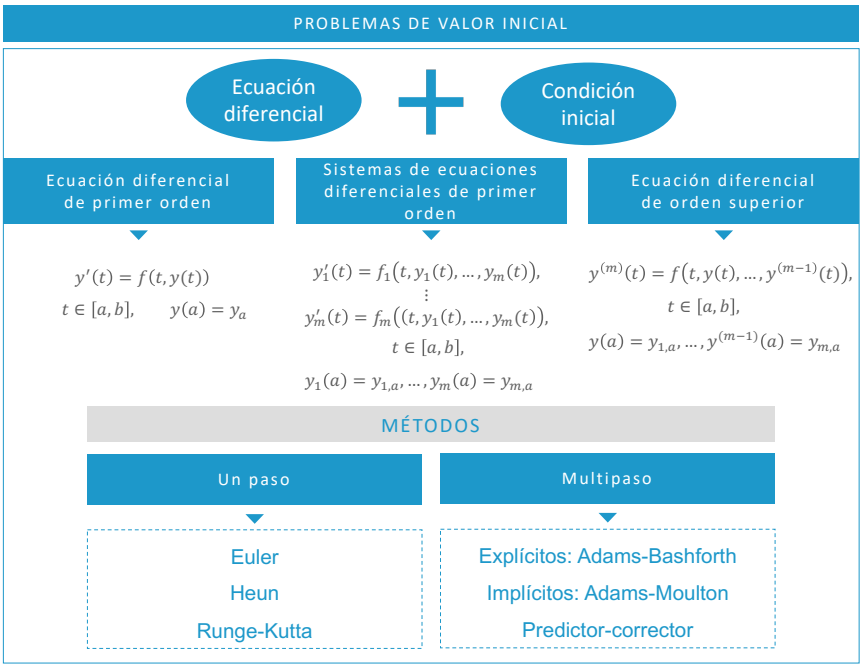
Métodos Numéricos Aplicados I

Problemas de valor inicial II

Índice

Esquema


PROBLEMAS DE VALOR INICIAL
Ecuación diferencial
Condición inicial
Ecuación diferencial de primer orden
Sistemas de ecuaciones diferenciales de primer orden
Ecuación diferencial de orden superior
$y'(t) = f(t, y(t))$
$t \in [a, b], \quad y(a) = y_a$
$y_1'(t) = f_1(t, y_1(t), \dots, y_m(t)),$
$y_m'(t) = f_m((t, y_1(t), \dots, y_m(t)),$
$t \in [a, b],$
$y_1(a) = y_{1,a}, \dots, y_m(a) = y_{m,a}$
$y^{(m)}(t) = f(t, y(t), \dots, y^{(m-1)}(t)),$
$t \in [a, b],$
$y(a) = y_{1,a}, \dots, y^{(m-1)}(a) = y_{m,a}$
MÉTODOS
Un paso
Multipaso
Euler
Heun
Runge-Kutta
Explícitos: Adams-Bashforth
Implícitos: Adams-Moulton
Predictor-corrector

Ideas clave

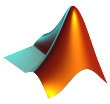
7.1 Introducción y objetivos

En el tema anterior estudiamos distintos métodos numéricos de un paso para resolver problemas de valor inicial. Estos métodos se denominan así porque la aproximación en el punto t_{k+1} solo implica conocer la información del punto anterior t_k. Asimismo, aunque estos métodos generalmente utilizan información de las evaluaciones funcionales entre los puntos t_k y t_{k+1}, no almacenan dicha información de forma directa para ser utilizada en futuras aproximaciones. Por tanto, la solución en un subintervalo está basada exclusivamente en la información disponible en dicho subintervalo.

Dado que para obtener la aproximación de la solución en el nodo t_{k+1} disponemos de la solución aproximada en cada uno de los nodos $t_0, t_1, \ldots, t_k$, parece razonable desarrollar métodos que, en lugar de utilizar solo la información del subintervalo, utilicen de forma más precisa toda la información que se dispone hasta este nodo. Los métodos que utilizan la aproximación en más de un nodo previo para determinar la solución aproximada en el siguiente nodo se denominan métodos multipaso. A lo largo de este tema vamos a definir de forma precisa algunos de estos métodos, así como los tipos de métodos multipaso existentes.

Tras distinguir entre métodos explícitos y métodos implícitos, presentaremos los esquemas multipaso fundamentales y los esquemas predictor-corrector como combinación de éstos. Por último, presentaremos un tipo de problemas de valor inicial, denominados problemas rígidos, para los cuales determinados métodos funcionan mejor que otros debido al comportamiento de la función que define el problema. Por tanto, los objetivos que trataremos de alcanzar son:

- Comprender la diferencia entre método numérico explícito o implícito.
- Estudiar los métodos explícitos e implícitos fundamentales.
- Estudiar el diseño de métodos predictor-corrector.
- Conocer las ecuaciones rígidas y los problemas de resolución numérica relacionados con estas ecuaciones.
- Analizar y comparar la estabilidad en problemas rígidos de los métodos implícitos frente a los explícitos.



