2.6 Методи на Рунге–Кута

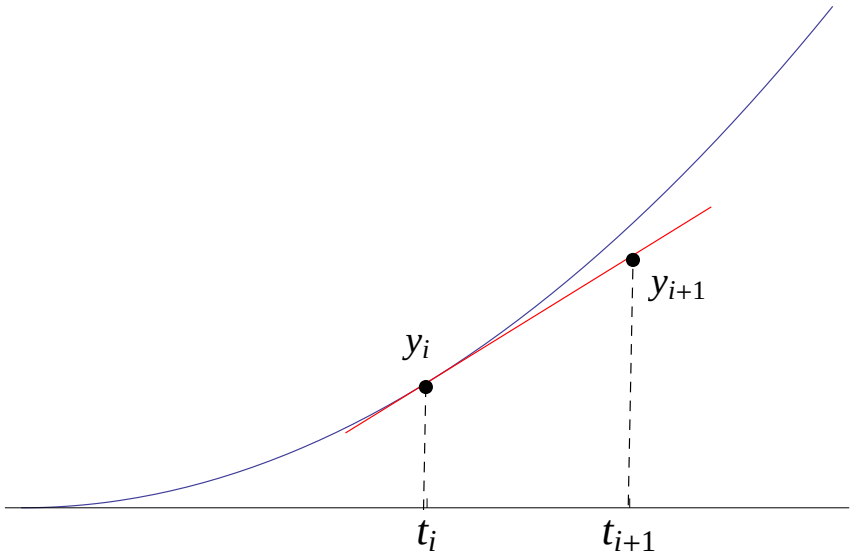
Основен недостатък на методите на Ойлер е, че те са бавно сходящи – да припомним, че те имат първи ред на сходимост, тъй като локалната грешка на апроксимация е $O(h)$. Това означава, че ако искаме да получим висока точност, трябва да работим с много малка стъпка, т.е. да правим голям брой операции, което, разбира се, е нецелесъобразно. Сега ще разгледаме първия от двата основни класа методи, които се използват, когато е необходим по-висок ред на сходимост.

2.6.1 Явни методи на Рунге–Кута

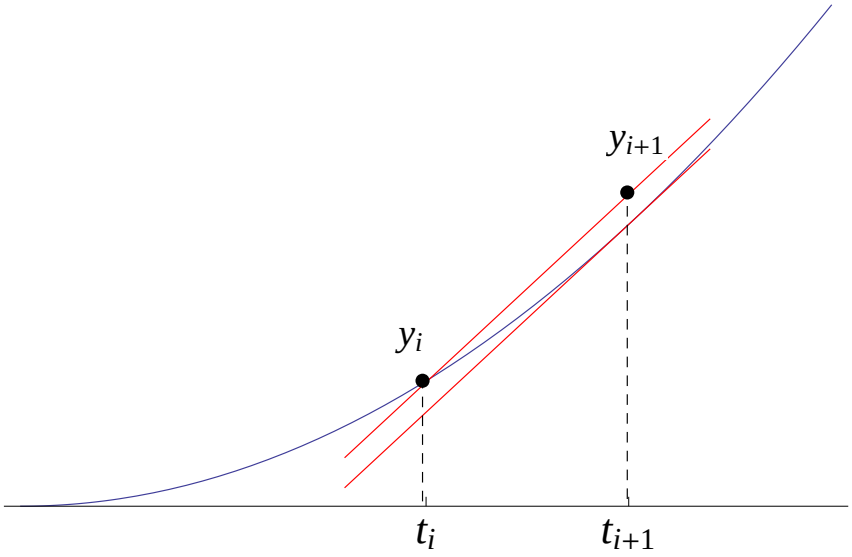
Явните методи на Рунге–Кута са най-често използваният клас методи за решаване на ОДУ. За да мотивираме тяхното използване, нека първо разгледаме от геометрична гледна точка методите на Ойлер. При явния метод на Ойлер имаме

$$y_{i+1} = y_i + hf(t_i, y_i),$$

т.е. правим една стъпка по направление на допирателната в точката t_i, за да получим приближението в точката t_{i+1}:



При неявния метод на Ойлер използваме направлението, зададено от допирателната в точката t_{i+1}:

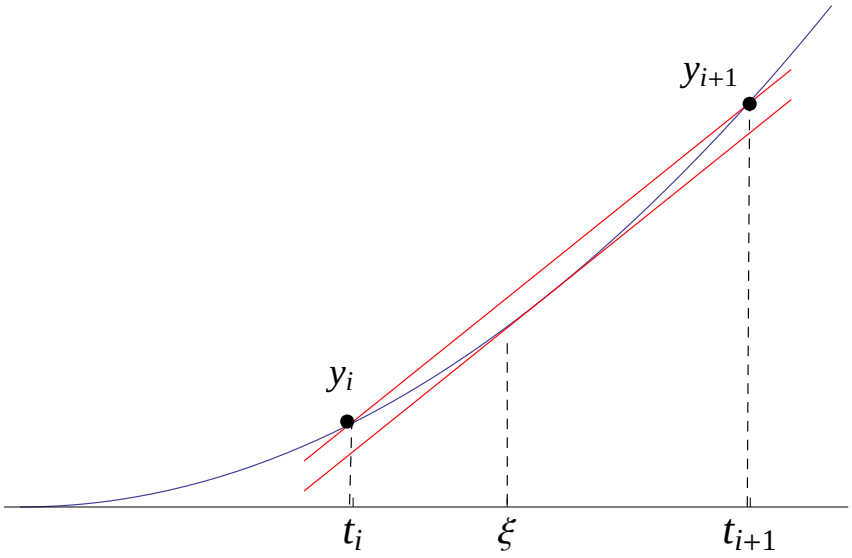


ЛГА за явния и за неявния метод на Ойлер е $O(h)$.

От друга страна, според Теоремата за крайните нараствания, съществува ξ, така че

$$u(t_i + h) - u(t_i) = u'(\xi)(t_{i+1} - t_i) = u'(\xi)h.$$

С други думи, ако използваме направлението, зададено от допирателната в тази точка, ЛГА би била 0:



Ние, разбира се, няма как да намерим точката ξ, но нейното съществуване ни дава основание да вземем производните в различни точки в интервала $[t_i, t_{i+1}]$ и да ги усредним с някакви тегла, така че да получим стойност, близка (в някакъв смисъл) до стойността в ξ. Нека означим стойностите на производните в тези точки (умножени по h) с $k_1, k_2, \ldots, k_s$. Общият вид на s-етапните методи на Рунге-Кута е

$$\frac{y_{i+1} - y_i}{h} = \frac{1}{h}(p_1 k_1 + p_2 k_2 + \cdots + p_s k_s),$$

където $p_1, p_2, \ldots, p_s$ са теглата, с които вземаме всяка от производните. Ще ги определим така, че ЛГА да е възможно най-малка.

k_1 е производната в точката t_i. За нея имаме

$$k_1 = hf(t_i, y_i).$$

k_2 е производната в някоя друга точка $t_i + \alpha_2 h$. За да я пресметнем обаче (т.е. да пресметнем дясната страна в диференциалното уравнение), ни е необходима стойността на решението в тази точка, което ни е неизвестно. Ще го апроксимираме на база на известната информация, като за неговата стойност вземем $y_i + \beta_{2,1} k_1$. Тогава

$$k_2 = hf(t_i + \alpha_2 h, y_i + \beta_{2,1} k_1).$$

Параметрите α_2, $\beta_{2,1}$ отново подлежат на определяне, така че ЛГА да е възможно най-малка.

Разсъждавайки аналогично и за производните в следващите точки, получаваме следния общ вид на методите на Рунге-Кута:

$$\begin{aligned}
&\frac{y_{i+1} - y_i}{h} = \frac{1}{h}(p_1 k_1 + p_2 k_2 + \cdots + p_s k_s),\\
&k_1 = hf(t_i, y_i)\\
&k_2 = hf(t_i + \alpha_2 h, y_i + \beta_{2,1} k_1)\\
&k_3 = hf(t_i + \alpha_3 h, y_i + \beta_{3,1} k_1 + \beta_{3,2} k_2)\\
&k_j = hf(t_i + \alpha_j h, y_i + \beta_{j,1} k_1 + \cdots + \beta_{j,j-1} k_{j-1}), \quad j = 4, \ldots, s.
\end{aligned}$$

Да обърнем внимание, че методите на Рунге-Кута са едностъпкови методи, тъй като при намирането на y_{i+1} използват само стойността на y_i, но не и на стойностите на приближеното решение в предходните точки.

