МГТУ им. Н. Э. Баумана
Факультет ФН «Фундаментальные Науки»
Кафедра ФН-12 «Математическое моделирование»

# Численные методы решения задач теории управления

## Домашнее задание №4

«Решение задачи Коши методами Рунге-Кутта»

Вариант №4

| | |
|---|---|
| Группа: | ФН12-61Б |
| Студент: | Дорохов М. А. |
| Преподаватель: | Тверская Е. С. |

Москва, 2024

# Постановка задачи

1. Составить программу интегрирования задачи Коши для системы из n уравнений первого порядка вида
$$y' = f(t, y),\ y(0) = y_0,\ y(t) \in \mathbb{R}^n$$
на произвольном отрезке $[a, b]$, используя метод Рунге-Кутта 3-го порядка точности с постоянным шагом $h$.

2. Для значений параметра $\beta_2 = 3;\ 3.48;\ 5$ при помощи разработанной процедуры рассчитать динамику популяции при различных начальных значениях размера опухоли $y_0 \in [0.5, 9]$. Привести гра- фики наиболее характерных решений в координатах $(x, y)$ и дать их интерпретацию. Параметры: $\lambda_1 = \lambda_2 = 1,\ \beta_1 = 1,\ c = 3,\ t \in [0, 20]$.

3. В разработанном программном коде, реализовать контроль точности на шаге с использованием правила Рунге.

# Ход работы

## Метод Рунге-Кутта 3-го порядка

Метод Рунге-Кутта - метод численного решения обыкновенных дифферницальных уравнений и их систем. В варианте задания приведена система ОДУ:

$$\begin{cases} x' = (-\lambda_1 + \beta_1 y^{\frac{2}{3}}(1 - \frac{x}{c})(1 + x))x \\ y' = \lambda_2 y - \beta_2 x y^{\frac{2}{3}}/(1 + x) \end{cases}$$

Метод Рунге-Кутта третьего порядка с постоянным шагом $h$ на сетке $\omega_h = \{t_0 + ih,\ i = \overline{0, n}\}$ для решения этой системы выглядит следующим образом:

$$x' = f(t, x, y), \quad y' = g(t, x, y),$$
$$k_1 = h \cdot f(t_n, x_n, y_n), \quad l_1 = h \cdot g(t_n, x_n, y_n),$$
$$k_2 = h \cdot f\left(t_n + \frac{h}{2}, x_n + \frac{hk_1}{2}, y_n + \frac{hl_1}{2}\right), \quad l_2 = h \cdot g\left(t_n + \frac{h}{2}, x_n + \frac{hk_1}{2}, y_n + \frac{hl_1}{2}\right),$$
$$k_3 = h \cdot f(t_n + h, x_n - hk_1 + 2hk_2, y_n - hl_1 + 2hl_2), \quad l_3 = h \cdot g(t_n + h, x_n - hk_1 + 2hk_2, y_n - hl_1 + 2hl_2),$$
$$x_{n+1} = x_n + \frac{1}{6}(k_1 + 4k_2 + k_3), \quad y_{n+1} = y_n + \frac{1}{6}(l_1 + 4l_2 + l_3).$$

## Правило Рунге на шаге для контроля точности

Правило Рунге - способ оценки точности численного решения ОДУ, который заключается в решении задачи с шагом $h$, а затем с шагом $h/2$. Формула для погрешности решения выглядит так:
$$R_n = \frac{|y_{n,h} - y_{n,h/2}|}{2^p - 1},$$
где $p$ - порядок точности метода Рунге-Кутта. В нашем случае ($p = 3$) формула принимает вид
$$R_n = \frac{|y_{n,h} - y_{n,h/2}|}{7}.$$
Приведённое правило применяют для изменения шага. Таким образом, вычисляют погрешность $R_n$ и сравнивают с некоторой величиной $\varepsilon$, отвечающей за желаемую точность. Если $R_n > \varepsilon$, то шаг $h$ уменьшают в два раза.

