APUNTES DE MATLAB

INTRODUCCIÓN A LA INFORMÁTICA

LICENCIATURA EN BIOLOGÍA

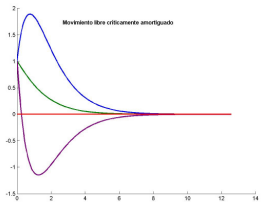
(última corrección : 24 de septiembre de 2009)

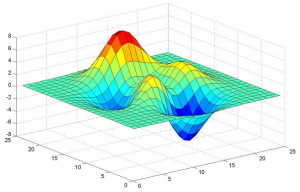
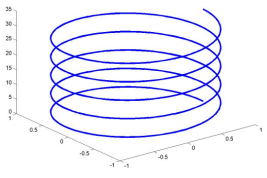
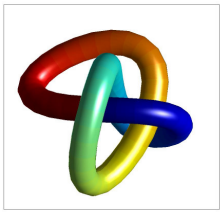
1. INTRODUCCIÓN

MATLAB es un programa para realizar cálculos de muy diversos tipos:

operaciones elementales, resolución de sistemas lineales, cálculo de integrales, cálculos con polinomios, resolución de ecuaciones diferenciales, ... y muchos otros.

Además, se pueden realizar con facilidad gráficos matemáticos de distintos tipos, en dimensión 2 y en dimensión 3.



1.1 ¿Cómo se escriben los números?

- Enteros (sin punto decimal):
  23 321 -34
- Reales (con punto decimal):
  23. -10.1 11.321
- Reales (notación científica o exponencial):
  2.e-2 = 2 x 10^{-2} = 0.02
  2.1e+5 = 2.1 x 10^{5} = 210000

Atención: para separar la parte entera de la parte decimal hay que usar PUNTO DECIMAL.

1.2 ¿Cómo escribir las operaciones aritméticas elementales ?

Suma:	**+**
Resta:	**-**
Multiplicación:	*****
División:	**/**
Exponenciación:	**^**

EJEMPLO

```
>> 2.01*4*3.1416
>> -2.98+0.23-14+2
>> 6+4/2+3.111
>> 5.22*3.1416/6-4
```

Se puede utilizar MATLAB como simple calculadora, escribiendo expresiones aritméticas y terminando por **RETURN** (**<R>**). Se obtiene el resultado inmediatamente a través de la variable del sistema **ans** (de answer). Si no se desea eco (es decir, la respuesta inmediata a cada orden) en el terminal, deben terminarse las órdenes por "**punto y coma**".

1.3 Orden en que se realizan las operaciones aritméticas

Cuando en una expresión hay varios operadores aritméticos, el orden en que se realizan las operaciones es determinante: las operaciones **NO SE EFECTÚAN SIEMPRE EN EL ORDEN EN QUE ESTÁN ESCRITAS**.

El orden viene determinado por las reglas siguientes:

1. Exponenciaciones

2. Multiplicaciones y divisiones
3. Sumas y restas
4. Dentro de cada grupo, de izquierda a derecha

PARA MODIFICAR ESTE ORDEN SE USAN PARÉNTESIS:

5. Si hay paréntesis, su contenido se calcula antes que el resto
6. Si hay paréntesis anidados, se efectúan primero los más internos

EJEMPLOS

```
>> 2+3*4 = 2+(3*4) = 2+12 = 14
>> (2+3)*4 = 5*4 = 20
>> 1/3*2 = (1/3)*2 = 0.3333*2 = 0.6666
>> 1/(3*2) = 1/6 = 0.1666
>> 2+3^4/2 = 2+(3^4)/2 = 2+81/2 = 2+(81/2) = 2+40.5 = 42.5
>> 2+3^(4/2) = 2+3^2 = 2+(3^2) = 2+9 = 11
>> (2+3^4)/2 = (2+(3^4))/2 = (2+81)/2 = 83/2 = 41.5
```

EJERCICIOS

Escribir en MATLAB

$$\frac{3+4^2}{\frac{2}{\sqrt[5]{3}}-\left(\frac{1}{3,1\cdot 2}\right)^{\frac{3}{4}}}$$

```
>> (3+4^2)/((2/3^(1/5))-(1/(3.1*2))^(3/4))
```

Escribir en MATLAB

$$\frac{1}{\frac{2}{0,1^{1/2}}-\frac{0,4}{2^{1/3}}}$$

```
>> 1/((2/0.1^(1/2))-(0.4/2^(1/3)))
```

Escribir en MATLAB

$$\frac{4,1^{\frac{0,2+1}{2}}}{\frac{2}{0,1^{1/2}}-\frac{0,4}{2^{1/3}}}$$

```
>> 4.1^((0.2+1)/2)/(2/0.1^(1/2)-0.4/2^(1/3))
```

1.4 Variables

Una VARIABLE es un nombre simbólico que identifica una parte de la memoria, y en la que podemos guardar números u otro tipo de datos.

ES UN "SITIO" EN LA MEMORIA DEL
ORDENADOR PARA "GUARDAR" DATOS

El contenido de una variable lo podemos recuperar y modificar cuantas veces queramos, a lo largo de una sesión de trabajo.

Se le pueden dar a las variables los nombres que queramos, formados por letras y números, hasta un máximo de 19, y comenzando por una letra. No se pueden utilizar los caracteres especiales:

+ - = * ^ < > ...

ATENCIÓN: MATLAB distingue entre letras mayúsculas y minúsculas

EJEMPLOS DE NOMBRES DE VARIABLES

a	peso	HarryPotter
XY	Peso	Darwin
ab12	PESO	PericoPalotes
kr10	pESO	JoseMari

Las variables en MATLAB no necesitan ningún tipo de declaración y pueden almacenar sucesivamente distintos tipos de datos: enteros, reales, escalares, matriciales, caracteres, etc.

Para CREAR una variable basta con asignarle un valor

Para ASIGNAR un valor a una variable se utiliza una instrucción de asignación:

>> nombre_de_variable = expresión

EJEMPLOS

```
>> ab=321
>> AB=3
>> x1=1/2
>> y1=1-4^2
```

1.4 Variables predefinidas

Algunos nombres están pre-definidos por MATLAB:

VARIABLES PREDEFINIDAS	
ans	variable del sistema para almacenar el resultado de evaluar expresiones
i , j	unidad imaginaria : raiz cuadrada de -1
pi	número π
Inf	"Infinito": número mayor que el más grande que se puede almacenar
NaN	"Not a Number : magnitud no numérica resultado de cálculos indefinidos

1.5 Funciones matemáticas elementales

FUNCIONES MATEMÁTICAS ELEMENTALES			
sqrt(x)	raiz cuadrada	sin(x)	seno (en radianes)
exp(x)	exponencial	cos(x)	coseno (en radianes)
log(x)	logaritmo neperiano	tan(z)	tangente (en radianes)
log10(x)	logaritmo decimal	asin(x)	arco-seno
		acos(x)	arco-coseno
		atan(x)	arco-tangente

El argumento de las funciones puede ser un número, una variable o una expresión conteniendo ambas cosas.

