Rezolvarea ecuaţiilor şi sistemelor de ecuaţii diferenţiale ordinare (I)

Metode unipas

Prof.dr.ing. Gabriela Ciuprina

Universitatea "Politehnica" Bucureşti, Facultatea de Inginerie Electrică

Suport didactic pentru disciplina *Metode numerice*, 2016-2017

Notes

Cuprins

Notes

Ecuaţie diferenţială

- Ecuaţie care conţine:

Notes

Formularea problemei
Metode θ
Metode explicite de ordin superior
Aspecte avansate
Ecuaţie diferenţială ordinară de ordinul 1
Sisteme de ODE de ordinul 1
Ecuaţii diferenţiale de ordin superior
Rezolvarea numerică - preliminarii

Ecuaţie diferenţială

- Ecuaţie care conţine:
  1. o funcţie necunoscută, care depinde de una sau mai multe variabile;
  2. derivate ale funcţiei necunoscute;
  3. unele dintre variabile.

3/61
Gabriela Ciuprina Ecuaţii şi sisteme diferenţiale ordinare

Notes

Ecuaţie diferenţială

- Ecuaţie care conţine:
  1. o funcţie necunoscută, care depinde de una sau mai multe variabile;
  2. derivate ale funcţiei necunoscute;
  3. unele dintre variabile.
- A rezolva o ecuaţie diferenţiala = a "integra" ecuaţia diferenţială = a găsi funcţia necunoscută.

Notes

Formularea problemei
Metode θ
Metode explicite de ordin superior
Aspecte avansate
Ecuaţie diferenţială ordinară de ordinul 1
Sisteme de ODE de ordinul 1
Ecuaţii diferenţiale de ordin superior
Rezolvarea numerică - preliminarii

Ecuaţie diferenţială

- Ecuaţie care conţine:
  1. o funcţie necunoscută, care depinde de una sau mai multe variabile;
  2. derivate ale funcţiei necunoscute;
  3. unele dintre variabile.
- A rezolva o ecuaţie diferenţiala = a "integra" ecuaţia diferenţială = a găsi funcţia necunoscută.
- Dacă funcţia necunoscută depinde de o singură variabilă $\Rightarrow$ ODE;

$$F(x(t), x'(t), x''(t), \ldots, x^{(o)}(t), t) = 0$$

$x : [t_0, T] \rightarrow \mathbb{R}$ necunoscută $\quad F : \mathbb{R}^{o+1} \rightarrow \mathbb{R}$ se dă.
- Dacă funcţia necunoscută depinde de cel puţin două variabile $\Rightarrow$ PDE;
- Ordinul ec. =

Notes

Ecuaţie diferenţială

- Ecuaţie care conţine:
  1. o funcţie necunoscută, care depinde de una sau mai multe variabile;
  2. derivate ale funcţiei necunoscute;
  3. unele dintre variabile.
- A rezolva o ecuaţie diferenţiala = a "integra" ecuaţia diferenţială = a găsi funcţia necunoscută.
- Dacă funcţia necunoscută depinde de o singură variabilă $\Rightarrow$ ODE;

$$F(x(t), x'(t), x''(t), \ldots, x^{(o)}(t), t) = 0$$

$x : [t_0, T] \to \mathbb{R}$ necunoscută $\quad F : \mathbb{R}^{o+1} \to \mathbb{R}$ se dă.

- Dacă funcţia necunoscută depinde de cel puţin două variabile $\Rightarrow$ PDE;
- Ordinul ec. = cel mai mare ordin al derivatelor care intervin.

Notes

Ecuaţie diferenţială ordinară (ODE) de ordinul 1

$$F(x(t), x'(t), t) = 0$$

$x : [t_0, T] \to \mathbb{R}$ necunoscută
$F : \mathbb{R}^3 \to \mathbb{R}$ se dă.

Caz particular - **ecuaţie diferenţială explicită**

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t)$$

$x : [t_0, T] \to \mathbb{R}$ necunoscută
$f : \mathbb{R}^2 \to \mathbb{R}$ se dă.

Numai astfel de ecuaţii formulate explicit vom considera în cele ce urmează.

Notes

Ecuaţie diferenţială ordinară (ODE) de ordinul 1

$$F(x(t), x'(t), t) = 0$$

$x : [t_0, T] \to \mathbb{R}$ necunoscută
$F : \mathbb{R}^3 \to \mathbb{R}$ se dă.

Caz particular - **ecuaţie diferenţială explicită**

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t)$$

$x : [t_0, T] \to \mathbb{R}$ necunoscută
$f : \mathbb{R}^2 \to \mathbb{R}$ se dă.

Numai astfel de ecuaţii formulate explicit vom considera în cele ce urmează.

Buna formulare?

Notes

Ecuaţie diferenţială ordinară (ODE) de ordinul 1

Se dau

$$f : \mathbb{R} \times [t_0, T] \to \mathbb{R} \qquad x_0 \in \mathbb{R}$$

Se cere

$$x : [t_0, T] \to \mathbb{R}$$

care satisface

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t) \qquad (1)$$

$$x(t_0) = x_0 \qquad (2)$$

Notes

Ecuaţie diferenţială ordinară (ODE) de ordinul 1

Se dau

$$f : \mathbb{R} \times [t_0, T] \to \mathbb{R} \qquad x_0 \in \mathbb{R}$$

Se cere

$$x : [t_0, T] \to \mathbb{R}$$

care satisface

$$\frac{dx}{dt} = f(x(t), t) \tag{1}$$
$$x(t_0) = x_0 \tag{2}$$

Problemă cu o valoare iniţială

Notes

Sisteme de ODE de ordinul 1

Sisteme de ODE formulate explicit.

Se dau

$$\mathbf{f} : \mathbb{R}^q \times [t_0, T] \to \mathbb{R}^q \qquad \mathbf{x}_0 \in \mathbb{R}^q$$

Se cere

$$\mathbf{x} : [t_0, T] \to \mathbb{R}^q$$

care satisface

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}(t), t) \tag{3}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{4}$$

Sistemul are dimensiunea q.

Notes

Sisteme de ODE de ordinul 1

Sisteme de ODE formulate explicit.

Se dau

$$\mathbf{f} : \mathbb{R}^q \times [t_0, T] \to \mathbb{R}^q \qquad \mathbf{x}_0 \in \mathbb{R}^q$$

Se cere

$$\mathbf{x} : [t_0, T] \to \mathbb{R}^q$$

care satisface

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(t), t) \tag{3}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{4}$$

Sistemul are dimensiunea q.

Problemă cu valori iniţiale

Notes

Ecuaţii diferenţiale de ordin superior

- Orice ecuaţie ODE de ordin superior poate fi transformată într-un sistem de ODE de ordinul 1 prin schimbare de variabilă. $\Rightarrow$ Ne vom ocupa doar de ecuaţii şi sisteme ODE de ordinul 1.

