Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу

«Введение в численные методы»
Задание 2

Числнные методы решения дифференцальных уравнений

ОТЧЕТ

о выполненном задании

гор. Москва
2021 г.

Содержание

Постановка задачи

Задача 1

Рассматривается ОДУ первого порядка, разрешённое относительно производной, с дополнительным начальным условием в точке a и имеющее вид:

$$\begin{cases} \frac{dy}{dx} = f(x,y) \quad a \leq x \leq b \\ y(a) = y_0 \end{cases} (1)$$

Необходимо найти решение данной задачи Коши в предположении, что правая часть уравнения $f = f(x, y)$ гарантирует существование и единственность решения задачи Коши.

Рассматривается система линейных ОДУ первого порядка, разрешённых относительно производной, с дополнительными условиями в точке a:

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2) \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2) \qquad a \leq x \leq b \ (2) \\ y_1(a) = y_1^0, y_2(a) = y_2^0 \end{cases}$$

Необходимо найти решение данной задачи Коши в предположении, что правые части уравнений гарантируют существование и единственность решения задачи Коши для системы.

Задача 2

Рассматривается краевая задача для дифференциального уравнения второго порядка с дополнительными условиями в граничных точках:

$$\begin{cases} y" + p(x)y' + q(x)y = f(x) \\ \sigma_1 y(a) + \gamma_1 y'(a) = \delta_1 \qquad a \leq x \leq b \ (3) \\ \sigma_2 y(b) + \gamma_2 y'(b) = \delta_2 \end{cases}$$

Необходимо найти решение данной краевой задачи.

Цели

- Часть 1

  Изучить методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задач Коши для дифференциального уравнения (или системы) первого порядка:

  - Решить задачу Коши (1) или (2) 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;
  - Найти численное решение задачи и построить его график;
  - Найденное численное решение сравнить с точным решением дифференциального уравнения

- Часть 2

  Изучить метод прогонки решения краевой задачи для дифференциального уравнения второго порядка:

  - Решить краевую задачу (3) методом конечных разностей, аппроксимировав ее разностной схемой второго порядка точности (на равномерной сетке); полученную систему конечно-разностных уравнений решить методом прогонки;
  - Найти разностное решение задачи и построить его график;
  - Найденное разностное решение сравнить с точным решением дифференциального уравнения

Описание алгоритмов

Метод Рунге-Кутты

Будем использовать следующие формулы для численного решения задачи Коши, приближающие точное решение с четвёртым порядком точности относительно диаметра разбиения отрезка, на котором решается поставленная задача.
Положим:

- n - число точек разбиения отрезка
- $h = \frac{a-b}{n}$ - диаметр разбиения отрезка
- $x_i = a + h * i, y_i = y(x_i), 0 \leq i \leq n$ - сетка и сеточная функция

Метод Рунге-Кутты 2 порядка точности для рекуррентного вычисления сеточной функции примет следующий вид:

$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_i + h, y_i + h * f(x_i, y_i)))$$

Метод Рунге-Кутты 4 порядка точности для рекуррентного вычисления сеточной функции примет следующий вид:

$$\begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \\ k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \\ k_4 = f(x_i + h, y_i + hk_3) \\ y_{i+1} = y_i + \frac{h}{6}(k_1 + 2 \cdot (k_2 + k_3) + k_4) \end{cases}$$

Для задачи (2) метод Рунге-Кутты 4 порядка для рекуррентного вычисления сеточной функции примет следующий вид:

$$\begin{cases} k_{11} = f_1(x_i, y_1^i, y_2^i) \\ k_{21} = f_2(x_i, y_1^i, y_2^i) \\ k_{12} = f_1(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_{11}, y_2^i + \frac{h}{2}k_{21}) \\ k_{22} = f_2(x_i + \frac{h}{2}, y_2^i + \frac{h}{2}k_{11}, y_2^i + \frac{h}{2}k_{21}) \\ k_{13} = f_1(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_{12}, y_2^i + \frac{h}{2}k_{22}) \\ k_{23} = f_2(x_i + \frac{h}{2}, y_2^i + \frac{h}{2}k_{12}, y_2^i + \frac{h}{2}k_{22}) \\ k_{14} = f_1(x_i + h, y_1^i + hk_{13}, y_2^i + hk_{23}) \\ k_{24} = f_2(x_i + h, y_1^i + hk_{13}, y_2^i + hk_{23}) \\ y_1^{i+1} = y_1^i + \frac{h}{6}(k_{11} + 2 \cdot (k_{12} + k_{13}) + k_{14}) \\ y_2^{i+1} = y_2^i + \frac{h}{6}(k_{21} + 2 \cdot (k_{22} + k_{23}) + k_{24}) \end{cases}$$

