Задача 6.1. Дана формула численного дифференцирования. Требуется исследовать поведение погрешностей при численном дифференцировании.

$f^{'}(x) \approx \frac{af(x)+bf(x-7h)+cf(x-14h)+df(x-6h)}{h}$ (1)

1.Определить коэффициенты a, b, c, d так, чтобы формула имела максимальный порядок точности.

Функции $f(x), f(x - 7h), f(x - 14h), f(x - 6h)$ разложим по формуле Тейлора до 4-го порядка включительно

$$f^{'}(x) - \frac{af(x)+bf(x-7h)+cf(x-14h)+df(x-6h)}{h} = f^{'}(x) - \frac{a}{h}f(x) -$$

$$-\frac{b}{h}\left(f(x) - 7hf^{'}(x) + \frac{(7h)^2}{2!}f^{''}(x) - \frac{(7h)^3}{3!}f^{(3)}(x) + \frac{(7h)^4}{4!}f^{(4)}(x) + o\right) -$$

$$-\frac{c}{h}\left(f(x) - 14hf^{'}(x) + \frac{(14h)^2}{2!}f^{''}(x) - \frac{(14h)^3}{3!}f^{(3)}(x) + \frac{(14h)^4}{4!}f^{(4)}(x)\right) -$$

$$-\frac{d}{h}\left(f(x) - 6hf^{'}(x) + \frac{(6h)^2}{2!}f^{''}(x) - \frac{(6h)^3}{3!}f^{(3)}(x) + \frac{(6h)^4}{4!}f^{(4)}(x)\right)$$

Соберем степени при $h^{-1}, h^{0}, h^{1}, h^{2}, h^{3}$

$h^{-1}: \ -a - b - c - d = 0$

$h^{0}: 1 + 7b + 14c + 6d = 0$

$h^{1}: -\frac{7^2}{2!}b - \frac{14^2}{2!}c - \frac{6^2}{2!}d = 0$

$h^{2}: \frac{7^3}{3!}b + \frac{14^3}{3!}c + \frac{6^3}{3!}d = 0$

$h^{3}: \frac{7^4}{4!}b + \frac{14^4}{4!}c + \frac{6^4}{4!}d = 0$

Получим СЛАУ

$\{a + b + c + d = 0 \ 7b + 14c + 6d = -1 \ 7^2b + 14^2c + 6^2d = 0 \ 7^3b + 14^3c + 6^3d = 0 \ 7^4b +$

Решим СЛАУ методом Гаусса и получим

$a = \frac{8}{21}$

$b = \frac{12}{7}$

$c = -\frac{3}{56}$

$d = -\frac{49}{24}$

Подставим a, b, c, d в формулу (1)

$f^{'}(x) \approx \frac{\frac{8}{21}f(x)+\frac{12}{7}f(x-7h)-\frac{3}{56}f(x-14h)-\frac{49}{24}f(x-6h)}{h}$

Формула имеет третий порядок точности по h

2. Реализовать программно полученную формулу численного дифференцирования и формулу правой разностной производной.

Формула численного дифференцирования:

$$f'(x) \approx \frac{\frac{8}{21}f(x) + \frac{12}{7}f(x-7h) - \frac{3}{56}f(x-14h) - \frac{49}{24}f(x-6h)}{h}$$

Формула правой разностной производной:

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

```
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy import integrate
#6.1
def dF(f, x, h):
    return (8/21*f(x) + 12/7*f(x - 7*h) - 3/56*f(x - 14*h) - 49/24*f(x - 6 *h))/h
def RightDifferenceDerivative(f,x, h):
    return (f(x + h) - f(x))/h
```

3. В качестве тестовой функции для проверки корректности работы программы взять функцию из задачи 5.1. На отрезке [a, b] построить графики точной производной и полученные по формулам численного дифференцирования, выбрав шаг h0=0.0001 .

