

MatLab/Octave: Manejo de Matrices

```
Command Window
>> a=[1,2,3;4,5,6;7,8,9]
a =

     1     2     3
     4     5     6
     7     8     9

>> b=a'
b =

     1     4     7
     2     5     8
     3     6     9

>> c=a*b
c =

    14    32    50
    32    77   122
    50   122   194

>> a.*b
ans =

     1     8    21
     8    25    48
    21    48    81

>> a.^b
ans =

     1    16   2187
    16   3125  1679616
   343  262144 387420489
```

Workspace

Var	Class	Dimension
a	double	3x3
b	double	3x3
c	double	3x3
d	double	3x3

Al definir matrices, las filas se separan con “;” (punto y coma). Al igual que los vectores, las operaciones respetan el álgebra matricial

Asimismo, pueden aplicarse las operaciones elemento a elemento

MatLab/Octave: Funciones básicas sobre matrices

Tipo	Definición	Ejemplo
inv	Genera la matriz inversa	<code>a=[2,5,9,1;8,0,3,6]</code> <code>inv(a)</code>
min, max	Devuelve el valor máximo (o mínimo) y su posición dentro del vector para cada columna (dim=1) o fila (dim=2)	<code>min(a,[],1)</code>
mean	Devuelve el promedio de la matriz para cada columna (dim=1) o fila (dim=2)	<code>mean(a,2)</code>
find	Devuelve la posición (fila, columna) de los elementos de la matriz cumplen una determinada condición	<code>[f,c]=find(a>3)</code>
Indización	El acceso a un valor de la matriz se efectúa a través de su posición fila (f) y columna (c). Puede accederse a submatrices como en el caso de los vectores.	<code>a(1,1)</code> Primer elemento <code>a(2,4)</code> Fila 2, Columna 4 <code>a(2,:)</code> Toda la fila 2 <code>a(1:2,3:4)</code> Submatriz

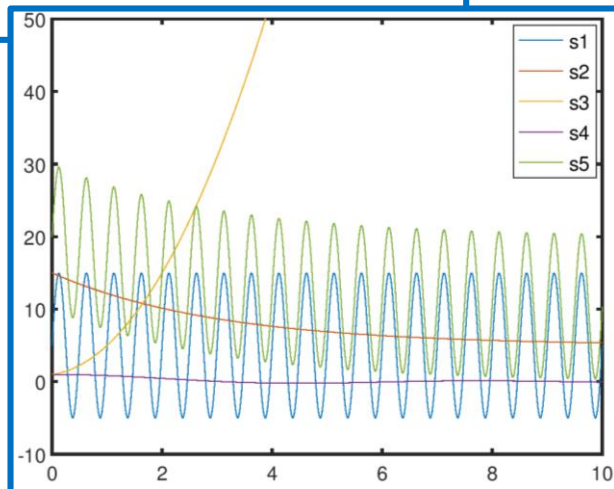
MatLab/Octave: Funciones matemáticas de uso general

Tipo	Definición	Ejemplo
Trigonométricas	$\sin(x)$, $\cos(x)$, $\tan(x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(x)$ (entre otras) y sus inversas $\operatorname{asin}(x)$, $\operatorname{atan}(x)$	<code>a=pi/2;</code> <code>sin(a) , cos(a) , tan(a)</code>
Exponenciales y logarítmicas	$\exp(x)$, $\log_{10}(x)$, $\log_2(x)$, $\log(x)$, $\operatorname{pow}_2(x)$	<code>a=8</code> <code>log2(a)</code>
Números Complejos	$\operatorname{abs}(z)$, $\operatorname{angle}(z)$, $\operatorname{complex}(x,y)$, $\operatorname{conj}(z)$, $\operatorname{imag}(z)$	<code>z=2+3i</code> <code>abs(z) , angle(z) , conj(z)</code>
Matemáticas	$\operatorname{factorial}(n)$, $\operatorname{fix}(x)$, $\operatorname{mod}(x,m)$, $\operatorname{ones}(a,b)$	<code>a=2.58</code> <code>fix(a)</code>