Краевая задача

Для решения данной задачи запишем заданное дифференциальное уравнение в узлах сетки и краевые условия:

$$\begin{cases} y_i'' + p_i y_i' + q_i y_i = f_x, x_i = a + i\frac{b-a}{n} \quad 0 \leq i \leq n \\ \sigma_1 y_0 + \gamma_1 y_0' = \delta_1 \\ \sigma_2 y_n + \gamma_2 y_n' = \delta_2 \end{cases}$$

Для $1 \leq i \leq n-1$ существует следующее разностное приближение для первой и второй производной и самой сеточной функции:

$$\begin{cases} y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \\ y_i' = \frac{y_{i+1} - y_{i-1}}{2h} \end{cases}$$

В результате подстановки этих разностных отношений в начальное уравнение в виде сеточной функции получим линейную систему из n+1 уравнений с n+1 неизвестными $y_0, y_1, ..., y_n$:

$$\begin{cases} y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p_i \frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i = f_i \quad 1 \leq i \leq n-1 \\ \sigma_1 y_0 + \gamma_1 \frac{y_0 - y_0}{h} = \delta_1 \\ \sigma_2 y_1 + \gamma_2 \frac{y_n - y_{n-1}}{h} = \delta_2 \end{cases}$$

Решим полученную систему, используется метод прогонки.
После преобразований получим:

$$\begin{cases} C_0 y_0 + B_0 y_1 = F_0 \\ A_i y_{i-1} + C_i y_i + B_i y_{i+1} = F_i \quad 1 \leq i \leq n-1 \\ A_n y_{n-1} + C_n y_n = F_n \end{cases}$$

$$\begin{cases} \alpha_0 = -\frac{B_0}{C_0} \\ \beta_0 = \frac{F_0}{C_0} \\ \left.\begin{array}{l} \alpha_i = -\frac{B_i}{C_i + A_i \alpha_{i-1}} \\ \\ \beta_i = \frac{F_i - A_i \beta_{i-1}}{C_i + A_i \alpha_{i-1}} \end{array}\right. \quad 1 \leq i \leq n-1 \\ \\ y_n = \frac{F_n - A_n \beta_{n-1}}{C_n + A_n \alpha_{n-1}} \\ y_i = \beta_i + \alpha_i * y_{i+1} \quad 0 \leq i \leq n-1 \end{cases}$$

