CAPÍTULO 4

Ecuaciones Diferenciales Ordinarias

En las siguientes sesiones se estudiarán ejemplos de ecuaciones diferenciales ordinarias; sobre todo problemas de una variable pero, si hay tiempo, se introducirán algunos ejemplos de varias variables.

1. Perfil de una etapa ciclista

El concepto más importante de la asignatura de Cálculo (y de todo el Cálculo Infinitesimal) es el de *derivada* $y'(x)$ de una función $y(x)$ en un punto x_0: la *pendiente de la gráfica de* $y(x)$ *en el punto* $(x_0, y(x_0))$. Lo usaremos con mucha frecuencia.

Sean $\mathbf{y}' = (y'_0, y'_1, \ldots, y'_{n-1})$ las distintas pendientes de la carretera en diferentes puntos de una etapa ciclista y sean $\mathbf{x} = (x_0, x_1, \ldots, x_n)$ las coordenadas x (horizontales) de cada uno de dichos puntos (téngase en cuenta que $\mathbf{x}$ no son los puntos kilométricos, sino las coordenadas en el eje OX de cada punto del trayecto). Obsérvese que hay un elemento menos en la lista de pendientes que en la de valores de x.

Si la carrera comienza en una altura y_0, ¿cómo podría calcularse de manera aproximada la altura en la que se está en cada punto x_i, para $i = 1, \ldots, n$?

Este problema tiene diversas soluciones (depende de cómo se plantee la aproximación). Parece que la manera más obvia de resolverlo es suponer que en cada intervalo $[x_i, x_{i+1})$ la pendiente es constante e igual a y'_i. De este modo, la altura en x_1 se aproximaría como $y_1 = y_0 + y'_0(x_1 - x_0)$ (la altura que se sube o baja es la pendiente por el incremento en horizontal). Una manera de comprobarlo es viendo que la ecuación de la recta que pasa por (x_0, y_0) y tiene pendiente y'_0 es

$$Y = y_0 + y'_0(x - x_0).$$

Repitiendo este argumento, se calcularía la altura aproximada en x_2 (que llamamos y_2), en x_3, etc. hasta x_n.

Es importante darse cuenta de que para resolver el problema se requiere la altura inicial y_0: sin ella no se puede ni siquiera intentar aproximar y_1.

Ejemplo 1 Hagamos un ejemplo con solo 5 nodos. Supongamos que la carretera tiene las indicaciones de pendiente: $+4\%$, -2%, $+7\%$, $+12\%$ en las coordenadas horizontales (en km): $0, 2, 5, 7p$ y que la carrera termina en el kilómetro 8. La etapa comienza a 850m. Dibújese un perfil aproximado.

Este ejemplo se puede hacer a mano completamente:

(1) Como el primer tramo tiene 2km de longitud *horizontal* y la pendiente es del 4%, suponiendo que esta es constante en todo el tramo, la carretera subirá $2\text{km} \times 4\% = 80\text{m}$. Como la etapa comienza a 850m, se alcanzará (aproximadamente) una nueva altura de $850\text{m} + 80\text{m} = 930\text{m}$.

(2) El segundo tramo tiene 3km de longitud (horizontal) y su pendiente es -2% al comienzo; suponiendo que es constante, el tramo bajará aproximadamente $3\text{km} \times 2\% = 60\text{m}$. Por tanto, termina a $930\text{m} - 60\text{m} = 870\text{m}$ de altura.

(3) El tercer tramo tiene 2km de longitud y suponemos que la pendiente es constante e igual a 7%. Así que se suben 140m y termina a $870\text{m} + 140\text{m} = 1010\text{m}$.

(4) Finalmente, hay un tramo de 1km (en horizontal) de pendiente que aproximamos como 12%: se suben 120m y la carrera termina a $1010\text{m} + 120\text{m} = 1130\text{m}$

El perfil aproximado se muestra en la Figura 1. —

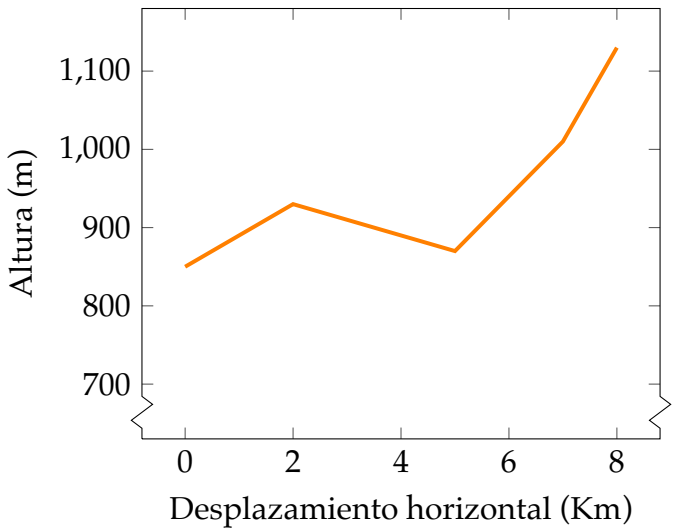

Figura 1. Perfil aproximado de una carrera.

Ejemplo 2 El mismo ejemplo con muchos más datos. Constrúyase un vector **x** de 25 componentes entre 0 y 200 (estas serán las posiciones en horizontal). Constrúyase un vector aleatorio $\mathbf{y}'$ (désele un nombre como yp) con valores entre -15% and 15% (razonables para una carretera). Fíjese un valor y_0 como altura inicial. Calcúlense los valores aproximados sucesivos para cada x_i y dibújese el perfil (aproximado) de la etapa.