В литературата могат да се открият коефициентите на различни методи от този тип с различен ред на сходимост. Често коефициентите се записват в т.нар. Таблица на Butcher:

$$\begin{array}{c|cccc}
0 & & & & \\
\alpha_2 & \beta_{2,1} & & & \\
\vdots & \vdots & \ddots & & \\
\alpha_s & \beta_{s,1} & \cdots & \beta_{s,s-1} & \\
\hline
 & p_1 & p_2 & \cdots & p_s
\end{array}$$

Нека разгледаме най-използвания метод за решаване на ОДУ при равномерна мрежа, а именно четириетапният метод на Рунге-Кута, който има следната таблица на Butcher:

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

Записан подробно, методът има следния вид:

$$\begin{aligned}
&\frac{y_{i+1} - y_i}{h} = \frac{1}{h}(p_1 k_1 + p_2 k_2 + p_3 k_3 + p_4 k_4),\\
&k_1 = hf(t_i, y_i),\\
&k_2 = hf(t_i + 1/2h, y_i + 1/2k_1),\\
&k_3 = hf(t_i + 1/2h, y_i + 1/2k_2),\\
&k_4 = hf(t_i + h, y_i + k_3).
\end{aligned}$$

Прилагаме функция в Mathematica, която реализира метода:

```
In[ ]:= RK4[f_, t0_, T_, u0_, h_] := (
        n = Ceiling[(T - t0) / h];
         tArr = Table[t0 + i * h, {i, 0, n}];
        yArr = Table[0, {i, 0, n}];
        yArr[[1]] = u0;


        For[i = 1, i ≤ n, i++,
        k1 = h * f[tArr[[i]], yArr[[i]]];
        k2 = h * f[tArr[[i]] + h / 2, yArr[[i]] + k1 / 2];
        k3 = h * f[tArr[[i]] + h / 2, yArr[[i]] + k2 / 2];
        k4 = h * f[tArr[[i]] + h, yArr[[i]] + k3];
        yArr[[i + 1]] = yArr[[i]] + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4);
        ];
        Transpose[{tArr, yArr}]
        )
```

Нека да приложим имплементираните методи върху задачата:

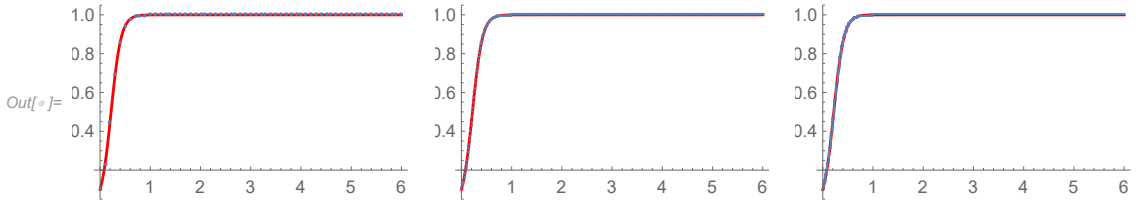
$$\begin{aligned} \frac{du}{dt} &= 10u(1-u),\ 0 < t \leq 6 \\ u(0) &= 0.1. \end{aligned} \tag{2.1}$$

със стъпка h=0.1, 0.01, 0.001. Кодът за стъпка h=0.1 изглежда така:

```
In[ ]:= f[t_, u_] = 10 * u * (1 - u);
        exact[t_] = DSolve[{u'[t] == 10 u[t] (1 - u[t]), u[0] == 0.1}, u[t], t][[1, 1, 2]];
        plotExact = Plot[exact[t], {t, 0, 6}, PlotRange → All, PlotStyle → Red];
        apprRK4 = RK4[f, 0, 6, 0.1, 0.1];
        plotApprRK4 = ListPlot[apprRK4, PlotRange → All];
        Show[plotExact, plotApprRK4]
        ListLinePlot[Abs[exact[apprRK4[[All, 1]]] - apprRK4[[All, 2]]], PlotRange → All]
```

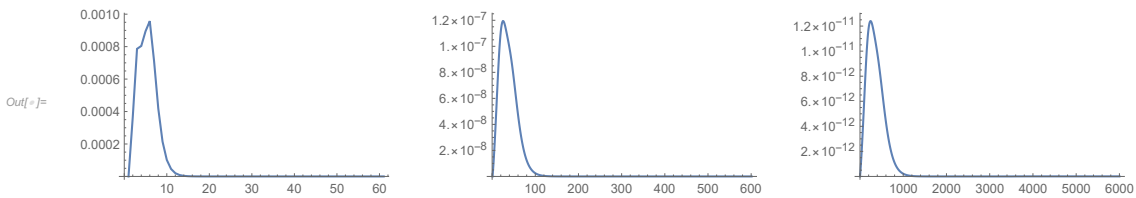
Графика с приближеното и точното решение при h=0.1, h=0.01 и h=0.001, прилагаме по-долу:

Out[ ]=



Графика на абсолютните грешки можете да намерите по-долу:

Out[ ]=



Ако намалим стъпката с 10, тогава очакваме грешката да намалее с 10^4, тъй като четириетапният метод на Рунге–Кута има четвърти ред на сходимост.