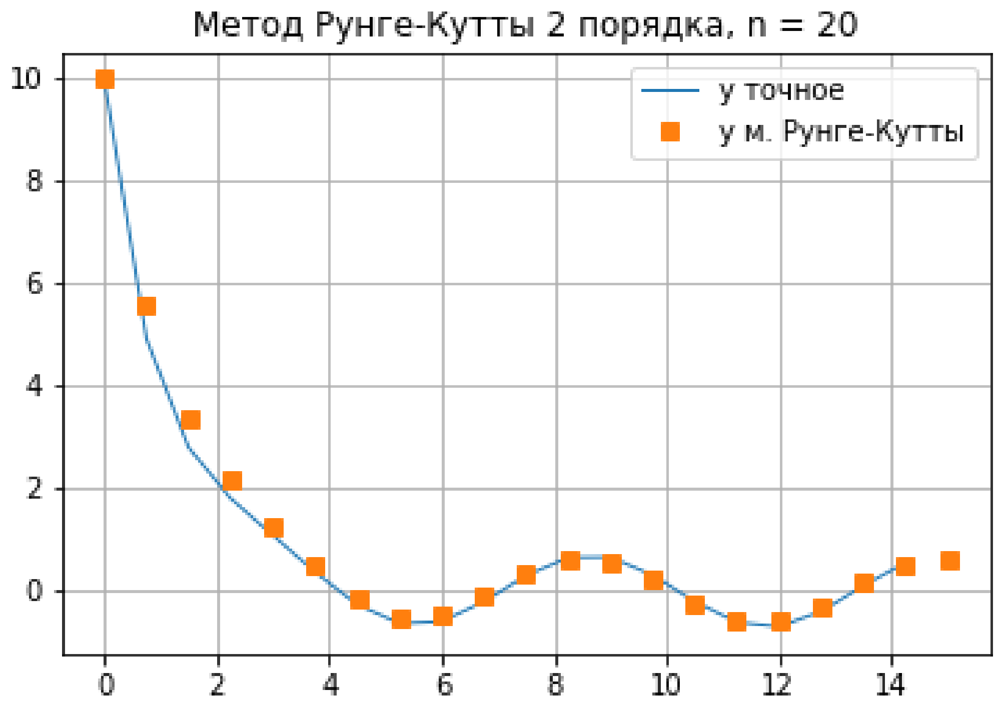
Откуда и находим все y_i

Тестировние функций

Тест1

$$\begin{cases} y' = sin(x) - y \\ x_0 = 0 \\ y_0 = 10 \end{cases}$$

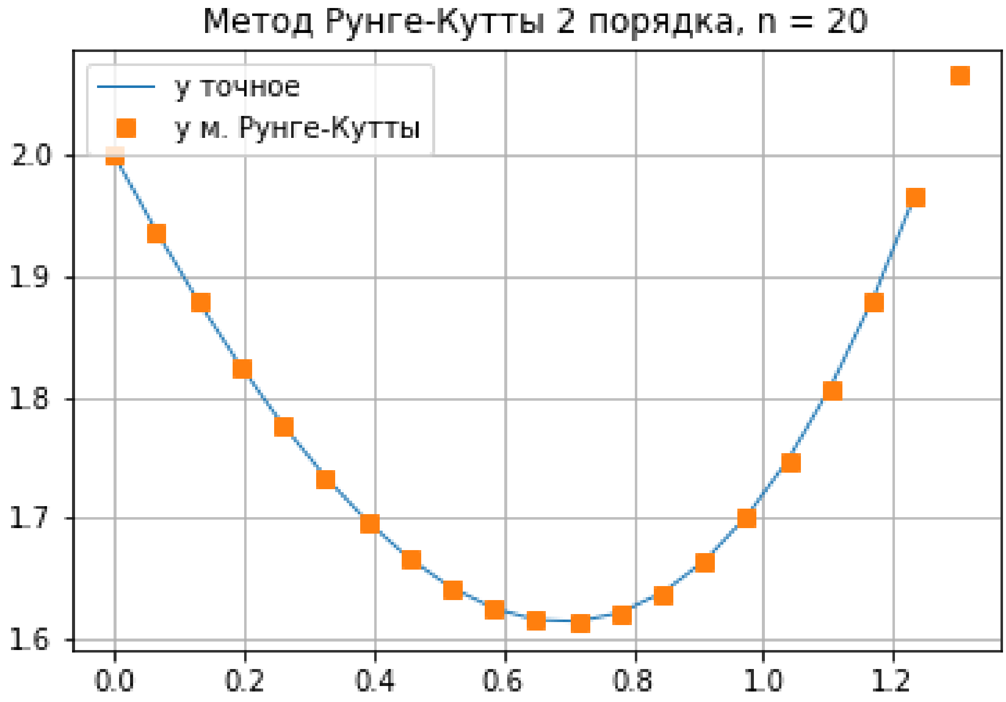
Точное решение: $y = -0.5cos(x) + 0.5sin(x) + 21/2 \times e^{-x}$



