Ecuaciones diferenciales ordinarias

■ Soluciones exactas con DSolve

Resumen de comandos

DSolve[edo==0,y,x] da la solución general de una EDO en términos de la función **y[x]**.

DSolve[{edo==0,condinic},y,x] resuelve una EDO en términos de la función **y[x]** con condiciones iniciales **condinic**.

DSolve[{edo1==0,edo2==0,...},{y1,y2,...},x] resuelve un sistema de EDOs en términos de las funciones **y1[x], y2[x]**, etc.

DSolve[edp==0,y,{x1,x2,...}] resuelve una EDP (ecuación en derivadas parciales) en términos de la función **y[x1,x2,...]**.

el paquete **<<Calculus`DSolveIntegrals`** proporciona "integrales completas" para EDPs mediante el comando:

CompleteIntegral[edp==0,y,{x1,x2,...}]

El comando **`DSolve`** encuentra soluciones analíticas (no numéricas) para un número cada vez mayor de ecuaciones diferenciales. Por ejemplo:

■ `DSolve` resuelve EDOs lineales con coeficientes constantes de cualquier orden y muchas ecuaciones lineales de hasta segundo orden con coeficientes no constantes.

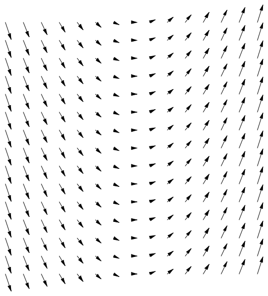
■ `DSolve` incluye procedimientos generales para EDOs no lineales cuyas soluciones vienen tratadas en libros de texto y manuales.

■ `DSolve` puede encontrar soluciones generales para ecuaciones en derivadas parciales lineales (y "débilmente no lineales"). Téngase en cuenta que las ecuaciones diferenciales en derivadas parciales no lineales no presentan usualmente una solución general explícita (éste es un campo de investigación importante donde es necesario avanzar)

Ejemplos de EDOs de orden 1. Isoclinas y método de Euler

Consideremos la ecuación diferencial $y'=x$. El campo de direcciones (isoclinas) asociado a esta ecuación diferencial es $\vec{v}(x, y) = (1, y') = (1, x)$, que representan vectores tangentes a la trayectoria $y(x)$ en cada punto. La representación gráfica del mismo se realiza mediante:

In[1]:=
```
<< Graphics`PlotField`
isoclinas = PlotVectorField[{1, x}, {x, -3, 3}, {y, -3, 3}];
```



La solución general de la ecuación diferencial *y'*=*x* viene dada por:

In[3]:=
```
DSolve[y'[x] == x, y[x], x]
```

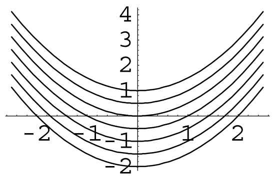
Out[3]= $\left\{\left\{y[x] \to \frac{x^2}{2} + C[1]\right\}\right\}$

que nos da una familia uniparamétrica de curvas, donde C[1] representa la constante de integración. Representemos varias curvas de esta familia pero, para ello, definamos la familia *y(x,c)* para distintas condiciones iniciales $y(0) = c$ como una función **sol[x,c]** de dos variables:

In[4]:=
```
Clear[y, x, c];
sol[x_, c_] := y[x] /. DSolve[{y'[x] == x, y[0] == c}, y[x], x]
```

Definamos una tabla con los 7 miembros c=-2,-1.5,...,0.5,1 de dicha familia uniparamétrica de curvas y representémosla gráficamente:

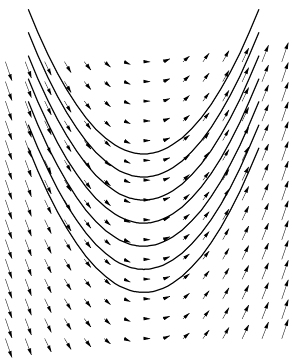
In[6]:=
```
familia = Table[sol[x, c], {c, -2, 1, 0.5}];
familiaplot = Plot[Evaluate[familia], {x, -2.5, 2.5}]
```



Out[7]= - Graphics -

Superponiendo dichas curvas con el campo de direcciones (isoclinas):

In[8]:=
```
Show[isoclinas, familiaplot]
```



Out[8]= - Graphics -

vemos que el campo de direcciones es, efectivamente, tangente a la familia de curvas en cada punto.

