Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №5

Чисельні методи

Розв’язування диференціальних рівнянь

Варіант 17

Виконала
студентка групи ІТ-91:

Луцай К. А.

Перевірила:

ас. Тимофєєва Ю. С.

Київ 2021

Мета: навчитися розв’язувати задачу Коші за допомогою мови програмування Python, використати такі методи: Ейлера, Ейлера-Коші, Рунге-Кутта-Мерсона, Адамса, Адамса-Башфорта, Адамса-Мультона.

Теоретичні відомості:

Задачу Коші можна сформулювати так: нехай задано диференціальне рівняння з початковою умовою:

$$y' = f(x, y),$$
$$y(x_0) = y_0.$$

Треба знайти функцію y(x), що задовольняє як указане рівняння, так і початкову умову. Зазвичай числовий розв'язок цієї задачі отримують, обчислюючи спочатку значення похідної, а потім, задаючи малий приріст x і переходячи до нової точки x1 = x0 + h. Положення нової точки визначається нахилом кривої, обчисленим за допомогою диференціального рівняння. Таким чином, графік числового рішення є послідовністю коротких прямолінійних відрізків, якими апроксимується дійсна крива y = f(x). Сам числовий метод визначає порядок дій при переході від даної точки кривої до наступної.

Метод Ейлера:

Метод Ейлера оснований на розвиненні y в ряд Тейлора в околі точки x_0:

$$y(x_0 + h) = y(x_0) + hy'(x_0) + \frac{1}{2}h^2 y''(x_0) + \ldots$$

Якщо h мале, то члени, що містять h у другому або вищих степенях, є малими вищих порядків і ними можна нехтувати. Тоді $y(x_0 + h) = y(x_0) + hy'(x_0)$. Величину $y'(x_0)$ знаходимо з диференціального рівняння, підставивши в нього початкову умову. Ураховуючи, що $\Delta y = y_1 - y_0$, і замінюючи похідну на праву частину диференціального рівняння, отримуємо співвідношення $y_1 = y_0 + hf(x_0, y_0)$. Вважаючи тепер точку (x_1, y_1) початковою і повторюючи всі попередні міркування, можна знайти наступне наближене значення залежної змінної в точці (x_2, y_2). Цей процес можна продовжити, використовуючи співвідношення (що і є розрахунковою формулою методу Ейлера):

$$y_{n+1} = y_n + hf(x_n, y_n)$$

Метод Ейлера-Коші:

У цьому методі на кожному інтервалі розрахунок проводиться у два етапи. На першому (етап прогнозу) визначається наближений розв'язок на правому кінці інтервалу за методом Ейлера, на другому (етап корекції) уточнюється значення розв'язку на правому кінці, використовуючи напівсуми тангенсів кутів нахилу на кінцях інтервалу

$$\tilde{y}_{k+1} = y_k + hf(x_k, y_k)$$

$$y_{k+1} = y_k + \frac{h(f(x_k, y_k) + f(x_{k+1}, \tilde{y}_{k+1})}{2}$$

$$x_{k+1} = x_k + h$$

Метод Рунге-Кутта-Мерсона:

Родина явних методів Рунге-Кутти p-го порядку записується як сукупність формул:

$$y_{k+1} = y_k + \Delta y_k$$

$$\Delta y_k = \sum_{i=1}^{p} c_i K_i^k$$

$$K_i^k = hf(x_k + a_i h, y_k + h\sum_{j=1}^{i-1} b_{ij} K_j^k)$$

$$i = 2,3.....p$$

Параметри a_i, b_{ij}, c_i підбираються так, щоб значення y_{k+1}, розраховане за співвідношенням збігалося зі значенням розкладання в точці x_{k+1} точного розв'язку в ряд Тейлора з похибкою $O(h^{p+1})$.

Мерсон запропонував модифікацію методу Рунге – Кутта 4-го порядку, яка дозволяє оцінювати похибку на кожному кроці та приймати рішення про зміну кроку. Схему Мерсона за допомогою еквівалентних перетворень зведемо до виду:

$$y(x_0 + h) = y_0 + \frac{k_4 + k_5}{2} 0(h^5),$$

де

$$k_1 = h_3 f(x_0, y_0), h_3 = \frac{h}{3};$$

$$k_2 = h_3 f(x_0 + h_3, y_0 + k_1);$$

$$k_3 = h_3 f(x_0 + h_3, y_0 + \frac{k_1 + k_2}{2});$$

$$k_4 = k_1 + 4h_3 f(x_0 + \frac{h}{2}, y_0 + 0,375(k_1 + k_3);$$

$$k_5 = h_3 f(x_0 + h, y_0 + 1,5(k_4 - k_3)).$$

Метод Адамса:

Якщо використовується інтерполяційний многочлен 3-го степеня, побудований за значеннями підінтегральної функції в останніх чотирьох вузлах, то отримуємо метод Адамса четвертого порядку точності:

$$y_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}),$$

де f_k – значення підінтегральної функції у вузлі x_k.

