Методи на Рунге-Кута

Методите на Рунге-Кута подобряват метода на Ойлер за решаване не началната задача за обикновени диференциални уравнения (ОДУ) от вида $y' = f(t, y)$, $y(t_0) = y_0$, без да се изисква изчисляването на производни от висок ред.

Съгласно теорията, общият вид на изчислителните етапи на явните методи на Рунге-Кута е:

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

$$k_i = f\left(t_n + c_i h,\ y_n + h \sum_{j=1}^{s} a_{ij} k_j\right)$$

където $a_{ij} = 0$ to $j \geq i$ и $\sum_{j=1}^{s} a_{ij} = c_i$.

Класическият метод на Рунге-Кута от втори ред с 2 етапа се записва със следната таблица на Бутчер:

0	0	.
c_2	c_2	0
.	b_1	b_2

където коефициентите удовлетворяват системата:

$$b_1 + b_2 = 1$$
$$b_2 c_2 = \frac{1}{2}$$

Оттук следва, че съществува фамилия от безброй много методи на Рунге-Кута от втори ред. Най-често използваните са:

а) **Модифициран метод на Ойлер**, който се получава при $b_1 = 0$, $b_2 = 1$, $c_2 = \frac{1}{2}$. Формулите на метода са: $y_{n+1} = y_n + h k_2$, където $k_1 = f(t_n, y_n)$ и $k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h k_1}{2}\right)$.

Следват две програмни процедури на ***Mathematica*** - **eulermod** и **eulermodgraf.** С тях се изчисляват таблици от стойности на приближеното решение и графичното му представяне за модифицирания метод на Ойлер.

```
eulermod[f_, h_, ini_, a_, b_] :=
Module[{yrk, t, y, rktable1, c},
 c = (b - a) / h;
 yrk[0] = ini;
 t[n_] := a + n h;
 yrk[n_] :=
  Module[{k1, k2},
   k1 = f[t[n - 1], yrk[n - 1]];
   k2 = f[t[n - 1] + h/2, yrk[n - 1] + h/2 k1];
   yrk[n] = yrk[n - 1] + h k2];
 rktable1 = Table[yrk[i], {i, 0, c}];
Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
eulermodgraf[f_, h_, ini_, a_, b_] := Module[
  {yrk, t, y, rktable1, c}, c = (b - a)/h; yrk[0] = ini; t[n_] := a + n h;
  yrk[n_] := Module[{k1, k2}, k1 = f[t[n - 1], yrk[n - 1]];
    k2 = f[t[n - 1] + h/2, yrk[n - 1] + h k1/2]; yrk[n] = yrk[n - 1] + h k2];
  rktable1 = Table[yrk[i], {i, 0, c}];
  ListPlot[Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}], Joined → True,
   PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable1[[c + 1]]]]
```

където **f** е дясната част на диференциалното уравнение, **h** е дължината на стъпката, **ini** е стойността на началното условие, а краищата на интервала са **a** и **b.**

б) **Подобрен метод на Ойлер**, който се получава при $b_1 = \frac{1}{2}, b_2 = \frac{1}{2}, c_2 = 1$. Неговите формули са: $y_{n+1} = y_n + \frac{h(k_1+k_2)}{2}$, където $k_1 = f(t_n, y_n)$ и $k_2 = f(t_n + h, y_n + h\,k_1)$.

Следват две програмни процедури на ***Mathematica*** - **mejoreuler** и **mejoreulergraf**. С тях се изчислява таблица от стойности и графика на приближеното решение, получавано по подобрения метод на Ойлер.

```
mejoreuler[f_, h_, ini_, a_, b_] :=
Module[{yrk, t, y, rktable1, c},
 c = (b - a) / h;
 yrk[0] = ini;
 t[n_] := a + n h;
 yrk[n_] :=
  Module[{k1, k2},
   k1 = f[t[n - 1], yrk[n - 1]];
   k2 = f[t[n - 1] + h, yrk[n - 1] + h k1];
   yrk[n] = yrk[n - 1] + (h / 2) (k1 + k2)];
 rktable1 = Table[yrk[i], {i, 0, c}];
Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
mejoreulergraf[f_, h_, ini_, a_, b_] :=
 Module[{yrk, t, y, rktable1, c}, c = (b - a)/h;
  yrk[0] = ini; t[n_] := a + n h; yrk[n_] := Module[{k1, k2},
    k1 = f[t[n - 1], yrk[n - 1]]; k2 = f[t[n - 1] + h, yrk[n - 1] + h k1];
    yrk[n] = yrk[n - 1] + 1/2 h (k1 + k2)];
  rktable1 = Table[yrk[i], {i, 0, c}];
  ListPlot[Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}], Joined → True,
   PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable1[[c + 1]]]]
```