Se da una solución a este enunciado en el listado 4.1 y se presenta un perfil aproximado en la Figura 2.

```
% Una 'etapa' de una carrera ciclista con pendientes aleatorias.

% Tengase en cuenta que hay una pendiente menos que coordenadas x
x  = linspace(0, 200, 25);

% Explicacion de la siguiente linea:
%   Tomense 24 valores aleatorios entre 0 y 1
%   Multipliquense por 0.30 para que esten entre 0 y 0.30
%   Restese 0.15 para que queden entre -0.15 y 0.15, como dice el enunciado
yp = rand(1,24) * .30 - .15;

% Altura inicial (elijase una)
y0 = 870;

% Construccion del perfil aproximado
% 1) Se fabrica la lista de "incrementos de altura"
%    (x(i+1) - x(i)) * yp(i)
h  = diff(x).*yp;

% 2) Se fabrica primero una lista de ceros para las alturas
%    y se pone el primer valor
y    = zeros(1, 25);
y(1) = y0;

% 3) Para cada segmento,
%      Anyadase a la lista de alturas el valor anterior mas
%      el incremento correspondiente
for s = 1:length(h)
    y(s+1) = y(s) + h(s);
end
% Seria mucho mas propio y claro escribir:
%  y(2:end) = y(1:end-1) + h;

plot(x,y);
```

Listado 4.1. Perfil aproximado de una etapa ciclista, primera versión.

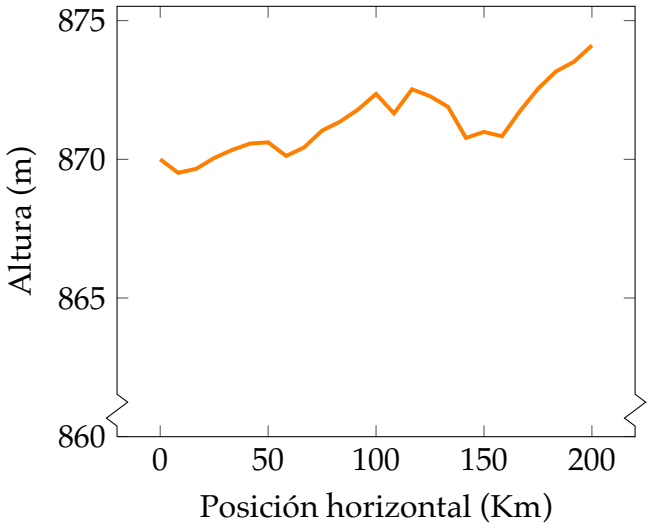

Figura 2. Perfil aproximado de una etapa (más larga).

Por supuesto, este método solo es *una* manera posible de resolver el problema del perfil de manera aproximada; hay más, aunque no muchas más razonables con los datos de que se dispone.

Ejemplo 3 Se puede afrontar el problema de un modo distinto. En el Ejemplo 2, se utiliza la pendiente *al comienzo del intervalo* como si fuera constante en todo él; esto puede ser poco adecuado. De hecho, uno podría pensar "¿por qué se va a usar el valor en el extremo izquierdo y no el del extremo derecho?" Quizás una manera más razonable de enfrentarse al problema sería usar, de algún modo, la información en ambos extremos para cada intervalo. En lugar de tomar y'_{i-1} o y'_i como la pendiente para *todo el intervalo* $[x_{i-1}, x_i]$, uno podría elegir utilizar el valor medio como "pendiente media" en tal intervalo:

$$\tilde{y}'_{i-1} = \frac{y'_{i-1} + y'_i}{2}$$

y calcular cada altura sucesiva y_i como:

$$y_i = y_{i-1} + (x_i - x_{i-1})\tilde{y}'_{i-1}.$$

Ha de tenerse en cuenta (y esto es importante) que para poder hacer este cálculo, es preciso conocer de antemano *la pendiente en el último punto*, así que ahora $\mathbf{y}'$ debe tener tantas componentes como $\mathbf{x}$ para que este método funcione.

Este argumento se utiliza en el listado 4.2.

```
% Etapa ciclista usando valores medios de pendientes

% Comenzamos con las coordenadas x equidistribuidas
x  = linspace(0, 200, 25);

% Creamos una lista de pendientes:
yp = rand(1,25) * .30 - .15;

% Pendiente inicial (a elegir)
y0 = 870;

% Pendientes medias en cada intervalo
yp_means = (yp(1:end-1) + yp(2:end))/2;

% Vector de "incrementos de alturas"
h  = diff(x).*yp_means;

% Vector de alturas "reales": la primera es y0
y    = zeros(1, 25);
y(1) = y0;

% Forma propia de Matlab de hacer la cuenta entera:
for s = 1:length(h)
    y(s+1) = y(s) + h(s);
end

plot(x,y);
```

Listado 4.2. Perfil aproximado de una etapa ciclista, versión del "valor medio".

Ejercicio 1: Usando los códigos de los listados 4.1 y 4.2, compárense las soluciones que se obtienen para el mismo problema: hágase "analíticamente" (es decir, utilizando diferencias absolutas y relativas) y gráficamente. —

Ejercicio 2: Escribir dos archivos de Matlab, uno para cada uno de los métodos de los ejemplos 2 y 3. En cada uno de ellos se ha de definir una función que recibe como argumentos dos vectores de la misma longitud `x` e `yp` y un número real `y0`. La salida será un vector `y` que contiene la lista de alturas de la etapa en cada valor de `x`. Llámese a dichas funciones `profile_euler.m` y `profile_mean.m`.

Úsense varias veces por pares (con los mismos datos las dos) y compárense las gráficas. ¿Cuáles son más suaves? ¿Por qué?

