ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. О.ГОНЧАРАФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИКАФЕДРА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА МЕТЕМАТИЧНОЇ КІБЕРНЕТИКИ

Лабораторна робота №1
«Методи розв’язання задачі Коші»
з курсу «Методи обчислень (дод. розділи)»

Варіант №9

Виконав:
студент групи ПА-18-2
Лобань Г.М.

Дніпро, 2020

Оглавление

Постановка задачі

Постановка 1

Нехай на відрізку $a \le x \le b$ треба знайти розв'язок диференціального рівняння

$$y'(x) = f(x, y(x)), x \in [a, b] \qquad (1)$$

який у точці $x \in [a,b]$ набуває заданих початкових значень

$$y(x_0) = y_0 \qquad (2)$$

Ця задача називається задачею Коші для рівняння (1). Умова (2) називається початковою через те, що найчастіше точкою x_0 є початок відрізка інтегрування $[a,b]$. Але це не обов'язково. Можливі випадки задачі Коші, коли $x_0 = b$ або x_0 - це якась внутрішня точка відрізка $[a,b]$

Далі вважаємо, що функція $f(x, y)$ є такою, що розв'язок задачі (1), (2) існує і є єдиним. Достатньою умовою існування єдиного розв'язку з класу $C[a,b]$ є неперервність функцій $f(x,y)$ за всіма аргументами і виконання умови Ліпшіца по змінній y .

Постановка 2

Задачу Коші можна формулювати для системи диференційних рівнянь першого порядку.

$$\begin{cases} y'_1(x) = f_1(x, y_1, y_{2,} \dots, y_n), \\ y'_2(x) = f_2(x, y_1, y_{2,\dots}, y_n), \\ \dots, \\ y'_n(x) = f_n(x, y_1, y_{2,\dots}, y_n) \end{cases} \qquad (3)$$

при таких початкових умовах

$$y_i(x_0) = y_{i0}, i = \overline{1,n} \qquad (4)$$

Задачу (3), (4) можна звести до задачі вигляду (1), (2), якщо впровадити такі векторні функції:

$$Y(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \dots \\ y_n(x) \end{bmatrix}, Y'(x) = \begin{bmatrix} y'_1(x) \\ y'_2(x) \\ \dots \\ y'_n(x) \end{bmatrix}, F(x, Y(x)) = \begin{bmatrix} f_1(x, y_1, y_{2,\dots}, y_n) \\ f_2(x, y_1, y_{2,\dots}, y_n) \\ \dots \\ f_n(x, y_1, y_{2,\dots}, y_n) \end{bmatrix}, Y_0 = \begin{bmatrix} y_{10} \\ y_{20} \\ \dots \\ y_{n0} \end{bmatrix}$$

Тепер задачу (3), (4) можна записати у векторному вигляді

$$Y'(x) = F(x, Y(x)), x \in [a, b], Y(x_0) = Y_0 \ .$$

Отже, задача (3), (4) зведена до вигляду (1), (2).

Постановка 3

Задачу Коші можна сформулювати для диференціального рівняння -го порядку

$$y^{(n)}=f\left(x,y(x),y'(x),\dots,y^{(n-1)}(x)\right),x\in[a,b] \tag{5}$$

$$\begin{cases} y(x_0)=y_{00}, \\ y'(x_0)=y_{10} \\ \dots, \\ y^{(n-1)}(x_0)=y_{n-1,0} \end{cases} \tag{6}$$

Задача (5), (6) може бути зведена до вигляду (3), (4), якщо впровадити такі позначення

$$\begin{cases} y_1(x)=y'(x), \\ y_2(x)=y_1'(x)=y''(x), \\ \dots, \\ y_{n-1}(x)=y_{n-2}'(x) \end{cases} \tag{7}$$

З урахуванням (7) перепишемо (5) у вигляді системи диференціальних рівняннь першого порядку.

$$\begin{cases} y'(x)=y_1 \\ y'_1(x)=y_2 \\ \dots, \\ y'_{n-2}(x)=y_{n-1}, \\ y'_{n-1}(x)=f(x,y_{1,}y_{2,}\dots,y_{n-1}) \end{cases} \tag{8}$$

Початкові умови (6) з урахуванням (7) запишемо у вигляді

$$\begin{cases} y(x_0)=y_{00}, \\ y_1(x_0)=y_{10}, \\ \dots, \\ y_{n-1}(x_0)=y_{n-1,0} \end{cases} \tag{9}$$

Отже, задача (5), (6) зведена до вигляду (8), (9), який, у свою чергу, можна звести до вигляду (1), (2).

Чисельні методи розв’язування задачі Коші дозволяють знайти наближений розв’язок у вигляді таблиці. Першим кроком будь-якого чисельного методу розв’язування задачі Коші є розбиття відрізка $[a,b]$ на скученну кількість частин за допомогою вузлових точок $x_i, i=\overline{0,n}, n\geq 1$ таких, що

$$a=x_0<x_1<x_2<\dots<x_n=b$$

Позначимо $h_i=x_{i+1}-x_i, (i=\overline{0,n-1})$ відстань між двома сусідніми точками, тобто крок, з яким змінюєтья координата точки x_i. Сітку вузлів вважаємо нерівномірною, хоча, в частинному випадку, вона може стати рівномірною, коли

$$h_i = h = \frac{b-a}{n}, (i = \overline{0, n-1})$$

Оскільки в точці x_0 значення функції $y(x_0)$ є відомим, то надалі розглянемо відрізок $[x_i, x_{i+1}]$, вважаючи, що в точці x_i значення $y(x_i)$ відомо. Поставимо собі задачу найти значення $y(x_{i+1})$. Для цього проінтегруємо рівняння (1) за відрізком $[x_i, x_{i+1}]$, одержимо

$$\int_{x_i}^{x_{i+1}} y'(x) \cdot dx = \int_{x_i}^{x_{i+1}} f(x, y(x)) \cdot dx \qquad (10)$$

Позначимо

$$\Delta y_i = \int_{x_i}^{x_{i+1}} f(x, y(x)) \cdot dx \qquad (11)$$

Значення Δy_i будемо називати ***приростом функції*** $y(x)$ у точці x_i. Але за формулою (11) неможливо обчислити приріст, бо невідома залежність $y(x)$. Інтеграл у лівій частині рівності (10) можна обчислити. Отже, маємо (10)

$$y(x_{i+1}) - y(x_i) = \Delta y_i$$

Звідси приходимо до формули

$$y(x_{i+1}) = y(x_i) + \Delta y_i, i = \overline{0, n-1} \qquad (12)$$

Формула (12) лежить в основі більшості чисельних методів розв’язування задачі Коші. Вона дає вираз значення шуканої функції $y(x)$ у кажній наступній точці x_{i+1} через її значення у попередній точці x_i. Різні методи розв’язування задачі Коші відрізняються один від одного способом наближеного обчислення інтеграла (11), тобто приросту функцій. Найпростішим з чисельних методів є метод Ейлера, який ще називають методом ламаних або методом дотичних.

1. Основні теоретичні частини

Метод Ейлера та його модифікації

Метод Ейлера є історично першим методом чисельного розв’язування задачі Коші (1), (2). У практиці обчислень цей метод використовується рідко через невисоку точність. Але на його прикладі зручно пояснювати деякі поняття та ідеї побудови і дослідження чисельний методів розв’язування задачі Коші.

Для наближеного обчислення інтеграла (11) застосуємо одноточкову квадратурну формулу лівих прямокутників

$$\Delta y_i = h_{if}(x_i, y(x_i)) + O(h_i^2)$$

Відкинувши член порядку $O(h_i^2)$ і позначивши через y_i наближене значення величини $y(x_i)$, одержимо з (12) таку розрахункову формулу

$$y_{i+1} = y_i + h_i \cdot f(x_i, y_i), i = 0,1,\dots, n-1 \tag{13}$$

Формула (13) має назву ***формули Ейлера***, а чисельний метод розв’язування задачі Коші, визначений формулою (13), називається ***методом Ейлера***.

Дамо геометричне пояснення методу Ейлера (1). Нехай $y = y^{(0)}(x)$ є графіком шуканої інтегральної кривої, що проходить через точку $A_0 = (x_0, y_0)$. Треба знайти $y^{(0)}(x_1)$, тобто ординату точки $A_1 = (x_1, y^{(0)}(x_1))$.

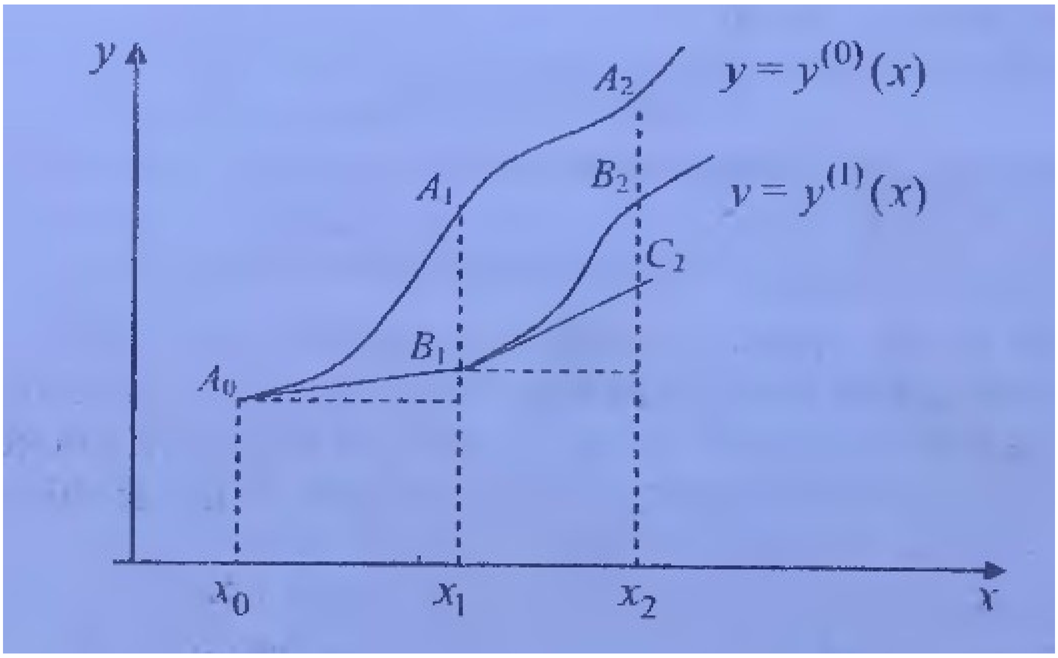


Рис 1. Геометричне зображення методу Ейлера

Однак, за формулою Ейлера обчислюється ордината точки $B_1 = (x_1, y_1)$, де $y_1 = y_0 + h_0 \cdot f(x_0, y_0)$. Точка B_1 розташована на дотичній, приведеній до графіка функції $y = y^{(0)}(x)$ у точці (x_0, y_0). Отже, замість того, щоб рухатись за графіком функції $y = y^{(0)}(x)$, метод дає можливість рухатись за дотичною $A_0 B_1$ до цього графіка. Довжина візрізка $A_1 B_1$ є похибкою на кроці в точці x_1.

Точка B_1 належить уже іншій інтегральній кривій диференціального рівняння (1). Нехай її рівняння буде $y = y^{(1)}(x)$. Аналогічно попереднім розрахункам за формулою Ейлера обчислюємо значення $y_2 = y_1 + h_1 \cdot f(x_1, y_1)$, тобто знаходимо ординату точки $C_2 = (x_2, y_2)$, що розташована на дотичній до графіка функції $y = y^{(1)}(x)$, проведеній в точці B_1. Довжина відрізка

B_2C_2 є похибкою на кроці, а довжина відрізка A_3C_2 є повною похибкою в точці x_2 .