Algunas funciones Matlab utilizadas en este tema

- `ode23`, `ode45`: resuelven EDO's y sistemas de EDO's
- `ode15s`: resuelven EDO's y sistemas de EDO's en problemas rígidos

7.2 Métodos numéricos explícitos para resolver PVIs

Consideremos la expresión general de un PVI definido por medio de una ecuación diferencial de primer orden

$$y'(t) = f(t, y(t)), \qquad t \in [a, b], \qquad y(a) = y_a.$$

En el tema anterior vimos que, como aproximación a la solución analítica del problema, calculamos de forma numérica la solución en los nodos equiespaciados t_k en $[a, b]$, $k = 0, 1, \ldots, N$, a una distancia o paso h. Entonces, integrando directamente la ecuación diferencial en cada subintervalo $[t_k, t_{k+1}]$ obtenemos

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(\tau, y(\tau))d\tau. \tag{1}$$

Para aproximar la integral obtenida, sustituiremos el integrando por un polinomio interpolador. Dependiendo de los puntos utilizados para definir el polinomio de interpolación, tendremos métodos explícitos o métodos implícitos. Dedicaremos esta sección al diseño de métodos explícitos, mientras que en la siguiente sección estudiaremos algunos métodos implícitos.

Para diseñar los métodos explícitos, aproximaremos la función por medio de un polinomio de interpolación que pasa por los puntos

$$\{(t_{k-n}, f(t_{k-n}, y_{k-n})), \ldots, (t_{k-1}, f(t_{k-1}, y_{k-1})), (t_k, f(t_k, y_k))\}.$$

Dependiendo de los puntos que consideremos para definir el polinomio, obtendremos un método numérico explícito distinto.

Método de Adams-Bashforth de dos pasos

Partiendo de los puntos $(t_{k-1}, f(t_{k-1}, y_{k-1}))$ y $(t_k, f(t_k, y_k))$, el único polinomio de interpolación que pasa por estos puntos es

$$\begin{aligned} p(\tau) &= f(t_k, y_k) + \frac{f(t_k, y_k) - f(t_{k-1}, y_{k-1})}{t_k - t_{k-1}}(\tau - t_k) \\ &= \frac{\tau - t_{k-1}}{h} f(t_k, y_k) + \frac{t_k - \tau}{h} f(t_{k-1}, y_{k-1}). \end{aligned}$$

Aproximando la función por el polinomio de interpolación, obtenemos en (1):

$$\begin{aligned} y_{k+1} &= y_k + \int_{t_k}^{t_{k+1}} p(\tau) d\tau \\ &= y_k + \int_{t_k}^{t_{k+1}} \left(\frac{\tau - t_{k-1}}{h} f(t_k, y_k) + \frac{t_k - \tau}{h} f(t_{k-1}, y_{k-1}) \right) d\tau. \end{aligned}$$

Calculando la integral polinómica y desarrollando la expresión:

$$\begin{aligned}
y_{k+1} &= y_k + \frac{1}{h}\int_{t_k}^{t_{k+1}} \left[\tau\left(f(t_k,y_k) - f(t_{k-1},y_{k-1})\right) + t_k f(t_{k-1},y_{k-1}) - t_{k-1}f(t_k,y_k)\right] d\tau \\
&= y_k + \frac{1}{h}\left[\left(\frac{t_{k+1}^2 - t_k^2}{2}\right)\left(f(t_k,y_k) - f(t_{k-1},y_{k-1})\right)\right. \\
&\quad \left. +(t_{k+1}-t_k)(t_k f(t_{k-1},y_{k-1}) - t_{k-1}f(t_k,y_k))\right] \\
&= y_k + \frac{1}{h}\left[\frac{h(2t_k+h)}{2}(f(t_k,y_k) - f(t_{k-1},y_{k-1})) + h(t_k f(t_{k-1},y_{k-1}) - t_{k-1}f(t_k,y_k))\right] \\
&= y_k + \left(\frac{2t_k+h}{2} - t_{k-1}\right) f(t_k,y_k) + \left(t_k - \frac{2t_k+h}{2}\right) f(t_{k-1},y_{k-1}) \\
&= y_k + \frac{3h}{2}f(t_k,y_k) - \frac{h}{2}f(t_{k-1},y_{k-1}).
\end{aligned}$$

Con este desarrollo, obtenemos el método Adams-Bashforth de dos pasos, que denotaremos AB2, con expresión:

$$y_{k+1} = y_k + \frac{h}{2}\left(3f(t_k,y_k) - f(t_{k-1},y_{k-1})\right).$$

Se trata de un método explícito de segundo orden con error global $\mathcal{O}(h^2)$.

Recordemos que el objetivo de utilizar el método AB2 es aproximar la solución analítica de un PVI por medio de la solución en cada nodo del intervalo $[a,b]$. Por tanto, pretendemos calcular las soluciones $y_0, y_1, \ldots, y_N$, donde y_0 es un valor conocido dado por la condición inicial del PVI. Sin embargo, para calcular la solución en el segundo nodo, sustituimos $k=0$ en el método de AB2 y obtenemos

$$y_1 = y_0 + \frac{h}{2}\left(3f(t_0,y_0) - f(t_{-1},y_{-1})\right).$$

Como la componente (t_{-1}, y_{-1}) no existe, calcularemos y_1 utilizando otro método numérico. Es razonable utilizar un método del mismo orden que AB2, por lo que utilizaremos el método de Heun para calcular y_1, y a partir de este punto calcularemos la solución en los demás puntos con AB2.