Тест2

$$\begin{cases} y' = y + 2x - 3 \\ x_0 = 0 \\ y_0 = 2 \end{cases}$$

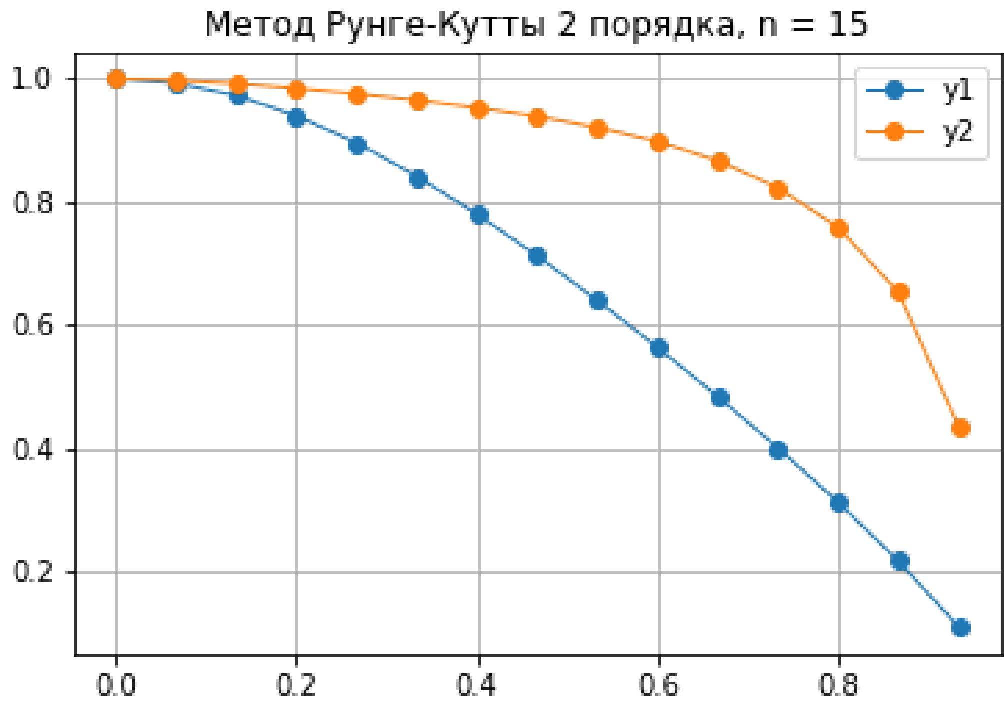
Точное решение: $y = 1 + e^x - 2x$

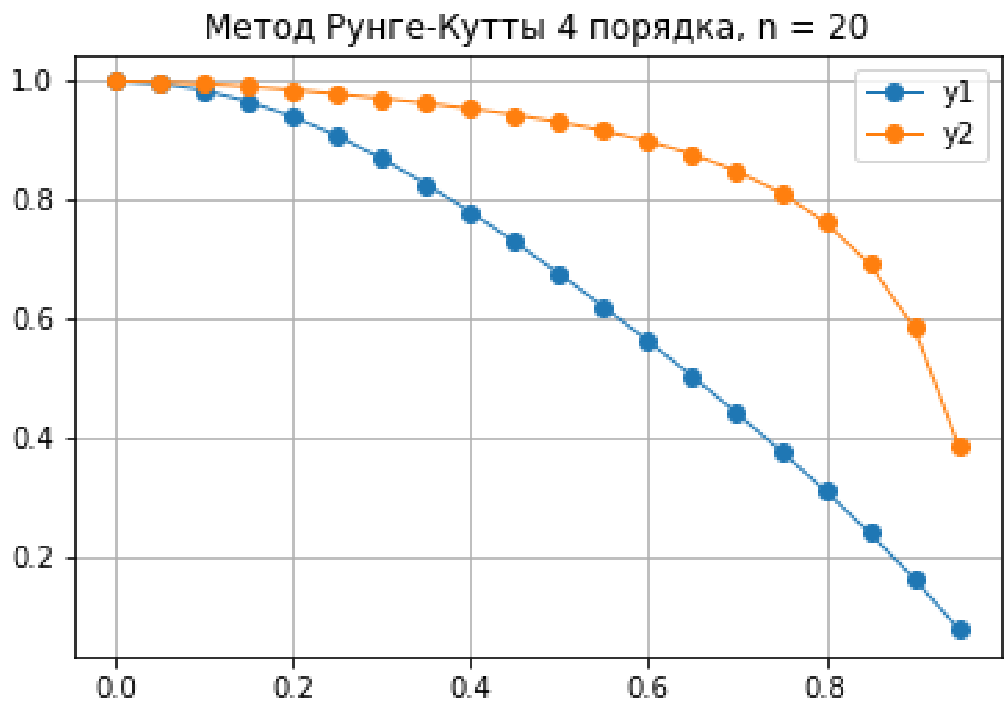


Тест3

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

