МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Костромской государственный университет»

(КГУ)

ИАСТ

Кафедра автоматизированных систем и технологий

09.03.02

Направление подготовки/Специальность Информационные системы и технологии

Дисциплина Численные методы

# Лабораторная №5.

# Решение ОДУ (Вариант 19).

Выполнил студент

Копосов Лев Владимирович

Группа 22-ИСбо-1б

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Кострома

**Постановка задачи.**

Решить задачу Коши модифицированным методом Эйлера и методом Хэмминга.

****

рис.1. Задача N19

**Краткая теория используемых методов.**

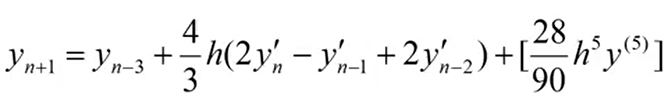
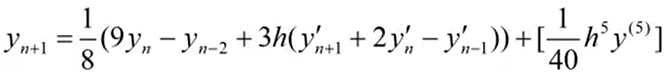
1. Модифицированный метод Эйлера.

Модифицированный метод Эйлера - численный метод для решения обыкновенных дифференциальных уравнений первого порядка задачи Коши. Этот метод представляет собой усовершенствованную версию метода Эйлера, обладающую более высокой точностью.

Модифицированный метод Эйлера заключается в улучшении точности аппроксимации решения дифференциального уравнения. На заданном интервале для точек с указанным шагом  (1.1), ищем тангенс  (1.2) Приближенное значение функции на следующем шаге рассчитывается, учитывая среднюю скорость изменения функции между текущим и следующим шагами  (1.3) и находится по формуле  (1.4). Повторяем шаги, пока не будет достигнута граница заданного интервала.

1. Метод Хэмминга.

Метод Хэмминга - численный метод четвертого порядка для решения обыкновенных дифференциальных уравнений.

Метод Хэмминга заключается в прогнозе  и последующей коррекции  4 предыдущих значений точки (которые можно получить, используя одношаговые методы). Прогноз в методе Хэмминга осуществляется по формуле  （2.1）, коррекция - по формуле  （2.2）. Вычисление нового приближенного решения ОДУ осуществляется пока не будет выполняться условие  （2.3), где ε - заданная точность. Повторяем шаги, пока не будет достигнута граница заданного интервала.

**Алгоритм решения дифференциальных уравнений задачей Коши**

1. **Модифицированный метод Эйлера.**

Шаг 1. Задаем начальную точку (x0, y0), шаг h и конечную точку вычисления x\_end;

Шаг 2. Вычисляем тангенс угла αn-1, tgαn-1, то есть ищем ее производную в точке (xn-1, yn-1);

Шаг 3. Вычисляем следующую абсциссу точки (1.1);

Шаг 4. Вычисляем тангенс угла αn, tgαn, то есть ищем ее производную в точке (xn, yn), где yn вычисляется по формуле (1.4);

Шаг 5. Вычисляем среднее арифметическое у тангенсов tgαn-1 и tgαn(1.3);

Шаг 6. Переходим к следующей точке (xn+1, yn+1), где xn+1 вычисляется по формуле (1.1), yn+1 - по формуле (1.4);

Шаг 7. Если xn+1 меньше x\_end, то Переходим к Шаг 2;

Иначе Выводим решение (xn+1, yn+1), Конец.

1. **Метод Хэмминга.**

Шаг 1. Задаем начальную точку (x0, y0), шаг h, конечную точку вычисления x\_end и точность вычисления ε;

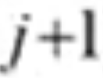
Шаг 2. Находим первые 4 точки, используя одношаговый метод Рунге-Кутта;

Шаг 3. Вычисляем абсциссу точки xn+1 = xn + h;

Шаг 4. Выполняем прогноз по формуле (2.1) и по найденному значению находим производную ;

Шаг 5. Выполняем коррекцию по формуле (2.2) и по найденному значению находим производную ;

Шаг 6. Если условие (2.3) выполняется, то Переходим к Шаг 7;

Иначе переходим к следующему значению  и Переходим к Шаг 5;

Шаг 7. Выполняем заключительную корректировку по формуле (2.2);

Шаг 8. Если x меньше x\_end, то Переходим к Шаг 3;

Иначе Выводим решение (xn+1, yn+1), Конец.

