

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

### **Отчёт по лабораторной работе № 3**

Дисциплина: Вычислительная математика

Выполнил студент гр. 3530901/10001 \_\_\_\_\_ Д.Л. Симоновский  
(подпись)

Руководитель \_\_\_\_\_ В.Н. Цыган  
(подпись)

“28” февраля 2023 г.

Санкт-Петербург

2023

## Оглавление

<b>Задание:</b> .....	<b>2</b>
<b>Инструменты:</b> .....	<b>2</b>
<b>Ход выполнения работы:</b> .....	<b>2</b>
<i>Порядок действий:</i> .....	2
<i>Первая задача:</i> .....	2
Полная формулировка: .....	2
Решение:.....	3
<i>Вторая задача:</i> .....	3
Полная формулировка: .....	3
Решение:.....	3
Результат: .....	6
<b>Вывод:</b> .....	<b>10</b>
<b>Листинг кода:</b> .....	<b>12</b>
<b>Ссылки:</b> .....	<b>15</b>

## Задание:

Вариант 11:

Привести дифференциальное уравнение:  $ty'' - (t + 1)y' - 2(t - 1)y = 0$   
к системе двух дифференциальных уравнений первого порядка.

Начальные условия:  $y(t = 1) = e^2$ ;  $y'(t = 1) = 2e^2$

Точное решение:  $y(t) = e^{2t}$

Решить на интервале:  $1 \leq t \leq 2$

1. Используя программу RKF45 с шагом печати  $h_{print} = 0.1$  и выбранной вами погрешностью EPS в диапазоне  $0.001 - 0.00001$ , а также составить собственную программу и решить с шагом интегрирования  $h_{int} = 0.1$
2. Используя метод Рунге-Кутты 3-й степени точности.

Сравнить результаты, полученные заданными приближенными способами, с точным решением.

Исследовать влияние величины шага интегрирования  $h_{int}$  на величины локальной и глобальной погрешностей решения заданного уравнения для чего решить уравнение, используя 2 – 3 значения шага интегрирования, существенно меньшие исходной величины 0.1 (например,  $h_{int} = 0.05$ ;  $h_{int} = 0.025$ ;  $h_{int} = 0.0125$ )

## Инструменты:

Для работы был выбран язык программирования Python версии 3.11 по причине удобства его использования для поставленной задачи. Были выбраны следующие библиотеки:

- NumPy – для большей скорости расчетов и простоты обработки
- SciPy – для функции расчета решения диффура
- PrettyTable – для красивого вывода таблицы в консоль
- Matplotlib – для вывода графиков

## Ход выполнения работы:

### Порядок действий:

Поставленное задание легко можно разбить на две глобальные задачи:

1. Сведение поставленной задачи к системе двух дифференциальных уравнений первого порядка.
2. Получить решение используя RKF45 и методы Рунге-Кутты 3-й степени

### Первая задача:

Полная формулировка:

Привести дифференциальное уравнение:  $ty'' - (t + 1)y' - 2(t - 1)y = 0$   
к системе двух дифференциальных уравнений первого порядка.

Решение:

В начале сделаем коэффициент при старшей степени равным 1, для этого поделим уравнение на  $t$ :

$$y'' - \frac{(t+1)}{t}y' - \frac{2(t-1)}{t}y = 0$$

Возьмем  $\alpha_1 = -\frac{(t+1)}{t}$  и  $\alpha_2 = -\frac{2(t-1)}{t}$ , получим:

$$y'' + \alpha_1 y' + \alpha_2 y = 0$$

Решение этого уравнение эквивалентно решению системы  $\frac{dx}{dt} = Ax + f(t)$ , где  $f(t) = 0$ ,  $A$  – матрица Фробениуса вида:  $A = \begin{pmatrix} -\alpha_1 & -\alpha_2 \\ 1 & 0 \end{pmatrix}$ ,  $x = \begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix} = \begin{pmatrix} y' \\ y \end{pmatrix}$ . Таким образом получим систему:

$$\begin{cases} x^{(2)'} = x^{(1)} \\ x^{(1)'} = \alpha_1 x^{(1)} + \alpha_2 x^{(2)} \end{cases}$$

### Вторая задача:

Полная формулировка:

Решить систему дифференциальных уравнений первого порядка.

Начальные условия:  $y(t=1) = e^2$ ;  $y'(t=1) = 2e^2$

Точное решение:  $y(t) = e^{2t}$

Решить на интервале:  $1 \leq t \leq 2$

1. Используя программу RKF45 с шагом печати  $h_{print} = 0.1$  и выбранной вами погрешностью EPS в диапазоне 0.001 – 0.00001, а также составить собственную программу и решить с шагом интегрирования  $h_{int} = 0.1$
2. Используя метод Рунге-Кутты 3-й степени точности.

Сравнить результаты, полученные заданными приближенными способами, с точным решением.