Метод Адамса як і всі багатокрокові методи не стартує самостійно, тобто для того, щоб використовувати метод Адамса необхідно мати розв'язок у перших чотирьох вузлах. У вузлі x_0 розв'язок y_0 відомий з початкових умов, а в інших трьох вузлах x_1, x_2, x_3 розв'язки y_1, y_2, y_3 можна отримати за допомогою підхожого однокрокового методу, наприклад: методу Рунге-Кутти четвертого порядку

Метод Адамса-Башфорта-Мультона:

Цей метод типу предиктор-коректор дозволяє підвищити точність обчислень методу Адамса внаслідок подвійного обчислення значення функції $f(x,y)$ при визначенні y_{k+1} на кожному новому кроці по x.

Етап предиктор

Аналогічно методу Адамса за значеннями у вузлах $x_{k-3}, x_{k-2}, x_{k-1}, x_k$ розраховується "попереднє" значення розв'язку у вузлі x_{k+1}.

$$\hat{y}_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}),$$

За допомогою отриманого значення $\hat{y}_{k+1}$ розраховується "попереднє" значення функції $f_{k+1} = f(x_{k+1}, \hat{y}_{k+1})$ в новій точці.

Етап коректор

На коригувальному етапі за методом Адамса 4-го порядку за значеннями у вузлах $x_{k-2}, x_{k-1}, x_k, x_{k+1}$ розраховується "остаточне" значення розв'язку у вузлі x_{k+1}.

$$y_{k+1} = y_k + \frac{h}{24}(9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2}),$$

Розв'язок задачі в аналітичній формі:

$f(x) = x + 1 + e^x$

$zy'' - (x+1)y' + y = 0$

$y(1) = 2 + e$

$y'(1) = 1 + e$

h = 0.1

x є [1,2]

$$f(x,y) = \frac{(x+1)(1+e^x) - y}{x}$$

Метод Ейлера:

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

x = 1 y = 2 + e

x = 1.1 y = 2 + e + $\frac{(1+1)(1+e^1)-2-e}{10}$ = 2 + 1.1e

похибка: $e^{1.1} + 0.1 - 1.1e$

x = 1.2 y = 2 + 1.1e + $\frac{(1.1+1)(1+e^{1.1})-2-1.1e}{11} = \frac{2.1e^{1.1}+21.9}{11} + e$

похибка: $\frac{e^{1.2}-2.1e^{1.1}+2.3}{11} - e$

x = 1.3 y = $\frac{2.1e^{1.1}+21.9}{12}+\frac{11e}{12}+\frac{(1.2+1)(1+e^{1.2})}{12}=\frac{2.1e^{1.1}+24.1+2.2e^{1.2}}{12}+\frac{11e}{12}$

похибка: $\frac{e^{1.3}-2.1e^{1.1}+3.5-2.2e^{1.2}}{12}-\frac{11e}{12}$

x = 1.4 y = 6.042576

похибка: 0.412623

x = 1.5 y = 6.477569

похибка: 0.504120

x = 1.6 y = 6.959346

похибка: 0.593686

x = 1.7 y = 7.491755

похибка: 0.682193

x = 1.8 y = 8.079278

похибка: 0.770369

x = 1.9 y = 8.727041

похибка: 0.858853

x = 2 y = 9.440834

похибка: 0.948222

Метод Адамса (на основі Ейлера)

$$y_i = y_{i-1} + \frac{h}{24}(55f(x_{i-1}, y_{i-1}) - 59f(x_{i-2}, y_{i-2}) + 37f(x_{i-3}, y_{i-3}) - 9f(x_{i-4}, y_{i-4})$$