Cuando en una expresión aparece alguna función, su valor se calcula antes que cualquier otra cosa.

EJEMPLOS

```
>> sqrt(7)
>> sqrt(7/5)
>> a=2.1; sqrt(2*a)
>> exp(3)
>> exp(x)
>> 7*exp(5/4)+3.54
```

2. VECTORES Y MATRICES

2.1 Definición de vectores y matrices

Un vector-fila de dimension **n** se puede definir en MATLAB escribiendo sus componentes entre corchetes rectos (**[ ]**) y separándolos por comas o espacios en blanco:

>> v=[1,-1,0,2.88]

La orden anterior crea en MATLAB una variable de nombre **v** que "contiene" un vector-fila de longitud 4.

Un vector-columna se crea igual, pero separando las componentes por "punto y coma":

>> w=[0;1;2;3;4;5]

crea una variable de nombre **w**, que "almacena" un vector-columna de longitud 6.

Las matrices se definen de forma similar a los vectores, introduciendo sus filas como vectores-fila y separando unas filas de otras mediante punto y coma o saltos de línea.

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

Si **A** y **B** son dos matrices que tienen el mismo número de filas, entonces **[A,B]** es la matriz que se obtiene "pegando" **B** al lado de **A**:

Análogamente, si **A** y **B** tienen el mismo número de columnas, entonces **[A;B]** es la matriz que se obtiene "pegando" **B** debajo de **A**:

EJEMPLO

```
>> A=[1,2;3,4]
>> B=[1;1]
>> C=[A,B]
```

$$A=\begin{pmatrix}1&2\\3&4\end{pmatrix};\ B=\begin{pmatrix}1\\1\end{pmatrix};\ C=[A,B]=\begin{pmatrix}1&2&1\\3&4&1\end{pmatrix};$$

2.2 Operaciones con vectores y matrices

Si son de las mismas dimensiones, los vectores / matrices se pueden sumar y restar

EJEMPLO

```
>> V=[1;-3;0]
>> W=[0;3;-2]
>> Z=V+W
```

$$v=\begin{pmatrix}1\\-3\\0\end{pmatrix};\ w=\begin{pmatrix}0\\3\\-2\end{pmatrix};\ z=v+w=\begin{pmatrix}1+0\\-3+3\\0-2\end{pmatrix}=\begin{pmatrix}1\\0\\-2\end{pmatrix}$$

Los vectores / matrices se pueden multiplicar por un número: se multiplica cada elemento por dicho número

EJEMPLO

```
>> A=[1, 2;-3, -1];
>> Z=3*A
```

$$A=\begin{pmatrix}1&2\\-3&-1\end{pmatrix};\ z=3A=\begin{pmatrix}3&6\\-9&-3\end{pmatrix};$$

Una matriz se puede multiplicar por un vector columna si coincide el número de columnas de la matriz con la longitud del vector

EJEMPLO

```
>> A=[1, 2;-3, -1]
>> V=[2;-1]
>> Z=A*V
```

$$A=\begin{pmatrix}1&2\\-3&-1\end{pmatrix};\ v=\begin{pmatrix}2\\-1\end{pmatrix};\ z=Av=\begin{pmatrix}2-2\\-6+1\end{pmatrix}=\begin{pmatrix}0\\-5\end{pmatrix}$$

Las funciónes

```
>> det(A)
>> rank(A)
```

calculan, respectivamente, el determinante y el rango de una matriz A

2.3 Resolución de sistemas lineales de ecuaciones

Un sistema lineal de ecuaciones

$$\begin{cases} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n=b_1\\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n=b_2\\ \qquad\qquad\ldots\\ a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n=b_n \end{cases}$$

se puede escribir en forma matricial **Ax=b**, con

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}; \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix};$$

Conocidos **A** y **b**, el problema de encontrar **x** tal que **Ax=b** se resuelve fácilmente con MATLAB, con la orden:

>> x=A\b

EJEMPLO

Resolver el sistema lineal de ecuaciones:

$$\begin{aligned} 3x_1 + 2x_2 - x_3 &= 1 \\ 2x_1 - x_2 + x_3 &= -1 \\ x_1 - 2x_2 + x_3 &= 2 \end{aligned}$$

>> A=[3,2,-1;2,-1,1;1,-2,1]; → la matriz del sistema
>> b=[1;-1;2] → el segundo miembro del sistema
>> rank(A) → como son ambos rangos iguales a 3, el sistema es
>> rank([A,b]) → compatible determinado, es decir, con solución única
>> x=A\b → cálculo de la solución
>> A*x → comprobación de la solución

3. REPRESENTACIÓN GRÁFICA DE FUNCIONES

La forma más sencilla de dibujar, con MATLAB, una función y=f(x) es con la orden:

>> ezplot('expresion de la funcion')

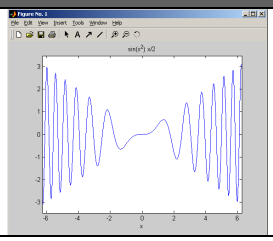
Esta orden dibuja la gráfica de la función dada por la expresión, para x variando en el intervalo $[-2\pi, 2\pi]$.

EJEMPLO

>> ezplot('sin(x^2)*x/2')

dibuja la gráfica de la función

$$f(x) = \frac{x\ \text{sen}(x^2)}{2} \quad \text{en } [-2\pi, 2\pi]$$



Si se quiere dibujar la función en un intervalo distinto, [a,b], hay que indicarlo expresamente en la orden:

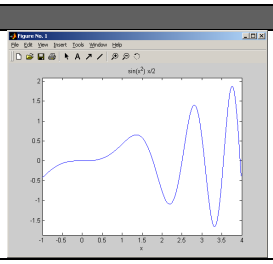
>> ezplot('expresion de la funcion',[a,b])

EJEMPLO

>> ezplot('sin(x^2)*x/2',[-1,4])

dibuja la gráfica de la función

$$f(x) = \frac{x\ \text{sen}(x^2)}{2} \quad \text{en } [-1, 4]$$

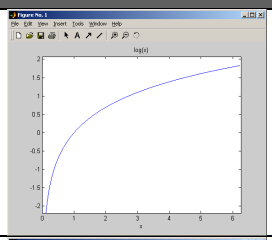


Si no se indica el intervalo y la función que se quiere dibujar no está definida en todo el intervalo $[-2\pi, 2\pi]$, MATLAB la dibujará sólo en el intervalo en que esté definida:

EJEMPLOS

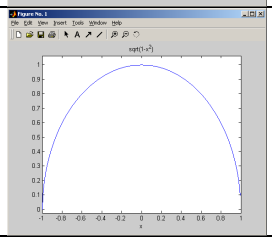
```
>> ezplot('log(x)')
```

La función **ln(x)** sólo está definida para **x>0;** la orden anterior dibuja su gráfica en $[0, 2\pi]$



```
>> ezplot('sqrt(1-x^2)')
```

sólo está definida en **[-1,1]**

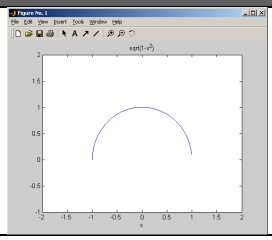


Una vez hecha una gráfica, se puede modificar la amplitud de los ejes (el rectángulo del plano XY que es visible), mediante la orden:

>> axis([xmin , xmax , ymin , ymax])

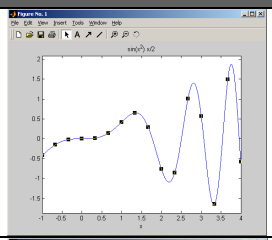
EJEMPLO

```
>> ezplot('sqrt(1-x^2)')

>> axis([-2,2,-1,2])
```

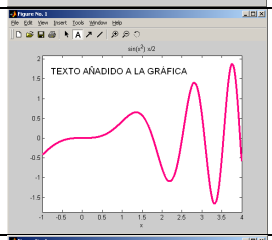


MODIFICACIONES DE LAS GRÁFICAS

Pulsando con el ratón en el botón "Edit Plot" de la barra de herramientas, se pueden modificar algunas características de la gráfica, como el grosor de la línea, el color, ...

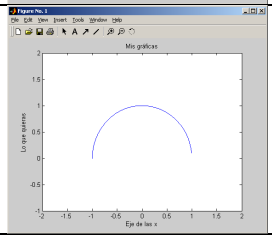


Pulsando con el ratón en el botón "Insert Text" de la barra de herramientas, se puede incluir texto en la gráfica ...



Se puede añadir un título y etiquetas a los ejes:

```
>> title('Mis gráficas')
>> xlabel('Eje de las x')
>> ylabel('Lo que quieras')
```



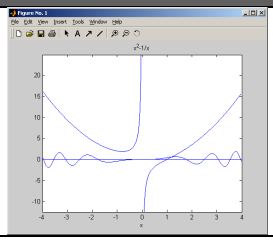
Cada vez que se dibuja una gráfica nueva se borra la anterior, si la había. Si se desean hacer varias gráficas, "una encima de otra", sin que se borren las anteriores, se pueden usar las órdenes

```
>> hold on
  ...
>> hold off
```

La orden **hold on** hace que no se borre el contenido de la ventana gráfica cuando se den nuevas órdenes de dibujo. Se suspende con **hold off**

EJEMPLO

```
>> ezplot('sin(x^2)*x/2',[-4,4])
>> hold on
>> ezplot('0',[-4,4])
>> ezplot('x^2-1/x',[-4,4])
>> hold off
```



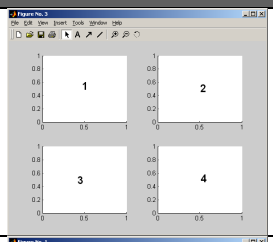
También se pueden dibujar varias gráficas **separadas** en la misma ventana, usando la orden

>> subplot(m,n,p)

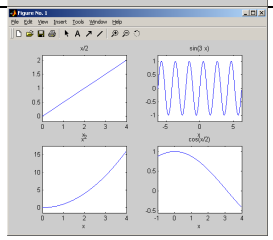
Esta orden divide la ventana gráfica en mxn "ejes" (cuadros blancos), y se dispone a dibujar en el p-ésimo de ellos. Los ejes se numeran correlativamente, de izquierda a derecha y de arriba hacia abajo.

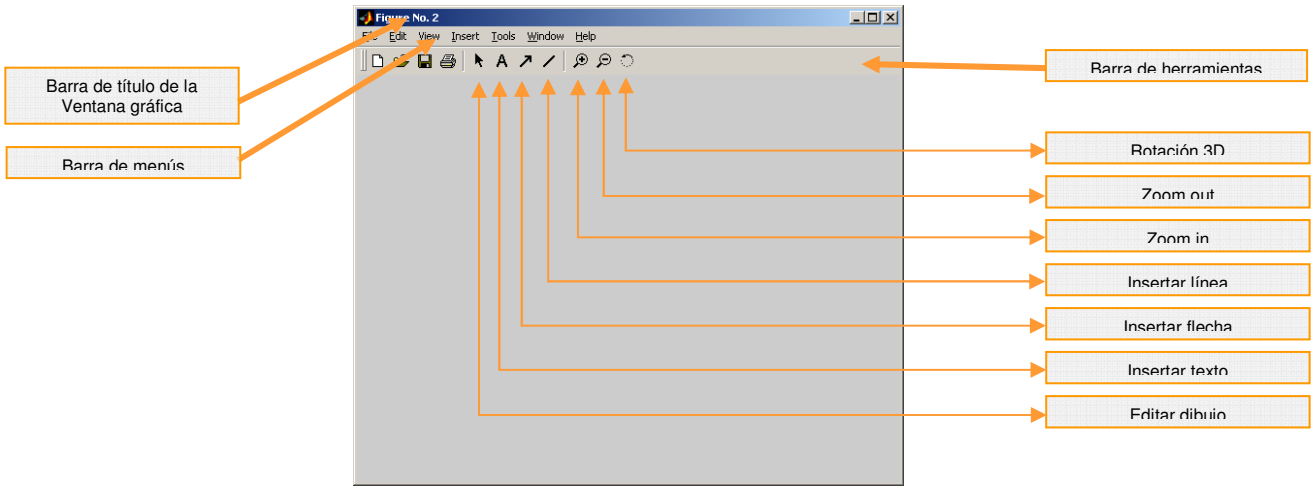
EJEMPLO

```
>> subplot(2,2,1)
>> ... dibujos ...
>> subplot(2,2,3)
>> ... dibujos ...
```




```
>> subplot(2,2,1)
>> ezplot('x/2',[0,4])
>> subplot(2,2,2)
>> ezplot('sin(3*x)')
>> subplot(2,2,3)
>> ezplot('x^2',[0,4])
>> subplot(2,2,4)
>> ezplot('cos(x/2)',[-1,4])
```





4. CÁLCULO DE RAÍCES DE ECUACIONES, MÍNIMOS DE FUNCIONES E INTEGRALES DEFINIDAS

4.1 Raíces de ecuaciones

Para calcular con MATLAB una raíz de la ecuación **f(x)=0** , es decir, un punto **x** en el cual la función **f** vale **0**, se usa la orden

```
>> fzero('expresion de la funcion',xprox)
```

donde **xprox** debe ser un valor "próximo" a la raíz buscada. Para elegir **xprox** se puede, en primer lugar, dibujar la función **y=f(x)** y buscar, "a ojo", un valor próximo.