Mostramos a continuación el código de la función de Matlab que implemeta el método de Adams-Bashforth de dos pasos.

AB2.m

```
function [t,y] = AB2(f,a,b,N,ya)
% Código para resolver un PVI con el metodo de ...
   Adams-Bashfort de dos pasos
h=(b-a)/N;
t=a:h:b;
t=t(:);
y=zeros(N+1,1);
y(1)=ya;
% Primer paso con el metodo de Heun
k1 = h*feval(f,t(1),y(1));
k2 = h*feval(f,t(2),y(1)+k1);
y(2) = y(1)+(k1+k2)/2;
% Siguientes pasos con el metodo AB2
for k=2:N
    k1 = feval(f,t(k),y(k));
    k2 = feval(f,t(k-1),y(k-1));
    y(k+1) = y(k)+h/2*(3*k1-k2);
end
end
```

Ejemplo 1.

Utiliza el método de Adams-Bashforth de dos pasos para calcular la solución numérica del PVI obtenido con el modelo poblacional de Verhulst

$$y'(t) = (k - py(t))y(t), \qquad t \in [0, 2], \qquad y(0) = 10,$$

y una estimación del orden del método sabiendo que la solución analítica es

$$y(t) = \frac{10k}{10p + (k - 10p)e^{-kt}}.$$

Consideremos $k = 3$ y $p = 0.1$, y $N = \{2, 4, 8, 16, 32, 64\}$ subintervalos en $[0, 2]$.

En primer lugar, implementamos la función que define la ecuación diferencial del modelo de Verhulst:

```
function dy = VerhulstPVI(t,y)
    k = 3;
    p = 0.1;
    dy = (k-p*y).*y;
end
```

Apliquemos el método AB2 para los diferentes subintervalos así como la solución analítica (para $N = 2$ y $N = 4$):

```
>> [t2,y2]=AB2('VerhulstPVI',0,2,2,10);
>> sol2=3*10./(0.1*10+(3-0.1*10).*exp(-3*t2));

>> [t4,y4]=AB2('VerhulstPVI',0,2,4,10);
>> sol4=3*10./(0.1*10+(3-0.1*10).*exp(-3*t4));
```

Calculamos el error máximo obtenido en cada uno de los subintervalos:

```
>> E2=max(abs(sol2-y2));
>> E4=max(abs(sol4-y4));
```

Y la sucesión de los cocientes entre los errores máximos:

```
>> E=[E2 E4 E8 E16 E32 E64];
>> orden=log2(E(1:end-1)./E(2:end));
```

N	E_N	$\log_2\left(E_{N/2}/E_N\right)$
2	10.1480	–
4	4.5230	1.1658
8	0.6324	2.8384
16	0.1938	1.7064
32	0.0543	1.8365
64	0.0144	1.9178

Tabla 1: Estimación del orden del método AB2

En la Tabla 1 observamos cómo el orden del método tiende a 2 si aumentamos el número de puntos de la discretización.

Métodos de Adams-Bashforth de más de dos pasos

Siguiendo un procedimiento similar al método de Adams-Bashforth de dos pasos, se pueden obtener métodos de mayor orden considerando polinomios interpoladores que pasen por más de dos puntos. Por ejemplo, podemos seguir los pasos descritos anteriormente para aproximar el integrando por un polinomio interpolador que pasa por los puntos

$$\{(t_{k-2}, f(t_{k-2}, y_{k-2})), (t_{k-1}, f(t_{k-1}, y_{k-1})), (t_k, f(t_k, y_k))\},$$

en cuyo caso obtendríamos el método de Adams-Bashforth de tres pasos:

$$y_{k+1} = y_k + \frac{h}{12}\left(23f(t_k, y_k) - 16f(t_{k-1}, y_{k-1}) + 5f(t_{k-2}, y_{k-2})\right),$$

que denotaremos método AB3, y cuyo error global es $\mathcal{O}(h^3)$. Para obtener los valores de $k = 0$ y $k = 1$, podemos utilizar el método de Runge-Kutta.

Análogamente, considerando el polinomio de interpolación que pasa por los siguientes cuatro puntos

$$\{(t_{k-3}, f(t_{k-3}, y_{k-3})), (t_{k-2}, f(t_{k-2}, y_{k-2})), (t_{k-1}, f(t_{k-1}, y_{k-1})), (t_k, f(t_k, y_k))\},$$

obtendríamos el método de Adams-Bashforth de cuatro pasos, denotado AB4, cuya expresión es

$$y_{k+1} = y_k + \frac{h}{24}\left(55f(t_k, y_k) - 59f(t_{k-1}, y_{k-1}) + 37f(t_{k-2}, y_{k-2}) - 9f(t_{k-3}, y_{k-3})\right).$$

El error global de AB4 es $\mathcal{O}(h^4)$. Sin embargo, para poder iniciar el proceso iterativo que define el método, necesitamos calcular previamente y_1, y_2 e y_3. Utilizaremos el método de Runge-Kutta para obtener el valor de la solución en los primeros puntos.