където отново **f** е дясната част на диференциалното уравнение, **h** е дължината на стъпката, **ini** е стойността на началното условие, а краищата на интервала са **a** и **b.**

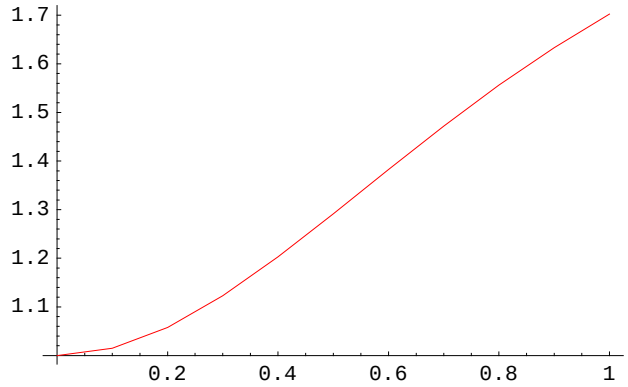
Двата метода са използвани за решаване на началната задача за следното ОДУ: $y' = -t\,y + \frac{4t}{y}$, $y(0)=1$ в интервала [0,1], със стъпка 0.1.

```
f[t_, y_] = -t y + 4 t / y;
```

eulermod[f, 0.1, 1, 0, 1]

0	1
0.1	1.015
0.2	1.05783
0.3	1.12286
0.4	1.20303
0.5	1.29151
0.6	1.38258
0.7	1.47185
0.8	1.55615
0.9	1.63337
1.	1.70225

eulermodgraf[f, 0.1, 1, 0, 1]

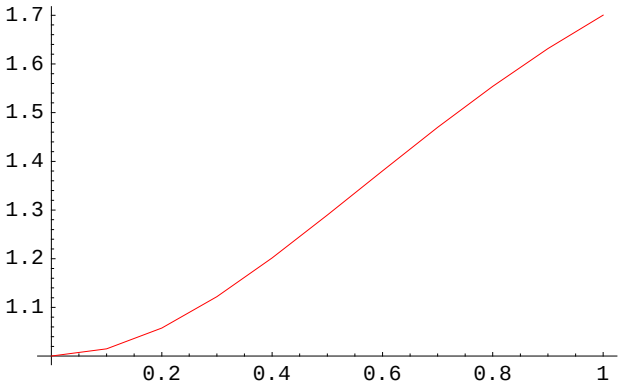


y[1.]=1.70225

mejoreuler[f, 0.1, 1, 0, 1]

0	1
0.1	1.015
0.2	1.05749
0.3	1.12202
0.4	1.20169
0.5	1.28977
0.6	1.38058
0.7	1.46972
0.8	1.55398
0.9	1.63123
1.	1.70021

mejoreulergraf[f, 0.1, 1, 0, 1]




```
y[1.]=1.70021
```

В предишната глава по едностъпкови методи за ОДУ бе показано, че точното решение на тази задача при $t = 1$ е равно на 1.70187. Следователно, в този случай, приближението, плучено по метода на Рунге-Кута от 1-ви ред е по-добро от сега полученото приближение с методи от 2-ри ред.

Класическите методи на Рунге-Кута от трети ред с 3 етапа се описват със следната таблица на Бутчер:

0	0	.	.
c_2	c_2	0	.
c_3	$c_3 - a_{32}$	a_{32}	0
.	b_1	b_2	b_3

където коефициентите удовлетворяват системата:

$$b_1 + b_2 + b_3 = 1$$
$$b_2\, c_2 + b_3\, c_3 = \frac{1}{2}$$
$$b_2\, c_2^2 + b_3\, c_3^2 = \frac{1}{3}$$
$$b_3\, c_2\, a_{32} = \frac{1}{6}$$

Тъй като отново уравненията имат повече неизвестни отколкото е броят на уравненията, се получават безброй методи на Рунге-Кута от трети ред. Най-често използваните от тях е:

0	0	.	.
$\frac{1}{2}$	$\frac{1}{2}$	0	.
1	−1	2	0
.	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

откъдето съответните формули се записват така: $y_{n+1} = y_n + \frac{h\,(k_1 + 4\,k_2 + k_3)}{6}$ при $k_1 = f(t_n,\, y_n)$, $k_2 = f\left(t_n + \frac{h}{2},\, y_n + \frac{h\,k_1}{2}\right)$ и $k_3 = f(t_n + h,\, y_n + 2\,h\,k_2 - h\,k_1)$.

