

Оглавление

Постановка задачи	3
Необходимый теоретический материал	3
Доказательство эквивалентности задач (1) и (2)	3
Вывод формулы метода Симпсона	7
Формула метода Рунге-Кутты 4-го порядка	8
Построение тестового примера	8
Результаты расчетов по тестовым примерам	9
Результаты вычислительного эксперимента	12
Протабулировать функцию $\operatorname{erf}(x)$ на отрезке $[0, 2]$ с шагом $h = 0.1$, вычисляя интеграл (1) методом Симпсона с некоторой точностью ε	12
Протабулировать функцию $\operatorname{erf}(x)$ на отрезке $[0, 2]$ с шагом $h = 0.1$, решая задачу Коши (2) методом Рунге – Кутты 4-го порядка с той же точностью ε	14
Сравнить время выполнения пп. 2 и 3.	15
Построить график зависимости времени табулирования от точности ε в том и другом случае.	16
Анализ полученных результатов	17
Заключение	17
Литература	17
Приложение. Код с комментариями	18

Постановка задачи

Задача 5.5. Функция ошибок задается выражением

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (1)$$

которое не выражается в элементарных функциях.

Для вычисления ее значения в точке x можно также использовать эквивалентную (1) задачу Коши

$$\begin{cases} y' = \frac{2}{\sqrt{\pi}} e^{-x^2} \\ y(0) = 0 \end{cases} \quad (2)$$

1. Доказать эквивалентность задач (1) и (2).
2. Протабулировать функцию $\operatorname{erf}(x)$ на отрезке $[0, 2]$ с шагом $h = 0.1$, вычисляя интеграл (1) методом Симпсона с некоторой точностью ε .
3. Протабулировать функцию $\operatorname{erf}(x)$ на отрезке $[0, 2]$ с шагом $h = 0.1$, решая задачу Коши (2) методом Рунге – Кутты 4-го порядка с той же точностью ε .
4. Сравнить время выполнения пп. 2 и 3.
5. Построить график зависимости времени табулирования от точности ε в том и другом случае.
6. При выводе таблицы значений функции ошибок округлять полученные значения в соответствии с точностью.

Необходимый теоретический материал

Доказательство эквивалентности задач (1) и (2)

Чтобы доказать эквивалентность задач (1) и (2), нужно доказать, что любое решение задачи (1) является решением задачи (2), и наоборот.

Пункт №1. Доказательство (1) \rightarrow (2)

Дана: функция ошибок

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

Она представляет собой интеграл с переменным верхним пределом.

Приведем теорему о дифференцируемости интеграла с переменным верхним пределом (теорему Барроу):

Пусть функция f непрерывна в точке $x_0 \in [a, b]$. Тогда функция $F(x) = \int_a^x f(t) dt$ дифференцируема в точке x_0 и справедливо равенство $F'(x_0) = f(x_0)$.

Таким образом, необходимо доказать, что функция $f(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}$ непрерывна для любого $x_0 \in [0, +\infty)$

1. Рассмотрим $f_1(x) = \frac{2}{\sqrt{\pi}} = \text{const}$

Для доказательства непрерывности $f_1(x)$ на всей числовой прямой используем определение непрерывности в терминах приращений аргумента и функции

Функция является непрерывной в точке $x = x_0$, если справедливо равенство

$$\lim_{\Delta x \rightarrow 0} \Delta f = \lim_{\Delta x \rightarrow 0} (f(x_0 + \Delta x) - f(x_0)) = 0, \text{ где } \Delta x = x - x_0$$

В нашем случае: $\forall x_0 \in \mathbb{R}$:

$$\lim_{\Delta x \rightarrow 0} f_1 = \lim_{\Delta x \rightarrow 0} (f_1(x_0 + \Delta x) - f_1(x_0)) = \lim_{\Delta x \rightarrow 0} \left(\frac{2}{\sqrt{\pi}} - \frac{2}{\sqrt{\pi}} \right) = 0$$

Следовательно, $f_1(x)$ является непрерывной в каждой точке $x_0 \in \mathbb{R}$. (3)

В силу того, что функция называется непрерывной на множестве \mathbb{E} , если она непрерывна в каждой точке данного множества, то из (3) следует, что $f_1(x) = \frac{2}{\sqrt{\pi}}$ непрерывна на \mathbb{R} (тогда непрерывность на $[0, +\infty)$ выполняется автоматически, так как $[0, +\infty) \in \mathbb{R}$)

2. Аналогично, докажем непрерывность $f_2(x) = e^{-x^2}$ на всей числовой прямой

$\forall x_0 \in \mathbb{R}$ верно:

$$\begin{aligned} \lim_{\Delta x \rightarrow 0} f_2 &= \lim_{\Delta x \rightarrow 0} (f_2(x_0 + \Delta x) - f_2(x_0)) = \lim_{\Delta x \rightarrow 0} (e^{-(x_0 + \Delta x)^2} - e^{-x_0^2}) = \\ &= \lim_{\Delta x \rightarrow 0} (e^{-x_0^2 - 2x_0 \cdot \Delta x - \Delta x^2} - e^{-x_0^2}) = \lim_{\Delta x \rightarrow 0} (e^{-x_0^2} (e^{-2x_0 \cdot \Delta x - \Delta x^2} - 1)) = \\ &= \lim_{\Delta x \rightarrow 0} (e^{-x_0^2} (e^{-\Delta x(2x_0 + \Delta x)} - 1)) = 0 \end{aligned}$$

Таким образом, $f_2(x) = e^{-x^2}$ непрерывна на \mathbb{R} (тогда непрерывность на $[0, +\infty)$ выполняется автоматически, так как $[0, +\infty) \in \mathbb{R}$)

3. Рассмотрим $f(x) = f_1(x) \cdot f_2(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}$

Приведем теорему об арифметических свойствах непрерывных функций:

Предположим, что две функции $f(x)$ и $g(x)$ непрерывны в точке $x=a$. Тогда произведение этих функций $f(x)g(x)$ также непрерывно в точке $x=a$. (4)

Вследствие (4) $f(x) = f_1(x) \cdot f_2(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}$ непрерывна $\forall x_0 \in \mathbb{R}$

4. Имеем: f непрерывна в любой точке $x_0 \in [0, +\infty)$. Тогда в силу теоремы о дифференцируемости интеграла с переменным верхним пределом: $F'(x_0) = f(x_0)$
 $\forall x_0 \in [0, +\infty)$.

Другими словами: $y'(x) = \frac{2}{\sqrt{\pi}} e^{-x^2} \forall x_0 \in [0, +\infty)$

5. Начальное условие для задачи Коши получаем подстановкой $x=0$ в (1)

$$\text{erf}(0) = \frac{2}{\sqrt{\pi}} \int_0^0 e^{-t^2} dt = 0$$

Получен определенный интеграл, взятый на отрезке нулевой длины, он равен 0.

6. Из п. 4. и 5. Получаем задачу Коши (2)

$$\begin{cases} y'(x) = \frac{2}{\sqrt{\pi}} e^{-x^2} \\ y(0) = 0 \end{cases}$$

Таким образом любое решение задачи (1) является решением задачи (2)

Пункт №2.