MatLab/Octave: Generación de series temporales

Command Window

```
>> dt=0.01;  
>> t=0:dt:10;  
>> s1=10*sin(2*pi*2*t)+5;  
>> s2=10*exp(-1/3*t)+5;  
>> s3=3*t.^2+t+1;  
>> s4=sin(t)./t;  
>> s5=s1+s2;  
>> |
```



En Matlab, las relaciones funcionales del tipo $y=f(x)$ se definen a partir de DOS VECTORES. En uno se colocan los valores (discretos) de la variable independiente y en el otro el resultado de APLICAR LA FUNCIÓN a dicho vector. Tener en cuenta que no están relacionados, son vectores numéricos por lo que ante un cambio la variable independiente deberá tener que aplicarse nuevamente la función

MATLAB TRABAJA CON VALORES DISCRETOS. LAS FUNCIONES DE VARIABLE CONTINUA PUEDEN “SIMULARSE” A TRAVÉS DE VARIABLES DISCRETAS QUE ÚTILIZAN INCREMENTOS MUY PEQUEÑOS (INFINITESIMALES)

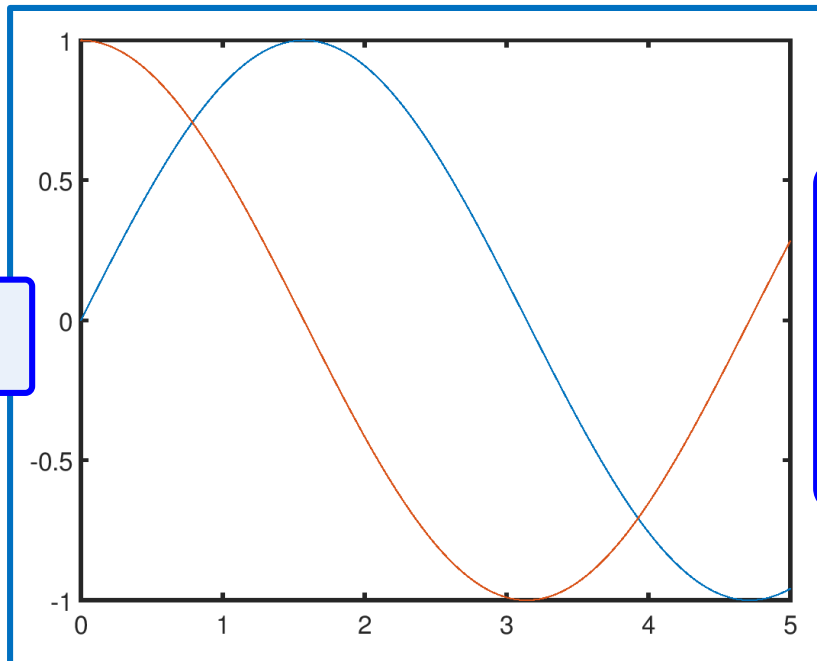


MatLab/Octave: Visualización de series temporales

```
>> t=0:0.001:5;  
>> f1=sin(t);  
>> f2=cos(t);  
>> plot(t,f1,'b',t,f2,'r--');
```

Para graficar una relación funcional, se utiliza el comando "PLOT", indicando primero el vector de la variable independiente y el luego el vector de la variable dependiente (vector).

VENTANA DE
GRAFICACIÓN



Asimismo puede seleccionarse el color, trazo y la presencia de marcadores utilizando indicadores tales como 'b' (trazo azul) o 'r--' (trazo rojo a rayas)

MatLab/Octave: Graficación de series temporales

Specifier	Marker Type
'+'	Plus sign
'o'	Circle
'*'	Asterisk
'.'	Point
'x'	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond
'^'	Upward-pointing triangle
'v'	Downward-pointing triangle
'>'	Right-pointing triangle
'<'	Left-pointing triangle
'pentagram' or 'p'	Five-pointed star (pentagram)
'hexagram' or 'h'	Six-pointed star (hexagram)

TIPO DE
MARCADOR

Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

COLOR

Line Style	Descripción
-	Solid line (default)
--	Dashed line
⋮	Dotted line
-.	Dash-dot line

TIPO DE LINEA

`plot(x,f,'bo:')`

La función PLOT grafica un MARCADOR en los puntos (x,y) definidos por los vectores y LOS UNE con una LINEA