Exemplu:

$$a\frac{\mathrm{d}^2 y(t)}{\mathrm{d}t^2} + b\frac{\mathrm{d}y(t)}{\mathrm{d}t} + cy(t) + d = 0.$$

$\Leftrightarrow$

$$\begin{cases} \frac{\mathrm{d}y(t)}{\mathrm{d}t} = z(t) \\ \frac{\mathrm{d}z(t)}{\mathrm{d}t} = -\frac{c}{a}y(t) - \frac{b}{a}z(t) - \frac{d}{a} \end{cases}$$

Dacă notăm $\mathbf{x}(t) = \begin{bmatrix} y(t) \\ z(t) \end{bmatrix}$ atunci $\mathbf{f}(\mathbf{x}(t)) = \begin{bmatrix} z(t) \\ -\frac{c}{a}y(t) - \frac{b}{a}z(t) - \frac{d}{a} \end{bmatrix}$

Notes

Buna formulare

Buna formulare a unei ecuaţii ODE de ordin superior (q) necesită impunerea a q condiţii care se referă la valori ale funcţiei necunosute sau/şi ale derivatei ei în puncte ale domeniului de definiţie.

- Dacă toate condiţiile sunt specificate la marginea inferioară a domeniului de definiţie (t_0) atunci se spune că problema este cu valori iniţiale[1];
- Dacă se impun condiţii la începutul şi la sfârşitul domeniului de definiţie, se spune că problema este cu valori de frontieră[2].

În acest curs ne ocupăm de probleme cu valori iniţiale.

[1] *IVP - initial boundary problem*
[2] *BVP - boundary boundary problem*

Notes

Rezultatele metodelor numerice

Metoda numerică nu va furniza o expresie analitică pentru funcţia necunoscută $\mathbf{x}(t)$ ci un tabel de valori:

t_0	t_1	t_2	$\cdots$	t_n
$\mathbf{x}_0$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\cdots$	$\mathbf{x}_n$

unde $t_n = T$

Obs:

- De multe ori t reprezintă *timp* şi se poate considera $t_0 = 0$;
- Vom nota: $\mathbf{x}(t_j)$ soluţia exactă şi $\mathbf{x}_j$ aproximaţia ei

$$\mathbf{x}_j \approx \mathbf{x}(t_j)$$

- În cele ce urmează vom pp: $t_j - t_{j-1} = h \quad \Leftrightarrow \quad t_j = t_0 + jh$

Notes

Ecuaţii vs. sisteme

Prezentarea multor metode numerice pentru rezolvarea sistemelor ODE nu diferă faţa de cazul ecuaţiilor.

- De aceea, în cele ce urmează vom adopta notaţia pentru ecuaţii, în care funcţia necunoscută este $x(t)$ şi membrul drept al ecuaţiei este $f(x(t), t)$.
- În cazul sistemelor, formulele ar trebui modificate astfel încât să apară $\mathbf{x}(t)$ şi $\mathbf{f}(\mathbf{x}(t), t)$.

Metodele numerice în care această notaţie nu este potrivită vor fi discutate separat. Ele sunt metode dezvoltate pentru clase speciale de probleme, care generează sisteme ODE cu o anumită structură.

Notes

Metoda Euler explicită

Dezvoltarea în serie Taylor în jurul lui t_j

$$x(t_j + h) = x(t_j) + \frac{h}{1!}x'(t_j) + \frac{h^2}{2!}x''(t_j) + \cdots \tag{5}$$

Formula Taylor

$$x(t_j + h) = x(t_j) + \frac{h}{1!}x'(t_j) + \frac{h^2}{2!}x''(\zeta) \tag{6}$$

$$x(t_j + h) = x(t_j) + hf(x(t_j), t_j) + \mathrm{O}(h^2) \tag{7}$$

Dacă am presupune că valoarea la iteraţia j a fost calculată exact $x_j = x(t_j)$ atunci

$$x(t_j + h) = x_j + hf(x_j, t_j) + \mathrm{O}(h^2) \tag{8}$$

Notes

Metoda Euler explicită

Dacă am presupune că valoarea la iteraţia j nu este afectată de erori $x_j = x(t_j)$ atunci

$$x(t_j + h) = x_j + hf(x_j, t_j) + \mathrm{O}(h^2)$$

Dacă adoptăm ca formulă de calcul (Euler explicit)

$$x_{j+1} = x_j + hf(x_j, t_j) \tag{9}$$

atunci *eroarea locală* la iteraţia j este

$$e_l = |x(t_{j+1}) - x_{j+1}| = \mathrm{O}(h^2) \tag{10}$$

Notes

Formularea problemei
Metode θ
Metode explicite de ordin superior
Aspecte avansate
Metoda Euler explicită
Metoda Euler implicită
Exemplu
Metode θ

Metoda Euler explicită

Dacă am presupune că valoarea la iteraţia j nu a fost calculată exact $x(t_j) = x_j + e_{x_j}$ atunci

$$x(t_j + h) = x_j + e_{x_j} + hf(x_j, t_j) + \mathrm{O}(h^2)$$

Dacă adoptăm ca formulă de calcul (Euler explicit)

$$x_{j+1} = x_j + hf(x_j, t_j) \tag{11}$$

atunci *eroarea locală* este

$$e_{x_{j+1}} = |x(t_{j+1}) - x_{j+1}| = e_{x_j} + \mathrm{O}(h^2) \tag{12}$$

Notes

Metoda Euler explicită

Eroarea globală este eroarea la ultimul moment de timp

$$\begin{aligned} e_g &= |x(t_n) - x_n| = e_{x_n} + \mathrm{O}(h^2) = e_{x_{n-1}} + \mathrm{O}(h^2) + \mathrm{O}(h^2) = \\ &= n\mathrm{O}(h^2) = \frac{T - t_0}{h}\mathrm{O}(h^2) = \mathrm{O}(h) \end{aligned} \quad (13)$$

Notes

Metoda Euler explicită

O altă variantă de deducere a relaţiei de calcul

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t) \quad (14)$$

- Se scrie ecuaţia la momentul de timp discret t_j;
- Pentru derivată: formulă de diferenţe finite progresive de ordinul 1

$$\frac{x_{j+1} - x_j}{h} = f(x_j, t_j) \quad \Rightarrow \quad (15)$$

$$x_{j+1} = x_j + hf(x_j, t_j) \quad (16)$$

Notes

Metoda Euler explicită

O altă variantă de deducere a relaţiei de calcul

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t) \tag{14}$$

- Se scrie ecuaţia la momentul de timp discret t_j;
- Pentru derivată: formulă de diferenţe finite progresive de ordinul 1

$$\frac{x_{j+1} - x_j}{h} = f(x_j, t_j) \quad \Rightarrow \tag{15}$$

$$x_{j+1} = x_j + hf(x_j, t_j) \tag{16}$$

Folosirea seriei Taylor este utilă pentru estimarea erorii de trunchiere.