x = 1.4 y = $\frac{2.1e^{1.1}+24.1+2.2e^{1.2}}{12}+\frac{11e}{12}+$

$$\frac{1}{240}\left(55\left(\frac{(1.3+1)(1+e^{1.3})-\frac{2.1e^{1.1}+24.1+2.2e^{1.2}}{12}-\frac{11e}{12}}{1.3}\right)-59\left(\frac{(1.2+1)(1+e^{1.2})-\frac{2.1e^{1.1}+21.9}{11}-e}{1.2}\right)+37\left(\frac{(1.1+1)(1+e^{1.1})-2-1.1e}{1.1}\right)-9f\left(\frac{(1+1)(1+e^{1})-2-e}{1}\right)\right)=$$

$$\frac{16698e^{1.3}+19668.6e^{1.1}+83641.8-12342e^{1.2}+44338.8e}{11*12*13*24}$$

похибка: $e^{1.4}-\frac{16698e^{1.3}+19668.6e^{1.1}-15199.8-12342e^{1.2}+44338.8e}{11*12*13*24}$

x = 1.5 y = 6.518815

похибка: 0.462874

x = 1.6 y = 7.023083

похибка: 0.529949

x = 1.7 y = 7.577904

похибка: 0.596043

x = 1.8 y = 8.189612
похибка: 0.660036
x = 1.9 y = 8.863171
похибка: 0.722724
x = 2 y = 9.604970
похибка: 0.784086

Лістинги програм розв'язування задачі:

Початкові задані за умовою значення:

```
import matplotlib.pyplot as plt
import math
import numpy as np

def func17(x):
    return x + 1 + math.exp(x)

def derv17(x):
    return 1 + math.exp(x)

def kosh17(x, y):
    return ((x+1) * derv17(x) - y ) /x

a = 1
b = 2
h = 0.1
y0 = 2 + math.exp(1)
X = np.linspace(a, b, int((b-a)/h)+1)
```