Se da un ejemplo del contenido del archivo `profile_euler.m` en el listado 4.3.

```
% profile_euler(x, yp, y0):
```

```
%
% Dadas listas de coordenadas x, de pendientes yp y una altura inicial y0
% devuelve las alturas en cada tramo de una carretera que tenga pendientes
% yp (para cada valor de x), comenzando en altura y0.
% Se supone que la pendiente es constante e igual a la del extremo izquierdo
% en cada tramo.
function [y] = profile_euler(x, yp, y0)
    y = zeros(size(x));
    y(1) = y0;
    for s = 1:length(x)-1
        y(s+1) = y(s) + yp(s)*(x(s+1) - x(s))
    end
end
```

Listado 4.3. Código de ejemplo para el archivo `profile_euler.m`.

2. Integración Numérica de EDO

La sección anterior era solo una introducción al problema de la integración (resolución) numérica de Ecuaciones Diferenciales Ordinarias (EDO de ahora en adelante). Solo consideraremos, en este curso, ecuaciones de la forma:

$$y' = f(x, y).$$

Y ahora vamos a explicar qué significa esta fórmula.

La derivada de una función $y(x)$ es (repetimos) la *pendiente de la gráfica* $(x, y(x))$ *en cada punto*: si $y(x)$ es derivable y x_0 es un número, entonces $y'(x_0)$ es justamente la *pendiente* (como la pendiente de una *carretera*, es lo mismo) de la gráfica de $y(x)$ en el punto $(x_0, y(x_0))$.

En la Figura 3 se muestra la gráfica de la función $y(x) = \sin(x)$ y la recta tangente a dicha gráfica en el punto $(\frac{2\pi}{6}, \frac{\sqrt{3}}{2})$. Obsérvese que la pendiente de esta recta es 0.5 (i.e. está inclinada $\pi/6$, 30 grados): se puede ver que sube dos unidades en vertical en un desplazamiento horizontal de 4. Este valor 0.5 es justamente lo mismo que el valor de $y'(x) = \cos(x)$ en el punto $x_0 = \frac{2\pi}{6}$. Así incidimos en la idea de que "derivada es lo mismo que pendiente".

Con esta idea clara, la expresión (que es una ecuación):

$$y' = f(x, y)$$

solo puede tener una interpretación: "la función $y(x)$ cumple que su pendiente en cada punto $(x, y(x))$ es justamente $f(x, y(x))$". Dicho brevemente, "la pendiente de y en x es $f(x, y)$".

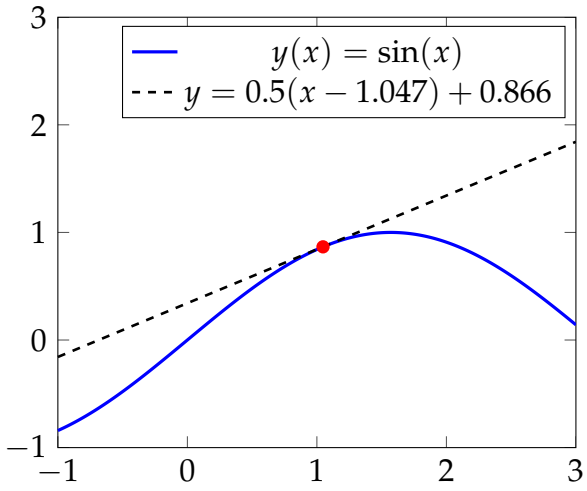

Figura 3. Gráfica de $y(x) = \sin(x)$ y su tangente en $\left(\frac{2\pi}{6}, \frac{\sqrt{3}}{2}\right)$.

En el ejemplo de la carrera ciclista, dar las pendientes de la carretera en cada punto no basta para calcular las alturas correspondientes: hace falta un dato inicial (la altura a la que se comienza). Esto pasa también con una EDO: plantear la ecuación $y' = f(x, y)$ no es bastante para tener una solución *concreta*, hace falta dar una condición inicial más: la "altura incial" de la gráfica de $y(x)$. Así se llega a la noción de *problema de condición inicial*:

Definición 1. Un *problema de condición inicial* es una EDO $y' = f(x, y)$ y un par (x_0, y_0) que se llama *condición inicial*.

Una vez que tal condición inicial $y(x_0)$ se da, la EDO $y' = f(x, y)$ tiene una única solución [1].

Ejemplo 4 El problema de condición inicial

$$y' = y, \;\; y(0) = 1,$$

tiene como solución la función $y(x) = e^x$. Compruébese.

Si en lugar de $y(0) = 1$ se toma $y(0) = K$, entonces la solución es

$$y(x) = Ke^x.$$

¿Qué ocurre si el valor inicial es $y(1) = 0.5$? ¿Se puede calular la solución? —

Por tanto, para enunciar un problema de condición inicial, se necesitan los siguientes datos:

[1] Bajo ciertas condiciones que son "habituales".

(1) La función $f(x, y)$.
(2) El par (x_0, y_0): un valor de la x y el correspondiente valor $y(x)$.

Pero, puesto que vamos a calcular una solución aproximada, se requiere un dato adicional: la lista de valores x_i en los que se desea calcular el valor aproximado de la solución (igual que en el ejemplo de la carrera ciclista). Este dato será habitualmente un vector de "coordenadas x". También es frecuente, en lugar de dar el vector $(x_0, x_1, \ldots, x_n)$, fijar una distancia h constante entre x_i y x_{i+1} y especificar cuántos intervalos hay: de esta manera se tendrá una lista de puntos $(x_0, x_0 + h, x_0 + 2h, \ldots, x_0 + nh)$.

La "solución" (aproximada) del problema de condición incial será la lista de valores aproximados $y(x)$ de una función que cumpla la ecuación. Cómo puede calcularse es el contenido del resto del capítulo.

