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. 3530901/10003 _____ Я.А. Иванов
(подпись)

Руководитель _____ В.Н. Цыган
(подпись)

“27” марта 2023 г.

Санкт-Петербург
2023

Оглавление

Задание:

Привести дифференциальное уравнение: $t^2 y'' - 6y = 0$
к системе двух дифференциальных уравнений первого порядка.
Решить на интервале $1 \le t \le 2$
Начальные условия: $y|_{t=1} = 1$; $y'|_{t=1} = 3$
Точное решение: $y(t) = t^3$
При выполнении лабораторной работы решить заданное уравнение:
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 $h_{int} = 0.1$:
10) используя явный метод ломаных Эйлера;
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, $h_{int} = 0.05$; $h_{int} = 0.025$; $h_{int} = 0.0125$).

Пункт 1 и 10. EPS=0.00001

Инструменты:

Для работы был выбран язык программирования Python версии 3.09 ввиду наличия необходимых библиотек для выполнения поставленной задачи, а именно:

- NumPy – для большей скорости расчетов
- pandas – для красивого вывода в консоль таблицы
- MatplotLib – для вывода графиков
- SciPy – для функций расчета интерполяции и интеграла

Ход выполнения работы:

Приведение к системе первого порядка:

Для преобразования дифференциального уравнения в систему двух уравнений первого порядка мы вводим новую переменную y1=y и y2=y', а затем находим их производные по t:

y1' = y' = y2

y2' = y'' = t^2 - 6y1

Таким образом, мы получаем систему:

y1' = y2

y2' = t^2 — 6y1

Зададим начальные условия и получившуюся систему диф. Уравнений:

```
# Функция правой части системы дифференциальных уравнений
def f(t, y):
    return [y[1], 6 * y[0] / t ** 2]


# Задаем начальные условия
y0 = [1, 3]
t0 = 1
tmax = 2
eps = 0.00001
```

RKF45

Функция solve_rkf45 использует метод RKF45 для решения системы дифференциальных уравнений. Она использует функцию правой части f, задает начальные условия и использует метод set_integrator для настройки метода RKF45. Она возвращает массив пар из значения t и соответствующего результата работы RKF45.

```
# Решение системы дифференциальных уравнений методом RKF45
def solve_rkf45(h):
    t = np.arange(t0, tmax + h, h)
    rk_integ = ode(f).set_integrator("dopri5", atol=eps).set_initial_value(y0, t[0])
    X = np.array([y0, *[rk_integ.integrate(t[i]) for i in range(1, len(t))]])
    return t, X[:, 0]
```

Метод Эйлера

Функция solve_euler принимает на вход шаг и возвращает данные аналогично с RKF45

```
# Решение системы дифференциальных уравнений методом Эйлера
def solve_euler(h):
    t = np.arange(t0, tmax + h, h)
    y = np.zeros_like(t)
    y[0] = y0[0]
    z = np.zeros_like(t)
    z[0] = y0[1]
    for i in range(1, len(t)):
        y[i] = y[i - 1] + h * z[i - 1]
        z[i] = z[i - 1] + h * 6 * y[i - 1] / t[i - 1] ** 2
    return t, y
```

Обработка результатов работы функции и вычисление погрешностей

Получаем данные и вычисляем погрешности

```
x_values = np.arange(t0, tmax + h_int, h_int)

# вычислим значения функций
rkf45_values = solve_rkf45(h_int)
euler_values = solve_euler(h_int)
exact_values = [t ** 3 for t in x_values]

# вычислим погрешности
rkf45_3_errors = np.abs(rkf45_values[1] - rkf45_values[0] ** 3)
euler_errors = np.abs(euler_values[1] - euler_values[0] ** 3)
print("RKF45 global err: ",np.sum(rkf45_3_errors))
print("Euler global err: ", np.sum(euler_errors))
```

