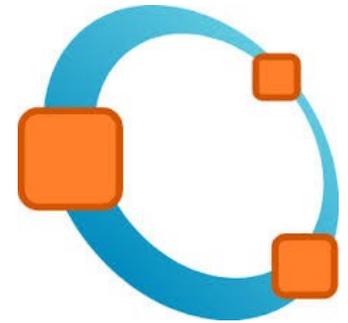


Introducción a Octave

Unidad 4



Daniel Millán

San Rafael, Argentina Marzo-Abril 2019



Departamento de
Ingeniería Mecánica



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



FACULTAD DE
**CIENCIAS APLICADAS
A LA INDUSTRIA**



Trazado de gráficos

1. Mejorando la presentación.
2. Funciones gráficas 2D elementales.
3. Función ***plot()***.
 - a) Estilos de línea y marcadores.
 - b) Añadir curvas a un gráfico ya existente.
 - c) Control de los ejes: ***axis()***.
4. Control de ventanas gráficas: ***figure()***.
5. Otras funciones gráficas 2D.
6. Resumen funciones para gráficas 3D.



1. Mejorando la presentación.

- La capacidad de proporcionar visualizaciones de calidad a partir de datos de salida es clave en el análisis de datos.
- Sin visualización, las simulaciones numéricas son difíciles y a veces imposibles de interpretar.
- Cuando las computadoras se introdujeron en el dominio técnico-científico, la generación de publicaciones con imágenes de calidad para realizar análisis detallados de los resultados numéricos de problemas complejos fue uno de los mayores desafíos para los ingenieros y científicos de todo el mundo.
- Octave proporciona esta funcionalidad. Sus características de trazado permite elegir entre varios tipos de gráficos en 2D y 3D, decorar figuras con: títulos, nombres de ejes, cuadrículas, etiquetas para datos, ecuaciones, etc.
- La visualización de los resultados de experimentos basados en la simulación permiten una comprensión intuitiva.



1. Mejorando la presentación.

- Los gráficos 2-D de Octave/Matlab están orientados a la representación gráfica de vectores (y matrices).
- Cuando una matriz aparezca como argumento, se considerará como un conjunto de vectores columna (fila).
- Octave/Matlab utiliza un tipo especial de ventanas para realizar las operaciones gráficas.
- Como se verá ciertos comandos abren una ventana nueva y otros dibujan sobre la ventana activa, bien sustituyendo lo que hubiera en ella, bien añadiendo nuevos elementos gráficos a un dibujo anterior.



2. Funciones gráficas 2D elementales.

- Dibujo simplificado de funciones: **ezplot**, **ezcontour**.
- Las **ezALGO** poseen varias formas de uso, siendo la más útil por su simplicidad cuando se emplean cadenas de caracteres: **ezplot("x+2")** ó **ezplot('x+2')**
- Por defecto ezALGO grafica entre $-2\pi < x | y < 2\pi$.

Ejercicio: comprobar el funcionamiento de ezcontour

a) `ezcontour("cos(x/2)*sin(y/2)")`

b) `ezcontour("cos(x/2)+sin(y/2)")`



2. Funciones gráficas 2D elementales.

- Funciones orientadas a *embellecer* la figura:
 - **title**('título') añade un título al dibujo
 - **xlabel**('tal') añade una etiqueta al eje de abscisas. Con **xlabel off** desaparece
 - **ylabel**('cual') añade una etiqueta al eje de ordenadas. Con **ylabel off** desaparece
 - **text**(x,y,'texto') introduce 'texto' en el lugar especificado por las coordenadas **x** e **y**. Si **x** e **y** son vectores, el texto se repite por cada par de elementos.
 - **legend**() define rótulos para las distintas líneas o ejes utilizados en la figura. Para más detalle, consultar el **Help/doc**
 - **grid** activa la inclusión de una cuadrícula en el dibujo. Con **grid off** desaparece la cuadrícula.
 - Las propiedades se establecen sobre los ejes actuales, a los que se accede con la función **gca** (*get current axis*).