Следват 2 процедури на ***Mathematica,*** наречени **Runge3** и **Runge3graf**. Те изчисляват таблица от значения на приближеното решение и неговата графика, получени по метода на

Рунге-Кута от 3-ти ред.

```
Runge3[f_, h_, ini_, a_, b_] :=
Module[{yrk3, t, rktable3, c},
 c = (b - a) / h;
 yrk3[0] = ini;
 t[n_] := a + n h;
 yrk3[n_] :=
  Module[{k1, k2, k3},
   k1 = f[t[n - 1], yrk3[n - 1]];
   k2 = f[t[n - 1] + h / 2, yrk3[n - 1] + (h / 2) k1];
   k3 = f[t[n - 1] + h, yrk3[n - 1] - h k1 + 2 h k2];
  yrk3[n] = yrk3[n - 1] + h ((1 / 6) k1 + (2 / 3) k2 + (1 / 6) k3)];
 rktable3 = Table[yrk3[i], {i, 0, c}];
Table[{t[i], rktable3[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
Runge3graf[f_, h_, ini_, a_, b_] := Module[
  {yrk3, t, rktable3, c}, c = (b - a)/h; yrk3[0] = ini; t[n_] := a + n h;
  yrk3[n_] := Module[{k1, k2, k3}, k1 = f[t[n - 1], yrk3[n - 1]];
    k2 = f[t[n - 1] + h/2, yrk3[n - 1] + (h k1)/2];
    k3 = f[t[n - 1] + h, yrk3[n - 1] - h k1 + 2 h k2];
    yrk3[n] = yrk3[n - 1] + h (k1/6 + (2 k2)/3 + k3/6)];
  rktable3 = Table[yrk3[i], {i, 0, c}];
  ListPlot[Table[{t[i], rktable3[[i + 1]]}, {i, 0, c}], Joined → True,
   PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable3[[c + 1]]]]
```

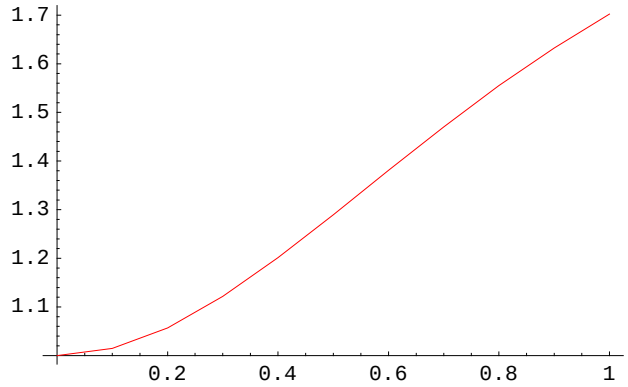
Тук, както обикновено, **f** е функцията от дясната част на ОДУ, **h** е стъпката, **ini** е началното значение, **a** и **b** са краищата на интервала.

Този метод е приложен за решаване на задачата $y' = -t\,y + \frac{4t}{y}$, *y(0)=1 в интервала* [0,1], със стъпка 0.1.

```
f[t_, y_] = -t y + 4 t/y;
```

```
Runge3[f, 0.1, 1, 0, 1]
0       1
0.1     1.01476
0.2     1.05708
0.3     1.12157
0.4     1.20135
0.5     1.28967
0.6     1.38082
0.7     1.47033
0.8     1.55497
0.9     1.63259
1.      1.70187
```

```
Runge3graf[f, 0.1, 1, 0, 1]
```




```
y[1.]=1.70187
```

Най-използваният метод на Рунге-Кута се представя със следната таблица на Бутчер:

0	0			
$\frac{1}{2}$	$\frac{1}{2}$	0		
$\frac{1}{2}$	0	$\frac{1}{2}$	0	
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

