МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 2_1

Численные методы решения дифференциальных уравнений

ОТЧЕТ

о выполненном задании

студента 203-ей учебной группы факультета ВМК МГУ

Кудисова Артёма Аркадьевича

гор. Москва

2020 г.

Цель работы

Освоить методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задачи Коши для дифференциального уравнения (или системы дифференциальных уравнений) первого порядка.

Постановка задачи

Рассматривается обыкновенное дифференциальное уравнение первого порядка, разрешенное относительно производной и имеющее вид:

$$\frac{dy}{dx} = f(x, y), x_0 < x, \tag{1}$$

с дополнительным начальным условием, заданным в точке $x = x_0$:

$$y(x_0) = y_0. \tag{2}$$

Предполагается, что правая часть уравнения (1) функция $f = f(x, y)$ такова, что гарантирует существование и единственность решения задачи Коши (1)-(2).

В том случае, если рассматривается не одно дифференциальное уравнение вида (1), а система обыкновенных дифференциальных уравнений первого порядка, разрешенных относительно производных неизвестных функций, то соответствующая задача Коши имеет вид (на примере двух дифференциальных уравнений):

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2) \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2), \ x > x_0 \end{cases} \tag{3}$$

Дополнительные (начальные) условия задаются в точке $x = x_0$:

$$y_1(x_0) = y_1^{(0)}, \ y_2(x_0) = y_2^{(0)}. \tag{4}$$

Также предполагается, что правые части уравнений из (3) заданы так, что это гарантирует существование и единственность решения задачи Коши (3)-(4), но уже для системы обыкновенных дифференциальных уравнений первого порядка в форме, разрешенной относительно производных неизвестных функций.

Основные цели

1. Решить задачу Коши (1)-(2) (или (3)-(4)) наиболее известными и широко используемыми на практике методами Рунге-Кутта

второго и четвертого порядка точности, аппроксимировав дифференциальную задачу соответствующей разностной схемой (на равномерной сетке); полученное конечно-разностное уравнение (или уравнения в случае системы), представляющее фактически некоторую рекуррентную формулу, просчитать численно;

2. Найти численное решение задачи и построить его график;
3. Найденное численное решение сравнить с точным решением дифференциального уравнения (подобрать специальные тесты, где аналитические решения находятся в классе элементарных функций, при проверке можно использовать ресурсы on-line системы http://www.wolframalpha.com или пакета Maple и т.п.).

Метод Рунге-Кутта

Рассмотрим задачу Коши для дифферениального уравнения

$$u'(x) = f(x, u(x))$$
$$u(x_0) = u_0$$

Пусть решение $u(x)$ имеет производные достаточно высокого порядка. Выпишем для него в таком случае разложение по формуле Тейлора

$$u_{i+1} = u(x_i + h) = u_i + u'(x_i)h + \tfrac{1}{2}u''(x_i)h^2 + \tfrac{1}{6}u'''(x_i)h^3 + ...$$

и оборвем его на члене порядка h^2.

$u'(x) = f(x, u(x))$
$u''(x) = f'(x, u(x)) = \frac{\partial f}{\partial x}(x, u(x)) + \frac{\partial f}{\partial u}(x, u(x))u'(x) = \frac{\partial f}{\partial x}(x, u(x)) + + \frac{\partial f}{\partial u}(x, u(x))f(x, u(x))$

Тогда, $u_{i+1} = u_i + f(x_i, u_i)h + \frac{1}{2}h^2[\ \frac{\partial f}{\partial x}(x_i, u_i) + \frac{\partial f}{\partial u}(x_i, u_i)f(x_i, u_i)]$

Положим, что $f(x_i, u_i) + \frac{1}{2}h[\ \frac{\partial f}{\partial x}(x_i, u_i) + \frac{\partial f}{\partial u}(x_i, u_i)f(x_i, u_i) = \beta f(x_i, u_i) + +\alpha f(x_i + \gamma h, u_i + \psi h)]h + \mathbb{O}(h^2)$

Разложим функцию $f(x_i + \gamma h, u_i + \psi h)$ по степеням h, подставим в уравнение выше и приравняем слева и справа члены, содержащие и не содержащие h. Получим 3 уравнения, связывающих параметры $\alpha, \beta, \gamma, \psi$:

$$\alpha + \beta = 1, \alpha\gamma = \frac{1}{2}, \alpha\psi = \frac{1}{2}f(x_i, u_i)$$
или же
$$\beta = 1 - \alpha, \gamma = \frac{1}{2\alpha}, \psi = \frac{1}{2\alpha}f(x_i, u_i)$$