Mostramos a continuación el algoritmo para implementar el método AB4, donde se observa el uso del método de Runge-Kutta en los puntos iniciales:

- Entrada: f, a, b, N, y_a
- Proceso:
  - Obtención de la variable independiente discretizada t
  - Inicialización del vector solución y en a
  - Para k desde 0 hasta 2 aplicar el método de Runge-Kutta:
    - Cálculo de k_1, k_2, k_3 y k_4:

$$k_1 = f(t_k, y_k), \qquad k_2 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}k_1\right),$$
$$k_3 = \left(t_k + \frac{h}{2}, y_k + \frac{h}{2}k_2\right), \qquad k_4 = (t_{k+1}, y_k + hk_3).$$

    - $y_{k+1} = y_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

- Fin para k
- Para k desde 4 hasta N:

$$y_{k+1} = y_k + \frac{h}{24}\left(55f(t_k, y_k) - 59f(t_{k-1}, y_{k-1}) + 37f(t_{k-2}, y_{k-2}) - 9f(t_{k-3}, y_{k-3})\right)$$

- Fin para k

► Salida: t, y

Ejemplo 2.

Obtén la solución del PVI dado por un caso particular del modelo de Verhulst

$$y'(t) = (3 - 0.1y(t))y(t), \quad t \in [0, 2], \quad y(0) = 10.$$

Utiliza el método de Adams-Bashforth de 4 pasos y $N = \{8, 16, 32, 64, 128\}$ subintervalos. Obtén una estimación del orden numérico sin conocer la solución analítica.

Siendo `AB4.m` la función de Matlab que implementa este método, para los diferentes subintervalos ejecutamos el método en Matlab como (para los dos primeros valores de N):

```
>> [t8,y8]=AB4('VerhulstPVI',0,2,8,10);
>> [t16,y16]=AB4('VerhulstPVI',0,2,16,10);
```

Aproximaremos numéricamente el orden del método duplicando el número de subintervalos como

$$\log_2\left(\lim_{N\to\infty} \frac{\epsilon_{N/2}}{\epsilon_N}\right)$$

donde $\epsilon_N = \left\| y_k^{(N)} - y_{1+2k}^{(2N)} \right\|$, $k = 0, 1, \ldots, N$. Por tanto, ejecutamos de forma sucesiva las instrucciones:

```
>> eps16=norm(y8-y16(1:2:end));
>> eps32=norm(y16-y32(1:2:end));
```

Y por último, siguiendo los mismos pasos para todos los valores de N:

```
>> epsilon=[eps16 eps32 eps64 eps128];
>> orden=log2(epsilon(1:end-1)./epsilon(2:end));
```

En la Tabla 2 vemos cómo la columna del orden va tendiendo a valores ligeramente inferiores a 4, por lo que el método no alcanza el orden 4 para este problema.

N	ϵ_N	$\log_2\left(\epsilon_{N/2}/\epsilon_N\right)$
8	–	–
16	1.2721	–
32	0.0377	5.0768
64	0.0051	2.8849
128	0.0005	3.2667

Tabla 2: Estimación del orden del método AB4

Al igual que en el tema anterior, podemos adaptar de forma directa el código de los métodos explícitos para la resolución de PVI obtenidos a partir de sistemas de ecuaciones diferenciales. Recordemos que debemos tener en cuenta las dimensiones de vectores y matrices para definir y evaluar las funciones del sistema.

7.3 Métodos numéricos implícitos para resolver PVIs

Si bien los métodos explícitos para aproximar el integrando en (1) utilizan puntos desde y_{k-n} hasta y_k, en los métodos implícitos también se utiliza y_{k+1}. Por tanto, para definir

el polinomio de interpolación utilizaremos puntos que no conocemos todavía .

Los métodos implícitos son más complejos que los métodos explícitos. Sin embargo, gozan de una mayor estabilidad. Los puntos sobre los que se obtiene el polinomio interpolador en los métodos implícitos son, en general, los siguientes:

$$\{(t_{k-n}, f(t_{k-n}, y_{k-n})), \ldots, (t_k, f(t_k, y_k)), (t_{k+1}, f(t_{k+1}, y_{k+1}))\}.$$

A continuación tomaremos distinos puntos de este conjunto para definir polinomios de interpolación y, a partir de éstos, diferentes métodos implícitos.

Método de Adams-Moulton de un paso

Partiendo de los puntos $(t_k, f(t_k, y_k))$ y $(t_{k+1}, f(t_{k+1}, y_{k+1}))$, el polinomio interpolador que pasa por estos dos puntos es

$$\begin{aligned} p(\tau) &= f(t_{k+1}, y_{k+1}) + \frac{f(t_{k+1}, y_{k+1}) - f(t_k, y_k)}{t_{k+1} - t_k}(\tau - t_{k+1}) \\ &= \frac{\tau - t_k}{h} f(t_{k+1}, y_{k+1}) + \frac{t_{k+1} - \tau}{h} f(t_k, y_k). \end{aligned}$$

Utilizando el polinomio de interpolación como aproximación al integrando y realizando un desarrollo análogo al presentado en la sección anterior con el método AB2, se obtiene

$$\begin{aligned} y(t_{k+1}) &= y(t_k) + \int_{t_k}^{t_{k+1}} f(\tau, y(\tau)) d\tau \\ &\approx y_k + \int_{t_k}^{t_{k+1}} p(\tau) d\tau = y_k + \frac{h}{2} f(t_{k+1}, y_{k+1}) + \frac{h}{2} f(t_k, y_k). \end{aligned}$$

De este modo, el método de Adams-Moulton de un paso, que denotaremos como AM2, tiene la expresión:

$$y_{k+1} = y_k + \frac{h}{2}\left(f(t_{k+1}, y_{k+1}) + f(t_k, y_k)\right)$$

y presenta un error global de $\mathcal{O}(h^2)$.