Отобразим таблицу и графики

```
    # вывод таблицы
    results = pd.DataFrame({
        'h': euler_values[0],
        'Value RKF45': rkf45_values[1],
        'Value Euler': euler_values[1],
        'Exact': exact_values,
        'Error RKF45': rkf45_3_errors,
        'Error Euler': euler_errors,
    })

    print(results.to_string(index=False))
    # выводим график значений
    plt.figure(figsize=(15, 4))
    y = rkf45_values[1]
    t = euler_values[0]
    print_graph(t, y, 'RKF45, step=' + str(h_int), 1, 3)

    t, y = euler_values
    print_graph(t, y, 'Эйлер', 2, 3)

    print_graph(x_values, exact_values, 'Exact solution', 3, 3)

    plt.savefig("Graphs_h_" + str(h_int) + ".jpg")
    plt.show()
    # выводим график погрешности
    plt.figure(figsize=(15, 4))

    y = rkf45_3_errors
    t = rkf45_values[0]
    print_graph(t, y, 'RKF45, step=' + str(h_int), 1, 2)

    y = euler_errors
    t = euler_values[0]
    print_graph(t, y, 'Эйлер', 2, 2)

    plt.savefig("error_h_" + str(h_int) + ".jpg")
    plt.show()
```

Выполним перебор для значений h_list

```
h_list = [0.1, 0.05, 0.025, 0.0125]
for h_int in h_list:
#далее идет код из пунктов выше
```

Результаты работы программы:

Для h_int = 0.1

Таблица:

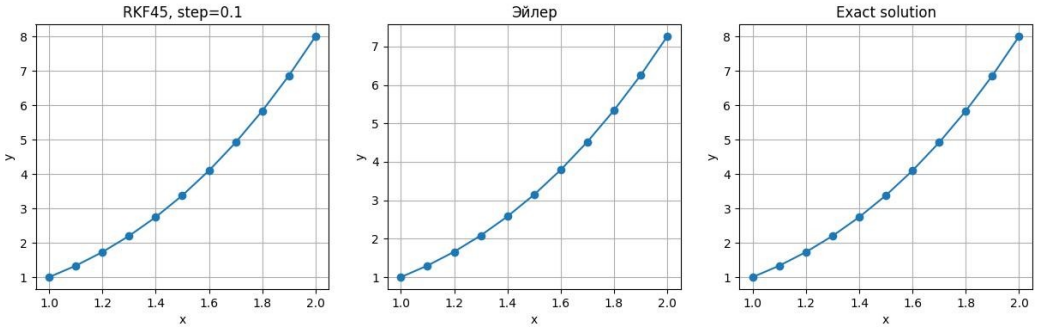
h_int = 0.1

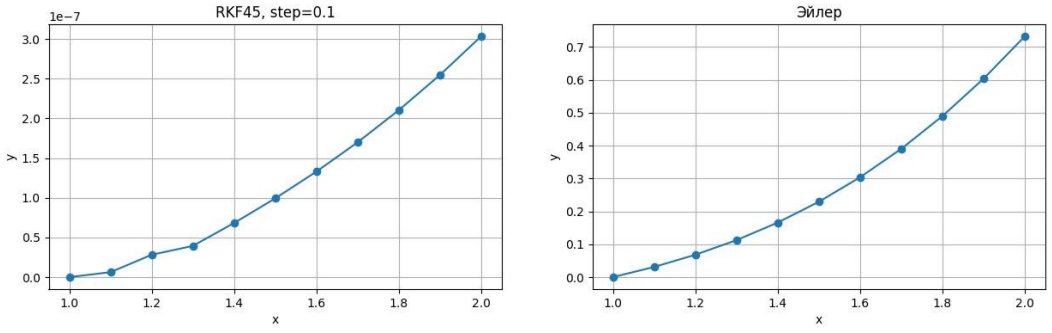
h	Value RKF45	Value Euler	Exact	Error RKF45	Error Euler
1.0	1.000	1.000000	1.000	0.000000e+00	0.000000
1.1	1.331	1.300000	1.331	6.254157e-09	0.031000
1.2	1.728	1.660000	1.728	2.818067e-08	0.068000
1.3	2.197	2.084463	2.197	3.931797e-08	0.112537
1.4	2.744	2.578092	2.744	6.818008e-08	0.165908
1.5	3.375	3.145726	3.375	9.921412e-08	0.229274
1.6	4.096	3.792282	4.096	1.329554e-07	0.303718
1.7	4.913	4.522723	4.913	1.698581e-07	0.390277
1.8	5.832	5.342046	5.832	2.103206e-07	0.489954
1.9	6.859	6.255266	6.859	2.547031e-07	0.603734
2.0	8.000	7.267413	8.000	3.033378e-07	0.732587

