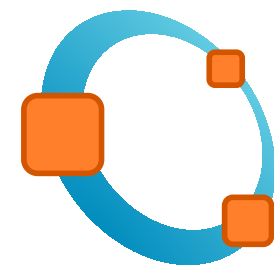




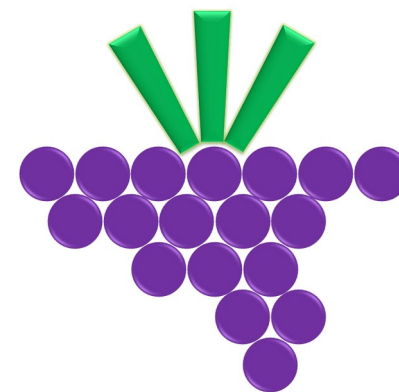
# Introducción a Octave



para ciencias aplicadas e ingeniería



## Unidad 2-B



Daniel Millán & Nicolas Muzi  
San Rafael, Argentina, Mayo–Junio de 2022



**UNCUYO**  
UNIVERSIDAD  
NACIONAL DE CUYO



FACULTAD DE  
**CIENCIAS APLICADAS  
A LA INDUSTRIA**



CONICET

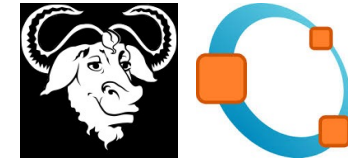


**MoCCAi**

MODELADO COMPUTACIONAL EN CIENCIAS APLICADAS E INGENIERÍA



# Unidad 2-B



## A. Operaciones con vectores/ matrices

1. Vectores/matrices como arreglos de números.
2. Operaciones con vectores/matrices.
3. Tipos de matrices predefinidos.
4. Operador (:). Matriz vacía [ ]. Borrado filas/columnas.
5. Operadores relacionales. Operadores lógicos.

## B. Trazado de gráficos

6. Función *plot()*.
  - a) Estilos de línea y marcadores.
  - b) Añadir curvas a un gráfico ya existente.
  - c) Control de los ejes: *axis()*.
7. Control de ventanas gráficas: *figure()*.
8. Otras funciones gráficas 2D.
9. Resumen funciones para gráficas 3D.