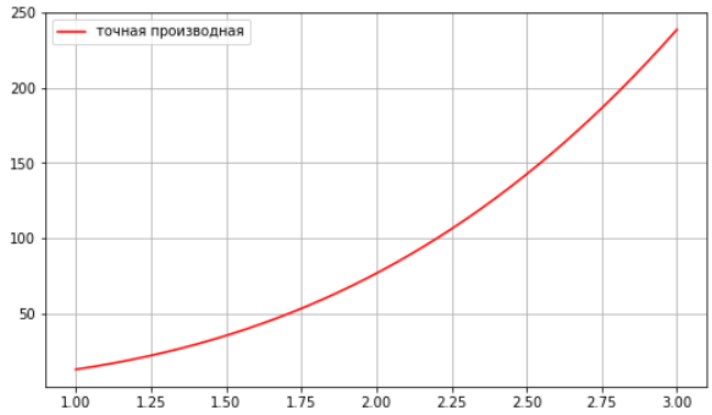
$$f(x) = 0.6 + 1.3x + 1.2x^3 + 1.9x^4$$

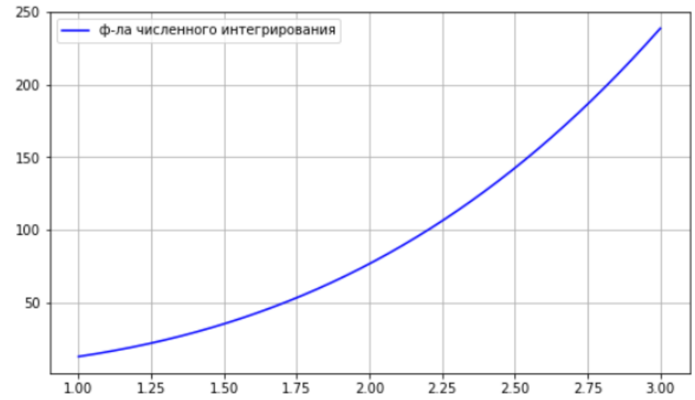
$$[a, b] = [1, 3]$$

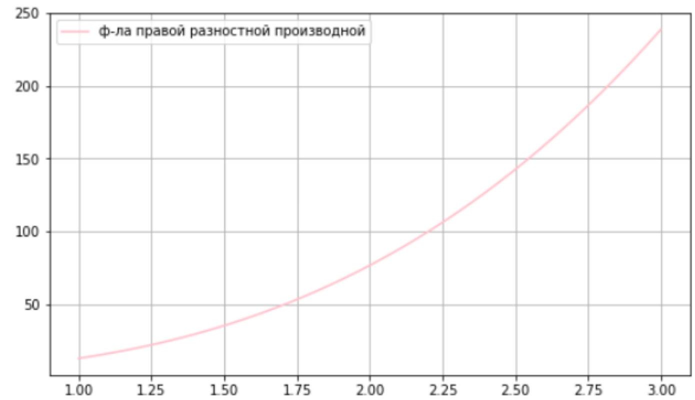
$$f'(x) = 1.3 + 3.6x^2 + 7.6x^3$$

```
def dF5_1(x):
    return 1.3+3.6*x**2+7.6*x**3
```

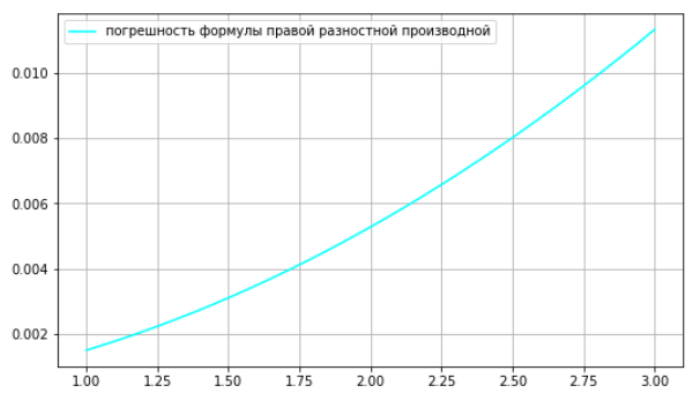
```
a = 1
b = 3
h0 = 0.0001
x = np.linspace(a, b, 1000)
```