Похибкою на кроці, або ***локальною похибкою***, будемо називати ту похибку, з якою обчислюється $y(x_{i+1})$, якщо припустити, що $y(x_i)$ відомо точно. Але відомим точно може бути лише $y(x_0)$. У точці x_1 значення $y(x_1)$ обчислюється вже наближено х похибкою $O(h_0^2)$. Ця похибка перейде до значення y_2 і потім в усі подальші значення $y_i, i=\overline{3,n}$. Отже, ***повна похибка*** при обчисленні значень $y(x_i), i=\overline{2,n}$ може бути більшою, ніж локальна, оскільки похибка на кажному кроці, переходячи в наступні кроки, може накопичуватися. Звуси випливає, що в точці x_n повна похибка може стати досить великою. Це є ***недоліком методу Ейлера***. Але його ***перевагою*** є простота розрахункової формули (13). Методом Ейлера рекомендєуться користуватися, якщо розв’язок потрібно знайти тільки на короткому проміжку і з невеликою точністю.

З рис. 1 видно, що в результаті його застосування отримана ламана $A_0B_1C_2$, кожна ланка якої дотикається відповідної інтегральної кривої. Ця ламана називається ***ламаною Ейлера***. Інколи сам метод Ейлера називають ***методом ламаних*** або ***методом дотичних***.

Метод Ейлера належить до класу ***однокрокових***, тобто таких методоів розв’язування задачі Коші, які дозволюсть знайти наближене значення розв’язку у вузлі x_{i+1} , використовуючи значення розв’язку тільки в одному попередньому вузлі x_i .

На відміну від однокрокових, ***багатокрокові*** методи розв’язування задачі Коші дозволяють відшукувати розв’язок у черговій вузловій точці x_{i+1} , використовуючи інформацію про розв’язок більш ніж ожній попередній вузловій точці.

Модифікований метод Ейлера можна добути, якщо для наближеного обчислення інтеграла (11) застосувати квадратурну формулу трапецій

$$\Delta y_i=\frac{h_i}{2}\left(f(x_i,y(x_i))+f(x_i+h_i,y(x_i+h_i))\right)+O(h_i^3) \qquad (14)$$

Відкинувши в (14) похибку $O(h_i^3)$, приходимо до наближеного рівняння відносно шуканого значення y_{i+1}

$$y_{i+1}=y_i+\frac{h_i}{2}\left(f(x_i,y_i)+f(x_{i+1},y_{i+1})\right) \qquad (15)$$

Рівняння (15) є прикладом ***неявного однокрокового методу***, оскільки воно не розв'язано відносно y_{i+1}. Але, якщо використати формулу Ейлера, то в межаї прийнятої похибки $O(h_i^3)$ нелінійне рівняння (15) можно лінеаризувати. Робиться це в такий спосіб. У правій частині формули (14) замінимо значення $y(x_i+h_i)$ таким розвиненням

$$y(x_i+h_i)=y(x_i)+h_i\cdot f(x_i,y(x_i))+O(h_i^2)=y(x_{i+1})+O(h_i^2)$$

де позначено

$$y^*(x_{i+1})\equiv y(x_i)+h_i\cdot f(x_i,y(x_i))$$

Тоді

$$f(x_{i+1},y(x_i+h_i))=f(x_{i+1},y(x_{i+1})+O(h_i^2))=f(x_{i+1},y(x_{i+1}))+O(h_i^2) \tag{16}$$

Підставивши (16) до (14), приходимо до двох розрахункових формул:

$$\begin{cases} y_{i+1}=y_i+h_i\cdot f(x_i,y_i), \\ y_{i+1}=y_i+\dfrac{h_i}{2}(f(x_i,y_i)+f(x_{i+1},y_{i+1})) \end{cases} \tag{17}$$

Локальна похибка формул (17) є $O(h_i^3)$, тобто на порядок меншою, ніж у формулі Ейлера. Розрахункові формули (17) і є ***модифікацією формули Ейлера.***

Якби для наближеного обчислення інтеграла (11) була використана квадратурна формула середніх прямокутників, то це дозволило б добути ще одну модифікацію методу Ейлера з локальноїї похибкою того ж порядку, що і в методі (17).

Методи Рунге-Кутта

Нехай треба знайти розв'язок задачі Коші (1), (2). Для цього на відрізку $[a,b]$ фіксуємо точки $x_i, i=\overline{0,n}$ із змінним кроком $h_i=x_{i+1}-x_i, i=\overline{0,n-1}$. Будемо шукати значення функції $y(x)$ у цих точках, використовуючи розрахункову формулу (12). Розглянемо окремий відрізок $[x_i,x_{i+1}]$, вважаючи, що значення $y(x_i)$ є відомим, та будемо шукати приріст (11).

Загальна ідея методів

Для наближеного обчислення приросту функції Δy_i за формулою (11) впровадимо до розгляду три набори параметрів:

$$\alpha_{1,}\alpha_{2,}\ldots,\alpha_q; \tag{18}$$

$$\begin{cases} \beta_{10} \\ \beta_{20}, \beta_{21}; \\ \dots; \\ \beta_{q0}, \beta_{q1}, \dots, \beta_{q,q-1}; \end{cases} \tag{19}$$

$$A_{0,} A-1, \dots, A_q \tag{20}$$

За допомогою параметрів (18), (19) складемо величини:

$$\begin{cases} \varphi_0 = h_i \cdot f(x_i, y_i), \\ \varphi_1 = h_i \cdot f(x_i + \alpha_1 \cdot h_i, y_i + \beta_{20} \cdot \varphi_0 + \beta_{21} \cdot \varphi_1), \\ \dots, \\ \varphi_q = h_i \cdot f(x_i + \alpha_q \cdot h_i, y_i + \sum_{m=0}^{q-1} \beta_{qm} \cdot \varphi_m) \end{cases} \tag{21}$$

Тепер запишемо лінійну комбінацію величин (21) із коефіцієнтами (20)

$$\sum_{j=0}^{q} A_j \cdot \varphi_j \tag{22}$$

Лінійна комбінація (22) буде аналогом квадратурної суми для обчислення інтеграла (11), тобто приросту функції Δy_i. Параметри (18), (19), (20) виберемо так, щоб (22) якомога точніше наближувала приріст функції Δy_i. Для цього похибку поближення, тобто величину

$$r_q(h_i) = \Delta y_i - \sum_{j=0}^{q} A_j \varphi_j$$

як фукнцію від кроку h_i, розвинемо у ряд Тейлора в околі значення $h_i = 0$. Маємо

$$r_q(h_i) = r_q(0) + \frac{h_i}{1!} r_q{}'(0) + \frac{h_i^2}{2!} r_q'' + \dots + h \frac{\%k_i}{k!} r_q^{(k)}(0) + \frac{h_i^{k+1}}{(k+1)!} r_q^{(k+1)}(\theta h_i), 0 \le \theta \le 1 \tag{23}$$

При цьому вважаємо, що права частина диференціального рівняння (1) є достатньо гладкою функцією, так що всі необхідні похідні існують і є неперервними функціями на $[a, b]$.

Якщо параметри (18), (19), (20) вдасться вибрати так, щоб для розвинення (23) виконувались умови

$$\begin{cases} r_q(0) = 0, \\ r'_q(0) = 0, \\ \dots, \\ r_q^{(k)}(0) = 0, \end{cases} \tag{24}$$

то похибка $r_q(h_i)$ буде величиною того же порядку мализни, що і h_i^{k+1}, тобто

$$r_q(h_i) = \frac{h_i^{k+1}}{(k+1)!} r_q^{(k+1)}(\theta h_i), 0 \le \theta \le 1 \tag{25}$$

Число k будемо називати ***порядком*** (ступенем) ***точності методу*** Рунге-Кутта. При цьому локальна похибка визначається формулою (25), а розрахункова формула приймає вигляд

$$y_{i+1}=y_i+\sum_{j=0}^{q} A_j\cdot\varphi_j \qquad (26)$$

Фіксуючи число q, будемо мати конкретний варіант методу Рунге-Кутта.

Записати в загальному вигляді (тобто при будь-якому значення числа q) систему рівняннь (24) для визнчання параметрів (18), (19), (20) важко. Тому розглянемо лише декілька прикладів побудови однокрокових методів за способом Рунге-Кутта.

Метод Рунге-Кутта першого порядку точності

При $q=0$ маємо лише один параметр A_0. Параметрів (18) та (19) не буде зовсім. Формула (26) набуває вигляду

$$y_{i+1}=y_i+A_0 h_i f(x_i,y_i)$$

Похибку $r_0(h_i)$ розкладемо в ряд за степенями h_i

$$r_0(h_i)=\Delta y_i-A_0 h_i f(x_i,y_i)=y(x_i+h)-y(x_i)-A_0 h_i f(x_i,y_i)=$$

$$=y(x_i)+h_i y'(x_i)+\frac{h_i^2}{2}y''(x_i)+\ldots-y(x_i)-A_0 h_i f(x_i,y_i)=$$

$$=h_i f(x_i,y_i)\cdot(1-A_0)+\frac{h_i^2}{2}(f'_x+f'_y\cdot f)\Big|_{\substack{x=x_i\\ y=y_i}}+O(h_i^3)$$

Від цього розвинення знаходимо похідні по :

$$r_0(h_i)=\Delta y_i-A_0 h_i f(x_i,y_i)=y(x_i+h_i)-y(x_i)-A_0 h_i f(x_i,y_i)=y(x_i)+h_i y'(x_i)+$$

$$+\frac{h_i^2}{2}(f'_x+f'_y\cdot f)\Big|_{\substack{x=x_i\\ y=y_i}}+O(h_i)$$

$$r_0''(h_i)=(f'_x+f'_y\cdot f)\Big|_{\substack{x=x_i\\ y=y_i}}+O(h_i)$$

Далі підставимо ці похідні до умови (24):

$$r_0(0)\equiv 0$$
$$r'_0(0)=(1-A_0)\cdot f(x_i,y_i)=0\Rightarrow A_0=1,$$
$$r_0''(0)=(f'_x+f'_y\cdot f)\Big|_{\substack{x=x_i\\ y=y_i}}.\neq 0$$

Отже, $k=1$. Розрахункова формула набуває вигляду

$$y_{i+1}=y_i+h_i\cdot f(x_i,y_i)$$

тобто вона збігається з формулою Ейлера. Похибку на кроці запишемо на підставі формули (25)

$$r_0(h_i)=\frac{h_i^2}{2}r_0''(\theta h_i)=O(h_i^2)$$

Отже, метод Ейлера є методом Рунге-Кутта першого порядку точності.