В якості аргументів методи приймають масив точок x координат, точну функцію для перевірки, функцію для якої шукаємо значення, значення у першої точки, крок. Методи Адамса та Адамса-Мультона також приймають масив точок у координат.

```
def euler(X, func, kosh, y, h):
    print("Метод Ейлера")
    Y = np.zeros(len(X))
    Y[0] = y
    emax = 0

    for i in range(1, len(Y)):
        Y[i] = Y[i-1] + kosh(X[i-1], Y[i-1]) * h

        eps = math.fabs(func(X[i]) - Y[i])
        if eps > emax:
            emax = eps

    print("x".ljust(5),"y".ljust(20),"eps".ljust(20))
    for i in range(0, len(Y)):
        print(str(round(X[i], 2)).ljust(5), str(Y[i]).ljust(20), str(math.fabs(fu
nc(X[i]) - Y[i]) ).ljust(20))
    print("Максимальна похибка:", emax)

    return X, Y

def euler_kosh(X, func, kosh, y, h):
    print("Метод Ейлера-Коші")
    Y = np.zeros(len(X))
    Y[0] = y
```

```
    emax = 0

    for i in range(1, len(Y)):
        yTemp = Y[i-1] + kosh(X[i-1], Y[i-1]) * h
        Y[i] = Y[i-1] + (kosh(X[i-1], Y[i-1]) + kosh(X[i], yTemp)) * h / 2

        eps = math.fabs(func(X[i]) - Y[i])
        if eps > emax:
            emax = eps

    print("x".ljust(5),"y".ljust(20),"eps".ljust(20))
    for i in range(0, len(Y)):
        print(str(round(X[i], 2)).ljust(5), str(Y[i]).ljust(20), str(math.fabs(fu
nc(X[i]) - Y[i]) ).ljust(20))
    print("Максимальна похибка:", emax)

    return X, Y

def rung_kut_mers(X, func, kosh, y, h):
    print("Метод Рунге-Кутта-Мероснa")
    Y = np.zeros(len(X))
    Y[0] = y
    emax = 0

    for i in range(1, len(Y)):
        H3 = h/3
        K1 = H3 * kosh(X[i-1], Y[i-1])
        K2 = H3 * kosh(X[i-1] + H3, Y[i-1] + K1)
        K3 = H3 * kosh(X[i-1] + H3, Y[i-1] + (K1 + K2)/2)
        K4 = K1 + 4 * H3 * kosh(X[i-1] + h/2, Y[i-1] + 0.375 * (K1 + K3))
        K5 = H3 * kosh(X[i-1] + h, Y[i-1] + 1.5 * (K4 - K3))
        Y[i] = Y[i-1] + (K4 + K5) / 2

        eps = math.fabs(func(X[i]) - Y[i])
        if eps > emax:
            emax = eps

    print("x".ljust(5),"y".ljust(20),"eps".ljust(20))
    for i in range(0, len(Y)):
        print(str(round(X[i], 2)).ljust(5), str(Y[i]).ljust(20), str(math.fabs(fu
nc(X[i]) - Y[i]) ).ljust(20))
    print("Максимальна похибка:", emax)

    return X, Y

def adams(X, Y, func, kosh, y, h):
    print("Метод Адамса")
    emax = 0

    for i in range(4, len(Y)):
```

```
        Y[i] = Y[i-1] + h/24 * (55 * kosh(X[i-1], Y[i-1]) - 59 * kosh(X[i-
2], Y[i-2]) + 37 * kosh(X[i-3], Y[i-3]) - 9 * kosh(X[i-4], Y[i-4]))

        eps = math.fabs(func(X[i]) - Y[i])
        if eps > emax:
            emax = eps

    print("x".ljust(5),"y".ljust(20),"eps".ljust(20))
    for i in range(0, len(Y)):
        print(str(round(X[i], 2)).ljust(5), str(Y[i]).ljust(20), str(math.fabs(fu
nc(X[i]) - Y[i]) ).ljust(20))
    print("Максимальна похибка:", emax)

    return X, Y

def adams_bash(X, func, kosh, y, h):
    print("Метод Адамса-Башфорта")
    Y = np.zeros(len(X))
    Y[0] = y
    emax = 0

    for i in range(5, len(Y)):
        Y[i-4] = Y[i-5] + h * kosh(X[i-5], Y[i-5])
        Y[i-3] = Y[i-4] + h * ( 1.5 * kosh(X[i-4], Y[i-4]) - 0.5 * kosh(X[i-
5], Y[i-5]))
        Y[i-2] = Y[i-3] + h * ( 23/12 * kosh(X[i-3], Y[i-3]) - 4/3 * kosh(X[i-
4], Y[i-4]) + 5/12 * kosh(X[i-5], Y[i-5]))
        Y[i-1] = Y[i-2] + h * ( 55/24 * kosh(X[i-2], Y[i-2]) - 59/24 * kosh(X[i-
3], Y[i-3]) + 37/24 * kosh(X[i-4], Y[i-4]) - 3/8 * kosh(X[i-5], Y[i-5]))
        Y[i] = Y[i-1] + h * ( 1901/720 * kosh(X[i-1], Y[i-
1]) - 1387/360 * kosh(X[i-2], Y[i-2]) + 109/30 * kosh(X[i-3], Y[i-
3]) - 637/360 * kosh(X[i-4], Y[i-4]) + 251/720 * kosh(X[i-5], Y[i-5]))

        eps = math.fabs(func(X[i]) - Y[i])
        if eps > emax:
            emax = eps

    print("x".ljust(5),"y".ljust(20),"eps".ljust(20))
    for i in range(0, len(Y)):
        print(str(round(X[i], 2)).ljust(5), str(Y[i]).ljust(20), str(math.fabs(fu
nc(X[i]) - Y[i]) ).ljust(20))
    print("Максимальна похибка:", emax)

    return X, Y

def adams_mult(X, Y, func, kosh, y, h):
    print("Метод Адамса-Мультона")
    emax = 0

    for i in range(4, len(Y)):
```

```
        Y[i] = Y[i-1] + h/24 * (55 * kosh(X[i-1], Y[i-1]) - 59 * kosh(X[i-
2], Y[i-2]) + 37 * kosh(X[i-3], Y[i-3]) - 9 * kosh(X[i-4], Y[i-4]))
    for i in range(3, len(Y)):
        Y[i] = Y[i-1] + h/24 * (9 * kosh(X[i], Y[i]) + 19 * kosh(X[i-1], Y[i-
1]) - 5 * kosh(X[i-2], Y[i-2]) + kosh(X[i-3], Y[i-3]))

        eps = math.fabs(func(X[i]) - Y[i])
        if eps > emax:
            emax = eps

    print("x".ljust(5),"y".ljust(20),"eps".ljust(20))
    for i in range(0, len(Y)):
        print(str(round(X[i], 2)).ljust(5), str(Y[i]).ljust(20), str(math.fabs(fu
nc(X[i]) - Y[i]) ).ljust(20))
    print("Максимальна похибка:", emax)

    return X, Y
```

