Написать программу численного решения задачи Коши для уравнения:

, ,

.

1. Реализовать какую-либо численную схему без использования готовых решателей.
2. Построить график решения.
3. Обосновать достоверность полученных результатов.

# Решение.

С самого начала стоит знать, что задача имеет точное аналитическое решение

Известно, что дифференциальное уравнение порядка может быть сведено к системе уравнений первого порядка путем переобозначений производных , , , ... . В нашем случае:

с соответствующим вектором начальных условий .

Каждое уравнение системы переписывается через разностною схему и на каждом новом шаге разбиения получаем сразу решение всей системы для всех функций . Нас, естественно, будет интересовать только функция , являющаяся решением исходной задачи.

В качестве решателя выбрана схема Рунге-Кутты 4-го порядка как самая популярная.

На одном шаге вектор-функция вычисляется 4 раза:

,

,

,

.

У метода 4-й порядок точности: суммарная ошибка имеет порядок .

Программировать будем на Python 3.8. Нам потребуется пакеты для работы с массивами numpy и рисования графиков matplotlib:

import numpy as np  
import matplotlib.pyplot as plt

Реализуем функцию runge\_kutta(x0, y0, xN, h), которая возвращает сетку решений . Входные параметры функции:

* x0 – начальная точка по ,
* y0 – вектор начальных значений ,
* xN – конечная точка по ,
* h – шаг сетки.

Пример вызова: начальные условия и интервал как в исходной задаче, шаг 0,01.

X, Y = runge\_kutta(x0=0, y0=[0, 3, -9, -8, 0], xN=5, h=0.01)

В результате X будет одномерный массив из , а Y будет двумерный массив (в строке с номером хранятся компоненты , которые соответствует i-й прогонке цикла по разбиению).

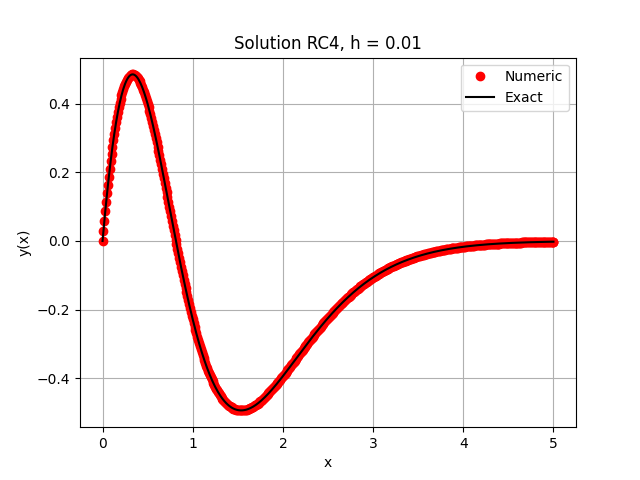
Программный код метода-функции runge\_kutta(x0, y0, xN, h) представлен ниже.

def runge\_kutta(x0, y0, xN, h):  
 *"""…Documentation…"""* X = []  
 Y = []  
 X.append(x0)  
 Y.append(y0)  
  
 xi, yi = x0, y0  
 # fill the grid by integration  
 while xi < xN:  
  
 h = min(h, xN-xi)  
  
 k1 = rhs(xi, yi)  
 k2 = rhs(xi + h/2, yi + k1/2)  
 k3 = rhs(xi + h/2, yi + k2/2)  
 k4 = rhs(xi + h, yi + k3)  
  
 yi = yi + h \* (k1 + 2\*k2 + 2\*k3 + k4) / 6  
 xi = xi + h  
  
 X.append(xi)  
 Y.append(yi)  
  
 return np.array(X), np.array(Y)