Метод Рунге-Кутта другого порядку точності

Нехай $q=1$, тоді із множини коефіцієнтів (18), (19), (20) шуканими будуть лише чотири

$$\alpha_{1,}\beta_{10}, A_0, A_1 \tag{27}$$

За формулами (21) маємо

$$\begin{cases} \varphi_0 = h_i \cdot f(x_i, y_i), \\ \varphi_1 = h_i \cdot f(x_i + \alpha_1 \cdot h_i, y_i + \beta_{10} \cdot h_i \cdot f) \end{cases}$$

Отже,

$$\Delta y_i \approx A_0 \cdot \varphi_0 + A_1 \cdot \varphi_1$$

Займемося вибором параметрів (27). Для цього запишемо формулу для локальнох похибки

$$r_1(h_i) = y(x_i + h_i) - y(x_i) - A_0 h_i f(x_i, y_i) - A_1 h_i f(x_i + \alpha_1 h_i, y_i + \beta_{10} h_i f(x_i, y_i)) \tag{28}$$

Праву частину рівності (28) розкладемо в ряд за степенями h_i , залишаючи тільки доданки з h_i^3 . Виконаємо це розвинення поступово

$$\begin{gathered} y(x_i + h_i) = y(x_i) + h_i y'(x_i) + \frac{h_i^2}{2} y''(x_i) + \frac{h_i^3}{3!} y'''(x_i) + \ldots = \\ = y(x_i) + h_i f(x_i, y_i) + \frac{h_i^2}{2}(f'_x + f'_y \cdot f)\Big|_{\substack{x = x_i \\ y = y_i}} + \\ + \frac{h_i^3}{6}(f''_{xx} + 2f''_{xy} + f''_{yy} \cdot f^2 + f'_y(f'_x + f'_y \cdot f))\Big|_{\substack{x = x_i \\ y = y_i}} + O(h_i^4) \end{gathered} \tag{29}$$

Щоб розкласти функцію в ряд за степенями використаємо таку формулу Тейлора

$$\begin{gathered} f(x + a, y + b) = f(x, y) + \left(a \frac{\partial}{\partial x} + b \frac{\partial}{\partial y}\right) f(x, y) + \frac{1}{2}\left(a \frac{\partial}{\partial x} + b \frac{\partial}{\partial y}\right)^{2f}(x, y) + \\ + \ldots + \frac{1}{m!}\left(a \frac{\partial}{\partial x} + b \frac{\partial}{\partial y}\right)^m f(x, y) + \ldots \end{gathered}$$

На підставі цієї формули маємо

$$\begin{gathered} \varphi_1(h_i) \equiv h_i f(x_i + \alpha_1 h_i, y_i + \beta_{10} h_i f(x_i, y_i)) = \\ = h_i f(x_i, y_i) + h_i^2 \cdot (\alpha_1 \cdot f'_x + \beta_{10} \cdot f \cdot f'_y)\Big|_{\substack{x = x_i \\ y = y_i}} + O(h_i^4) \end{gathered} \tag{30}$$

Підставивши (29), (30) до (28), маємо шукане розкладання

$$r_i(h_i) = (1 - A_0 - A_1) \cdot h_i f(x_i, y_i) + \frac{h_i^2}{2} D_1 + \frac{h_i^3}{6} D_2 + O(h_i^4) \tag{31}$$

де

$$D_1 = f'_x + f\cdot f'_y - 2A_1(\alpha_1\cdot f'_x + \beta_{10}\cdot f\cdot f'_x) = (1-2\alpha_1 A_1) f'_x + (1-2\beta_{10}A_1) f\cdot f'_y,$$
$$D_2 = f''_{xx} + 2\cdot f\cdot f''_{xy} + f^2\cdot f''_{yy} + f'_y\cdot(f'_x + f\cdot f'_y) - \tag{32}$$
$$-3A_1(\alpha_1^2\cdot f''_{xx} + 2\alpha_1\cdot\beta_{10}\cdot f\cdot f''_{xy} + \beta_{10}^2\cdot f^2\cdot f''_{yy})$$

У коефіцієнтах $D_{1,} D_2$ функція $f(x, y)$ та всі її частинні похідні обчислюються в точці (x_i, y_i) . Від функції (31) знаходимо потрібні похідні:

$$r'_1(h_i) = (1 - A_0 - A_1)\cdot f(x_i, y_i) + h_i D_1 + \frac{h_i^2}{2} D - 2 + O(h_i^3),$$
$$r''_1(h_i) = D_1 + h_i D_2 + O(h_i^2),$$
$$r'''_1(h_i) = D_2 + O(h_i)$$

Підставимо ці похіні до умови (24):

$$r_1(0) \equiv 0,$$
$$r'(0) = (1 - A_0 - A_1) f(x_i, y_i) = 0, \Rightarrow A_0 + A_1 = 1$$
$$r''(0) = D_1 = 0, \Rightarrow 2\alpha_1 A_1 = 1, 2\beta_{10} A_1 = 1$$
$$r'''(0) = D_2 \neq 0$$

Отже, для чотирьох невідомих коефіцієнтів (27) маємо систему трьох рівнянь

$$\begin{cases} A_0 + A_1 = 1, \\ 2\alpha_1 A_1 = 1, \\ 2\beta_{10} A_1 = 1 \end{cases} \tag{33}$$

Добута система (33) має безліч розв’язків. Кожен з них визначає окрему розрахункову формулу другого порядку точності, тобто формулу з локальною похибкою $O(h_i^3)$.

Якщо вибрати $\alpha_1 = \beta_{10} = 1, A_0 = A_1 = \frac{1}{2}$, то добудемо розрахункову формулу модифікованого методу Ейлура (17). При $\alpha_1 = \beta_{10} = \frac{1}{2,} A_0 = A_1 = \frac{1}{2}$ добудемо ще одну модифікацію методу Ейлера. Зрозуміло, що таких модифікацій можна побудувати скільки завгодно.

Якщо виконуються умови (33), то формула (31) для локальної похики стане такою

$$r_1(h_i) = \frac{h_i^3}{6} D_2 + O(h_i^4)$$

У деяких випадках за разунок вдалого вибору параметрів (27) можна зменшити коефіцієнт D_2 , а значить і похибку. Наприклад, якщо до виразу (32) підставимо значення $\alpha_1 = \beta_{10} = \frac{1}{2A_1}$, взяті із умов (33), то будемо мати

$$D_2 = (f''_{xx} + 2\cdot f\cdot f''_{xy} + f^2\cdot f''_{yy})\left(1 - \frac{3}{4A_1}\right) + f'_y\cdot(f'_x + f\cdot f'_y)$$

Якщо далі вибрати $A_1=\frac{3}{4}$, то $D_2=f'_y\cdot(f'_x+f\cdot f'_y)$. При такому виборі розрахункова формула (26) набуває віигляду

$$y_{i+1}=y_i+\frac{1}{4}(\varphi_0+3\varphi_1)$$

$$\text{де}\quad \begin{cases} \varphi_0=h_i\cdot f(x_i,y_i), \\ \varphi_1=h_i\cdot f(x_i+\frac{2}{3}\cdot h_i, y_i+\frac{2}{3}\varphi_0) \end{cases}$$

Автоматичний вибір кроку в методах Рунге-Кутта

При розв'язуванні задачі Коші методом Рунге-Кутта *k*-го порядку точності замість точного значення $y(x_{i+1})$ можна знайти лише наближене значення y_{i+1} (за розрахунковою формулою (26)). Повною похибкою наближеного значення є різниця

$$y(x_{i+1})-y_{i+1}=O(h_i^k)=h_i^k\cdot\rho(x_{i+1},h_i)$$

Крок h_i вважаємо величиною малою. Зрозуміло, що похибку можна зменшувати двома способами: або збільшивши порядок методу *k*, або зменшивши крок h_i. Приймаємо, що порядок методу не змінюється і розглянемо алгоритм, який дозволяє регулювати величину похибки за допомогою зміни кроку інтегрування h_i.

Інтуїтивно зрозуміло, що можна проводити інтегрування з відносно великим кроком там, де шукана функція змінюється повільно. Там, де розв'язок змінюється швидко, необхідно проводити розрахунок з меншим кроком. Оскільки розв'язок заздалегідь невідомий, то проблема в тому, як визначити величину кроку *h*, з яким слід провести наступний крок інтегрування. Звичайний підхід оцінки кроку *h* спирається на оцінку повної похибки і залежно від її величини дозволяє зменшити або збільшити поточне значення кроку *h*.

Отже, похначимо через *x* точку, в котрій треба обчислити . Нехай у цю точку ми прийшли з кроком h_1 за допомогою методу Рунге-Кутта *k*-го порядку точності, тоді

$$\underbrace{y(x)}_{\text{точне значення}} = \underbrace{y_{h_1}}_{\text{наближене значення}} + \underbrace{h_1^k\rho(x,h_1)}_{\text{повна похибка}}$$

Якщо в ту саму точку *x* прийти з кроком h_2, то одержимо, взагалі кажучи, інше наближене значення y_{h2}, тоді

$$y(x)=y_{h_2}+h_2^k\cdot\rho(x,h_2)$$

В останніі двої формулах коефіцієнт $\rho(x,h)$ є невідомим. Він залежить від координати точки *x* та від кроку *h*. Цей коефіцієнт можна записати у вигляді розкладання $\rho(x,h)=\rho(x,0)+O(h)=\rho_0+O(h)$, вважаючи крок *h* величиною малою, та обмежитись першим членом цього розкладання. Тоді приходимо до таких двох наближений рівностей:

$$y(x)\approx y_{h_1}+h_1^k\rho_{0,} \tag{34}$$

$$y(x)\approx y_{h_2}+h_2^k\rho_0 \tag{35}$$

Віднявши (34) від (35), знаходимо наближено коефіцієнт ρ_0 при головному члені повної похибки

$$\rho_0\approx\frac{y_{h_2}-y_{h_1}}{h_1^k-h_2^k}$$

Найчастіше один крок вибирають вдвічі більшим, ніж інший, як показано на рис. 2. Тоді вузли більшої сітки попадають у вузли дрібної.

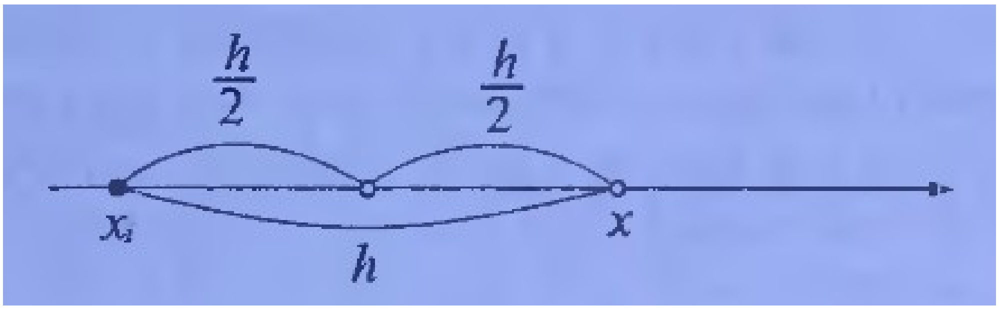


Рис 2. Схема вибору кроків інтегрування

Нехай $h_1=h, h_2=\frac{h}{2}$, тоді

$$\rho_0\approx\frac{(y_{h/2}-y_h)\cdot 2^k}{h^k\cdot(2^k-1)} \tag{36}$$

Підставимо (36) у (34), (35), приходимо до таких двох формул:

$$\begin{aligned} y(x)&\approx y_h+\varepsilon_h, \\ y(x)&\approx y_{h/2}+\varepsilon_{h/2}, \end{aligned} \tag{37}$$

де головні члени похибок знаходяться за формулами:

$$\varepsilon_h=\frac{(y_{h/2}-y_h)\cdot 2^k}{2^k-1}, \varepsilon_{h/2}=\frac{(y_{h/2}-y_h)}{2^k-1} \tag{38}$$

Нехай у наступній точці *x* треба наблизитись до значення *y(x)* з заданою похибкою ε, при цьому координата точки *x* заздалегідь невідома. Знаходити саму точку *x* та значення функції *y(x)* у цій точці будемо за таким алгоритмом.

Алгоритм автоматичного вибору кроку інтегрування

1. Припускаємо, що в точці x_i відоме значення та вибрано початковий крок *h*.

2. Відштовхуючись від значення y_i , із двома кроками h та $\frac{h}{2}$ методом Рунге-утта k-го порядку точності обчислюємо значення y_h, $y_{h/2}$. За формулами (38) обчислюємо $\varepsilon_h, \varepsilon_{h/2}$.

3. Якщо $|\varepsilon_{h/2}| \le \varepsilon$, то обчислюємо за формулою (37) значення $y(x_i + h) \approx y_{h/2} + \varepsilon_{h/2}$ і точку x_i пересуваємо в точку x_i+h, тобто індекс i збільшуємо на одиницю. Для очислення значення шуканої функції в наступній точці треба визначитися із початковим кроком h для цієї точки. Для цього переходимо на п. 4.

Якщо $|\varepsilon_{h/2}| > \varepsilon$, то крок h зменшуємо вдвічі, тобто точку x присуваємо вдвічі ближче до точки x_i і переходимо на п.2 (але вже із вдвічі меншим кроком).

4. Визначаємо початковий крок для наступної точки залежно від ε_h .

Якщо $|\varepsilon_h| \le \varepsilon$, то той крок h, з яким прийшли у поперрепублика точку, збільшуємо вдічі і це буде початковий крок для наступної точки.

Якщо $|\varepsilon_h| > \varepsilon$, то той крок h, з яким прийшли до попередньої точки залишаємо як початковий крок для наступної точки.

