Числени методи за обикновени диференциални уравнения (ОДУ): Някои едностъпкови методи

Въведение:

В тази глава ще покажем как се използва ***Mathematica*** при изучаването на различни едностъпкови числени методи за решаване на обикновени диференциални уравнения (ОДУ) По-специално, ще покажем как се намира приближено решение на начална задача за ОДУ от вида $y' = f(t, y)$, $y(t_0) = y_0$, като се получават стойностите на решението в равноотдалечени точки в интервала $[t_0, t_N]$, т.е. $t_n = t_0 + n\,h$, to $n = 0, \ldots, N$. Разстоянието $h = \frac{t_N - t_0}{N}$ се нарича **дължина на стъпката**. Ще означаваме с y_n приближената стойност на $y(t_n)$ и ще записваме $f_n = f(t_n, y_n)$.

1. Използване на фунцията NDSolve.

Добре е известно, че ОДУ не винаги могат да се интегрират точно. Ако ***Mathematica*** не може да реши точно едно ОДУ, резултатът (Output) от командата **`DSolve`** има същият вид, както инструкцията за вход Input, например (Изпълняваме командата с едновременно натискане на клавишите **Shift**+**Enter**)

```
ecu = DSolve[y'[t] == 1 + y[t]^2 - t^3, y[t], t]
```

```
DSolve[y'[t] == 1 - t^3 + y[t]^2, y[t], t]
```

Когато имаме подобен случай, ОДУ може да се реши приближено, при условие, че ОДУ има само едно решение, т.е. задачата е начална. Съответната инструкция за получаване на приближено решение на дадено ОДУ е **`NDSolve`** като се използва синтаксисът:

```
NDSolve[{equation, initials conditions}, function, {variable , interval}]
```

В общия случай, **`NDSolve[{equation,initials conditions},x[t],{t,a,b}]`** дава приближение на решението *x(t)* на началната задача в интервал $[a, b]$. Например, ако се търси решението на началната задача $y' = 1 + y^2 - t^3$, $y(0) = 0$ в интервала [0, 4], съответната инструкция е:

```
solution = NDSolve[{y'[t] == 1 + y[t]^2 - t^3, y[0] == 0}, y[t], {t, 0, 4}]
```

```
{{y[t] → InterpolatingFunction[{{0., 4.}}, <>][t]}}
```

Както се вижда, изходът от командата **`NDSolve`** се присвоява на функция, дефинирана като интерполирана функция. Следователно, изразът може да се преобразува в таблица, за да се начертае полученото приближено решение. Това постигаме по следния начин:

```
yapprox[t_] = y[t] /. solution[[1]]
```

```
InterpolatingFunction[{{0., 4.}}, <>][t]
```

Сега всяка стойност от функцията в произволна точка от разглеждания интервал може да се изчисли така:

```
yapprox[1]
```

```
1.16197
```

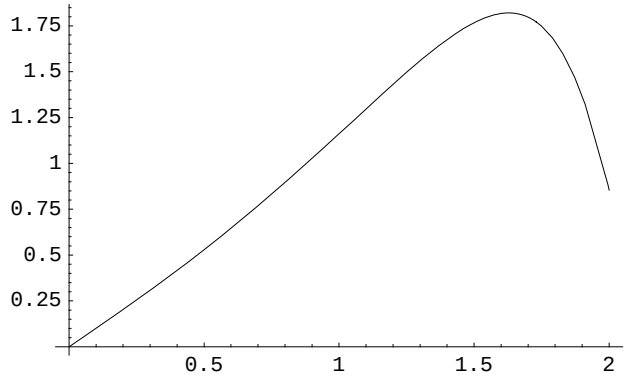
Като се използва стъпка $h = 0.1$ може да се получи таблица на решението в равноотстоящи точки в интевала [0, 1]:

```
Table[{t, yapprox[t]}, {t, 0, 1, 0.1}] // TableForm // Chop
```

```
0       0
0.1     0.10031
0.2     0.202305
0.3     0.307248
0.4     0.416024
0.5     0.529206
0.6     0.647092
0.7     0.769722
0.8     0.896859
0.9     1.02794
1.      1.16197
```

и също да че представи графично с функцията **Plot**. Например, графиката може да се начертае в интервала [0, 2] :

```
Plot[yapprox[t], {t, 0, 2}]
```