и съответните изрази са:

$$y_{n+1} = y_n + \frac{h}{6}\,(K_1 + 2\,K_2 + 2\,K_3 + K_4)$$

$$K_1 = f(t_n,\, y_n)$$

$$K_2 = f\left(t_n + \frac{h}{2},\, y_n + \frac{h}{2}\,K_1\right)$$

$$K_3 = f\left(t_n + \frac{h}{2},\, y_n + \frac{h}{2}\,K_2\right)$$

$$K_4 = f(t_n + h,\, y_n + h\,K_3)$$

Следват две процедури на ***Mathematica***, наречени съответно **Runge4** и **Runge4graf**, с които се изчисляват стойности на приближеното решение и се построява неговата графика за горния метод на Рунге-Кута от 4-ти ред:

```
Runge4[f_, h_, ini_, a_, b_] :=
Module[{yrk4, t, rktable4, c},
 c = (b - a) / h;
 yrk4[0] = ini;
 t[n_] := a + n h;
 yrk4[n_] :=
  Module[{k1, k2, k3, k4},
   k1 = f[t[n - 1], yrk4[n - 1]];
   k2 = f[t[n - 1] + h / 2, yrk4[n - 1] + (h / 2) k1];
   k3 = f[t[n - 1] + h / 2, yrk4[n - 1] + (h / 2) k2];
   k4 = f[t[n - 1] + h, yrk4[n - 1] + h k3];
   yrk4[n] =
    yrk4[n - 1] +
     (h / 6) (k1 + 2 k2 + 2 k3 + k4)];
 rktable4 = Table[yrk4[i], {i, 0, c}];
Table[{t[i], rktable4[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
Runge4graf[f_, h_, ini_, a_, b_] := Module[
  {yrk4, t, rktable4, c}, c = (b - a)/h; yrk4[0] = ini; t[n_] := a + n h;
  yrk4[n_] := Module[{k1, k2, k3, k4}, k1 = f[t[n - 1], yrk4[n - 1]];
    k2 = f[t[n - 1] + h/2, yrk4[n - 1] + (h k1)/2];
    k3 = f[t[n - 1] + h/2, yrk4[n - 1] + (h k2)/2];
    k4 = f[t[n - 1] + h, yrk4[n - 1] + h k3];
    yrk4[n] = yrk4[n - 1] + 1/6 h (k1 + 2 k2 + 2 k3 + k4)];
  rktable4 = Table[yrk4[i], {i, 0, c}];
  ListPlot[Table[{t[i], rktable4[[i + 1]]}, {i, 0, c}], Joined → True,
   PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable4[[c + 1]]]]
```

Тук **f** е функцията от дясната част на ОДУ, **h** е стъпката, **ini** е началното значение, **a** и **b** са краищата на интервала.

По-надолу този метод е използван за решаване на началната задача: $y' = -t\,y + \frac{4t}{y}$, $y(0)=1$

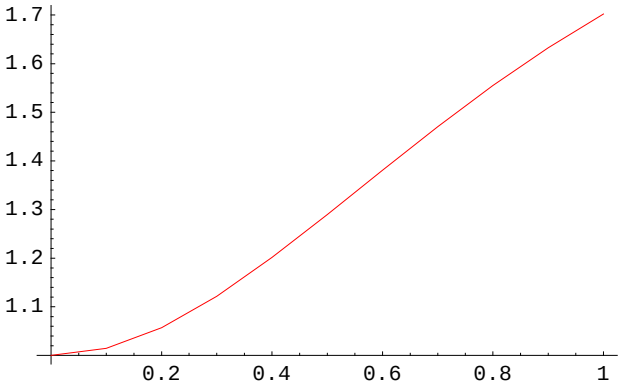
в интервала [0,1], със стъпка 0.1.

```
f[t_, y_] = -t y + 4 t / y;
```

```
Runge4[f, 0.1, 1, 0, 1]
```

```
0       1
0.1     1.01482
0.2     1.05718
0.3     1.1217
0.4     1.20149
0.5     1.28981
0.6     1.38093
0.7     1.47042
0.8     1.55503
0.9     1.63261
1.      1.70187
```

```
Runge4graf[f, 0.1, 1, 0, 1]
```




```
y[1.]=1.70187
```

Пример 1.

С помощта на метода на Рунге-Кута от 4-ти ред да се реши началната диференциална задача: $y' = \frac{y^2-3t^2-2ty}{t^2+2ty}$, *y(1)=2, в интервала [1,2] със стъпка h=0.1*

