**Лабораторная работа №3**

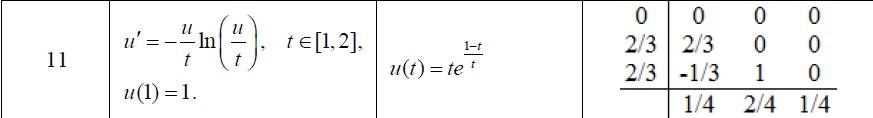
**Численные методы решения задачи Коши для ОДУ**

**Вариант 11**

**Чеботаревский Никита**

**3 курс, 8 группа**

**Постановка задачи**

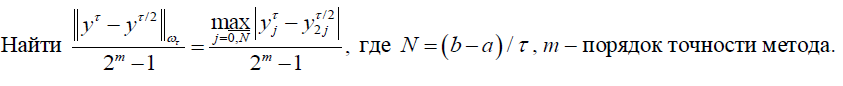
****

На равномерной сетке с помощью неявного метода трапеций и метода, указанного в варианте задания, найти с шагами 0.1 и / 2 0.05 численное решение задачи Коши и соответственно. Для неявного метода трапеций решение уравнений выполнить с помощью методаНьютона. Определить порядок точности явного метода Рунге-Кутта.

Сравнить найденное численное решение с точным решением *u*(*t*) , т.е. найти



В одной системе координат построить график функции *u*(*t*) и график полученного численного решения



По результатам лабораторной работы оформляется отчёт, в котором должна быть приведена следующая информация:

1) постановку задачи;

краткие теоретические сведения (записать методы, применяемые для численного решения поставленной задачи Коши);

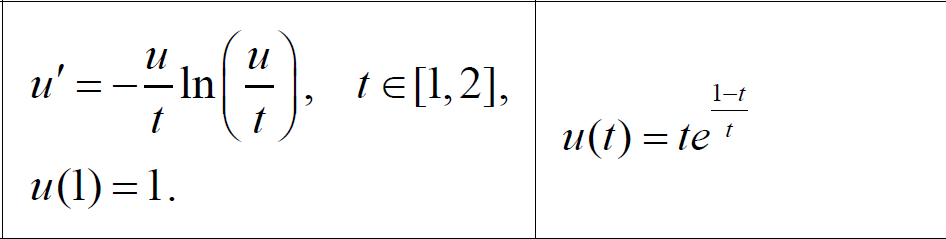
численные результаты;

выводы;

листинг программы с комментариями.

**Теоретические сведения:**

У нас есть следующая задача Коши:



**Неявный метод трапеций:**

Итак, пусть на отрезке *t* ≤ *t* ≤ *T* 0 требуется найти решение u(t) дифференциального уравнения:



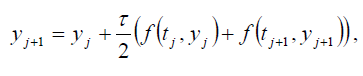
удовлетворяющего начальному условию:



Проинтегрировав данное уравнение по отрезку [], получим равенство



Воспользуемся для замены интеграла формулой трапеций. Из-за чего получим одношаговый метод численного интегрирования (неявный метод трапеций)



который является методом второго порядка.

Также для нахождения нам придётся решить следующее нелинейное уравнение:

Т.е уравнение имеет вид:

, которое необходимо решить относительно .

Производная для данного уравнения по имеет вид:

**Метод Ньютона с постоянной производной**:

Для наилучшей сходимости в точке очередного приближения должно выполняться условие  Решение данного уравнения ищут в виде:



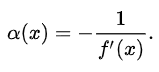
Тогда:



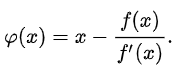
В предположении, что точка приближения достаточно близка к корню x,

И что заданная функция непрерывна, то окончательная формула для

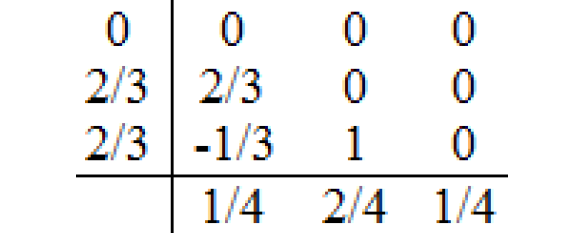
примет вид:

**

С учётом этого формула для примет вид:

**

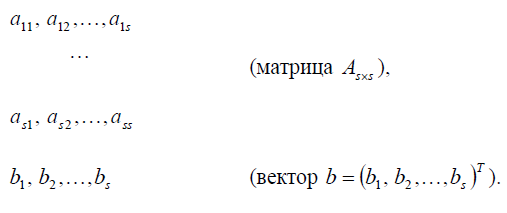
**Явный метод Рунге-Кутта:**

****

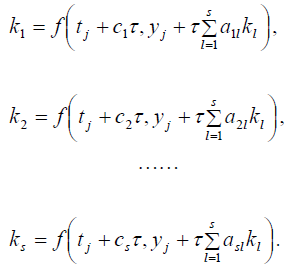


Зададим три набора параметров:





С помощью первых двух наборов построим величины:



Откуда получим потом явный метод:

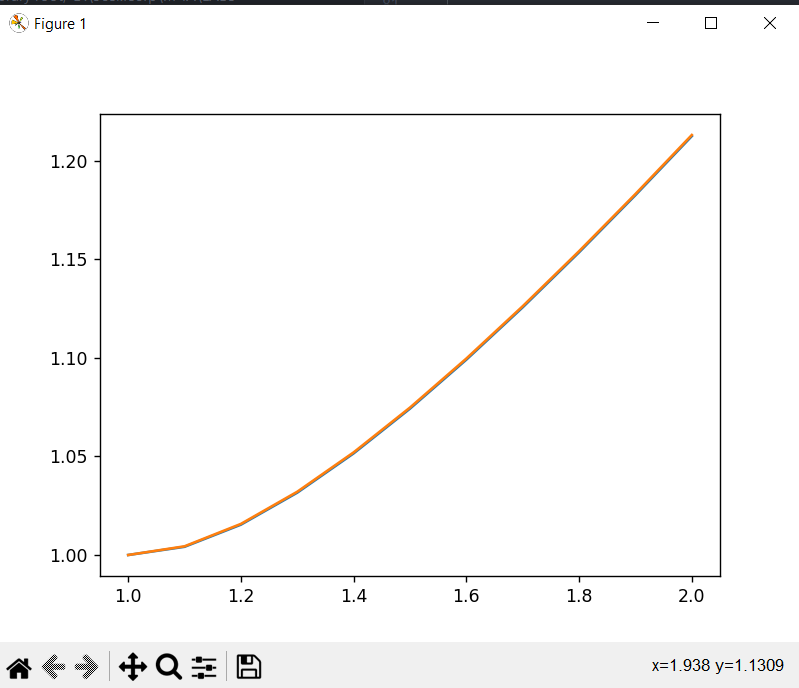
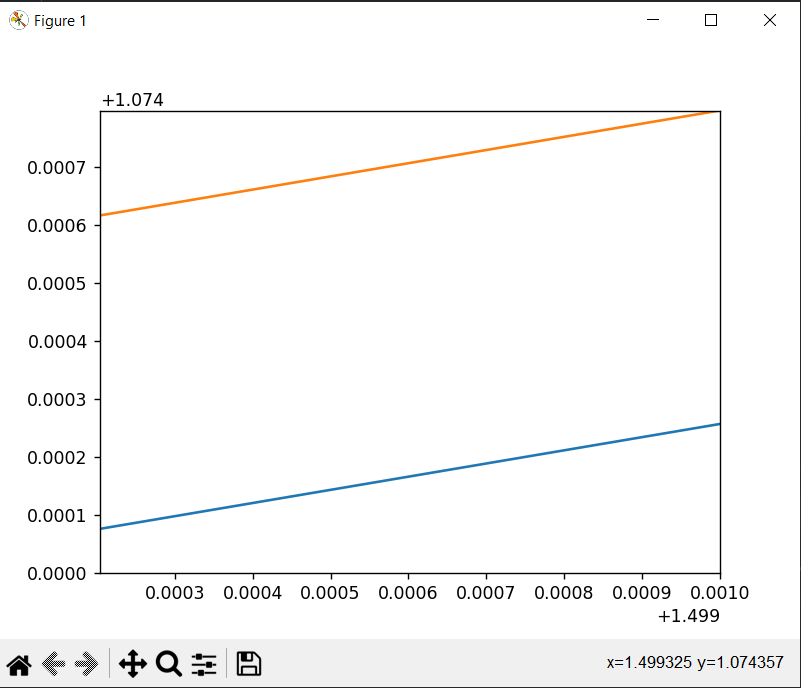