EJEMPLO

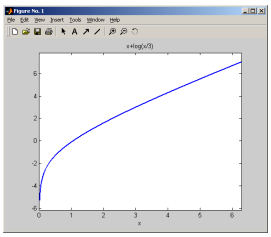
Calcular, con MATLAB, una ráiz de la ecuación

$$x + \ln(\frac{x}{3}) = 0$$

```
>> ezplot('x+log(x/3)')
```

Vemos, a "simple vista", que la raiz está cerca de **x=1**

```
>> fzero('x+log(x/3)',1)
ans =
    1.0499
```



Si la ecuación **f(x)=0** tiene más de una raíz, es necesario tomar "puntos próximos" distintos para cada raíz:

EJEMPLO

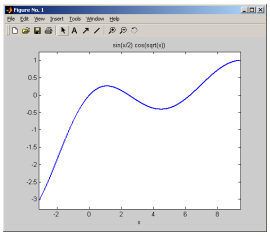
Calcular, con MATLAB, las raíces de la ecuación

$$\text{sen}(\frac{x}{2})\cos(\sqrt{x}) = 0 \quad \text{en } [-\pi, 3\pi]$$

```
>> ezplot('sin(x/2)*cos(sqrt(x))',[-pi,3*pi])
```

A "simple vista" se observa que tiene 3 raíces: una "cerca" de **x=0,** otra "cerca" de **x=2** y otra "cerca" de **x=6**

```
>> fzero('sin(x/2)*cos(sqrt(x))',0)
ans =
     0
>> fzero('sin(x/2)*cos(sqrt(x))',2)
ans =
     2.4674

>> fzero('sin(x/2)*cos(sqrt(x))',6)
ans =
     6.2832
```



4.2 Raíces de polinomios

Si lo que queremos calcular son las raíces de un polinomio

$$c_1 x^N + c_2 x^{N-1} + \ldots + c_N x + c_{N+1} = 0$$

se puede usar la orden **roots**, que calcula TODAS las raíces del polinomio (incluídas las raíces complejas, si las tiene):

```
>> roots(p)
```

donde p es el vector cuyas componentes son los coeficientes del polinomio, ordenados en orden decreciente de potencias de x:

$$p = \begin{pmatrix} c_1 & c_2 & \ldots & c_N & c_{N+1} \end{pmatrix}$$

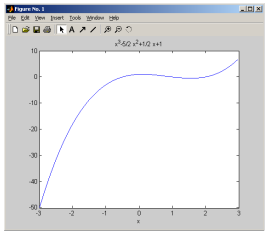
EJEMPLO

Calcular las raíces de la ecuación

$$x^3 - \frac{5}{2}x^2 + \frac{1}{2}x + 1 = 0$$

Vemos que se trata de un polinomio de grado 3:

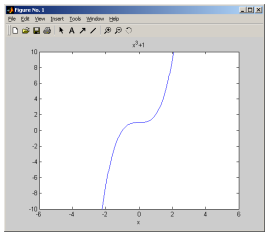
```
>> p=[1,-5/2,1/2,1]
>> roots(p)
ans =
      2.0000
      1.0000
     -0.5000
```



Calcular las raíces de la ecuación

$$x^3 + 1 = 0$$

```
>> p=[1,0,0,1]
>> roots(p)
ans =
  -1.0000
   0.5000 + 0.8660i
   0.5000 - 0.8660i
```



4.3 Mínimos de funciones

Para calcular el (punto en el que se produce el) mínimo de una función **y=f(x)** en un intervalo **[a,b]**, se puede usar la orden:

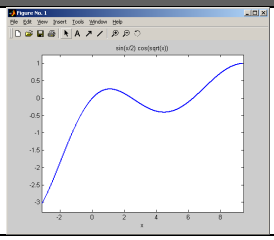
>> fminbnd('expresion_de_la_funcion',a,b)

EJEMPLO

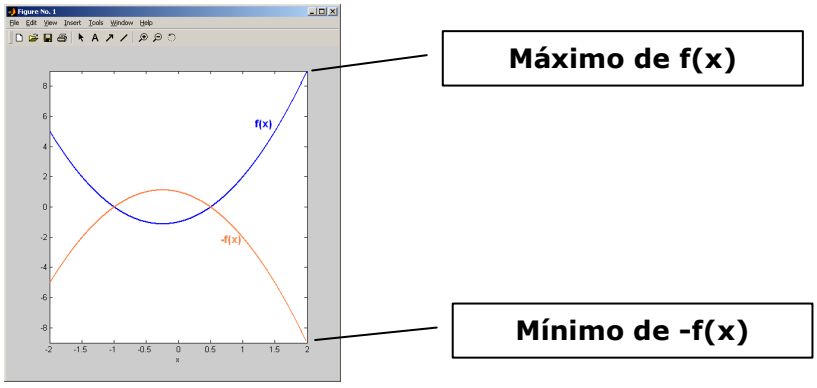
Calcular el mínimo de la función

$$f(x) = 2x^2 + x - 1 \quad \text{en } [-2, 2]$$

```
>> fminbnd('2*x^2+x-1',-2,2)
ans =
   -0.2500
```



Para calcular el máximo de una función **y=f(x)** en un intervalo **[a,b]**, hay que calcular el mínimo de la función **y=-f(x)** en el mismo intervalo

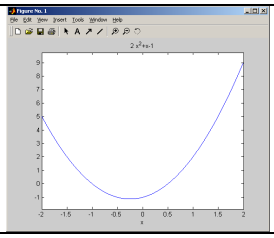


EJEMPLO

Calcular el máximo de la función

$$f(x) = 2x^2 + x - 1 \quad \text{en } [-2, 2]$$

```
>> fminbnd('-(2*x^2+x-1)',-2,2)
ans =
2
```



4.4 Cálculo de integrales definidas

Para calcular el valor de la integral definida

$$\int_a^b f(x)\,dx$$

se puede usar la orden

>> quad(vectorize('expresion'),a,b)

EJEMPLO

Calcular el valor de la integral

$$\int_{-\pi/2}^{\pi/2} \text{sen}(x)\,dx$$

```
>> quad(vectorize('sin(x)'),-pi/2,pi/2)
```

5. RESOLUCIÓN NUMÉRICA DE ECUACIONES DIFERENCIALES ORDINARIAS: APLICACIÓN A MODELOS DIFERENCIALES DE LA DINÁMICA DE POBLACIONES

5.1 Problema de valor inicial para una ecuación diferencial ordinaria

Un problema de valor inicial para una ecuación diferencial ordinaria (en adelante EDO) es un problema del tipo

$$(P)\ \begin{cases} y' = f(t,y) \\ y(t_0) = y_0 \end{cases}$$

en el que se trata de encontrar, de entre todas las soluciones de la ecuación diferencial, aquella que en $\mathbf{t_0}$ vale $\mathbf{y_0}$.