MatLab/Octave: Graficación de series temporales

```
plot(t,f1,'b',t,f2,'r--')
```

Trigonometric functions

```
title('Trigonometric  
functions')
```

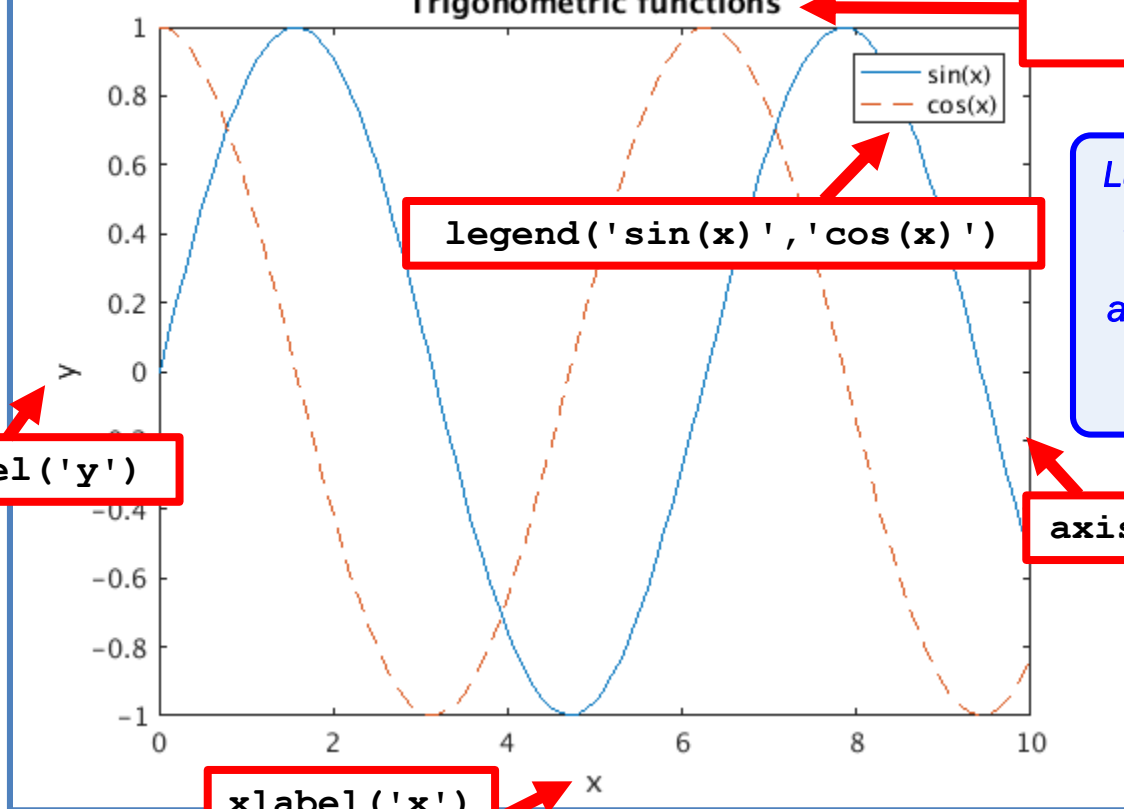
```
legend('sin(x)', 'cos(x)')
```

```
ylabel('y')
```