Итак, наш явный метод Рунге-Кутта имеет вид:

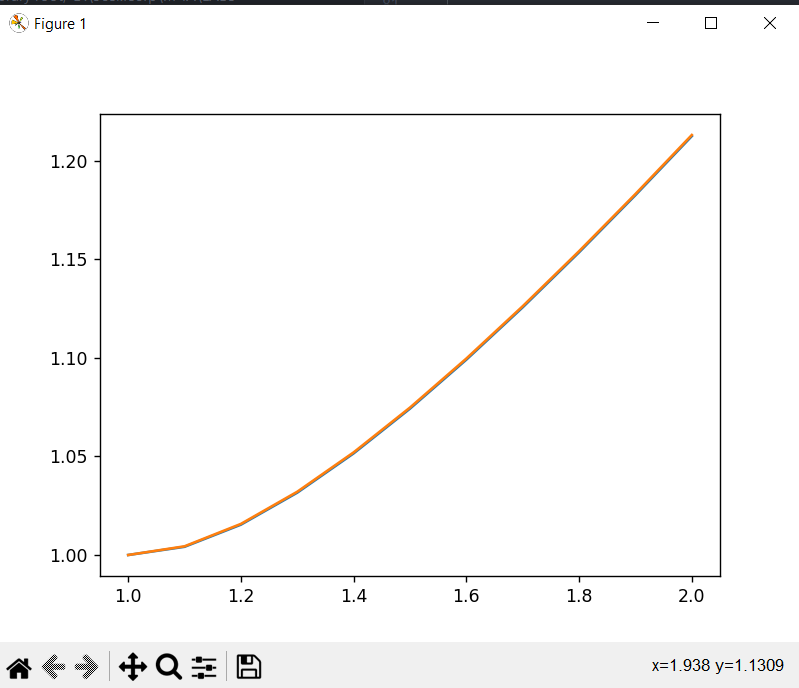
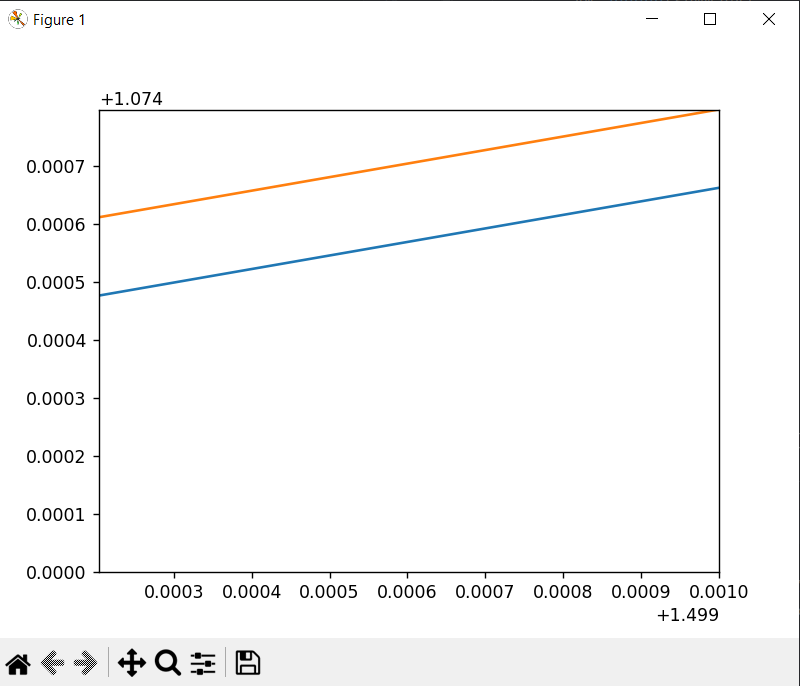
*.*