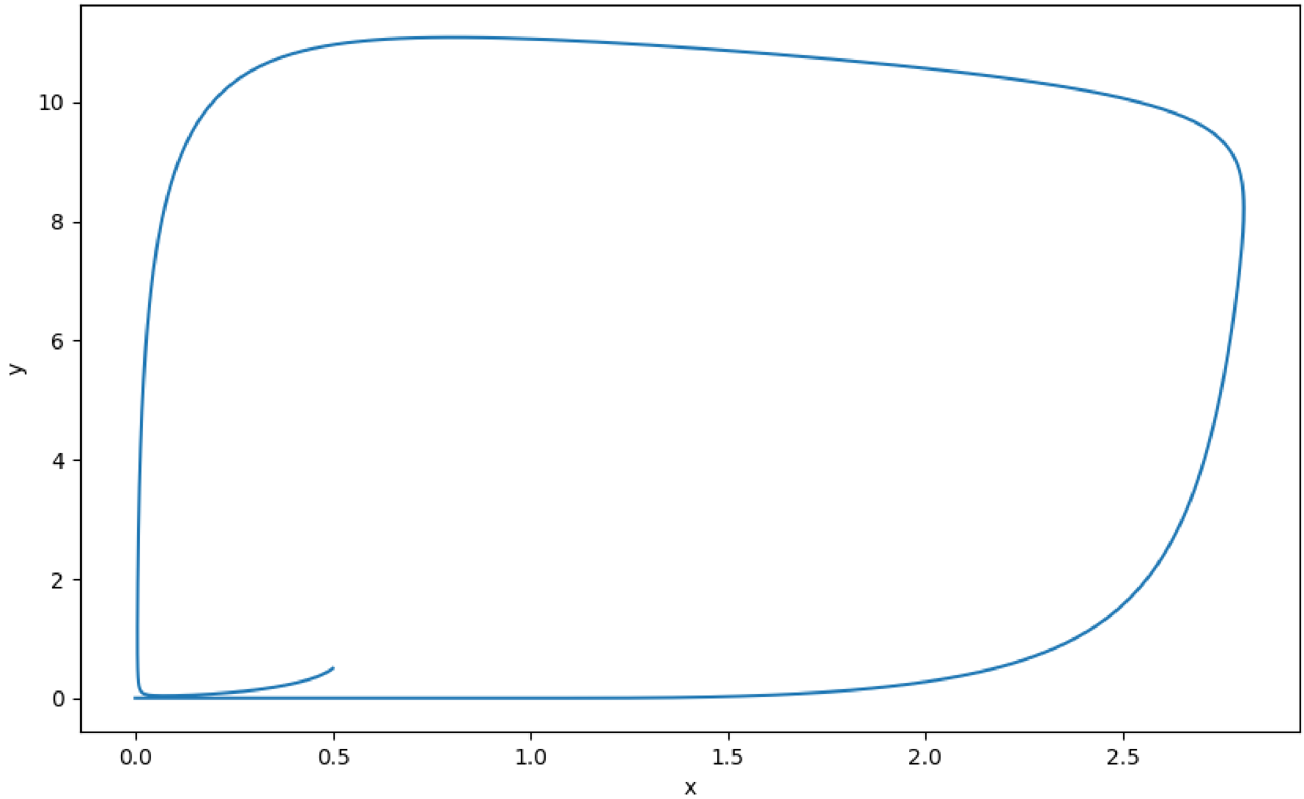
## Код программы

```
import numpy as np
import matplotlib.pyplot as plt
def plot_solution(path, num = '1'):
  plt.figure(figsize=(10, 6))
  x = [p[0] for p in path]
  y = [p[1] for p in path]
  plt.xlabel('x')
  plt.ylabel('y')
  plt.plot(x, y)
  #for i in range(0, len(path)):
  #  plt.plot(x[i], y[i], "ro")
  plt.savefig("график" + str(num))
def solveRK_3ord(tspan, f, g, x0, y0):
  h = 0.01
  t = tspan[0]
  x, y = x0, y0
  path = [(x0, y0)]
  while t <= tspan[-1]:
    k1 = h * f(t, x, y)
    l1 = h * g(t, x, y)
    k2 = h * f(t + h/2, x + h*k1/2, y + h*l1/2)
    l2 = h * g(t + h/2, x + h*k1/2, y + h*l1/2)
    k3 = h * f(t + h, x - h*k1 + 2 * h*k2, y - h*l1 + 2 * h*l2)
    l3 = h * g(t + h, x - h*k1 + 2 * h*k2, y - h*l1 + 2 * h*l2)

    t += h
    x += 1/6 * (k1 + 4*k2 + k3)
    y += 1/6 * (l1 + 4*l2 + l3)


    path.append([x, y])
  return [x, y], path
def solveRK_3ord_adaptive(tspan, f, g, x0, y0, eps = 1e-2):
  h = 0.01
  h1 = h/2
  t = tspan[0]
  t1 = t
  x, y = x0, y0
  x1, y1 = x, y
  path = [(x0, y0)]
  while t <= tspan[-1]:
    k1 = h * f(t, x, y)
    l1 = h * g(t, x, y)
    k2 = h * f(t + h/2, x + h*k1/2, y + h*l1/2)
    l2 = h * g(t + h/2, x + h*k1/2, y + h*l1/2)
    k3 = h * f(t + h, x - h*k1 + 2 * h*k2, y - h*l1 + 2 * h*l2)
    l3 = h * g(t + h, x - h*k1 + 2 * h*k2, y - h*l1 + 2 * h*l2)

    t += h
    x += 1/6 * (k1 + 4*k2 + k3)
    y += 1/6 * (l1 + 4*l2 + l3)
```