ЗАБЕЛЕЖКИ

1.- Полученото решение е определено само в разглеждания интервал. Ако трябва да се изчисли решението извън дадения интервал, *Mathematica* съобщава на потребителя, че се прилага екстраполационен метод:

```
yapprox[6]
```

```
InterpolatingFunction::dmval :
 Input value {6} lies outside the range of data in the interpolating
   function. Extrapolation will be used. More…

-14.5568
```

2.- Ако се извика програмата за помощ Help, може да се види какви различни опции за числени методи за приближаване на решението се предлагат с функцията **NDSolve**. Тук са включени методите на Адамс, Рунге-Кута, Ойлер и др.

Пример 1.

Да се получат приближените стойности на решението на началната задача за ОДУ, y'=y(1-sin t), y(0)=1, в интервала [0,1], в равноодалечени точки със стъпка 0.05..

Решение

Командата **NDSolve** се използва за директно реваване на задачата.

```
sol11 = NDSolve[{y'[t] == y[t] (1 - Sin[t]), y[0] == 1}, y[t], {t, 0, 1}]
```

```
{{y[t] → InterpolatingFunction[{{0., 1.}}, <>][t]}}
```

Определена е функцията на решението и е изчислена търсената таблица:

```
yap11[t_] = y[t] /. sol11[[1]]
```

```
InterpolatingFunction[{{0., 1.}}, <>][t]
```

```
Table[{t, yap11[t]}, {t, 0, 1, 0.05}] // TableForm
```

```
0       1.
0.05    1.04996
0.1     1.09966
0.15    1.14886
0.2     1.1973
0.25    1.24472
0.3     1.2909
0.35    1.33559
0.4     1.37859
0.45    1.4197
0.5     1.45875
0.55    1.4956
0.6     1.5301
0.65    1.56218
0.7     1.59176
0.75    1.61881
0.8     1.6433
0.85    1.66526
0.9     1.68474
0.95    1.70179
1.      1.71653
```

2. Метод на Ойлер.

Методът на Ойлер е най-простият едностъпков метод. В стандартни означения алгоритъмът на Ойлер за приближено намиране на решението на началната задача за ОДУ $y' = f(t, y)$ при $y(t_0) = y_0$ в интервала $[t_0 = a, b]$, се дава с израза:

$$y_{n+1} = y_n + h\, f(t_n, y_n) = y_n + h\, f_n \qquad n = 0, \ldots, m-1$$

където $h = \dfrac{b-a}{m}$ е дължината на стъпката и $t_n = t_{n-1} + h = a + n\,h$.

В съответствие с общата теория на числените методи, характеристичното уравнение е $p(x) = x - 1$, така че е лесно да се види, че методът е сходящ, тъй като е устойчив и апроксимира диференциалната задача. Последният израз може да се кодира с *Mathematica* както следва:

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

където e функция, описваща дясната част на ОДУ, разрешено отнисно първата производна, **h** е стъпката, **ini** е стойността на началното условие, **a** и **b** са границите на интервала. Изходът (Output) от изпълнението на програмата ще даде приближеното решение на задачата.

По-нататък, да приложим предишната процедура за да решим приближено началната задача $y' = -t\,y + \frac{4t}{y}$, *y(0)=1* в интервала [0,1], със стъпка 0.1. Съответната функция **f** се дефинира така:

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

и предишният алгоритъм се прилага като зададем услвията $h = 0.1,\ ini = 1,\ a = 0,\ b = 1$:

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

```
0                       1
0.1`                    1
0.2`                    1.03`
0.30000000000000004`    1.0870699029126214`
0.4`                    1.1648462911273196`
0.5`                    1.255609618037652`
0.6000000000000001`     1.3521143139350462`
0.7000000000000001`     1.4484872400849704`
0.8`                    1.5403982534425766`
0.9`                    1.6249048785133102`
1.`                     1.7002148697864552`
```

За да изчислим точността на апроксимацията ще сравним за този пример полученото решение по метода на Ойлер с точното решение, намерено с командата **DSolve:**

```
exactsol = DSolve[{y'[t] == -t y[t] + 4 t / y[t], y[0] == 1}, y[t], t]
```

$$\left\{\left\{y[t] \to e^{-\frac{t^2}{2}} \sqrt{-3 + 4\, e^{t^2}}\right\}\right\}$$