**Графики функций**

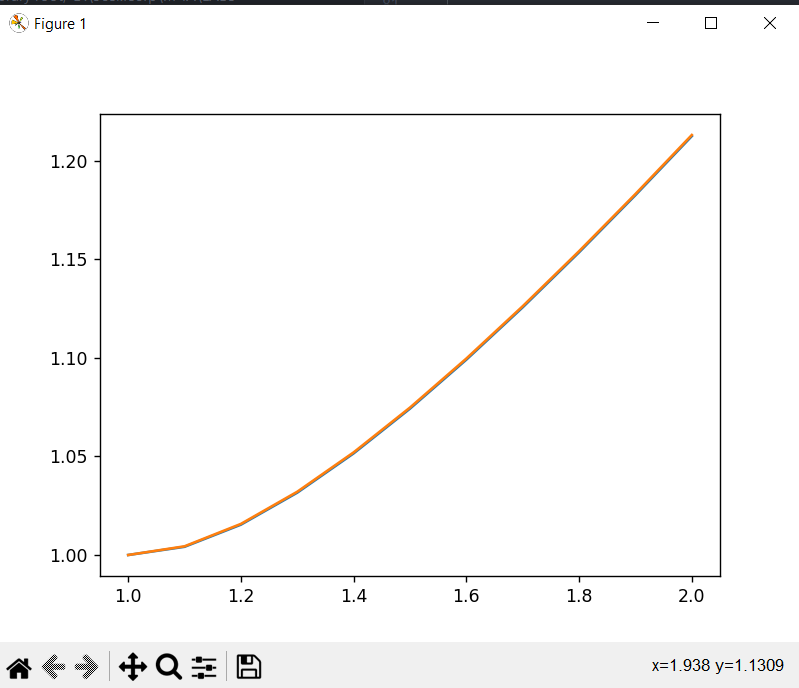
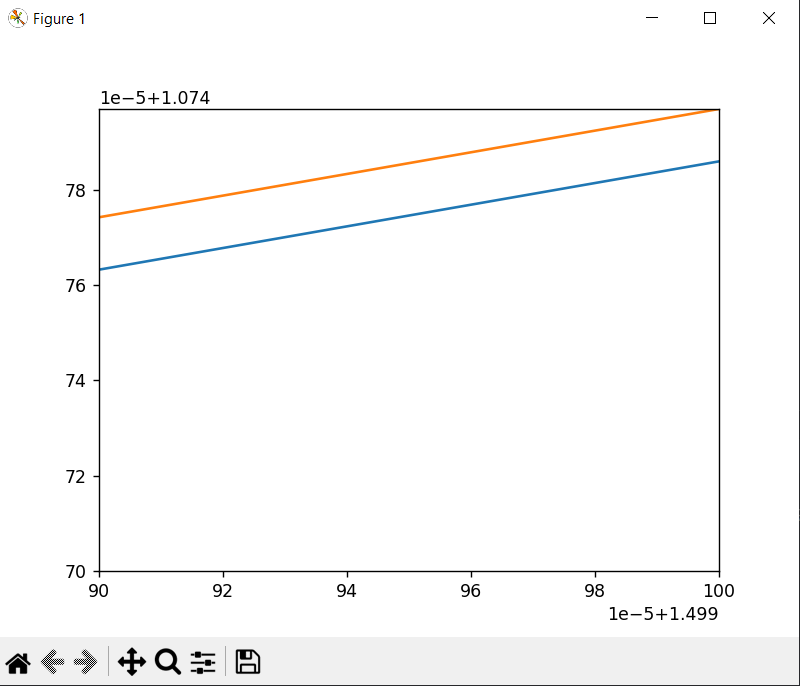
**1) Неявный метод трапеций (шаг 0.1):**