Аналитического решения не существует



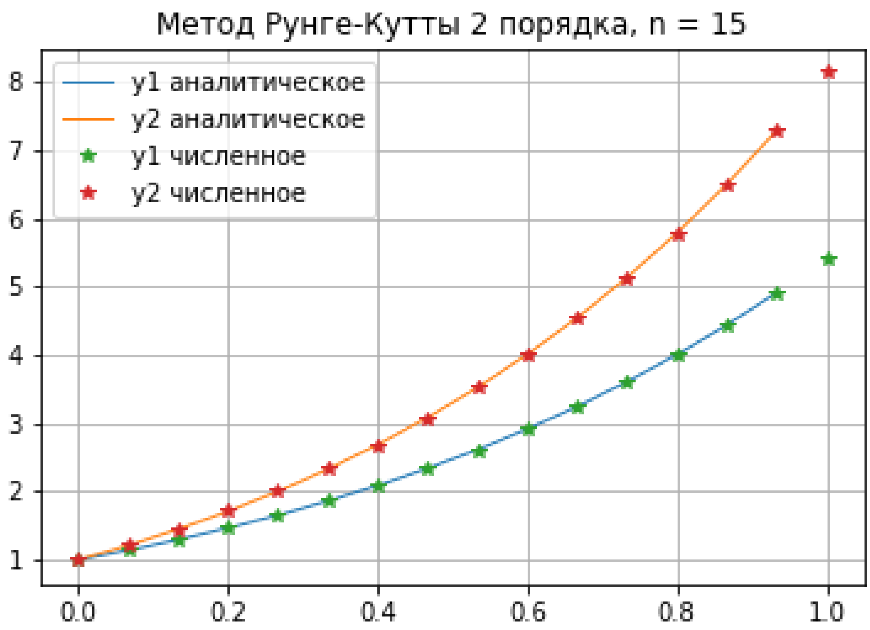


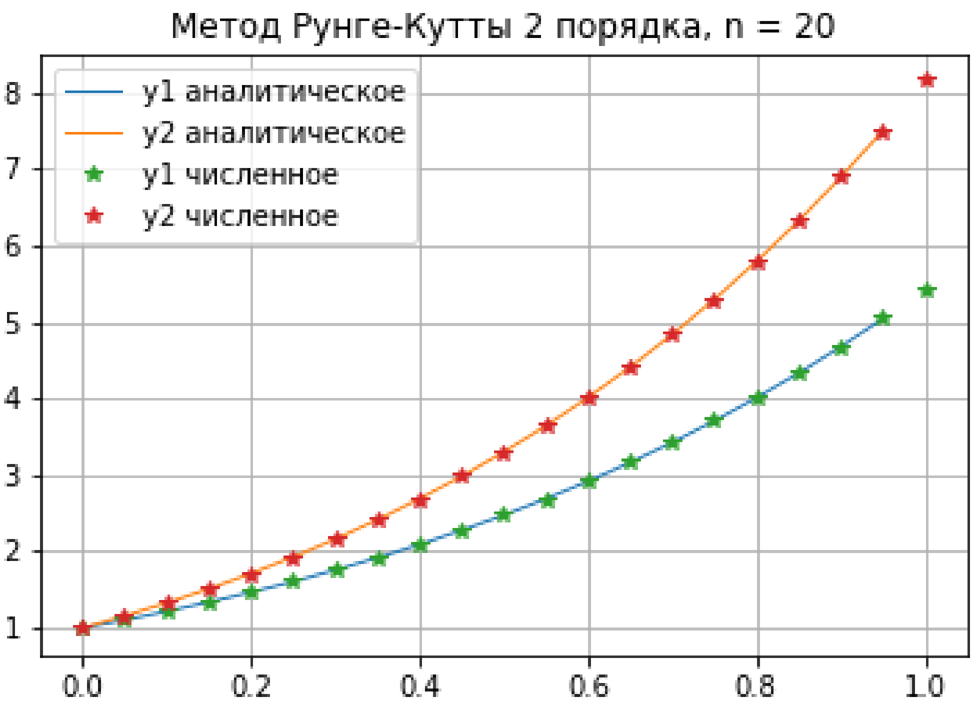
Тест4

$$\begin{cases} u' = 3u - v \\ v' = 4u - v \\ u(0) = 1 \\ v(0) = 1 \end{cases}$$

Аналитического решение:

$$\begin{cases} u = (1+x)e^x \\ v = (1+2x)e^x \end{cases}$$

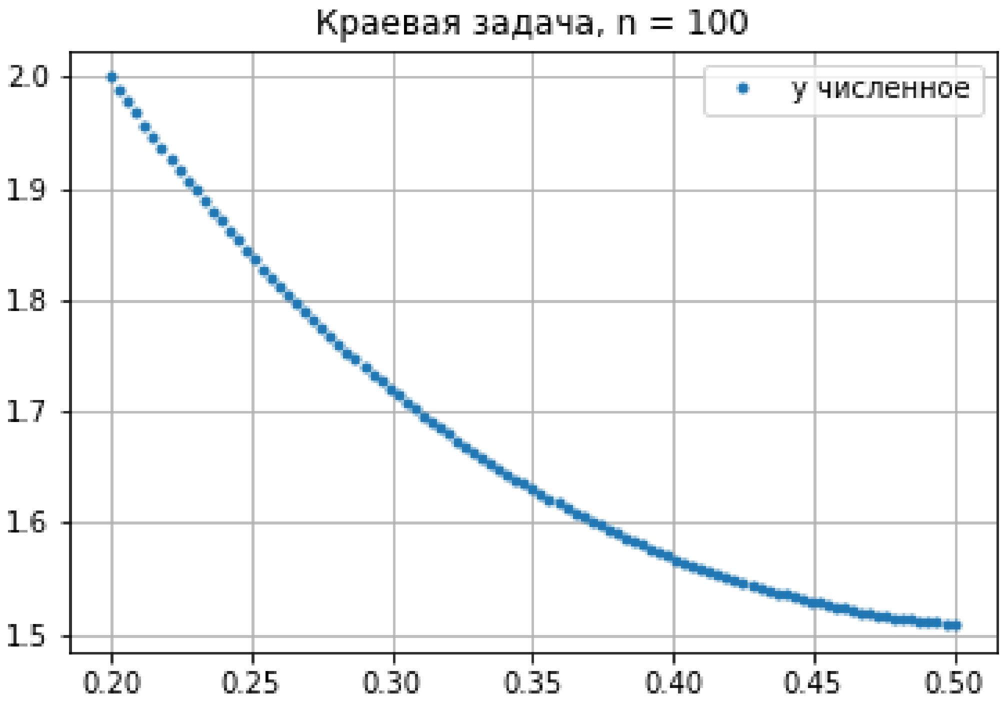




Тест5

$$\begin{cases} y'' + 2xy' - y/x = 3 \\ y'(0.2) = 2 \\ 0.5 \cdot y(0.5) - y'(0.5) = 1 \end{cases}$$