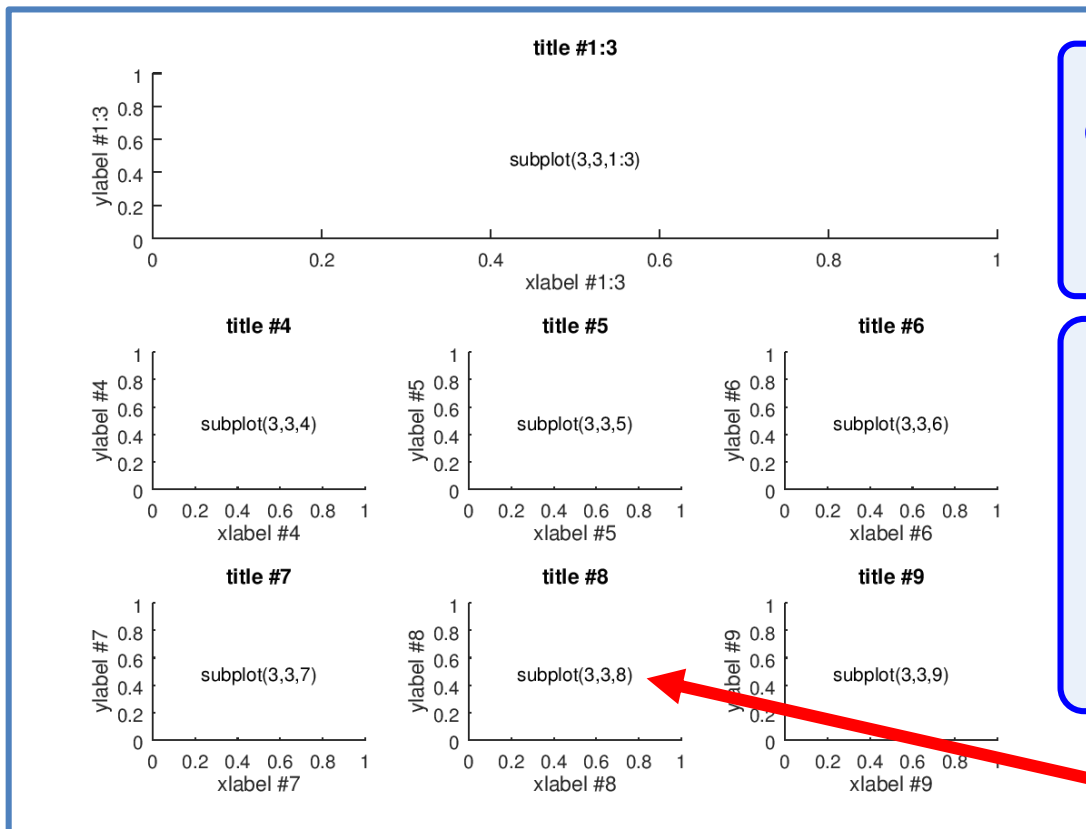
*Luego de llevarse a cabo
la graficación, pueden
ejecutarse comandos
accesorios para agregar
leyendas, títulos y
nombres a los ejes*

```
axis tight
```

```
xlabel('x')
```