Після того, як початковий крок для наступної точки вибрано, слід прослідкувати, щоб наступна точка, тобто точка з координатою (x_i+h), не виявилася за межами відрізка інтегрування. Для цього обчислюємо $\Delta_i = b - x_i$. Якщо $\Delta_i \le \varepsilon_1$, то зупиняємось (ε_1 — це та похибка, з якою треба наблизитись до точки $x=b$). Якщо $\Delta_i > \varepsilon$, то інтегрування продовжуємо. Якщо $h > \Delta_i$, то беремо $h = \Delta_i$ і продовжуємо інтегрування.

Для продовження інтегрування переходимо на п. 2.

Багатокрокові методи розв’язання задачі Коші

Розв’язуючи задачу Коші, багатокрокові методи, на відміну від однокрокових, при визначенні y_{i+1} використовують декілька вже відомих значень y_i, y_{i-1}, y_{i-2}, ..., y_{i-n}. У випадку змінного кроку h_i використанні багатокрокових методів значно ускладнюється, тому надалі вважаємо всі вузлові точки рівновіддаленими. Серед багатокрокових методів гироко відомими є методи Адамса.

Екстраполяційні методи Адамса

Розв’язуємо задачу Коші (1), (2). Приймаємо, що $x_i, i=\overline{0,N}$ — система рівновіддалених вузлових точок із сталим кроком h, тобто

$$x_i = x_0 + ih,\, i = 0,1,\ldots,N;\, h = \frac{b-a}{N}$$

Вважаємо, що $y_i, i=\overline{0,n}$ — відомі значення, вони можуть бути обчислені, наприклад, методом Рунге-Кутта. Шукаємо $y(x_{i+1})$ за формулою (12), в якій

$$\Delta y_i = \int_{x_i}^{x_{i+1}} f(x, y(x)) \cdot dx = \int_{x_i}^{x_{i+1}} y'(x) \cdot dx \tag{39}$$

Для наближного обчислення інтеграла (39) фунцію $y'(x), x \in [x_i, x_{i+1}]$ наблизимо алгебраїчним інтерполяційним многочленом степеня *n*, побудованим за вузлами x_i, x_{i-1}, ..., x_{i-n} (рис. 3). Таке інтерполювання за межі таблиця значень функції має назву естраполювання. З цим і пов’язана назва методу.

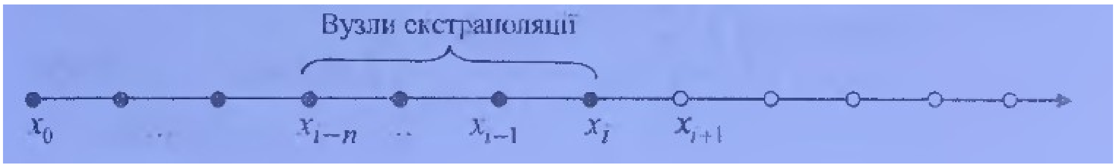


Рис 3: Схема вузлів, що використовуються

Побудуємо поліном за формулою Лагранжа

$$y'(x) = L_n(x) + R_n(x) \tag{40}$$

Тут

$$L_n(x) = \sum_{k=0}^{n} \left(y'_{i-k} \prod_0^n \frac{x - x_{i-m}}{x_{i-k} - x_{i-m}} \right), \tag{41}$$

$$R_n(x) = \frac{y^{(n+2)}(\xi(x)) \cdot \omega_{n+1}(x)}{(n+1)!} \tag{42}$$

Приймемо до уваги те, що вузли розташовані рівновіддалено. Позначимо

$$x = x_i + h\alpha,\, \alpha \in [0;1],\, x_{i-k} = x_i - kh,\, x_{i-m} = x_i - mh$$

Запишемо окремо добутки, що присутні у виразах (41), (42).

$$\prod_{\substack{m=0\\ m\neq k}}^{n} \frac{(x - x_{i-m})}{(x_{i-k} - x_{i-m})} = \prod_{\substack{m=0\\ m\neq k}}^{n} \frac{(\alpha+m)}{(m-k)} = \frac{1}{(-k)(-k+1)\ldots(-1)\cdot 1\cdot 2\ldots(n-k)} \prod_{\substack{m=0\\ m\neq k}}^{n} (\alpha+m$$
$$\frac{(-1)^k}{k!(n-k)!} \prod_{\substack{m=0\\ m\neq k}}^{n} (\alpha+m) = \frac{(-1)^k}{k!(n-k)!} \prod_{\substack{m=0\\ m\neq k}}^{n} (\alpha+m), \tag{43}$$

$$\omega_{n-1}(x) = \prod_{m=0}^{n} (x - x_{i-m}) = h^{n+1} \cdot \prod_{m=0}^{n} (\alpha+m) \tag{44}$$

Підставивши (43) та (44) до функцій (41), (42), будемо мати

$$L_n(x_i + \alpha h) = \sum_{k=0}^{n} y'_{i-k} \frac{(-1)^k}{k!(n-k)!} \prod_{\substack{m=0\\ m\neq k}}^{n} (\alpha+m), \tag{45}$$

$$R_n(x_i+\alpha h)=\frac{h^{n+1}}{(n+1)!}\cdot y^{(n+2)}(\xi(x_i+\alpha h))\cdot\prod_{m=0}^{n}\alpha+m \qquad (46)$$

Тепер підставимо (45), (46) до формули (40), а потім під знак інтеграла (39). У результаті знаходимо приріст функції

$$\Delta y_i=\int_{x_i}^{x_{i+1}} y'(x)dx=h\int_0^1 y'(x_i+\alpha h)\cdot d\alpha=$$

$$=h\sum_{k=0}^{n} y'_{i-k}\frac{(-1)^k}{k!(n-k)!}\int_0^1\prod_{\substack{m=0\\ m\neq k}}^{n}(\alpha+m)d\alpha+$$

$$+\frac{h^{n+2}}{(n+1)!}\int_0^1 y^{(n+2)}(\xi(x_i+\alpha h))\prod_{m=0}^{n}(\alpha+m)\cdot d\alpha$$

Позначимо

$$A_k=\frac{(-1)^k}{k!(n-k)!}\int_0^1\prod_{\substack{m=0\\ m\neq k}}^{n}(\alpha+m)\cdot d\alpha, k=\overline{0,n} \qquad (47)$$

$$r_n=\frac{h^{n+2}}{(n+1)!}\int_0^1 y^{(n+2)}(\xi(x_i+\alpha h))\prod_{m=0}^{n}(\alpha+m)\cdot d\alpha$$

Тут r_n — похибка (залишковий член) екстраполяційної формули Адамса. Виділимо головний член цієї похибки, для чого запишемо розкладання

$$y^{(n+2)}(\xi)=y^{(n+2)}(x_i)+(\xi-x_i)\cdot y^{(n+3)}(\zeta), \zeta\in(\zeta, x_i)$$

і підставимо його у вираз для похибки r_n , одержимо

$$r_n=\frac{h^{n+2}y^{(n+2)}(x_i)}{(n+1)!}\int_0^1\prod_{m=0}^{n}(\alpha+m)d\alpha+$$

$$+\frac{h^{n+3}}{(n+1)!}\int_0^1 y^{(n+3)}(\zeta(\alpha))\frac{(\xi-x_i)}{h}\prod_{m=0}^{n}(\alpha+m)d\alpha$$

Позначимо

$$C_{n+1}=\frac{1}{(n+1)!}\int_0^1\prod_{m=0}^{n}(\alpha+m)\cdot d\alpha,$$

тоді

$$r_n=h^{n+2}y^{(n+2)}(x_i)\cdot C_{n+1}+O(h^{n+3}) \qquad (48)$$

Розрахункова формула екстраполяційного методу Адамса набуває вигляду

$$y_{i+1}=y_i+h\sum_{k=0}^{n} A_k\cdot f(x_{i-k}, y_{i-k}) \qquad (49)$$

Далі розглянемо декілька частинних випадків.

1. При *n=0* маємо одноточковий варіант екстраполяціного методу Адамса. За формулою (49) знаходимо

$$y_{i=1}=y_i+h\cdot A_0 cdit f(x_i, y_i)$$

Невідомий коефіцієнт обчислюємо за формулою (47)

$$A_0=\frac{1}{1\cdot 1}\int_0^1 1\cdot d\alpha=1$$

Отже, розрахункова формула набуває вигляду

$$y_{i+1}=y_i+h\cdot f(x_i,y_i),i=\overline{0,N-1}$$

Використовуючи (48), знаходимо похибку цієї розрахункової формули $r_0=O(h^2)$. Отже, отримали формулу Ейлера.

2. При *n=1* будемо мати двоточковий екстраполяційний метод Адамса. На підставі формули (49) знаходимо

$$y_{i+1}=y_i+h\cdot(A_0\cdot f(x_i,y_i)+A_1\cdot f(x_{i-1},y_{i-1}))$$

Коефіцієнти A_0,A_1 обчислюємо за формулою (47)

$$A_0=\frac{1}{1\cdot 1}\int_0^1(1+\alpha)\cdot d\alpha=\left(\alpha+\frac{\alpha^2}{2}\right)\Big|_{\alpha=0}^{\alpha=1}=1+\frac{1}{2}=\frac{3}{2},$$

$$A_1=\frac{-1}{1\cdot 1}\int_0^1\alpha\cdot d\alpha=\frac{-\alpha^2}{2}\Big|_{\alpha=0}^{\alpha=1}=-\frac{1}{2}$$

Остаточно розрахункова формула набуває вигляду

$$y_{i+1}=y_i+\frac{h}{2}\cdot(3f(x_i,y_i)-f(x_{i-1},y_{i-1})),i=\overline{0,N-1}¿$$

Локальна похибка добутої формули випливає з (46) $r_1=O(h^3)$.

3. Розглянемо ще один випадок екстраполяційного методу Адамса при *n=2*. Формула (49) має вигляд

$$y_{i+1}=y_i+h\cdot(A_9\cdot f(x_i,y_i)+A_1\cdot f(x_{i-1},y_{i-1})+A_2\cdot f(x_{i-2},y_{i-2}))$$

Аналогічно двом попереднім випадкам обчислюємо коефіцієнти

$$A_0=\frac{23}{12},A_1=\frac{-4}{3},A_2=\frac{5}{12}$$

Тепер можна записати розрахункову формулу

$$y_{i+1}=y_i+\frac{h}{12}\cdot(23f(x_i,y_i)-16f(x_{i-1},y_{i-1})+5f(x_{i-2},y_{i-2})),i=\overline{0,N-1}$$

із похибкою на кроці $r_2=O(h^4)$.

Інтеполяційний много член для функції $y'(x)=f(x,y(x))$ можна представити не тільки у формі Лагранжа, а також у формуі Ньютона для інтерполювання в кінці таблиці. У цьому випадку екстраполяційний метод Адамса може бути записаним через скінченні різниці функції $y'(x)$.

Інтерполяційні методи Адамса

Цей метод будує розрахункові формули аналогічно екстраполяційному методу, але при обчисленні приросту функції (39) наближує функцію $y'(x)$ на відрізку $[x_i,x_{i+1}]$ інтерполяційним алгебраїчним многочленом степеня *n+1*, побудованим за вузлами x_{i+1}, x_i, x_{i-1}, ..., x_{i-n} (рис. 4).

Рис 4: Схема вузлів, що використовуються

Запишемо інтерполяційний многочлен разом із залишком

$$y'(x)=\sum_{k=-1}^{n}\left(y'_{i-k}\prod_{\substack{m=-1\\ m\neq k}}^{n}\frac{x-x_{i-m}}{x_{i-k}-x_{i-m}}\right)+\frac{y^{(n+3)}(\xi(x))\cdot\omega_{n+2}(x)}{(n+2)!} \tag{50}$$

Тут $\omega_{n+2}(x)=\prod_{m=-1}^{n}(x-x_{i-m})$.