****

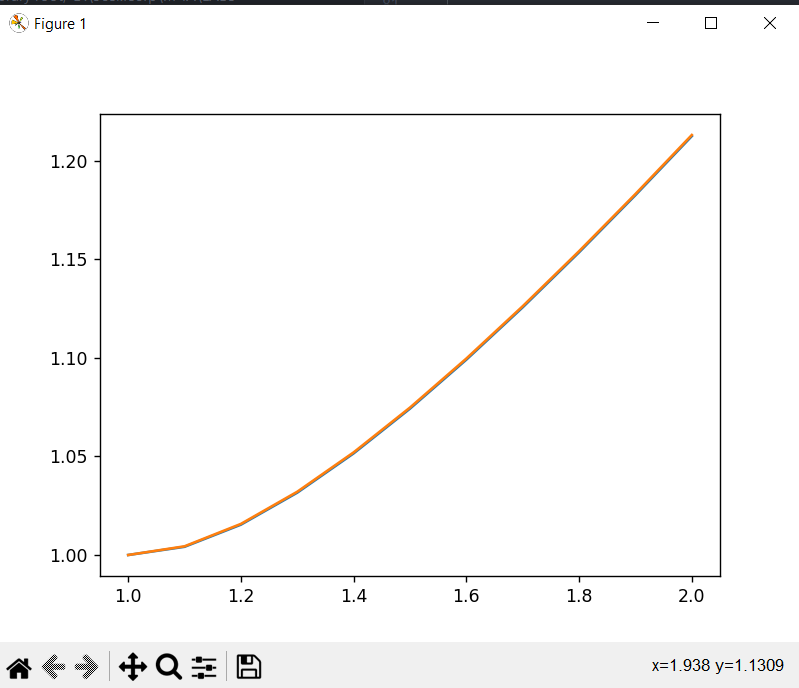
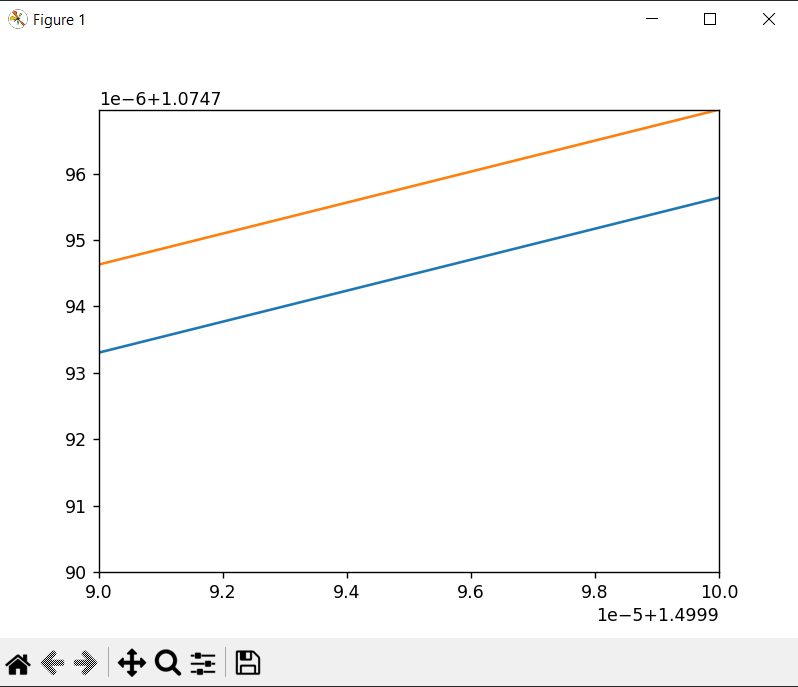
**2) Неявный метод трапеций (шаг 0.05):**