MatLab/Octave: Graficación de series temporales

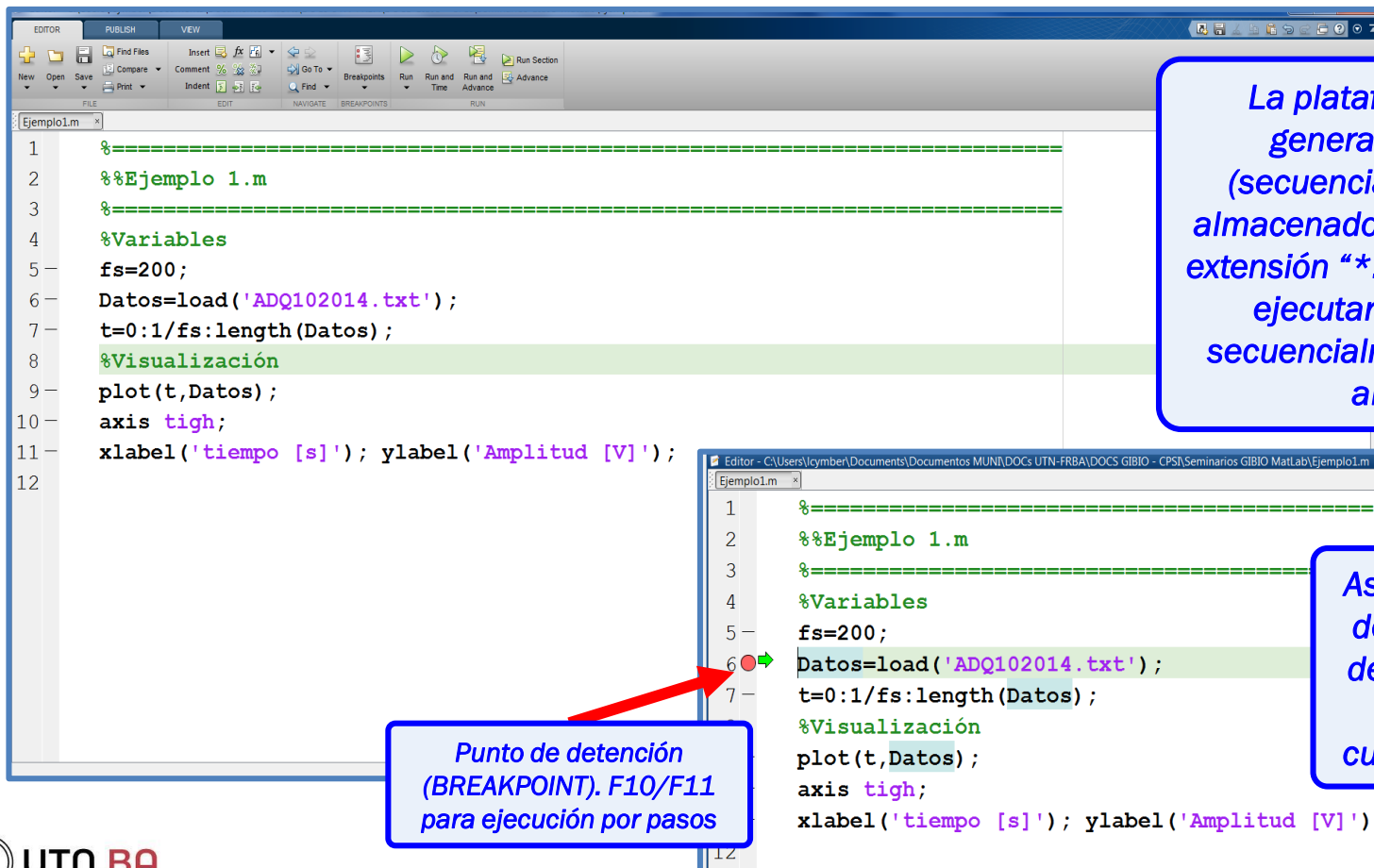


A través del comando **SUBPLOT** (previo a la ejecución de un comando **PLOT**) puede subdividirse la ventana de modo de graficar varios ejes en forma simultánea

La ventana de visualización debe pensarse como una matriz. El comando "**subplot (f,c,x)**" define una cantidad de **fxc** ejes (en el caso de la figura $3 \times 3 = 9$), donde la posición de cada eje se identifica con la numeración "**x**" (para el ejemplo, de 1 a 9, de izquierda a derecha

```
subplot(3,3,8);  
plot(t,f1);
```

MatLab/Octave: Generación de algoritmos (.m)



La plataforma admite la generación SCRIPTS (secuencias de comandos almacenados en un archivo de extensión “*.m”), de manera de ejecutar los comandos secuencialmente, a modo de algoritmo

Asimismo permite su depuración, a partir de ejecuciones paso a paso, como en cualquier compilador

MatLab/Octave: Sentencias Condicionales y Bucles

```
if (P1==1)
%.....
elseif (P2==2)
%.....
else
%.....
end
```