Исследовать влияние величины шага интегрирования  $h_{int}$  на величины локальной и глобальной погрешностей решения заданного уравнения для чего решить уравнение, используя 2 – 3 значения шага интегрирования, существенно меньшие исходной величины 0.1 (например,  $h_{int} = 0.05$ ;  $h_{int} = 0.025$ ;  $h_{int} = 0.0125$ )

Решение:

Для дальнейшего использования сразу же зададим функцию, для получения значений системы уравнений:

```
def f(t, x):
    dX = np.zeros(X.shape)
    dX[0] = X[1]
    dX[1] = (t + 1) / t * X[1] + 2 * (t - 1) / t * X[0]
    return dX
```

Так же нам понадобится функция для получения точно значения решения (чтоб сравнивать погрешности):

```
def g(T):
    return np.e ** (2 * T)
```

Далее нам понадобится функция, которая будет моделировать RKF45, к счастью, в библиотеке `scipy` уже имеется подходящий вариант, правда требующий дополнительной настройки, а конкретно передачу параметра `'dopri5'` для настройки интегратора и выставления параметра погрешности `atol` на значение `0.0001`:

```
def rkf45(f, T, X0):
    runge = ode(f).set_integrator('dopri5', atol=0.0001).set_initial_value(X0, T[0])
    X = [X0, *[runge.integrate(T[i]) for i in range(1, len(T))]]
    return np.array([i[0] for i in X])
```

Метод Рунге-Кутты по заданию необходимо написать самостоятельно. Для начала вспомним, как выглядит метод Рунге-Кутты третьей степени:

$$\begin{cases} x_{n+1} = x_n + \frac{2k_1 + 3k_2 + 4k_3}{9}, \\ k_1 = hf(t_n, x_n), \\ k_2 = hf(t_n + \frac{h}{2}, x_n + \frac{k_1}{2}), \\ k_3 = hf(t_n + \frac{3h}{4}, x_n + \frac{3k_2}{4}) \end{cases}$$

Теперь необходимо реализовать его в виде метода:

```
def Runge_Kutta(f, T, X0):
    X = np.zeros((len(T), len(X0)))
    X[0] = X0
    h = T[1] - T[0]
    for i in range(0, len(T) - 1):
        k_1 = h * f(T[i], X[i])
        k_2 = h * f(T[i] + h / 2, X[i] + k_1 / 2)
        k_3 = h * f(T[i] + 3 * h / 4, X[i] + 3 * k_2 / 4)
        X[i + 1] = (X[i] + (2 * k_1 + 3 * k_2 + 4 * k_3) / 9)
    return X[:, 0]
```

Все методы, необходимые для подсчета реализованы. Создадим отдельную функцию для подсчета решения с разным шагом (т.к. этого требует задание), в ней сразу же зададим начальные значения, узлы, по которым будет искаться решение и значение функции в этих точках (для дальнейшего подсчета погрешности):

```
def evaluate(h):
    # Начальные значения
    X0 = np.array([np.e ** 2, 2 * np.e ** 2])
    # Значения в узлах
    T = np.arange(1, 2 + h, h)
    Y = g(T)
```

Воспользовавшись методами, которые были приведены выше, выполним расчеты:

```
# Расчет RKF45
Y_RKF45 = rkf45(f, T, X0)
# Расчет Рунге-Кутты
Y_Runge_Kutta = Runge_Kutta(f, T, X0)
# Погрешности
Y_RKF45_error = Y - Y_RKF45
Y_Runge_Kutta_error = Y - Y_Runge_Kutta
```

Далее необходимо вывести полученные значения на экран, используем для этого библиотеку `Matplotlib`. Отдельно выведем графики погрешности и полученные значения:

```

def print_one_graph(t, y, title, id, count_graphs):
    """
    Функция для отрисовки одного графика
    """
    plt.subplot(1, count_graphs, id)
    plt.xlabel('t')
    plt.ylabel('y')
    plt.grid()
    plt.title(title)
    plt.plot(t, y, '-o')

def print_graph(t_find, y_real, Y_RKF45, Y_Runge_Kutta, h):
    """
    Функция для отрисовки всех графиков
    """
    mpl.use('TkAgg')
    plt.figure(figsize=(15, 4))
    print_one_graph(t_find, y_real, 'Исходный график', 1, 3)
    print_one_graph(t_find, Y_RKF45, 'График RKF45', 2, 3)
    print_one_graph(t_find, Y_Runge_Kutta, 'График Рунге-Кутты', 3, 3)
    plt.savefig(f"Graphs_{h}.jpg")
    plt.show()

def print_error_graph(t_find, Y_RKF45_error, Y_Runge_Kutta_error, h):
    """
    Функция для отрисовки погрешности
    """
    mpl.use('TkAgg')
    plt.figure(figsize=(15, 4))
    # Собственно сам график
    print_one_graph(t_find, Y_RKF45_error, 'Погрешность RKF45', 1, 2)
    print_one_graph(t_find, Y_Runge_Kutta_error, 'Погрешность Рунге-Кутты', 2, 2)
    plt.savefig(f"Error_{h}.jpg")
    plt.show()

```

Вызовем их из функции evaluate, для отрисовки всех графиков:

```

# Рисуем графики
print_graph(T, Y, Y_RKF45, Y_Runge_Kutta, h)
print_error_graph(T, Y_RKF45_error, Y_Runge_Kutta_error, h)