Доказательство (2) \rightarrow (1)

Постановка задачи Коши:

Пусть дано ОДУ первого порядка. Требуется найти функцию $y(t)$, удовлетворяющую дифференциальному уравнению и начальному условию:

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \text{ при } t \geq t_0$$

Дана задача Коши:

$$\begin{cases} y'(x) = f(x, y) = \frac{2}{\sqrt{\pi}} e^{-x^2} \\ y(0) = y_0 = 0 \end{cases} \quad (2)$$

1. Докажем сначала, что решение данной задачи Коши существует и единственно.

Приведем теорему существования и единственности решения задачи Коши.

Пусть вектор-функция $f(x, y) \in \mathbb{C}_n(G)$ удовлетворяет на каждом компакте области G

условию Липшица

$$\exists L > 0: \forall (x, y_1), (x, y_2) \in G \quad |f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|$$

Тогда решение задачи Коши существует и единственно

В нашем случае:

$$f(x, y) \in \mathbb{C}_n(G) = \mathbb{C}_n(\mathbb{R} \times \mathbb{R})$$

$$|f(x, y_1) - f(x, y_2)| = \left| \frac{2}{\sqrt{\pi}} e^{-x^2} - \frac{2}{\sqrt{\pi}} e^{-x^2} \right| = 0$$

$$|y_1 - y_2| = |0 - 0| = 0$$

$$\exists L > 0: \forall (x, y_1), (x, y_2) \in G \quad |f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|$$

$$0 \leq L \cdot 0 - \text{это верно для любого } L > 0 \Rightarrow$$

Условие Липшица выполнено \Rightarrow решение задачи Коши (2) существует и единственно.

2. Найдем решение задачи Коши в виде:

$$y(x) = y_0 + \int_{x_0}^x f(\xi, y(\xi)) d\xi$$

Заменим переменную интегрирования ξ на t , подставим $y_0 = 0, x_0 = 0$

$$f(t, y(t)) = \frac{2}{\sqrt{\pi}} e^{-t^2}$$

Получим:

$$y(x) = \int_0^x \frac{2}{\sqrt{\pi}} e^{-t^2} dt = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

Обозначим $y(x) = \operatorname{erf}(x)$

Тогда $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$, что совпадает с (1)

Таким образом, мы доказали, что каждое решение (2) является решением (1)

Из пунктов №1 и № 2 следует, что задачи (1) и (2) эквивалентны, что и требовалось доказать ■

Вывод формулы метода Симпсона

Выведем формулу Симпсона

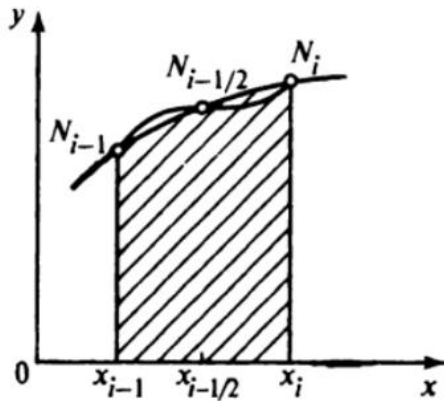


Рисунок 1

Рассмотрим рисунок 1. Площадь элементарной криволинейной трапеции заменим площадью фигуры, находящейся под параболой, проходящей через точки N_{i-1} , $N_{i-\frac{1}{2}}$ и N_i

Получим приближенное равенство $I_i \approx \int_{x_{i-1}}^{x_i} P_2(x) dx$, где $P_2(x)$ – интерполяционный многочлен второй степени с узлами x_{i-1} , $x_{i-1/2}$ и x_i .

$$P_2(x) = f_{i-1/2} + \frac{f_i - f_{i-1}}{h} (x - x_{i-1/2}) + \frac{f_i - 2f_{i-1/2} + f_{i-1}}{h^2/2} (x - x_{i-1/2})^2$$

$$h = x_i - x_{i-1}, \quad x_i - x_{i-1/2} = x_{i-1/2} - x_{i-1} = \frac{h}{2}$$

Ее интегрирование приводит к равенству

$$\begin{aligned} \int_{x_{i-1}}^{x_i} P_2(x) dx &= hf_{i-1/2} + \frac{f_i - f_{i-1}}{h} \int_{x_{i-1}}^{x_i} (x - x_{i-1/2}) dx + \\ &+ \frac{f_i - 2f_{i-1/2} + f_{i-1}}{h^2/2} \int_{x_{i-1}}^{x_i} (x - x_{i-1/2})^2 dx = \end{aligned}$$

$$= hf_{i-1/2} + \frac{h}{6} (f_i - 2f_{i-1/2} + f_{i-1}) = \frac{h}{6} (f_{i-1} + 4f_{i-1/2} + f_i)$$

Мы получили элементарную квадратурную формулу Симпсона

$$I_i \approx \frac{h}{6} (f_{i-1} + 4f_{i-1/2} + f_i)$$

Применим эту формулу на каждом элементарном отрезке

$$\begin{aligned} I &\approx \frac{h}{6} (f_0 + 4f_{1/2} + 2f_1 + 4f_{3/2} + 2f_2 + \dots + 2f_{n-1} + 4f_{n-1/2} + f_n) = \\ &= \frac{h}{6} \left(f_0 + f_n + 4 \sum_{i=1}^n f_{i-1/2} + 2 \sum_{i=1}^{n-1} f_i \right) \end{aligned}$$

$$\text{Остаточный член } R = \frac{M_4(b-a)}{2880} h^4$$

Формула метода Рунге-Кутты 4-го порядка

$$y_{n+1} = y_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(x_n, y_n)$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(x_n + h, y_n + hk_3)$$

Построение тестового примера

1) Возьмем подынтегральную функцию e^t . Рассмотрим интеграл $I(x) = \int_0^x e^t dt$. Его решение в точке фиксированной точке $x \in [0, 2]$ имеет вид

$$I(x) = \int_0^x e^t dt = e^x - e^0 = e^x - 1$$

2) Пусть $y = e^x + 1$.

$$y' = e^x$$

$$y(0) = 2$$

Тестовый пример:

$$\begin{cases} y'(x) = e^x \\ y(0) = y_0 = 2 \end{cases}$$

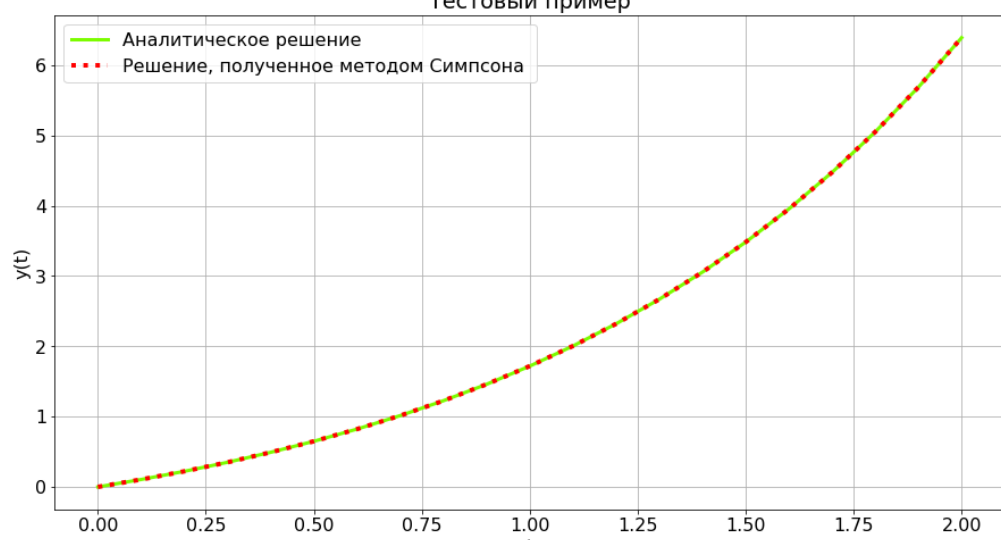
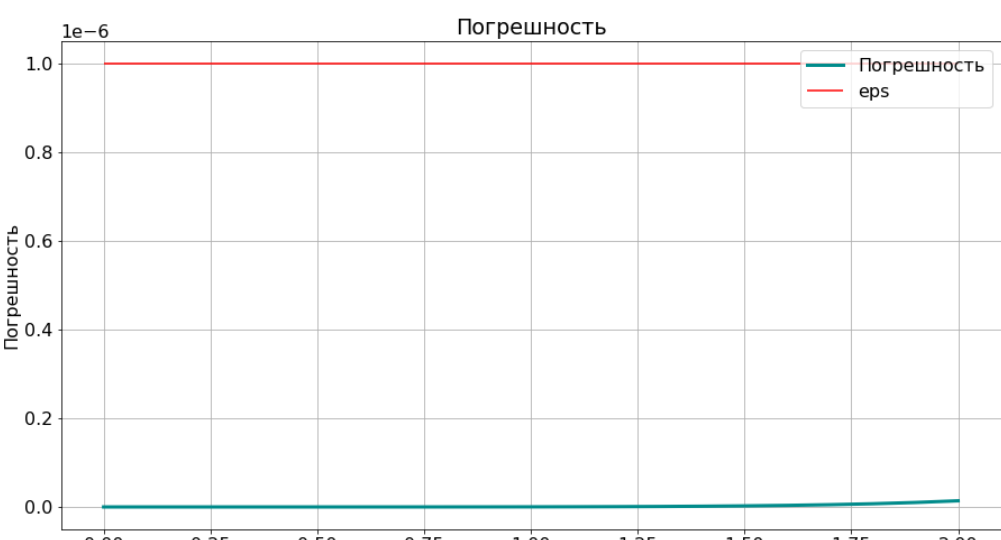
Проверка:

$$y = e^x + 1$$

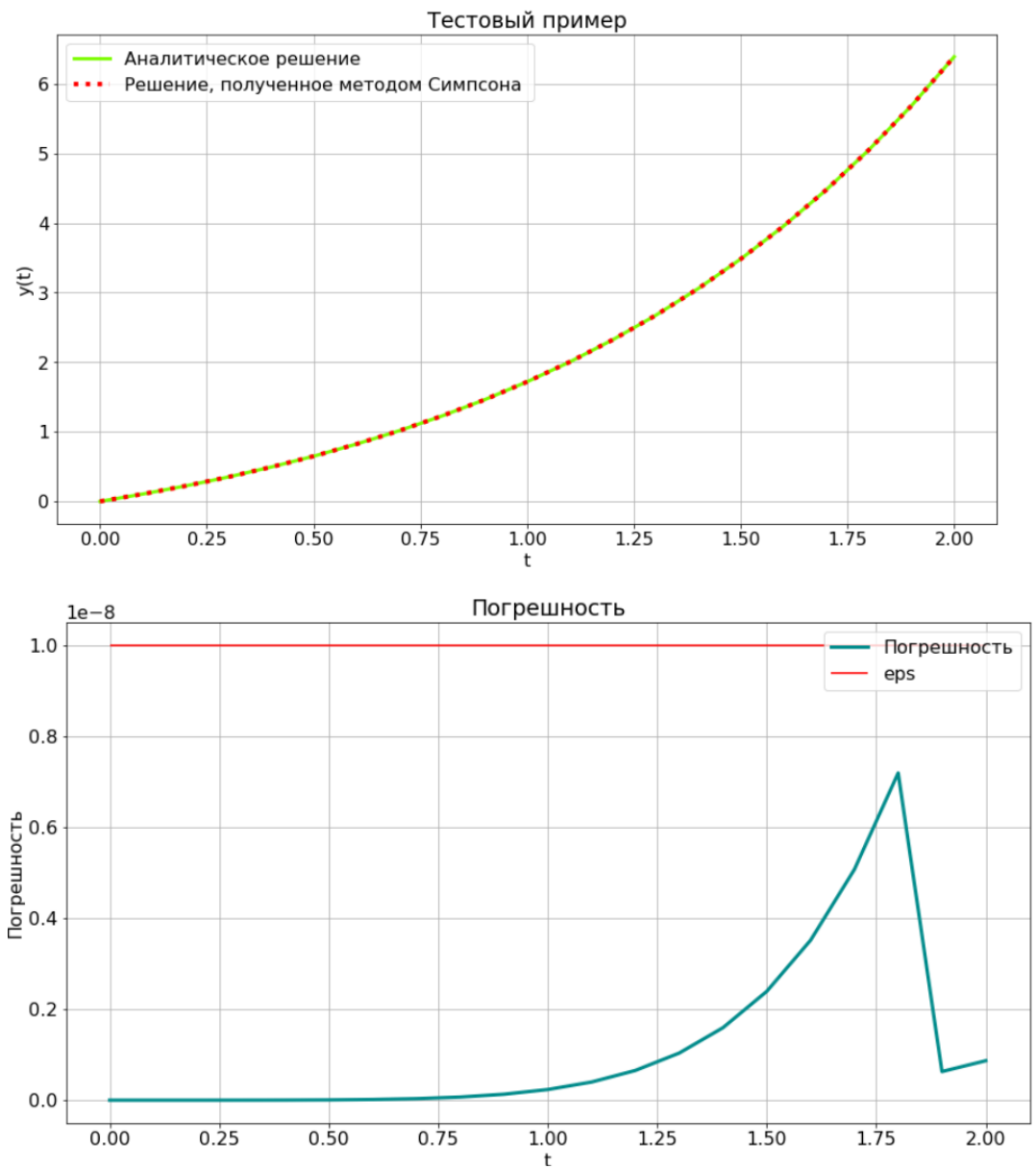
- Определена на $[0, 2]$
- При подстановке функции в уравнение $y'(x) = e^x$, оно обращается в тождество
- Удовлетворяет начальному условию