Нехай так же як і в екстраполяційному методі

$$x=x_i+h\alpha,\alpha\in[0;1],x_{i-k}=x_i-kh,x_{i-m}=x_i-mh,$$

тоді

$$\begin{gathered}\prod_{\substack{m=-1\\ m\neq k}}^{n}\frac{(x-x_{i-m})}{(x_{i-k}-x_{i-m})}=\prod_{\substack{m=-1\\ m\neq k}}^{n}\frac{(\alpha+m)}{(m-k)}=\\ =\frac{1}{(-1-k)(-k)(-k+1)\ldots(-1)\cdot 1\ldots(n-k)}\prod_{\substack{m=-1\\ m\neq k}}^{n}(\alpha+m)=\\ =\frac{(-1)^{k+1}}{(k+1)!(n-k)!}\prod_{\substack{m=-1\\ m\neq k}}^{n}(\alpha+m),\end{gathered} \tag{51}$$

$$\omega_{n+2}(x)=\prod_{m=-1}^{n}(x-x_{i-m})=h^{n+2}\cdot\prod_{m=-1}^{n}(\alpha+m) \tag{52}$$

Підставивши (51), (52) у функцію (50), добудемо формулу для $y'(x)$

$$\begin{gathered}y'(x)=y'(x_i+\alpha h)=\sum_{k=-1}^{n}y'_{i-k}\frac{(-1)^{k+1}}{(k+1)!(n-k)!}\prod_{\substack{m=-1\\ m\neq k}}^{n}(\alpha+m)+\\ +\frac{y^{(n+3)}(\xi(x_i+\alpha h))}{(n+2)!}\cdot h^{n+2}\prod_{m=-1}^{n}\text{¿}\end{gathered} \tag{53}$$

Тепер інтегруємо вираз (53) для того, щоб обчислити приріст функціїї (39).

$$\begin{gathered}\Delta y_i=h\int_0^1 y'(x_i+\alpha h)d\alpha=h\sum_{k=-1}^{n}y'_{i-k}\frac{(-1)^{k+1}}{(k+1)!(n-k)!}\int_0^1\prod_{\substack{m=-1\\ m\neq k}}^{n}(\alpha+m)d\alpha+\\ +h^{n+3}\frac{\int_0^1 y^{(n+3)}(\xi(x_i+\alpha h))}{(n+2)!}\prod_{m=-1}^{n}(\alpha+m)d\alpha\end{gathered}$$

Позначимо

$$A_k=\frac{(-1)^{k+1}}{(k+1)!(n-k)!}\int_0^1\prod_{\substack{m=-1\\ m\neq k}}^{n}(\alpha+m)d\alpha,k=\overline{-1,n}, \tag{54}$$

$$r_n=h^{n+3}\int_0^1 y^{(n+3)}\frac{(\xi(x_i+\alpha h))}{(n+2)!}\prod_{m=-1}^{n}(\alpha+m)d\alpha=O(h^{n+3}) \tag{55}$$

та приходимо до наближеної розрахункової формули

$$y_{i+1}=y_i+h\sum_{k=-1}^{n} A_k\cdot f(x_{i-k}, y_{i-k}) \qquad (56)$$

з похибкою $r_n=O(h^{n+3})$.

Наведемо декілька частинних випадків розрахункової формули інтерполяційних методів Адамса.

1. При *n=-1* маємо із формули (56)

$$y_{i+1}=y_i+h\cdot A_{-1}\cdot f(x_{i+1}, y_{i+1})$$

Невідомий коефіцієнт A_{-1} обчислюємо за формулою (54)

$$A_{-1}=\frac{1}{1\cdot 1}\int_0^1 1\cdot d\alpha=1$$

Отже, розрахункова формула набуває вигляду

$$y_{i+1}=y_i+h\cdot f(x_{i+1}, y_{i+1}), i=\overline{0,N-1}$$

Це неявний метод Ейлера з локальною похибкою $r_{-1}=O(h^2)$.

2. При *n=0* будемо мати двоточковий інтерполяційний метод Адамса. На підставі формули (56) знаходимо

$$y_{i+1}=y_i+h\cdot(A_{-1}\cdot f(x_{i+1}, y_{i+1})+A_0\cdot f(x_i, y_i))$$

Коефіцієнти A_{-1}, A_0 обчислюємо за формулою (54)

$$A_{-1}=\frac{1}{1\cdot 1}\int_0^1 \alpha\cdot d\alpha=\frac{1}{2}, A_0=\frac{-1}{1\cdot 1}\int_0^1(\alpha-1)\cdot d\alpha=-\left(\frac{\alpha^2}{2}-\alpha\right)\Big|_{\alpha_0}^{\alpha=1}=\frac{1}{2}$$

Остаточно розрахункова формула набуває вигляду

$$y_{i+1}=y_i+\frac{h}{2}\cdot(f(x_{i+1}, y_{i+1})+f(x_i, y_i)), i=\overline{0,N-1}$$

Локальна похибка добутоє формули випливає з (55): $r_0=O(h^3)$. Це неявний модифікований метод Ейлера (див. (15)).

3. Аналогічно попереднім двом випадкам добуваємо розрахункову формулу інтерполяційного методу Адамса при *n=1*

$$y_{i+1}=y_i+\frac{h}{15}\cdot(5f(x_{i+1}, y_{i+1})+8f(x_i, y_i)-f(x_{i-1}, y_{i-1})), i=\overline{1,N-1}$$

із похибкою на кроці $r_1=O(h^4)$.

Порівнюючи екстраполяційні та інтерполяційні методи Адамса, можна відмітити таке: екстраполяційні методи є явними, а інтерполяційні — неявними; при одному і тому ж *n* порядок локальної похибки (відносно *h*) інтерполяційного методу Адамса є на одиницю більшим, тобто інтерполяційні методи Адамса є точнішими за екстраполяційні.

Методи Адамса явний і неявний використовається разом у багатокрокових методах, які мають назву "***прогноз-корекція***". На кожному кроці явний метод використовується один раз для обчислення “прогнозу”. Потім за допомогою неявного методу будується ітераційний процес. У цілому метод “прогно-корекція” є явним. Обидва методи рекомендується брати одного порядку точності.

Порівняємо між собою методи Рунге-Кутта та методи Адамса

1. Методи Рунге-Кутта є однокроковими, а методи Адамса — багатокроковими. Отже, у методів Рунге-Кутта на старті немає проблем (вони стастартуючі), а методи Адамса на старті потребують допомоги інших методів.

2. У методах Рунге-Кутта функція *f(x,y)* в одному і тому вузлі обчислюється декілька разів, а добутий результат використовується тільки один раз (для обчислення наближного значення функції *f(x,y)* у наступному вузлі). У методах Адамса функція для кожного вузла обчислюється один раз, а використовується для обчислення шуканих значень функції *y(x)* у декількох наступних вузлах. Якщо *f(x,y)* є складної функцією, то вказана особливість робить методи Адамса економнішими.

3. Методи Рунге-Кутта легко узагальнюються на випадок змінного кроку інтегрування, а методи Адамса дуже ускладнюються у випадку нерівномірного розтаушвання вузлів сітки.

2. Чисельний експеримент та аналіз результатів

Згідно з моїм номером у журналі, мені потрібно було реалізувати наступний метод Рунге-Кутта (9%6=3):

$$\begin{aligned}
&y_{i+1}=y_i=\frac{1}{6}(\varphi_0+4\varphi_1+\varphi_2),\\
&\varphi_0=h\cdot f(x_i,y_i),\\
&\varphi_1=h\cdot f\left(x_i+\frac{h}{2,}\,y_i+\frac{\varphi_0}{2}\right),\\
&\varphi_2=h\cdot f(x_i+h,y_i-\varphi_0+2\varphi_1)\\
&\varepsilon=10^{-3}\\
&k=3
\end{aligned}$$

Задача Коші мого варіанту (9), відповідно, мала такий вигляд:

$$\begin{cases} y'(x)=xy^3-1\\ y(0)=0\end{cases}$$

Згідно загальних умов, початковий крок при автоматичному підборі кроку має бути $h_0=0{,}5$, а умова зупинки методу: $b-x_n\le\varepsilon_1=10^{-6}$, а відрізок — *[a, b]*.

За приблизно точний розв’язок був узятий метод більш високого порядку, який зі стабільним малим кроком проходив від 0 до 1. Функції, які розв’язували задачу Коші з автоматичним подбором кроку та стабільним, повертали списки точок, які потом використовувались для побудови таблиць, де порівнювалися значення в точках з більш точним рішенням, та для побудови графіків.

Для реалізації цих методів була обрана мова програмування Python 3.7 з такими модулями як NumPy (деякі математичні функції) та Matplotlib (графік), у доданок з PrettyTable (побудова таблиці).

Вихідний текст програми, який був перенапрямлений у файл output.txt:

```
Кол-во пар при автоматическом: 147
Таблица для автоматического шага:
+------------+----------------------+----------------------+---------------------------+
|    X_k     |   Y(X_k) (точное)    |   Y_k (численное)    | Y(X_k) - Y_k (погрешность) |
+------------+----------------------+----------------------+---------------------------+
|    0.0     |         0.0          |         0.0          |            0.0            |
| 0.0078125  | -0.010000000020833335 | -0.007812500003637979 |   -0.0021875000171953565  |
|  0.015625  | -0.020000000641666703 | -0.015625000164921708 |   -0.004375000476744995   |
| 0.0234375  | -0.030000004862501307 | -0.023437501327862396 |    -0.0065625035346389 1  |
|  0.03125   | -0.04000002048335078 | -0.03125000571890449 |   -0.008750014764446291   |
| 0.0390625  | -0.04000002048335078 | -0.039062517645423414 |   -0.0009375028379273662  |
|  0.046875  | -0.050000062504296776 | -0.046875044194236784 |   -0.003125018310059992   |
| 0.0546875  | -0.0600001555256716  | -0.05468759593014568 |   -0.005312559595525922   |
|   0.0625   | -0.07000033614852308 | -0.06250018759453296 |   -0.007500148553990116   |
| 0.0703125  | -0.08000065537561366 | -0.07031283880405881 |   -0.009687816571554847   |
|  0.078125  | -0.08000065537561366 | -0.07812557474950915 |   -0.0018750806261045028  |
| 0.0859375  | -0.0900011810133269  | -0.0859384268948712  |   -0.0040627541184556915  |
|  0.09375   | -0.10000200007499951 | -0.09375143367673093 |   -0.006250566398268584   |
| 0.1015625  | -0.11000322118636699 | -0.10156464120411121 |   -0.008438579982255778   |
|  0.109375  | -0.11000322118636699 | -0.10937810395889547 |   -0.0006251172274715222  |
| 0.1171875  | -0.12000497699400454 | -0.11719188549701037 |   -0.0028130914969941717  |
|   0.125    | -0.13000742657786396 |  -0.125006059150573  |   -0.005001367427290959   |
| 0.1328125  | -0.14001075786925002 | -0.13282070873124174 |   -0.007190049138008275   |
|  0.140625  | -0.15001519007584854 | -0.14063592923504767 |   -0.009379260840800874   |
| 0.1484375  | -0.15001519007584854 | -0.14845182754902206 |   -0.001563362526826484   |
|  0.15625   | -0.1600209761157115  |  -0.156268523159979  |   -0.0037524529557325104  |
| 0.1640625  | -0.17002840506242473 | -0.16408614886585673 |   -0.005942256196567991   |
|  0.171875  | -0.18003780460402855 | -0.17190485149006968 |   -0.008132953113958868   |
| 0.1796875  | -0.18003780460402855 | -0.17972479259937427 |  -0.00031301200465427836  |
|   0.1875   | -0.19004954351863562 | -0.18754614922580568 |   -0.002503394292829947   |
| 0.1953125  | -0.20006403417009047 | -0.19536911459329978 |    -0.00469491957679069   |
|  0.203125  | -0.2100817350274453  | -0.20319389884967473 |   -0.006887836177770557   |
| 0.2109375  | -0.2201031532124871  | -0.21102072980471076 |   -0.009082423407776347   |
|  0.21875   | -0.2201031532124871  | -0.21884985367513318 |   -0.0012532995373539246  |
| 0.2265625  | -0.23012884708004333 | -0.22668153583737477 |   -0.0034473112426685604  |
|  0.234375  | -0.24015942883631863 | -0.23451606158906768 |   -0.005643367247250952   |
| 0.2421875  | -0.2501955672010781  | -0.2423537369202936  |    -0.0078418302807845    |
|    0.25    | -0.2501955672010781  | -0.25019488929570327 |   -6.779053748351416e-07  |
| 0.2578125  | -0.2602379901200915  |  -0.258039868448703  |   -0.0021981216713884977  |
|  0.265625  | -0.27028748753489523 | -0.26588904718899814 |   -0.004398440345897092   |
| 0.2734375  | -0.2803449142176139  | -0.27374282222487845 |   -0.0066020919927354815  |
|  0.28125   | -0.2904111926793196  |  -0.281601615001734  |   -0.008809577677585612   |
| 0.2890625  | -0.2904111926793196  | -0.28946587255839556 |   -0.0009453201209240203  |
|  0.296875  | -0.30048731616119245 | -0.2973360684030088  |   -0.0031512477581836507  |
| 0.3046875  | -0.31057435171859393 | -0.30521270341027007 |   -0.005361648308323863   |
|   0.3125   | -0.3206734434090716  | -0.31309630674197997 |   -0.007577136667091644   |
| 0.3203125  | -0.33078581559629594 | -0.3209874367930044  |   -0.009798378803291541   |
|  0.328125  | -0.33078581559629594 | -0.3288866821648769  |   -0.0018991334314190644  |
| 0.3359375  | -0.3409127763829862  | -0.33679466266942676 |   -0.004118113713559468   |
|  0.34375   | -0.3510557211870276  | -0.34471203036498044 |   -0.0063436908220471655  |
| 0.3515625  | -0.36121613647622186 | -0.35263947062785406 |   -0.008576665848367804   |
|  0.359375  | -0.36121613647622186 | -0.36057770326204025 |   -0.0006384332141816107  |
| 0.3671875  | -0.37139560367846064 | -0.36852748365018717 |   -0.0028681200282734687  |
|   0.375    | -0.3815958032855787  | -0.3764896039491783  |   -0.0051061993364003855  |
| 0.3828125  | -0.39181851917074295 | -0.3844648943338448  |   -0.007353624836898165   |
|  0.390625  | -0.4020656431409841  | -0.39245422429258464 |   -0.009611418848399444   |
| 0.3984375  | -0.4020656431409841  | -0.40045850397891924 |   -0.0016071391620648412  |
|  0.40625   | -0.4123391797483931  | -0.40847868562329753 |   -0.0038604941250955926  |
| 0.4140625  | -0.42264125138561065 | -0.41651576500975573 |   -0.006125486375854916   |
|  0.421875  | -0.4329741036935513  | -0.4245707830223632  |   -0.008403320671188153   |
| 0.4296875  | -0.4329741036935513  | -0.43264482726673253 |   -0.000329276426818792   |
|   0.4375   | -0.44334011131185813 | -0.4407390337722479  |   -0.0026010775396102526  |
| 0.4453125  | -0.453741784005 4005 | -0.4488545887810704  |   -0.004887195224330121   |
|  0.453125  | -0.4641817732032497  | -0.4569927306304213  |    -0.00718904257282843   |
| 0.4609375  | -0.474662878990 0242 | -0.46515475173511855 |    -0.00950812725490563   |
```