**Вывод результата решения задачи.**

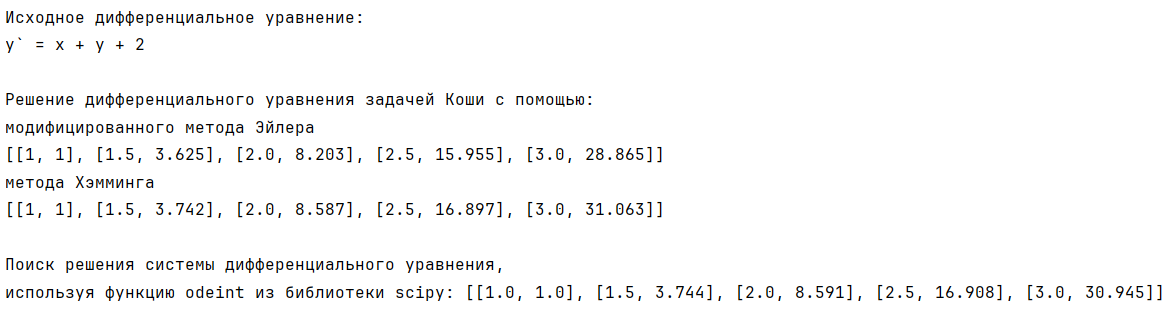
****

рис.4. Вывод результата задачи

**Проверка правильности решения.**

Правильность решений была проверена с помощью unit тестов.

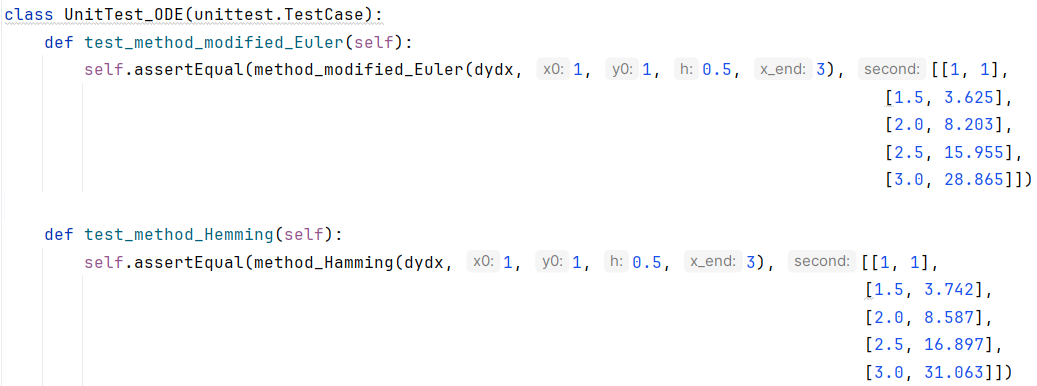


рис.5. Проверка корректности программы с помощью unit тестов

**Выводы.**

Решили задачу Коши модифицированным методом Эйлера и методом Хэмминга. Проверили решение с помощью unit тестов.