ф-ла численного интегрирования
250
200
150
100
50
1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00


ф-ла правой разностной производной
250
200
150
100
50
1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00


1e−9
погрешность формулы численного дифференцирования
3.0
2.5
2.0
1.5
1.0
1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00


погрешность формулы правой разностной производной
0.010
0.008
0.006
0.004
0.002
1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00

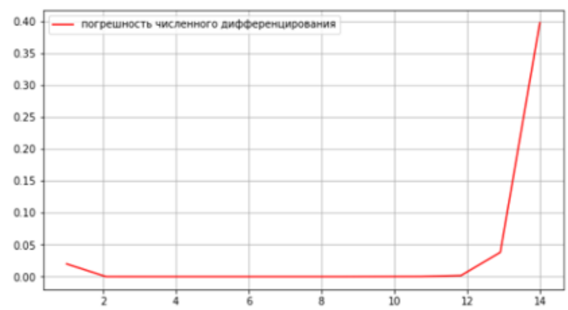
```
a = 1
b = 3
h0 = 0.0001
x = np.linspace(a, b, 1000)
#график точной производной
fig, axs = plt.subplots(1, 5, figsize = (50, 5)) #3
#plt.interactive(True)
axs[0].plot(x, dF5_1(x), label = "точная производная", color = "red")
axs[0].grid()
axs[0].legend()

axs[1].plot(x, dF(F5_1, x, h0), label = "ф-ла численного интегрирования", color = "blue")
axs[1].grid()
axs[1].legend()

axs[2].plot(x, RightDifferenceDerivative(F5_1,x, h0), label = "ф-ла правой разностной производной", color = "pink")
axs[2].grid()
axs[2].legend()
#график погрешности индивидуальной формулы численного дифференцирования
axs[3].plot(x, abs(dF5_1(x) - dF(F5_1, x, h0)), label = "погрешность формулы численного дифференцирования", color = "green",linew
axs[3].grid()
axs[3].legend()
#график погрешности формулы правой разностной производной
axs[4].plot(x,abs(dF5_1(x) - RightDifferenceDerivative( F5_1,x, h0)), label = "погрешность формулы правой разностной производной'
axs[4].grid()
axs[4].legend()
plt.show()
#axs.legend.show()
```

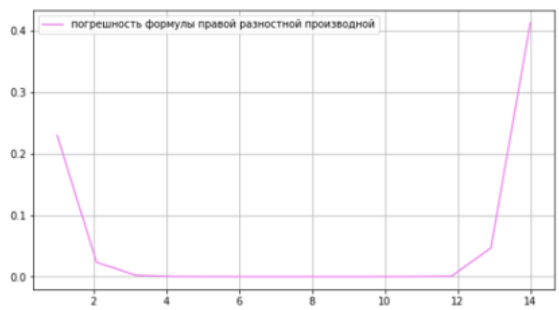
4. Взять функцию из задачи 5.2. Выбрать фиксированную точку на отрезке [a,b] и вычислить значения производных по формулам численного дифференцирования, уменьшая шаг дифференцирования h0=0.1 последовательно в 10 раз: $h_k = h_0 * 10^{-k}$, k=0,1,2,…Найти оптимальное значение шага дифференцирования для каждой формулы численного дифференцирования. По полученным данным построить графики погрешностей.

$f(x) = e^x \sin sin\,(\pi x)$, $[a,\ b] = [2,\ 6]$

$$f'(x) = e^x(\pi x)\ +\ \pi cos(\pi x))$$

i	шаг h	R численного дифференцирования	R правой разностной производной
00	1.000000000000000	3.710719657928883919	2.021448462004443059
01	0.100000000000000	0.019932895412466678	0.229441954661684377
02	0.010000000000000	0.000020185242558313	0.023186792223572894
03	0.001000000000000	0.000000020102589815	0.002321074661185918
04	0.000100000000000	0.000000000211997531	0.000232131292435156
05	0.000010000000000	0.000000000093216102	0.000023219847260947
06	0.000001000000000	0.000000059392910146	0.000002281870042253
07	0.000000100000000	0.000000962138113181	0.000000389547921031
08	0.000000010000000	0.000002037999127680	0.000000836907304347
09	0.000000001000000	0.000112850068124004	0.000007378815404024
10	0.000000000100000	0.000196447641791053	0.000123879277182937
11	0.000000000010000	0.001719954823887093	0.000780159502998146
12	0.000000000001000	0.038125617630793585	0.046719760707066627
13	0.000000000000100	0.397050536434090162	0.412676251122519488
14	0.000000000000010	5.868861053514738302	3.525003850291977159