Результаты расчетов по тестовым примерам

Для метода Симпсона

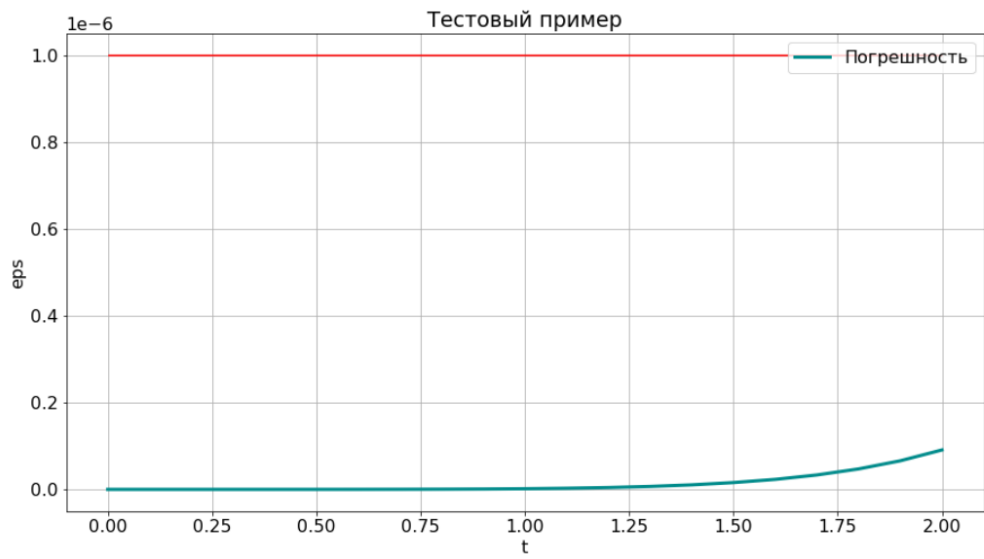
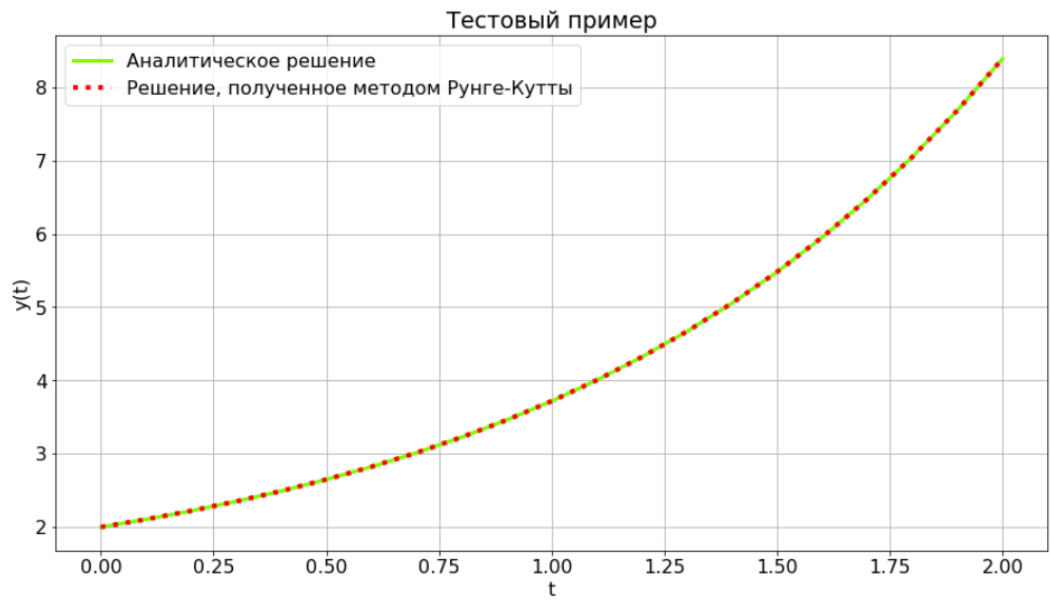
eps	Результат
	метод Симпсона
1e-06	<div><div>Тестовый пример</div><div>Погрешность</div></div>

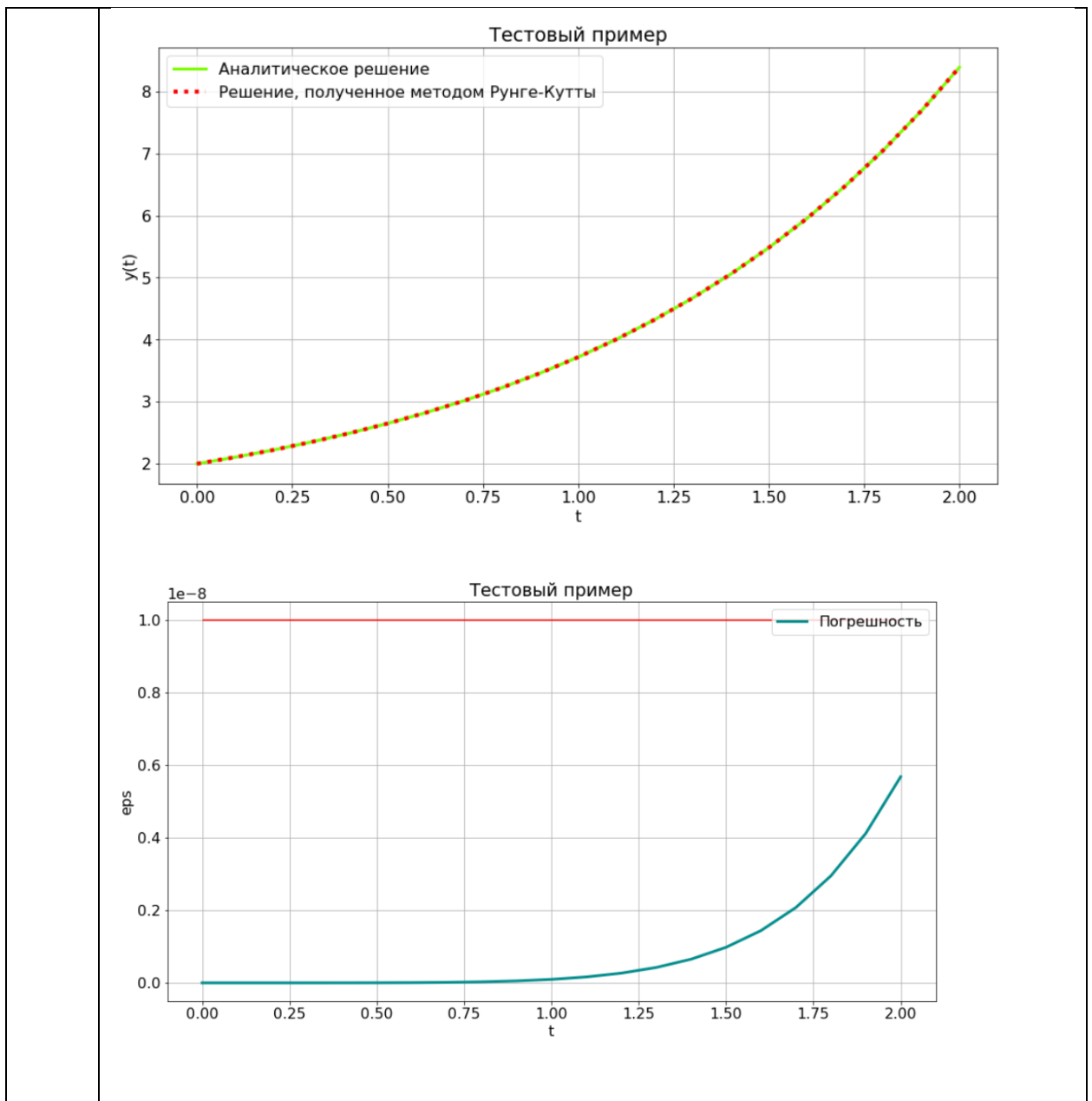
1e-08



eps
1e-06

метод Рунге-Кутты 4-го порядка





Заданная точность соблюдается, следовательно, методы построены верно, можно использовать их в дальнейших вычислениях.

