1 Problemas de valor inicial

En esta práctica se analizan los métodos de resolución de problemas de valor inicial con ecuaciones diferenciales ordinarias (E.D.O.s, o en ingles, O.D.E.s). Consideraremos inicialmente métodos numéricos para resolver problemas de valor inicial donde se dan las condiciones iniciales en un único punto. El problema de Cauchy para las E.D.O.s consiste en encontrar una función $y(t):[a,b]\to\Re$ tal que dada la función $f(t,y):[a,b]\times\Re\to\Re$ y $\eta\in\Re$ se verifique

$$\begin{cases} y'(t)=f\left(t,y(t)\right) & a\le t\le b \\ y'(a)=\eta \end{cases}$$

que es, dicho de otra forma, el problema de resolver una ecuación diferencial de primer orden con una condición inicial.

MATLAB puede resolver diferentes tipos de problemas de valor inicial, incluido el de Cauchy:

1. EDO explícitas $y'(t)=f\left(t,y(t)\right)$
2. EDO linealmente implícitas $M(t,y)\cdot y'(t)=f\left(t,y(t)\right)$ donde $M(t,y)$ es una matriz
3. EDo totalmente implícitas $f\left(t,y(t),y'(t)\right)=0$

Generalmente hay más de una función *y(t)* que satisface la ecuación diferencial, por lo que es necesario imponerle una condición inicial, tal y como se ha mostrado en el problema de Cauchy. Si la función $f\left(t,y(t)\right)$ cumple ciertas condiciones (básicamente ser suficientemente suave), existe solución única al problema de valor inicial, aunque no siempre es posible obtenerla de forma analítica.

La mayoría de los métodos de resolución sólo admiten ecuaciones diferenciales de primer orden. Sin embargo las ecuaciones diferenciales a menudo involucran variables dependientes e independientes con derivadas de orden superior. En este caso las ecuaciones se pueden reescribir de forma vectorial para trasformarlas en ODE's de primer orden pero con variables vectoriales, o lo que es lo mismo, en un sistema de ODEs de primer orden. Por ejemplo, la O.D.E. de orden n

$$y^{(n)}=f\left(t,y,y',\cdots y^{(n-1)}\right)$$

mediante el cambio

$$y=y_1 \quad y'=y_2 \quad \cdots \quad y^{(n-1)}=y_{n-1}$$

se transforma en un sistema de ecuaciones diferenciales de primer orden

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \vdots \\ y_{n-1}' = f\left(t, y_1, y_2, \cdots y_{n-1}\right) \end{cases} \Leftrightarrow Y' = f\left(t, Y\right) \quad \text{con} \quad Y = \left(y_1, y_2, \cdots y_{n-1}\right)$$

Sólo en ocasiones es posible obtener la solución analítica. La instrucción *dsolve* obtiene dicha solución, cuando es posible. Su sintaxis es:

r = *dsolve*('eq1,eq2,...', 'cond1,cond2,...','v') ó r = *dsolv*e('eq1','eq2',...,'cond1','cond2',...,'v').

donde las EDO vienen representadas por eq1, eq2,..., v es la variable independiente y esta sujeta a las condiciones de contorno especificadas mediante cond1, cond2,....Por defecto se toma *t* como variable independiente, representándose la diferenciación mediante la letra D mayúscula. Si la D va seguida por un dígito, éste indica el orden de la derivación. El resultado puede ser una variable simbólica cunado hay un solo resultado, un vector cuando la ecuación tiene más de un resultado, o una estructura cuando hay varias ecuaciones y un solo resultado. Si no se encuentra la solución explícita, intenta obtenerla explícita y sino da un mensaje de aviso.

Veámos algunos ejemplos:

Ejemplo	***Instrucción***	***Solución***
$y'(t) = -a \cdot y(t)$	`dsolve('Dy=-a*y')`	$c_1 e^{-at}$
$y'(t) = a \cdot y(t)$ $y(0) = b$	`dsolve('Dy=a*y','y(0)=b')`	be^{at}
$y'(t) = y(t) + \sin(t)$	`dsolve('Df=f+sin(t)')`	$-\frac{1}{2}\left(\cos(t) + \sin(t)\right) + c_1 e^{t}$
$y'(t)^2 + y(t)^2 = 1$	`dsolve('(Dy)^2+y^2=1','s')`	$\{\pm 1, \pm\sin(s\text{-}c_1)\}$
$y'(t)^2 + y(t)^2 = 1$ $y(0) = 0$	`y=dsolve('(Dy)^2+y^2=1','y(0)=0')`	$\pm\sin(t)$
$y'(t) = x(t)$ $x'(t) = y(t)$	`r=dsolve('Dx=y','Dy=-x')`	$r.x = c_1\sin(t) + c_2\cos(t)$ $r.y = c_1\cos(t) - c_2\sin(t)$

Ejercicios recomendados:

1. Resolver las siguientes ecuaciones diferenciales:

$$y' + 2y = te^{2t} \quad 0 \le t \le 1 \quad y(0) = 0$$

$$y'' - 2y' + 2y = e^{2t}\sin t \quad 0 \le t \le 1 \quad y(0) = -0.4 \quad y'(0) = -0.6$$

$$y' = 5e^{5t}\left(y - t\right)^2 + 1 \quad 0 \le t \le 1 \quad y(0) = -1$$

1.1 Instrucciones de MATLAB

La siguiente tabla muestra las instrucciones para resolver numéricamente ecuaciones diferenciales ordinarias (ODEs) y ecuaciones diferenciales algebraicas (DAOs) con los diferentes tipos de problemas de valor inicial y el método que utiliza para su resolución:

Instrucción	*Problemas que resuelve*	*Método*
ode45	Ecuaciones diferenciales no rígidas	Runge-Kutta
ode23	Ecuaciones diferenciales no rígidas	Runge-Kutta
ode113	Ecuaciones diferenciales no rígidas	Adams Bashford Moulton
ode15s	Ecuaciones diferenciales rígidas y ecuaciones diferenciales algebraicas	NDFs (BDF[1]s)
ode23s	Ecuaciones diferenciales rígidas	Rosenbrock
ode23t	Ecuaciones diferenciales semi-rígidas y ecuaciones diferenciales algebraicas	Regla trapezoidal
ode23tb	Ecuaciones diferenciales rígidas	TR-BDF2
ode15i	Ecuaciones diferenciales no rígidas totalmente implícitas	BDFs

La sintaxis general en todos ellos es

```
[t,y] = solver(odefun,tspan,y0,options)
```

donde "solver" es una de las funciones indicadas anteriormente. Los argumentos básicos son:

odefun: maneja la function que evalua el sistemas de ODE. Viene definida mediante "dydt= odefun(t,y)", donde t es un escalar y dydt e y son vectores columna.

tspan: Vector que define el intervalo de integración. La condición inicial esta definida en *tspan(1)* y se integra hasta *tspan(end).*

y0: Vector con las condiciones iniciales del problema.

options: Estructura con los parámetros opcionales, algunos dependientes del solver, que modifica alguna de las características del método de integración

MATLAB también dispone de instrucciones para evaluar los resultados, extender la solución o representarla, así como diferentes opciones para la resolución y evaluación:

Instrucción	*Descripción*
deval	Evalúa la solución numérica usando la salida de uno de los solver anteriores
odeextend	Extiende la solución de un problema de valor inicial para una ODE
odeset	Modifican y recuperan respectivamente la estructura que contiene las propiedades cuyos valores se pasan a los solvers y que afectan a la solución del problema
odeget	
OutputFcn	Si en las opciones de la estructura se define una función de salida, el solver la ejecuta en cada paso de la iteración. Dicha función puede ser especificada por el usuario, o bien utilizar alguna de las predefinidas:

[1] Fórmulas de diferenciación numérica (numerical differentiation formulas, NDF) y de diferencias finitas atrasadas (backward differentiation formulas, BDF)

odeplot (representación para series temporales) odephas2 (representación en plano de fases bidimensional), odephas3 (idem tridimensional), odeprint (imprime el resultado).

1.2 Opciones más usuales de los solvers

La instrucción odeset permite ajustar los parámetros de resolución de las ecuaciones diferenciales implícitas (ode15i) y problemas de valor inicial (ode23, ode45,ode113, ode15s, ode23s, ode23t, ode23tb).

1.2.1 Control de error

En cada paso de la resolución, el solver estima el error local en la i-esima componente de la solución. Este valor debe ser menor o igual que un error admisible indicado mediante el máximo de la tolerancia relativa (RelTol) y la tolerancia obsoluta (AbsTol), esto es:

AbsTol.|e(i)| = max(RelTol*abs(y(i)),AbsTol(i))

Propiedad	*Valor*	*Descripción*
RelTol	Escalar positivo {10^{-3}}	Tolerancia relativa aplicada a todas las componentes del vector solución
AbsTol	Escalar positivo {10^{-6}}	Tolerancia absoluta aplicada a todas las componentes del vector solución
NormControl	On | {off}	Control utilizando la relación superior o una más estricta interna al programa

1.2.2 Control de resultados

La siguiente tabla muestra los resultados que genera el solver seleccionado.

Propiedad	*Valor*	*Descripción*
NonNegative	Vector de enteros {[ ]}	Especifica los componentes de la solución que no pueden ser negativos
Refine	Entero positivo	Incrementa el numero de valores de salida en un factor dado por refine
OutputFcn	Función	Especifica una determinada función de salida.
Stats	On | {off}	Determina si el solver debería Mostar estadísticas sobre el proceso.

