Metode Numerice în Mecanică - Test 1

Corneliu Meseșan

14 ianuarie 2023

Problema 1

11. Rezolvați și estimați eroarea pentru

$$y' = \frac{1}{t^2} - \frac{y}{t} - y^2, \quad 1 \le t \le 2, \quad y(1) = -1,$$

folosind metodele imbricate date prin tabelele Butcher de mai jos.

Tabela Butcher pentru metoda Runge-Kutta de ordinul 5

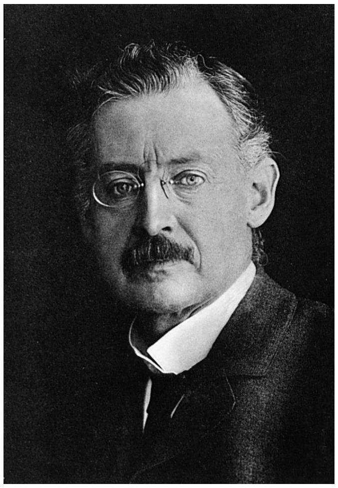
$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{2}{9} & \frac{2}{9} & & & & \\
\frac{1}{3} & \frac{1}{12} & \frac{1}{4} & & & \\
\frac{3}{4} & \frac{69}{128} & -\frac{243}{128} & \frac{135}{64} & & \\
1 & -\frac{17}{12} & \frac{27}{4} & -\frac{27}{5} & \frac{16}{5} & \\
\hline
 & \frac{1}{9} & 0 & \frac{9}{20} & \frac{16}{45} & \frac{1}{12}
\end{array}$$

Tabela Butcher pentru metoda Runge-Kutta de ordinul 6

$$\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{2}{9} & \frac{2}{9} & & & & & \\
\frac{1}{3} & \frac{1}{12} & \frac{1}{4} & & & & \\
\frac{3}{4} & \frac{69}{128} & -\frac{243}{128} & \frac{135}{64} & & & \\
1 & -\frac{17}{12} & \frac{27}{4} & -\frac{27}{5} & \frac{16}{5} & & \\
\frac{5}{6} & \frac{65}{432} & -\frac{5}{16} & \frac{13}{16} & \frac{4}{27} & \frac{5}{144} & \\
\hline
 & \frac{47}{450} & 0 & \frac{12}{25} & \frac{32}{225} & \frac{1}{30} & \frac{6}{25}
\end{array}$$

Comaprați cu soluția analitică $y(t) = -\frac{1}{t}$.

Descrierea metodei



Carl Runge (1856 - 1927)

Martin Kutta (1867–1944)

Metodele Runge-Kutta au fost dezvoltate în jurul anilor 1900 de către matematicienii germani Carl Runge și Martin Kutta.

Metodele de ordin 5 și 6 folosesc o formă generalizată a metodei Runge-Kutta de ordinul 4. Pentru rezolvarea numerică a unei ecuații diferențiale ordinare, metodele explicite iau valori de forma

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i,$$

unde

$$\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f(t_n + c_2 h, y_n + (a_{21} k_1) h) \\
k_3 &= f(t_n + c_3 h, y_n + (a_{31} k_1 + a_{32} k2) h) \\
&\vdots \\
k_s &= f(t_n + c_s h, y_n + (a_{s1} k_1 + a_{s2} k_2 + \cdots + a_{s,s-1} k_{s-1}) h).
\end{aligned}$$

Aceste date sunt afișate, de obicei, într-o tabelă Butcher, astfel:

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}$$

Implementarea

Metoda Runge-Kutta de ordin 5

Implementarea metodei presupune inițializarea capetelor `a` și `b` ale intervalului, selectarea unui pas convenabil `N`, inițializarea numărului de noduri `h` folosind formula $h = \dfrac{b-a}{N-1}$, inițializarea vectorului soluției numerice, `y`, și a soluției exacte, `ye`, introducerea condițiilor inițiale, stabilirea pașilor de timp $t \in (a, b)$ cu pasul `h` și inițializarea celor cinci `K` care generează soluția numerică a ecuației cu ajutorul tabelei Butcher corespunzătoare.

```
a = 1; b = 2; %capetele intervalului
N = 10; %pasul retelei
h = (b - a) / (N - 1); %numarul de noduri
y = zeros(N, 1); %initializam vectorul solutie
ye = zeros(N, 1); %solutia exacta
y(1)=-1; ye(1)=-1; %conditiile initiale
t = a:h:b; %pasii de timp
for i = 2 : N
    K1 = fEx(t(i - 1), y(i-1));
    K2 = fEx(t(i - 1) + 2 * h / 9, y(i - 1) + 2 * h * K1 / 9);
    K3 = fEx(t(i - 1) + h / 3, y(i - 1) + K1 * h / 12 + h * ...
        K2 / 4);
    K4 = fEx(t(i - 1) + 3 * h / 4, y(i - 1) + K1 * h * 69 / ...
        128 + K2 * h * (-243) / 128 + h * K3 * 125 / 64);
    K5 = fEx(t(i - 1) + h, y(i - 1) + K1 * h * (-17) / 12 + ...
        K2 * h * 27 / 4 + K3 * h * (-27) / 5 + K4 * h * 16/5);
    y(i) = y(i - 1) + h * (K1 / 9 + 9 * K3 / 20 + 16 * K4 / ...
        45 + K5 / 12);
    ye(i) = -1 / t(i);
end
```

Metoda Runge-Kutta de ordin 6

Metoda de ordinul 6 se implementează identic cu cea de ordin 5, singura diferență constând în numărul coeficienților `K`, care acum sunt 6, în loc de 5.

```
for i = 2 : N
    K1 = fEx(t(i - 1), y(i-1));
    K2 = fEx(t(i - 1) + 2 * h / 9, y(i - 1) + 2 * h * K1 / 9);
    K3 = fEx(t(i - 1) + h / 3, y(i - 1) + K1 * h / 12 + h * K2 / 4);
    K4 = fEx(t(i - 1) + 3 * h / 4, y(i - 1) + K1 * h * 69 / 128 + ...
        K2 * h * (-243) / 128 + h * K3 * 125 / 64);
    K5 = fEx(t(i - 1) + h, y(i - 1) + K1 * h * (-17) / 12 + K2 * ...
        h * 27 / 4 + K3 * h * (-27) / 5 + K5 * h * 16/5);
    K6 = fEx(t(i - 1) + 5 * h / 6, y(i - 1) + K1 * h * 65 / 432 + ...
        K2 * h * (-5) / 16 + K3 * h * 13 / 16 + K4 * h * 4 / 27 ...
        + K5 * h * 5 / 144);
    y(i) = y(i - 1) + h * (47 * K1 / 450 + 12 * K3 / 25 + 32 * K4 ...
        / 225 + K5 / 30 + 6 * K6 / 25);
    ye(i) = -1 / t(i);
end
```