Ejercicio

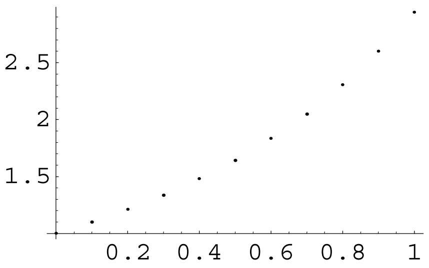
Dibuje el campo de direcciones (isoclinas) de la ecuación diferencial $y'=x^3-x$ en el cuadrado $(x,y)\epsilon[-2,2]\times[-3,3]$ y superponga dicho gráfico al de la familia de soluciones con condiciones iniciales $y(0)=c=-1,-0.5,0,0.5,1$

Método de Euler: Aunque *Mathematica* dispone de métodos numéricos relativamente exactos para resolver ecuaciones diferenciales, diseñemos nosotros mismos un algoritmo para un método "grosero" (poco preciso, no refinado) cómo es el de Euler. Esto nos sirve como "modelo de juguete" para saber cualitativamente cómo funcionan otros métodos numéricos más sofisticados y cómo evaluar el error numérico cometido. Por ejemplo, tomemos la ecuación lineal no homogénea de primer orden $y' = f(x, y) = x^2 + y$, con la condición inicial $y(0)=1$. Aunque sabemos resolver dicha ecuación de forma exacta, abordaremos el problema de forma numérica y compararemos la solución numérica con la exacta para evaluar el error cometido y así hacer un test a nuestro método numérico. Supongamos que queremos una solución numérica de la EDO anterior en el intervalo $x\in[0,1]$. Para ello dividimos el intervalo [0,1] en, por ejemplo, 10 trozos, tomando un paso $h=0.1$ y obteniendo así 11 puntos equiespaciados: $x_0 = 0$, $x_1 = 0.1$, ..., $x_k = x_0 + k\,h$, ..., $x_{10} = 1$. Introduzcamos dicha información en el programa:

In[9]:=
```
Clear[f, x, y, h, k];
f[x_, y_] := x^2 + y;
h = 0.1; x[0] = 0;
x[k_] := x[0] + k * h
```

Discretizando la derivada $y'(x_k) \approx (y(x_{k+1}) - y(x_k))/h = (y_{k+1} - y_k)/h$ obtenemos la versión en diferencias finitas $y_k = y_{k-1} + h\, f(x_{k-1}, y_{k-1})$ de la ecuación diferencial $y' = f(x, y)$. Introduzcamos dicha ecuación en el programa y representemos los 11 puntos (x_0, y_0), ..., (x_{10}, y_{10}):

In[13]:=
```
y[0] = 1;
y[k_] := y[k - 1] + h * f[x[k - 1], y[k - 1]];
eulerplot = ListPlot[Table[{x[k], y[k]}, {k, 0, 10}]]
```

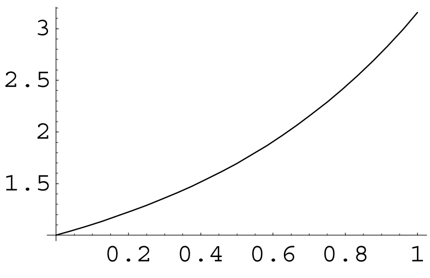


Out[15]= - Graphics -

Calculemos la solución exacta y representémosla gráficamente (cambiaremos x↔t, y↔z para evitar incompatibilidades)

In[16]:=
```
exacta = DSolve[{z'[t] == t^2 + z[t], z[0] == 1}, z, t]
exactaplot = Plot[z[t] /. exacta, {t, 0, 1}]
```

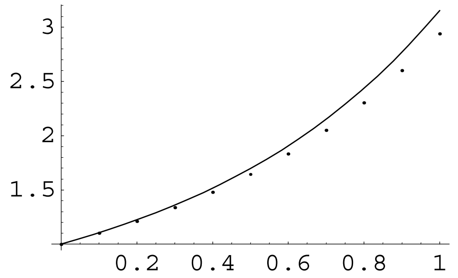
Out[16]= $\{\{z \to (-2 + 3\, e^{\#1} - 2\, \#1 - \#1^2\ \&)\}\}$