****

**3) Явный метод Рунге-Кутта (шаг 0.1):**

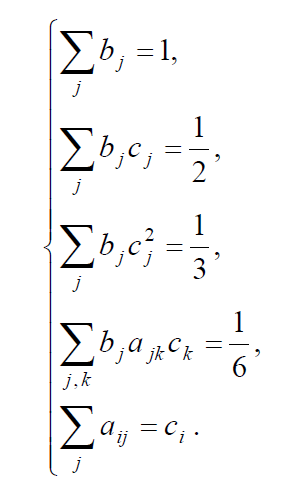
****

**4) Явный метод Рунге-Кутта (шаг 0.05):**

****

Явный метод Рунге-Кутта имеет порядок точности равный 3.

Так как он удовлетворяет следующим условиям:



1) 0.25 + 0.5 + 0.25 = 1

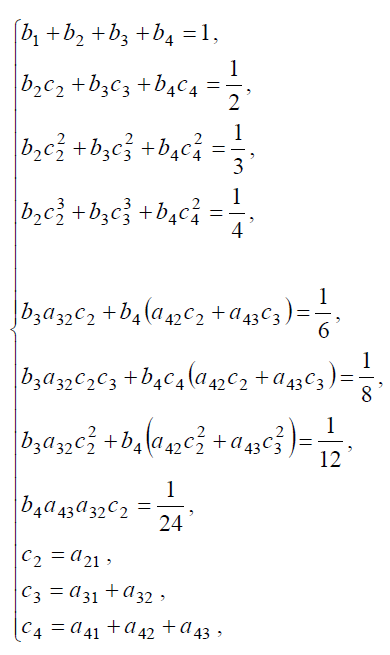
2) 0.25 \* 0 + 0.5 \* + 0.25 \* = 0.5

3) 0.25 \* 0 + 0.5 \* + 0.25 \* =

4) 0.25 \* \* 1 =

5)

Данный метод не может иметь порядок 4, так как не выполняется следующее условие:



**Листинг программы**

*import* numpy *as* np  
  
  
an = [[0, 0, 0], [2 / 3, 0, 0], [-1 / 3, 1, 0]]  
bn = [1 / 4, 2 / 4, 1 / 4]  
cn = [0, 2 / 3, 2 / 3]  
m\_trapezia = 2  
m\_runge\_kutta = 3  
  
  
*class* CauchyTask:  
 eps = 10 \*\* (-6)  
  
 *def* \_\_init\_\_(*self*, h, a=1, b=2, u0=1):  
 *self*.a = a  
 *self*.b = b  
 *self*.h = h  
 *self*.u0 = u0  
  
 @staticmethod  
 *def* f(t, u):  
 *return* (-u / t) \* np.log(u / t)  
  
 *def* t(*self*):  
 n = int((*self*.b - *self*.a) / *self*.h)  
 result = [*self*.a + i \* *self*.h *for* i *in* range(n + 1)]  
 *return* result  
  
 @staticmethod  
 *def* u(t):  
 *return* t \* np.exp((1 - t) / t)  
  
 *def* new\_f(*self*, yj1, yj, tj1, tj):  
 *return* yj1 - (*self*.h / 2) \* *self*.f(tj1, yj1) - yj - (*self*.h / 2) \* *self*.f(tj, yj)  
  
 *def* derivative\_new\_f(*self*, yj1, tj1):  
 *return* 1 + (*self*.h / 2) \* ((1 / tj1) \* np.log(yj1 / tj1) + (1 / tj1))  
  
 *def* newton\_method(*self*, yj, tj, tj1):  
 temp = yj  
 y\_res = temp - *self*.new\_f(temp, yj, tj1, tj) / *self*.derivative\_new\_f(temp, tj1)  
 *while* abs(y\_res - temp) > *self*.eps:  
 temp = y\_res  
 y\_res = temp - *self*.new\_f(temp, yj, tj1, tj) / *self*.derivative\_new\_f(temp, tj1)  
  