En la expresión de AM2 podemos observar que, por tratarse de un método implícito, para calcular el valor de la solución en el nodo t_{k+1} es necesario el uso de este valor. Como consecuencia, para obtener y_{k+1} debemos resolver para cada k la siguiente ecuación no lineal:

$$g(y_{k+1}) = y_{k+1} - y_k - \frac{h}{2}\left(f(t_{k+1}, y_{k+1}) + f(t_k, y_k)\right) = 0. \tag{2}$$

Como método de resolución de la ecuación no lineal (2) utilizaremos el método de Newton. Aunque describiremos en los temas siguientes con más detalle las propiedades fundamentales de este método iterativo, debemos saber que el método de Newton se utiliza fundamentalmente para aproximar raíces de funciones no lineales. Partiendo de una primera aproximación x_0 de la raíz de una ecuación no lineal $g(x) = 0$, el método de Newton, cuya expresión iterativa esta dada por

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}, \qquad n = 0, 1, 2, \ldots,$$

genera una secuencia $\{x_n\}_{n\in\mathbb{N}}$ de estimaciones cada vez más próximas a la solución de la ecuación no lineal si se cumplen determinadas condiciones de convergencia.

Por tanto, para resolver la ecuación (2) es necesario calcular la derivada de la función, de modo que:

$$g'(y_{k+1}) = \frac{dg(y_{k+1})}{y_{k+1}} = 1 - \frac{h}{2}\frac{df(t_{k+1}, y_{k+1})}{dy_{k+1}}.$$

Este hecho implica que cuando implementemos en Matlab la función $y'(t) = f(t, y(t))$ que define el PVI, tendremos dos parámetros de salida: la función $f(t, y(t))$ y su derivada $\frac{df(t,y(t))}{dy(t)}$.

La implementación en Matlab del método de Adams-Moulton de un paso se muestra en el siguiente código.

AM2.m

```
function [t,y] = AM2(f,a,b,N,ya)
```

```
% Código para resolver un PVI con el metodo de ...
   Adams-Moulton de un paso
h=(b-a)/N;
t=a:h:b;
t=t(:);
y=zeros(N+1,1);
y(1)=ya;
maxiter=10;
tol=1e-6;
% Primer paso con el metodo de Heun
k1=h*feval(f,t(1),y(1));
k2=h*feval(f,t(2),y(1)+k1);
y(2)=y(1)+k1/2+k2/2;
% Siguientes pasos con el metodo AM2
for k=2:N
    ff=feval(f,t(k),y(k));
    % Metodo de Newton
    iter=1; dif=tol+1;
    x0=y(k);
    while and(iter<maxiter,dif>tol)
        [fx0,dfx0]=feval(f,t(k+1),x0);
        g=x0-y(k)-h/2*(fx0+ff);
        dg=1-h/2*dfx0;
        x1=x0-g/dg;
        dif=abs(x1-x0);
        iter=iter+1;
        x0=x1;
    end
    y(k+1)=y(k)+h/2*(feval(f,t(k+1),x0)+ff);
end
end
```

Notemos que en la implementación de AM2 se utiliza el método de Newton el cual, por tratarse de un algoritmo iterativo, requiere de unas condiciones para detener el proceso. En `AM2.m` se han utilizado como criterios de parada las variables `maxiter` y `tol`, que indican el máximo de iteraciones y la mínima diferencia entre dos puntos consecutivos, respectivamente, para detener el proceso de aproximación a la raíz de la ecuación no lineal. En este código, se han fijado los valores `maxiter=10` y `tol=1e-6`.

Ejemplo 3.

Otén y representa la solución del PVI

$$y'(t) = (3 - 0.1y(t))y(t), \qquad t \in [0, 2], \qquad y(0) = 10,$$

utilizando el método de Adams-Moulton de un paso y tomando $h = 0.2$.

La función que define el PVI se mostró en el Ejemplo 1. Sin embargo, para utilizar este método implícito se requiere resolver una ecuación no lineal con el método de Newton y por tanto necesitamos conocer la función y su derivada:

$$f(t, y(t)) = (3 - 0.1y(t))y(t), \qquad \frac{df(t, y(t))}{dy(t)} = 3 - 0.2y(t).$$

Redefinimos la función `VerhulstPVI.m` del Ejemplo 1 para que también muestre como dato de salida la derivada de la función:

```
function [fun,dfun] = Verhulst2PVI(t,y)
    k = 3; p = 0.1;
    fun = (k-p*y).*y;
    dfun = k-2*p*y;
end
```

Aplicamos el método AM2 y representamos la solución ejecutando en Matlab:

```
>> [t,y] = AM2('Verhulst2PVI',0,2,9,10);
>> plot(t,y,'o-')
```