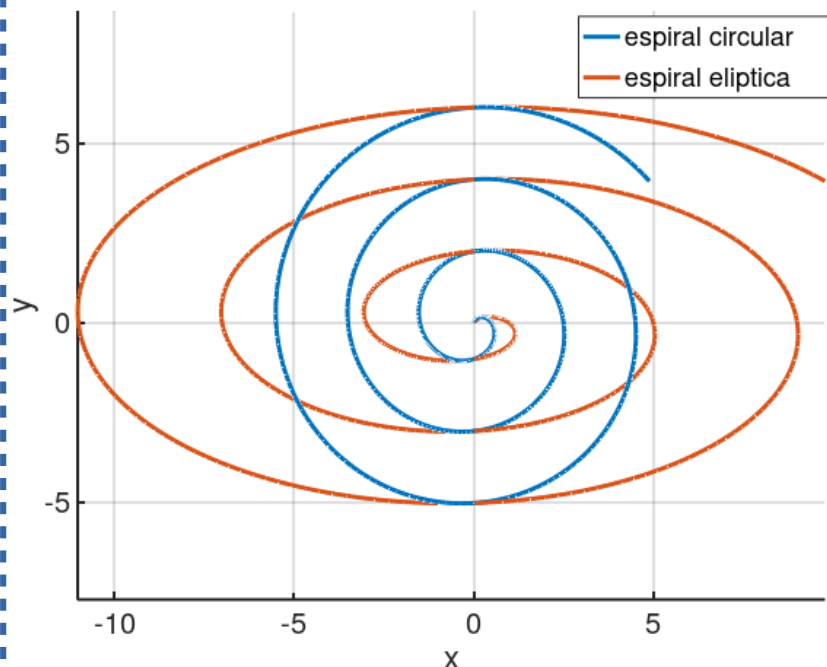


# Trazado de gráficos

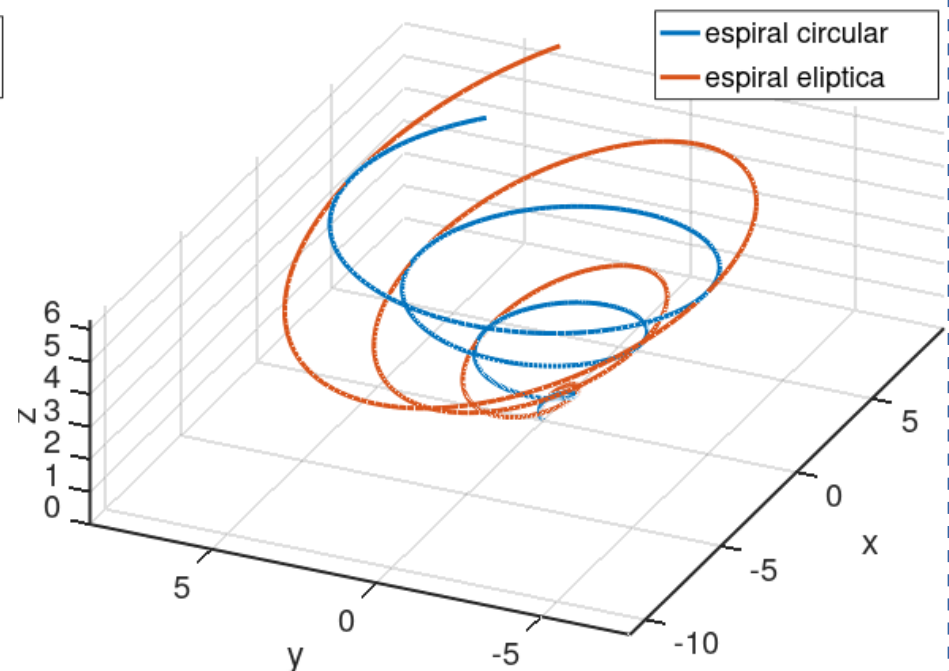
**Ejemplo:** empleo del paquete "**symbolic**" para crear variables simbólicas. Graficamos con **ezplot3** una espiral circular y otra elíptica parametrizada por  $t$  en 3D.

➔ **Ayuda:** utilizar el script [U2\\_ej\\_ezplot3\\_espiral3D.m](#) subido a la web del curso.

Curvas parametrizadas en 3D



Curvas parametrizadas en 3D



**Ejercicio:** empleando el script anterior grafique una hélice circular y otra elíptica.



## 6. Función *plot()*

- Función clave de todos los gráficos 2-D en Octave/Matlab.
- El elemento básico de los gráficos bidimensionales es el **vector**.
- Se utilizan también cadenas de 1, 2 ó 3 caracteres para indicar *colores* y *tipos de línea*.

**Ejemplo:** Realice una figura empleando la función **plot** de  $\sin(x)$  y  $\cos(x)$  para  $x$  en  $[0, 2\pi]$ . Además coloque título, nombre a los ejes, nombre a las curvas y señale el punto  $(1, \sin(1))$ . Defina el tamaño de letra en 20.

```
x=linspace(0,2*pi,20);
plot(x, sin(x))
title('Grafica del sen(x)', 'fontsize', 20)
xlabel('x', 'fontsize', 20)
ylabel('sen(x)', 'fontsize', 20)
text(1, sin(1), '(1, sen(1))', 'fontsize', 20,
      'fontweight', 'bold', 'color', 'blue')
```

*Los 3 puntos indican que la secuencia de órdenes continúa en la línea sgte.*



# 6. Función *plot()*

## a) Estilos de línea y marcadores en la función *plot*

Símbolo	Color	Símbolo	Marcadores (markers)
y		.	puntos
m		o	círculos
c		x	marcas en x
r		+	marcas en +
g		*	marcas en *
b		s	marcas cuadradas (square)
w		d	marcas en diamante (diamond)
k		^	triángulo apuntando arriba
		v	triángulo apuntando abajo
		>	triángulo apuntando a la dcha
		<	triángulo apuntando a la izda
		p	estrella de 5 puntas
		h	estrella se seis puntas
<b>Símbolo</b>	<b>Estilo de línea</b>		
-	líneas continuas		
:	líneas a puntos		
-.	líneas a barra-punto		
--	líneas a trazos		



## 6. Función *plot()*

- Es posible añadir en la función *plot* algunos especificadores de línea que controlan el espesor de la línea, el tamaño de los marcadores, etc.

**Ejemplo:** emplee las siguientes especificaciones en el ejemplo anterior.

```
>> plot(x,sin(x),'- .rs', ... %r:red  
      'LineWidth',4, ...  
      'MarkerEdgeColor','k', ... %k:black  
      'MarkerFaceColor','g', ... %g:green  
      'MarkerSize',40)
```





## 6. Función *plot()*

### b) Añadir curvas a un gráfico ya existente.

- Es posible añadir líneas/curvas a un gráfico ya existente, sin destruirlo o sin abrir una nueva ventana empleando los comandos *hold on* y *hold off*.
- El primero de ellos hace que los gráficos sucesivos respeten los que ya se han dibujado en la figura (es posible que haya que modificar la escala de los ejes); el comando *hold off* deshace el efecto de *hold on*.