 *return* y\_res  
  
 *def* implicit\_trapezoid\_method(*self*):  
 ts = *self*.t()  
 n = int((*self*.b - *self*.a) / *self*.h)  
 y0 = [*self*.u0]  
  
 *for* i *in* range(n):  
 y0.append(*self*.newton\_method(y0[i], ts[i], ts[i + 1]))  
  
 *return* y0  
  
 *def* k(*self*, coef\_a, coef\_c, yj, tj):  
 k = [*self*.f(tj, yj)]  
 *for* i *in* range(1, len(coef\_c)):  
 summary = yj  
 *for* j *in* range(i):  
 summary += (*self*.h \* coef\_a[i][j] \* k[j])  
 k.append(*self*.f(tj + coef\_c[i] \* *self*.h, summary))  
 *return* k  
  
 *def* runge\_kutta(*self*, coef\_a, coef\_b, coef\_c):  
 ts = *self*.t()  
 n = int((*self*.b - *self*.a) / *self*.h)  
 y0 = [*self*.u0]  
  
 *for* i *in* range(n):  
 summary = 0  
 ks = *self*.k(coef\_a, coef\_c, y0[i], ts[i])  
 *for* j *in* range(len(coef\_b)):  
 summary += coef\_b[j] \* ks[j]  
 summary \*= *self*.h  
 summary += y0[i]  
 y0.append(summary)  
  
 *return* y0  
  
 @staticmethod  
 *def* count\_accuracy(u, y):  
 result = []  
 *for* i *in* range(len(u)):  
 result.append(np.abs(u[i] - y[i]))  
 *return* max(result)  
  
 @staticmethod  
 *def* count\_runge\_accuracy(y1, y2, m):  
 result = []  
 *for* i *in* range(len(y1)):  
 result.append(np.abs(y1[i] - y2[2 \* i]))  
 *return* max(result) / (2 \*\* m - 1)  
  
 @classmethod  
 *def* show\_result(*cls*, h, \*args):  
 solver = CauchyTask(h)  
 x = solver.t()  
 u\_result = [solver.u(i) *for* i *in* x]  
 print(f"Точное решение задачи Коши: {u\_result}")  
 *if* args == tuple():  
 y = solver.implicit\_trapezoid\_method()  
 print(f"Приближённое решение задачи Коши: {y}")  
 *else*:  
 y = solver.runge\_kutta(args[0], args[1], args[2])  
 print(f"Приближённое решение задачи Коши: {y}")  
 print(f"Погрешность равна: {*cls*.count\_accuracy(u\_result, y)}", end="\n" \* 2)  
  
  
sol1\_1 = CauchyTask(0.1)  
y1\_1 = sol1\_1.implicit\_trapezoid\_method()  
sol1\_2 = CauchyTask(0.05)  
y1\_2 = sol1\_2.implicit\_trapezoid\_method()  
CauchyTask.show\_result(0.1)  
CauchyTask.show\_result(0.05)  
  
print(f"Погрешность по правилу Рунге равна: {CauchyTask.count\_runge\_accuracy(y1\_1, y1\_2, m\_trapezia)}", end="\n" \* 2)  
  
sol2\_1 = CauchyTask(0.1)  
y2\_1 = sol2\_1.runge\_kutta(an, bn, cn)  
sol2\_2 = CauchyTask(0.05)  
y2\_2 = sol2\_2.runge\_kutta(an, bn, cn)  
CauchyTask.show\_result(0.1, an, bn, cn)  
CauchyTask.show\_result(0.05, an, bn, cn)  
  
print(f"Погрешность по правилу Рунге равна: {CauchyTask.count\_runge\_accuracy(y2\_1, y2\_2, m\_runge\_kutta)}", end="\n" \* 2)