Виклик методів та графічне відображення результатів

```
xE, yE = euler(X, func17, kosh17, y0, h)
xEk, yEk = euler_kosh(X, func17, kosh17, y0, h)
xRkm, yRkm = rung_kut_mers(X, func17, kosh17, y0, h)
xA, yA = adams(X, yEk.copy(), func17, kosh17, y0, h)
xAb, yAb = adams_bash(X, func17, kosh17, y0, h)
xAm, yAm = adams_mult(X, yEk.copy(), func17, kosh17, y0, h)

x = np.linspace(1, 2, 1000)
y = np.exp(x) + x + 1
fig = plt.figure(figsize=(10, 10))
ax = plt.subplot(ylim=(4, 11), xlim=(0.9, 2.1))
ax.plot(x, y, color="black", label="e^x + x + 1", lw=1)
ax.plot(xE, yE, color="blue", label="euler", lw=1)
ax.plot(xEk, yEk, color="green", label="euler koshi", lw=1)
ax.plot(xRkm, yRkm, color="magenta", label="runge kutta mersona", lw=1)
ax.plot(xA, yA, color="aqua", label="adamsa", lw=1)
ax.plot(xAb, yAb, color="red", label="adamsa bashforta", lw=1)
ax.plot(xAm, yAm, color="orange", label="adamsa multona", lw=1)
plt.xticks([1 + h*i for i in range(int((b-a)/h) +1)])
ax.legend(fontsize=16)
ax.grid(True)
plt.title("f(x) = e^x + x + 1", fontsize=20)
plt.tight_layout();
plt.show()
```

Результати виконання програм:

Метод Ейлера

x	y	eps
1.0	4.718281828459045	0.0
1.1	4.9901100113049495	0.11405601264148402
1.2	5.300895342121546	0.21922158061500152
1.3	5.651175499446451	0.31812116817279357
1.4	6.04257602529859	0.41262394154608373
1.5	6.477569160664921	0.5041199096731432
1.6	6.959346061676937	0.5936863627181772
1.7	7.491754701786335	0.682192689940865
1.8	8.079278422720282	0.7703690416926641
1.9	8.727041449255614	0.858852993023655
2.0	9.44083368259005	0.9482224163406006

Максимальна похибка: 0.9482224163406006

Метод Ейлера-Коші

x	y	eps
1.0	4.718281828459045	0.0
1.1	5.009588585290295	0.09457743865613821
1.2	5.338498114862267	0.18161880787428064
1.3	5.706445935470733	0.26285073214851096
1.4	6.115694877792519	0.33950508905215493
1.5	6.56920530581543	0.41248376452263447
1.6	7.07057081730423	0.4824616070908849
1.7	7.623994365141944	0.5499530265852561
1.8	8.234291090912382	0.615356373500564
1.9	8.906910159363472	0.6789842829157973
2.0	9.647971205880344	0.7410848930503064

Максимальна похибка: 0.7410848930503064

Метод Рунге-Кутта-Мерosна

x	y	eps
1.0	4.718281828459045	0.0
1.1	5.008711533073731	0.09545449087270264
1.2	5.336783695645919	0.18333322709062827
1.3	5.703912208579582	0.26538445903966235
1.4	6.112343029925125	0.3428569369195493
1.5	6.565022659540414	0.4166664107976503
1.6	7.065532731253493	0.487499693141622
1.7	7.618065398328749	0.5558819933984509
1.8	8.22742565670014	0.6222218077128066
1.9	8.899052809335457	0.6868416329438123
2.0	9.63905663245498	0.7499994664756713

Максимальна похибка: 0.7499994664756713

Метод Адамса

x	y	eps
1.0	4.718281828459045	0.0
1.1	5.009588585290295	0.09457743865613821
1.2	5.338498114862267	0.18161880787428064
1.3	5.706445935470733	0.26285073214851096
1.4	6.114722111876405	0.34047785496826943
1.5	6.567334448947369	0.4143546213906957
1.6	7.0676774221463825	0.48535500224873207
1.7	7.620096564601893	0.5538508271253066
1.8	8.22933443146156	0.6203130329513868
1.9	8.900853376844612	0.6850410654346568
2.0	9.640752074447633	0.7483040244830175