2.1. Método de Euler. El primer método que a uno se le ocurre para aproximar la solución de un problema de condición inicial es el de Euler, que consiste en aplicar el mismo principio que en el perfil de la carrera ciclista, como en el Ejemplo 2. Si n es la longitud del vector **x**, se puede describir el método de Euler como en el Algoritmo 2.1: tan fácil como hacerlo. Téngase en cuenta que x_0 e y_0 son *parte de los datos iniciales*.

Algoritmo 1 Método de Euler.

for $i = 1 \ldots n$ **do**
  $y_i = y_{i-1} + f(x_i, y_i)(x_i - x_{i-1})$
end for
return $(y_0, y_1, \ldots, y_n)$

La línea interior del bucle **for** es exactamente la misma que para el perfil de la carrera ciclista: la "altura" y_i en la posición horizontal x_i se aproxima como la altura y_{i-1} en el punto anterior x_{i-1} más la pendiente que indica la ecuación, $f(x_{i-1}, y_{i-1})$ multiplicada por el desplazamiento horizontal $(x_i - x_{i-1})$. El algoritmo devuelve la lista de alturas al final.

La implementación del algoritmo 2.1 como una función en un archivo-`m` de Matlab debería ser directa. Incluimos el código en el Ejemplo 5 como ayuda al lector. La función ha sido llamada `euler` y, como consecuencia, el archivo ha de llamarse `euler.m`.

Ejemplo 5 Dados una función $f(x, y)$ (que supondremos que es una *función anónima*), un vector **x** de posición horizontales y un valor inicial y_0, el listado 4.4 sirve como código para implementar el algoritmo de Euler con dichos datos. Recuérdese que, para que funcione, el nombre del archivo ha de ser euler.m. —

```
% Metodo de Euler para integrar EDOs numericamente
% ENTRADA:
% 1) Una funcion anonima f, de dos variables (importante)
% 2) Un vector con las coordenadas x de los puntos en que aproximar la solucion
% 3) un valor inicial y0 correspondiente a x0 (que es x(1))

% SALIDA:
% un vector de valores y(i) que aproximan la solucion en x(i)
function [y] = euler(f, x, y0)
    % se creea la lista y con la misma longitud que x
    y = zeros(size(x));
    % se almacena la condicion inicial en el primer punto
    y(1) = y0;

    % Bucle de Euler
    for s = 2:length(x)
         y(s) = y(s-1) + f(x(s-1), y(s-1)).*(x(s) - x(s-1));
    end
end
```

Listado 4.4. Código de Matlab para el Algoritmo de Euler.

Ejercicio 3: Utilícese la función `euler` que se ha implementado para resolver numéricamente los siguientes problemas de condición inicial. Úsense tantos puntos como se desee (no más de 100 ni menos de 10), cuando no s especifiquen. Dibújense las soluciones que se encuentran.

- Para $x \in [0, 1]$, resolver $y' = 2x$ con $y(0) = 1$. Úsense 10 puntos. Dibújese, en la misma gráfica, la solución real: $y(x) = x^2 + 1$.
- Para $x \in [0, 1]$, resolver $y' = y$ con $y(0) = 1$. Úsense 15 puntos. Dibujar, en la misma gráfica, la solución real: $y(x) = e^x$.
- Para $x \in [0, 1]$, resuélvase $y' = xy$ con $y(0) = 1$. Dibujar, en la misma gráfica, la solución real: $y(x) = e^{\frac{x^2}{2}}$.
- Para $x \in [-1, 1]$, resuélvase $y' = x + y$ con $y(-1) = 0$.
- Para $x \in [-\pi, 0]$, resuélvase $y' = \cos(y)$ con $y(-\pi) = -1$.
- Para $x \in [1, 3]$, resuélvase $y' = y - x$ con $y(1) = -1$.

El método de Euler no es, en general, la *mejor manera* de aproximar soluciones de EDOs. Como se ve en los tres primeros ejemplos del Ejercicio 3, las soluciones que se calculan con este método están siempre debajo de la solución real si esta es convexa (la derivada es creciente), y están siempre encima si esta en cóncava (la derivada es decreciente). Esta falta de precisión puede evitarse de diversas maneras. La que vamos a estudiar en detalle es el *método de Heun*.

2.2. Método de Heun. Este método consiste en aplicar la idea que se explicó en el Ejemplo 3 para "mejorar" la aproximación al perfil de una carrera: usar el valor medio entre las pendientes inicial y final en un tramo.

Sin embargo, hay una diferencia importante con respecto al problema de la carrera ciclista: en esta, la pendiente al final del tramo se sabe con exactitud porque se conoce la carretera (i.e. por donde pasa realmente la carrera) mientras que en una ecuación diferencial, precisamente porque no se conoce la solución, no puede conocerse su pendiente al final de un tramo. Dada la EDO:

$$y' = f(x, y)$$

y el valor inicial (x_0, y_0), la pendiente de la solución en este punto es conocida (es $f(x_0, y_0)$) pero aun no se conoce el valor aproximado de y_1, y por tanto, $f(x_1, y_1)$ no se conoce. Lo que se hace es una "predicción" de cuál sería el valor de y_1 utilizando el método de Euler, que llamamos $\tilde{y}_1$ y, usando esta predicción, calcular el valor medio entre la pendiente en el punto (x_0, y_0) y la pendiente en este punto predicho $(x_1, \tilde{y}_1)$. Con este valor medio se *corrige* la predicción un poco. Más o menos, como se explica:

(1) Póngase uno en (x_0, y_0).
(2) Calcúlese $(x_1, \tilde{y}_1)$ usando el método de Euler como "predicción": para ello se utiliza $k_1 = f(x_0, y_0)$ como pendiente.
(3) Calcúlese $k_2 = f(x_1, \tilde{y}_1)$: la pendiente en el punto predicho por Euler.
(4) En lugar de k_1 ó k_2, tómese como pendiente el valor medio $k = (k_1 + k_2)/2$.
(5) Úsese k como pendiente (x_0, y_0). Es decir, calcúlese:

$$y_1 = y_0 + k(x_1 - x_0).$$

Y este argumento se repite en cada iteración.