0.46875	-0.4746628789900242	-0.4733420006778603	-0.0013208783121638756
0.4765625	-0.4851880575933403	-0.4815558844153116	-0.003632173178028719
0.484375	-0.4957604294153801	-0.48979787060866153	-0.0059625588067185875
0.4921875	-0.5063832876613545	-0.49806949008798357	-0.008313797573370973
0.5	-0.5063832876613545	-0.5063723394604588	-1.09482008957551e-05
0.5078125	-0.5170601076229666	-0.5147080838733125	-0.002352023749654153
0.515625	-0.5277945566809357	-0.5230784599431807	-0.004716096737754971
0.5234375	-0.5385905050973213	-0.5314852788645701	-0.007105226232751227
0.53125	-0.5494520376758835	-0.5399304297111084	-0.00952160796477508
0.5390625	-0.5494520376758835	-0.5484158829444215	-0.0010361547314620134
0.546875	-0.5603834663771486	-0.556943694146716	-0.0034397722304326805
0.5546875	-0.5713893439843545	-0.565516007994517	-0.005873335989837525
0.5625	-0.5824744789271732	-0.5741350624925132	-0.008339416434659963
0.5703125	-0.5936439513822456	-0.5828031934881205	-0.010840757894125086
0.578125	-0.5936439513822456	-0.5915228394891981	-0.0021211118930474226
0.5859375	-0.6049031307833049	-0.600296546809369	-0.004606583973935985
0.59375	-0.6162576948892725	-0.6091269750676164	-0.00713071982165614
0.6015625	-0.6277136505764589	-0.6180169030712963	-0.009696747505162584
0.609375	-0.6277136505764589	-0.6269692351144224	-0.0007444154620365184
0.6171875	-0.6392773565412306	-0.6359870077261074	-0.003290348815123245
0.625	-0.6509555481226054	-0.645073396907397	-0.005882151215208409
0.6328125	-0.6627553644806747	-0.6542317258984574	-0.008523638582217385
0.640625	-0.6746843783970772	-0.6634654735222307	-0.011218904874846447
0.6484375	-0.6746843783970772	-0.6727782831552935	-0.0019060952417836452
0.65625	-0.6867506289985957	-0.6821739723818154	-0.004576656616780306
0.6640625	-0.6989626577451122	-0.6916565433922836	-0.007306114352828685
0.671875	-0.711329548069524	-0.7012301941951138	-0.010099353874410189
0.6796875	-0.711329548069524	-0.7108993307165059	-0.00043021735301806974
0.6875	-0.7238609691109176	-0.7206685798720274	-0.003192389238890203
0.6953125	-0.7365672240446112	-0.7305428037025447	-0.0060244203420665166
0.703125	-0.7494593035851944	-0.7405271146774126	-0.008932188907781802
0.7109375	-0.7625489453233215	-0.7506268922794427	-0.011922053043878833
0.71875	-0.7625489453233215	-0.760847800999292	-0.0017011443240295687
0.7265625	-0.7758486996560453	-0.7711958098817706	-0.004652889774274693
0.734375	-0.789372003186719	-0.7816772137834153	-0.007694789403303703
0.7421875	-0.8031332606073361	-0.7922986565198237	-0.010834604087512467
0.75	-0.8031332606073361	-0.8030671561030495	-6.610450428667924e-05
0.7578125	-0.8171479362378107	-0.8139901322942273	-0.003157803943583337
0.765625	-0.8314326565882219	-0.8250754367250385	-0.006357219863183383
0.7734375	-0.8460053255377972	-0.8363313858742046	-0.009673939663592535
0.78125	-0.8608852539961902	-0.8477667972226102	-0.013118456773580034
0.7890625	-0.8608852539961902	-0.8593910289537033	-0.001494225042486863
0.796875	-0.8760933062381763	-0.8712140236154944	-0.004879282622681891
0.8046875	-0.8916520654944031	-0.8832463562178977	-0.008405709276505413
0.8125	-0.9075860218535924	-0.8954992873057502	-0.012086734547842148
0.8203125	-0.9239217861048924	-0.9079848216252243	-0.01593696447966808
0.828125	-0.9239217861048924	-0.9207157730915574	-0.0032060130133350073
0.8359375	-0.9406883338474487	-0.9337058368714409	-0.006982496976007724
0.84375	-0.9579172850489663	-0.9469696695169897	-0.010947615531976651
0.8515625	-0.975643225286259	-0.960522978233508	-0.015120247052751057
0.859375	-0.975643225286259	-0.9743826205346402	-0.0012606047516188212
0.86328125	-0.9939040762004008	-0.9814341417493754	-0.0124699344510254
0.8671875	-0.9939040762004008	-0.988569184916056	-0.005334891284344723
0.87109375	-1.0127415243145956	-0.9957902341808627	-0.016951290133732977
0.875	-1.0127415243145956	-1.0030998690953452	-0.009641655219250422
0.87890625	-1.0127415243145956	-1.0105007694123656	-0.00224075490223008
0.8828125	-1.0322015193825622	-1.0179957201805696	-0.014205799201992608
0.88671875	-1.0322015193825622	-1.0255876171595741	-0.006613902222988077
0.890625	-1.0523348559755403	-1.0332794725799783	-0.01905538339556201
0.89453125	-1.0523348559755403	-1.0410744212744232	-0.01126043470111715
0.8984375	-1.0523348559755403	-1.0489757272082532	-0.0033591287672871672
0.90234375	-1.0731978552316284	-1.0569867904408965	-0.016211064790731866
0.90625	-1.0731978552316284	-1.0651111545519125	-0.008086700679715841
0.91015625	-1.094853167789131	-1.0733525145687786	-0.02150065322035255
0.9140625	-1.094853167789131	-1.0817147254369424	-0.01313844235218875
0.91796875	-1.094853167789131	-1.0902018110764915	-0.004651356712639609
0.921875	-1.1173707241855628	-1.0988179740740298	-0.018552750111533012
0.92578125	-1.1173707241855628	-1.1075676060630508	-0.009803118122512
0.9296875	-1.1173707241855628	-1.1164552988513288	-0.0009154253342340546
0.93359375	-1.1408288658068177	-1.1254858563596555	-0.01534300944716227
0.9375	-1.1408288658068177	-1.1346643074427363	-0.006164558364081474
0.94140625	-1.1653156983406523	-1.143995919670276	-0.021319778670376266
0.9453125	-1.1653156983406523	-1.1534862141543667	-0.0118294841862856
0.94921875	-1.1653156983406523	-1.1631409815183182	-0.002174716822334055
0.953125	-1.1909307213527558	-1.1729662991121979	-0.017964422240557942
0.95703125	-1.1909307213527558	-1.182968549591701	-0.007962171761054737
0.9609375	-1.2177868030841326	-1.1931544409897528	-0.0246323620943798
0.96484375	-1.2177868030841326	-1.2035310284246081	-0.014255774659524434
0.96875	-1.2177868030841326	-1.214105737604445	-0.0036810654796874953
0.97265625	-1.2460125903166175	-1.224886390306756	-0.02112620000986154
0.9765625	-1.2460125903166175	-1.2358812320315804	-0.010131358285037084
0.98046875	-1.2757554712558463	-1.247098962051132	-0.02865650920471441
0.984375	-1.2757554712558463	-1.2585487661050756	-0.017206705150770718
0.98828125	-1.2757554712558463	-1.270240352021109	-0.0055151192347373534
0.9921875	-1.3071852478781207	-1.2821839885751685	-0.025001259302952272
0.99609375	-1.3071852478781207	-1.294390547945196	-0.012794699932924702
1.0	-1.3071852478781207	-1.3068715521577783	-0.0003136957203424551