Оптимальный шаг индивидуального метода численного дифференцирования: $h\ =\ 10^{-5}$.

Оптимальный шаг метода численного дифференцирования: $h = 10^{-7}$.

```
def f4(x):
    return (np.exp(x)*np.sin(math.pi*x))

def df4(x):
    return np.exp(x)*(np.sin(math.pi*x) + math.pi*np.cos(math.pi*x))

def optimization(func, f, df, x):
    h0 = 0.1
    y = np.zeros(15)
    R = np.zeros(15)
    for k in range(15):
        h = h0* 10**(-k)
        y[k] = func(f, x, h)
        R[k] = abs(y[k] - df(x))
    return y, R
x0=2

y1, R1 = optimization(dF, f4, df4, x0)
y2,R2 = optimization(RightDifferenceDerivative, f4, df4, x0)
```

```
print("  i   |         шаг h            |      R численного дифференцирования            |        R правой разностной производной
print("-----------------------------------------------------------------------------------------------------------------------'
for i in range(0, 15):
    print("%.2i    |   %.15lf     |  %.18lf                           | %.18lf    " % (i, 0.1**i, R1[i], R2[i]))

fig, axs = plt.subplots(1, 2, figsize = (20, 5)) #3
axs[0].plot(np.linspace(1, 14, 13), R1[1:14], label = "погрешность численного дифференцирования", color = "red")
axs[0].grid()
axs[0].legend()

axs[1].plot(np.linspace(1, 14, 13), R2[1:14], label = "погрешность формулы правой разностной производной", color = "violet")
axs[1].grid()
axs[1].legend()
plt.show()
```

Вывод: погрешность индивидуального метода численного дифференцирования при оптимальном шаге: $h = 10^{-5}$ равна примерно $9 \cdot 10^{-11}$, для формулы правой разностной производной и оптимальном для нее шаге $h = 10^{-7}$ равна $3 \cdot 10^{-7}$. Так и должно быть, потому что формула численного дифференцирования имеет третий порядок точности, а формула правой разностной производной – первый.

Задача 6.2. Найти приближенное решение задачи Коши для обыкновенного дифференциального уравнения (ОДУ) 1 порядка с точностью $\varepsilon = 10^{-6}$.

1. Найти аналитическое решение задачи

$$y' = -\,tg(0.5t)y$$

$$y_0 = 1$$

$$t \in \left[0, \frac{\pi}{2}\right]$$

$$y_0 = y(t_0)$$

Уравнение решается методом разделения переменных

$$\frac{dy}{dt} = -\frac{\sin\sin(0.5t)}{\cos\cos(0.5t)} * y$$

$$\frac{dy}{y} = -\frac{\sin\sin(0.5t)}{\cos\cos(0.5t)} * dt$$

$\int \frac{dy}{y} =- \int \frac{\sin sin\,(0.5t)}{\cos cos\,(0.5t)} * dt + C$

$\ln ln\,|y| = 2 \ln ln\,|cos(0.5t)| + C$

$e^{\ln ln\,|y|} = e^{ln|\cos cos\,(0.5t)\,|^2 + C}$

$y = \widetilde{C} cos(0.5t)^2$ – общее решение

Подставим начальное условие

$y(0) = \widetilde{C} \cdot 1 = 1 \;\; => \widetilde{C} = 1$

Решение задачи Коши

$y = cos(0.5t)^2$

2.Составить программу вычисления решения методом Эйлера с заданной точностью, используя правило Рунге. Найти решение задачи с точностью ε $= 10^{-6}$, число точек N и шаг, при котором точность достигается. Построить график решения.