В итоге, отбросив члены порядка $\mathbb{O}(h^2)$, имеем следующее рекуррентное соотношение:
$u_{i+1} = u_i + h[(1-\alpha)f(x_i, u_i) + \alpha f(x_i + \frac{h}{2\alpha}, u_i + \frac{h}{2\alpha}f(x_i, u_i))]$

При $\alpha = \frac{1}{2}$ (что является наиболее предпочтительным значением параметра) рекуррентная формула принимает вид

$$u_{i+1} = u_i + \frac{h}{2}[f(x_i, u_i) + f(x_i + h, u_i + hf(x_i, u_i))]$$

Это удобная формула, однако более точные результаты дает метод Рунге-Кутта 4-ого порядка точности

$$u_{i+1} = u_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

где $k_1 = f(x_i, u_i)$, $k_2 = f(x_i + \frac{h}{2}, u_i + \frac{h}{2}k_1)$,
$k_3 = f(x_i + \frac{h}{2}, u_i + \frac{h}{2}k_2)$, $k_4 = f(x_i + h, u_i + hk_3)$

Описание программы

Программа написана на языке python без использования посторонних библиотек.

Структура:

1. def f0(x: float, y: list) -> float

   $y = [y]$

   Вычисляет значение функции $-y - x^2$

2. def f1(x: float, y: list) -> float

   $y = [y]$

   Вычисляет значение функции $3 - y - x$

3. def f2(x: float, y: list) -> float

   $y = [y]$

   Вычисляет значение функции $y - yx$

4. def f3(x: float, y: list) -> float

   $y = [u, v]$

   Вычисляет значение функции $-2xu^2 + v^2 - x - 1$

5. def f4(x: float, y: list) -> float

   $y = [u, v]$

   Вычисляет значение функции $\frac{1}{v^2} - u - \frac{x}{u}$

6. def runge_kutta2(x: float, y: list, count_y: int,
   f: list, h: float) -> list

   Выполняет 1 шаг метода Рунге-Кутта 2-ого порядка точности.

   Аргументы:

   - x - значение x
   - y - список значений неизвестных функций
   - count_y - количество неизвестных функций
   - f - список функций f из правой части уравнений (1), (3)
   - h - размер шага

7. def runge_kutta4(x: float, y: list, count_y: int,
   f: list, h: float) -> list

   Выполняет 1 шаг метода Рунге-Кутта 4-ого порядка точности.

   Аргументы:

   - x - значение x
   - y - список значений неизвестных функций
   - count_y - количество неизвестных функций
   - f - список функций f из правой части уравнений (1), (3)
   - h - размер шага

8. def main()

   Главная функция, считывающая номер теста, формирующая входные данные, запускающая выполнение метода Рунге-Кутта и выводящая ответ.