**Ejercicio:** analice el funcionamiento de las siguientes líneas de órdenes (copie y pegue en un script)

```
x=-2:0.1:2;
```

```
hold on
```

```
plot(x,x,'r-','LineWidth',2) %r:red
```

```
plot(x,x.^2,'b--','LineWidth',2) %b:blue
```

```
plot(x,x.^3,'m-.','LineWidth',2) %m:magenta
```

```
hold off
```





## 6. Función *plot()*

### c) Control de los ejes: función *axis()*

- Por defecto, se ajusta la escala de cada uno de los ejes de modo que varíe entre el valor mín/máx de los vectores a representar.
- Este es el llamado modo "auto", o modo automático. Es posible definir de modo explícito los valores máx/mín según cada eje:

**`axis([xmin, xmax, ymin, ymax])`.**

- **`v=axis`** devuelve un vector **v** con los valores [xmin, xmax, ymin, ymax]
- **`axis('ij')`** utiliza *ejes de pantalla*, eje **j** en dirección vertical descendente
- **`axis('xy')`** utiliza *ejes cartesianos*, eje **y** vertical ascendente
- **`axis('auto x')`** utiliza el escalado automático sólo en dirección **x**
- **`axis(axis)`** fija los ejes a su valores actuales, de cara a posibles nuevas gráficas añadidas con ***hold on***
- **`axis('tight')`** establece los límites de los datos
- **`axis('equal')`** el escalado es igual en ambos ejes
- **`axis('square')`** la ventana será cuadrada
- **`axis('normal')`** elimina las restricciones hechas por 'equal' y 'square'







## 6. Función *plot()*

### c) Control de los ejes: función *axis()*

- **axis('off')** elimina las etiquetas, los números y los ejes
- **axis('on')** restituye las etiquetas, los números y los ejes
- **XLim, YLim** permiten modificar selectivamente los valores máximo y mínimo de los ejes en las direcciones x e y.

**Ejercicio:** en una terminal comprobar el comportamiento de las opciones para *axis*:

```
>> x=-pi:pi/20:pi;  
>> plot(x,sin(x),'r-')  
>> axis('auto')  
>> axis('equal')  
>> axis([-4,4,-2,2])  
>> axis('square')  
>> axis('ij')  
>> axis('xy')  
>> axis('tight')
```





## 6. Función *plot()*

### c) Control de los ejes: función *gca*

**Ejemplo:** Es posible también tener un control preciso sobre las marcas y los rótulos que aparecen en los ejes:

```
x = -pi:0.1:pi; y = sin(x);
plot(x,y)
set(gca, 'Xtick', -pi:pi/2:pi)
set(gca, 'XTickLabel',
    {'-pi', '-pi/2', '0', '\pi/2', '\pi'})
```

¿Qué efecto tiene “\”?

- Observe cómo las propiedades se establecen sobre los ejes actuales, a los que se accede con la función *gca* (*get current axis*).





# 7. Control de ventanas gráficas: *figure()*

- La función ***figure*** sin argumentos, crea una nueva ventana gráfica con el número consecutivo que le corresponda.
- El comando ***figure(n)*** hace que la ventana **n** pase a ser la ventana o figura activa. Si dicha ventana no existe, se crea una nueva ventana con el número consecutivo que le corresponda.
- La función ***close*** cierra la figura activa, mientras que ***close(n)*** cierra la ventana o figura número **n** y ***close all*** cierra todas.
- El comando ***clf*** elimina el contenido de la figura activa, es decir, la deja abierta pero vacía.
- La función ***gcf*** devuelve el número de la figura activa en ese momento.





# 7. Control de ventanas gráficas: *figure()*

- **gcf**: *get current figure* provee un mecanismo para actuar e inspeccionar las propiedades sobre la figura activa actual.

## Ejemplo:

```
>> fplot (@sin, [-10, 10]);  
>> fig = gcf();  
>> set (fig, "numbertitle", "off", ...  
        "name", "sin plot")
```