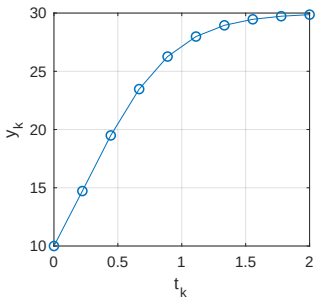


Figura 1: Solución del PVI con AM2 y 10 puntos

Método de Adams-Moulton de más de un paso

Podemos obtener expresiones del método de Adams-Moulton de mayor precisión utilizando más puntos en el polinomio de interpolación de forma similar a como hicimos en la sección anterior. Por ejemplo, utilizando tres puntos en el polinomio de interpolación, tendríamos el método de Adams-Moulton de dos pasos

$$y_{k+1} = y_k + \frac{h}{12}\left(-f(t_{k-1}, y_{k-1}) + 8f(t_k, y_k) + 5f(t_{k+1}, y_{k+1})\right),$$

que denotamos AM3 y tiene orden 3. En cambio, si definimos el polinomio interpolador utilizando cuatro puntos, obtenemos el método de Adams-Moulton de 3 pasos, denotado AM4, con expresión

$$y_{k+1} = y_k + \frac{h}{24}\left(f(t_{k-2}, y_{k-2}) - 5f(t_{k-1}, y_{k-1}) + 19f(t_k, y_k) + 9f(t_{k+1}, y_{k+1})\right)$$

y con un error global $\mathcal{O}(h^4)$.

En general, un método implícito de Adams-Moulton de n pasos, tendrá orden $n+1$. Asimismo, será necesario utilizar otros métodos para inicializar la solución y se deberá resolver una ecuación no lineal para aplicar el método y calcular y_{k+1}.

Concretamente, en el método AM3 utilizaremos un método distinto para inicializar y_1, y con AM4, deberemos inicializar y_1 e y_2. Podemos utilizar el método de Runge-Kutta para calcular la solución en los primeros puntos. Por tanto, el algoritmo para implementar el método de AM4 es el siguiente:

- Entrada: f, a, b, N, y_a
- Proceso:
  - Discretización de de la variable independiente: $t_0, t_1, \ldots, t_N$
  - Inicialización del vector solución y en a: y_0
  - Aplicar el método de Runge-Kutta para calcular y_1 e y_2
  - Para k desde 2 hasta $N-1$, encontrar cada nuevo valor y_{k+1} con AM4 resolviendo la ecuación no lineal:

$$g(y_{k+1}) = y_{k+1} - y_k - \frac{h}{24}(f(t_{k-2}, y_{k-2}) - 5f(t_{k-1}, y_{k-1}) + 19f(t_k, y_k) + 9f(t_{k+1}, y_{k+1})) = 0,$$

    utilizando el método de Newton con $x_0 = y_k$ como estimación inicial:

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}, \quad n = 0, 1, 2, \ldots$$

- Salida: t, y

Ejemplo 4.

Obtén la solución del PVI

$$y'(t) = (3 - 0.1y(t))y(t), \qquad t \in [0, 2], \qquad y(0) = 10,$$

con el método de Adams-Moulton de tres pasos tomando $N = \{8, 16, 32, 64, 128\}$. Calcula el error máximo cometido para cada número de subintervalos y una estimación del orden de AM4 sabiendo que la solución exacta es:

$$y(t) = \frac{30}{1 + 2e^{-3t}}.$$

Llamamos `AM4.m` a la función de Matlab que programa el método AM4. Ejecutaríamos para cada N las siguientes instrucciones (mostramos $N = 8$):

```
>> [t8,y8] = AM4('Verhulst2PVI',0,2,8,10);
>> sol8=30./(1+2*exp(-3*t8));
>> E8=max(abs(sol8-y8));
```

Finalmente, estimamos el orden del método con el cociente de los errores obtenidos duplicando el número de subintervalos:

```
>> E=[E8 E16 E32 E64 E128];
>> log2(E(1:end-1)./E(2:end));
```

En la Tabla 3 podemos observar que el error cometido disminuye cuando N aumenta y también el orden se aproxima a 4.

N	E_N	$\log_2\left(E_{N/2}/E_N\right)$
8	0.0130	–
16	0.0022	2.5818
32	1.8657e-04	3.5381
64	1.2851e-05	3.8597
128	8.3529e-07	3.9435

Tabla 3: Estimación del orden del método AM4

Partiendo de la función `AM2.m` implementada para PVI definidos por una ecuación diferencial de primer orden, en el siguiente vídeo mostramos cómo adaptar el código a sistemas de EDOs de primer orden.

Accede al vídeo: Método de Adams-Moulton para sistemas.

7.4 Métodos predictor-corrector

En general, los métodos implícitos no se utilizan como hemos visto en el apartado anterior. Se suelen utilizar para mejorar los resultados obtenidos con un método explícito. La combinación de los métodos explícito e implícito dan lugar a lo que se conoce como métodos predictor-corrector. En estos métodos la aproximación y_{k+1} sa calcula como una predicción $y_{k+1}^{(p)}$ utilizando un método explícito como los estudiados con anterioridad y posteriormente se utiliza un método implícito del mismo orden para mejorar o corregir esta predicción.