Код программы

```
def f0(x: float, y: list) -> float:
    return -y[0] - x**2


def f1(x: float, y: list) -> float:
    return 3 - y[0] - x


def f2(x: float, y: list) -> float:
    return y[0] - y[0] * x


def f3(x: float, y: list) -> float:
    return -2 * x * y[0] ** 2 + y[1] ** 2 - x - 1


def f4(x: float, y: list) -> float:
    return 1 / (y[1] ** 2) - y[0] - x / y[0]


def runge_kutta2(x: float, y: list, count_y: int,
                 f: list, h: float) -> list:
    y_ = [y[i] + f[i](x, y) * h for i in range(count_y)]
    return [y[i] + (f[i](x, y) + f[i](x + h, y_)) / 2 * h
            for i in range(count_y)]


def runge_kutta4(x: float, y: list, count_y: int,
                 f: list, h: float) -> list:
    k1 = [f[i](x, y) for i in range(count_y)]

    tmp_y = [y[i] + h / 2 * k1[i] for i in range(count_y)]
    k2 = [f[i](x + h / 2, tmp_y) for i in range(count_y)]

    tmp_y = [y[i] + h / 2 * k2[i] for i in range(count_y)]
    k3 = [f[i](x + h / 2, tmp_y) for i in range(count_y)]

    tmp_y = [y[i] + h * k3[i] for i in range(count_y)]
    k4 = [f[i](x + h, tmp_y) for i in range(count_y)]
    return [y[i] + h / 6 * (k1[i] + 2 * k2[i] + 2 * k3[i] + k4[i])
            for i in range(count_y)]


def main():
    try:
        print("Test 1:\ny' = -y + x^2\ny(0) = 10")
```

```
    print("\nTest 2:\ny' = 3 - y - x\ny(0) = 0")
    print("\nTest 3:\ny' = y - yx\ny(0) = 5")
    print("\nTest 4:\nu' = -2x(u^2) + v^2 - x - 1\n"
          "v' = 1/(v^2) - u - x/u\nu(0) = 1\nv(0) = 1\n")
    test = int(input("Choose the test (1, 2, 3, or 4): "))
    print()

    steps_number = int(input("Enter the steps number: "))
    step_size = float(input("Enter the size of step: "))
    print()

    if test not in [1, 2, 3, 4]:
        raise ValueError

except ValueError:
    print("Wrong input")
    return

if test == 1:
    x, f, count_y = 0, [f0], 1
    y2, y4 = [10], [10]
    print("x\t\ty (RK-2)\t\ty (RK-4)")
    print(f"{x:.3f}\t{y2[0]:.10f}\t{y4[0]:.10f}")
elif test == 2:
    x, f, count_y = 0, [f1], 1
    y2, y4 = [0], [0]
    print("x\t\ty (RK-2)\t\ty (RK-4)")
    print(f"{x:.3f}\t{y2[0]:.10f}\t{y4[0]:.10f}")
elif test == 3:
    x, f, count_y = 0, [f2], 1
    y2, y4 = [5], [5]
    print("x\t\ty (RK-2)\t\ty (RK-4)")
    print(f"{x:.3f}\t{y2[0]:.10f}\t{y4[0]:.10f}")
else:
    x, f, count_y = 0, [f3, f4], 2
    y2, y4 = [1, 1], [1, 1]
    print("x\t\ty_1 (RK-2)\t\ty_2 (RK-2)"
          "\t\ty_1 (RK-4)\t\ty_2 (RK-4)")
    print(f"{x:.3f}\t{y2[0]:.10f}\t{y2[1]:.10f}"
          f"\t{y4[0]:.10f}\t{y4[1]:.10f}")

for step in range(steps_number):
    y2 = runge_kutta2(x, y2, count_y, f, step_size)
    y4 = runge_kutta4(x, y4, count_y, f, step_size)
    x += step_size

    print(f"{x:.3f}\t", end="")
    for i in range(count_y):
```

```
            print(f"{y2[i]:.10f}\t", end="")
        for i in range(count_y):
            print(f"{y4[i]:.10f}\t", end="")
        print()


if __name__ == '__main__':
    main()
```

Тестирование

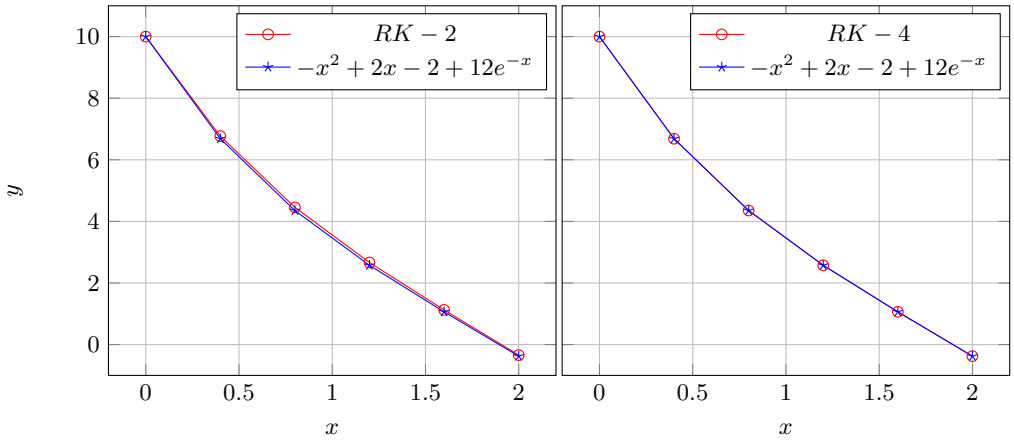
Таблица 1 - 3, таблица 2 - 12
Для проверки решений использовался ресурс www.wolframalpha.com

