Ingeniate en Octave

Clase 3

Nicolás Muzi, Daniel Millán, Juan Caro Boldrini, Germán Greulach, Emanuel Romero, Román Molina, Fernando Gutiérrez, Agustín Morcillo

San Rafael, Argentina Abril-Mayo 2023

CONICET

MoCCAi

MODELADO COMPUTACIONAL EN CIENCIAS APLICADAS E INGENIERÍA

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD DE
CIENCIAS APLICADAS
A LA INDUSTRIA

Contenido Clase 3

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.

Trazado de gráficos

6. Función ***plot*()**.
7. Control de ventanas gráficas: ***figure*()**.
8. Otras funciones gráficas 2D.
9. Resumen funciones para gráficas 3D.

Operaciones con vectores y 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.

1. Vectores/matrices

- Los vectores se pueden definir directamente introduciendo los elementos que lo componen, p.ej.
  ```
  >> u=[1,2,3]  % vector fila
  ```
- Observamos que **u** es un vector fila.
- En el caso de desear un vector columna utilizamos **;**
  ```
  >> v=[1;2;3]  % vector columna
  ```

Ejercicio: ¿Es posible calcular la suma de **u** y **v**?

- Para transformar un vector fila en columna se debe transponer dicho vector mediante la orden **'** (comilla).
- Octave genera por defecto vectores fila:
  ```
  >> w(1)=1; w(2)=4; w(3)=9;
  >> w
  ```

1. Vectores/matrices

- Las matrices se crean introduciendo los elementos

```
>> A=[1 4 -3; 2 1 5; -2 5 3]
```

- Adicionalmente es posible crear matrices mediante vectores filas/columnas

```
>> f1=[1 4 -3]; f2=[2 1 5]; f3=[2 5 -3];
>> Af=[f1;f2;f3]

>> c1=[1 2 2]; c2=[4 1 5]; c3=[-3 5 -3];
>> Ac=[c1,c2,c3]
```

2. Operaciones con vectores/matrices

- Operadores aritméticos:

  + adición o suma

  – sustracción o resta

  * multiplicación

  ' traspuesta

  ^ potenciación

  \ división-izquierda

  / división-derecha

  .* producto elemento a elemento

  ./ y .\ división elemento a elemento

  .^ elevar a una potencia elemento a elemento

2. Operaciones con vectores/matrices

- Operadores matriciales elemento a elemento (*, ^, \ y /). Para ello basta precederlos por un punto (.)

```
>> [1 2 3 4].^2
>> [1 2 3 4].*[1 -1 1 -1]

>> [1 -1;1 -1].^3
>> [1 2;3 4]*[1 -1;1 -1]
>> [1 2;3 4].*[1 -1;1 -1]


>> [1 2;3 4]./[1 -1;1 -1]
>> [1 2;3 4].\[1 -1;1 -1]
```

3. Tipos de matrices predefinidos

- **eye(4)** forma la matriz unidad de tamaño (4×4)
- **zeros(3,5)** forma una matriz de *ceros* de tamaño (3×5)
- **zeros(4)** ídem de tamaño (4×4)
- **ones(2,3)** forma una matriz de *unos* de tamaño (2×3)
- **linspace(0,1,7)** genera un vector con **7** valores igualmente espaciados entre **0** y **1**
- **rand(3)** crea una matriz de números aleatorios entre 0 y 1, con distribución uniforme, de tamaño (3×3)
- **randn(4)** matriz de números aleatorios de 4×4, con distribución normal, de valor medio 0 y varianza 1.
- **magic(4)** crea una matriz (4×4) con los números 1, 2, ... 4*4, tal que todas las filas y columnas suman lo mismo.
- **hilb(5)** matriz de Hilbert de 5×5. El elemento (i,j) está dado por (1/(i+j-1)). Esta matriz produce grandes errores numéricos al resolver sistemas lineales **A x = b**.

3. Tipos de matrices predefinidos

Formación de un matriz a partir de Otras ya definidas

- recibiendo alguna de sus propiedades (por ejemplo el tamaño),
- por composición de varias submatrices más pequeñas,
- modificándola de alguna forma.