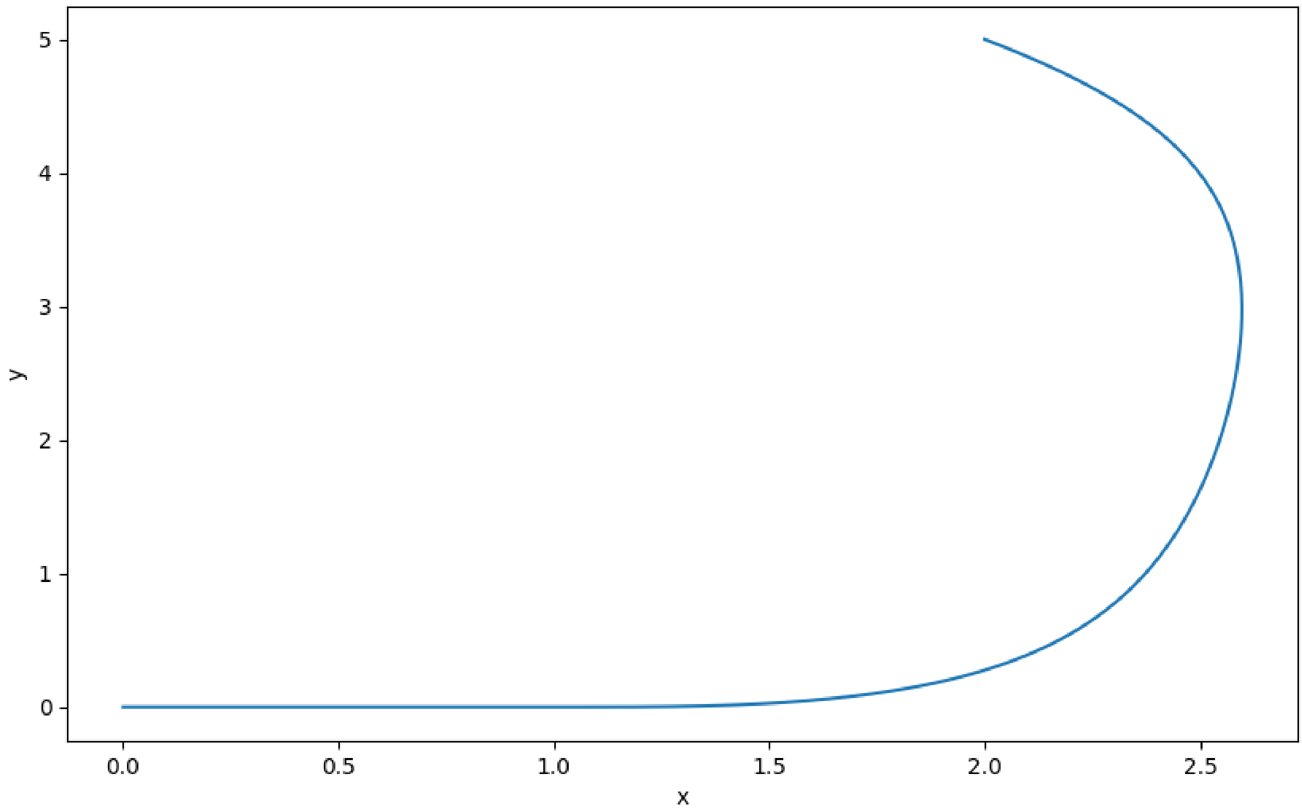
```
    for d in [1, 2]:
      k1 = h1 * f(t, x, y)
      l1 = h1 * g(t, x, y)
      k2 = h1 * f(t + h1/2, x + h1*k1/2, y + h1*l1/2)
      l2 = h1 * g(t + h1/2, x + h1*k1/2, y + h1*l1/2)
      k3 = h1 * f(t + h1, x - h1*k1 + 2 * h1*k2, y - h1*l1 + 2 * h1*l2)
      l3 = h1 * g(t + h1, x - h1*k1 + 2 * h1*k2, y - h1*l1 + 2 * h1*l2)
      t1 += h1
      x1 += 1/6 * (k1 + 4*k2 + k3)
      y1 += 1/6 * (l1 + 4*l2 + l3)

    R = max(abs(x - x1)/7, abs(y - y1)/7)
    if (R > eps):
      h /= 2
      print(f"Шаг изменён: h = {h}")
      if(h == 0):
        break
    path.append([x, y])

  return [x, y], path
f = lambda t, x, y: (-lambd1 + beta1 * (y ** (2/3))*(1- x/c)*(1+x)) * x
g = lambda t, x, y: lambd2 * y - beta2 * x * (y ** (2/3))/(1+x)
lambd1 = 1
lambd2 = 1
beta1 = 1
beta2 = 5
c = 3
tspan = [0, 20]
x0, y0 = 0.5, 0.5
solution, path = solveRK_3ord(tspan, f, g, x0, y0)
print(path)
plot_solution(path)
```