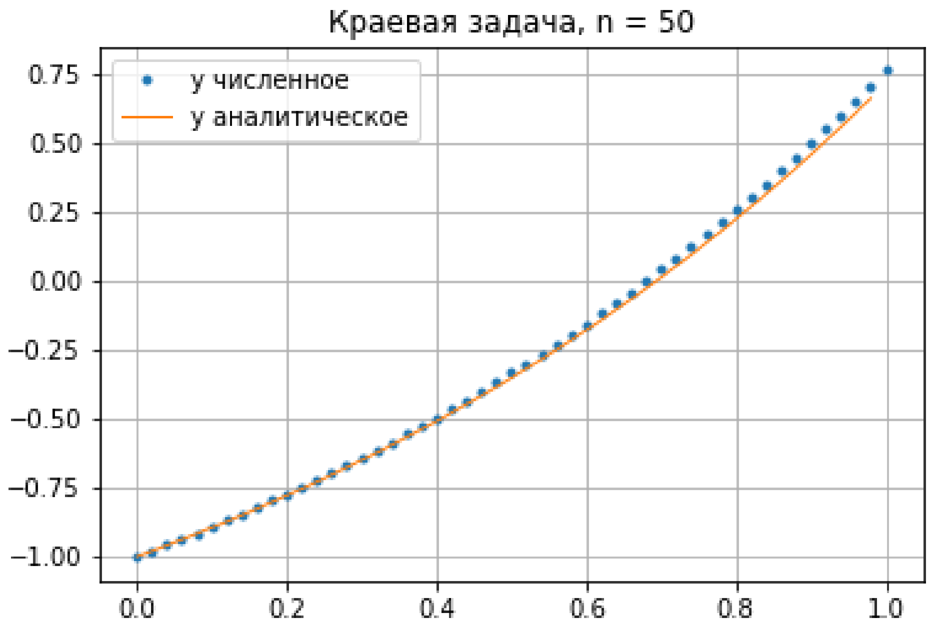
Аналитического решения не существует



Тест6

$$\begin{cases} y'' - y' = 0 \\ y(0) = -1 \\ -y(1) + y'(1) = 2 \end{cases}$$

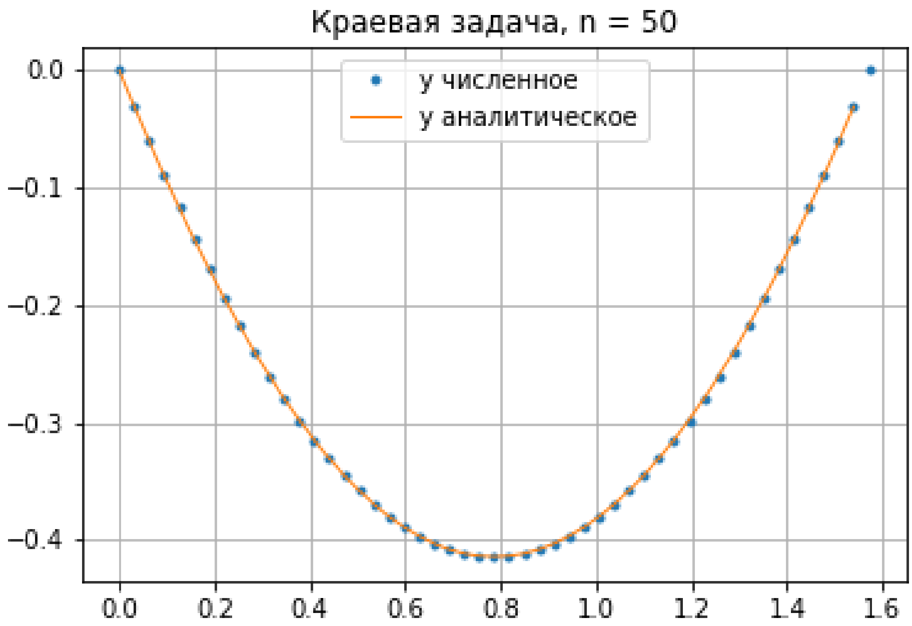
Аналитического решение: $y = -2 + e^x$



Тест7

$$\begin{cases} y'' + y = 1 \\ y(0) = 0 \\ y(\frac{\pi}{2}) = 0 \end{cases}$$

Аналитическое решение: $y = 1 - \sin x - \cos x$



Исходный код

```
import numpy as np
import matplotlib.pyplot as plt

def method_runge2(f, a, b, y0, n): # метод РунгеКутты- 2 порядка
  h = (b - a) / n
  x = a
  y = y0
  X = [x]
  Y = [y]
  for i in np.arange(n):
    x = a + h*i
    y = y + h/2*(f(x,y)+f(x+h, y+f(x,y)*h))
    X.append(x+h)
    Y.append(y)
  return X, Y

def method_runge4(f, a, b, y0, n): # метод РунгеКутты- 4 порядка
  h = (b - a)/n
  x = a
  y = y0
  Y = [y]
  X = [x]
  for i in np.arange(n):
    x = a + h * i
    k1 = f(x, y)
    k2 = f(x + h / 2, y + h / 2 * k1)
    k3 = f(x + h / 2, y + h / 2 *k2)
    k4 = f(x+h, y + h * k3)
    y = y + h / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
    X.append(x+h)
    Y.append(y)
  return X, Y

def method_runge2s(f1, f2, a, b, y01, y02, n): # метод РунгеКутты- 2 порядка для системы
  h = (b - a)/n
  x = a
  y1 = y01
  y2 = y02
  Y1 = [y1]
  Y2 = [y2]
  X = [x]
  for i in np.arange(n-1):
    x = a + h*i
    y1_n = y1 + h / 2 * (f1(x, y1, y2) + f1(x+h, y1+f1(x,y1,y2)*h, y2+f2(x,y1,y2)*h))
    y2_n = y2 + h / 2 * (f2(x, y1, y2) + f2(x + h, y1 + f1(x,y1,y2) * h, y2 + f2(x,y1,y2)*h))
    y1 = y1_n
    y2 = y2_n
    X.append(x+h)
    Y1.append(y1)
    Y2.append(y2)
  return X, Y1, Y2

def method_runge4s(f1, f2, a, b, y01, y02, n): # метод РунгеКутты- 4 порядка для системы
  h = (b - a) / n
  x = a
  y1 = y01
  y2 = y02
  Y1 = [y1]
  Y2 = [y2]
  X = [x]
  for i in np.arange(n-1):
```

```
    x = a + h * i
    k11 = f1(x, y1, y2)
    k12 = f2(x, y1, y2)
    k21 = f1(x + h / 2, y1 + h / 2 * k11, y2 + h / 2 * k12)
    k22 = f2(x + h / 2, y1 + h / 2 * k11, y2 + h / 2 * k12)