Решение

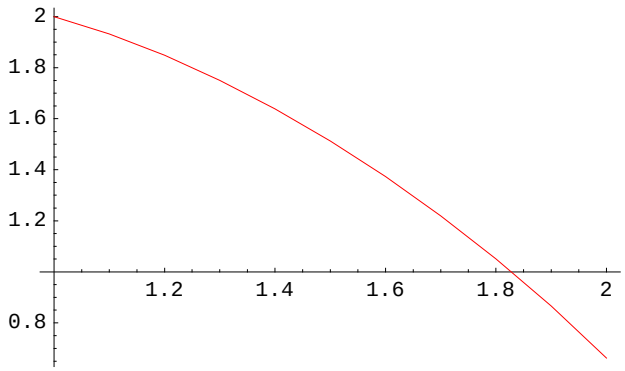
Декларираме функцията и програмираме числения метод:

```
f[t_, y_] := (y^2 - 3 t^2 - 2 t y) / (t^2 + 2 t y)
```

```
Runge4[f, 0.1, 2, 1, 2]
```

```
1       2
1.1     1.93191
1.2     1.84842
1.3     1.75041
1.4     1.63842
1.5     1.5127
1.6     1.37319
1.7     1.21949
1.8     1.05082
1.9     0.865842
2.      0.662386
```

```
Runge4graf[f, 0.1, 2, 1, 2]
```




```
y[2.]=0.662386
```

Пример 2.

Решете началната задача $y' = \sqrt{y} - \frac{20\, e^{-100\,(t-2)^2}}{\sqrt{\pi}}$, $y(1) = 1$, *като получите приближената стойност за* $t=3$, *с помощта на процедурата* *Runge4* *и стъпка* $h=0.01$. *Анализирайте графиката и ч сравнете с тази на задачата* $y' = \sqrt{y}$, $y(1) = 1$.

Решение

Важно е да се види, че ***Mathematica*** не може да реши точно тази задача, например така:

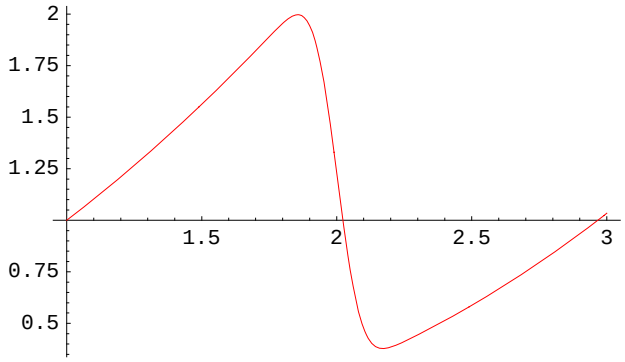
```
DSolve[{y'[t] == Sqrt[y[t]] - 20 E^(-100 (t-2)^2)/Sqrt[π], y[1] == 1}, y[t], t]
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so some solutions
   may not be found; use Reduce for
   complete solution information. More…
```

```
DSolve[{y'[t] == -(20 E^(-100 (-2+t)^2)/Sqrt[π]) + Sqrt[y[t]], y[1] == 1}, y[t], t]
```

Следва дефиниране на функцията от дясната страна на уравнението и прилагането на числения метод

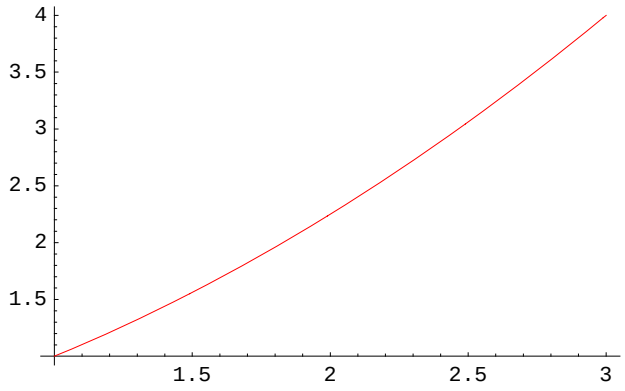
```
f[t_, y_] := Sqrt[y] - 20 E^(-100 (t-2)^2)/Sqrt[Pi]
Runge4graf[f, 0.01, 1, 1, 3]
```




```
y[3.]=1.03349
```

По-долу решаваме втората диференциална задача със същата процедура

```
f[t_, y_] := Sqrt[y]
Runge4graf[f, 0.01, 1, 1, 3]
```




```
y[3.]=4.
```

Графиките съвпадат малко преди достигане до точката $t = 2$, след което бързо се разхождат. Рязкото спадане на решението на първото уравнение е защото неговото поведение се модифицира от импулса $\frac{20\, e^{-100\,(t-2)^2}}{\sqrt{\pi}}$. Импулсът се представя както следва:

$$\text{Plot}\left[\frac{20\, e^{-100\,(t-2)^2}}{\sqrt{\pi}},\ \{t,\ 1,\ 4\},\ \text{PlotRange} \to \text{All}\right]$$