Notes

Metoda Euler explicită - algoritm

procedură Euler_explicit (xinit,$t0$,T,h,x)
real xinit
real $t0, T$
real h
$n = [(T - t0)/h]$
tablou real $t[n+1]$
tablou real $x[n+1]$
$t_0 = \mathrm{t0}$
$x_0 = \mathrm{xinit}$
pentru $j = 0, n-1$
    $x_{j+1} = x_j + hf(x_j, t_j)$
    $t_{j+1} = t_j + h$
•
retur

Notes

Metoda Euler implicită

Derivata: formulă de diferenţe finite regresive de ordinul 1

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t) \tag{17}$$

$$\frac{x_j - x_{j-1}}{h} = f(x_j, t_j) \quad \Rightarrow \tag{18}$$

$$x_j = x_{j-1} + hf(x_j, t_j) \tag{19}$$

Notes

Formularea problemei | Metoda Euler explicită
Metode θ | Metoda Euler implicită
Metode explicite de ordin superior | Exemplu
Aspecte avansate | Metode θ

Metoda Euler implicită

Derivata: formulă de diferenţe finite regresive de ordinul 1

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t) \tag{17}$$

$$\frac{x_j - x_{j-1}}{h} = f(x_j, t_j) \quad \Rightarrow \tag{18}$$

$$x_j = x_{j-1} + hf(x_j, t_j) \tag{19}$$

Relaţia este implicită, la fiecare pas se rezolvă o ecuaţie algebrică neliniară pentru determinarea mărimii x_j
Metoda Euler implicită

Notes

Metoda Euler implicită

Derivata: formulă de diferenţe finite regresive de ordinul 1

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), t) \tag{17}$$

$$\frac{x_j - x_{j-1}}{h} = f(x_j, t_j) \quad \Rightarrow \tag{18}$$

$$x_j = x_{j-1} + hf(x_j, t_j) \tag{19}$$

Relaţia este implicită, la fiecare pas se rezolvă o ecuaţie algebrică neliniară pentru determinarea mărimii x_j

Metoda Euler implicită

Şi în acest caz

$$e_l = \mathrm{O}(h^2) \qquad e_g = \mathrm{O}(h)$$

Notes

Metoda Euler implicită

$$x_j = x_{j-1} + hf(x_j, t_j) \tag{20}$$

Ecuaţia neliniară de rezolvat: $F(x) = 0$, unde

$$F(x) = x - x_{j-1} - hf(x, t_j)$$

- Iteraţii simple:

$$x^{(n)} = x^{(v)} + cF(x^{(v)}) \quad \text{aici} \quad x^{(n)} = x_j^{(n)} \quad x^{(v)} = x_j^{(v)}$$

$$x_j^{(n)} = x_j^{(v)} + c(x_j^{(v)} - x_{j-1} - hf(x_j^{(v)}, t_j))$$

De exemplu, dacă $c = -1$

$$x_j^{(n)} = x_{j-1} + hf(x_j^{(v)}, t_j))$$

Notes

Metoda Euler implicită

$$x_j = x_{j-1} + hf(x_j, t_j) \tag{20}$$

Ecuaţia neliniară de rezolvat: $F(x) = 0$, unde

$$F(x) = x - x_{j-1} - hf(x, t_j)$$

- Newton:

$$x^{(n)} = x^{(v)} + cF(x^{(v)}) \quad \text{aici} \quad x^{(n)} = x_j^{(n)} \quad x^{(v)} = x_j^{(v)}$$

$c = -1/F'(x^{(v)})$ unde $F'(x) = 1 - h\frac{\partial f}{\partial x}(x, t_j)$

$$x_j^{(n)} = x_j^{(v)} - \frac{x_j^{(v)} - x_{j-1} - hf(x_j^{(v)}, t_j)}{1 - h\frac{\partial f}{\partial x}(x_j^{(v)}, t_j)}$$

Notes

Metoda Euler implicită

- La fiecare pas de timp se rezolvă o ecuaţie neliniară;
- Iniţializarea rezolvării ecuaţiei neliniare - de exemplu cu soluţia de la Euler explicit (metodă *predictor-corector*);
- Eroarea şi numărul maxim admis de iteraţii pentru procedura neliniară - trebuie să fie parametri de intrare pentru Euler implicit;
- Dacă f este o funcţie liniară, atunci $F(x) = 0$ este o ecuaţie liniară, soluţia se poate calcula explicit.
- Metoda Euler este o metodă cu un pas, de ordinul 1.

Metoda cu un pas = valoarea într-un punct t_{j+1} se calculează în funcţie de comportarea funcţiei în intervalul $[t_j, t_{j+1})$. În metoda Euler, valoarea t_{j+1} se calculează în funcţie de valoarea în t_j.

Ordinul metodei = se referă la ordinul erorii globale, aici $e_g = \mathrm{O}(h)$.

Notes

Metoda Euler implicită

procedură Euler_implicit (xinit,$t0$,T,h,err,maxit,x)
real xinit
real $t0, T$
real h
real err
întreg maxit
$n = [(T - t0)/h]$
tablou real $t[n+1]$
tablou real $x[n+1]$
$t_0 = $ t0
$x_0 = $ xinit
....

Notes

Metoda Euler implicită

procedură Euler_implicit (xinit,$t0$,T,h,err,maxit,x)
....
pentru $j = 0, n-1$
$xn = x_j + hf(x_j, t_j)$; iniţializare ca la Euler explicit
; iteratii simple (c=-1)
$k = 0$
repetă
xv = xn
$xn = x_j + hf(xv, t_j)$
$k = k + 1$
$d = |xv - xn|$
până când ($d <$ err) **sau** $k >$ maxit
dacă $k >$ maxit **scrie** procedura neliniară neconvergentă
$t_{j+1} = t_j + h$
$x_{j+1} = xn$
retur *sau cu Newton:*

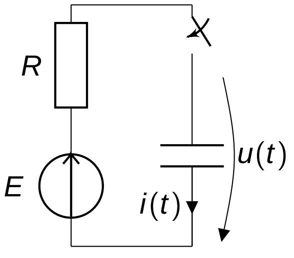
Notes

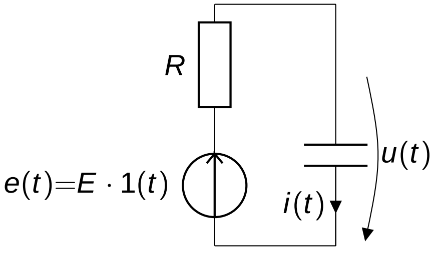
Metoda Euler implicită

procedură Euler_implicit (xinit,t0,T,h,err,maxit,x)
....
pentru $j = 0, n-1$
  $xn = x_j + hf(x_j, t_j)$; inițializare ca la Euler explicit
  ; Newton
  $k = 0$
  repetă
    xv = xn
    $xn = x_j - (xv - x_{j-1} - hf(xv, t_j))/(1 - h \cdot \mathrm{fder}(xv, t_j))$
    $k = k + 1$
    $d = |xv - xn|$
  până când ($d <$ err) **sau** $k >$ maxit
  dacă $k >$ maxit **scrie** procedura neliniară neconvergentă
  $t_{j+1} = t_j + h$
  $x_{j+1} = xn$
retur

