

**Міністерство освіти і науки України**  
Харківський національний університет імені В.Н. Каразіна

**Розрахунково-графічне завдання #1**  
**Інтегрування диференціальних рівнянь**

**Виконав:**  
Захаров Дмитро Олегович  
Група МП-41

Харків – 2025

# Зміст

<b>1</b>	<b>Постановка задачі</b>	<b>2</b>
<b>2</b>	<b>Опис методів</b>	<b>3</b>
2.1	Метод Ейлера . . . . .	3
2.2	Метод Рунге-Кутти . . . . .	3
<b>3</b>	<b>Код програми</b>	<b>4</b>

# 1 Постановка задачі

Знайти розв'язок задачі Коші

$$\begin{cases} y' = f(x, y), & x \in [x_0, b], \\ y(x_0) = y_0 \end{cases}$$

в точках  $x_i = x_0 + ih$  для  $i \in [n]$ ,  $h = (b - x_0)/n$  з точністю  $\varepsilon = 10^{-6}$  за допомогою:

- Методу Ейлера;
- Методу Рунге-Кутти, використовуючи двійний перерахунок і оцінку за Рунге.

На друк вивести результати у вигляді таблиці

- $x_i, y_i := y(x_i)$ ;
- $u_i$  — найближче значення розв'язку у точці  $x_i$ .
- $|y_i - u_i|$ .

**Варіант 5.**

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

Параметри  $x_0 = 1, b = 2, y_0 = 5, n = 10$ .

## 2 Опис методів

### 2.1 Метод Ейлера

Ітеративно застосовуємо формулу:

$$x_{i+1} \leftarrow x_i + h, \quad y_{i+1} \leftarrow y_i + hf(x_i, y_i).$$

Помилка в цьому випадку  $\mathcal{O}(h^2)$ . Покращений метод Ейлера:

$$x_{i+1} \leftarrow x_i + h, \quad y_{i+1} \leftarrow y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y_{i+1})).$$

Цей метод дає помилку  $\mathcal{O}(h^3)$ .

### 2.2 Метод Рунге-Кутти

Ітеративно застосовуємо наступні дії:

$$\begin{aligned} (1) \quad & y_{i+1} \leftarrow y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ (2) \quad & k_1 \leftarrow hf(x_i, y_i), \\ (3) \quad & k_2 \leftarrow hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1), \\ (4) \quad & k_3 \leftarrow hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2), \\ (5) \quad & k_4 \leftarrow hf(x_i + h, y_i + k_3) \end{aligned}$$

Щоб скористатися оцінкою за Рунге, потрібно паралельно до виконання основного методу з кроком  $h$ , виконати його з кроком  $h/2$ . В такому разі, якщо задати точність  $\varepsilon$ , то ми виконуємо перерахунок до тих пір, поки

$$\left| y_j^{(h)} - y_j^{(h/2)} \right| < (2^\rho - 1)\varepsilon,$$

де  $\rho$  – порядок методу. В нашому випадку  $\rho = 4$ , тому

$$\left| y_j^{(h)} - y_j^{(h/2)} \right| < 15\varepsilon.$$

### 3 Код програми

Код програми наведений на мові *Wolfram Mathematica* у файлі, що прикріплений далі.

**Перегорніть на наступну сторінку →**

---

## Section 1. Euler Methods

### ■ Section 1.1. Simple Euler Method

```
In[1]:= f[x_, y_] = -2 y^2 Log[x] - y/x; (* RHE of DE *)
r[x_] = 1/(x (1 + Log[x]^2)); (* Solution to the Cauchy DE *)
x0 = 1;
y0 = 1;
b = 2;
n = 10; (* Parameters used in the problem statement *)
```

**Step 1.** Verify that  $r[x]$  is indeed a solution to  $y' = f(x, y)$  with  $y(1) = 1$

```
In[4]:= Simplify[r[x0] - y0] (* Must be 0 *)
Simplify[r'[x] - f[x, r[x]]] (* Must be 0 *)
```

Out[4]= 0

Out[5]= 0

Alternatively, we can simply solve the equation via DSolve method...

```
In[6]:= DSolve[{y'[x] == f[x, y[x]], y[x0] == y0}, y[x], x]
```

Out[6]=  $\left\{ \left\{ y[x] \rightarrow \frac{1}{x (1 + \log(x)^2)} \right\} \right\}$