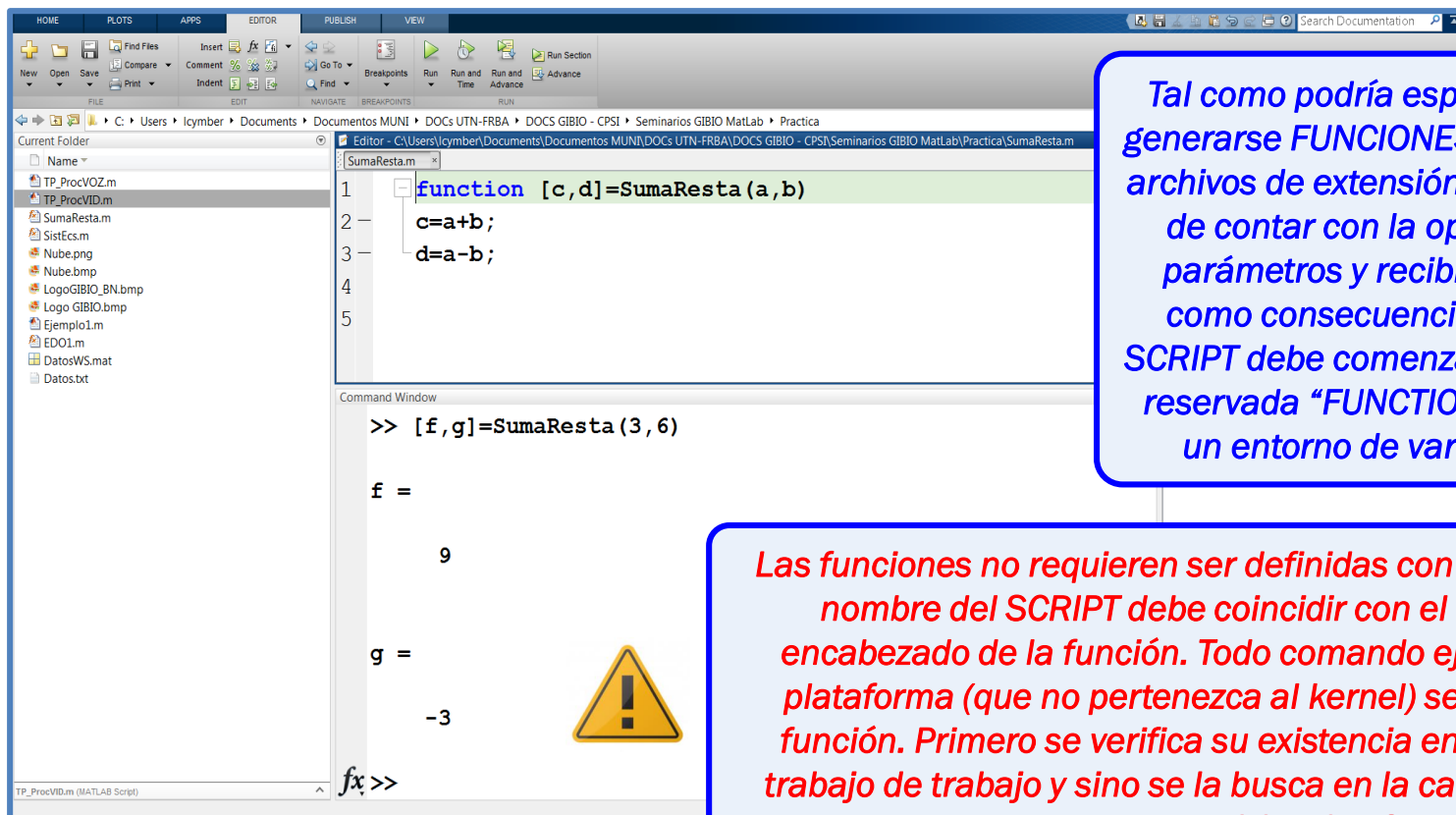
```
switch (P)
    case 1
        disp('hola');
    case { 6, 7 }
        disp('mundo');
    otherwise
        disp ('chau');
end
```

Por otra parte, se admite la utilización de sentencias CONDICIONALES y CICLOS tales como “IF”, “FOR”, “SWITCH” y “WHILE” (entre otras)

```
for k=m
disp(k)
end
```

```
while (P>0)
disp(P);
P=P-1;
end
```

MatLab/Octave: Generación de Funciones



*Tal como podría esperarse, pueden generarse **FUNCIONES** (también como archivos de extensión “*.m”) de modo de contar con la opción de pasar parámetros y recibir un resultado como consecuencia. Para ello, el **SCRIPT** debe comenzar con la palabra reservada “**FUNCTION**”. Trabajan en un entorno de variables **LOCAL***

*Las funciones no requieren ser definidas con anterioridad. El nombre del **SCRIPT** debe coincidir con el nombre del encabezado de la función. Todo comando ejecutado en la plataforma (que no pertenezca al kernel) se asume como función. Primero se verifica su existencia en la carpeta de trabajo de trabajo y sino se la busca en la carpeta donde se alojan las **TOOLBOXES***

MatLab/Octave: Publicación de scripts

El comando **PUBLISH** permite la ejecución de un algoritmo y que toda entrada o salida del mismo (incluidos los gráficos generados) **quede registrada en un archivo**, ya sea **doc**, **html** o **pdf**.