Notes

Exemplu





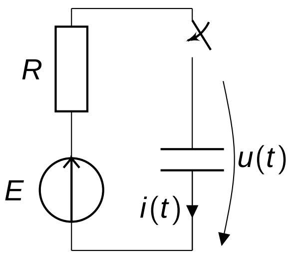
$C = 4\mu$F, $E = 20$ mV, $R = 10\ \Omega$, $u(0) = 0$.

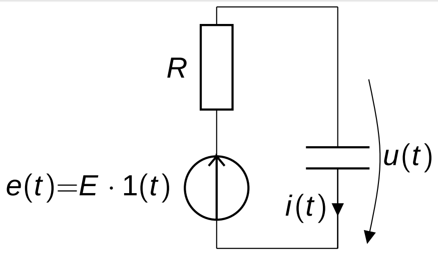
$$Ri(t) + u(t) = E \qquad i(t) = C\frac{\mathrm{d}u(t)}{\mathrm{d}t}, \tag{21}$$

$$RC\frac{\mathrm{d}u(t)}{\mathrm{d}t} + u(t) = E. \qquad u(0) = u_0 = 0. \tag{22}$$

Notes

Exemplu





Soluţie analitică

$$u(t) = (u_0 - E)\exp(-t/\tau) + E. \qquad (21)$$

unde $\tau = RC$ este *constanta de timp* a circuitului.

Notes

Exemplu

Ecuaţia (22) o rescriem ca

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} + \frac{1}{\tau}u(t) = \frac{1}{\tau}E. \qquad (22)$$

Vom urmări calculul numeric în intervalul de timp $[0, t_{max}]$ unde $t_{max} = 10\tau$ într-o reţea echidistantă de n puncte t_j, unde pasul de discretizare h este

$$t_{j+1} - t_j = h, \quad \text{pentru} \quad j = 1, \ldots, n-1. \qquad (23)$$

Vom nota valorile discrete obţinute prin rezolvare numerică cu u_j. Ele vor fi aproximaţii ale mărimii reale u.

$$u_j \approx u(t_j). \qquad (24)$$

Notes

Exemplu

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} + \frac{1}{\tau}u(t) = \frac{1}{\tau}E. \tag{25}$$

Varianta I - Euler explicit (dif. finite progresive de ord. 1)

$$\frac{u_{j+1} - u_j}{h} + \frac{1}{\tau}u_j = \frac{1}{\tau}E, \tag{26}$$

$\Rightarrow u_{j+1}$ poate fi calculată explicit cu formula

$$u_{j+1} = u_j\left(1 - \frac{h}{\tau}\right) + \frac{h}{\tau}E. \tag{27}$$

Notes

Exemplu

$$u_{j+1} = u_j\left(1 - \frac{h}{\tau}\right) + \frac{h}{\tau}E. \tag{28}$$

```
procedură euler_explicit_RC(u0,E,tau,h,N,u)
; rezolvă ecuaţia du(t)/dt + (1/τ)u(t) = (1/τ)E cu metoda diferenţelor finite
; d/dt se discretizează folosind diferenţe progresive de ordinul 1
real u0                      ; condiţia iniţială - dată
real E                       ; coeficient în ecuaţie - dată
real tau                     ; constantă de timp - dată
real h                       ; pas de discretizare al intervalului de timp - dat
intreg n                     ; număr de valori de timp - dat
tablou real u[n]             ; soluţia discretă - rezultat
u(1) = u0
pentru j = 1,n-1
        u(j+1) = u(j)*(1-h/tau) + h*E/tau
•
retur
```

Această metodă, este instabilă pentru $h > \tau$.

Notes

Exemplu

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} + \frac{1}{\tau}u(t) = \frac{1}{\tau}E. \tag{29}$$

Varianta a II-a - Euler implicit (dif. finite regresive de ord. 1)

$$\frac{u_j - u_{j-1}}{h} + \frac{1}{\tau}u_j = \frac{1}{\tau}E, \tag{30}$$

$\Rightarrow u_j$ poate fi calculată explicit:

$$u_j = \left(u_{j-1} + \frac{h}{\tau}E\right) / \left(1 + \frac{h}{\tau}\right). \tag{31}$$

Notes

Formularea problemei
Metode θ
Metode explicite de ordin superior
Aspecte avansate
Metoda Euler explicită
Metoda Euler implicită
Exemplu
Metode θ

Exemplu

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} + \frac{1}{\tau}u(t) = \frac{1}{\tau}E. \tag{29}$$

Varianta a II-a - Euler implicit (dif. finite regresive de ord. 1)

$$\frac{u_j - u_{j-1}}{h} + \frac{1}{\tau}u_j = \frac{1}{\tau}E, \tag{30}$$

$\Rightarrow u_j$ poate fi calculată explicit:

$$u_j = \left(u_{j-1} + \frac{h}{\tau}E\right) / \left(1 + \frac{h}{\tau}\right). \tag{31}$$

În acest caz nu este nevoie de rezolvarea unei ecuaţii neliniare

Notes

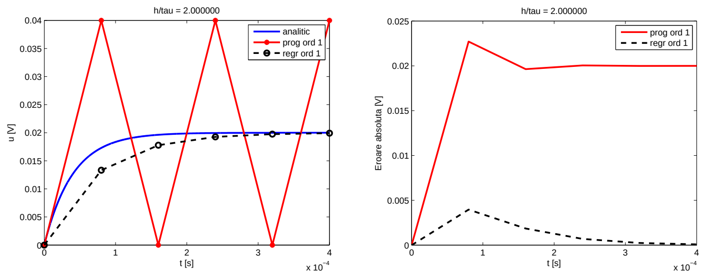
Exemplu

$$u_j = \left(u_{j-1} + \frac{h}{\tau}E\right) / \left(1 + \frac{h}{\tau}\right). \quad (32)$$

procedură euler_implicit_RC(u0,E,tau,h,n,u)
; rezolvă ecuaţia $\frac{du(t)}{dt} + \frac{1}{\tau}u(t) = \frac{1}{\tau}E$ cu metoda diferenţelor finite
; d/dt se discretizează folosind diferenţe regresive de ordinul 1
real u0 ; condiţia iniţială - dată
real E ; coeficient în ecuaţie - dată
real tau ; constantă de timp - dată
real h ; pas de discretizare al intervalului de timp - dat
întreg n ; număr de valori de timp - dat
tablou real u[n] ; soluţia discretă - rezultat
u(1) = u0
pentru j = 2,n
    u(j) = (h*E/tau + u(j-1))/(1 + h/tau)
• **retur**