2. Funciones gráficas 2D elementales.

- Dado gráfico es posible definir el valor de retorno opcional de su “controlador de gráficos”.

H = ezALGO (...)

Ejercicio: comprobar las funciones anteriores en la curva plana que describe la “Espiral de Arquímedes”.

```
h=ezplot('t*cos(t)', 't*sin(t)', [0,4*pi])
title('Espiral Arquimides')
xlabel('x(t)')
ylabel('y(t)')
legend('Espiral')
text(1,1, 'Hola Mundo')
grid on
set(gca, 'fontsize', 20)
set(h, 'linewidth', 3)
```



2. Funciones gráficas 2D elementales.

- Se dispone de cinco funciones básicas para crear gráficos 2-D a partir de datos generados o cargados por el usuario desde un archivo.
- Estas funciones se diferencian principalmente por el *tipo de escala* que utilizan en los ejes de abscisas y de ordenadas.
 - **plot()** crea un gráfico a partir de vectores y/o columnas de matrices, con escalas lineales sobre ambos ejes
 - **plotyy()** dibuja dos funciones con dos escalas diferentes para las ordenadas, una a la derecha y otra a la izquierda de la figura.
 - **loglog()** ídem con escala logarítmica en ambos ejes
 - **semilogx()** ídem con escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas
 - **semilogy()** ídem con escala lineal en el eje de abscisas y logarítmica en el eje de ordenadas



2. Funciones gráficas 2D elementales.

IMPORTANTE:

- En lo sucesivo se hará referencia casi exclusiva a la primera de estas funciones (*plot*).
- Las demás funciones para graficar se pueden utilizar de un modo similar.
- La función *plot()*, en sus diversas variantes, no hace otra cosa que dibujar vectores.

Ejercicio

```
>> x=[1 3 2 4 5 3 7]
```

```
>> plot(x)
```

```
>> x=[1 6 5 2 1]; y=[1 0 4 3 1];
```

```
>> plot(x,y)
```



3. 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*.

Ejercicio: 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.



3. Función *plot()*

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

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



3. 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.

Ejercicio: emplee las siguientes especificaciones en el ejemplo anterior.

Los tres puntos permiten continuar la secuencia de órdenes en la línea sgte.

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



3. 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: para $x = -2 : 0.1 : 2$

```
>> 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
```



3. 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'¹⁴



3. 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: para $x = -\pi : \pi/20 : \pi$, `plot(x,sin(x),'r-')` chequear:

```
>> axis('auto')
>> axis('equal')
>> axis([-4,4,-2,2])
>> axis('square')
>> axis('ij')
>> axis('xy')
>> axis('tight')
```



3. Función *plot()*

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

Ejercicio: 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'})
```

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



3. Función *plot()*

Ejercicio

```
>> x=[-4*pi:pi/20:4*pi];  
>> plot(x,sin(x),'r',x,cos(x),'g')  
>> title('Funciones seno(x)-rojo- ...  
        y coseno(x)-verde-')  
>> xlabel('angulo en radianes')  
>> ylabel('valor de la funcion trigonometrica')  
>> axis([-12,12,-1.5,1.5])  
>> axis('off')  
>> axis('on'), grid
```



4. 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.



4. Control de ventanas gráficas: *figure()*

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

Ejercicio

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

Ejercicio: Estime y grafique la distancia $s(t)$ que ha recorrido una partícula sobre una curva parametrizada en función de t . La curva está definida por $\mathbf{r}(t) = (\cos(t), 2 \sin(t))$, para t en $[0, 2\pi]$.



5. 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



6. 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***, etc.
 - Dibujo de puntos y líneas: ***plot3***
 - Dibujo de mallas: ***meshgrid***, ***mesh*** y ***surf***
 - Dibujo de líneas de contorno: ***contour*** y ***contour3***