EJEMPLO

La ecuación diferencial $y' = y - 1$ tiene infinitas soluciones:

$$y = Ce^t + 1$$

Para cada valor que tome C se obtiene una solución distinta:

$$y = 2e^t + 1 \qquad y = \frac{1}{2}e^t + 1 \qquad y = -e^t + 1$$

Si buscamos, entre todas estas funciones, alguna que en t=0 tome el valor y=2, encontraremos que la única que verifica esa condición es:

$$y = e^t + 1$$

Este problema se escribe:

$$(P)\ \begin{cases} y' = y - 1 \\ y(0) = 2 \end{cases}$$

Con MATLAB no se puede calcular la expresión EXACTA de la solución de (P): sólo se pueden calcular, de forma APROXIMADA, los valores de la función solución en algunos puntos.

Para calcular, con MATLAB, la solución aproximada del problema:

$$(P)\ \begin{cases} y' = f(t,y) \text{ en } [t_0, t_f] \\ y(t_0) = y_0 \end{cases}$$

se usan los comandos:

```
>> f=inline('expresion de f(t,y)','t','y')
>> ode23(f,[t0,tf],y0)
```

Esta orden calcula una aproximación numérica de la solución del problema (P) y dibuja su gráfica.

```
>> [t,y]=ode23(f,[t0,tf],y0)
```

Esta orden calcula una aproximación numérica de la solución, pero no dibuja su gráfica. La aproximación calculada queda almacenada en los vectores, de la misma dimensión **t** e **y**, cuya interpretación es la siguiente: el valor de la solución del problema (P) en **t(k)** es aproximadamente **y(k).**
La longitud de los vectores **t** e **y** no se conoce "a priori". Se puede averiguar con la orden:

```
>> length(t)
```

que nos dice el número de componentes del vector t.

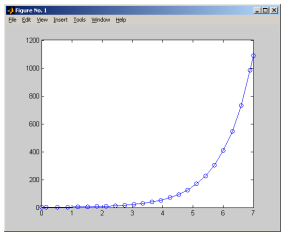
EJEMPLO

Hallar la solución del problema

$$(P)\ \begin{cases} y' = y - 1 \\ y(0) = 2 \end{cases}$$

en el intervalo **[0,7]**.

```
>> f=inline('y-1','t','y')
>> ode23(f,[0,7],2)
```



EJEMPLO

Se ha comprobado que, en algunas circunstancias, el número de individuos de determinadas poblaciones de bacterias se rige por la ley siguiente

$$y'(t) = 0.2y(t)$$

(que se suele escribir $y' = 0.2y$).

La variable **t** es el tiempo, medido en horas, e **y(t)** es el número de miles de bacterias que hay en el instante **t.**
Al comienzo de un determinado experimento hay 30 miles de bacterias.

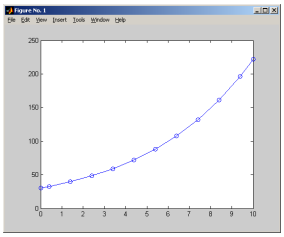
¿Cuántas habrá 10 horas más tarde?
¿En qué instante habrá 100.000 bacterias?

Si se mide el tiempo en horas y se empieza a contar al comienzo del experimento, puesto que se desea saber cuántas bacterias habrá pasadas **10 horas**, se tiene que el problema a resolver es:

$$\begin{cases} y' = 0.2y & \text{en } [0,10] \\ y(0) = 30 \end{cases}$$

```
>> f=inline('0.2*y','t','y')
f =
     Inline function:
     f(t,y) = 0.2*y

>> ode23(f,[0,10],30)
```

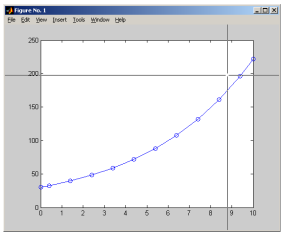
"A simple vista" se puede ver que el valor en **t=10** es aproximadamente **220**, lo que significa que pasadas **10 horas** habrá (aprox.) **220.000 bacterias.**



Si se quiere el resultado con un poco más de exactitud, se puede usar el comando

```
>> ginput(1)
```

que activa un cursor gráfico con el que se puede señalar el punto de la gráfica que se quiera, y obtener sus coordenadas.



Si se desea aún mayor exactitud, se puede usar la orden siguiente

```
>> [t,y]=ode23(f,[0,10],30)
```

y ver cuanto vale la última componente del vector **y**:

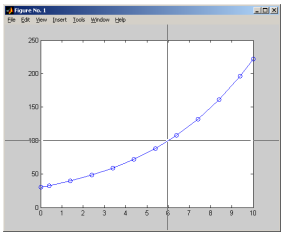
```
>> y(end)
ans =
  221.5562
```

¿Cuándo habrá 100.000 bacterias?
Usando de nuevo el comando

```
>> ginput(1)
ans =
5.9793   99.7807
```

Lo que signfica que, en el instante **t= 5.9793** (es decir, pasadas aproximadamente **6 horas**) el número de bacterias es **y= 99.7807** (aprox. **100.000**).

Observación: Debe tenerse en cuenta que este procedimiento proporciona una aproximación de muy poca calidad, ya que se obtiene "a pulso" y sobre una gráfica de la solución.



EJEMPLO (LEY DE MALTHUS)

Se supone que la población de un país era de 39.5 millones en el año 2000 y que crece siguiendo la ley (de Malthus)

$$y' = 0.05y$$

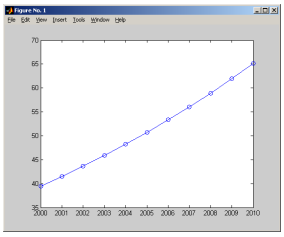
donde **y(t)** representa el número de millones de habitantes en el instante **t**.
¿Cuantos millones de habitantes habrá en 2010?
¿Cuando se alcanzarán los 50 millones de habitantes?

Puesto que se desea saber el valor de la solución en **t=2010**, se debe resolver la ecuación entre **t=2000** (tiempo inicial) y **t=2010**, luego el problema a resolver es:

$$\begin{cases} y' = 0.05y & \text{en } [2000, 2010] \\ y(2000) = 39.5 \end{cases}$$

```
>> f=inline('0.05*y','t','y')
f =
     Inline function:
     f(t,y) = 0.05*y

>> ode23(f,[2000,2010],39.5)
```

"A simple vista" se puede ver que el valor en **t=2010** es aproximadamente **65**, lo cual significa que, en el año **2010** habrá, aproximadamente, **65** millones de habitantes.



¿Cuando se alcanzarán los 50 millones de habitantes?