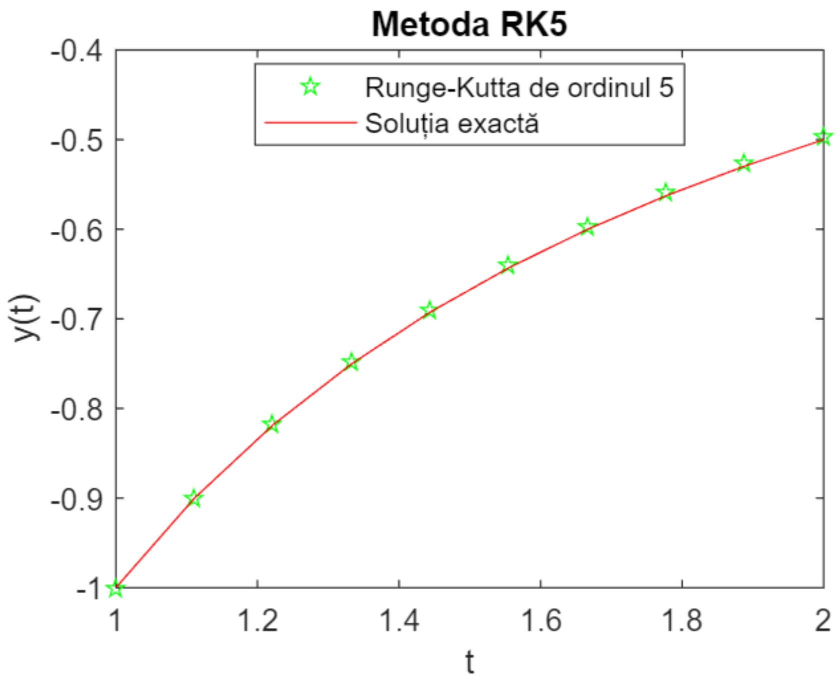
Eroarea

Erorile sunt calculate folosind comanda `norm` din MATLAB, cu setarea `Inf`, pentru a obține norma Cebîșev.

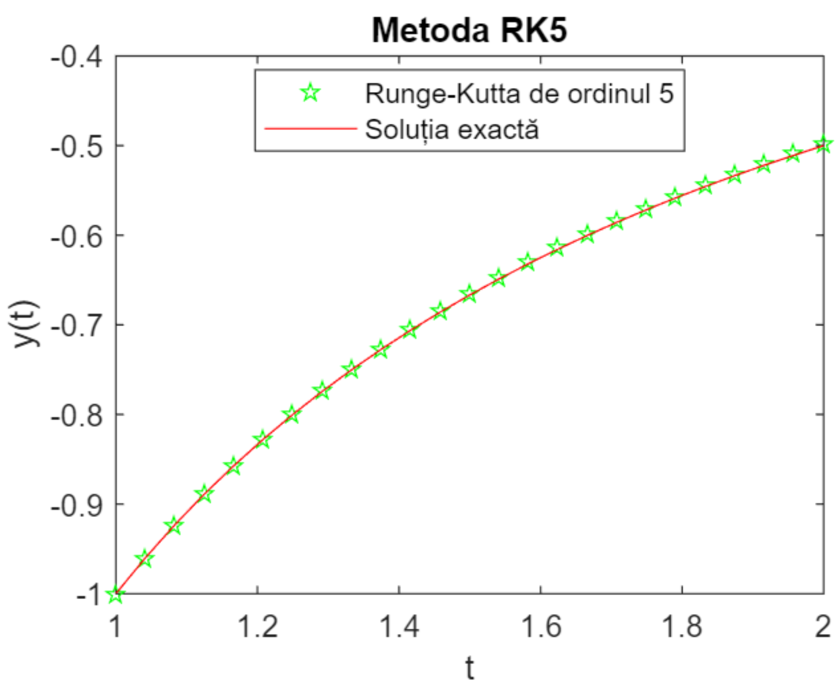
```
fprintf('Eroarea este %.16e.', norm(ye-y,Inf))
```

Pentru Metoda Runge-Kutta de ordin 5, comparată cu soluția exactă, eroarea este de $\approx 4.4708230057637910 \cdot 10^{-3}$ pentru `N` $= 10$ și $\approx 2.4415580702475959 \cdot 10^{-3}$ pentru `N` $= 25$, iar pentru metoda Runge-Kutta de ordin 6 eroarea este de $\approx 1.6086497165913038 \cdot 10^{-3}$ pentru `N` $= 10$ și $\approx 9.965813662961194 \cdot 10^{-4}$ pentru `N` $= 25$.