```

Далее для дополнительного анализа выведем значения в консоль, используя библиотеку prettytable:

```
def print_table(t_find, y_real, Y_RKF45, Y_RKF45_error, Y_Runge_Kutta, Y_Runge_Kutta_error,
h):
    print(f'h = {h}')
    koef = {0.1: 1, 0.05: 2, 0.025: 4, 0.0125: 8}.get(h)
    pt = PrettyTable()
    pt.add_column('t', [f'{i:.1f}' for i in t_find[::koef]])
    pt.add_column('real y', [f'{i:.15f}' for i in y_real[::koef]])
    pt.add_column('RKF45 y', [f'{i:.15f}' for i in Y_RKF45[::koef]])
    pt.add_column('Delta RKF45 y', [f'{i:.15f}' for i in Y_RKF45_error[::koef]])
    pt.add_column('Runge Kutta y', [f'{i:.15f}' for i in Y_Runge_Kutta[::koef]])
    pt.add_column('Delta Runge Kutta y', [f'{i:.15f}' for i in
Y_Runge_Kutta_error[::koef]])
    print(pt)
    print('First step of RKF45:', Y_RKF45_error[1])
    print('First step of Runge Kutta:', Y_Runge_Kutta_error[1])
    print('Global of RKF45:', Y_RKF45_error[::koef].sum())
    print('Global of Runge Kutta:', Y_Runge_Kutta_error[::koef].sum())
    print('h^4 is about:', h ** 4)
    print('h^4 / Runge Kutta first step:', h ** 4 / Y_Runge_Kutta_error[1])
    print('=' * 110)
```

Еще необходимо выполнить анализ погрешности, для этого будем сохранять её при вызове функции evaluate, а после выведем все это на экран в виде таблицы, используя уже известную библиотеку:

```
def print_table_error(Y_RKF45_error, Y_Runge_Kutta_error, h_list):
    pt = PrettyTable()
    pt.add_column('h', [f'{i:.4f}' for i in h_list])
    pt.add_column("Runge Kutta Error local", [f'{i[1]:.15f}' for i in Y_Runge_Kutta_error])
    pt.add_column('h**4 / Runge Kutta Error local',
[f'{i ** 4 / j[1]:.15f}' for i, j in zip(h_list, Y_Runge_Kutta_error)])

    print(pt)
    pt.clear()
    pt.add_column('h', [f'{i:.4f}' for i in h_list])
    pt.add_column("Runge Kutta Error global", [f'{i.sum():.15f}' for i in
Y_Runge_Kutta_error])
    pt.add_column('h**2 / Runge Kutta Error global',
[f'{i ** 2 / j.sum():.15f}' for i, j in zip(h_list, Y_Runge_Kutta_error)])

    print(pt)
    pt.clear()
    pt.add_column('h', [f'{i:.4f}' for i in h_list])
    pt.add_column("RKF45 Error local", [f'{i[1]:.15f}' for i in Y_RKF45_error])
    pt.add_column("RKF45 Error global", [f'{i.sum():.15f}' for i in Y_RKF45_error])
    pt.add_column("Runge Kutta Error local", [f'{i[1]:.15f}' for i in Y_Runge_Kutta_error])
    pt.add_column("Runge Kutta Error global", [f'{i.sum():.15f}' for i in
Y_Runge_Kutta_error])
    print(pt)
```

## Результат:

Как и ожидалось, полученные графики практически не отличаются:

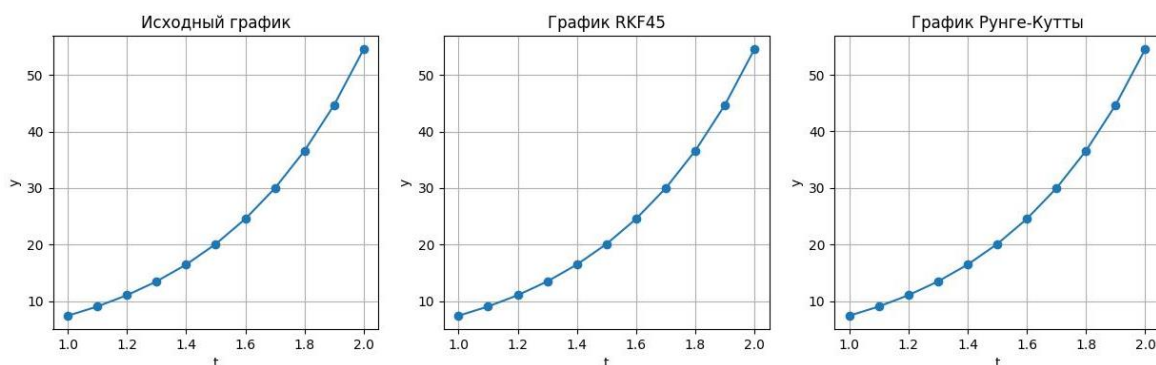


Рис. 1. График решения при  $h = 0.1$

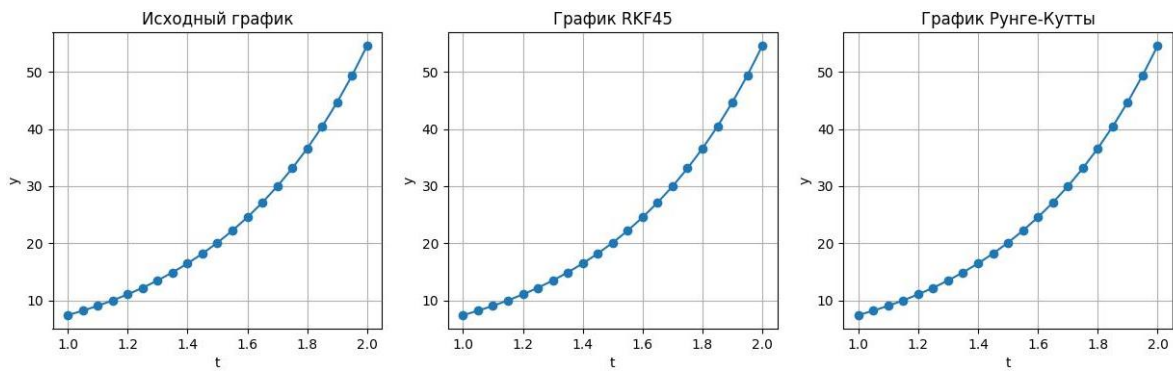


Рис. 2. График решения при  $h = 0.05$

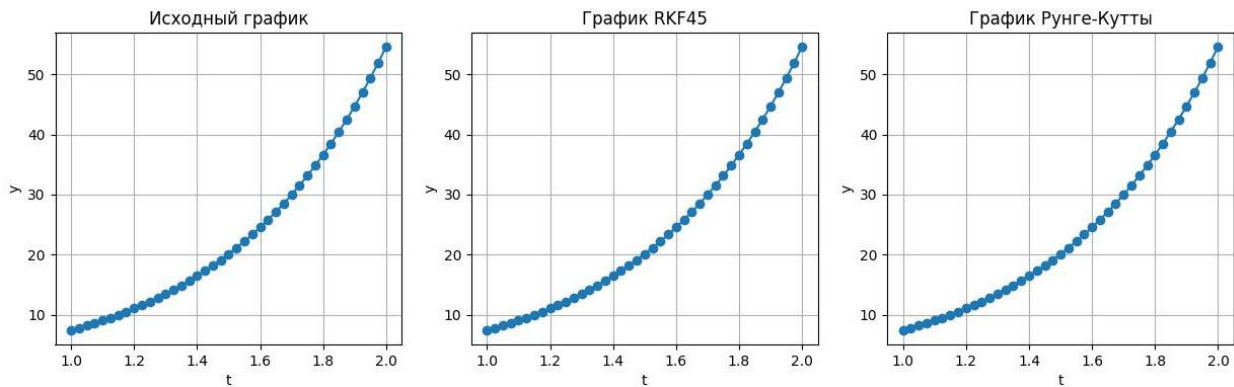


Рис. 3. График решения при  $h = 0.025$

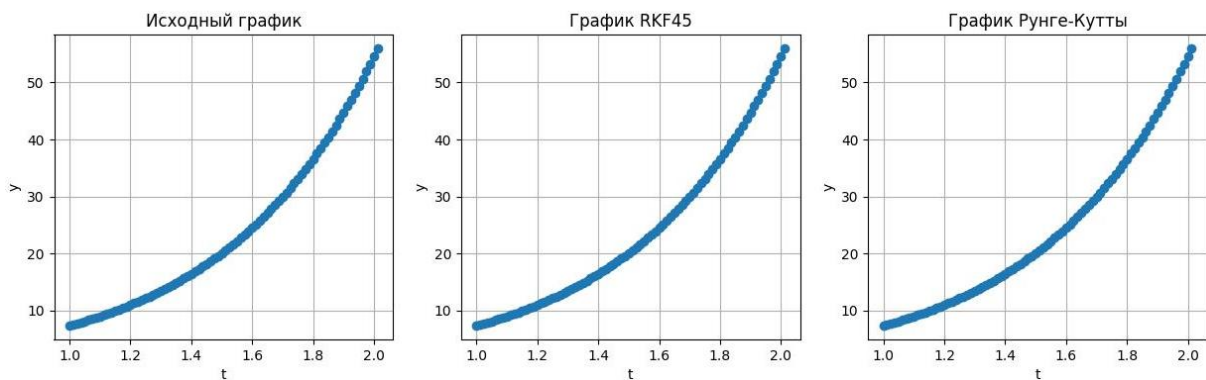


Рис. 4. График решения при  $h = 0.0125$

Большой интерес представляют графики погрешности:

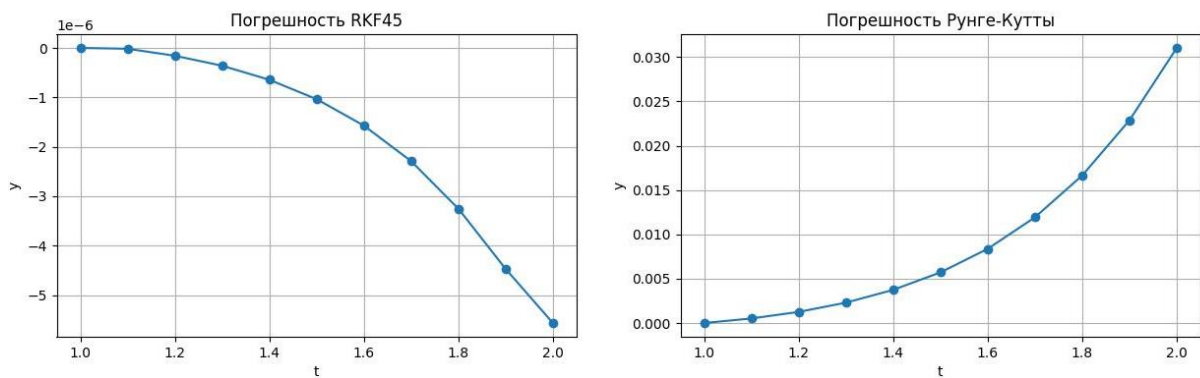


Рис. 5. График погрешности при  $h = 0.1$



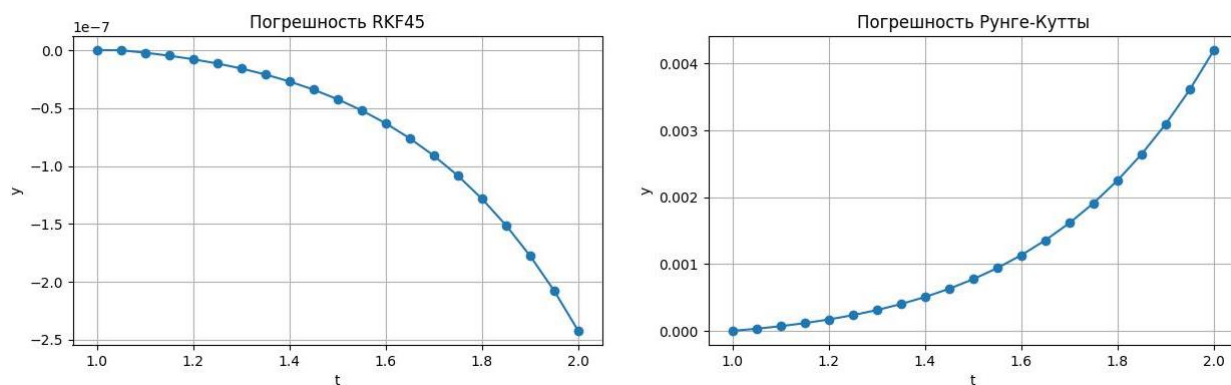


Рис. 6. График погрешности при  $h = 0.05$

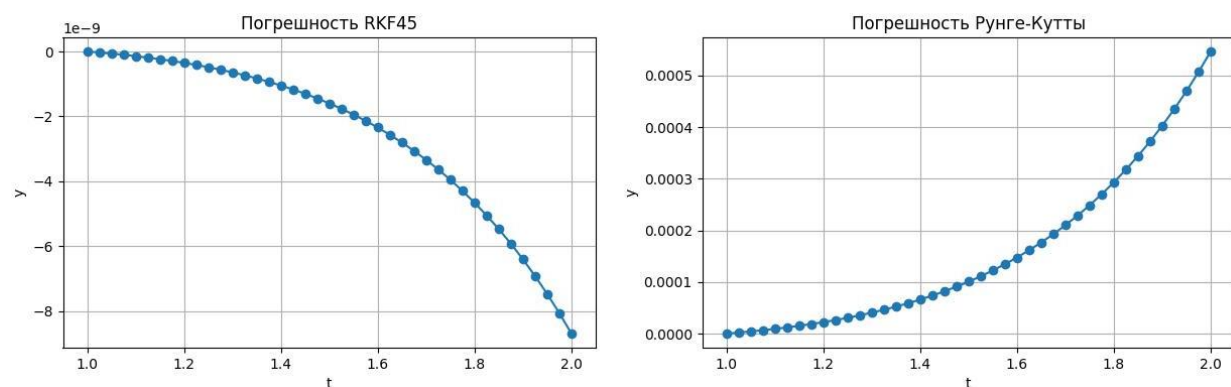


Рис. 7. График погрешности при  $h = 0.025$

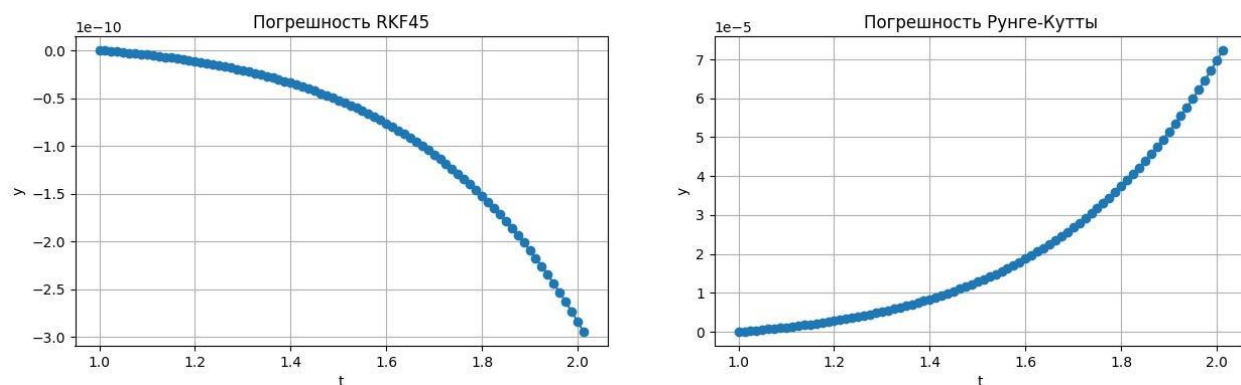


Рис. 8. График погрешности при  $h = 0.0125$

Как можно заметить, форма самого графика остается одинаковой, однако порядок сильно меняется, при изменении шага.

Так же даже по графику заметно, что погрешность метода Рунге-Кутты третьей степени значительно проигрывает программе RKF45, что достаточно ожидаемо, ведь в основе этой программы лежат методы Рунге-Кутты четвертой и пятой степени точности.