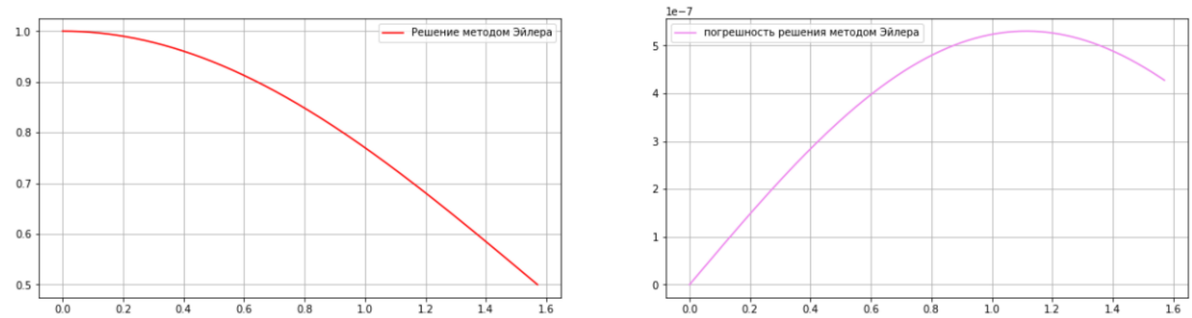
формула метода Эйлера:

$y_{i+1} = y_i + hf\left(t_i, y_i\right)$

```
#6.2
def f(t, y):
    return -np.sin(0.5 * t)/np.cos(0.5 * t)*y

def y(t):
    return np.cos(0.5 * t)**2 #аналитическое решение задачи Коши

eps = 10**(-6)
t0 = 0
T = np.pi/2
y0 = 1

def Euler(t0, T, y0, eps, f, p):
    flag = True
    n = 1
    h = (T-t0)/n
    y = np.zeros(n+1)
    y[0] = y0
    y[1] = y[0] + h*f(t0, y[0])
    while flag:
        flag = False
        y1 = y.copy()
        n = 2*n
        h = (T-t0)/n
        y = np.zeros(n+1)
        y[0] = y0
        for i in range(1, int(n/2) + 1):
            y[2*i-1] = y[2*i-2] + h*f(t0 + (2*i - 2)*h, y[2*i-2])
            y[2*i] = y[2*i-1] + h*f(t0 + (2*i - 1)*h, y[2*i-1])
            if abs(y[2*i] - y1[i])/(2**p - 1) >= eps:
                flag = True
    return y, n, h
y_arr, n, h = Euler(t0, T, y0, eps, f, 1)
print("метод Эйлера с использованием правила Рунге")
print("n = ", n)
print("h = ", h)
```

```
x = np.linspace(t0, T, n + 1)
fig, axs = plt.subplots(1, 2, figsize = (20, 5)) #3
axs[0].plot(x,  y_arr, label = "Решение методом Эйлера", color = "red")
#axs[0].plot(x,  y(x), label = "Аналитическое решение задачи Коши", color = "blue")
axs[0].grid()
axs[0].legend()

axs[1].plot(x,abs(y(x) - y_arr), label = "погрешность решения методом Эйлера", color = "violet")
axs[1].grid()
axs[1].legend()
plt.show()
```

```
метод Эйлера с использованием правила Рунге
n =  524288
h =  2.996056226339143e-06
```



3. Составить программу вычисления решения с заданной точностью методом индивидуального варианта. Найти решение задачи с заданной точностью, число точек N и шаг, при котором точность достигается. Построить график решения задачи.