Результаты вычислительного эксперимента

Протабулировать функцию $\text{erf}(x)$ на отрезке $[0, 2]$ с шагом $h = 0.1$, вычисляя интеграл (1) методом Симпсона с некоторой точностью ϵ .

При выводе таблицы значений функции ошибок округлять полученные значения в соответствии с точностью

№	Данные	Результат
1	$\epsilon = 1e-6$	Точность ϵ : $1e-06$ Отрезок: $[0; 2]$ Время выполнения табуляции 0.01100063 секунд Количество отрезков табуляции: 20

		<pre> x erf(x) 0.0 0.000000 0.1 0.112463 0.2 0.222703 0.3 0.328627 0.4 0.428392 0.5 0.520500 0.6 0.603856 0.7 0.677801 0.8 0.742101 0.9 0.796908 1.0 0.842701 1.1 0.880205 1.2 0.910314 1.3 0.934008 1.4 0.952285 1.5 0.966105 1.6 0.976348 1.7 0.983790 1.8 0.989091 1.9 0.992790 2.0 0.995322 </pre>
2	eps =1e-8	<p>Точность eps: 1e-08</p> <p>Отрезок: [0 ; 2]</p> <p>Время выполнения табуляции 0.00999427 секунд</p> <p>Количество отрезков табуляции: 20</p>

		<pre> x erf(x) 0.0 0.00000000 0.1 0.11246292 0.2 0.22270259 0.3 0.32862676 0.4 0.42839236 0.5 0.52049988 0.6 0.60385609 0.7 0.67780119 0.8 0.74210096 0.9 0.79690821 1.0 0.84270079 1.1 0.88020507 1.2 0.91031398 1.3 0.93400794 1.4 0.95228512 1.5 0.96610515 1.6 0.97634838 1.7 0.98379046 1.8 0.98909050 1.9 0.99279043 2.0 0.99532226 </pre>
--	--	--

Протабулировать функцию $\text{erf}(x)$ на отрезке $[0, 2]$ с шагом $h = 0.1$, решая задачу Коши (2) методом Рунге – Кутты 4-го порядка с той же точностью ε .

При выводе таблицы значений функции ошибок округлять полученные значения в соответствии с точностью.

№	Данные	Результат
1	$\text{eps} = 1\text{e-}6$	<p>Точность eps: $1\text{e-}06$ Отрезок: $[0 ; 2]$ Время выполнения табуляции 0.10808873176574707 секунд Количество отрезков табуляции: 20</p> <p>Значение $\text{erf}(x)$, посчитанное методом Рунге-Кутты 4-го порядка</p> <pre> x erf(x) 0.0 0.000000 0.1 0.112463 0.2 0.222703 0.3 0.328627 0.4 0.428392 0.5 0.520500 0.6 0.603856 0.7 0.677801 0.8 0.742101 0.9 0.796908 1.0 0.842701 1.1 0.880205 1.2 0.910314 </pre>

		1.3 0.934008 1.4 0.952285 1.5 0.966105 1.6 0.976348 1.7 0.983790 1.8 0.989090 1.9 0.992790 2.0 0.995322
2	eps =1e-8	Точность eps: 1e-08 Отрезок: [0 ; 2] Время выполнения табуляции 0.14436006546020508 секунд Количество отрезков табуляции: 20 Значение erf(x), посчитанное методом Рунге-Кутты 4-го порядка x erf(x) 0.0 0.00000000 0.1 0.11246292 0.2 0.22270259 0.3 0.32862676 0.4 0.42839236 0.5 0.52049988 0.6 0.60385609 0.7 0.67780119 0.8 0.74210096 0.9 0.79690821 1.0 0.84270079 1.1 0.88020507 1.2 0.91031398 1.3 0.93400794 1.4 0.95228512 1.5 0.96610515 1.6 0.97634838 1.7 0.98379046 1.8 0.98909050 1.9 0.99279043 2.0 0.99532226

Сравнить время выполнения пп. 2 и 3.

Точность eps: 1e-08

Время выполнение табулирования с помощью метода Симпсона: 0.01999021 секунд

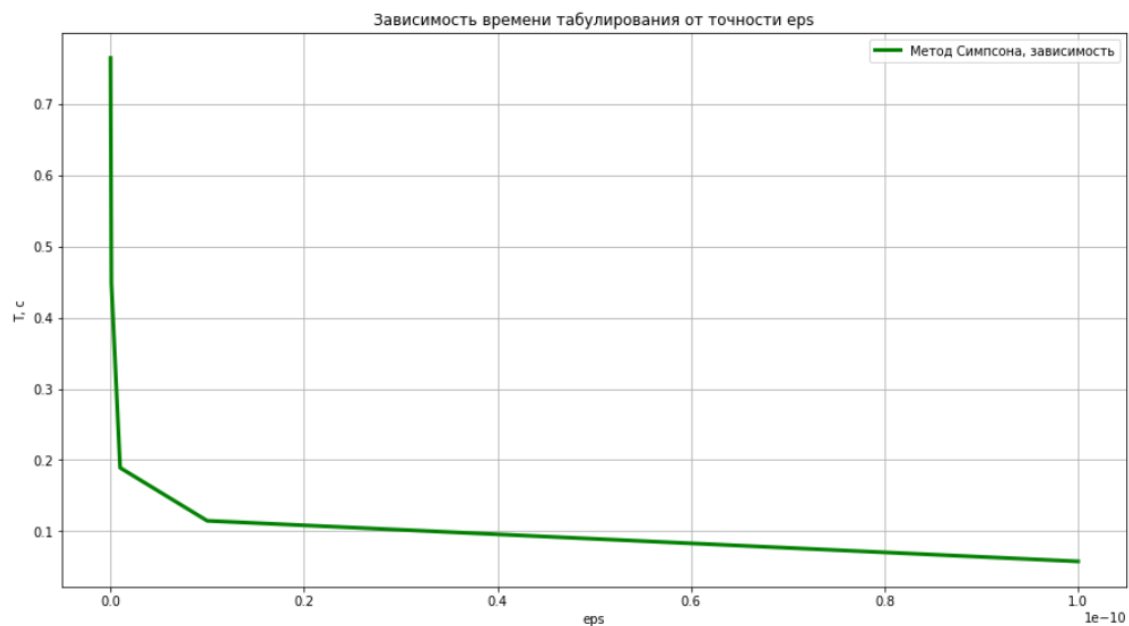
Время выполнение табулирования с помощью метода Рунге-Кутты 4-го порядка: 0.14436007 секунд

Разность времени выполнения м.Симпсона и м.Рунге-Кутты 4-го порядка: 0.12436986 секунд