Usando de nuevo el comando

```
>> ginput(1)
```

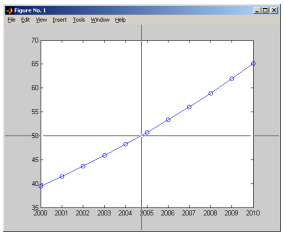
y señalando el punto de la gráfica de la solución en el que la línea horizontal coincide con el valor y=50 del eje vertical, se tendrá:

```
ans =

  1.0e+003 *

    2.0047    0.0500
```

que es lo mismo que

```
ans =
    2004.7    50.
```

lo que significa que en el instante **t=2004,7** (aproximadamente agosto del año 2004) habrá (aprox.) **50** millones de habitantes.

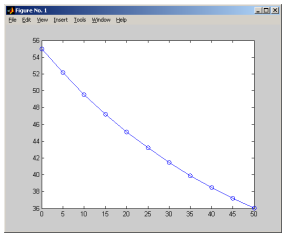


EJEMPLO (LEY DE ENFRIAMIENTO DE NEWTON) (LINEAL)

Un objeto se coloca en una habitación que está a temperatura constante de **25º**. Se sabe que la constante de enfriamiento del objeto es **0.02**.
Si al comienzo la temperatura del objeto es de **55º**.
¿Que temperatura tendrá después de **50 minutos**?
¿Cuánto tarda en ponerse a una temp. de **46º**?
¿Que pasa con la temperatura si pasa mucho tiempo?

De acuerdo con la Ley de enfriamiento de Newton, la temperatura del objeto es la solución del problema:

$$(P)\ \begin{cases} y' = 0.02(25 - y) \\ y(0) = 55 \end{cases}$$



Puesto que se desea saber cuánto vale en t=50, lo adecuado es resolver este problema en el intervalo **[0,50]**.

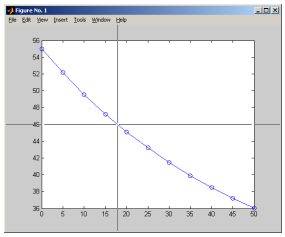
```
>> f=inline('0.02*(25-y)','t','y')
>> ode23(f,[0,50],55)
```

"A simple vista" se ve que la respuesta a la primera pregunta es es **36º**

¿Cuánto tarda en ponerse a una temp. de **46º**?

Usando de nuevo el comando

```
>> ginput(1)
```



y señalando el punto de la gráfica de la solución en el que la línea horizontal coincide con el valor y=46 del eje vertical, se tendrá (aproximadamente):

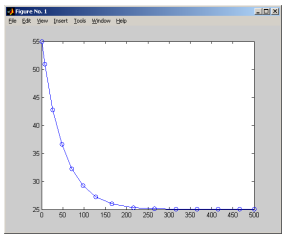
```
ans =
   18.0300   45.9708
```

lo que significa que la respuesta es **18 minutos**

¿Que pasa con la temperatura si pasa mucho tiempo?

Para contestar a esta pregunta, se resuelve el problema en un intervalo grande de tiempo:

```
>> ode23(f,[0,500],55)
```



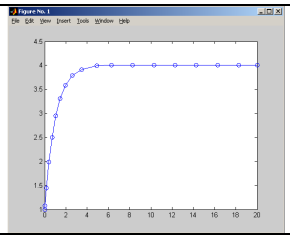
la simple observación de la gráfica muestra que, si se deja pasar mucho tiempo, la temperatura del objeto se estabiliza en el valor de **25º**, es decir, la temperatura del ambiente.

EJEMPLO (LEY LINEAL)

Calcular una solución aproximada del siguiente problema

$$(P)\ \begin{cases} y' = -y + 4 \text{ en } [0, 20] \\ y(0) = 1 \end{cases}$$

```
>> f=inline('-y+4','t','y')
>> ode23(f,[0,20],1)
```



Para comparar el comportamiento de distintas soluciones de una ecuación correspondientes a distintas condiciones iniciales, puede interesar dibujarlas juntas. Para ello se puede utilizar el comando **hold on**, que no borra la ventana gráfica antes de realizar un nuevo dibujo. Para evitar que cada nueva orden **ode23** redefina el cuadro de dibujo, se puede utilizar previamente la orden **axis** para fijarlo.

EJEMPLO (LEY LINEAL)

Dibujar en la misma ventana gráfica, en el cuadro **[0,20]x[-15,15]**, usando el comando **hold on**, las soluciones del problema

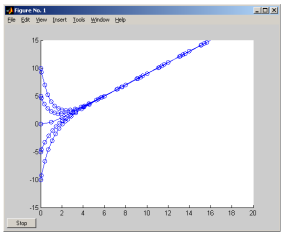
$$(P)\ \begin{cases} y' = -y + t \text{ en } [0,20] \\ y(0) = y_0 \end{cases}$$

correspondientes a distintos valores de y_0 entre **-10** y **10**

```
>> f=inline('-y+t','t','y')
>> axis([0,20,-15,15]); hold on
>> ode23(f,[0,20],-10)
>> ode23(f,[0,20],-5)
>> ode23(f,[0,20],0)
>> ode23(f,[0,20],5)
>> ode23(f,[0,20],10)
```

Obsérvese que todas las soluciones se "acercan" a la solución **y=t-1**.

Esto significa que, sea cual sea la condición inicial de la que se parte, todas las soluciones de esta ecuación tienden, pasado un cierto tiempo, a comportarse de la misma manera.



EJEMPLO (LEY LINEAL)

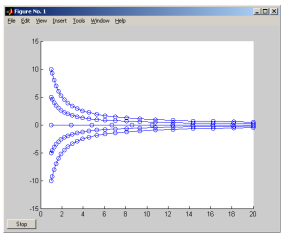
Dibujar en la misma ventana gráfica, en el cuadro **[0,20]x[-15,15]**, usando el comando **hold on**, las soluciones del problema

$$(P)\ \begin{cases} y' = -y/t \text{ en } [1,20] \\ y(1) = y_0 \end{cases}$$

correspondientes a distintos valores de y_0 entre **-10** y **10**

```
>> f=inline('-y/t','t','y')
>> axis([0,20,-15,15]); hold on
>> ode23(f,[1,20],-10)
>> ode23(f,[1,20],-5)
>> ode23(f,[1,20],0)
>> ode23(f,[1,20],5)
>> ode23(f,[1,20],10)
```

Obsérvese que todas las soluciones se "acercan" a la solución (ESTACIONARIA) **y=0**, aunque tras un intervalo de tiempo mayor que en el ejemplo anterior.