Out[17]= - Graphics -

Superpongamos la solución exacta y la numérica:

In[18]:= `Show[eulerplot, exactaplot]`



Out[18]= `- Graphics -`

Vemos cómo la solución numérica se aparta progresivamente de la exacta conforme nos alejamos del punto inicial $(x_0,\ y_0) = (0,\ 1)$. Podemos evaluar el error cometido en el último punto $(x_{10},\ y_{10})$ como $y(1) - y_{10}$:

In[19]:= `Evaluate[z[1] /. exacta] - y[10]`

Out[19]= `{0.214244}`

Es decir, el error es de 0.214244

Ejercicio

Repita los pasos anteriores para 21 puntos, es decir, tomando como paso la mitad del anterior: *h=0.05*. Evalúe el error cometido en el cálculo numérico de *y(1)*. ¿Es el error mayor o menor que para el caso *h=0.1*?.

Ejemplos de EDOs de orden 2. Diagrama de fases

Consideremos la ecuación diferencial lineal de segundo orden con coeficientes constantes x*''(t)=kx(t)* que puede representar la ecuación de un oscilador armónico simple atractivo para *k<0* y repulsivo para *k>0.* La solución general de esta ecuación se obtiene:

In[20]:=
```
Clear[x, t];
DSolve[x''[t] == k x[t], x[t], t]
```

Out[21]= $\left\{\left\{x[t] \to e^{-\sqrt{k}\ t}\ C[1] + e^{\sqrt{k}\ t}\ C[2]\right\}\right\}$

Vemos que contiene dos constantes arbitrarias C[1] y C[2], como corresponde a una EDO de orden 2.

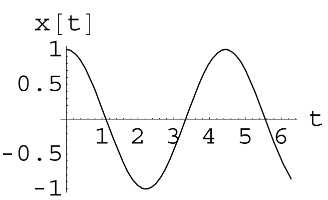
Tomemos el oscilador atractivo con $k = -\omega_0^2$ =-2 y condiciones iniciales x*(0)=1, x'(0)=0* (velocidad inicial nula); ahora la solución será:

In[22]:= `xoscatract = DSolve[{x''[t] == -2 x[t], x[0] == 1, x'[0] == 0}, x, t]`

Out[22]= $\{\{x \to (\text{Cos}[\sqrt{2}\ \#1]\ \&)\}\}$

que representa un movimiento armónico simple de amplitud 1 y frecuencia $\omega_0 = \sqrt{2}$. Para representar gráficamente la posición **x** en función del tiempo **t** escribiremos:

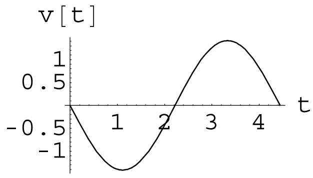
In[23]:= `Plot[x[t] /. xoscatract, {t, 0, 2 Pi}, AxesLabel -> {"t", "x[t]"}]`



Out[23]= `- Graphics -`

donde observamos que el periodo del movimiento es $T=\frac{2\pi}{\omega_0} = 2\pi/\sqrt{2} \approx 4.44288$. Para representar la velocidad **v[t]=x'[t]** en función del tiempo escribimos:

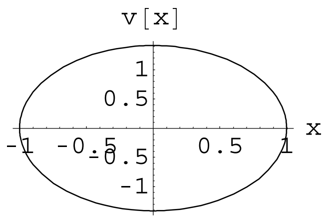
In[24]:=
```
Plot[x'[t] /. xoscatract,
  {t, 0, 2 Pi/Sqrt[2]}, AxesLabel -> {"t", "v[t]"}]
```



Out[24]= - Graphics -

También podemos hacer una representación de la velocidad **v[x]** en función de la posición **x** ("diagrama de fases") por medio de un gráfico en forma paramétrica:

In[25]:=
```
ParametricPlot[Evaluate[{x[t], x'[t]} /. xoscatract],
  {t, 0, 2 Pi/Sqrt[2]}, AxesLabel -> {"x", "v[x]"}]
```



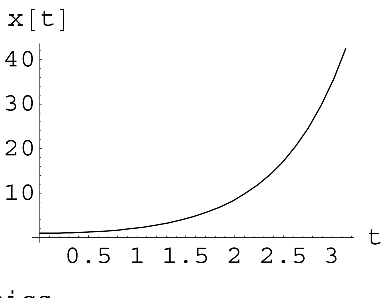
Out[25]= - Graphics -

que nos da una órbita cerrada (elipse), característica de los sistemas periódicos.