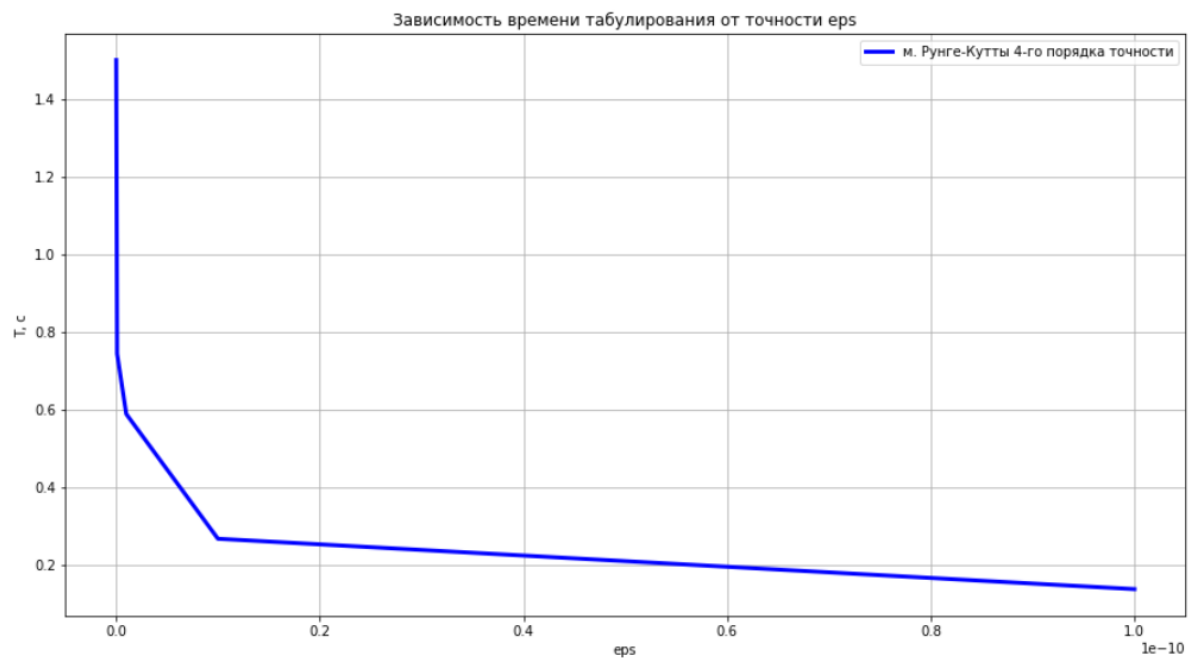
Таким образом, метод Рунге-Кутты 4-го порядка точности выполняется дольше.

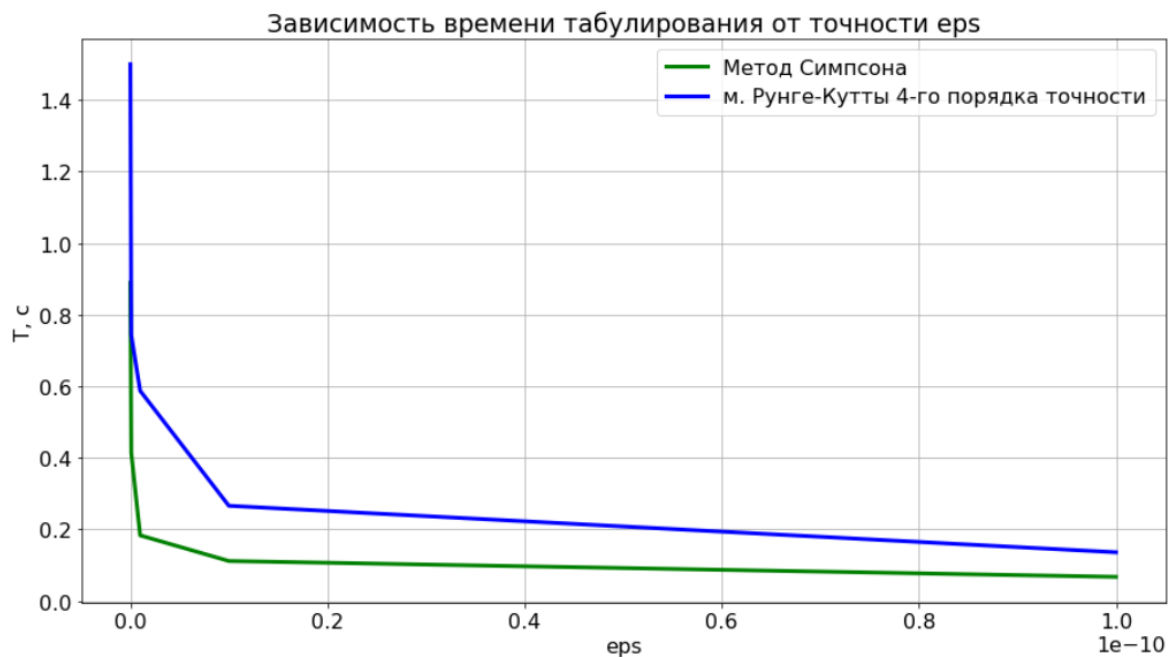
Построить график зависимости времени табулирования от точности ε в том и другом случае.

Метод Симпсона



Метод Рунге-Кутты 4-го порядка





Анализ полученных результатов

Полученные графики соответствуют реальности. Действительно, чем меньше eps, тем точность выше и ее достижение требует большего времени по сравнению с меньшей точностью, поэтому графики обеих зависимостей времени от eps убывают при увеличении eps. Также из таблиц мы видим, что при одинаковых eps, значения функции ошибок полученные табулированием на отрезке $[0, 2]$ с шагом $h = 0.1$ при вычислении интеграла (1) методом Симпсона и при решении задачи Коши методом Рунге-Кутты 4-го порядка одинаковые.

Заключение

В работе рассмотрены методы Симпсона и Рунге-Кутты четвертого порядка.

Была протабулирована функция ошибок на отрезке $[0, 2]$.

Исследована зависимость времени выполнения алгоритмов от точности eps.

Литература

[1]. Амосов А.А., Дубинский Ю.А., Копченкова Н.В. Вычислительные методы. М: Издательский дом МЭИ, 2008.