Por ejemplo, utilizando los métodos de Adams-Bashforth y Adams-Moulton de orden dos presentados anteriormente, podemos obtener un método predictor-corrector, que denotamos ABM2, de la forma:

$$\text{Predictor AB2:} \quad y_{k+1}^{(p)} = y_k + \frac{h}{2}(3f(t_k, y_k) - f(t_{k-1}, y_{k-1})),$$
$$\text{Corrector AM2:} \quad y_{k+1} = y_k + \frac{h}{2}(f(t_{k+1}, y_{k+1}^{(p)}) + f(t_k, y_k)).$$

El código de Matlab completo de este método predictor-corrector, que llamamos método de Adams-Bashforth-Moulton de dos pasos, se muestra a continuación.

ABM2.m

```
function [t,y] = ABM2(f,a,b,N,ya)
% Código para resolver un PVI con el metodo de
% Adams-Bashfort-Moulton de dos pasos
h=(b-a)/N;
t=a:h:b;
t=t(:);
y=zeros(N+1,1);
y(1)=ya;
% Primer paso con el metodo de Heun
```

```
k1 = h*feval(f,t(1),y(1));
k2 = h*feval(f,t(2),y(1)+k1);
y(2) = y(1)+(k1+k2)/2;
for k=2:N
    % predictor AB2
    k1 = feval(f,t(k),y(k));
    k2 = feval(f,t(k-1),y(k-1));
    yp = y(k)+h/2*(3*k1-k2);
    % corrector AM2
    y(k+1)=y(k)+h/2*(feval(f,t(k+1),yp)+k1);
end
end
```

De la misma forma, si utilizamos los métodos de Adams-Bashforth y Adams-Moulton de orden cuatro, obtenemos un método predictor-corrector que tiene la expresión

$$y_{k+1}^{(p)} = y_k + \frac{h}{24}(55f(t_k, y_k) - 59f(t_{k-1}, y_{k-1}) + 37f(t_{k-2}, y_{k-2}) - 9f(t_{k-3}, y_{k-3})),$$
$$y_{k+1} = y_k + \frac{h}{24}(f(t_{k-2}, y_{k-2}) - 5f(t_{k-1}, y_{k-1}) + 19f(t_k, y_k) + 9f(t_{k+1}, y_{k+1})),$$

denominado método de Adams-Bashforth-Moulton de cuatro pasos y denotado ABM4.

En el siguiente ejemplo resolvemos el PVI obtenido con el modelo de Verhulst y realizamos una comparativa entre los métodos explícitos y los métodos predictor-corrector de dos y cuatro pasos.

Ejemplo 5.

Consideremos el problema de valor inicial

$$y'(t) = (3 - 0.1y(t))y(t), \qquad t \in [0, 2], \qquad y(0) = 10.$$

En la Tabla 4 mostramos la solución del PVI con los métodos explícitos de Adams-Bashforth y los métodos predictor-corrector de Adams-Bashforth-Moulton de 2 y 4 pasos, para $N = 30$ y $t_k \in \{0, 0.6, 1, 1.4, 2\}$.

t_k	AB2	ABM2	AB4	ABM4
0	10	10	10	10
0.6	22.6066	22.5309	22.5434	22.5464
1	27.3014	27.2794	27.2832	27.2832
1.4	29.1204	29.1289	29.1267	29.1263
2	29.8465	29.8538	29.8519	29.8520

Tabla 4: Solución del PVI con AB2, ABM2, AB4 y ABM4

Podemos comparar la calidad de las aproximaciones con el error máximo cometido por cada método con respecto a la solución analítica $y(t) = \dfrac{30}{1 + 2e^{-3t}}$.

Método	AB2	ABM2	AB4	ABM4
máx $\lvert y(t_k) - y_k \rvert$	0.0613	0.0157	0.0029	2.5127e-04

Tabla 5: Error máximo cometido por los métodos

En la Tabla 5 podemos observar que en los métodos predictor-corrector el error es inferior a los respectivos métodos explícitos de dos y cuatro pasos.

7.5 Métodos numéricos para problemas rígidos

A grandes rasgos, llamamos ecuaciones rígidas o stiff a las ecuaciones diferenciales para las cuales determinados métodos numéricos son numéricamente inestables si no se toma un número muy elevado de puntos en la discretización. Normalmente, este mal funcionamiento se debe a que la ecuación presenta cambios muy bruscos en un espacio reducido de tiempo. Como consecuencia, trabajar con pocos puntos puede producir errores muy grandes en la solución discreta obtenida con determinados métodos numéricos.

En la Figura 2 podemos observar un ejemplo de función rígida. A pesar de que el intervalo de la variable temporal en $t \in [0, 0.1]$ es reducido, se trata de una función que en unas pocas milésimas experimenta un gran crecimiento. Por el contrario, tras este crecimiento la función tiene una tasa de cambio pequeña, siendo un comportamiento muy estable.