Veamos un ejemplo:

Ejemplo 6 Dado el problema de condición inicial (PCI):

$$y' = x - y \;\; y(2) = 0.$$

sea $\mathbf{x} = (2, 2.25, 2.5)$ una secuencia de coordenadas x. Usemos el método de Heun para calcular una aproximación de la solución del PCI en $x = 2.25$ y $x = 2.5$.

Debemos realizar dos pasos del algoritmo de Heun. Haremos el primero en detalle y el segundo lo mostraremos rápidamente.

Primer paso: $x_0 = 2, y_0 = 0$.

- Según el método de Euler, la pendiente sería $k_1 = f(x_0, y_0) = 2$ (este punto (x_0, y_0) es la condición inicial), así que se tendría $\tilde{y}_1 = 0 + 2 \times 0.25 = 0.5$.
- Con estos valores, $k_2 = f(2.25, 0.5) = 1.75$.
- El valor medio entre k_1 y k_2 es $k = 1.875$.
- Por tanto, el valor de y_1 dado por Heun es $y_1 = y_0 + k(x_1 - x_0) = 0 + 1.875 \times 0.25 = 0.46875$.

En fin, $(x_1, y_1) = (2.25, 0.46875)$.

Segundo paso: $x_1 = 2.25, y_1 = 0.46875$.

De estos datos sale $k_1 = f(x_1, y_1) = 1.7812$ y $\tilde{y}_2 = 0.46875 + 1.7812 \times 0.25 = 0.91405$, de donde $k_2 = f(x_2, \tilde{y}_2) = 1.5859$. Así pues, $k = 1.6835$ y el resultado final es $(x_2, y_2) = (2.5, 0.88962)$.

La solución real del problema es $y(x) = x - e^{2-x} - 1$, que vale en $x = 2.5$: $y(2.5) = 0.89347$, el error relativo que se ha cometido es $\frac{|0.89347 - 0.88962|}{0.89347} \simeq 0.004$, que es bastante pequeño (téngase en cuenta que los pasos en la variable x son muy grandes: 0.25 cada uno). —

En el Algoritmo 2.2 se expresa este método con precisión. Se supone que $f(x, y)$ y (x_0, y_0) son datos.

Algoritmo 2 Método de Heun.

for $i = 1 \ldots n$ **do**
  $k_1 = f(x_{i-1}, y_{i-1})$
  $\tilde{y}_i = y_{i-1} + k_1(x_i - x_{i-1})$
  $k_2 = f(x_i, \tilde{y}_i)$
  $k = \frac{k_1 + k_2}{2}$
  $y_i = y_{i-1} + k(x_i - x_{i-1})$
end for
return $(y_0, \ldots, y_n)$

Ejercicio 4: Implementar el método de Heun en un archivo `m`, usando una función que se llame `heun` (recuérdese que el archivo ha de

llamarse `heun.m`). Utilícese para encontrar soluciones aproximadas a los PCI del Ejercicio 3 y compárense estas con las encontradas usando el algoritmo de Euler. ¿Cuáles son mejores? ¿Siempre es mejor un método que otro? —

Ejercicio 5: Impleméntense los métodos de Euler y Heun utilizando una hoja de cálculo cualquiera (Excel, Numbers, LibreOffice…). Explíquese cómo se haría y utilícense para los ejemplos del Ejercicio 3. Úsese la hoja de cálculo para dibujar las soluciones. —

CAPÍTULO 5

Ecuaciones Diferenciales Ordinarias (II)

En este capítulo vamos a desarrollar en algo de detalle unos ejemplos elementales de EDOs de varias variables, que son más gráficos y realistas que los del capítulo anterior.

1. El modelo predador/presa de Lotka-Volterra

El primer modelo diferencial que modela un sistema biológico complejo es el de "Lotka-Volterra": se utiliza para describir la evolución de un entorno en el que dos especies conviven; una de ellas actúa como depredador y otra como presa (el ejemplo más habitual que se da es el de zorros y conejos).

Supongamos que $x(t)$ denota la población de la especie "presa" en el momento t y que $y(t)$ denota la población de la especie "predador". La ecuación diferencial de Lotka-Volterra que describe la evolución conjunta de las dos especies se basa en las siguientes suposiciones (simplistas, claro está):

(1) La población de presas crece en proporción a su tamaño.
(2) Una presa muere solo como consecuencia de ser víctima de un depredador. Esto ocurre con una probabilidad constante.
(3) La población de depredadores muere en proporción a su tamaño.
(4) Los predadores solo se multiplican en proporción a las presas que comen.

Estas cuatro reglas elementales (insistimos, muy simples), se trasladan a ecuaciones como se explica a continuación.

Del punto 1 se deduce que existe una constante $\alpha > 0$ tal que

$$\dot{x}(t) = \alpha x(t) + \dots$$

donde los puntos denotan otra expresión que habrá que descubrir. Si solo se tuviera esta ecuación, habría un crecimiento de presas exponencial (la ecuación $\dot{x}(t) = \alpha x(t)$ tiene como solución $x(t) = Ke^{\alpha t}$, donde K es el valor inicial).