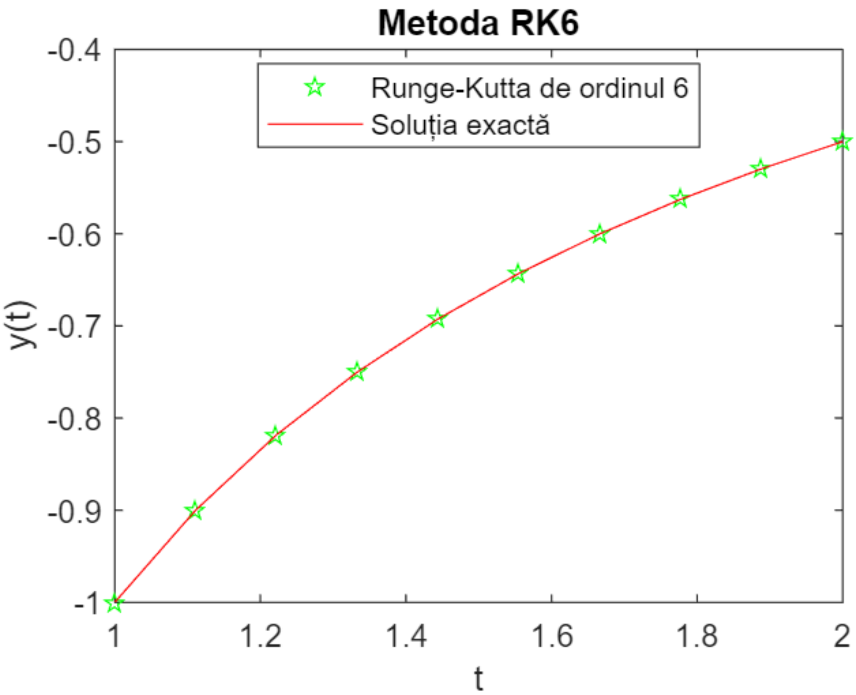
Graficele



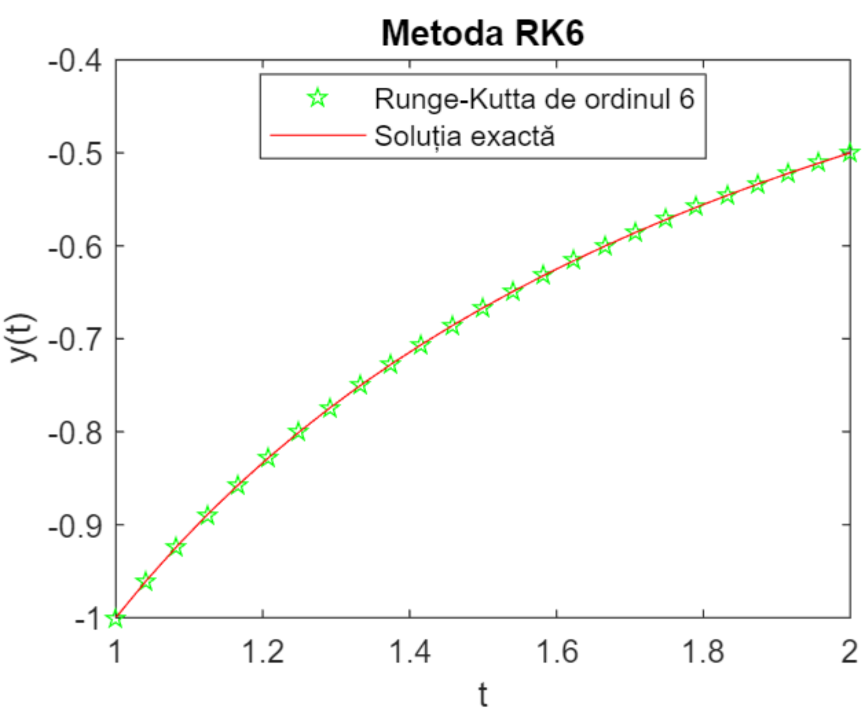
Soluția numerică obținută cu ajutorul metodei de ordin 5 și soluția exactă, `N` $= 10$



Soluția numerică obținută cu ajutorul metodei de ordin 5 și soluția exactă, `N` $= 25$



Soluția numerică obținută cu ajutorul metodei de ordin 6 și soluția exactă, $\mathtt{N} = 10$