[2]. Интеграл с переменным верхним пределом [электронный ресурс]-

https://neerc.ifmo.ru/wiki/index.php?title=Интеграл_с_переменным_верхним_пределом

Приложение. Код с комментариями

```
eps=1e-8
erf_format = 8
a = 0
b = 2
h = 0.1
y0 = 0
x0 = a
n_segment = int((b-a)/h)
p = 4
def f_S(t):
    return (2/np.sqrt(np.pi)*np.exp(-t**2))
def f_test(t):
    return np.exp(t)
def real_solution(x):
    return np.exp(x) - 1

#метод Симпсона для частичного отрезка
def Simpson_i(a,b,h,f_S):
    S1 = 0
    S2 = 0
    n = round((b - a)/ h) + 1
    x_temp = np.arange(a , b+h/2 , h)
    x_temp1 = np.arange(a+h/2 , b , h)
    S1 = 2 * sum(f_S(x_temp[i]) for i in range(1 , n - 1 ))
    S2 = 4 * sum(f_S(x_temp1[i-1]) for i in range(1 , n))
    S = (h/6)*(f_S(a) + S1 + S2 + f_S(b))
    return S

#метод Симпсона с учетом eps
def Simpson_eps_i(a, b, n, f_S, eps, p):
    f1 = True
    while(f1):
        f1 = False
        h = (b-a)/n
        _2n = 2 * n
        h_2n = (b-a)/_2n
        S_h = Simpson_i(a, b, h, f_S)
        S_2h = Simpson_i(a, b, h_2n, f_S)
        err = abs(S_2h - S_h)/(2**p - 1)
        if (err>eps):
            f1 = True
            _2n *= 2
            n *= 2

    return S_2h

#табулирование с помощью метода Симпсона
def tab_Simpson(a, b, h, f_S, y0,eps):
    n=round((b-a)/h)
    #n= 4
    y_tab = [y0]
    x_tab = [a]
    for i in range(1, n_segment+1):
        x_tab.append(a+i*h)
        y_tab.append(Simpson_eps_i(a, a+i*h, n, f_S, eps, p))
    return n, x_tab, y_tab
```

```

#тестовый пример
# тест Симпсон
import matplotlib
from matplotlib import pyplot as plt

h = 0.1
n_segment=round((b-a)/h)
print ("Тестовый пример для метода Симпсона: ")
print ("Точность eps: ", eps)
print("Кол-во отрезков табуляции: ", n_segment)
x_tab_test_S = [a]
for i in range(1, n_segment+1):
    x_tab_test_S.append(a+i*h)
x_tab_test_S = np.array(x_tab_test_S)

y0_test = 0
n, x_tab, y_tab = tab_Simpson(a, b, h, f_test, y0,eps)

fig, ax = plt.subplots(figsize =(15,8))
plt.title('Тестовый пример')
ax.plot(x_tab_test_S, real_solution(x_tab_test_S), label = 'Аналитическое решение',
color = 'lawngreen',linewidth =3)
ax.plot(x_tab, y_tab, label = 'Решение, полученное методом Симпсона ', color =
'r',linewidth =4,linestyle = 'dotted')
ax.grid()
ax.set_xlabel('t')
ax.set_ylabel('y(t)')
ax.legend()

fig, ax = plt.subplots(figsize =(15,8))
plt.title('Погрешность')
ax.hlines(eps, 0, 2, color = 'r', label = 'eps')
ax.plot(x_tab_test_S, abs(real_solution(x_tab) - y_tab), label = 'Погрешность', color
= 'darkcyan',linewidth =3)
ax.grid()
ax.set_xlabel('t')
ax.set_ylabel('Погрешность')
ax.legend()

start_time = time.time()
n_Simpson, x_tab_Simpson, y_tab_Simpson = tab_Simpson(a, b, h, f_S, y0,eps)
t_Simpson = (time.time() - start_time)

print ("Точность eps: ", eps)
print("Отрезок: [", a, ";",b,"]" )
print("Время выполнения табуляции {:.8f} секунд " .format(t_Simpson))
print("Количество отрезков табуляции:", n_segment, "\n")
print("Значение erf(x), посчитанное методом Симпсона \n")
print(' {:.6s}      {:.6s}'.format("x", "erf(x)"))

for i in range(n_segment+1):
    print('{:.1f}   {:.{}}f'.format(x_tab_Simpson[i], y_tab_Simpson[i],
erf_format))

print('\n\n')

def outputTable_Simpson(t_RK,time_list, eps_list, n_segmet, n_format,x_tab, y_tab):
    print("-----")
    print("Время выполнения табуляции {:.8f} секунд " .format(t_RK) )
    time_list.append(t_RK)

```

```

        print ("Точность eps: ", "{:.0e}".format(eps_list[i]) )
        print("Значение erf(x), посчитанное методом Симпсона \n")
        for j in range(n_segment+1):
            print('{:.1f}   {:.{}}f'.format(x_tab[j], y_tab[j], n_format))

#метод Симпсона, рассматриваем eps: [1.e-10 1.e-11 1.e-12 1.e-13 1.e-14]
h_tab= 0.1
eps_list = [1e-10]
time_list = []
n_segmet = 20

k = 5
for i in range (1, k):
    eps_list.append(eps_list[i-1]/10)
x_Simpson = np.array(eps_list)

for i in range (0, k):
    start_time = time.time()
    n_Simpson, x_tab_Simpson, y_tab_Simpson = tab_Simpson(a, b, h, f_S,
y0,eps_list[i])
    t_RK = time.time() - start_time
    n_format = i + 10
    outputTable_Simpson(t_RK,time_list, eps_list, n_segmet, n_format,x_tab, y_tab)

print('\n')
print ("eps: ", x_Simpson)
y_Simpson = np.array(time_list)
print ("time: ", y_Simpson)
print("-----")
print(' {:.6s}      {:.6s}'.format("eps", "t(eps)"))
for i in range(k):
    print('{:.0e}   {:.{}}f'.format(x_Simpson[i], y_Simpson[i], 8))
print('\n\n')

fig, ax = plt.subplots(figsize =(15,8))
plt.title('Зависимость времени табулирования от точности eps')
ax.plot(x_Simpson, y_Simpson, label = 'Метод Симпсона, зависимость', color =
'g',linewidth =3)
ax.grid()
ax.set_xlabel('eps')
ax.set_ylabel('T, c')
ax.legend()

def f(x,y):
    return (2/np.sqrt(np.pi)*np.exp(-x**2))
def f_test(x, y):
    return np.exp(x)

def RK_y_next(y_prev, x_prev, h_i, f):
    k1 = f(x_prev, y_prev)
    k2 = f(x_prev + h_i/2, y_prev + h_i/2 * k1)
    k3 = f(x_prev + h_i/2, y_prev + h_i/2 * k2)
    k4 = f(x_prev + h_i, y_prev + h_i * k3)
    y_i= y_prev + h_i/6*(k1 + 2*k2 + 2*k3 + k4)
    return y_i

def RK_i(a,y0, x, n_1h,f , eps):