Del punto 3 se deduce que existe un número $\gamma > 0$ tal que

$$\dot{y}(t) = -\gamma y(t) + \dots$$

igual que antes para las presas. Esto hace que, de por sí, los depredadores decrezcan según una ley exponencial negativa.

Es fácil convencerse de que la probabilidad de que un depredador se encuentre con una presa en algún lugar del terreno es proporcional al producto del número de predadores y de presas. Pero como no todo encuentro mutuo termina en una caza, se modela la probabilidad de que un predador coma a una presa como una constante β por dicho producto: $\beta x(t)y(t)$ (con $\beta > 0$). Por otro lado, el hecho de que un depredador se alimente no garantiza que se reproduzca; haremos que esto ocurra con un factor δ. Así, los puntos suspensivos que hemos dejado arriba han de sustituirse por $-\beta x(t)y(t)$ (en la parte de las presas) y por $\delta x(t)y(t)$ (en la parte de los depredadores). Se obtiene el sistema de ecuaciones diferenciales de Lotka-Volterra:

$$\begin{aligned} \dot{x}(t) &= \alpha x(t) - \beta x(t)y(t) \\ \dot{y}(t) &= -\gamma y(t) + \delta x(t)y(t) \end{aligned} \tag{1}$$

Ejemplo 7 Modélese un sistema del tipo Lotka-Volterra con parámetros $\alpha = 0.8$, $\beta = 0.4$, $\gamma = 2$, $\delta = 0.2$ y con poblaciones iniciales $x(0) = 18$, $y(0) = 3$. Utilícese el método de Euler para calcular una aproximación de su evolución con un paso temporal de 0.1, durante 12 unidades de tiempo.

Primero lo haremos a mano y después intentaremos escribir un programa que sirva para el caso general. El listado 5.1 muestra una manera (complicada) de hacer este ejemplo y de dibujar la evolución de ambas poblaciones en el tiempo.

```
% Simulacion de Lotka-Volterra, version 1
% El tiempo va de 0 a 12 en incrementos de 0.1
t=[0:.1:12];
% Se crea una lista de ceros del tamanyo adecuado para ambas variables
x=zeros(size(t));
y=zeros(size(t));
% Valores inciales
x(1)=18;
y(1)=3;

% Estas son las ecuaciones de Lotka-Volterra con los parametros del ejercicio
xp = @(t,x,y) 0.8*x - 0.4*x*y;
yp = @(t,x,y) -2*y + 0.2*x*y;

% El paso de Euler para ambas variables
for k=1:length(t)-1
    x(k+1) = x(k) + xp(t(k),x(k),y(k)) * (t(k+1) - t(k));
    y(k+1) = y(k) + yp(t(k),x(k),y(k)) * (t(k+1) - t(k));
```

```
end

% Dibujo de ambas especies en la misma grafica
plot(t,x)
hold on
plot(t,y,'r')
```

Listado 5.1. Una primera manera de resolver la ecuación de Lotka-Volterra.

Obsérvese (esto es quizás lo más importante de este sistema) que ambas poblaciones tienen un comportamiento creciente y decreciente con un desfase entre ellas.

Con algo de esfuerzo se puede verificar que los puntos críticos de la función $x(t)$ se alcanzan cuando la función $y(t)$ vale γ/δ y que los puntos críticos de $y(t)$ se alcanzan cuando la función $x(t)$ vale α/β (¿cómo se puede comprobar esto? es fácil pero requiere una explicación).

Sin embargo, se sabe (desde el punto de vista teórico) que el sistema de Lotka-Volterra *es periódico*. Las soluciones que se han dibujado no lo son (si uno calculara las soluciones para tiempos mucho más largos, vería cómo las funciones se vuelven cada vez más exageradamente "puntiagudas"). Esta anomalía se debe a la utilización del método de Euler: es demasiado simple, y esto se "paga". —

Debería ser fácil darse cuenta de que el proceso llevado a cabo en el ejemplo anterior es generalizable si uno escribe un programa que ejecute el algoritmo de Euler para EDOs de varias variables.

La expresión

$$\begin{aligned}\dot{x}(t) &= \alpha x(t) - \beta x(t)y(t)\\ \dot{y}(t) &= -\gamma y(t) + \delta x(t)y(t)\end{aligned}$$

se puede reescribir, en forma vectorial, como

$$\begin{pmatrix}\dot{x}(t)\\ \dot{y}(t)\end{pmatrix} = \begin{pmatrix}f_1(t,x,y)\\ f_2(t,x,y)\end{pmatrix}$$

donde $f_1(t,x,y) = \alpha x - \beta xy$ y $f_2(t,x,y) = -\gamma y + \delta xy$. En el caso general, se tendrá un vector de más de dos variables y uno podría escribir el sistema, de manera totalmente genérica como:

$$\begin{pmatrix}\dot{x}_1(t)\\ \vdots\\ \dot{x}_n(t)\end{pmatrix} = \begin{pmatrix}F_1(t,x_1,\ldots,x_n)\\ \vdots\\ F_n(t,x_1,\ldots,x_n)\end{pmatrix}$$

donde $F_1, \ldots, F_n$ son funciones de $n+1$ variables. Así pues, en el caso más general, para describir el problema de condición inicial, hace falta:

- Un *vector columna* de n componentes: las n funciones de $n+1$ variables.
- Los n valores iniciales para $t = 0$: $x_1(0), \ldots, x_n(0)$.

Para programar el método de Euler en este caso, hace falta por tanto definir una función, digamos `eulervector` que tiene por entrada:

- Una función anónima que devuelve un vector columna (la lista de $F_1, \ldots, F_n$ de arriba),
- Una lista de valores de tiempo en que calcular las soluciones aproximadas,
- Un vector columna con los n valores iniciales.