RKF45 global err: 1.3123220110600187e-06

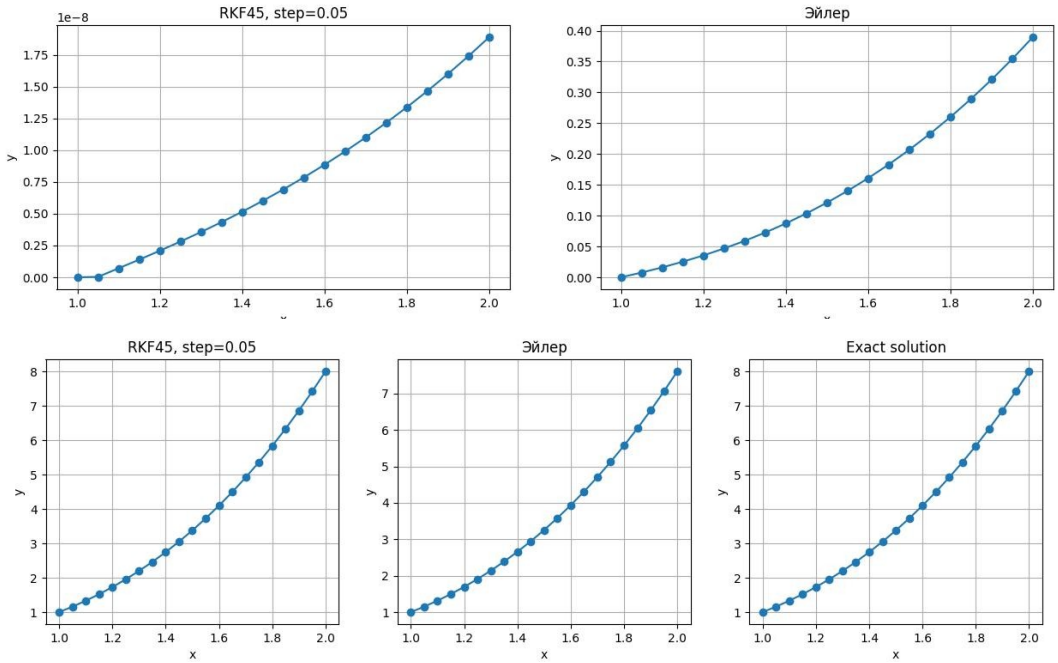
Euler global err: 3.1269889304508016

Графики:

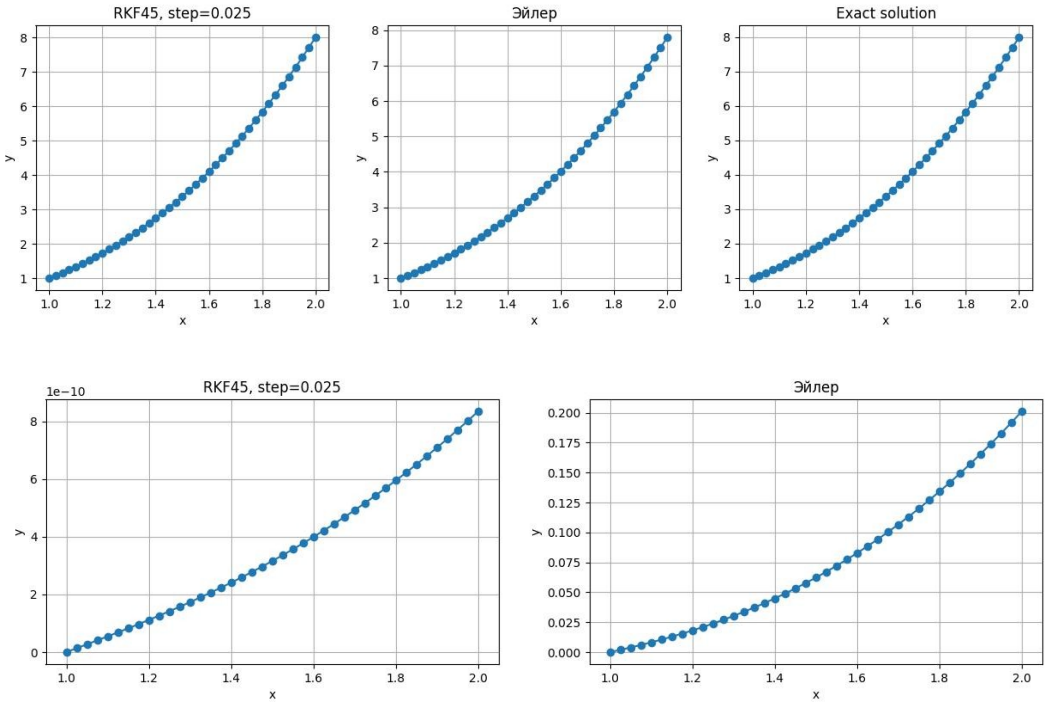




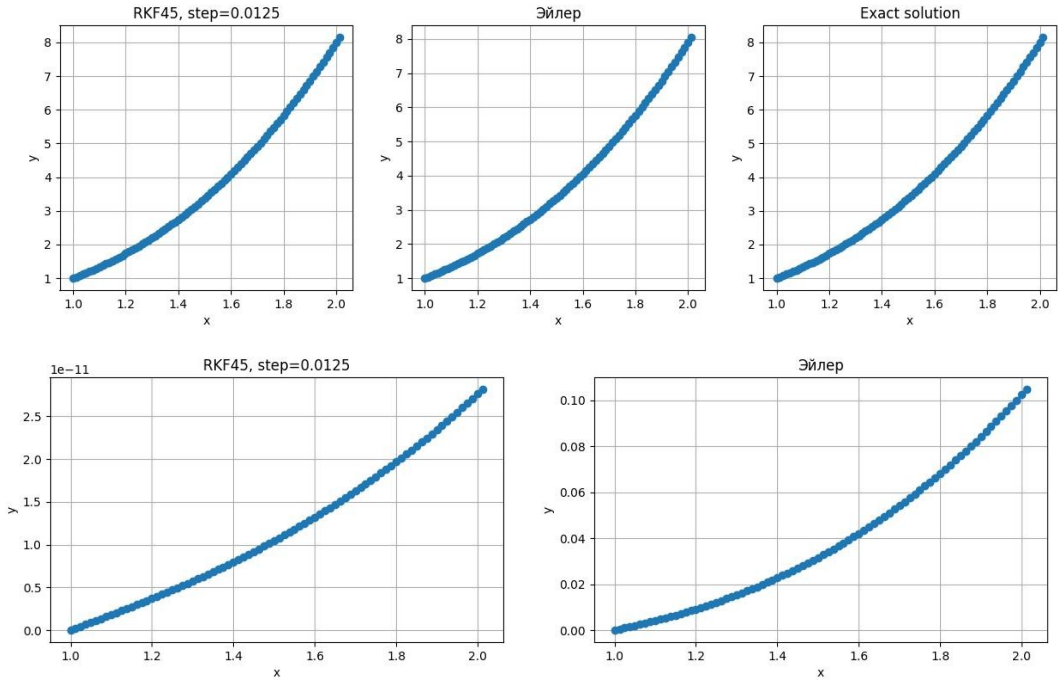
Далее не будем приводить таблицу полностью. Ограничимся лишь графики (вывод таблиц в разделе ссылки)
Для h_int = 0.05



Для h_int = 0.025



Для h_int = 0.0125



Вывод:

Из табличных значений можно сделать выводы:

- Локальная погрешностей увеличивается к концу отрезка
- Глобальная погрешность уменьшается у RKF45
- RKF45 имеет сильно меньшую погрешность, чем метод ломанной Эйлера

Ссылки:

Листин кода на github: https://github.com/vitaya-para/lab3_2023/blob/main/main.py

Полные выводы консоли: https://github.com/vitaya-para/lab3_2023/blob/main/output.txt