Notes

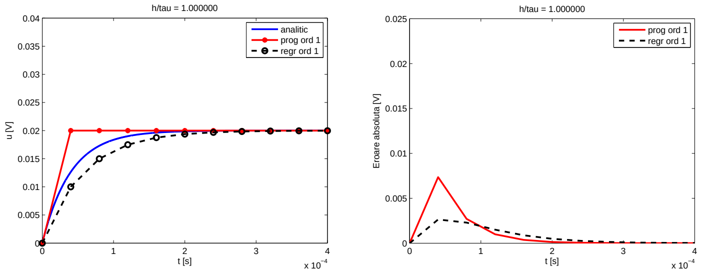
Exemplu



Cazul $h = 2\tau$.

Notes

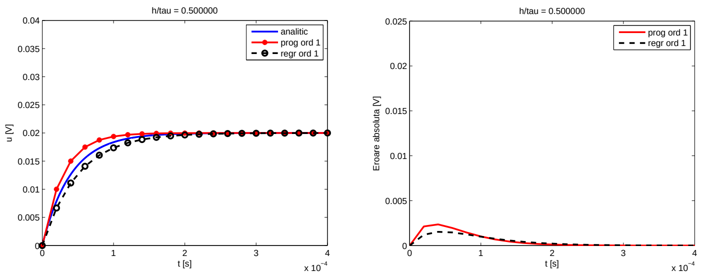
Exemplu



Cazul $h = \tau$.

Notes

Exemplu



Cazul $h = \tau/2$.

Notes

Metode θ

Metoda Euler face parte din categoria metodelor numite "θ". în care aproximaţia x_j se calculează prin rezolvarea ecuaţiei

$$-x_{j-1} + x_j = h\left[\theta f(x_{j-1}, t_{j-1}) + (1-\theta) f(x_j, t_j)\right], \qquad (33)$$

Există următoarele scheme de calcul celebre

Notes

Metode într-un pas - metode θ

- Metoda Euler explicită (sau progresivă) $\theta = 1$:
$$x_j = x_{j-1} + hf(x_{j-1}, t_{j-1}) \qquad (34)$$
unde $h = t_j - t_{j-1}$.
(Derivata: diferenţe finite progresive de ordinul 1).
- Metoda Euler implicită (sau regresivă) $\theta = 0$:
$$x_j = x_{j-1} + hf(x_j, t_j) \qquad (35)$$
unde $h = t_j - t_{j-1}$.
(Derivata: diferenţe finite regresive de ordinul 1.)
- Metoda trapezelor $\theta = 1/2$:
$$x_j = x_{j-1} + \frac{h}{2}\left(f(x_{j-1}, t_{j-1}) + f(x_j, t_j)\right) \qquad (36)$$
unde $h = t_j - t_{j-1}$.
(Pe această variantă de calcul se bazează metoda Crank-Nicolson pentru rezolvarea PDE.)

Notes

Metode într-un pas - metode θ

Justificarea numelui metodei trapezelor.

$$x(t) = \int_{t_0}^{t} f(x(t'), t')\, \mathrm{d}t'. \tag{37}$$

Considerând momentele discrete t_{j-1} şi t_j are loc

$$x(t_j) = \int_{t_0}^{t_{j-1}} f(x(t), t)\, \mathrm{d}t + \int_{t_{j-1}}^{t_j} f(x(t), t)\, \mathrm{d}t. \tag{38}$$

Schema de calcul va înlocui valorile exacte cu unele aproximative, în consecinţă

Notes

Metode într-un pas - metode θ

$$x_j = x_{j-1} + I, \tag{39}$$

unde I reprezinta o aproximare pentru integrala

$$\int_{t_{j-1}}^{t_j} f(x(t), t)\, \mathrm{d}t$$

Cea mai simplă aproximare a integralei = aria trapezului corespunzător intervalului $[t_{j-1}, t_j] \Rightarrow$

$$x_j = x_{j-1} + \frac{h}{2}\left(f(x_{j-1}, t_{j-1}) + f(x_j, t_j)\right)$$

adică exact formula (36).

Metoda trapezelor este de ordinul 2.

Notes

Metode într-un pas - metode θ

Temă (facultativ):

1. Transformaţi pseudocodul metodei Euler implicit într-o metoda generală θ, în care θ este un parametru.
2. Trataţi separat cazul $\theta = 1$ astfel încât algoritmul să includă şi metoda Euler explicită.

Notes

Metoda Taylor

Pentru a avea o acurateţe mai mare $\rightarrow$ seria Taylor cu un număr suplimentar de termeni:
Formula Taylor

$$x(t_j + h) = x(t_j) + \frac{h}{1!}x'(t_j) + \frac{h^2}{2!}x''(t_j) + \frac{h^3}{3!}x^{(3)}(\zeta) \qquad (40)$$

$$x(t_j + h) = x(t_j) + hf(x(t_j), t_j) + h^2 f'(x(t_j), t_j) + \mathrm{O}(h^3) \qquad (41)$$

$$x_{j+1} = x_j + hf(x_j, t_j) + \frac{h^2}{2}f'(x_j, t_j) \qquad (42)$$

E nevoie de f' - poate fi costisitor de evaluat.
Se preferă variante care folosesc numai evaluări ale funcţiei f.

Notes

Metoda Euler modificată

Se estimează soluţia în punctul central ca la Euler explicit

$$x_{j+1/2} = x_j + \frac{h}{2} f(x_j, t_j) \tag{43}$$

Această estimare se foloseşte pentru a aproxima derivata lui x în $t_{i+1/2}$

$$x'_{j+1/2} = f(x_{j+1/2}, t_{j+1/2}) \tag{44}$$

care se foloseşte pentru a aproxima derivata pe întregul interval $[t_j, t_{j+1}]$

$$x_{j+1} = x_j + hf(x_{j+1/2}, t_{j+1/2}) \tag{45}$$

Aceasta idee duce la familia de metode **Runge-Kutta** (RK). (Euler modificată = este o varianta de RK cu două etape)

Notes

Familia de metode Runge-Kutta - explicite

Metoda RK cu ν etape:

$$x_{j+1} = x_j + h\Phi \tag{46}$$

$\Phi = \Phi(x_j, t_j, h)$

$$\Phi = \sum_{i=1}^{\nu} b_i K_i \tag{47}$$

- b_i coeficienţi constanţi, numiţi ponderi, se aleg convenabil
- $K_i = K_i(x_j, t_j, h)$ se calculeaza astfel:

Notes

Familia de metode Runge-Kutta - explicite

$$K_1 = f(x_j, t_j) \tag{48}$$
$$K_2 = f(x_j + h(a_{21}K_1), t_j + c_2 h) \tag{49}$$
$$K_3 = f(x_j + h(a_{31}K_1 + a_{32}K_2), t_j + c_3 h) \tag{50}$$
$$\vdots$$
$$K_\nu = f(x_j + h\sum_{p=1}^{\nu-1} a_{\nu p}K_p, t_j + c_\nu h) \tag{51}$$

unde

- a_{li} coeficienţi constanţi, formează matricea Runge-Kutta, se aleg convenabil,
- c_i coeficienţi constanţi, se numesc noduri, se aleg convenabil.