Dicha función debe devolver una lista de vectores columna de la misma longitud que la lista de valores temporales (por tanto, una matriz de n filas —número de variables— y l columnas —valores temporales)

Ejemplo 8 El código del listado 5.2 es una posible implementación de la función `eulervector` descrita arriba.

```
% Metodo de Euler para integrar EDOs, version vectorial.
% ENTRADA:
% 1) una funcion anonima f(t,x)
%    de dos variables:
%    t: numerica
%    x: vector columna
% 2) un vector T de valores temporales en que calcular la solucion
% 3) un vector x0 de valores iniciales de la x para t = T(1)

% SALIDA:
% una matriz de tantas filas como x0 y tantas columnas como T, que
% aproxima la solucion de la EDO dada por f

function [y] = eulervector(f, x, y0)
    % creamos y lleno de ceros con el tamanyo adecuado
    y = zeros(length(y0),length(x));
    % almacenamos las condiciones iniciales en la primera posicion
    y(:,1) = y0;

    % Bucle de Euler (es igual que siempre porque Matlab es vectorial)
    for s = 2:length(x)
         y(:,s) = y(:,s-1) + (x(s) - x(s-1)) * f(x(s-1),y(:,s-1));
    end
```

```
end
```

Listado 5.2. Implementación del método de Euler para funciones vectoriales.

Obsérvese en dicho código como la única diferencia entre nuestra implementación anterior del algoritmo de Euler en el listado 4.4, es la aparición explícita de las filas en el vector y tanto al inicio como al final de la línea:

```
y = zeros(length(y0), length(x));
```

y en el resto de ocasiones, con los puntos suspensivos en la expresión `y(:, ...)`.

Usando esta función, el modelo de Lotka-Volterra del Ejemplo 7 se puede resolver con el código del listado 5.3.

```
% Definase la funcion (con los mismos parametros)
% Observese que, como x es un vector (columna)
% uno puede hacer referencia a cada componente con un solo indice:
f = @(t, x) [0.8*x(1) - 0.4*x(1)*x(2) ; -2*x(2) + 0.2*x(1)*x(2)];

% Inicializamos la lista de tiempos
T = [0:.1:12];
% Condiciones iniciales de poblacion
X0 = [18; 3];
% Resolvemos el problema
X = eulervector(f, T, X0);
% Hacemos la grafica (que Matlab se encarga de colorear)
plot(T, X)
```

Listado 5.3. Lotka-Volterra usando `eulervector`.

Se ve también cómo no hace falta utilizar dos variables diferentes `x` e `y`, sino un simple vector "columna" `x` y que las referencias a cada componente se hacen con los índices correspondientes. Así, al definir `f`, en lugar de usar `x` como anteriormente, se usa `x(1)` y, en lugar de `y`, se usa `x(2)`. También se ve esto en la solución, que se almacena en una sola variables (en el ejemplo, `X`) que (en el ejemplo) tiene dos columnas, una para cada componente del sistema.

En fin, nótese que Matlab dibuja cada fila de una matriz con un color distinto, como listas diferentes. Esto hace que sea innecesario el `hold on`. —

Ejercicio 6: Escríbase una función `heunvector` que implemente el método de Heun para ecuaciones diferenciales vectoriales. Debería ser obvio que la única diferencia con el listado 5.2 ha de consistir en cambiar la línea

```
y(:,s) = y(:,s-1) + (x(s) - x(s-1)) * f(x(s-1),y(:,s-1));
```

por algo un poco más complicado (el paso de Heun de "predecir y corregir"). —

Ejercicio 7: Utilizando la función `heunvector` recién definida, dibújense las soluciones aproximadas de la ecuación de Lotka-Volterra del Ejemplo 7. Dibújese también la solución dada por el método de Euler y compárense. ¿Cuál parece más razonable y por qué? ¿Cuáles son las diferencias en valor absoluto y relativo?

Verifíquese (de alguna manera razonable) que los máximos y mínimos de cada variable (los puntos críticos) corresponden con los cocientes de los parámetros de la otra. ¿Cómo puede hacerse esto?

Debería ser claro que las soluciones calculadas con el método de Heun tienen un aspecto más periódico que las calculadas con el método de Euler: esto refleja el hecho de que el método de Heun es mejor que el de Euler para aproximar soluciones de ecuaciones de Lotka-Volterra. —

2. Modelos de epidemias (SIR)

Ejercicio 8: El modelo epidémico SIR describe la evolución de una enfermedad infecciosa mediante la siguiente ecuación diferencial:

$$\begin{aligned}\dot{S}(t) &= -\alpha S(t)I(t)\\ \dot{I}(t) &= -\beta I(t) + \alpha S(t)I(t)\\ \dot{R}(t) &= \beta I(t)\end{aligned}$$

donde α y β son dos constantes. Las variables S, I y R indican, respectivamente, el número de individuos susceptibles S de contraer la enfermedad, el de infectados I y el de recuperados R (de "recovered", eliminados bien por inmunidad o bien por curación y subsiguiente inmunidad). Úsese la función de Matlab `heunvector` definida en el Ejercicio 6 para dibujar diversos modelos de esta ecuación, tomando valores diferentes de los parámetros α y β y/o valores diferentes de las condiciones iniciales. Un ejemplo interesante se da con $S(0) = 1$, $I(0) = 0.001$, $R(0) = 0$ y $\alpha = 0.1$, $\beta = 0.05$, para un intervalo de tiempo de 0 a 1500 usando 2000 puntos, por ejemplo. Compárese la evolución de este sistema con uno que tenga $\alpha = 0.05$ y $\beta = 0.01$. ¿Se comportan de manera distinta si $\alpha > \beta$ o si $\beta > \alpha$? ¿Cómo cambian?