1. $\begin{cases} y' = -y - x^2 \\ y(0) = 10 \end{cases}$

   Аналитическое решение: $y = -x^2 + 2x - 2 + 12e^{-x}$

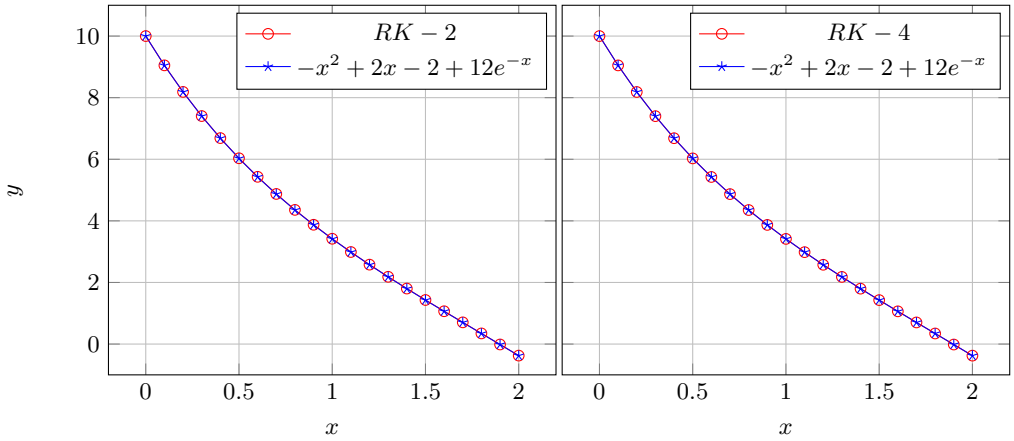
   - 5 шагов размером 0.4

x	y (RK-2)	y (RK-4)	y
0.000	10.0000000000	10.0000000000	10.0000000000
0.400	6.7680000000	6.6845866667	6.6838405524
0.800	4.4550400000	4.3528775680	4.3519475694
1.200	2.6646272000	2.5751717883	2.5743305429
1.600	1.1271464960	1.0633978335	1.0627582159
2.000	-0.3407403827	-0.3755674257	-0.3759766012