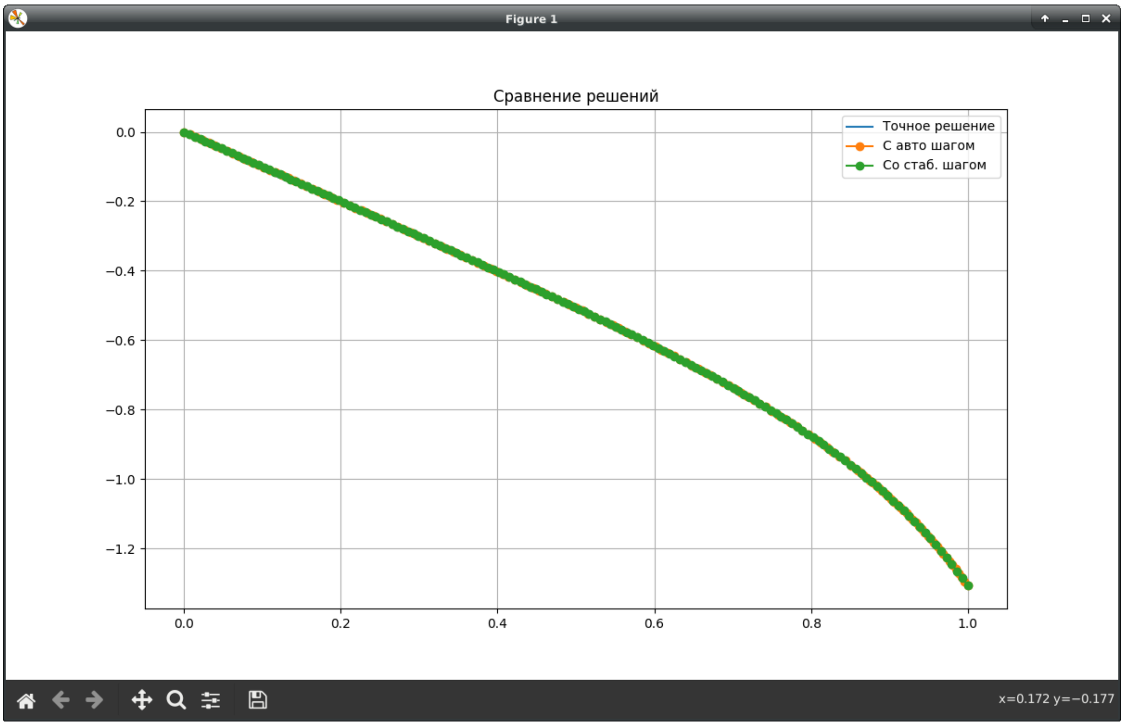
Таблица для фиксированного шага (h = 0.006802721088435374):

X_k	Y(X_k) (точное)	Y_k (численное)	Y(X_k) - Y_k (погрешность)

0.0	0.0	0.0	0.0
0.006802721088435374	-0.010000000020833335	-0.006802721090256429	-0.003197278930576906
0.013605442176870748	-0.020000000641666703	-0.013605442259425293	-0.00639455838224141
0.02040816326530612	-0.030000004862501307	-0.020408163929991657	-0.00959184093250965
0.027210884353741496	-0.030000004862501307	-0.027210887216442294	-0.0027891176460590125
0.03401360544217687	-0.04000002048335078	-0.034013614274910155	-0.0059864062084406255
0.04081632653061224	-0.050000062504296776	-0.04081634865282268	-0.009183713851474096
0.047619047619047616	-0.050000062504296776	-0.04761909563855879	-0.0023809668657379865
0.05442176870748299	-0.0600001555256716	-0.05442186261112228	-0.005578292914549324
0.061224489795918366	-0.07000033614852308	-0.06122465938984319	-0.008775676758679886
0.06802721088435373	-0.07000033614852308	-0.0680274985841232	-0.001972837564399879
0.0748299319727891	-0.08000065537561366	-0.07483039594324628	-0.0051702594323673745
0.08163265306122447	-0.0900011810133269	-0.08163337070628211	-0.008367810307044782
0.08843537414965984	-0.0900011810133269	-0.0884364459521162	-0.0015647350612107008
0.0952380952380952	-0.10000200007499951	-0.09523964894964844	-0.004762351125351075
0.10204081632653057	-0.11000322118636699	-0.10204301150821014	-0.007960209678156846
0.10884353741496594	-0.11000322118636699	-0.1088465703282584	-0.0011566508581085932
0.11564625850340131	-0.12000497699400454	-0.11565036735241671	-0.004354609641587834
0.12244897959183668	-0.13000742657786396	-0.12245445011694137	-0.007552976460922595
0.12925170068027206	-0.13000742657786396	-0.1292588721037043	-0.0007485544741596684
0.13605442176870744	-0.14001075786925002	-0.13606369309279542	-0.003947064776454595
0.14285714285714282	-0.15001519007584854	-0.1428689795158604	-0.007146210559988153
0.1496598639455782	-0.15001519007584854	-0.1496748048103032	-0.0003403852655453343
0.15646258503401358	-0.1600209761157115	-0.1564812497744982	-0.0035397263412133007
0.16326530612244897	-0.17002840506242473	-0.16328840292417054	-0.006740002138254181
0.17006802721088435	-0.18003780460402855	-0.1700963608501213	-0.009941443753907236
0.17687074829931973	-0.18003780460402855	-0.17690522857748944	-0.0031325760265391023
0.1836734693877551	-0.19004954351863562	-0.18371511992676184	-0.006334423591873778
0.1904761904761905	-0.20006403417009047	-0.19052615787676028	-0.009537876293330194
0.19727891156462588	-0.20006403417009047	-0.19733847492985487	-0.0027255592402355977
0.20408163265306126	-0.2100817350274453	-0.20415221347967363	-0.0059295215477716545
0.21088435374149664	-0.2201031532124871	-0.2109675261815992	-0.00913562703088791
0.21768707482993202	-0.2201031532124871	-0.2177845763263672	-0.0023185768861199163
0.2244897959183674	-0.23012884708004333	-0.22460353821710354	-0.005525308862939837
0.23129251700680278	-0.24015942883631863	-0.2314245975501627	-0.00873483128615593
0.23809523809523817	-0.24015942883631863	-0.2382479518001563	-0.0019114770361623457
0.24489795918367355	-0.2501955672010781	-0.2450738106095854	-0.005121756591492693
0.2517006802721089	-0.2602379901200915	-0.2519023961835221	-0.008335593936569408
0.2585034013605443	-0.2602379901200915	-0.25873394368981206	-0.0015040464302794598
0.26530612244897966	-0.27028748753489523	-0.26556870166530283	-0.004718785869592401
0.27210884353741505	-0.2803449142176139	-0.2724069324286348	-0.007937981788979143
0.2789115646258504	-0.2803449142176139	-0.27924891250016465	-0.0010960017174492798
0.2857142857142858	-0.2904111926793196	-0.2860949330296284	-0.004316259649691179
0.2925170068027212	-0.30048731616119245	-0.29294530023218734	-0.007542015929005108
0.2993197278911566	-0.30048731616119245	-0.2998003358335399	-0.0006869803276525754
0.30612244897959195	-0.31057435171859393	-0.3066603775248239	-0.003913974193770042
0.31292517006802734	-0.3206734434090716	-0.3135257794280769	-0.007147663980994701
0.3197278911564627	-0.3206734434090716	-0.3203969125730676	-0.0002765308360039853
0.3265306122448981	-0.33078581559629594	-0.3272741653863603	-0.0035116502099356572
0.3333333333333335	-0.3409127763829862	-0.33415794419352407	-0.006754832189462157
0.34013605442176886	-0.3510557211870276	-0.34104867373545367	-0.010007047451573936
0.34693877551020424	-0.3510557211870276	-0.34794679769982295	-0.003108923487204651
0.3537414965986396	-0.36121613647622186	-0.3548527792687543	-0.006363357207467557
0.360544217687075	-0.37139560367846064	-0.3617671016838492	-0.00962850199461146
0.3673469387755104	-0.37139560367846064	-0.3686902688297927	-0.0027053348486679174
0.37414965986394577	-0.3815958032855787	-0.37562280583781643	-0.005972997447762252
0.38095238095238115	-0.39181851917074295	-0.38256525971037814	-0.009253259460364816
0.38775510204081653	-0.39181851917074295	-0.38951819996849996	-0.0023003192022429952
0.3945578231292519	-0.4020656431409841	-0.3964822193232887	-0.005583423817695388
0.4013605442176873	-0.4123391797483931	-0.40345793437325633	-0.008881245375136793
0.4081632653061227	-0.4123391797483931	-0.410445986329153	-0.0018931934192401023
0.41496598639455806	-0.42264125138561065	-0.4174470417681305	-0.005194209617480128
0.42176870748299344	-0.4329741036935513	-0.4244617934191632	-0.008512310274388146
0.4285714285714288	-0.4329741036935513	-0.4314909609817726	-0.00148314271177874
0.4353741496598642	-0.44334011131185813	-0.43853529198022806	-0.0048048193316300725
0.4421768707482996	-0.4537417840054005	-0.4455955626555307	-0.008146221349869809
0.44897959183673497	-0.4537417840054005	-0.45267257889763357	-0.001069205107766924
0.45578231292517035	-0.4641817732032497	-0.45976717722050703	-0.004414595982742686
0.46258503401360573	-0.4746628789900242	-0.4668802257828243	-0.007782653207199897
0.4693877551020411	-0.4746628789900242	-0.47401262545722267	-0.0006502535328015147
0.4761904761904765	-0.4851880575933403	-0.481165310951289	-0.0040227466420513225
0.4829931972789119	-0.4957604294153801	-0.48833925198362527	-0.007421177431754855
0.48979591836734726	-0.4957604294153801	-0.49553545451857517	-0.00022497489680495253
0.49659863945578264	-0.5063832876613545	-0.502754962063433	-0.003628325597921589
0.503401360544218	-0.5170601076229666	-0.5099988570322167	-0.007061250590749912
0.5102040816326534	-0.5277945566809357	-0.5172682621803708	-0.010526294500564859
0.5170068027210888	-0.5277945566809357	-0.5245643421150639	-0.003230214565871803
0.5238095238095242	-0.5385905050973213	-0.5318883048860805	-0.006702200211240794
0.5306122448979596	-0.5494520376758835	-0.5392414036626583	-0.010210634013225195
0.5374149659863949	-0.5494520376758835	-0.5466249385020101	-0.0028270991738733597
0.5442176870748303	-0.5603834663771486	-0.5540402582156879	-0.006343208161460723
0.5510204081632657	-0.5713893439843545	-0.5614887623403999	-0.009900581643954576
0.5578231292517011	-0.5713893439843545	-0.5689719032203847	-0.0024174407639697693
0.5646258503401365	-0.5824744789271732	-0.5764911882089835	-0.005983290718189704
0.5714285714285718	-0.5936439513822456	-0.5840481819976341	-0.009595769384611486
0.5782312925170072	-0.5936439513822456	-0.5916445090811452	-0.001999442301100385
0.5850340136054426	-0.6049031307833049	-0.599281856368803	-0.005621274414501887
0.591836734693878	-0.6162576948892725	-0.606961975951616	-0.00929571893765646
0.5986394557823134	-0.6162576948892725	-0.6146866880368298	-0.0015710068524427134
0.6054421768707487	-0.6277136505764589	-0.6224578840617431	-0.005255766514715821