Для анализа зависимости шага интегрирования  $h_{int}$  и величины локальной и глобальной погрешностей решения заданного уравнения обратимся к численным результатам исследования:

h = 0.1

t	real y	RKF45 y	Delta RKF45 y	Runge Kutta y	Delta Runge Kutta y
1.0	7.389056098930650	7.389056098930650	0.000000000000000	7.389056098930650	0.000000000000000
1.1	9.025013499434122	9.025013516869686	-0.000000017435564	9.024500515493969	0.000512983940153
1.2	11.023176380641605	11.023176538875457	-0.000000158233853	11.021923296256636	0.001253084384969
1.3	13.463738035001695	13.463738395525420	-0.000000360523725	13.461442319161440	0.002295715840255
1.4	16.444646771097059	16.444647415729236	-0.000000644632177	16.440908219135842	0.003738551961217
1.5	20.085536923187682	20.085537960060527	-0.000001036872845	20.079829238304580	0.005707684883102
1.6	24.532530197109370	24.532531768309873	-0.000001571200503	24.524164776382666	0.008365420726705
1.7	29.964100047397046	29.964102338701789	-0.000002291304742	29.952179913555369	0.011920133841677
1.8	36.598234443678031	36.598237696934191	-0.000003253256160	36.581595734422294	0.016638709255737
1.9	44.701184493300886	44.701188964687368	-0.000004471386482	44.678322256974440	0.022862236326446
2.0	54.598150033144321	54.598155598718769	-0.000005565574448	54.567124249851460	0.031025783292861

First step of RKF45: -1.743556410360725e-08

First step of Runge Kutta: 0.0005129839401529779

Global of RKF45: -1.937042049959814e-05

Global of Runge Kutta: 0.10432030445312179

h^4 is about: 0.0001000000000000000

h^4 / Runge Kutta first step: 0.1949378765545347

Рис. 9. Результат исследования при h = 0.1

h = 0.05

t	real y	RKF45 y	Delta RKF45 y	Runge Kutta y	Delta Runge Kutta y
1.0	7.389056098930650	7.389056098930650	0.000000000000000	7.389056098930650	0.000000000000000
1.1	9.025013499434122	9.025013501625160	-0.000000002191038	9.024944064111462	0.000069435322660
1.2	11.023176380641605	11.023176388458154	-0.000000007816549	11.023006764304872	0.000169616336732
1.3	13.463738035001695	13.463738050827361	-0.000000015825666	13.463427281404844	0.000310753596851
1.4	16.444646771097059	16.444646798095160	-0.000000026998102	16.444140699310093	0.000506071786965
1.5	20.085536923187682	20.085536965529673	-0.000000042341991	20.084764279314374	0.000772643873308
1.6	24.532530197109370	24.532530260266185	-0.000000063156815	24.531397750236174	0.001132446873196
1.7	29.964100047397046	29.964100138510030	-0.000000091112984	29.962486350914514	0.001613696482533
1.8	36.598234443678031	36.598234572030442	-0.000000128352411	36.595981911389273	0.002252532288757
1.9	44.701184493300886	44.701184670916255	-0.000000177615370	44.698089349913182	0.003095143387704
2.0	54.598150033144321	54.598150275544839	-0.000000242400517	54.593949586335228	0.004200446809094

First step of RKF45: -7.847944516470307e-11

First step of Runge Kutta: 3.1413899460375205e-05

Global of RKF45: -1.4647554884561487e-06

Global of Runge Kutta: 0.025995142229312762

h^4 is about: 6.250000000000001e-06

h^4 / Runge Kutta first step: 0.1989565162988954

Рис. 10. Результат исследования при h = 0.05

h = 0.025

t	real y	RKF45 y	Delta RKF45 y	Runge Kutta y	Delta Runge Kutta y
1.0	7.389056098930650	7.389056098930650	0.000000000000000	7.389056098930650	0.000000000000000
1.1	9.025013499434113	9.025013499577801	-0.000000000143688	9.025004466700494	0.000009032733619
1.2	11.023176380641585	11.023176380992583	-0.000000000350997	11.023154315441113	0.000022065200472
1.3	13.463738035001660	13.463738035644724	-0.000000000643064	13.463697609276819	0.000040425724841
1.4	16.444646771097002	16.444646772144257	-0.0000000001047255	16.444580936340852	0.000065834756150
1.5	20.085536923187593	20.085536924786492	-0.0000000001598899	20.085436409796969	0.000100513390624
1.6	24.532530197109239	24.532530199452719	-0.0000000002343480	24.532382876383917	0.000147320725322
1.7	29.964100047396858	29.964100050736249	-0.0000000003339391	29.963890119904967	0.000209927491891
1.8	36.598234443677775	36.598234448339191	-0.0000000004661416	36.597941408375718	0.000293035302057
1.9	44.701184493300531	44.701184499705683	-0.0000000006405152	44.700781840110089	0.000402653190442
2.0	54.598150033143838	54.598150041836362	-0.0000000008692524	54.597603587064640	0.000546446079198

First step of RKF45: -3.091749078976136e-11

First step of Runge Kutta: 1.9436371978542866e-06

Global of RKF45: -1.0308740350239987e-07

Global of Runge Kutta: 0.006480486576304401

h^4 is about: 3.9062500000000007e-07

h^4 / Runge Kutta first step: 0.2009762935342242

Рис. 11. Результат исследования при h = 0.025

h = 0.0125

t	real y	RKF45 y	Delta RKF45 y	Runge Kutta y	Delta Runge Kutta y
1.0	7.389056098930650	7.389056098930650	0.000000000000000	7.389056098930650	0.000000000000000
1.1	9.025013499434113	9.025013499438806	-0.000000000004693	9.025012347561770	0.000001151872343
1.2	11.023176380641585	11.023176380653045	-0.000000000011459	11.023173566841647	0.000002813799938
1.3	13.463738035001660	13.463738035022654	-0.000000000020995	13.463732879827486	0.000005155174174
1.4	16.444646771097002	16.444646771131190	-0.000000000034188	16.444638375705598	0.000008395391404
1.5	20.085536923187593	20.085536923239793	-0.000000000052200	20.085524105495647	0.000012817691946
1.6	24.532530197109239	24.532530197185743	-0.000000000076504	24.532511410433283	0.000018786675955
1.7	29.964100047396858	29.964100047505873	-0.000000000109015	29.964073276951098	0.000026770445761
1.8	36.598234443677775	36.598234443829952	-0.000000000152177	36.598197075112964	0.000037368564811
1.9	44.701184493300531	44.701184493509636	-0.000000000209106	44.701133145977167	0.000051347323364
2.0	54.598150033143838	54.598150033427601	-0.000000000283762	54.598080348967869	0.000069684175969