```

```

n_2h = round(n_1h/2)
n_1h = n_2h * 2
f1 = True
while (f1):
    f1 = False
    _2h = (x-a)/(n_2h)
    h = ((x-a)/(n_1h))
    y_1h = np.zeros(n_1h+1)
    y_1h[0] = y0
    y_2h = np.zeros(n_2h+1)
    y_2h[0] = y0
    x0 = a
    for i in range(1, n_2h + 1):
        y_2h[i] = RK_y_next(y_2h[i-1], x0+2*(i-1)*h, _2h, f)

        y_1h[2*i-1] = RK_y_next(y_1h[2*i-2], x0+(2*i-2)*h, h, f)
        y_1h[2*i] = RK_y_next(y_1h[2*i-1], x0+(2*i-1)*h, h, f)

    err = abs(y_2h[n_2h] - y_1h[2*n_2h])/15
    if (err >= eps):
        f1 = True
        n_2h = n_2h*2 #увеличиваем кол-во отрезков разбиения
        n_1h = n_1h*2
    return y_1h, y_2h, h, y_2h[n_2h]

a=0
x=0.6
n_1h=100
y0=0
y1h, y2h, h, result=RK_i(a,y0, x, n_1h, f, eps)
print("=====")
print("eps: ", eps)
print("Отрезок: [", a, ';', x, "]")
print("Решение: ", '{:.{}f}'.format(result, erf_format) )

def tab(a, b, h_tab, f, y0, eps):
    n = round((b-a)/h_tab)
    y_tab = [y0]
    x_tab = [a]
    for i in range(1, n_segment+1):
        x_tab.append(a+i*h_tab)
        y_tab.append(RK_i(a,y0, a+i*h_tab, n_1h, f, eps)[3] )
    return n, x_tab, y_tab

def outputTable_RK(t_RK, time_list, eps_list, n_segmet, n_format, x_tab, y_tab):
    print("-----")
    print("Время выполнения табуляции {:.8f} секунд ".format(t_RK) )
    time_list.append(t_RK)
    print("Точность eps: ", "{:.0e}".format(eps_list[i]) )
    print("Значение erf(x), посчитанное методом Рунге-Кутты 4-го порядка \n")
    for j in range(n_segment+1):
        print('{:.1f}    {:.{}f}'.format(x_tab[j], y_tab[j], n_format))

h_tab = 0.1
start_time = time.time()
n, x_tab, y_tab = tab(a, b, h_tab, f, y0, eps)
print("Точность eps: ", eps)
print("Отрезок: [", a, ';', b, "]")
t_RK = time.time() - start_time
print("Время выполнения табуляции %s секунд " % t_RK )

```

```

print("Количество отрезков табуляции:", n_segment, "\n")
print("Значение erf(x), посчитанное методом Рунге-Кутты 4-го порядка \n")
print(' {:.6s}      {:.6s}'.format("x", "erf(x)"))

for i in range(n_segment+1):
    print(' {:.1f}      {:.{}}f'.format(x_tab[i], y_tab[i], erf_format))

print('\n\n')

h_tab= 0.1
start_time = time.time()
n, x_tab, y_tab = tab(a, b, h_tab, f, y0,eps)
print ("Точность eps: ", eps)
print("Отрезок: [", a, ';',b,"]" )
t_RK = time.time() - start_time
print("Время выполнения табуляции %s секунд " %t_RK )
print("Количество отрезков табуляции:", n_segment, "\n")
print("Значение erf(x), посчитанное методом Рунге-Кутты 4-го порядка \n")
print(' {:.6s}      {:.6s}'.format("x", "erf(x)"))

for i in range(n_segment+1):
    print(' {:.1f}      {:.{}}f'.format(x_tab[i], y_tab[i], erf_format))

print('\n\n')

# тест PK - 4
import matplotlib
from matplotlib import pyplot as plt

def f_test(x, y):
    return np.exp(x)
def real_solution(x):
    return np.exp(x) + 1

h = 0.1
n_segment=round((b-a)/h)
print ("Тестовый пример для метода Рунге-Кутты 4-го порядка: ")
print ("Точность eps: ", eps)
print("Кол-во отрезков табуляции: ", n_segment)
x_tab_test_RK = [a]
for i in range(1, n_segment+1):
    x_tab_test_RK.append(a+i*h)
x_tab_test_RK = np.array(x_tab_test_RK)

y0_test = 2
n, x_tab, y_tab = tab(a, b, h_tab,f_test , y0_test, eps)

fig, ax = plt.subplots(figsize =(15,8))
plt.title('Тестовый пример')
ax.plot(x_tab_test_RK, real_solution(x_tab_test_RK), label = 'Аналитическое решение',
color = 'lawngreen',linewidth =3)
ax.plot(x_tab, y_tab, label = 'Решение, полученное методом Рунге-Кутты', color =
'r',linewidth =4,linestyle = 'dotted')
ax.grid()
ax.set_xlabel('t')
ax.set_ylabel('y(t)')
ax.legend()

```

```

fig, ax = plt.subplots(figsize =(15,8))
plt.title('Тестовый пример')
ax.hlines(eps, 0, 2, color = 'r')
ax.plot(x_tab, abs(real_solution(x_tab_test_RK) - y_tab), label = 'Погрешность',
color = 'darkcyan',linewidth =3)
ax.grid()
ax.set_xlabel('t')
ax.set_ylabel('eps')
ax.legend()

print ("Точность eps: ", eps)
print("Время выполнение табулирования с помощью метода Симпсона: {:.8f} секунд "
.format(t_Simpson) )
print("Время выполнение табулирования с помощью метода Рунге-Кутты 4-го порядка:{:.8f}
секунд " .format(t_RK))
print("Разность времени выполнения м.Симпсона и м.Рунге-Кутты 4-го порядка:{:.8f}
секунд " .format(abs(t_RK-t_Simpson)))

def outputTable_RK(t_RK,time_list, eps_list, n_segmet, n_format,x_tab, y_tab):
    print("-----")
    print("Время выполнения табуляции {:.8f} секунд " .format(t_RK) )
    time_list.append(t_RK)
    print ("Точность eps: ", "{:.0e}".format(eps_list[i]) )
    print("Значение erf(x), посчитанное методом Рунге-Кутты 4-го порядка \n")
    for j in range(n_segment+1):
        print('{:.1f}    {:.{}}f}'.format(x_tab[j], y_tab[j], n_format))

#Метод Рунге-Кутты, eps:  [1.e-10 1.e-11 1.e-12 1.e-13 1.e-14]
h_tab= 0.1
eps_list = [1e-10]
time_list = []
n_segmet = 20

k = 5
print('\n\n')
print ("eps: ", x_Simpson)
for i in range (1, k):
    eps_list.append(eps_list[i-1]/10)
x_RK = np.array(eps_list)

for i in range (0, k):
    start_time = time.time()
    n, x_tab, y_tab = tab(a, b, h_tab, f, y0,eps_list[i])
    t_RK = time.time() - start_time
    outputTable_RK(t_RK,time_list, eps_list, n_segmet, i+10,x_tab, y_tab)

print('\n')
print ("eps: ", x_RK)
y_RK= np.array(time_list)
print ("time: ", y_RK)
print("-----")
print(' {:.6s}      {:.6s}'.format("eps", "t(eps)"))
for i in range(k):
    print('{:.0e}    {:.{}}f'.format(x_RK[i], y_RK[i], 8))
print('\n\n')

```

```

import matplotlib
from matplotlib import pyplot as plt
fig, ax = plt.subplots(figsize =(15,8))
plt.title('Зависимость времени табулирования от точности eps')
ax.plot(x_RK, y_RK, label = 'м. Рунге-Кутты 4-го порядка точности', color =
'b',linewidth =3)
ax.grid()
ax.set_xlabel('eps')
ax.set_ylabel('T, с')
ax.legend()

fig, ax = plt.subplots(figsize =(15,8))
plt.title('Зависимость времени табулирования от точности eps')
ax.plot(x_Simpson, y_Simpson, label = 'Метод Симпсона', color = 'g',linewidth =3)
ax.plot(x_RK, y_RK, label = 'м. Рунге-Кутты 4-го порядка точности', color =
'b',linewidth =3)
ax.grid()
ax.set_xlabel('eps')
ax.set_ylabel('T, с')
ax.legend()

```