**Step 2.** Executing the Euler Method for finding the solution

```
In[7]:= h = (b - x0)/n; (* Step Size *)
xC = x0 + 0.0; yC = y0 + 0.0; (* Current values of x and y *)
simpleEulerMethod = Table[
  m = f[xC, yC];
  y1 = yC + h m;
  x1 = xC + h;
  xC = x1;
  yC = y1;
  {x1, y1}, (* Store (x1, y1) in the list *)
  {i, 1, n}
];
```

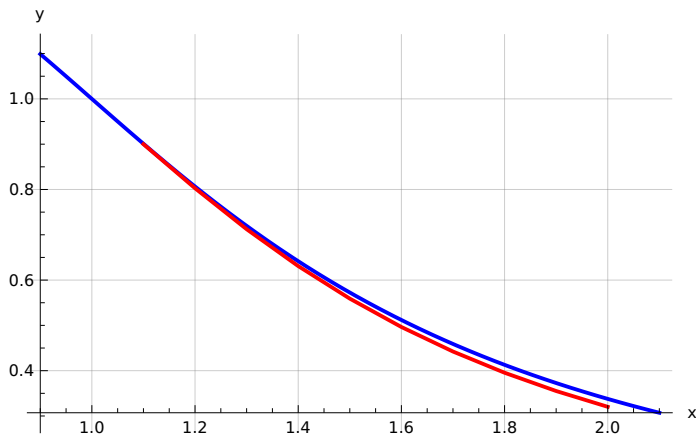
```
In[10]:= simpleEulerMethod
```

Out[10]=  $\{\{1.1, 0.9\}, \{1.2, 0.802742\}, \{1.3, 0.712349\}, \{1.4, 0.630926\}, \{1.5, 0.559072\},$   
 $\{1.6, 0.496454\}, \{1.7, 0.442258\}, \{1.8, 0.395485\}, \{1.9, 0.355127\}, \{2., 0.320246\}\}$

**Step 3.** Validating that the result is correct.

```
In[19]:= Show[
  Plot[r[x], {x, x0-0.1, b+0.1}, PlotStyle -> {Blue, Thick}], (* Function curve *)
  ListPlot[simpleEulerMethod, PlotStyle -> {Red, PointSize[Large]}, Joined -> True],
  (* Numerical points *)
  PlotTheme -> "Detailed",
  AxesLabel -> {"x", "y"},
  PlotLegends -> {"y[x]", "Computed Points"},
  GridLines -> Automatic
]
```

Out[19]=



Plotting differences between real and approximated results

```
In[12]:= t = Table[Abs[simpleEulerMethod[[i]][[2]] - r[simpleEulerMethod[[i]][[1]]], {i, 1, n}]
Max[t]
```

Out[12]=

```
{0.000907042, 0.00378199, 0.00734179, 0.0107169,
 0.0134678, 0.0154619, 0.0167394, 0.0174158, 0.0176243, 0.017488}
```

Out[13]=

```
0.0176243
```

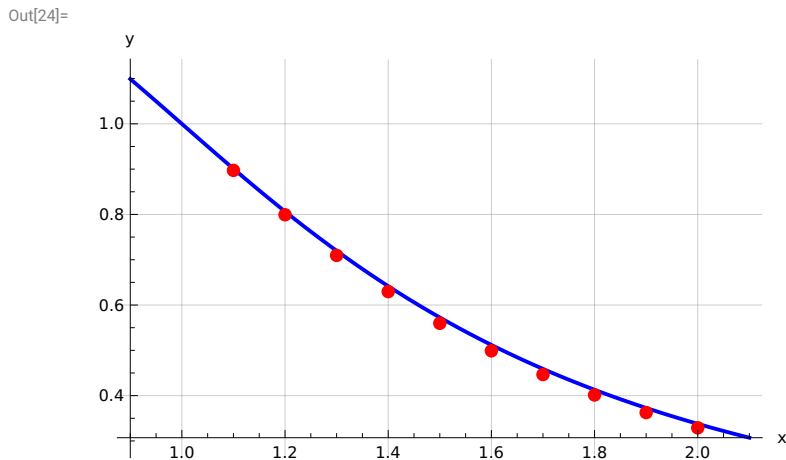
## ■ Section 1.2. Enhanced Euler Method

```

In[20]:= n = 10;
h = (b - x0)/n; (* Step Size *)
xC = x0 + 0.0; yC = y0 + 0.0; (* Current values of x and y *)
enhancedEulerMethod = Table[
  x1 = xC + h;
  y1 = yC + h m;
  y2 = yC + (h/2)(f[xC, yC] + f[x1, y1]);
  xC = x1;
  yC = y2;
  {x1, y2}, (* Store (x1, y1) in the list *)
  {i, 1, n}
];

In[24]:= Show[
  Plot[r[x], {x, x0 - 0.1, b + 0.1}, PlotStyle → {Blue, Thick}], (* Function curve *)
  ListPlot[enhancedEulerMethod, PlotStyle → {Red, PointSize[Large]}, Joined → False],
  (* Numerical points *)
  PlotTheme → "Detailed",
  AxesLabel → {"x", "y"},
  PlotLegends → {"y[x]", "Computed Points"},
  GridLines → Automatic
]