**Приложение: код программы.**

| **import numpy as np**  **from scipy.integrate import odeint**  **def dydx(x, y):**  ***""" Исходное дифференциальное уравнение """***  **return x + y + 2**  **def method\_modified\_Euler(func, x0, y0, h, x\_end):**  ***"""***  ***Решение дифференциального уравнения задачей Коши***  ***с помощью модифицированного метода Эйлера***  ***"""***  **x = x0**  **y = y0**  **result = [[x, y]]**  **while x < x\_end:**  **tga\_0 = h \* func(x, y)**  **x1 = x + h**  **tga\_1 = h \* func(x1, y + tga\_0)**  **tga\_av = (tga\_0 + tga\_1) / 2**  **y = y + tga\_av**  **x = x + h**  **segment = [x, round(y, 3)]**  **result.append(segment)**  **return result**  **def method\_Hamming(func, x0, y0, h, x\_end,**  **accuracy = 1e-3):**  ***"""***  ***Решение дифференциального уравнения задачей Коши***  ***с помощью метода Хемминга***  ***"""***  **def method\_Runge\_Kutta(f, x, y, h):**  **res = [[x, y]]**  **for i in range(3):**  **k1 = h \* f(x, y)**  **k2 = h \* f(x + h / 2, y + k1 / 2)**  **k3 = h \* f(x + h / 2, y + k2 / 2)**  **k4 = h \* f(x + h, y + k3)**  **y = y + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6**  **x = x + h**  **res.append([x, round(y, 3)])**  **return res**  **def dif5t(x, y):**  **for i in range(5):**  **y = dydx(x, y)**  **return y**  **def forecast(func, h, x, y):**  **y\_forecast = y[0] + 4/3 \* h \* (2 \* func(x, y[3]) - func(x, y[2]) + 2 \* func(x, y[1])) + (28/90 \* h\*\*5 \* dif5t(x, y[3]))**  **return y\_forecast**  **def correction(func, h, x, y, y\_forecast):**  **y\_correction = (9 \* y[3] - y[1] + 3 \* h \* (func(x, y\_forecast) + 2 \* func(x, y[3]) - func(x, y[2]))) / 8 + (h\*\*5 \* dif5t(x, y[3]) / 40)**  **return y\_correction**  **result = []**  **arr = method\_Runge\_Kutta(func, x0, y0, h)**  **result.extend(arr)**  **x = arr[-1][0]**  **while x < x\_end:**  **x = x + h**  **y\_arr = [segm[1] for segm in arr]**  **y\_forecast = forecast(func, h, x, y\_arr)**  **yp\_forecast = func(x, y\_forecast)**  **y\_correction = correction(func, h, x, y\_arr, y\_forecast)**  **yp\_correction = func(x, y\_correction)**  **while abs(yp\_forecast - yp\_correction) > accuracy:**  **y\_correction = correction(func, h, x, y\_arr, y\_forecast)**  **yp\_correction = func(x, y\_correction)**  **y\_forecast = y\_correction**  **yp\_forecast = func(x, y\_forecast)**  **y\_correction = correction(func, h, x, y\_arr, y\_forecast)**  **segment = [x, round(y\_correction, 3)]**  **arr = [\*arr[1:], segment]**  **result.append(segment)**  **return result**  **if \_\_name\_\_ == "\_\_main\_\_":**  **print(f"Исходное дифференциальное уравнение:\n"**  **f"y` = x + y + 2\n")**  **y0 = 1**  **x0 = 1**  **x\_segment = [1, 3]**  **h = 0.5**  **print("Решение дифференциального уравнения задачей Коши с помощью:")**  **print("модифицированного метода Эйлера")**  **res1 = method\_modified\_Euler(dydx, x0, y0, h, x\_segment[-1])**  **print(res1)**  **print("метода Хэмминга")**  **res2 = method\_Hamming(dydx, x0, y0, h, x\_segment[-1])**  **print(res2)**  **x\_seg = np.linspace(x\_segment[0], x\_segment[1], 5)**  **correct\_result = odeint(dydx, y0, x\_seg)**  **result = [[x, round(\*y, 3)] for x, y in zip(x\_seg, correct\_result)]**  **print(f"\nПоиск решения системы дифференциального уравнения, \n"**  **f"используя функцию odeint из библиотеки scipy: {result}")** |
| --- |