Ejercicio: Dada la matriz **A=rand(3)** y el vector **x=[1;2;3]**, comprobar:

- ✓ **`[m,n]=size(A)`** devuelve el número de filas y de columnas de la matriz **A**. **`n=length(x)`** calcula el número de elementos de un vector **x.**
- ✓ **zeros(size(A))** forma una matriz de *ceros* del mismo tamaño que una matriz **A** **ones(size(A))** ídem con *unos*
- ✓ **A=diag(x)** forma una matriz diagonal **A** cuyos elementos diagonales son los elementos del vector **x.**
- ✓ **x=diag(A)** forma un vector **x** a partir de los elementos de la diagonal de **A.**
- ✓ **diag(diag(A))** crea una matriz diagonal a partir de la diagonal de la matriz **A.**

```
>> B=diag(diag(A))
>> C=[A, eye(3); zeros(3), B]
```

4. Operador (:). Matriz vacía [ ]. Borrado filas/columnas

- El operador ":" es muy importante y puede usarse de varias formas.
  ¡Probar y jugar, es la mejor forma de aprender!.

Arreglo de números o vector

```
>> x=1:10
>> x=1:2:10
>> x=1:1.5:10
>> x=10:-1:1

>> x=[0.0:pi/50:2*pi]';
>> y=sin(x); z=cos(x);
>> [x y z]
```

Matrices

```
>> A=magic(5)
>> A(2,3)
>> A(5,1:4)
>> A(3,:)
>> A(end,:)
>> A(3:5,:)
>> A([1 2 5],:)
>> A(:) % vector columna
>> B=eye(size(A));
>> B([2 4 5],:)=A(1:3,:)
```

4. Operador (:). Matriz vacía [ ]. Borrado filas/columnas

- Una matriz definida sin ningún elemento entre los corchetes es una matriz que *existe*, pero que está *vacía*, o lo que es lo mismo que tiene *dimensión cero*.

```
>> A=magic(3)
>> B=[]
>> exist("B"), exist("C")
>> isempty(A),isempty(B)
>> A(:,3)=[]
```

- Las funciones ***exist()*** e ***isempty()*** permiten chequear si una variable existe y si está vacía.
- En el último ejemplo se ha eliminado la 3ª columna de **A** asignándole la matriz vacía.

5. Operadores relacionales. Operadores lógicos.

Operadores relacionales, se aplican a vectores y matrices:

< menor que

> mayor que

<= menor o igual que

>= mayor o igual que

== igual que

~= distinto que

- La comparación se realiza elemento a elemento, y el resultado es otra matriz de unos y ceros del mismo tamaño

```
>> A=[1 2;0 3]; B=[4 2;1 5];
>> A==B
>> A~=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.

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*.

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

```
>>x=linspace(0,2*pi,100);
>>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)
```

6. 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		

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.

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)
```

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: 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
```

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: 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')
```

6. Función *plot*()

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

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'})
```

¿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* 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")
```

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

Ejercicio: cree un *script* y analice las siguientes órdenes

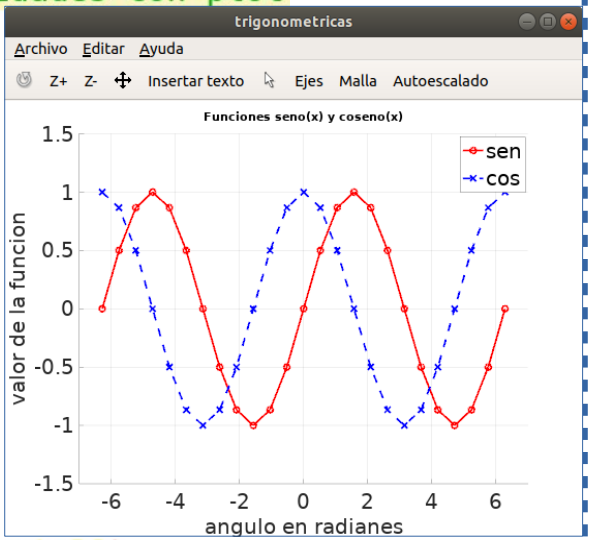
```
%Este es un ejercicio para probar las posibilidades con plot
clear
close all

x=[-2*pi:pi/6:2*pi];
figure(10);clf
pause(3)

hold on
plot(x,sin(x),'ro-','linewidth',2)
plot(x,cos(x),'bx--','linewidth',2)
hold off
pause(3)

title('Funciones seno(x) y coseno(x)','fontsize',20)
xlabel('angulo en radianes','fontsize',20)
ylabel('valor de la funcion','fontsize',20)
leg=legend('sen','cos')
axis([-7,7,-1.5,1.5])
axis('on'), grid
pause(3)

set(leg,'fontsize',16)
set(gca,'fontsize',16)
set(gcf,"numbertitle","off","name","trigonometricas")
```



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