First step of RKF45: -4.920508445138694e-13

First step of Runge Kutta: 1.2086842193781422e-07

Global of RKF45: -6.880966907374386e-09

Global of Runge Kutta: 0.001689689679643891

h^4 is about: 2.4414062500000004e-08

h^4 / Runge Kutta first step: 0.201988758590402

Рис. 12. Результат исследования при h = 0.0125

Можно заметить, что локальная погрешность первого шага метода Рунге-Кутты пропорциональна четвертой степени шага интегрирования, как и было предсказано:

h	Runge Kutta Error local	h**4 / Runge Kutta Error local
0.1000	0.000512983940153	0.194937876554535
0.0500	0.000031413899460	0.198956516298895
0.0250	0.000001943637198	0.200976293534224
0.0125	0.000000120868422	0.201988758590402

Из анализа результатов глобальной погрешности заметна её зависимость от квадрата шага интегрирования, что явно сильно хуже, чем если рассматривать локальную погрешность:

h	Runge Kutta Error global	h**2 / Runge Kutta Error global
0.1000	0.104320304453122	0.095858615946560
0.0500	0.025995142229313	0.096171814639311
0.0250	0.006480486576304	0.096443375453517
0.0125	0.001689689679644	0.092472601260683

Так же из приведенных результатов (рис. 5-12) более заметна разница между методами Рунге-Кутты 3 степени точности и программой RKF45, как после первого шага, так и при подсчете глобальной погрешности, что сильнее проявляется при уменьшении шага:

h	RKF45 Error local	RKF45 Error global	Runge Kutta Error local	Runge Kutta Error global
0.1000	-0.000000017435564	-0.000019370420500	0.000512983940153	0.104320304453122
0.0500	-0.000000000078479	-0.000001464755488	0.000031413899460	0.025995142229313
0.0250	-0.000000000030917	-0.000000103087404	0.000001943637198	0.006480486576304
0.0125	-0.00000000000492	-0.000000006880967	0.000000120868422	0.001689689679644

## Вывод:

В ходе выполненной работы мы привели линейное дифференциальное уравнение к системе двух дифференциальных уравнений первого порядка, после чего решили эту систему при заданных НУ, используя методы Рунге-Кутты третьей степени точности, а также программу RKF45. Была найдена зависимость шага интегрирования h и величины глобальной и

локальной погрешности а также наглядно продемонстрирована разница между используемыми методами.

## Листинг кода:

```
import numpy as np
from scipy.integrate import ode
import matplotlib as mpl
import matplotlib.pyplot as plt
from prettytable import PrettyTable

def rkf45(f, T, X0):
    """
    Решает `x' = f(t, x)` для каждого `t` в `T`
    С начальным значением `X0`, используя аналог rkf45
    """
    runge = ode(f).set_integrator('dopri5', atol=0.0001).set_initial_value(X0, T[0])
    X = [X0, *[runge.integrate(T[i]) for i in range(1, len(T))]]
    return np.array([i[0] for i in X])

def Runge_Kutta(f, T, X0):
    """
    Решает `x' = f(t, x)` для каждого `t` в `T`
    С начальным значением `X0`, используя формулы Рунге-Кутты 3 степени
    """
    X = np.zeros((len(T), len(X0)))
    X[0] = X0
    h = T[1] - T[0]
    for i in range(0, len(T) - 1):
        k_1 = h * f(T[i], X[i])
        k_2 = h * f(T[i] + h / 2, X[i] + k_1 / 2)
        k_3 = h * f(T[i] + 3 * h / 4, X[i] + 3 * k_2 / 4)
        X[i + 1] = (X[i] + (2 * k_1 + 3 * k_2 + 4 * k_3) / 9)
    return X[:, 0]

def f(t, X):
    """
    Правая часть `x' = f(t, x)`.
    """
    dX = np.zeros(X.shape)
    dX[0] = X[1]
    dX[1] = (t + 1) / t * X[1] + 2 * (t - 1) / t * X[0]
    return dX

def g(T):
    """
    Точное решение
    """
    return np.e ** (2 * T)

def print_one_graph(t, y, title, id, count_graphs):
    """
    Функция для отрисовки одного графика
    """
    plt.subplot(1, count_graphs, id)
    plt.xlabel('t')
    plt.ylabel('y')
    plt.grid()
    plt.title(title)
    plt.plot(t, y, '-o')
```

```

def print_graph(t_find, y_real, Y_RKF45, Y_Runge_Kutta, h):
    """
    Функция для отрисовки всех графиков
    """
    mpl.use('TkAgg')
    plt.figure(figsize=(15, 4))
    print_one_graph(t_find, y_real, 'Исходный график', 1, 3)
    print_one_graph(t_find, Y_RKF45, 'График RKF45', 2, 3)
    print_one_graph(t_find, Y_Runge_Kutta, 'График Рунге-Кутты', 3, 3)
    plt.savefig(f"Graphs_{h}.jpg")
    plt.show()

def print_error_graph(t_find, Y_RKF45_error, Y_Runge_Kutta_error, h):
    """
    Функция для отрисовки погрешности
    """
    mpl.use('TkAgg')
    plt.figure(figsize=(15, 4))
    # Собственно сам график
    print_one_graph(t_find, Y_RKF45_error, 'Погрешность RKF45', 1, 2)
    print_one_graph(t_find, Y_Runge_Kutta_error, 'Погрешность Рунге-Кутты', 2, 2)
    plt.savefig(f"Error_{h}.jpg")
    plt.show()

def print_table(t_find, y_real, Y_RKF45, Y_RKF45_error, Y_Runge_Kutta,
Y_Runge_Kutta_error, h):
    """
    Функция для отрисовки таблицы
    """
    print(f'h = {h}')
    koef = {0.1: 1, 0.05: 2, 0.025: 4, 0.0125: 8}.get(h)
    pt = PrettyTable()
    pt.add_column('t', [f'{i:.1f}' for i in t_find[::koef]])
    pt.add_column('real y', [f'{i:.15f}' for i in y_real[::koef]])
    pt.add_column('RKF45 y', [f'{i:.15f}' for i in Y_RKF45[::koef]])
    pt.add_column('Delta RKF45 y', [f'{i:.15f}' for i in Y_RKF45_error[::koef]])
    pt.add_column('Runge Kutta y', [f'{i:.15f}' for i in Y_Runge_Kutta[::koef]])
    pt.add_column('Delta Runge Kutta y', [f'{i:.15f}' for i in
Y_Runge_Kutta_error[::koef]])
    print(pt)
    print('First step of RKF45:', Y_RKF45_error[1])
    print('First step of Runge Kutta:', Y_Runge_Kutta_error[1])
    print('Global of RKF45:', Y_RKF45_error.sum())
    print('Global of Runge Kutta:', Y_Runge_Kutta_error.sum())
    print('h^4 is about:', h ** 4)
    print('h^4 / Runge Kutta first step:', h ** 4 / Y_Runge_Kutta_error[1])
    print('=' * 110)