1.3 Ejemplo de resolución con MATLAB

Se emplea como ejemplo la ecuación de Van der Pool

$$y'' - \mu\left(1-y^2\right)y' + y = 0 \quad ,t \geq 0 \quad y(0) = 2 \quad y'(0) = 0$$

1.3.1 Problema de valor inicial no rígido (μ=1, 0≤t≤20)

Se debe comenzar reescribiendo el problema como un sistema:

$$y'' - \mu\left(1-y^2\right)y' + y = 0 \xrightarrow[y'=y_2]{y=y_1} \begin{cases} y_1' = y_2 \\ y_2' = \left(1-y_1^2\right)y_2 - y_1 \end{cases}$$

y definiendo la función

```
function dydt = vdp1(t,y)
dydt = [y(2); (1-y(1)^2)*y(2)-y(1)];
```

A continuación se ejecuta el solver

```
[t,y] = ode45(@vdp1,[0 20],[2; 0]);
```

Podemos representar la solución y su derivada mediante

```
plot(t,y(:,1),'-',t,y(:,2),'--')
```

```
title('Solución de la ecuación de van der Pol, \mu = 1');
xlabel('tiempo t');
ylabel('solución y');
legend('y_1','y_2')
```

1.3.2 Problema de valor inicial rígido (μ=1000, 0≤t≤3000)

En este tipo de problemas la solución tiene dos componentes con frecuencias muy diferentes; una tiene cambios en periodos muy inferiores al intervalo total de integración, mientras la otra se pueden considerar del orden del intervalo. Los métodos no diseñados específicamente para problemas rígidos pueden fallar o ser poco eficientes al exigir pasos de tiempo muy pequeños para seguir la rçapida variación de la primera componente.

Se define la función

```
function dydt = vdp1000(t,y)
dydt = [y(2); 1000*(1-y(1)^2)*y(2)-y(1)];
```

A continuación se ejecuta el solver

```
[t,y] = ode15s(@vdp1000,[0 3000],[2; 0]);
```

Podemos representar la solución y su derivada mediante

```
subplot(2,1,1);plot(t,y(:,1),'-')
title('Solución de la ecuación de van der Pol, \mu = 1');
xlabel('tiempo t');ylabel('solución y');legend('y_1')
subplot(2,1,2);plot(t,y(:,2),'-')
title('Solución de la ecuación de van der Pol, \mu = 1');
xlabel('tiempo t');ylabel('solución y');legend('y_2')
```

1.3.3 Evaluación de la solución en puntos dados

Para evaluar la solución en puntos específicos se utiliza la función *deval*, por ejemplo, para evaluar la ecuación de van der Pool no rígida en t=1,2,3,4,5, se escribe

```
sol = ode45(@vdp1,[0 20],[2; 0]);
```

que devuelve la solución en una estructura

```
xint= 1:5
deval(sol,xint)
```

Ejercicios recomendados:

2. Resolver de forma aproximada las siguientes ecuaciones diferenciales con el método más adecuado, sabiendo que la última es rígida. Comparar gráficamente la solución exacta y la real

$$y' + 2y = te^{2t} \quad 0 \le t \le 1 \quad y(0) = 0$$

$$y'' - 2y' + 2y = e^{2t}\sin t \quad 0 \le t \le 1 \quad y(0) = -0.4 \quad y'(0) = -0.6$$

$$y' = 5e^{5t}(y - t)^2 + 1 \quad 0 \le t \le 1 \quad y(0) = -1$$

1.4 Ejemplos de demostración de resolución de ODEs en MATLAB

A continuación se muestran varias instrucciones con ejemplos preprogramados de MATLAB, con las diferentes opciones vistas.

Ejemplo	*Descripción*
amp1dae	Ecuación diferencial algebraica rígida (circuito eléctrico)
ballode	Posición de un cuerpo (pelota elástica)
batonode	ODE con matriz de masa dependiente del tiempo y del estado (equilibrio de un bastón)
brussode	ODE rígida (difusión en una reacción química autocatalítica, brusselator)
burgersode	ODE con matriz de masa fuertemente dependiente del estado (ecuación de Burger resuelta con una técnica de mallado móvil)
fem1ode	ODE rígida con matriz de masa dependiente del tiempo (Método de elementos finitos)
fem2ode	ODE rígida con matriz de masa independiente del tiempo (Método de elementos finitos)
hb1ode	ODE rígida resuelta sobre largos intervalos de tiempo (reacciones química de Robertson)
hb1dae	Ecuación algebraica en diferencias rígida y linealmente implícita (Problema de Robertson abtenido a partir de las leyes de conservación)
ihb1dae	Ecuación algebraica en diferencias rígida y totalmente implícita (Problema de Robertson abtenido a partir de las leyes de conservación)
iburgersode	Sistema de ODEs para resolver la ecuación de Burger
kneeode	Movimiento de una articulación, con restricciones no negativas
orbitode	Posición de tres cuerpos con restricciones
rigidode	Problema no rígido (ecuación de Euler de un cuerpo rígido sin fuerzas externas)
vdpode	Ecuación de Van der Pool parametrizable (rígida para valores elevados)