

Computació Numèrica

Laboratori 14. Equacions Diferencials amb Matlab

M. Àngela Grau Gotés

Departament de Matemàtiques
Universitat Politècnica de Catalunya · BarcelonaTech.

29 de maig de 2018

drets d'autor

“Donat el caràcter i la finalitat exclusivament docent i eminentment il·lustrativa de les explicacions a classe d'aquesta presentació, l'autor s'acull a l'article 32 de la Llei de propietat intel·lectual vigent respecte de l'ús parcial d'obres alienes com ara imatges, gràfics o altre material contingudes en les diferents diapositives”

Índex

- 1 Mètode d'Euler
- 2 Mètode de Heun
- 3 Mètode de Runge-Kutta
- 4 Mètode de Runge-Kutta
- 5 Matlab
- 6 Referències

Del problema de valors inicials:

$$\begin{aligned}y(t)' &= f(t, y(t)) \\ y(a) &= \alpha\end{aligned}\tag{1}$$

calcular $y(t)$ per a $t \in [a, b]$.

Discretització

Donat N prenem $t_i = a + ih$, per $i = 0, 1, 2, \dots, N$, amb
$$h = t_i - t_{i-1} = (b - a)/N.$$

Mètode d'Euler

Mètode d'Euler

El mètode d'Euler construeix

$$\omega_i \approx y(t_i)$$

tal que

$$\begin{aligned}\omega_0 &= \alpha \\ \omega_{i+1} &= \omega_i + h f(t_i, \omega_i)\end{aligned}\tag{2}$$

Fórmula de Taylor

$$y(t_{i+1}) = y(t_i) + h f(t_i, y(t_i)) + \frac{h^2}{2} y''(\xi_i)$$

Mètode d'Euler Modificat

Per $\omega_0 = \alpha$ el mètode és:

$$\begin{aligned}w_i &= \omega_i + \frac{h}{2} f(t_i, \omega_i) \\ \omega_{i+1} &= \omega_i + h f\left(t_i + \frac{h}{2}, w_i\right)\end{aligned}\tag{3}$$

Tal que $\omega_i \approx y(t_i)$, i per tant, $\omega_N \approx y(b)$.

Aquest mètode és un mètode de Runge-Kutta de segon orde.

Mètode de Heun

Mètode d'Euler millorat

Per $\omega_0 = \alpha$ el mètode és:

$$\begin{aligned}k_1 &= h f(t_i, \omega_i) \\k_2 &= h f(t_{i+1}, \omega_i + k_1) \\ \omega_{i+1} &= \omega_i + \frac{1}{2} (k_1 + k_2)\end{aligned}\tag{4}$$

Tal que $\omega_i \approx y(t_i)$, i per tant, $\omega_N \approx y(b)$

Aquest mètode és conegut per mètode de Heun de segon orde.

Mètode de Heun de tercer ordre

Si $\omega_0 = \alpha$ el mètode és:

$$\begin{aligned}k_1 &= h f(t_i, \omega_i) \\k_2 &= h f\left(t_i + \frac{h}{3}, \omega_i + \frac{k_1}{3}\right) \\k_3 &= h f\left(t_i + \frac{2h}{3}, \omega_i + \frac{2k_2}{3}\right) \\ \omega_{i+1} &= \omega_i + \frac{1}{4} (k_1 + 3k_3)\end{aligned}\tag{5}$$

Tal que $\omega_i \approx y(t_i)$, i per tant, $\omega_N \approx y(b)$.

Mètode de Runge-Kutta

Mètode de Runge-Kutta de quart ordre

El mètode construeix $\omega_i \approx y(t_i)$ tal que

$$\begin{aligned}\omega_0 &= \alpha \\ k_1 &= h f(t_i, \omega_i) \\ k_2 &= h f(t_i + \frac{h}{2}, \omega_i + \frac{k_1}{2}) \\ k_3 &= h f(t_i + \frac{h}{2}, \omega_i + \frac{k_2}{2}) \\ k_4 &= h f(t_{i+1}, \omega_i + k_3) \\ \omega_{i+1} &= \omega_i + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)\end{aligned}\tag{6}$$

Mètode de Runge-Kutta

Mètode de Runge-Kutta de quart ordre

El mètode construeix $\omega_i \approx y(t_i)$ tal que

$$\begin{aligned}\omega_0 &= \alpha \\ k_1 &= h f(t_i, \omega_i) \\ k_2 &= h f\left(t_i + \frac{h}{2}, \omega_i + \frac{k_1}{2}\right) \\ k_3 &= h f\left(t_i + \frac{h}{2}, \omega_i + \frac{k_2}{2}\right) \\ k_4 &= h f(t_{i+1}, \omega_i + k_3) \\ \omega_{i+1} &= \omega_i + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)\end{aligned}\tag{7}$$

Matlab

Soluciones amb Matlab

Para resolver el problema de valores iniciales

$$u' = \frac{1}{2} u, \quad u(0) = \frac{1}{4},$$

basta escribir

```
>> u = dsolve('Du = u/2','u(0) = 1/4')
```

Se obtiene

```
u =  
1/4*exp(1/2*t)
```

es decir, la solución es

$$u(t) = \frac{1}{4} e^{t/2}$$

La solución se puede representar gráficamente usando `ezplot`; por ejemplo, en el intervalo $[0,3]$.

```
>> ezplot(u,[0 3])
```


Soluciones amb Matlab

Para resolver numéricamente el problema

$$\begin{cases} u' &= (0,7 - 0,01 \cdot u)u, & t \in [0,10], \\ u(0) &= 20. \end{cases}$$

basta escribir el archivo de función

```
function du = f(t,u)
du = (0.7 - 0.01 * u) * u ;
```

y luego ejecutar la orden

```
>> [t,u] = ode45('f',[0 10],20);
```

Solvers de Matlab





Solver	Solves These Kinds of Problems	Method
<code>ode45</code>	Nonstiff differential equations	Runge-Kutta
<code>ode23</code>	Nonstiff differential equations	Runge-Kutta
<code>ode113</code>	Nonstiff differential equations	Adams
<code>ode15s</code>	Stiff differential equations and DAEs	NDFs (BDFs)
<code>ode23s</code>	Stiff differential equations	Rosenbrock
<code>ode23t</code>	Moderately stiff differential equations and DAEs	Trapezoidal rule
<code>ode23tb</code>	Stiff differential equations	TR-BDF2
<code>ode15i</code>	Fully implicit differential equations	BDFs

Solvers de Matlab

- Nonstiff problems:
 - ▶ ode45: medium accuracy. Use most of the time.
 - ▶ ode23: low accuracy. Use large error tolerances or moderately stiff problems.
 - ▶ ode113: low to high accuracy.
- Stiff problems:
 - ▶ ode15s: low to medium accuracy. Use if ode45 is slow.
 - ▶ ode23s: low accuracy. Use large error tolerances.
 - ▶ ode23t: low accuracy. Use for moderately stiff problems where you need a solution without numerical damping.
 - ▶ ode23ts: low accuracy.

Ordinary differential equations

Guies de MATLAB

-  [MathWorks Documentation Center, Matlab Users's Guide online](#)
-  [MathWorks Documentation Center, Matlab Functions's Guide online](#)
-  [MathWorks Documentation Center, Matlab Users's Guide in pdf](#)
-  [MathWorks Documentation Center, Tutorials](#)