Notes

Familia de metode Runge-Kutta - explicite

Scris pe scurt (util pentru scrierea pseudocodului):

RK explicite

$$x_{j+1} = x_j + h\Phi, \quad j = 0, \ldots, n-1 \tag{52}$$
$$\Phi = \sum_{i=1}^{\nu} b_i K_i \tag{53}$$
$$K_1 = f(x_j, t_j) \tag{54}$$
$$K_i = f(x_j + h\sum_{p=1}^{i-1} a_{ip}K_p, t_j + c_i h) \quad i = 2, \ldots, \nu \tag{55}$$

Notes

Familia de metode Runge-Kutta - explicite

Pentru a aplica metoda RK, trebuie specificat numărul de paşi (ordinul) ν şi valorile constante din structurile a, b, c.

Tabelul lui Butcher

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & & & & \\
c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{\nu-1} & b_\nu
\end{array}$$

Proprietăţi:

$$\sum_{i=1}^{\nu} b_i = 1$$

$$\sum_{k=1}^{i-1} a_{ik} = c_i, \quad i = 2, \nu$$

Coeficienţii se determină astfel încât relaţia RK de ordinul ν să fie echivalentă cu dezvoltarea în serie Taylor de ordinul ν.

Notes

Familia de metode Runge-Kutta - explicite

Runge-Kutta cu 1 pas

$$\begin{aligned}
x_{j+1} &= x_j + h\Phi \\
\Phi &= b_1 K_1 \\
K1 &= f(x_j, t_j)
\end{aligned} \quad \Rightarrow \quad x_{j+1} = x_j + h b_1 f(x_j, t_j)$$

Taylor

$$x_{j+1} = x_j + hf(x_j, t_j) + \mathrm{O}(h^2)$$

$\Rightarrow b_1 = 1 \Rightarrow x_{j+1} = x_j + hf(x_j, t_j)$ este Euler explicit

$$\begin{array}{c|c}
0 & \\
\hline
 & 1
\end{array}$$

Notes

Familia de metode Runge-Kutta - explicite

Runge-Kutta cu 2 paşi

$$\begin{aligned} x_{j+1} &= x_j + h\Phi \\ \Phi &= b_1 K_1 + b_2 K_2 \\ K1 &= f(x_j, t_j) \\ K2 &= f(x_j + h(a_{21}K_1), t_j + c_2 h) \end{aligned}$$

$$\Rightarrow x_{j+1} = x_j + h\left[b_1 f(x_j, t_j) + b_2 f(x_j + h(a_{21}K_1), t_j + c_2 h)\right]$$

Taylor

$$x_{j+1} = x_j + hf(x_j, t_j) + \frac{h^2}{2} f'(x_j, t_j) \mathrm{O}(h^3)$$

Notes

Familia de metode Runge-Kutta - explicite

Runge-Kutta cu 2 paşi
Se foloseşte $f'(x,t) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{\partial x}{\partial t}$ şi prin identificare rezultă condiţiile

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 c_2 &= \frac{1}{2} \\ b_2 a_{21} &= \frac{1}{2} \end{aligned}$$

3 ecuaţii, 4 necunosute, una se alege parametru (α)

Notes

Familia de metode Runge-Kutta - explicite

Runge-Kutta cu 2 paşi

$$
\begin{array}{c|cc}
0 & & \\
c_2 = \alpha & a_{21} = \alpha & \\
\hline
 & b_1 = 1 - \frac{1}{2\alpha} & b_2 = \frac{1}{2\alpha}
\end{array}
$$

Există o infinitate de metode RK cu 2 paşi, dar următoarele sunt cele mai folosite.

- $\alpha = 1/2$ -metoda Euler modificată;
- $\alpha = 1$ - metoda Heun;
- $\alpha = 2/3$ - metoda Ralston.

Notes

Familia de metode Runge-Kutta - explicite

Runge-Kutta explicită, cu 4 paşi - cea mai populară

Varianta clasică

$$
\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
1/2 & 0 & 1/2 & & \\
1 & 0 & 0 & 1 & \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}
$$

$$
\begin{aligned}
x_{j+1} &= x_j + h\Phi \\
\Phi &= (K_1 + 2K_2 + 2K_3 + K_4)/6 \\
K1 &= f(x_j, t_j) \\
K2 &= f(x_j + h/2 * K_1, t_j + h/2) \\
K3 &= f(x_j + h/2 * K_2, t_j + h/2) \\
K4 &= f(x_j + h * K_3, t_j + h)
\end{aligned}
$$

Notes

Familia de metode Runge-Kutta - explicite

procedură RungeKutta_explicit (ν,xinit,t0,T,h,x)
întreg ν ; metoda cu ν paşi
real xinit
real $t0, T$
real h
$n = [(T - t0)/h]$
tablou real $t[n + 1]$
tablou real $x[n + 1]$
tablou real $K[\nu]$
real Φ
tablou real $a[\nu, \nu]$, $b[\nu]$, $b[\nu]$; tablou Butcher
Butcher (ν,a,b,c) ; instanţiază tabelul Butcher
$t_0 = $ t0
$x_0 = $ xinit
pentru $j = 0, n - 1$; parcurge paşii de timp

Notes

Familia de metode Runge-Kutta - explicite

pentru $j = 0, n - 1$; parcurge paşii de timp
    $K_1 = f(x_j, t_j)$
    $\Phi = \Phi + b_1 K_1$
    pentru $i = 2, \nu$
        $s = 0$
        pentru $p = 1, i - 1$
            $s = s + a_{ip} K_p$
        •
        $K_i = f(x_j + hs, t_j + c_i h)$
        $\Phi = \Phi + b_i K_i$
    •
    $x_{j+1} = x_j + h\Phi$
    $t_{j+1} = t_j + h$
•
retur

Notes

Metode RK integrate

Embedded Runge-Kutta

- Estimează eroarea de trunchiere dintr-un singur pas RK, permiţând astfel adaptarea pasului de integrare.
- Pentru aceasta se folosesc două metode, una de ordin ν şi una de ordin $\nu - 1$. Tabelul Butcher al metodei de ordin superior se extinde cu ponderile din metoda de ordin inferior.