Veamos qué pasa con el oscilador repulsivo para *k=2* e idénticas condiciones iniciales que para el atractivo

In[26]:=
```
xoscrepul = DSolve[{x''[t] == 2 x[t], x[0] == 1, x'[0] == 0}, x, t]
Plot[x[t] /. xoscrepul, {t, 0, Pi}, AxesLabel -> {"t", "x[t]"}]
```

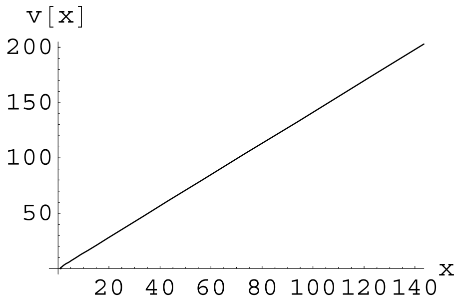
Out[26]= $\left\{\left\{x \to \left(e^{-\sqrt{2}\,\#1}\left(\frac{1}{2}+\frac{1}{2}\,e^{2\sqrt{2}\,\#1}\right)\&\right)\right\}\right\}$



Out[27]= - Graphics -

Vemos que la partícula se aleja de su posición inicial *x(0)=1* de forma exponencial. La trayectoria no se repite a intervalos de tiempo fijos *T;* es decir, a diferencia del oscilador atractivo, el oscilador repulsivo no es un sistema periódico. Veamos que sus trayectorias en el plano de fases tampoco son cerradas:

In[28]:=
```
ParametricPlot[Evaluate[{x[t], x'[t]} /. xoscrepul],
  {t, 0, 2 Pi/√2}, AxesLabel -> {"x", "v[x]"}]
```



Out[28]= - Graphics -

Ejercicio

Dibuje la trayectoria en el plano **x-t** y en el plano de fases **x-x'** de un oscilador armónico simple atractivo con $k = -\omega_0^2 = -5$ que parte de la posición *x(0)=0* con velocidad *x'(0)=1*. Tome el intervalo *t∈[0,T =* $\frac{2\pi}{\omega_0}$ *]*.

Sistemas de EDOs

También podemos resolver sistemas de EDOs,introduciendo éstas mediante una lista.Por ejemplo, para resolver el sistema de dos EDOs lineales con coeficientes constantes $\begin{cases} y_1 = -y_2' \\ y_2 = -y_1' \end{cases}$ escribiremos:

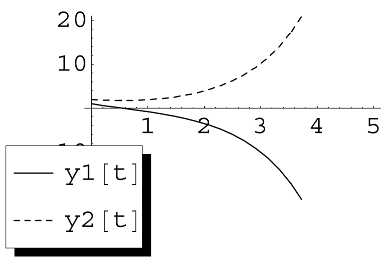
In[29]:=
```
DSolve[{y1[x] == -y2′[x], y2[x] == -y1′[x]}, {y1[x], y2[x]}, x]
```

Out[29]= $\left\{\left\{y1[x] \to \frac{1}{2} e^{-x} (C[1] + e^{2x} C[1] + C[2] - e^{2x} C[2]),\right.\right.$
$\left.\left. y2[x] \to -\frac{1}{2} e^{-x} (-C[1] + e^{2x} C[1] - C[2] - e^{2x} C[2])\right\}\right\}$

que nos da la solución general dependiente de dos constantes arbitrarias C[1] y C[2].Veamos cómo calcular la solución particular que pasa por **{y1[0],y2[0]}={1,2}** y cómo representar **y1[t]** e **y2[t]** superpuestas en un mismo gráfico:

In[30]:=
```
sistema = DSolve[{y1[x] == -y2′[x],
    y1[0] == 1, y2[x] == -y1′[x], y2[0] == 2}, {y1, y2}, x]
<< Graphics`Legend`
Plot[{Evaluate[y1[t] /. sistema], Evaluate[y2[t] /. sistema]},
  {t, 0, 5}, PlotStyle -> {GrayLevel[0.], Dashing[{0.02, 0.02}]},
  PlotLegend -> {"y1[t]", "y2[t]"}]
```

Out[30]= $\left\{\left\{y1 \to \left(-\frac{1}{2} e^{-\#1} (-3 + e^{2\#1}) \&\right), y2 \to \left(\frac{1}{2} e^{-\#1} (3 + e^{2\#1}) \&\right)\right\}\right\}$



Out[32]= - Graphics -

Vemos que ambas funciones se separan indefinidamente de forma exponencial.