метод Эйлера-Коши:

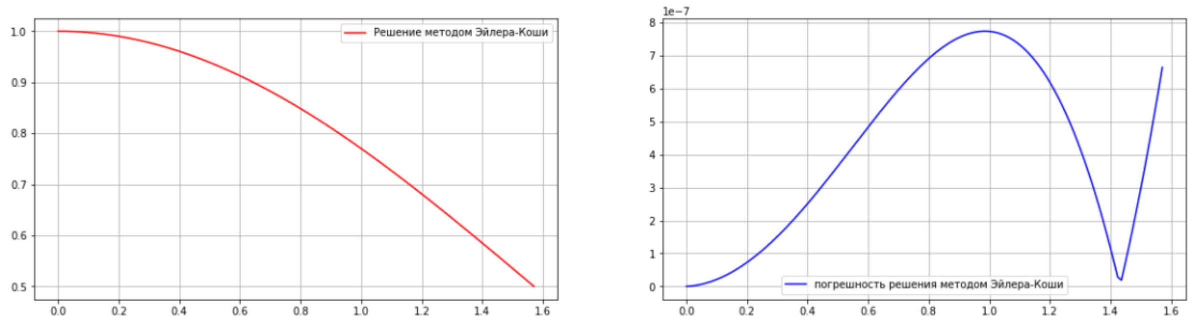
$$\overline{y}_{i+1} = y_i + hf(t_i, y_i)$$

$$y_{i+1} = y_i + \frac{h}{2}\left(f\left(t_i, y_i\right) + f\left(t_{i+1}, y_{i+1}\right)\right)$$

```python
def Euler_Cauchy(t0, T, y0, eps, f, p):
    flag = True
    n = 1
    h = (T-t0)/n
    y = np.zeros(n+1)
    y[0] = y0
    y_ = y[0] + h*(f(t0, y[0]))
    y[1] = y[0] + h/2*(f(t0, y[0]) + f(T, y_))
    while flag:
        flag = False
        y1 = y.copy()
        n = 2*n
        h = (T-t0)/n
        y = np.zeros(n+1)
        y[0] = y0
        for i in range(1, int(n/2) + 1):
            y_ = y[2*i-2] + h*f(t0 + (2*i - 2)*h, y[2*i-2])
            y[2*i-1] = y[2*i-2] + h/2*(f(t0 + (2*i - 2)*h, y[2*i-2]) +
                                       f(t0 + (2*i - 1)*h, y_))
            y_ = y[2*i-1] + h*f(t0 + (2*i - 1)*h, y[2*i-1])
            y[2*i] = y[2*i-1] + h/2*(f(t0 + (2*i - 1)*h, y[2*i-1]) +
                                     f(t0 + (2*i)*h, y_))
            if abs(y[2*i] - y1[i])/(2**p - 1) >= eps:
                flag = True
    return y, n, h

y_arr2, n2, h2= Euler_Cauchy(t0, T, y0, eps, f, 2)
print("метод Эйлера-Коши")
print("n = ", n2)
print ("h = ", h2)
```

```python
x = np.linspace(t0, T, n2+ 1)
fig, axs = plt.subplots(1, 2, figsize = (20, 5)) #3
axs[0].plot(x,  y_arr2, label = "Решение методом Эйлера-Коши", color = "red")
axs[0].grid()
axs[0].legend()

axs[1].plot(x,abs(y(x) - y_arr2), label = "погрешность решения методом Эйлера-Коши", color = "blue")
axs[1].grid()
axs[1].legend()
plt.show()
```

```
метод Эйлера-Коши
n =  128
h =  0.01227184630308513
```



<u>Вывод</u>: для достижения требуемой точности $\varepsilon = 10^{-6}$ в методе Эйлера нужно n = 524288 разбиений, а в методе Эйлера-Коши примерно в 4 тысячи раз меньше разбиений. Геометрическая интерпретация метода Эйлера: интегральная кривая y(x) на отрезке [a;b] приближается к ломаной, наклон которой определяется наклоном интегральной кривой уравнения в точке $[x_i; y_i]$. Метод Эйлера-Коши отличается от метода Эйлера тем, что значения y вычисляются не только в точках сетки (шаг h), но и в середине отрезков (шаг h/2), этот метод имеет второй порядок точности. В практических целях гораздо удобнее применять метод Эйлера-Коши так он имеет более высокий порядок точности по h (метод Эйлера имеет первый порядок точности).