EJEMPLO (LEY LINEAL)

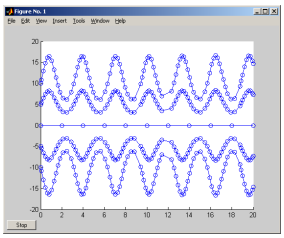
Dibujar en la misma ventana gráfica, en el cuadro **[0,20]x[-20,20]**, usando el comando **hold on**, las soluciones del problema

$$(P)\begin{cases} y' = cos(2t)y \text{ en } [0,20] \\ y(0) = y_0 \end{cases}$$

correspondientes a distintos valores de y_0 entre **-10** y **10**

```
>> f=inline('cos(2*t)*y','t','y')
>> axis([0,20,-20,20]); hold on
>> ode23(f,[0,20],-10)
>> ode23(f,[0,20],-5)
>> ode23(f,[0,20],0)
>> ode23(f,[0,20],5)
>> ode23(f,[0,20],10)
```

Aquí también aparece la solución estacionaria **y=0**, pero las demás no se "acercan" a ella.



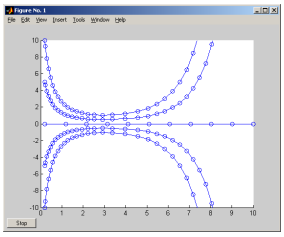
EJEMPLO (LEY LINEAL)

Dibujar en la misma ventana gráfica, en el cuadro **[0,10]x[-10,10]**, usando el comando **hold on**, las soluciones del problema

$$(P)\begin{cases} y' = \ln(t/3)y \text{ en } [0.2,10] \\ y(0.2) = y_0 \end{cases}$$

correspondientes a distintos valores de y_0 entre **-10** y **10**

```
>> f=inline('log(t/3)*y','t','y')
>> axis([0,10,-10,10]); hold on
>> ode23(f,[0.2,10], -10)
>> ode23(f,[0.2, 10],-5)
>> ode23(f,[0.2, 10], 0)
>> ode23(f,[0.2, 10], 5)
>> ode23(f,[0.2, 10],10)
```



EJEMPLO (LEY LOGÍSTICA)

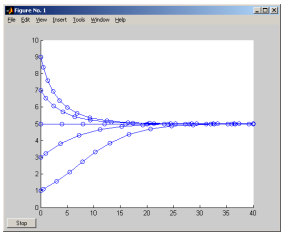
Dibujar en la misma ventana gráfica, en el cuadro [0,40]x[0,10], usando el comando **hold on**, las soluciones del problema

$$(P)\begin{cases} y' = 0.2y - 0.04y^2 \text{ en } [0,40] \\ y(0) = y_0 \end{cases}$$

correspondientes a distintos valores de y_0 entre **0** y **10**

```
>> f=inline('0.2*y-0.04*y^2','t','y')
>> axis([0,40,0,10]); hold on
>> ode23(f,[0, 40], 1)
>> ode23(f,[0, 40], 3)
>> ode23(f,[0, 40], 5)
>> ode23(f,[0, 40], 7)
>> ode23(f,[0, 40], 9)
```

Obsérvese que todas las soluciones tienden a "acercarse" a la solución estacionaria **y=5**



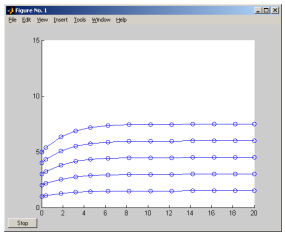
EJEMPLO (LEY DE GOMPERTZ)

Dibujar en la misma ventana gráfica, en el cuadro [0,20]x[0,15], usando el comando **hold on**, las soluciones del problema

$$(P)\ \begin{cases} y' = 0.2\ e^{-\frac{1}{2}t}\ y \text{ en } [0,20] \\ y(0) = y_0 \end{cases}$$

correspondientes a distintos valores de y_0 entre **0** y **5**

```
>> f=inline('0.2*exp(-0.5*t)*y','t','y')
>> axis([0,20,0,15]); hold on
>> ode23(f,[0, 20], 1)
>> ode23(f,[0, 20], 2)
>> ode23(f,[0, 20], 3)
>> ode23(f,[0, 20], 4)
>> ode23(f,[0, 20], 5)
```



Si se desean comparar soluciones correspondientes a distintos valores de un parámetro que aparece en la ecuación, se puede redefinir, antes de cada resolución, la función f del segundo miembro.

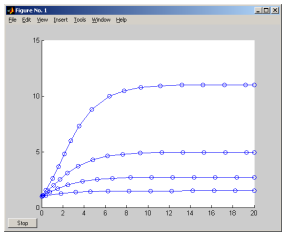
EJEMPLO (LEY DE GOMPERTZ)

Dibujar en la misma ventana gráfica, en el cuadro [0,20]x[0,15], usando el comando **hold on**, las soluciones del problema

$$(P)\ \begin{cases} y' = k\ e^{-\frac{1}{2}t}\ y \text{ en } [0,20] \\ y(0) = 1 \end{cases}$$

correspondientes a distintos valores de **K** entre **0.1** y **1.5**

```
>> axis([0,20,0,15]); hold on
>> f=inline('0.2*exp(-0.5*t)*y','t','y')
>> ode23(f,[0, 20], 1)
>> f=inline('0.5*exp(-0.5*t)*y','t','y')
>> ode23(f,[0, 20], 1)
>> f=inline('0.8*exp(-0.5*t)*y','t','y')
>> ode23(f,[0, 20], 1)
>> f=inline('1.2*exp(-0.5*t)*y','t','y')
>> ode23(f,[0, 20], 1)
```



EJEMPLO (MODELO DE PESCA)

En algunos modelos de pesca se utiliza la ecuación:

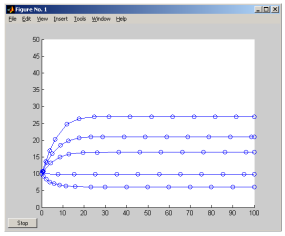
$$y' = \alpha\ \ln(\tfrac{k}{y})\ y - q\ y$$

donde el término **-qy** mide el efecto negativo que ejerce la pesca sobre el crecimiento de la población. Estos modelos ayudan a analizar la sostenibilidad de los bancos de pesca.
Calcular y dibujar (en el cuadro **[0,100] x [0,50]**, y usando **hold on**) las soluciones del problema

$$\begin{cases} y' = 0.2\ \ln(\frac{27}{y})\ y - q\ y \\ y(0) = 10 \end{cases}$$

para los valores del parámetro **q** : **0** , **0.05** , **0.1** , **0.2** , **0.3**

```
>> axis([0,100,0,50]); hold on
>> f=inline('0.2*log(27/y)*y-0*y','t','y')
>> ode23(f,[0, 100], 10)
>> f=inline('0.2*log(27/y)*y-0.05*y','t','y')
>> ode23(f,[0, 100], 10)
>> f=inline('0.2*log(27/y)*y-0.1*y','t','y')
>> ode23(f,[0, 100], 10)
>> f=inline('0.2*log(27/y)*y-0.2*y','t','y')
>> ode23(f,[0, 100], 10)
>> f=inline('0.2*log(27/y)*y-0.3*y','t','y')
>> ode23(f,[0, 100], 10)
```



5.2 Problema de valor inicial para un sistema diferencial ordinario

En un sistema diferencial ordinario aparecen varias ecuaciones diferenciales y varias incógnitas. Estos sistemas permiten modelizar situaciones en las que varias poblaciones conviven e interactúan en un mismo habitat.