Ejercicio

Resolver el sistema $\left\{ \begin{array}{c} y_1' = y_1 \\ y_2' = -y_1 + 2\, y_2 \end{array} \right.$ con condiciones iniciales $y_1(0) = -1$, $y_2(0) = 1$ y representar superpuestas las gráficas de $y_1(x)$ e $y_2(x)$ en el intervalo x∈[0,10].

Ecuaciones en derivadas parciales

Finalmente, veamos cómo introducir una ecuación en derivadas parciales como, por ejemplo, $\partial_x U(x, y, z) = x^2 y / z$

```
In[33]:= DSolve[Derivative[1, 0, 0][U][x, y, z] == x^2 y / z,
          U[x, y, z], {x, y, z}]
```

Out[33]= $\left\{\left\{U[x,\ y,\ z] \to \frac{x^3\, y}{3\, z} + C[1][y,\ z]\right\}\right\}$

que nos da la función *U(x,y,z)*= $\frac{x^3\, y}{3\, z}$ salvo una función arbitraria $C_1(x, y)$ dependiente de sólo dos de las tres variables independientes. También se dispone del paquete:

```
In[34]:= << Calculus`DSolveIntegrals`
         CompleteIntegral[
          Derivative[1, 0, 0][U][x, y, z] == yz, U[x, y, z], {x, y, z}]
```

Out[35]= $\{\{U[x,\ y,\ z] \to x\, yz + B[1] + y\, B[2] + z\, B[3]\}\}$

Ejercicio

Resuelva la ecuación $y\, \partial_x U(x, y) + x\, \partial_y U(x, y) = 0$

■ Soluciones numéricas con NDSolve. Ecuación de Van der Pol

El comando `NDSolve` encuentra soluciones numéricas de ecuaciones diferenciales. Selecciona de forma automática el algoritmo óptimo a usar, o nos permite que lo seleccionemos nosotros si lo conocemos previamente. *Mathematica,* usa automáticamente funciones interpoladoras en vez de listas de números para representar las soluciones numéricas de las ecuaciones diferenciales, haciendo más factible la manipulación posterior de dichas soluciones como, por ejemplo, su integración, derivación, etc.

El comando `NDSolve` utiliza los siguientes algoritmos:

• Por defecto, o mediante la opción `Method->Automatic`, `NDSolve` cambia entre un método de Adams no rígido y un método de Adams rígido adaptado.

• El método de Adams se toma con orden entre 1 y 12

• El método de la fórmula de diferencias atrasadas se toma con orden entre 1 y 5.

• Un orden 4–5 para el método Runge- Kutta se aplica a ecuaciones no rígidas.

• Para problemas lineales con condiciones en la frontera, `NDSolve` emplea el método de búsqueda de Gel'fand- Lokutsiyevskii.

• Para ecuaciones en derivadas parciales de dos variables se utiliza el llamado método de las lineas.

Los algoritmos adaptativos de *Mathematica* y su control de una precisión arbitraria garantiza la precisión de los resultados sin errores de redondeo o soluciones espúreas debidas a errores numéricos.

Ejemplo: solución numérica de la ecuación de Van der Pol

Veamos cómo *Mathematica* es capaz de proporcionar una solución numérica para una EDO no lineal como la ecuación de Van der Pol

$$x''[t] - 0.2\,(1 - x[t]^2)\,x'[t] + x[t] = 0.$$

Esta ecuación aparece en el estudio de las oscilaciones de un circuito RCL con resistencia variable dependiente de la amplitud *x(t).*

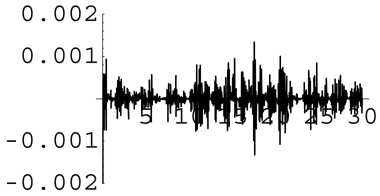
La solución numérica con velocidad inicial nula *x'(0)=0* y posición inicial *x(0)=1* en los primeros 30 segundos (es decir, en el intervalo 0≤t≤30) es:

```
In[36]:= solu1 = NDSolve[{x''[t] - 0.2 (1 - x[t]^2) x'[t] + x[t] == 0,
             x[0] == 1, x'[0] == 0}, x, {t, 0, 30}]

Out[36]= {{x → InterpolatingFunction[{{0., 30.}}, <>]}}
```