Soluția numerică obținută cu ajutorul metodei de ordin 6 și soluția exactă, $\mathtt{N} = 25$

Problema 2

11. Rezolvați ecuația

$$y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2}, \quad 1 \le x \le 2, \quad y(1) = 1, \quad y(2) = 2.$$

Comparați cu soluția analitică

$$y = c_1 x + \frac{c_2}{x^2} - \frac{3}{10}\sin(\ln x) - \frac{1}{10}\cos(\ln x),$$

$$c_2 = \frac{1}{70}[8 - 12\sin(\ln 2) - 4\cos(\ln 2)] \approx -0.03920701320$$

$$c_1 = \frac{11}{10} - c_2 \approx 1.1392070132.$$

Descrierea metodei

Rezolvarea problemei cu valori pe frontiere dată presupune rescrierea ecuației sub formă de sistem și implementarea acestuia sub formă de funcție auxiliară, implementarea unei funcții care memorează valorile pe frontieră, implementarea Jacobianului și rezolvarea ecuației cu valori pe frontieră cu ajutorul funcției `bvp4c` din Matlab.

Implementarea

Implementarea presupune inițializarea structurii `opts` cu ajutorul funcției `bvpset`, făcând referință la Jacobian, stabilind toleranțele și activarea informațiilor legate de numărul de puncte utilizare și apelările funcțiilor, inițializarea intervalului din enunț și a soluțiilor inițiale și a soluției exacte.

```
opts = bvpset('FJacobian',@jac,'RelTol',0.1,'AbsTol',0.1,'Stats','on');
xmesh = linspace(1, 2, 30);
solinit = bvpinit(xmesh, [1; 2]);
sol4c = bvp4c(@bvpfcn, @bcfcn, solinit, opts);
xplot = linspace(1,2,30);
c2 = 1/70*(8 - 12*sin(log(2)) - 4*cos(log(2)));
c1 = 11/10 - c2;
yplot = c1*xplot + c2./xplot.^2 - 3/10*sin(log(xplot)) - ...
    ↪ 1/10*cos(log(xplot));
```

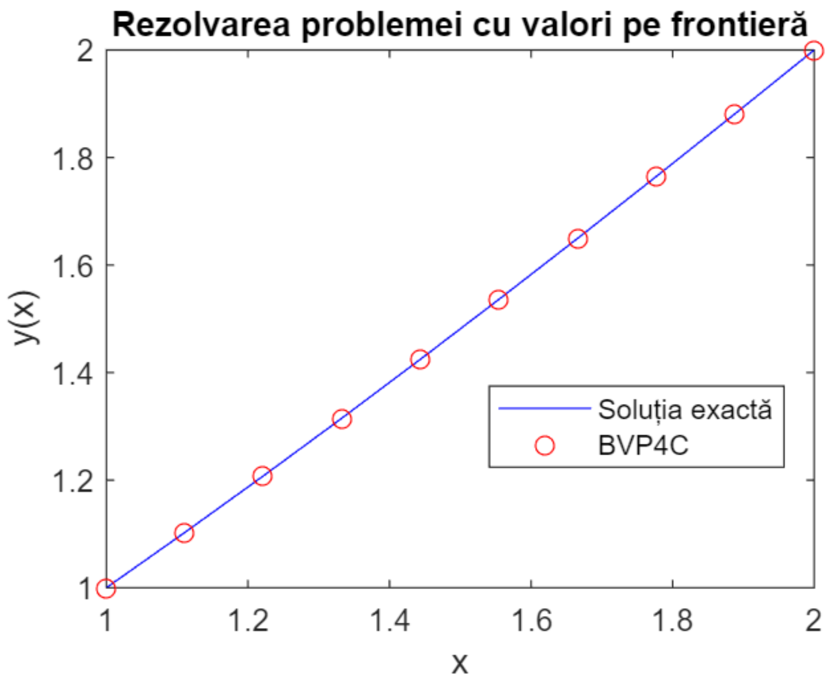
Eroarea

Eroarea este calculată folosind comanda `norm` din Matlab, cu setarea `Inf`, pentru a obține norma Cebîșev.