```



Again, plotting differences between real and approximated results. Also print the maximum deviation.



```
In[25]:= t = Table[Abs[enhancedEulerMethod[[i]][[2]] - r[enhancedEulerMethod[[i]][[1]]], {i, 1, n}]  
Max[t]
```

```
Out[25]=
```

```
{0.00365384, 0.00721888, 0.0100709, 0.0119443,  
 0.0128453, 0.0129251, 0.0123824, 0.0114077, 0.0101599, 0.00876047}
```

```
Out[26]=
```

```
0.0129251
```

## Section 2. Runge–Kutta Method

```

In[27]:= h = (b - x0)/n; (* Step Size *)
ε = 10^(-5) + 0.0;
xC = x0; (* x value for step=h *)
yC = y0; (* y value for step=h *)
rungeKutta = {};

rkStep[x_, y_, h_] := Module[{result},
  k1 = h * f[x, y] + 0.0;
  k2 = h * f[x + h/2, y + h * k1/2];
  k3 = h * f[x + h/2, y + h * k2/2];
  k4 = h * f[x + h, y + h * k3];
  Return[y + (k1 + 2 k2 + 2 k3 + k4)/6]
];

While[xC < b,
  (* Store the computed (x, y) points *)
  AppendTo[rungeKutta, {xC, yC}];

  (* Compute RK4 coefficients for h *)
  y1 = rkStep[xC, yC, h];
  y2 = rkStep[xC, yC, h/2];
  y3 = rkStep[xC + h/2, y2, h/2];

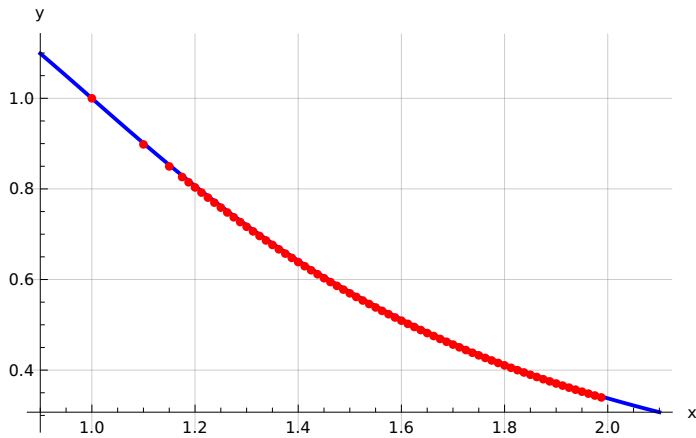
  (* Update x values *)
  xC += h;
  yC = y3;

  If[Abs[y1 - y3] ≥ 15 ε, h = h/2];
  If[h < 10^(-6), Break[]];
]

Show[
  Plot[r[x], {x, x0 - 0.1, b + 0.1}, PlotStyle → {Blue, Thick}], (* Function curve *)
  ListPlot[rungeKutta, PlotStyle → {Red, PointSize[Medium]}], (* Numerical points *)
  PlotTheme → "Detailed",
  AxesLabel → {"x", "y"},
  PlotLegends → {"y[x]", "Computed Points"},
  GridLines → Automatic
]

```

Out[34]=



Printing differences and the maximal difference:

```
In[35]:= t = Table[Abs[rungeKutta[[i]][[2]] - r[rungeKutta[[i]][[1]]], {i, 5, 50}]
Max[t]
```

Out[35]=

```
{0.00331736, 0.00330905, 0.00330025, 0.00329097, 0.00328118, 0.00327089, 0.00326009,
 0.00324878, 0.00323697, 0.00322466, 0.00321184, 0.00319854, 0.00318474,
 0.00317048, 0.00315574, 0.00314055, 0.00312492, 0.00310887, 0.0030924, 0.00307552,
 0.00305827, 0.00304065, 0.00302267, 0.00300436, 0.00298573, 0.0029668, 0.00294758,
 0.00292809, 0.00290836, 0.00288839, 0.0028682, 0.00284781, 0.00282723,
 0.00280648, 0.00278558, 0.00276455, 0.00274339, 0.00272212, 0.00270075,
 0.0026793, 0.00265779, 0.00263622, 0.00261461, 0.00259297, 0.00257131, 0.00254964}
```