Получаваме функцията на решението и изчисляваме съответната таблица

```
yex[t_] = exactsol[[1, 1, 2]]
```

$$e^{-\frac{t^2}{2}} \sqrt{-3 + 4\, e^{t^2}}$$

```
Table[{t, yex[t]}, {t, 0, 1, 0.1}] // TableForm
```

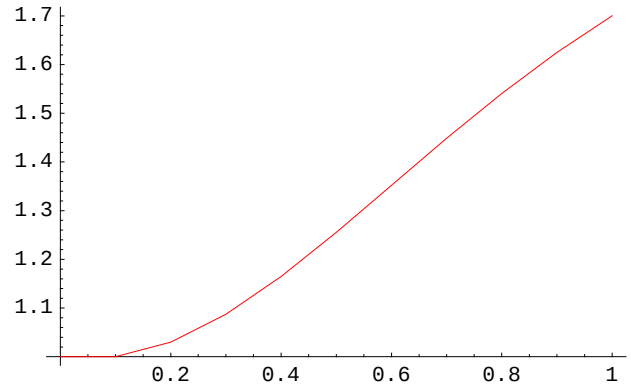
```
0      1
0.1    1.01482
0.2    1.05718
0.3    1.1217
0.4    1.20149
0.5    1.28981
0.6    1.38093
0.7    1.47042
0.8    1.55503
0.9    1.63261
1.     1.70187
```

Сравняването на решенията може да се види и графично. Следва процедура, програмирана на ***Mathematica,*** показваща в червено решението, получено по метода на Ойлер. Новата процедура **grafeuler** има същият вход и значения на по-горе използваните параметри.

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

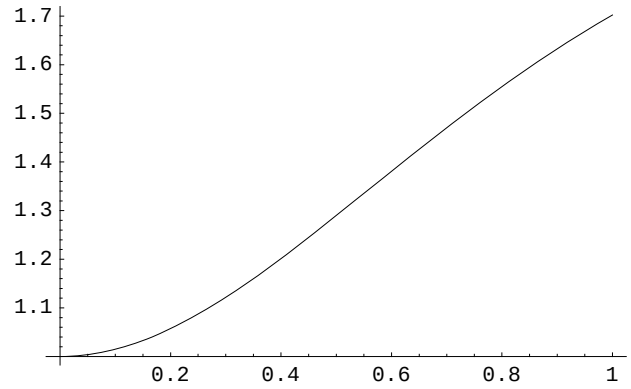
Приближеното и точното решение са изобразени графично, за да ги сравним.

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

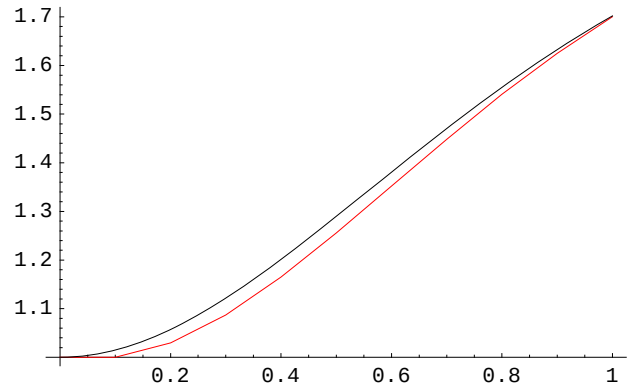



```
aproxi = %;
```

```
exact = Plot[yex[t], {t, 0, 1}]
```




```
Show[aproxi, exact]
```



Пример 2.

С помощта на метода на Ойлер намерете приближено решение на началната задача за ОДУ y' = 3 (y + t), y(0) = 1, в интервала [0,1], в равноодалечени точки със стъпка 0.1. Представете графично полученото решение и го сравнетие с точното, като използвате командата ***DSolve****. Коментирайте лошото поведение на приближеното решение и получете подобрено решение с намаляване на стъпката. .*

Решение

Намираме грубото приближено решение и точното решение с **DSolve**:

```
f[t_, y_] := 3 (y + t)
```