### Результаты

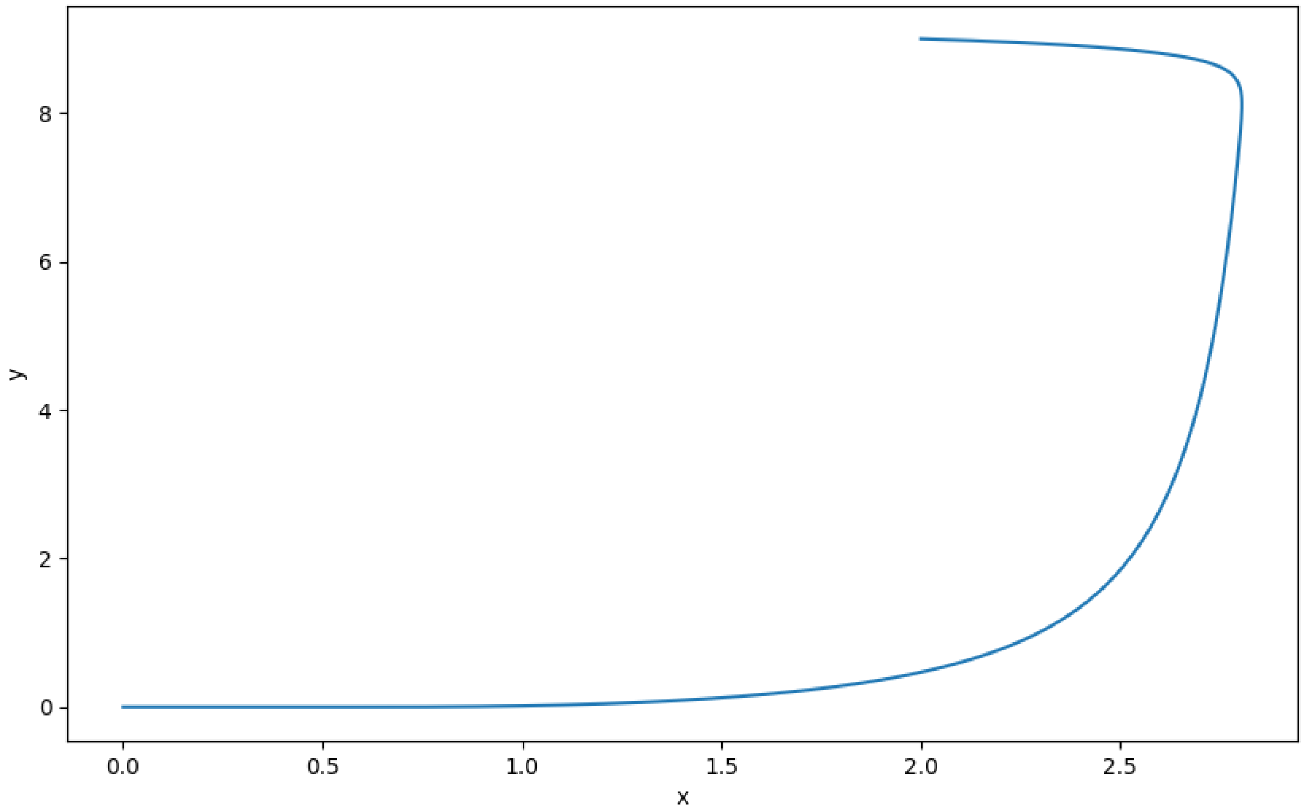
1. $\beta_2 = 5, x_0 = 0.5, y_0 = 0.5$



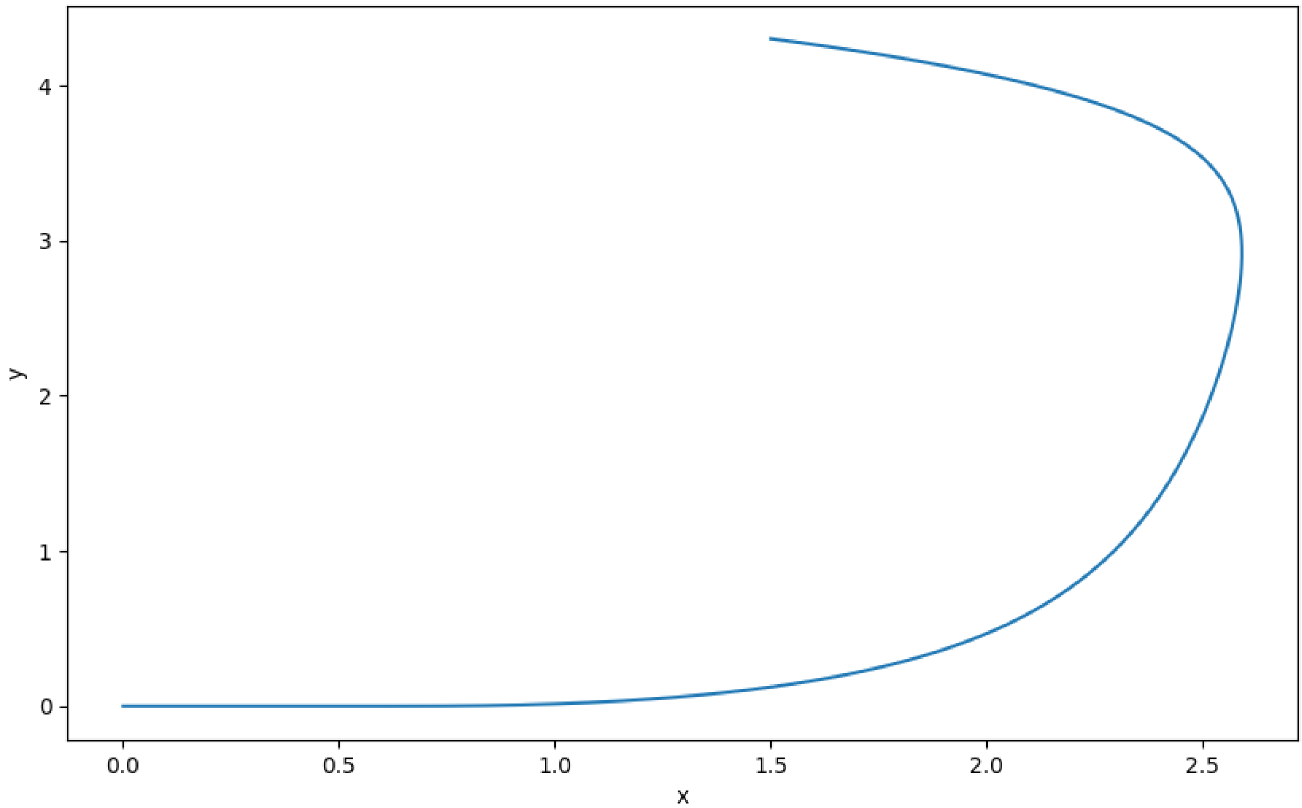