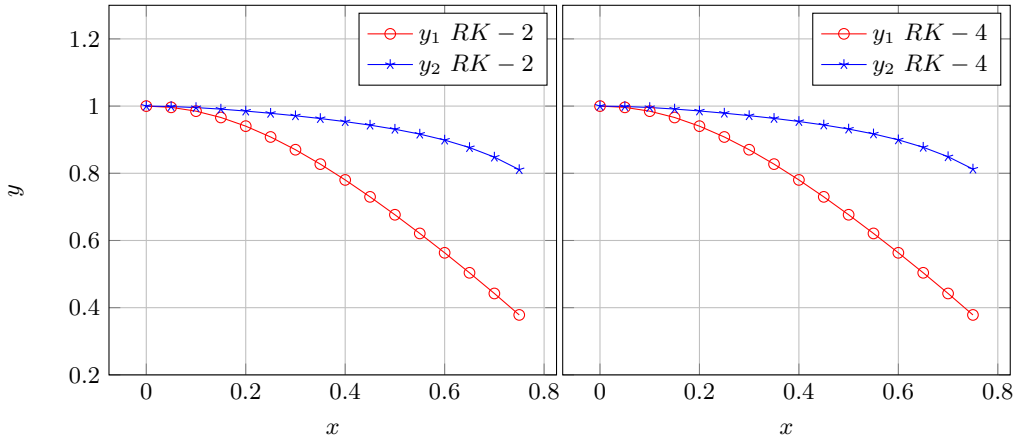
   - 20 шагов размером 0.1

x	y (RK-2)	y (RK-4)	y
0.000	10.0000000000	10.0000000000	10.0000000000
0.100	9.0495000000	9.0480497917	9.0480490164
0.200	8.1873475000	8.1847704200	8.1847690369
0.300	7.4032494875	7.3998204966	7.3998186482
0.400	6.6878907862	6.6838427453	6.6838405524
0.500	6.0328411615	6.0283703517	6.0283679166
0.600	5.4304712512	5.4257422248	5.4257396331
0.700	4.8738764823	4.8690263220	4.8690236455
0.800	4.3568082165	4.3519502713	4.3519475694
0.900	3.8736114359	3.8688385952	3.8688359169
1.000	3.4191683495	3.4145559091	3.4145532941
1.100	2.9888473563	2.9844555241	2.9844530044
1.200	2.5784568575	2.5743329419	2.5743305429
1.300	2.1842034560	2.1803837750	2.1803815164
1.400	1.8026541277	1.7991656707	1.7991635673
1.500	1.4307019855	1.4275638592	1.4275619218
1.600	1.0655352969	1.0627599801	1.0627582159
1.700	0.7046094437	0.7022038752	0.7022022886
1.800	0.3456215466	0.3435880656	0.3435866587
1.900	-0.0135125004	-0.0151753420	-0.0151765693
2.000	-0.3746788128	-0.3759755519	-0.3759766012



2. $$\begin{cases} u' = -2xu^2 + v^2 - x - 1 \\ v' = \frac{1}{v^2} - u - \frac{x}{u} \\ u(0) = 1 \\ v(0) = 1 \end{cases}$$

x	y_1 (RK-2)	y_2 (RK-2)	y_1 (RK-4)	y_2 (RK-4)
0.000	1.0000000000	1.0000000000	1.0000000000	1.0000000000
0.050	0.9962500000	0.9987500000	0.9962203644	0.9988497195
0.100	0.9848969493	0.9955796596	0.9848566351	0.9957619191
0.150	0.9661099923	0.9909506097	0.9660735387	0.9911986501
0.200	0.9402984577	0.9852094858	0.9402752932	0.9855080638
0.250	0.9080599833	0.9785828440	0.9080544294	0.9789194414
0.300	0.8701174944	0.9711727746	0.8701294313	0.9715387852
0.350	0.8272527027	0.9629530412	0.8272787167	0.9633448517
0.400	0.7802429133	0.9537644755	0.7802776681	0.9541844445
0.450	0.7298059233	0.9433071731	0.7298434932	0.9437646224
0.500	0.6765553008	0.9311255199	0.6765902553	0.9316380015
0.550	0.6209658930	0.9165796636	0.6209939893	0.9171750027
0.600	0.5633473328	0.8987924759	0.5633657155	0.8995125500
0.650	0.5038215844	0.8765514161	0.5038283771	0.8774596895
0.700	0.4422988841	0.8481225296	0.4422919167	0.8493201472
0.750	0.3784440943	0.8108774100	0.3784181000	0.8125410401



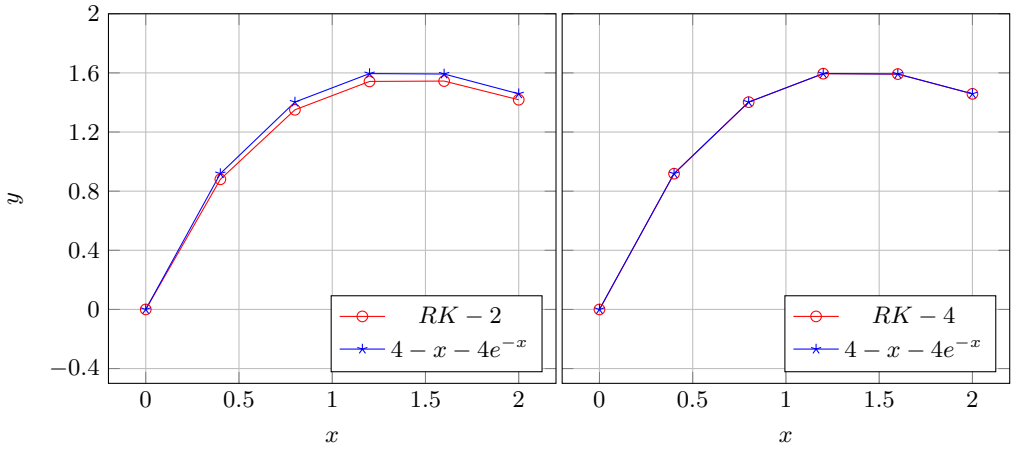
Полученное решение системы совпадает с аналитически найденными.