0.6122448979591841	-0.6392773565412306	-0.6302775299998457	-0.008999826541384981
0.6190476190476195	-0.6392773565412306	-0.638147669873373	-0.0011296866678576212
0.6258503401360549	-0.6509555481226054	-0.6460704294875611	-0.004885118635044261
0.6326530612244903	-0.6627553644806747	-0.6540480204031796	-0.008707344077495183
0.6394557823129257	-0.6627553644806747	-0.6620827441653471	-0.0006726203153276655
0.646258503401361	-0.6746843783970772	-0.6701769968082032	-0.004507381588873982
0.6530612244897964	-0.6867506289985957	-0.678333273656731	-0.008417355341864718
0.6598639455782318	-0.6867506289985957	-0.6865541744489305	-0.0001964545496652148
0.6666666666666672	-0.6989626577451122	-0.6948424088036336	-0.004120248941478666
0.6734693877551026	-0.711329548069524	-0.7032008020615705	-0.008128746007953436
0.680272108843538	-0.7238609691109176	-0.7116323015298541	-0.012228667581063446
0.6870748299319733	-0.7238609691109176	-0.7201399831628784	-0.003720985948039157
0.6938775510204087	-0.7365672240446112	-0.7287270587157713	-0.00784016532883991
0.7006802721088441	-0.7494593035851944	-0.7373968834100173	-0.012062420175177113
0.7074829931972795	-0.7494593035851944	-0.7461529641547399	-0.0033063394304545524
0.7142857142857149	-0.7625489453233215	-0.7549989683714311	-0.007549976951890414
0.7210884353741502	-0.7758486996560453	-0.7639387334747147	-0.011909966181330556
0.7278911564625856	-0.7758486996560453	-0.7729762770670718	-0.0028724225889734667
0.734693877551021	-0.789372003186719	-0.7821158079114319	-0.007256195275287092
0.7414965986394564	-0.8031332606073361	-0.7913617377522145	-0.011771522855121619
0.7482993197278918	-0.8031332606073361	-0.8007186940628905	-0.0024145665444456066
0.7551020408163271	-0.8171479362378107	-0.8101915338065299	-0.006956402431280817
0.7619047619047625	-0.8314326565882219	-0.8197853583052434	-0.011647298282978502
0.7687074829931979	-0.8314326565882219	-0.8295055293250492	-0.0019271272631726788
0.7755102040816333	-0.8460053255377972	-0.8393576864946732	-0.006647639043123932
0.7823129251700687	-0.8608852539961902	-0.8493477661903236	-0.011537487805866609
0.789115646258504	-0.8608852539961902	-0.8594820220337864	-0.0014032319624037726
0.7959183673469394	-0.8760933062381763	-0.8697670471685437	-0.006326259069632578
0.8027210884353748	-0.8916520654944031	-0.8802097984983176	-0.011442266996085526
0.8095238095238102	-0.8916520654944031	-0.8908176230948684	-0.0008344423995346739
0.8163265306122456	-0.9075860218535924	-0.9015982870074353	-0.005987734846157111
0.823129251700681	-0.9239217861048924	-0.9125600067354109	-0.011361779369481462
0.8299319727891163	-0.9239217861048924	-0.923711483659278	-0.00021030244561437428
0.8367346938775517	-0.9406883338474487	-0.935061941763192	-0.005626392084256637
0.8435374149659871	-0.9579172850489663	-0.9466211690267108	-0.011296116022255553
0.8503401360544225	-0.975643225286259	-0.9583995629140164	-0.01724366237224262
0.8571428571428579	-0.975643225286259	-0.9704081804477254	-0.005235044838533653
0.8639455782312933	-0.9939040762004008	-0.9826587934224356	-0.011245282777965215
0.8707482993197286	-1.0127415243145956	-0.9951639493921879	-0.017577574922407746
0.877551020408164	-1.0127415243145956	-1.0079370391580467	-0.0048044851565489655
0.8843537414965994	-1.0322015193825622	-1.0209923715894633	-0.011209147793098895
0.8911564625850348	-1.0523348559755403	-1.0343452567389226	-0.017989599236617693
0.8979591836734702	-1.0523348559755403	-1.0480120983571837	-0.004322757618356654
0.9047619047619055	-1.0731978552316284	-1.0620104970905695	-0.011187358141058867
0.9115646258503409	-1.094853167789131	-1.0763593658476167	-0.0184938019415144
0.9183673469387763	-1.094853167789131	-1.0910790590665302	-0.0037741087226008663
0.9251700680272117	-1.1173707241855628	-1.1061915179054667	-0.011179206280096166
0.9319727891156471	-1.1408288658068177	-1.121720433724755	-0.0191084320820627
0.9387755102040825	-1.1408288658068177	-1.1376914326463605	-0.003137433160457226
0.9455782312925178	-1.1653156983406523	-1.1541322844769173	-0.011183413863735003
0.9523809523809532	-1.1909307213527558	-1.1710731398863299	-0.01985758146642591
0.9591836734693886	-1.1909307213527558	-1.1885468004692816	-0.0023839208834741665
0.965986394557824	-1.2177868030841326	-1.2065890272138393	-0.011197775870293247
0.9727891156462594	-1.2460125903166175	-1.225238894000396	-0.0207736963162215
0.9795918367346947	-1.2460125903166175	-1.2445391941077746	-0.0014733962088429031
0.9863945578231301	-1.2757554712558463	-1.2645369093791616	-0.011218561876684685
0.9931972789115655	-1.3071852478781207	-1.2852837537868518	-0.021901494091268958
1.0000000000000009	-1.3404987275717841	-1.306836805747265	-0.0336619218245191

Графік:



Посилання на GitHub-репозиторій з вихідним кодом, звітом (.pdf, .odt) та output.txt: https://github.com/AlexValder/METHODS_Labs/tree/master/lab01

Висновки

Під час виконання цієї лабораторної работи мені вдалося познайомитися з методами Рунге-Кутта різного порядку точності та програмною реалізацією цих методів на ЕОМ.

Перелік використанних джерел

1. Бойко Л.Т. “Основи чисельних методів” Навчальний посібник, 2009.

Додаток. Код програми.

main.py

```
#
# y'(x) = f(x, y(x)),  x є [a, b] = [0, 1]
# y(x_0) = y_0
#
# h_0 = 0.5
# e_1 = 1e-6
#
# Variant 9
#
# f(x, y) = xy^3 - 1
# x_0 = 0
# y_0 = 0
#

import numpy as np
from typing import Callable, List, Tuple
import methods as mth, graphics as gr, more_precise as mp
```

```
def funct(x: float, y: float) -> float:
    return x*y**3 - 1


x_0: float = 0.0
y_0: float = 0.0
a: float = 0.0
b: float = 1.0
h_0: float = 0.5

accurate_step = 0.01

if __name__ == "__main__":

    auto_pairs : List[Tuple[float, float]] = [(x_0, y_0)]
    manual_pairs : List[Tuple[float, float]] = [(x_0, y_0)]

    xs_auto: List[float] = []
    ys_auto: List[float] = []
    xs_stable: List[float] = []
    ys_stable: List[float] = []

    auto_pairs = mth.get_auto_pairs(funct, x_0, y_0, a, b, h_0)
    stable_pairs = mth.get_stable_pairs(funct, x_0, y_0, a, b,
len(auto_pairs))

    print(f"Кол-во пар при автоматическом: {len(auto_pairs)}")

    for x, y in auto_pairs:
        xs_auto.append(x)
        ys_auto.append(y)

    for x, y in stable_pairs:
        xs_stable.append(x)
        ys_stable.append(y)

    def f(x: float) -> float:
        return mp.more_precise_value(funct, x_0, y_0, x, accurate_step)

    print("Таблица для автоматического шага:")
    gr.print_table(auto_pairs, f)
    print(f"Таблица для фиксированного шага (h = {(b -
a)/len(auto_pairs)}):")
    gr.print_table(stable_pairs, f)

    accurate_xs = [x for x in np.arange(0.0, 1.0 + accurate_step,
accurate_step)]
    accurate_ys = [f(x) for x in np.arange(0.0, 1.0 + accurate_step,
accurate_step)]

    gr.draw_graph((accurate_xs, accurate_ys, "Точное решение"), (xs_auto,
ys_auto, "С авто шагом"), (xs_stable, ys_stable, "Со стаб. шагом"))
```

methods.py

```
from typing import Callable, List, Tuple

# Automatic step
# h_0 = 0.5
#
# Variant 3 (Runge) // third
#
```

```
# e = 1e-3
#
# y_{i+1} = y_i + 1/6*(phi_0 + 4*phi_1 + phi_2)
# phi_0 = h * f(x_i, y_i)
# phi_1 = h * f(x_i + h/2, y_i + phi_0/2)
# phi_2 = h * f(x_i + h, y_i - phi_0 + 2*phi_1)
#

k: int = 3
eps_1: float = 1e-6
eps: float = 1e-3


def _runge(f: Callable[[float, float], float], x_i: float, y_i: float, h:
float) -> float:

    phi_0 = h * f(x_i, y_i)
    phi_1 = h * f(x_i + h/2, y_i + phi_0/2)
    phi_2 = h * f(x_i + h/2, y_i - phi_0 + 2*phi_1)

    return (phi_0 + 4*phi_1 + phi_2)/6


def get_stable_pairs(f: Callable[[float, float], float], x_0: float, y_0:
float, a: float, b: float, N: int) -> List[Tuple[float, float]]:

    h = (b - a) / N
    res_pairs: List[Tuple[float, float]] = [(x_0, y_0)]

    def _recursive(x_n: float, y_n: float) -> List[Tuple[float, float]]:

        if b - x_n <= eps_1:
            return res_pairs

        y_n += _runge(f, x_n, y_n, h)
        x_n += h

        res_pairs.append((x_n, y_n))

        return _recursive(x_n, y_n)

    return _recursive(x_0, y_0)


def get_auto_pairs(f: Callable[[float, float], float], x_0: float, y_0:
float, a: float, b: float, h_0: float) -> List[Tuple[float, float]]:

    res_pairs: List[Tuple[float, float]] = [(x_0, y_0)]

    x_n = x_0; y_n = y_0; h_n = h_0; h_n2 = h_0 / 2

    def _recursive(x_n: float, y_n: float, h_n: float, h_n2: float):

        if b - x_n < eps:
            return res_pairs
        elif abs(b - x_n) < h_n:
            h_n = abs(b - x_n)

        step_full = y_n + _runge(f, x_n, y_n, h_n)
        step_half = y_n + _runge(f, x_n + h_n2, y_n + _runge(f, x_n, y_n,
h_n2), h_n2)

        eps_full = abs((step_full - step_half) * 2**k / (2**k - 1))
```

```
        eps_half = abs((step_full - step_half) / (2**k - 1))

        if eps_half > eps:
            h_n = h_n2
            h_n2 /= 2
            return _recursive(x_n, y_n, h_n, h_n2)

        x_n += h_n
        y_n = step_full

        res_pairs.append((x_n, y_n))

        if eps_full <= eps:
            h_n2 = h_n
            h_n *= 2

        return _recursive(x_n, y_n, h_n, h_n2)

    return _recursive(x_n, y_n, h_n, h_n2)
```

more_precise.py

```
from typing import Callable, Tuple, List


def more_precise_list(f: Callable[[float, float], float], x_0: float, y_0:
float, x_end: float, h: float) -> List[Tuple[float, float]]:

    for_return : List[Tuple[float, float]] = [(x_0, y_0)]

    def _recursive(x_n: float, y_n: float) -> List[Tuple[float, float]]:

        k1 = h * f(x_n, y_n)
        k2 = h * f(x_n + h/2, y_n + k1/2)
        k3 = h * f(x_n + h/2, y_n + k2/2)
        k4 = h * f(x_n + h, y_n + k3)

        x_n += h
        y_n += 1/6 * (k1 + 2 * k2 + 2 * k3 + k4)

        for_return.append((x_n, y_n))

        if x_n + h >= x_end:
            return for_return
        else:
            return _recursive(x_n, y_n)

    return _recursive(x_0, y_0)


def more_precise_value(f: Callable[[float, float], float], x_0: float, y_0:
float, x: float, h: float) -> float:

    def _recursive(x_n: float, y_n: float) -> float:

        if x_n >= x:
            return y_n

        k1 = h * f(x_n, y_n)
        k2 = h * f(x_n + h/2, y_n + k1/2)
        k3 = h * f(x_n + h/2, y_n + k2/2)
        k4 = h * f(x_n + h, y_n + k3)

        x_n += h
```

```python
        y_n += (k1 + 2 * k2 + 2 * k3 + k4) / 6.0

        return _recursive(x_n, y_n)

    return _recursive(x_0, y_0)
```

graphics.py

```python
from prettytable import PrettyTable
import matplotlib.pyplot as plt
from typing import List, Tuple, Callable

# Table: x_k, y(x_k), y_k, y(x_k) - y_k,

def print_table(pairs: List[Tuple[float, float]], f: Callable[[float, float],
float]) -> None:
    t = PrettyTable(["X_k", "Y(X_k) (точное)", "Y_k (численное)", "Y(X_k) -
Y_k (погрешность)"])
    for x, y in pairs:
        t.add_row([f'{x}', f'{f(x)}', f'{y}', f'{f(x) - y}'])
    print(t)


def draw_graph(*input_points: Tuple[List[float], List[float], str]) -> None:
    plt.figure(figsize=(12, 7))
    plt.grid(True)

    plt.plot(input_points[0][0], input_points[0][1], '-',
label=input_points[0][2])
    for xy in input_points[1::]:
        plt.plot(xy[0], xy[1], 'o-', label=xy[2])
    plt.legend()
    plt.title("Сравнение решений")

    plt.show()
```