Exemple:

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & & & & \\
c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{\nu-1} & b_\nu \\
 & b_1^* & b_2^* & \cdots & b_{\nu-1}^* & 0
\end{array}$$

- Bogacki-Shampine: RK 2 şi 3;
- Fehlberg: RK 4 şi 5;
- Cash-Karp: modifică Fehlberg, RK4(5);
- Dormand-Prince: tot RK4(5).
- etc.

Notes

Metode RK integrate

Embedded Runge-Kutta

Temă (facultativ): scrieţi pseudocodul unei algoritm integrat Heun (RK2)-Euler(RK1).

Notes

Metode RK implicite

Sunt folosite atunci când RK explicite sunt instabile.

$$x_{j+1} = x_j + h\Phi, \quad j = 0, \dots, n-1 \tag{56}$$

$$\Phi = \sum_{i=1}^{\nu} b_i K_i \tag{57}$$

$$K_1 = f(x_j, t_j) \tag{58}$$

$$K_i = f(x_j + h\sum_{p=1}^{\nu} a_{ip} K_p, t_j + c_i h) \quad i = 2, \dots, \nu \tag{59}$$

Diferenţa faţa de RK explicită - suma din expresia lui k_i se face până la ν. La metoda explicită sumarea era până la $i-1$.

Notes

Metode RK implicite

Tabelul Butcher are acum matricea **a** plină.

$$\begin{array}{c|ccccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1,\nu-1} & a_{1,\nu} \\
c_2 & a_{11} & a_{12} & \cdots & a_{1,\nu-1} & a_{1,\nu} \\
\vdots & \vdots & & & & \\
c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu-1} & a_{\nu,\nu} \\
\hline
 & b_1 & b_2 & \cdots & b_{\nu-1} & b_\nu \\
 & b_1^* & b_2^* & \cdots & b_{\nu-1}^* & b_\nu^*
\end{array}
\qquad
\begin{array}{c|c}
\mathbf{c} & \mathbf{a} \\
\hline
 & \mathbf{b}^T
\end{array}$$

Şi aici se pot face algoritmi integraţi, pentru a adapta pasul de integrare.

Notes

Metode RK implicite

Exemplu:

Euler implicit (RK impl.,ord.1)

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Metoda trapezelor (RK impl.,ord.2)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

Integrarea lor 1(2)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \\ & 1 & 0 \end{array}$$

Notes

Formularea problemei
Metode θ
Metode explicite de ordin superior
Aspecte avansate
Adaptarea pasului - metode RK integrate
Metode implicite
Sisteme *stiff* (rigide)
Exemple din mediile uzuale în care lucraţi

Metode RK implicite

Exemplu:

Euler implicit (RK impl.,ord.1)

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Metoda trapezelor (RK impl.,ord.2)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

Integrarea lor 1(2)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \\ & 1 & 0 \end{array}$$

Temă (facultativ) - scrieţi pseudocodul acestei metode.

50/61
Gabriela Ciuprina
Ecuaţii şi sisteme diferenţiale ordinare

Notes

Metode RK implicite

Variante celebre:

- Gauss-Legendre - bazate pe integrarea numerică Gauss;
- Lobato (3 familii de metode IIIA, IIIB şi IIIC) - au $c_1 = 0, c_\nu = 1$;
- Radau (2 familii de metode, IA şi IIA).

Notes

Sisteme rigide

Sistem rigid = sistem pentru care o anumită metodă numerică nu converge dacă pasul de integrare nu este ales foarte mic.
Obs:

- Într-un astfel de punct soluţia sistemului nu are variaţii mari. Este vina sistemului, nu a soluţiei lui.
- Nu există o definiţie riguroasă pentru un sistem rigid.

Dacă sistemul de rezolvat poate fi pus sub forma

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{A}\mathbf{x} + \mathbf{g}(t)$$

atunci el este "rigid" dacă raportul modulelor părţilor reale ale valorilor proprii extreme ale matricei **A** este mare.
Soluţia are componente care descresc mult mai repede decât altele $\Leftrightarrow$ constantele lui de timp au ordine de mărime diferite.
Pentru astfel de sisteme trebuie folosite metode implicite.

Notes

Sisteme rigide

Sistem rigid = sistem pentru care o anumită metodă numerică nu converge dacă pasul de integrare nu este ales foarte mic.
Obs:

- Într-un astfel de punct soluţia sistemului nu are variaţii mari. Este vina sistemului, nu a soluţiei lui.
- Nu există o definiţie riguroasă pentru un sistem rigid.

Dacă sistemul de rezolvat poate fi pus sub forma

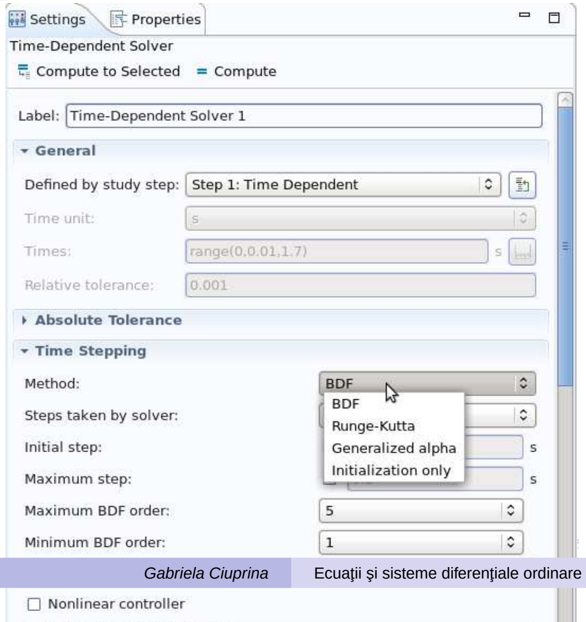
$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{A}\mathbf{x} + \mathbf{g}(t)$$

atunci el este "rigid" dacă raportul modulelor părţilor reale ale valorilor proprii extreme ale matricei **A** este mare.
Soluţia are componente care descresc mult mai repede decât altele $\Leftrightarrow$ constantele lui de timp au ordine de mărime diferite.
Pentru astfel de sisteme trebuie folosite metode implicite.

Temă (facultativ) Studiaţi problema sistemelor rigide, criterii de stabilitate (A, L) pentru metodele unipas.