Out[36]=

```
0.00331736
```

```

In[37]:= h = 1/80; (* Step Size *)
ε = 10^(-5) + 0.0;
xC = x0; (* x value for step=h *)
yC = y0; (* y value for step=h *)
rungeKutta = {};

rkStep[x_, y_, h_] := Module[{result},
  k1 = h*f[x, y] + 0.0;
  k2 = h*f[x+h/2, y+h*k1/2];
  k3 = h*f[x+h/2, y+h*k2/2];
  k4 = h*f[x+h, y+h*k3];
  Return[y + (k1 + 2 k2 + 2 k3 + k4)/6]
];

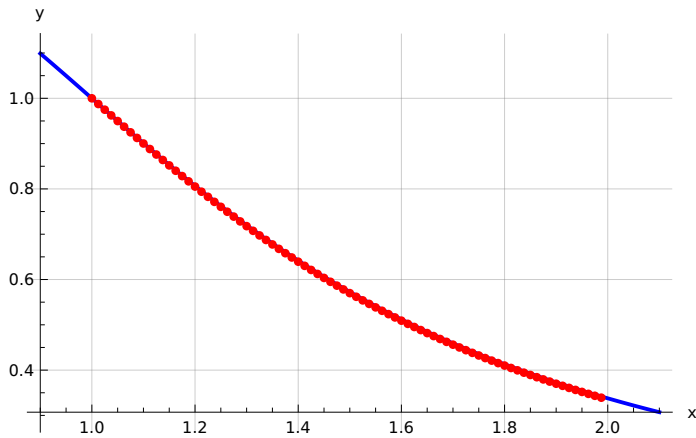
While[xC < b,
  (* Store the computed (x, y) points *)
  AppendTo[rungeKutta, {xC, yC}];

  (* Compute RK4 coefficients for h *)
  y1 = rkStep[xC, yC, h];

  (* Update x and y values *)
  xC += h;
  yC = y1;
]
Show[
  Plot[r[x], {x, x0-0.1, b+0.1}, PlotStyle → {Blue, Thick}], (* Function curve *)
  ListPlot[rungeKutta, PlotStyle → {Red, PointSize[Medium]}], (* Numerical points *)
  PlotTheme → "Detailed",
  AxesLabel → {"x", "y"},
  PlotLegends → {"y[x]", "Computed Points"},
  GridLines → Automatic
]

```

Out[44]=



```
In[45]:= t = Table[Abs[rungeKutta[[i]][[2]] - r[rungeKutta[[i]][[1]]], {i, 1, 80}]
```

```
Max[t]
```

Out[45]=

```
{0, 0.0000790486, 0.000159762, 0.000241852, 0.00032504, 0.000409051, 0.000493622,
 0.000578499, 0.000663439, 0.00074821, 0.000832595, 0.000916388, 0.000999395,
 0.00108144, 0.00116235, 0.00124198, 0.00132019, 0.00139685, 0.00147186, 0.0015451,
 0.00161648, 0.00168594, 0.00175341, 0.00181882, 0.00188213, 0.00194331, 0.00200232,
 0.00205915, 0.00211378, 0.00216621, 0.00221644, 0.00226447, 0.00231031, 0.002354,
 0.00239554, 0.00243496, 0.0024723, 0.00250758, 0.00254085, 0.00257214, 0.00260149,
 0.00262895, 0.00265456, 0.00267836, 0.00270041, 0.00272074, 0.00273942, 0.00275648,
 0.00277198, 0.00278596, 0.00279848, 0.00280958, 0.00281931, 0.00282772,
 0.00283485, 0.00284077, 0.00284549, 0.00284909, 0.00285159, 0.00285305, 0.0028535,
 0.00285299, 0.00285156, 0.00284925, 0.00284609, 0.00284212, 0.00283739,
 0.00283193, 0.00282576, 0.00281893, 0.00281147, 0.00280341, 0.00279477,
 0.0027856, 0.00277591, 0.00276573, 0.0027551, 0.00274404, 0.00273256, 0.0027207}
```

Out[46]=

```
0.0028535
```