Нека да разгледаме триетапния метод на Рунге–Кута, зададен с таблицата на Butcher:

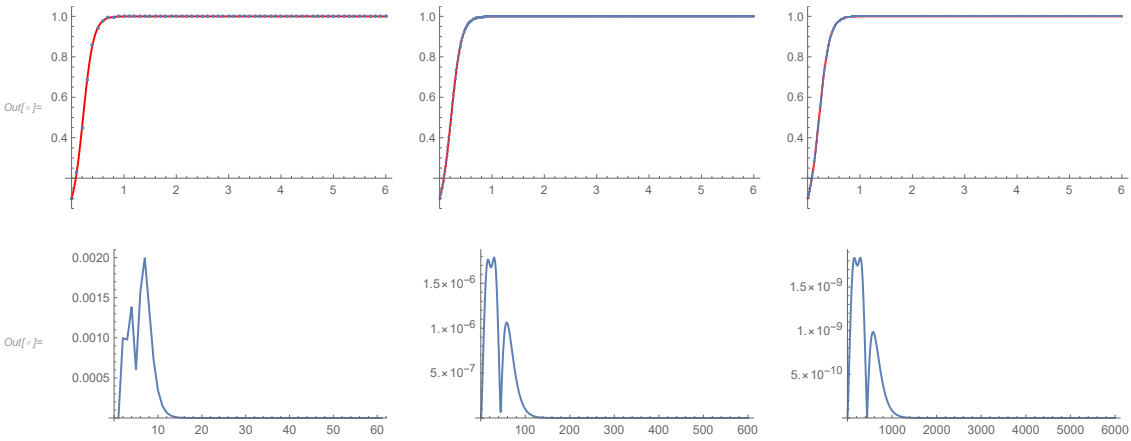
$$\begin{array}{c|ccc} 0 & & & \\ 1/3 & 1/3 & & \\ 2/3 & 0 & 2/3 & \\ \hline & 1/4 & 0 & 3/4 \end{array}$$

Този метод има ЛГА $O(h^3)$. Имплементация на метод можете да намерите по-долу:

```
In[ ]:= RK3[f_, t0_, T_, u0_, h_] := (
         n = Floor[(T - t0) / h];
         tArr = Table[t0 + i * h, {i, 0, n}];
         yArr = Table[0, {i, 0, n}];
         yArr[[1]] = u0;

         For[i = 1, i ≤ n, i++,
          k1 = h * f[tArr[[i]], yArr[[i]]];
          k2 = h * f[tArr[[i]] + h / 3, yArr[[i]] + k1 / 3];
          k3 = h * f[tArr[[i]] + h * 2 / 3, yArr[[i]] + k2 * 2 / 3];
          yArr[[i + 1]] = yArr[[i]] + 1 / 4 * k1 + 3 / 4 * k3;
         ];
         Transpose[{tArr, yArr}]
         )
```

Като го приложим върху задачата (2.1), получаваме следните резултати:



Тъй като ЛГА на триетапен метод на Рунге–Кута е $O(h^3)$, то като намалим стъпката 10 пъти, грешката намалява 10^3 пъти.

> Повече информация за начина, по който се избират коефициентите в методите на Рунге–Кута може да бъде намерена в [3] и [1].

Библиография

[1] J. Butcher, Numerical Methods for Ordinary Differential Equations. Wiley, 2nd edition, 2009.

[2] E. Coddington, An Introduction to Ordinary Differential Equations. Dover Publications, Unabridged edition, 1989.

[3] С. Димова, Т. Черногорова, А. Йотова, Числени методи за диференциални уравнения. Университетско издателство "Св. Климент Охридски", София, 2010.

[4] J. Hale, Ordinary Differential Equations. Dover Publications, 2009.

[5] M. Hirsh, S. Smale, R. Devaney, Differential Equations, Dynamical Systems, and an Introduction to Chaos. Academic Press, 3rd edition, 2012.

[6] M.H. Holmes, Introduction to Numerical Methods in Differential Equations. Springer, 2007.

[7] J. Murray, Mathematical Biology I. An Introduction. Springer, 3rd edition, 2002.

[8] M. Tenenbaum, H. Pollard, Ordinary Differential Equations. Dover Publications, Revised ed., 1985.