Podemos verificar que la solución numérica obtenida verifica con bastante precisión la ecuación diferencial. En efecto, la representación gráfica de $x''[t] - 0.2\,(1 - x[t]^2)\,x'[t] + x[t]$ evaluada en la solución **solu1** anterior

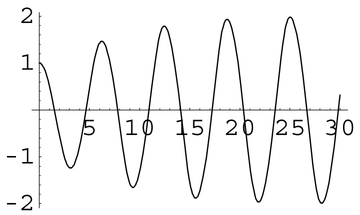
In[37]:=
```
Plot[N[Evaluate[x''[t] - 0.2 (1 - x[t]^2) x'[t] + x[t] /. solu1], 3],
  {t, 0, 30}, PlotRange -> {-0.002, 0.002}]
```



Out[37]= - Graphics -

nos dice que la solución numérica **solu1** verifica la ecuación diferencial en el intervalo t∈[0,30] salvo errores del orden de 0.001. Una vez que hemos ganado confianza en nuestra solución, representémosla en el intervalo escogido:

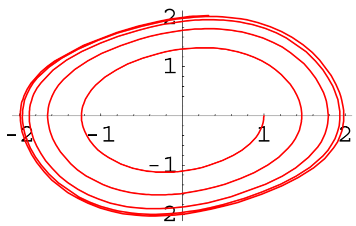
In[38]:=
```
Plot[x[t] /. solu1, {t, 0, 30}]
```



Out[38]= - Graphics -

No queda claro en el gráfico anterior si dicha solución es periódica o no. Una representación en el plano de fases: "*v(x) frente a x*"

In[39]:=
```
ParametricPlot[Evaluate[{x[t], x'[t]} /. solu1[[1]]],
  {t, 0, 30}, PlotStyle -> {Thickness[0.005], RGBColor[1, 0, 0]}]
```



Out[39]= - Graphics -

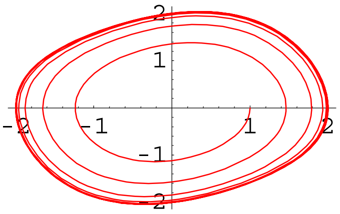
nos muestra que la órbita no es cerrada (al menos en el intervalo [0,30]), lo que da idea de "falta de periodicidad". No obstante, nótese que la órbita se acerca "asintóticamente" ("a largo plazo") a una órbita cerrada "atractriz". Para ver este comportamiento asintótico de la órbita con mayor nitidez, podemos extender el intervalo de tiempo de 30 a 90 segundos y aumentar el número máximo de pasos que **NDSolve** utiliza (por defecto 500) a 2000 mediante la opción **MaxSteps->2000** en:

In[40]:=
```
Clear[x] (* es necesario borrar antes el valor de x*)

solu2 = NDSolve[{x''[t] - 0.2 (1 - x[t]^2) x'[t] + x[t] == 0,
   x[0] == 1, x'[0] == 0}, x, {t, 0, 90}, MaxSteps -> 2000]

ParametricPlot[Evaluate[{x[t], x'[t]} /. solu2[[1]]],
 {t, 0, 90}, PlotStyle -> {Thickness[0.005], RGBColor[1, 0, 0]}]
```

Out[41]=
```
{{x → InterpolatingFunction[{{0., 90.}}, <>]}}
```



Out[42]=
```
- Graphics -
```

Aquí se observa con mayor nitidez que existe una órbita "límite, asintótica o atractriz" (nótese la zona de mayor densidad de líneas) cerrada donde converge (a la cual se acerca cada vez más) nuestro sistema. Esto quiere decir que, aunque el sistema no sea periódico, se comporta como tal a largo plazo (este es un caso particular de un teorema general debido a Alfred Liénard).

Ejercicio

Hallar la solución numérica de la ecuación de Van der Pol con velocidad inicial *x'(0)=2* y posición inicial *x(0)=0* en los primeros 50 segundos (es decir, en el intervalo 0≤t≤50) y representar las gráficas: **x-t** (posición frente a tiempo) y **x'-x** (velocidad frente a posición).