Un ejemplo es el modelo de Lotka-Volterra, también conocido como modelo de presa-depredador, ya que modeliza la situación en la que hay dos especies que conviven y una de ellas es depredadora de la otra.

Si denotamos por **y_1(t)** el número de presas en el instante **t** y por **y_2(t)** el número de depredadores en el instante **t**, el modelo de Lotka-Volterra establece que el número de individuos de cada especie evoluciona en el tiempo de acuerdo con el sistema diferencial:

$$\begin{cases} y_1' = \ \ a\ y_1 - b\ y_1\ y_2 \\ y_2' = -c\ y_2 + d\ y_1\ y_2 \end{cases}$$

en el que las constantes **a**, **b**, **c** y **d** varían de un caso a otro, ya que dependen de la natalidad y agresividad de cada especie. Obsérvese que ahora se tienen dos incógnitas y dos ecuaciones.

A este sistema habrá que añadir, como en el caso de una sóla ecuación, unas condiciones iniciales que indiquen cúal es la situación de partida, es decir, cúantos individuos de cada especie hay en el enstante inicial:

$$\begin{cases} y_1' = \ \ a\ y_1 - b\ y_1\ y_2 \\ y_2' = -c\ y_2 + d\ y_1\ y_2 \\ y_1(t_0) = A \\ y_2(t_0) = B \end{cases}$$

Para resolver con MATLAB este sistema se debe, en primer lugar, escribir con notación vectorial:

$$\begin{cases} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} a\ y_1 - b\ y_1\ y_2 \\ -c\ y_2 + d\ y_1\ y_2 \end{pmatrix} \\ \begin{pmatrix} y_1(t_0) \\ y_2(t_0) \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix} \end{cases}$$

Ahora es necesario definir la función **f** que depende de **t** y del vector **y**, y que toma valores vectoriales:

$$f(t,y) = f\left(t, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right) = \begin{pmatrix} a\ y_1 - b\ y_1\ y_2 \\ -c\ y_2 + d\ y_1\ y_2 \end{pmatrix}$$

```
>> f=inline('[a*y(1)-b*y(1)*y(2);-c*y(2)+d*y(1)*y(2)]','t','y')
```

Después, la resolución es análoga, observando que la condición inicial también es ahora un vector:

```
>> ode23(f,[t0,tf],[A;B])
```

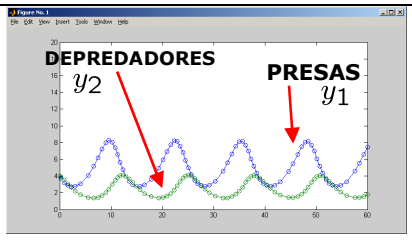
EJEMPLO (MODELO DE LOTKA-VOLTERRA)

Calcular la solución de:

$$\begin{cases} y_1' = \ \ 0.5\ y_1 - 0.2\ y_1\ y_2 \\ y_2' = -0.5\ y_2 + 0.1\ y_1\ y_2 \\ y_1(0) = 4 \\ y_2(0) = 4 \end{cases}$$

```
>> f=inline('[0.5*y(1)-0.2*y(1)*y(2);
    -0.5*y(2)+0.1*y(1)*y(2)]','t','y')
>> ode23(f,[0,60],[4;4])
>> axis([0,60,0,20])
```

Obsérvese el comportamiento periódico de la solución, típico de las soluciones de este modelo.

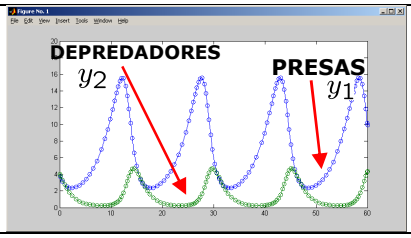


EJEMPLO (MODELO DE LOTKA-VOLTERRA)

Calcular la solución de:

$$\begin{cases} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} 0.3\ y_1 - 0.2\ y_1\ y_2 \\ -0.7\ y_2 + 0.1\ y_1\ y_2 \end{pmatrix} \\ \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} \end{cases}$$

```
>> f=inline('[0.3*y(1)-0.2*y(1)*y(2);
    -0.7*y(2)+0.1*y(1)*y(2)]','t','y')
>> ode23(f,[0,60],[4;4])
>> axis([0,60,0,20])
```

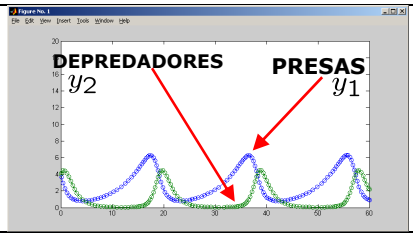


EJEMPLO (MODELO DE LOTKA-VOLTERRA)

Calcular la solución de:

$$\begin{cases} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} 0.2\ y_1 - 0.2\ y_1\ y_2 \\ -0.8\ y_2 + 0.3\ y_1\ y_2 \end{pmatrix} \\ \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} \end{cases}$$

```
>> f=inline('[0.2*y(1)-0.2*y(1)*y(2);
    -0.8*y(2)+0.3*y(1)*y(2)]','t','y')
>> ode23(f,[0,60],[4;4])
>> axis([0,60,0,20])
```



Otro ejemplo de sistema diferencial lo proporciona el modelo para especies en competición: se trata ahora de modelizar la situación en la que dos especies que comparten un mismo hábitat compiten entre ellas por los recursos:

$$\begin{cases} y_1' = a_1\ y_1 - b_1\ y_1^2 - c_1\ y_1\ y_2 \\ y_2' = a_2\ y_2 - b_2\ y_2^2 - c_2\ y_1\ y_2 \\ y_1(t_0) = A \\ y_2(t_0) = B \end{cases}$$

Este problema se resuelve con MATLAB de forma análoga al anterior.

EJEMPLO (MODELO DE ESPECIES QUE COMPITEN)

Calcular la solución de:

$$\begin{cases} y_1' = 0.02y_1 - 0.001y_1^2 - 0.005y_1y_2 \\ y_2' = 0.03y_2 - 0.002y_2^2 - 0.002y_1y_2 \\ y_1(t_0) = 4 \\ y_2(t_0) = 4 \end{cases}$$

```
f=inline('[0.02*y(1)-0.001*y(1)^2-
  0.005*y(1)*y(2); 0.03*y(2)-
  0.002*y(2)^2-0.002*y(1)*y(2)]','t','y')
>> ode23(f,[0,1000],[4;4])
```

Obsérvese que una de las dos especies se extingue.