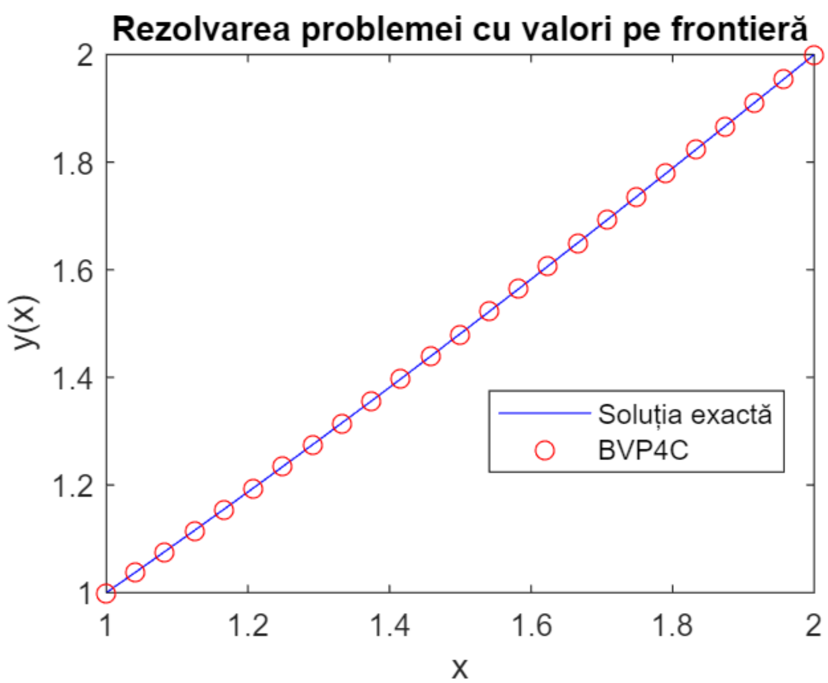
```
fprintf('Eroarea este %.16e.', norm(sol4c.y(1,:) - yplot,Inf))
```

Eroarea obținută este $\approx 3,4711620577887459 \cdot 10^{-7}$ pentru $\texttt{N} = 10$ și $\approx 6.8858752033662540 \cdot 10^{-9}$ pentru $\texttt{N} = 25$.

Graficele



Soluția numerică obținută cu ajutorul funcției `bvp4c` și soluția exactă, $\texttt{N} = 10$



Soluția numerică obținută cu ajutorul funcției `bvp4c` și soluția exactă, $\texttt{N} = 25$

Funcții auxiliare

Problema 1

Funcția `fEx` reprezintă membrul drept al ecuației diferențiale inițiale.

```
function rez = fEx(t, y)
rez = 1 / (t ^ 2) - y /t - y ^ 2;
end
```

Problema 2

Funcția `bvpfcn` corespunde sistemului

$$\begin{cases} y_1' = y_2, \\ y_2' = -\frac{2}{x}y_2 + \frac{2}{x^2}y_1 + \frac{\sin(\ln x)}{x^2} \end{cases},$$

unde $y_1 = y$ și $y_2 = y'$.

```
function dydx = bvpfcn(x,y)
    dydx = [y(2)
        -2*y(2)/x + 2*y(1)/x^2 + sin(log(x))/x^2];
end
```

Funcția `bcfcn` corespunde valorilor pe frontiere

$$y(1) = 1, \quad y(2) = 2.$$

```
function res = bcfcn(ya,yb)
    res = [ya(1)-1
           yb(1)-2];
end
```

Funcția `jac` corespunde Jacobianului sistemului dat de

$$J = \frac{\partial f_i}{\partial y} = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{2}{x^2} & -\frac{2}{x} \end{pmatrix}.$$

```
function dfdy = jac(x,y)
    dfdy = [0       1
            2/x^2  -2/x];
end
```