**Результаты программы:**

**Неявный метод трапеций:**

Точное решение задачи Коши: [1.0, 1.0044107879104887, 1.0157780698687369, 1.0320994551633367, 1.0520682103054002, 1.074796965860684, 1.0996628460655555, 1.1262162301696748, 1.1541246991739182, 1.183137343210725, 1.2130613194252668]

Приближённое решение задачи Коши: [1, 1.0041607870029556, 1.0153890914707662, 1.0316311073440392, 1.0515541060706315, 1.0742567941879047, 1.0991084790771217, 1.1256549791831731, 1.1535611285303522, 1.182574318887095, 1.2125006347242677]

Погрешность равна: 0.000563570643566047

Точное решение задачи Коши: [1.0, 1.0011718025751506, 1.0044107879104887, 1.0093707932745097, 1.0157780698687369, 1.0234134413474774, 1.0320994551633367, 1.0416909780658048, 1.0520682103054002, 1.0631314262680263, 1.074796965860684, 1.0869941444714109, 1.0996628460655555, 1.1127516302458953, 1.1262162301696748, 1.1400183506793473, 1.1541246991739182, 1.1685061984812675, 1.183137343210725, 1.1979956700799117, 1.2130613194252668]

Приближённое решение задачи Коши: [1, 1.0011359288698238, 1.0043483760965077, 1.009288558905314, 1.0156809171270538, 1.0233049996150478, 1.0319824421926644, 1.0415674502394807, 1.0519397379157822, 1.0629992181116636, 1.074661959008234, 1.0868570697512354, 1.0995242763765056, 1.112612016575805, 1.1260759287077695, 1.1398776434068714, 1.153983809630786, 1.1683653039310842, 1.1829965840932994, 1.1978551574042295, 1.2129211405868048]

Погрешность равна: 0.00014089455018329033

Погрешность по правилу Рунге равна: 0.00014089370014458424

**Явный метод Рунге-Кутта:**

Точное решение задачи Коши: [1.0, 1.0044107879104887, 1.0157780698687369, 1.0320994551633367, 1.0520682103054002, 1.074796965860684, 1.0996628460655555, 1.1262162301696748, 1.1541246991739182, 1.183137343210725, 1.2130613194252668]

Приближённое решение задачи Коши: [1, 1.0044047804478966, 1.015769250790445, 1.0320893022055035, 1.0520574494242534, 1.0747859690261647, 1.0996518086546816, 1.1262052553911142, 1.1541138404468363, 1.183126626389763, 1.2130507548808747]

Погрешность равна: 1.1037410873937503e-05

Точное решение задачи Коши: [1.0, 1.0011718025751506, 1.0044107879104887, 1.0093707932745097, 1.0157780698687369, 1.0234134413474774, 1.0320994551633367, 1.0416909780658048, 1.0520682103054002, 1.0631314262680263, 1.074796965860684, 1.0869941444714109, 1.0996628460655555, 1.1127516302458953, 1.1262162301696748, 1.1400183506793473, 1.1541246991739182, 1.1685061984812675, 1.183137343210725, 1.1979956700799117, 1.2130613194252668]

Приближённое решение задачи Коши: [1, 1.001171374563978, 1.0044100681691857, 1.0093698726996383, 1.0157770102590975, 1.0234122854193188, 1.032098232863949, 1.041689710643511, 1.0520669129866294, 1.0631301100505615, 1.0747956387348865, 1.0869928122644565, 1.0996615130330756, 1.1127502994917757, 1.1262149039473104, 1.1400170306100592, 1.1541233864067952, 1.1685048938111358, 1.183136047165882, 1.1979943829878206, 1.21306004146205]

Погрешность равна: 1.3330324799554205e-06

Погрешность по правилу Рунге равна: 1.386339770568869e-06

**Вывод**

Из рассмотренных методов самым точным оказался явный метод Рунге-Кутта. Это связано с тем, что у него порядок точности выше, чем у неявного метода трапеций. У явного метода Рунге-Кутта порядок 3, а у трапеций – 2.