    k31 = f1(x + h / 2, y1 + h / 2 * k21, y2 + h / 2 * k22)
    k32 = f2(x + h / 2, y1 + h / 2 * k21, y2 + h / 2 * k22)
    k41 = f1(x + h, y1 + h * k31, y2 + h * k32)
    k42 = f2(x + h, y1 + h * k31, y2 + h * k32)
    y1 = y1 + h / 6 * (k11 + 2 * k21 + 2 * k31 + k41)
    y2 = y2 + h / 6 * (k12 + 2 * k22 + 2 * k32 + k42)

    X.append(x+h)
    Y1.append(y1)
    Y2.append(y2)
  return X, Y1, Y2

def bound_problem(p, q, f, sigma, gamma, delta, n, a, b): # функция формирует и решает трехдиагональную систему краевой задачи
  h = (b - a) / n

  def A_func(x):
    return 1 - p(x) * h / 2
  def C_func(x):
    return -2 + q(x) * h**2
  def B_func(x):
    return 1 + p(x) * h / 2
  def F_func(x):
    return f(x) * h**2

  A = [A_func(a + i * h) for i in np.arange(n+1)]
  B = [B_func(a + i * h) for i in np.arange(n+1)]
  C = [C_func(a + i * h) for i in np.arange(n+1)]
  F = [F_func(a + i * h) for i in np.arange(n+1)]

  F[0] = delta[0]
  F[n] = delta[1]
  C[0] = sigma[0]-gamma[0] / h
  B[0] = gamma[0] / h
  A[n] = -gamma[1] / h
  C[n] = sigma[1] + gamma[1] / h

  alpha = np.zeros(n)
  beta = np.zeros(n)
  alpha[0] = -B[0] / C[0]
  beta[0] = F[0] / C[0]

  for i in np.arange(1, n):
    alpha[i] = -B[i] / (A[i] * alpha[i-1] + C[i])
    beta[i] = (F[i] - A[i] * beta[i-1])/(C[i] + A[i] * alpha[i-1])

  y = np.zeros(n+1)
  y[n] = (F[n]- A[n] * beta[n-1]) / (C[n] + A[n] * alpha[n-1])

  for i in np.arange(n - 1, -1, -1):
    y[i] = alpha[i] * y[i+1] + beta[i]
  return y
# некоторые математические функции
def f(x, y):
  return np.sin(x)-y

def ans1(x):
  return -0.5*np.cos(x)+0.5*np.sin(x)+21/2 * np.exp(-x)

```

```
def f1(x, u, v):
  return -2*x*u**2+v**2-x-1

def f2(x, u, v):
  return 1/(v**2)-u-x/u

def f(x, y):
  return y+2*x -3

def ans1(x):
  return 1+np.exp(x) -2*x

def f1(x, u, v):
  return 3*u-v

def f2(x, u, v):
  return 4*u-v

def ans1(x):
  return (1+x) * np.exp(x)

def ans2(x):
  return (1+2*x) * np.exp(x)
def ans(x):
  return -2 + np.exp(x)
```