Cómparense los resultados que se obtienen usando `heunvector` y `eulervector`. ¿Cuál parece más preciso? —

Ejercicio 9: Modifíquese el modelo del Ejercicio 8 para permitir que algunos de los infectados puedan convertirse en susceptibles (i.e. individuos que se curan pero pueden volver a contagiarse), mediante un coeficiente γ. Compárese la evolución de este sistema con la del anterior, utilizando `heunvector` en ambos casos. —

Ejercicio 10: En el modelo del Ejercicio 9, considérese un posible crecimiento de la población susceptible (digamos, por nacimiento), que siga una ley proporcional (es decir, $S(t)$ crece proporcionalmente a la suma de $S(t), I(t)$ y $R(t)$ con un pequeño coeficiente δ). Compárese este modelo con los de los Ejercicios 9 y 8. Explíquense las gráficas que se obtienen y la (notable) diferencia que se ve en $R(t)$ entre los dos ejercicios previos y en este. —

Ejercicio 11: Además del nacimiento, considérese la posibilidad de muerte en el Ejercicio 10: cada uno de los grupos $S(t), R(t)$ e $I(t)$ decrecen con diferentes probabilidades ϵ_1, ϵ_2 y ϵ_3, suponiendo que $\epsilon_1 = \epsilon_3$ y que $\epsilon_2 > \epsilon_1$. ¿Cambia mucho el modelo? Compárese con los anteriores. —

3. Sistemas Físicos

Ejemplo 9 Se puede simular un péndulo simple sin rozamiento utilizando la siguiente ecuación (en coordenadas polares) con centro el extremo del hilo:

$$l\ddot{\theta} = -g\sin(\theta)$$

donde θ es el ángulo con respecto a la vertical y l es la longitud (constante) del péndulo. Como no hay rozamiento, la masa del péndulo es irrelevante. Usaremos `heunvector` para calcular una aproximación a la solución de esta ecuación, usando por ejemplo $l = 1$ y $g = 9.81$ y dos condiciones iniciales distintas: $\theta(0) = \pi/4$ y $\theta(0) = \pi/3$, ambas con velocidad inicial 0. Dejaremos que el tiempo vaya de 0 a 50 en pasos de 0.01. El código del Listado 5.4 muestra cómo hacerlo.

```
% Simulacion de un pendulo, para dos condiciones iniciales diferentes.
% La masa es irrelevante en este problema.

% La ecuacion del pendulo es theta'' = -sen(theta)
% que se escribe con DOS variables:
% theta=X(1)
```

```
% theta'=X(2)
P = @(t, X) [X(2) ; -sin(X(1))];

% Tiempo
T=[0:.01:50];
% Condicion inicial pi/4
X0=[pi/4; 0];

% Resolvemos con Heun
Y = heunvector(P, T, X0);
% Dibujamos (el angulo y la velocidad, las dos cosas)
plot(T, Y)
hold on

% Condicion inicial pi/6
X1=[pi/6;0];

% Resolvemos y dibujamos como antes
Y2 = heunvector(P, T, X1);
plot(T, Y2)
```

Listado 5.4. Simulación de un péndulo utilizando el método de Heun.

Ejercicio 12: Calcúlense soluciones aproximadas a los mismos sistemas que en el Ejemplo 9 pero utilizando `eulervector`; compárense con las obtenidas en dicho ejemplo. ¿Cuáles parecen mejores y por qué?

Ejercicio 13: Considérese el péndulo del Ejemplo 9 pero con rozamiento. Supóngase que la fuerza de rozamiento es proporcional a la velocidad angular (y, obviamente, de signo contrario), con constante de proporcionalidad 0.1. La ecuación debe escribirse ahora teniendo en cuenta la masa del sistema (póngase $m = 0.5$, por ejemplo). Calcúlese la solución aproximada (utilizando `heunvector`) y compárese con la del ejercicio 9. El efecto que se observa se denomina "amortiguamiento exponencial". Téngase en cuenta que (si se hace un dibujo) la ecuacióon del movimiento queda, en las polares del Ejemplo 9:

$$ml\ddot{\theta} = -g\,\mathrm{sen}(\theta) - k\dot{\theta}.$$

Ejercicio 14: El movimiento armónico simple viene descrito por la EDO de segundo orden

$$\ddot{x} = -\frac{k}{m}x$$

para cierta constante de elasticidad k y una masa m del cuerpo en movimiento. Se dice que está *amortiguado* si hay una fuerza de amortiguamiento que va contra el movimiento, que da lugar a la ecuación

$$\ddot{x} = -\frac{k}{m}x - r\dot{x}$$

para cierta constante $r > 0$. Estúdiese el comportamiento de un oscilador armónico con y sin rozamiento utilizando tanto *eulervector* como *heunvector* y compárense los resultados. ¿Qué ocurre si $r < 0$?

Ejercicio 15: La ecuación balística describe el movimiento de un cuerpo en libertad bajo la acción de la gravedad. Si $(x(t), y(t))$ es el vector de posición, la ecuación es:

$$\ddot{x} = 0$$
$$\ddot{y} = -g.$$

Dadas una posición inicial $(0, 0)$ y una velocidad inicial $(1, 1)$, dibújese la gráfica $(x(t), y(t))$ de posiciones del cuerpo en movimiento para t de 0 a 20 en pasos de 0.01. Úsese `heunvector`.

¿Cuál es la ecuación diferencial si hay fricción y es proporcional a la velocidad (obviamente, en sentido contrario)? Plantéese y dibújese la trayectoria correspondiente para las mismas condiciones iniciales de antes.

Finalmente, hágase el mismo cálculo con el rozamiento proporcional al *cuadrado* de la velocidad.