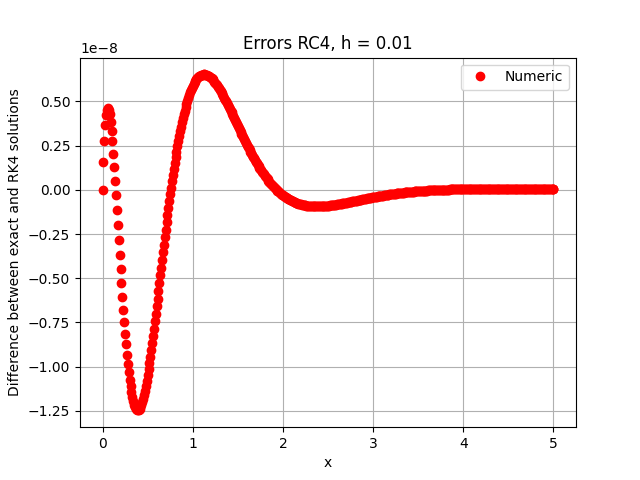
Внутри происходит вызов функции rhs(x, y) – аббревиатура от «right hand side», которая умеет считать правые части исходной системы.

def rhs(x, y):  
 *"""Right hand side function"""* f = np.array([y[1], y[2], y[3], y[4],  
 (-1)\*(15\*y[4] + 90\*y[3] + 270\*y[2] + 405\*y[1] + 243\*y[0])  
 ])  
 return f

Главный результат работы программы (т.е. решение исходной задачи Коши) представлен на графике. Шаг выбран . Все точки численного решения нанесены на график в виде красных кружков. Визуально они прекрасно накладываются на точное аналитическое решение (сплошная черная линия).



Найдем разницу между точным и численным решением, т.е. погрешность работы алгоритма. Из графика видно, что эта ошибка порядка , что соответствует ожиданиям для данного шага и методики (Рунге-Кутты 4-го порядка).



Так как на практике аналитическое решение может отсутствовать, было бы полезно иметь еще какой-либо ориентир для сравнения. Без сильной модификации метода можно сравнивать схему Рунге-Кутты 4-го порядка со схемой Рунге-Кутты 1-го порядка (схемой Эйлера).

,

.

Оценка погрешности:

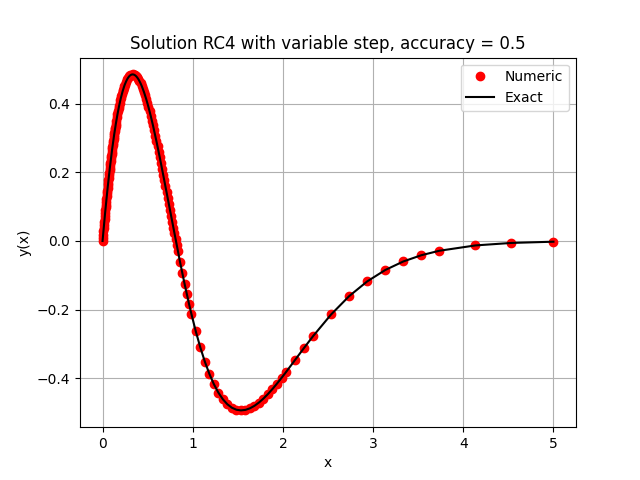
.

Если погрешность больше допуска , то инкремент нужно пересчитать с уменьшенным шагом . Если погрешность достаточно маленькая , то следующий шаг берем вдвое большим . Модифицированная функция носит название runge\_kutta\_tolerance(x0, y0, xN, h, t) и имеет дополнительный параметр t, отвечающий за точность.

Пример работы модифицированного метода

X, Y = runge\_kutta\_tolerance(x0, y0, xN, h=0.1, t=0.5)

Допуск t=0.5 специально был взят широким, чтобы увидеть переменный шаг.



Функции runge\_kutta (x0, y0, xN, h), runge\_kutta\_tolerance(x0, y0, xN, h, t), а также rhs(x, y) находятся в модуле models.py. В файле main.py происходит вызов этих функций и визуализация решения.

Репозиторий:

<https://github.com/alexande-popov/Runge_Kutta.git>