En nuestro caso se utilizará la salida en PDF

Para publicar una función o script debe ejecutarse el siguiente comando:

```
publish('Ejemplo.m', 'format', 'pdf')
```

NOTA: En OCTAVE es necesario tener preinstalado
LATEX para generar el PDF

scriptejemplo

Contents

Ejemplo 1

1

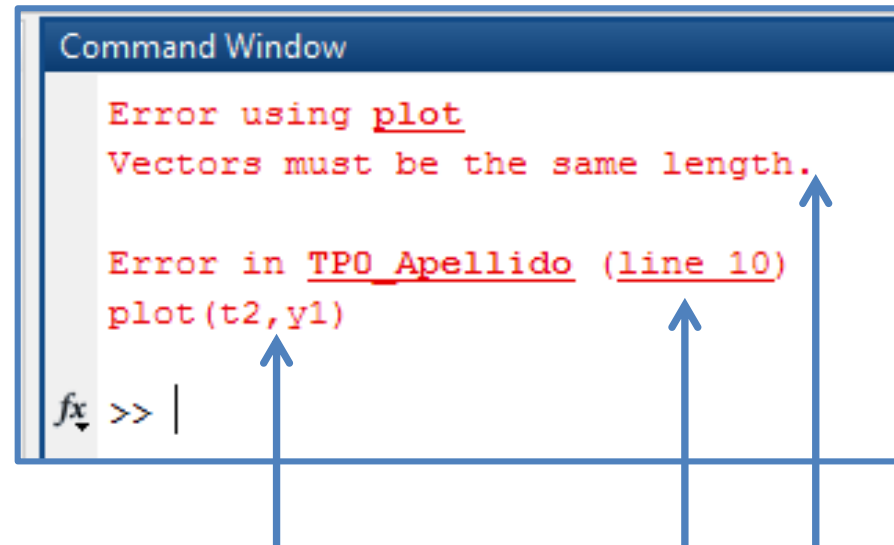
Ejemplo 1

```
dt=0.1;  
t=0:dt:5;  
st=3*sin(5*t);  
plot(t,st);  
Prueba(5);
```

5
4
3
2
1

MatLab/Octave: Manejo de Errores

```
%% TP0
% Apellidos: xxx
clc; % help clc
close all ; % help close
clear all ; % help clear
t1=0: 0.01:1;%Comentarios xxx
y1= 2*t1 ; % Usar ; !!!!
t2=0: 0.001 :1 ;
figure;
plot(t2,y1) MAL ERROR
¿Dónde buscamos el error?
```



Vectors must be the same length.

Name	Value
t1	1x101 double
t2	1x1001 double
y1	1x101 double

Consigna de la clase #A (15 minutos)

Implementar un *script en Matlab* que ejecute las siguientes acciones:

1. Generar un vector A de números aleatorios de tamaño 1×1000
2. Calcular el valor medio, el máximo, el mínimo y verificar su número de elementos
3. Generar un nuevo vector B con los elementos mayores a 3 de A

Se deberán colocar *comentarios explicativos* de cada sentencia (%)

Ayuda: Utilizar el comando *help* de Matlab/Octave para las funciones *randn*, *length*, *mean*, *max* y *min*

Consigna de la clase #B (15 minutos)

Implementar un *script en Matlab* que ejecute las siguientes acciones:

1. Generar un vector denominado "*t*", definido entre 0 y 4π a pasos temporales de valor $dt=0.001$
2. Calcular $\sin(2t)$ y almacenarlo en el vector *Sa*
3. Calcular $\cos(4t)$ y almacenarlo en el vector *Sb*
4. Generar una función denominada **CALC** que reciba los vectores *Sa* y *Sb* y proporcione como resultado la suma y el producto elemento a elemento
5. Graficar en una misma ventana: un eje con *Sa* en azul, otro con *Sb* en rojo, un tercero con el resultado de la suma (en verde, punteado) y un cuarto con el resultado del producto (en magenta, a rayas). Los ejes deberán estar ajustados, con sus etiquetas correspondientes y una leyenda identificando cada relación funcional.