2. $\beta_2 = 5, x_0 = 2, y_0 = 5$



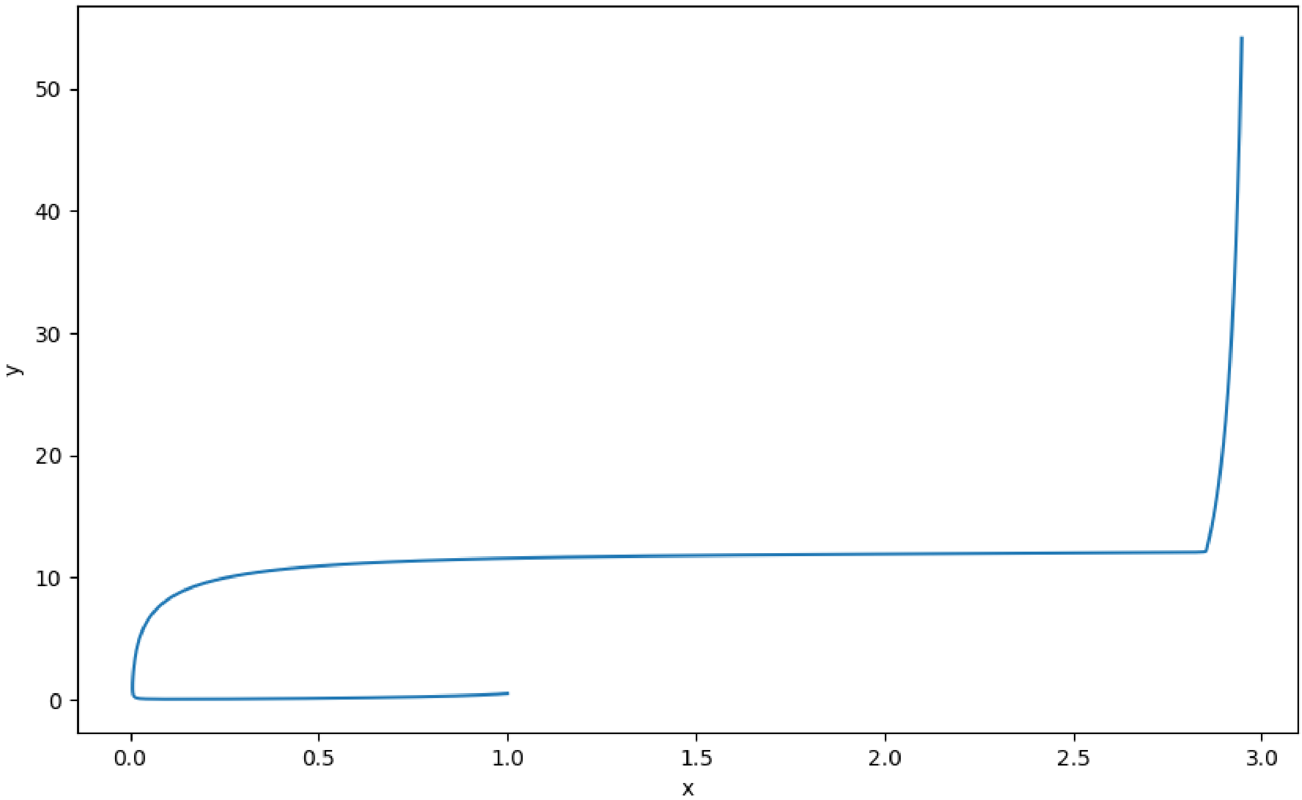
3. $\beta_2 = 3.48, x_0 = 2, y_0 = 9$




4. $\beta_2 = 3.48, x_0 = 1.5, y_0 = 4.3$



5. $\beta_2 = 3, x_0 = 1, y_0 = 0.5$




6. $\beta_2 = 3, x_0 = 0.5, y_0 = 0.5$