```

```

def evaluate(h):
    """
    Получение решения при разных шагах
    """
    # Начальные значения
    X0 = np.array([np.e ** 2, 2 * np.e ** 2])
    # Значения в узлах
    T = np.arange(1, 2 + h, h)
    Y = g(T)
    # Расчет RKF45
    Y_RKF45 = rkf45(f, T, X0)
    # Расчет Рунге-Кутты
    Y_Runge_Kutta = Runge_Kutta(f, T, X0)
    # Погрешности
    Y_RKF45_error = Y - Y_RKF45
    Y_Runge_Kutta_error = Y - Y_Runge_Kutta
    # Рисуем графики
    print_graph(T, Y, Y_RKF45, Y_Runge_Kutta, h)
    print_error_graph(T, Y_RKF45_error, Y_Runge_Kutta_error, h)
    # Выводим данные в консоль
    print_table(T, Y, Y_RKF45, Y_RKF45_error, Y_Runge_Kutta, Y_Runge_Kutta_error, h)
    return Y_RKF45_error, Y_Runge_Kutta_error

def print_table_error(Y_RKF45_error, Y_Runge_Kutta_error, h_list):
    pt = PrettyTable()
    pt.add_column('h', [f'{i:.4f}' for i in h_list])
    pt.add_column("Runge Kutta Error local", [f'{i[1]:.15f}' for i in
Y_Runge_Kutta_error])
    pt.add_column('h**4 / Runge Kutta Error local',
[f'{i ** 4 / j[1]:.15f}' for i, j in zip(h_list, Y_Runge_Kutta_error)])
    print(pt)
    pt.clear()
    pt.add_column('h', [f'{i:.4f}' for i in h_list])
    pt.add_column("Runge Kutta Error global", [f'{i.sum():.15f}' for i in
Y_Runge_Kutta_error])
    pt.add_column('h**2 / Runge Kutta Error global',
[f'{i ** 2 / j.sum():.15f}' for i, j in zip(h_list,
Y_Runge_Kutta_error)])
    print(pt)
    pt.clear()
    pt.add_column('h', [f'{i:.4f}' for i in h_list])
    pt.add_column("RK45 Error local", [f'{i[1]:.15f}' for i in Y_RKF45_error])
    pt.add_column("RK45 Error global", [f'{i.sum():.15f}' for i in Y_RKF45_error])
    pt.add_column("Runge Kutta Error local", [f'{i[1]:.15f}' for i in
Y_Runge_Kutta_error])
    pt.add_column("Runge Kutta Error global", [f'{i.sum():.15f}' for i in
Y_Runge_Kutta_error])
    print(pt)

def main():
    h_list = [0.1 / (2 ** i) for i in range(4)]
    Y_RKF45_error = []
    Y_Runge_Kutta_error = []
    for h in h_list:
        error = evaluate(h)
        Y_RKF45_error.append(error[0])
        Y_Runge_Kutta_error.append(error[1])
    print_table_error(Y_RKF45_error, Y_Runge_Kutta_error, h_list)

if __name__ == '__main__':
    main()

```

## Ссылки:

Листинг код: [github.com](https://github.com)

Документация по SciPy: [docs.scipy.org](https://docs.scipy.org)