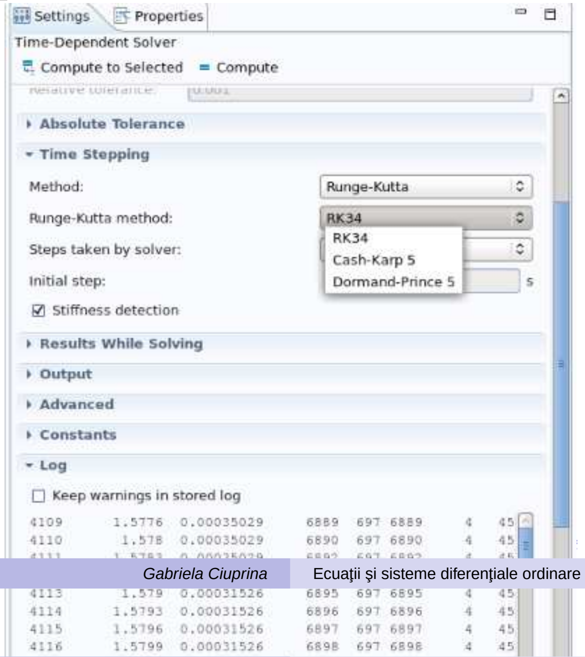
Notes

COMSOL



Notes

COMSOL



Notes

Matlab (non-stiff)

https://ch.mathworks.com/help/matlab/math/choose-an-ode-solver.html

Solver	Problem Type	Accuracy	When to Use
ode45	Nonstiff	Medium	Most of the time. ode45 should be the first solver you try.
ode23		Low	ode23 can be more efficient than ode45 at problems with crude tolerances, or in the presence of moderate stiffness.
ode113		Low to High	ode113 can be more efficient than ode45 at problems with stringent error tolerances, or when the ODE function is expensive to evaluate.

Notes

Matlab (non-stiff) https://ch.mathworks.com/help/matlab/math/choose-an-ode-solver.html

- Single-step

ode45 is based on an explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair.

ode23 is an implementation of an explicit Runge-Kutta (2,3) pair of Bogacki and Shampine. It may be more efficient than ode45 at crude tolerances and in the presence of moderate stiffness.

- Multi-step

ode113 is a variable-step, variable-order Adams-Bashforth-Moulton PECE solver of orders 1 to 13. The highest order used appears to be 12, however, a formula of order 13 is used to form the error estimate and the function does local extrapolation to advance the integration at order 13. It may be more efficient than ode45 at stringent tolerances or if the ODE function is particularly expensive to evaluate.

Notes

Matlab (stiff) https://ch.mathworks.com/help/matlab/math/choose-an-ode-solver.html

`ode15s`	Stiff	Low to Medium	Try `ode15s` when `ode45` fails or is inefficient and you suspect that the problem is stiff. Also use `ode15s` when solving differential algebraic equations (DAEs).
`ode23s`		Low	`ode23s` can be more efficient than `ode15s` at problems with crude error tolerances. It can solve some stiff problems for which `ode15s` is not effective. `ode23s` computes the Jacobian in each step, so it is beneficial to provide the Jacobian via `odeset` to maximize efficiency and accuracy. If there is a mass matrix, it must be constant.
`ode23t`		Low	Use `ode23t` if the problem is only moderately stiff and you need a solution without numerical damping. `ode23t` can solve differential algebraic equations (DAEs).
`ode23tb`		Low	Like `ode23s`, the `ode23tb` solver might be more efficient than `ode15s` at problems with crude

Notes

Matlab (stiff) https://ch.mathworks.com/help/matlab/math/choose-an-ode-solver.html

- Single-step

ode23s is based on a modified Rosenbrock formula of order 2. Because it is a single-step solver, it may be more efficient than ode15s at solving problems that permit crude tolerances or problems with solutions that change rapidly. It can solve some kinds of stiff problems for which ode15s is not effective. The ode23s solver evaluates the Jacobian during each step of the integration, so supplying it with the Jacobian matrix is critical to its reliability and efficiency.

ode23t is an implementation of the trapezoidal rule using a "free" interpolant. This solver is preferred over ode15s if the problem is only moderately stiff and you need a solution without numerical damping. ode23t also can solve differential algebraic equations (DAEs)

- Multi-step ode15s, ode23tb

Notes

Matlab (stiff) https://ch.mathworks.com/help/matlab/math/choose-an-ode-solver.html

- Single-step ode23s, ode23t
- Multi-step

ode15s is a variable-step, variable-order (VSVO) solver based on the numerical differentiation formulas (NDFs) of orders 1 to 5. Optionally, it can use the backward differentiation formulas (BDFs, also known as Gearş method) that are usually less efficient. Like ode113, ode15s is a multistep solver. Use ode15s if ode45 fails or is very inefficient and you suspect that the problem is stiff, or when solving a differential-algebraic equation (DAE).

ode23tb is an implementation of TR-BDF2, an implicit Runge-Kutta formula with a trapezoidal rule step as its first stage and a backward differentiation formula of order two as its second stage. By construction, the same iteration matrix is used in evaluating both stages. Like ode23s and ode23t, this solver may be more efficient than ode15s for problems with crude tolerances.

Notes

Matlab (fully-implicit) https://ch.mathworks.com/help/matlab/math/choose-an-ode-solver.html

`ode15i`	Fully implicit	Low	Use `ode15i` for fully implicit problems $f(t,y,y') = 0$ and for differential algebraic equations (DAEs) of index 1.

- Multi-step

ode15i is a variable-step, variable-order (VSVO) solver based on the backward differentiation formulas (BDFs) of orders 1 to 5. ode15i is designed to be used with fully implicit differential equations and index-1 differential algebraic equations (DAEs). The helper function decic computes consistent initial conditions that are suitable to be used with ode15i.

Notes

Formularea problemei | Adaptarea pasului - metode RK integrate
Metode θ | Metode implicite
Metode explicite de ordin superior | Sisteme *stiff* (rigide)
Aspecte avansate | Exemple din mediile uzuale în care lucraţi

Referinţe

- [Cheney08] W.Cheney, D.Kincaid, *Numerical Mathematics and Computing*, Brooks/Cole publishing Company,2000. (Capitolele 10 şi 11)

  Disponibilă la http://www.physics.brocku.ca/Courses/5P10/References/cheneykincaid.pdf
- [Press02] W.H.Press, S.A.Teukolsky, W.T. etterling, B.P. Flannery, *Numerical Recipies in C*, 2002. (Capitolul 16)

  Disponibilă la https://www2.units.it/ipl/students_area/imm2/files/Numerical_Recipes.pdf
- [Strang&Moler15] *Introduction to Differential Equations and the MATLAB ODE Suite* - Open Courseware at MIT

  Disponibil la http://ocw.mit.edu/RES-18-009F15 sau https://www.youtube.com/watch?v=ZvL88xqYSak
- [Trefethen18] N.Trefethen, *Exploring ODEs*, SIAM 2018.

  Disponibil la https://people.maths.ox.ac.uk/trefethen/books.html

Notes