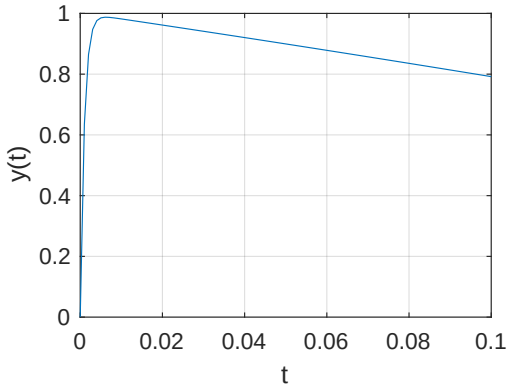


Figura 2: Gráfica de $y(t) = 3 - 1.9980e^{t} - 1.002e^{-1000t}$

Para trabajar con este tipo de problemas, mantener el paso de integración constante no suele ser una buena opción ya que todas las regiones de la solución no requieren ser estudiadas con el mismo detalle. Como consecuencia, no todos los métodos numéricos proporcionan buenos resultados. Una de las particularidades de las ecuaciones rígidas es que los métodos numéricos funcionan mejor cuanto más bajo sea su orden y suelen tener mejor funcionamiento los métodos implícitos frente a los explícitos.

No obstante, existen diversos métodos implementados en la librería de Matlab que utilizan paso adaptativo, es decir, utilizan un mayor o menor número de puntos en cada región del intervalo dependiendo del problema. Por tanto, no se definen nodos equiespaciados y se concentra un mayor número de puntos en las regiones en las cuales el comportamiento de la función cambia más rápidamente. De esta forma se recoge más información sobre la función. Entre las funciones de Matlab más utilizadas podemos encontrar `ode23`, `ode45`, `ode113`, y especialmente para problemas rígidos `ode15s` y `ode23s`, entre otras. Todos los algoritmos, además de utilizar paso adaptatipo, pueden aproximar la solución en los nodos solicitados.

Ejemplo 6.

Consideremos el siguiente PVI definido por medio de una ecuación rígida:

$$y'(t) = -1000y(t) + 3000 - 2000e^t, \qquad t \in [0, 0.1], \qquad y(0) = 0,$$

cuya solución exacta es:

$$y(t) = 3 - 1.9980e^t - 1.002e^{-1000t}.$$

En la Tabla 6 se ha comparado el error máximo cometido en la resolución del PVI con distintos métodos numéricos tanto explícitos como implícitos.

Método	N	E_N
Euler	10	3.4938e+09
	100	0.3686
Euler implícito	10	0.0911
	100	0.1324
AB4	10	8.0190e+16
	100	1.3998e+37
AM4	10	5.9765e+06
	100	0.0071
ABM4	10	3.0556e+20
	100	0.0240
`ode23`	10	0.0011
	20	0.0016
`ode45`	10	1.8939e-04
	20	6.4718e-04
`ode15s`	10	2.4195e-05
	20	2.7014e-04

Tabla 6: Error máximo cometido por los métodos

En general, los métodos explícitos no funcionan correctamente y requieren de muchos puntos para tratar de reducir el error máximo. Aun así, en el método AB4 el error se dispara también con $N = 100$. En cambio, los métodos de paso adaptativo ya tienen un buen funcionamiento considerando únicamente 10 puntos en la resolución.

Podemos analizar analíticamente el comportamiento de los métodos para saber a priori si van a converger o cuántos subintrevalos van a necesitar para converger en un problema rígido. Para ello se realiza un estudio de la estabilidad del método.

A continuación analizaremos la estabilidad de los esquemas de Euler explícito e implícito. Para ello, tomaremos como punto de partida la ecuación rígida más sencilla:

$$y'(t) = -\lambda y(t), \qquad \lambda > 0. \tag{3}$$

Para estudiar la estabilidad de los métodos, consideraremos el PVI definido por la ecuación diferencial (3) con condición inicial:

$$y'(t) = -\lambda y(t), \qquad t \in [0, 0.1], \qquad y(0) = y_0,$$

cuya solución exacta es $y(t) = y_0 e^{-\lambda t}$. Si aplicamos el método de Euler al PVI, obtenemos

$$y_{k+1} = y_k + hf(t_k, y_k) = y_k + h(-\lambda y_k) = y_k(1 - \lambda h).$$

De forma recursiva, podemos escribir

$$y_{k+1} = y_0(1 - \lambda h)^{k+1}.$$

Por tanto, la solución numérica estará acotada si se verifica

$$|1 - \lambda h| < 1.$$

Por otro lado, si aplicamos al PVI el método de Euler implícito, se tiene

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}) = y_k + h(-\lambda y_{k+1}) \quad \Leftrightarrow \quad y_{k+1}(1 + \lambda h) = y_k,$$

y de aquí,

$$y_{k+1} = y_0 \left(\frac{1}{1+\lambda h} \right)^{k+1}.$$

Como siempre se verifica $1 < 1 + \lambda h$, entonces la solución numérica estará acotada, ya que para todo valor de $\lambda > 0$ siempre se satisface

$$\left| \frac{1}{1+\lambda h} \right| < 1.$$

Como consecuencia, el método de Euler implícito será estable para los problemas rígidos de estas características, mientras que en el método de Euler explícito será necesario tomar un gran número de puntos.

Accede al vídeo: Estudio del comportamiento de métodos numéricos sobre un problema rígido: modelo de propagación de una llama