ANEXO

Instalación de Octave en Windows

- Ingresar al sitio:
<https://www.gnu.org/software/octave/download.html/>
- **Descargar** el archivo “octave-xxx-wxx.zip” (32 o 64 bits)
- **Descomprimir** y **Ejecutar** “post-install.bat” (instala paquetes)
- **Abrir** Octave (la primera vez, ejecutar la aplicación “octavefirst”)
- **Chequear** los Toolboxes instalados ejecutando: **pkg list**
- **Recordar** que para poder utilizar los Toolboxes en Octave, los mismos deben ser cargados previamente:

pkg load signal (carga el paquete signal)

NOTA: Si se descarga el instalador en lugar del .zip, no es necesario ejecutar el archivo “post-install.bat”

Instalación de Octave en Linux (varios métodos)

1. Instalación tipo “SANDBOX” (FLATPAK, snap, docker, etc...)

Esta opción suele ser independiente de la distribución de Linux utilizada (**RECOMENDADA**). Desde la consola (tener en cuenta que lleva un tiempo...):

- *sudo apt-get install flatpak*
- *flatpak install flathub org.octave.Octave*

Más información en:

- <https://flatpak.org/setup/>
- <https://flathub.org/apps/details/org.octave.Octave>

Instalación de Octave en Linux (varios métodos)

2. Instalación desde el repositorio de la distribución de linux (suele ser una versión desactualizada). Desde la consola:

a) Para **determinar la versión**:

- `sudo apt-get update`
- `sudo apt-get upgrade`
- `sudo apt-cache policy octave`

b) Para **instalar**:

- `sudo apt-get install octave liboctave-dev octave-dbg octave-doc`

Instalación de Octave en Linux (varios métodos)

3. Descarga y compilación de los fuentes manualmente (como ultimo recurso)

- *Descargar los fuentes del último release y descomprimir:*
<https://espejito.fder.edu.uy/gnu/octave/>
- *Instalar todas las dependencias de octave (en lo posible utilizando **apt-get**)*
<https://wiki.octave.org/Building#Dependencies>
- *Efectuar la compilación*
https://wiki.octave.org/Building#General_steps

Instalación de Librerías de Octave en Linux

Serán necesarias a medida que se desarrollen los contenidos de la materia. Desde la consola de octave mismo (toma un tiempo!):

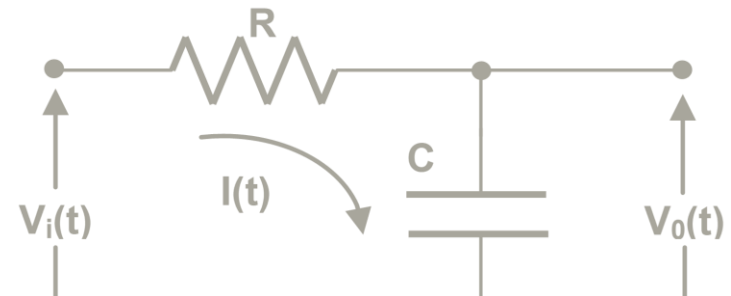
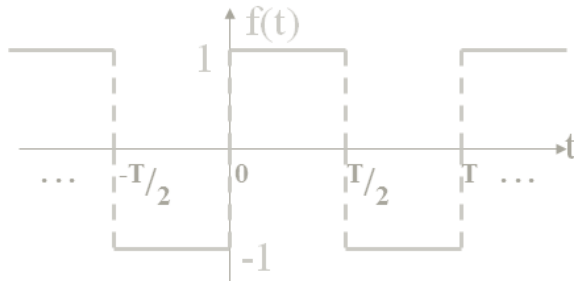
- `pkg install <LIBRERÍA> -forge -verbose`

Algunas librerías de utilidad:

- `control`
- `signal`
- `symbolic`

Para verificar las librerías instaladas: `pkg list`

Para cargar las librerías y utilizarlas: `pkg load <LIBRERÍA>`



Actividad Práctica ¿CONSULTAS? Foro Campus Virtual: MatLab/Octave