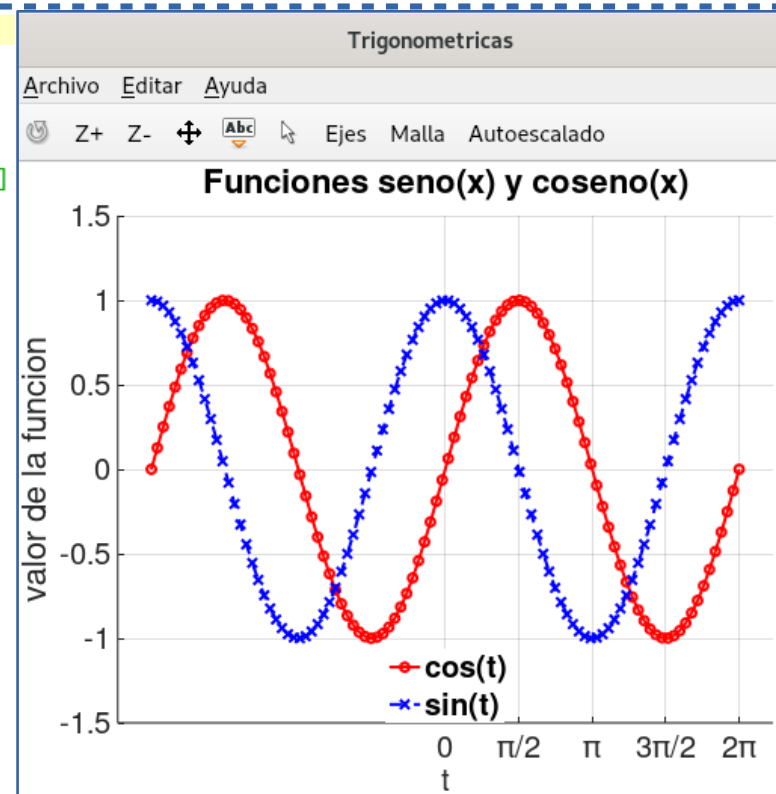
# 7. Control de ventanas gráficas: *figure()*

## Ejemplo

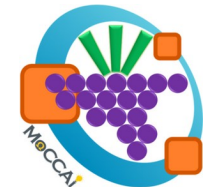
```

3 % Ejemplo: prueba de algunas posibilidades de plot
4
5 clear
6 close all
7
8 %vector x conteniendo 100 valores equiespaciados en [-2*pi,2*pi]
9 x=linspace(-2*pi,2*pi,100);
10
11 %creamos la figura
12 figure(1);clf
13 pause(4) %espera 4 segundos
14
15 %graficamos las funciones seno y coseno para cada valor de x
16 hold on
17 plot(x,sin(x),'ro-', 'linewidth',2)
18 plot(x,cos(x),'bx--', 'linewidth',2)
19 hold off
20 pause(4) %espera 4 segundos
21
22 %agregamos el titulo, nombre a los ejes, curvas, etc
23 title('Funciones seno(x) y coseno(x)', 'fontsize',20)
24 xlabel('angulo en radianes', 'fontsize',20)
25 ylabel('valor de la funcion', 'fontsize',20)
26 leg=legend('cos(t)', 'sin(t)', 'location', 'north');
27 legend boxoff
28 set(gca, 'fontsize',20)
29 axis([-7,7, -1.5,1.5])
30 axis('on'), grid
31 pause(4) %espera 4 segundos
32
33 %cambiamos algunas de las propiedades definidas anteriormente
34 xlabel('t', 'fontsize',20)
35 leg=legend('cos(t)', 'sin(t)', 'location', 'south');
36 legend boxoff
37 set(leg, 'fontsize',20, 'fontweight', 'b')
38 set(gca, 'Xtick', 0:pi/2:2*pi)
39 set(gca, 'XtickLabel', {'0', '\pi/2', '\pi', '3\pi/2', '2\pi'})
40 set(gcf, 'number', 'off', 'name', 'Trigonometricas')

```



**Ayuda:** utilizar el script [U2\\_ej\\_trigo\\_funcs.m](#) subido a la web del curso. 





## 8. Otras funciones gráficas 2D

- Funciones gráficas 2D orientadas a generar otro tipo de gráficos distintos de los que produce la función **plot()** y sus análogas.
  - **bar()** crea diagramas de barras
  - **barh()** diagramas de barras horizontales
  - **pie()** gráficos con forma de “tarta”
  - **pie3()** gráficos con forma de “tarta” y aspecto 3-D
  - **area()** similar **plot()**, pero rellenando en ordenadas de 0 a y
  - **stairs()** función análoga a **bar()** sin líneas internas
  - **errorbar()** representa sobre una gráfica –mediante barras– valores de errores
  - **hist()** dibuja histogramas de un vector
  - **rose()** histograma de ángulos (en radianes)
  - **polar()** gráfica en coordenadas polares
  - **quiver()** dibujo de campos vectoriales como conjunto de vectores





## 9. Algunas funciones gráficas 3D

- En general las opciones vistas anteriormente se pueden aplicar a las funciones que permiten graficar puntos, líneas y superficies en 3D.
- Un gran “resumen” de las funciones disponibles sería:
  - Dibujo simplificado de funciones: ***ezplot3***, ***ezsurf***, ***ezsurfc***, etc.
  - Dibujo de puntos y líneas: ***plot3***
  - Dibujo de mallas: ***meshgrid***, ***mesh*** y ***surf***
  - Dibujo de líneas de contorno: ***contour*** y ***contour3***



**FIN.**

San Rafael, Argentina 2022