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

```
0       1
0.1     1.3
0.2     1.72
0.3     2.296
0.4     3.0748
0.5     4.11724
0.6     5.50241
0.7     7.33314
0.8     9.74308
0.9     12.906
1.      17.0478
```

```
solexacta22 = DSolve[{y'[t] == 3 (y[t] + t), y[0] == 1}, y[t], t]
```

$$\left\{\left\{y[t] \to \frac{1}{3}\left(-1+4\, e^{3\, t}-3\, t\right)\right\}\right\}$$

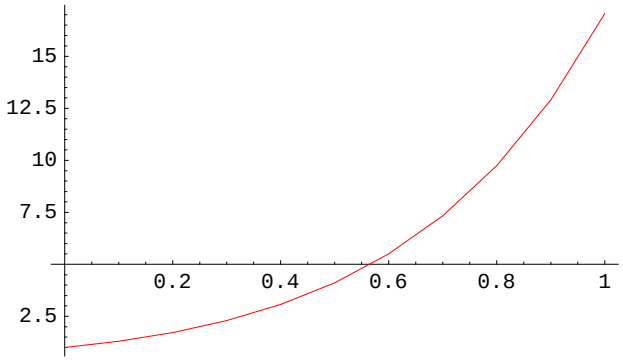
Намерената функция на решението е определена с

```
yex22[t_] = solexacta22[[1, 1, 2]]
```

$$\frac{1}{3}\left(-1+4\, e^{3\, t}-3\, t\right)$$

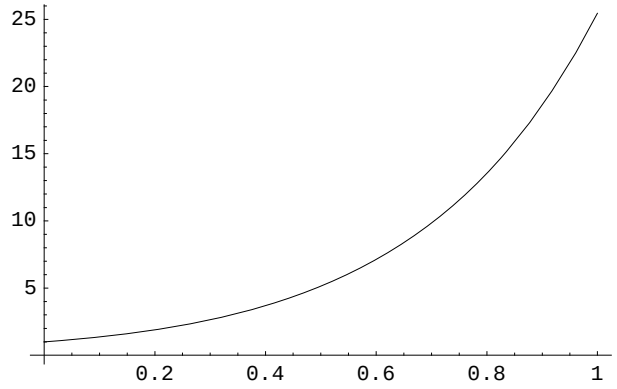
Както по-горе получаваме графиките на двете решения:

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

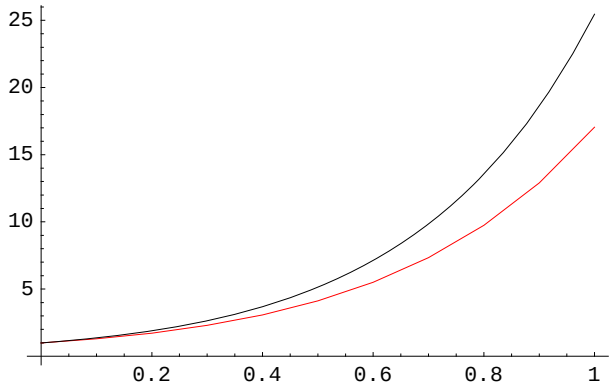



```
aproximada22 = %;
```

```
exacta22 = Plot[yex22[t], {t, 0, 1}]
```

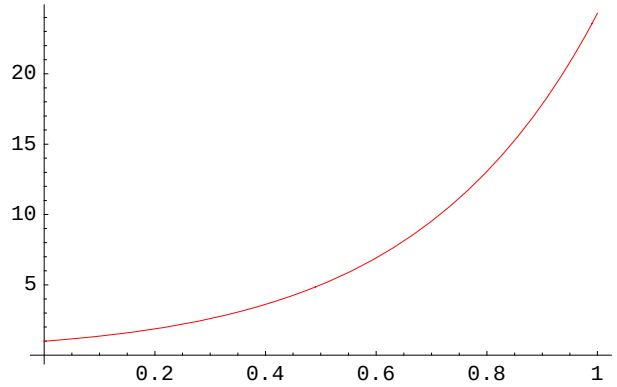



```
Show[aproximada22, exacta22]
```



Както се вижда от графиките, приближението, изчислено по метода на Ойлер не е добро. Това е така, защото точното решение нараства прекалено бързо, тъй като съдържа експоненциален член. По-добро приближение ще получим с намаляване на стъпката, например до 0.01:

```
grafeuler[f, 0.01, 1, 0, 1]
```




```
aproximada22bis = %;
```

```
Show[aproximada22bis, exacta22]
```