Максимальна похибка: 0.7483040244830175

Метод Адамса-Башфорта

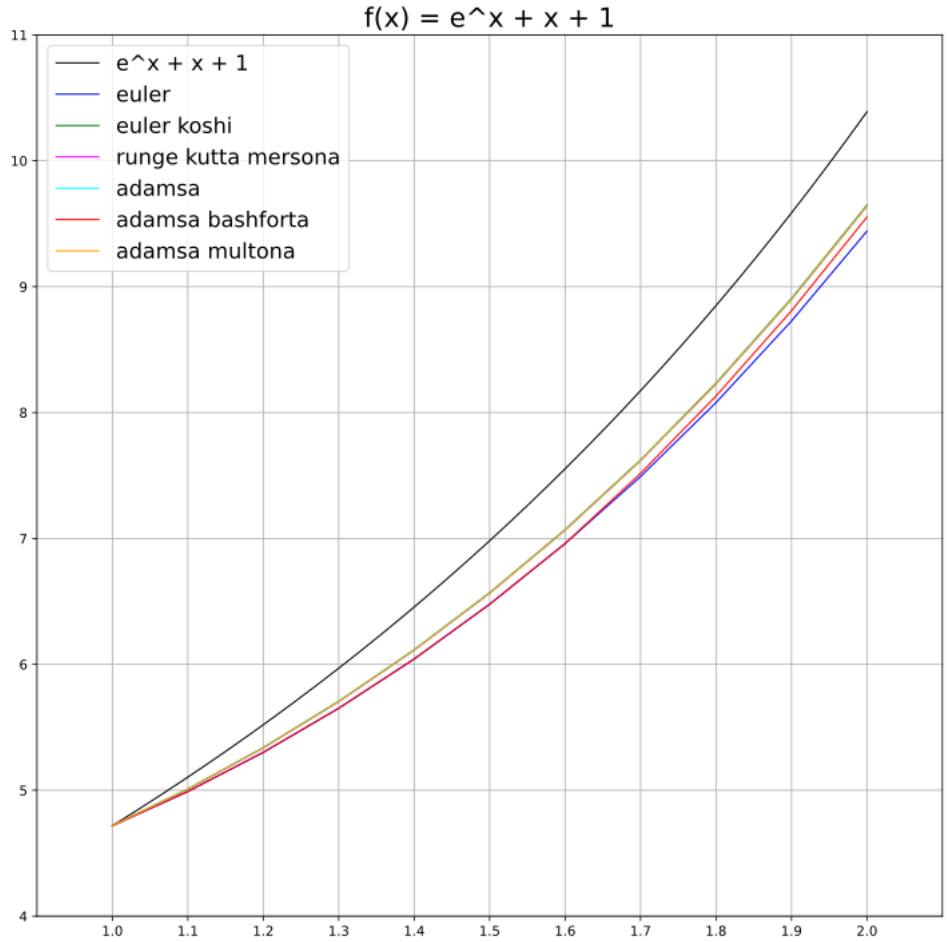
x	y	eps
1.0	4.718281828459045	0.0
1.1	4.9901100113049495	0.11405601264148402
1.2	5.300895342121546	0.21922158061500152
1.3	5.651175499446451	0.31812116817279357
1.4	6.04257602529859	0.41262394154608373
1.5	6.477569160664921	0.5041199096731432
1.6	6.959346061676937	0.5936863627181772
1.7	7.517070571335026	0.6568768203921742
1.8	8.13116565154002	0.7184818128729269
1.9	8.808478616040823	0.7774158262384461
2.0	9.552334975270531	0.8367211236601193

Максимальна похибка: 0.8367211236601193

Метод Адамса-Мультона

x	y	eps
1.0	4.718281828459045	0.0
1.1	5.009588585290295	0.09457743865613821
1.2	5.338498114862267	0.18161880787428064
1.3	5.705451501376787	0.263845166242457
1.4	6.1137483224083	0.34145164443637377
1.5	6.56630794251151	0.4153811278265547
1.6	7.066715379279478	0.4863170451156362
1.7	7.619158520216958	0.554788871510242
1.8	8.228440525328063	0.6212069390848836
1.9	8.899998771860888	0.6858956704183807
2.0	9.639941712452206	0.7491143864784444

Максимальна похибка: 0.7491143864784444



Висновки: було написано програму для розв’язання задачі Коші на мові Python, було реалізовано такі методи: Ейлера, Ейлера-Коші, Рунге-Кутта-Мерсона, Адамса, Адамса-Башфорта, Адамса-Мультона.