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

Аналитическое решение: $y = 4 - x - 4e^{-x}$

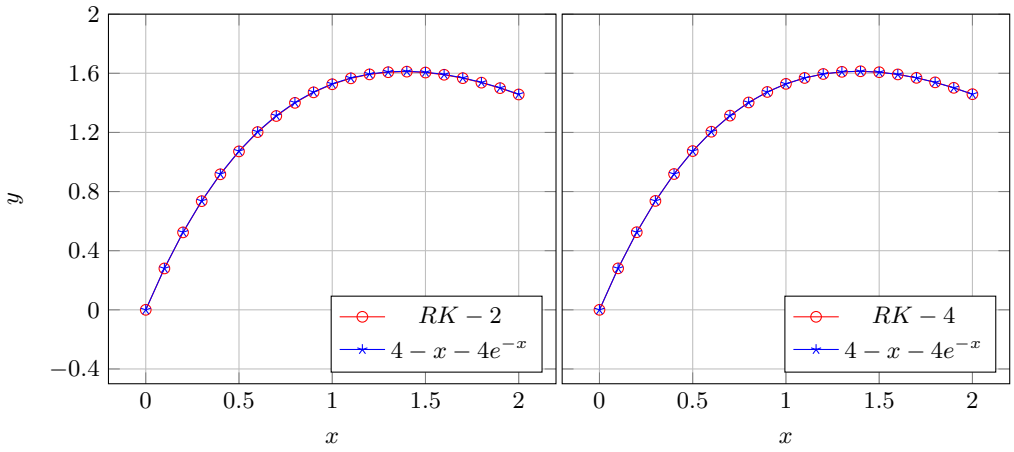
- 5 шагов размером 0.4

x	y (RK-2)	y (RK-4)	y
0.000	0.0000000000	0.0000000000	0.0000000000
0.400	0.8800000000	0.9184000000	0.9187198159
0.800	1.3504000000	1.4022553600	1.4026841435
1.200	1.5422720000	1.5947919933	1.5952231524
1.600	1.5447449600	1.5920285523	1.5924139280
2.000	1.4184265728	1.4583359415	1.4586588671



- 20 шагов размером 0.1

x	y (RK-2)	y (RK-4)	y
0.000	0.0000000000	0.0000000000	0.0000000000
0.100	0.2800000000	0.2806500000	0.2806503279
0.200	0.5239000000	0.5250763944	0.5250769877
0.300	0.7351295000	0.7367263120	0.7367271173
0.400	0.9167921975	0.9187188443	0.9187198159
0.500	1.0716969387	1.0738762623	1.0738773611
0.600	1.2023857296	1.2047522625	1.2047534556
0.700	1.3111590852	1.3136575253	1.3136587848
0.800	1.4000989722	1.4026828411	1.4026841435
0.900	1.4710895698	1.4737200352	1.4737213610
1.000	1.5258360607	1.5284809024	1.5284822353
1.100	1.5658816349	1.5685143385	1.5685156652
1.200	1.5926228796	1.5952218427	1.5952231524
1.300	1.6073237060	1.6098715441	1.6098728279
1.400	1.6111279540	1.6136108933	1.6136121442
1.500	1.6050707983	1.6074781467	1.6074793594
1.600	1.5900890725	1.5924127575	1.5924139280
1.700	1.5670306106	1.5692647785	1.5692659038
1.800	1.5366627026	1.5388033690	1.5388044471
1.900	1.4996797458	1.5017244934	1.5017255231
2.000	1.4567101700	1.4586578863	1.4586588671



Вывод

В ходе работы был рассмотрен метод Рунге-Кутта 2-ого и 4-ого порядков точности, применяемый для численного решения задачи Коши дифференциального уравнения (или системы дифференциальных уравнений) первого порядка.

На конкретных тестах была доказана работоспособность написанной программы и показано, что точность метода Рунге-Кутта напрямую зависит от выбранного размера шага.

Важно заметить, что метод Рунге-Кутта 4-ого порядка несколько точнее, чем 2-ого (тесты 1 и 3, даже при большом шаге метод Рунге-Кутта 4-ого порядка сохранял относительно высокую точность), однако релизация его сложнее. Так на каждом шаге метода Рунге-Кутта 2-ого порядка функцию $f(x, y)$ приходилось вычислять дважды, в то время как метод Рунге-Кутта 4-ого порядка на каждом своем шаге требует вычисление функции $f(x, y)$ 4 раза.