Выполнил Галиуллин Арслан, 1 курс факультета физики, группа 171.
1233550v@mail.ru

Все программы написаны мной. Документ составлен в Latex'е. Времени потратил много. Желаемая оценка - 10.

1. Задача 1

a)

$s_2 = 0$
$e_2 = 1110$
$Mantissa_2 = (1.f)_2 = 1.1010\ 0000\ 0000\ 0000\ 0000\ 000$

b)

$e_{10} = 14$
$p_{10} = e_{10} - 127 = -113$

c)

$Mantissa_{10} = 1 + 2^{-1} \cdot 1 + 2^{-2} \cdot 0 + 2^{-3} \cdot 1 + 0 = 1.625$

d)

$A_{10} = (-1)^0 \cdot 1.f \cdot 2^p \approx 1.56 \cdot 10^{-34}$

2. Задача 2

a)

```
under = 1.0
over = 1.0
N = 2000
for i in range (N):
    under = under/2
    over = over*2
    print ("Loop: ", i, " ", under, " ", over)
```

Отсюда, соответственно, граница *underflow* и *overflow* для чисел с плавающей запятой с двойной точностью:
$under \approx 5 \cdot 10^{-324}$

$over \approx 9 \cdot 10^{307}$
В Python 3.0 нет чисел с плавающей запятой одинарной точности

b)

```
import math
under = -1
over = 1
N = 1000000000
for i in range (N):
    under = abs(under)*1000000
    over = over*1000000
    print ("Loop: ", i, " ", -math.log10(abs(under)), " ", math.log10(
        over))
```

Для целых чисел:
$under \to \infty$
$over \to -\infty$

3. Задача 3

a)

```
eps = 1.0
one_Plus_eps = 2.0
while (one_Plus_eps != 1):
    eps = eps /2
    one_Plus_eps = 1.0 + eps
    print ( " eps = " , eps , " , one + eps = " , one_Plus_eps )
```

Для типа *float* $\varepsilon \approx 1 \cdot 10^{-16}$

a)

```
eps = complex(1.0, 0.0)
one_Plus_eps = (2.0, 0.0)
while (one_Plus_eps != complex(1.0, 0.0)):
    eps = eps /2
    one_Plus_eps = 1.0 + eps
    print ( " eps = " , eps , " , one + eps = " , one_Plus_eps )
```

Для типа *complex* $\varepsilon \approx 1 \cdot 10^{-16}$

4. Задача 4

1)

```
import math
def my_sin(x):
    term = x
    sum = x
    eps = 10**(-8)
    n = 1
    while (abs(term) > abs(sum * eps)):
        n += 1
        term = -term * x**2 /( (2*n-1) * (2*n-2) )
        sum = sum + term
    return sum
while (True):
    try:
        x = float(input())
    except:
        print ("Exit")
        break
    if (math.sin(x) != 0):
        print (abs ((my_sin(x) - math.sin(x))/(math.sin(x))))
    else:
        print ("sin(x) = 0, my_sin(x) = ", my_sin(x))
```

2)

x	$\varepsilon_{\sin(x)}$
0 (0)	0.0/0.0
3.14159 (π)	$4 \cdot 10^{-11}$
3.141592654 (π)	$8 \cdot 10^{-7}$
6.28318 (2π)	$2 \cdot 10^{-10}$
6.283185307 (2π)	$6 \cdot 10^{-6}$
1.57 ($\pi/2$)	$6 \cdot 10^{-12}$
4.71 ($3\pi/2$)	$6 \cdot 10^{-12}$
31.4159 (10π)	$7 \cdot 10^{0}$
31.415926 (10π)	$1 \cdot 10^{3}$
314.159 (100π)	$6 \cdot 10^{121}$
314.1592654 ($3\pi/2$)	$2 \cdot 10^{126}$
32.99 ($10\pi + \pi/2$)	$4 \cdot 10^{-4}$
32.98672286 ($10\pi + \pi/2$)	$8 \cdot 10^{-4}$
315.73 ($100\pi + \pi/2$)	$5 \cdot 10^{118}$
315.7300617 ($100\pi + \pi/2$)	$2 \cdot 10^{119}$

4)

Действительно, при малых x алгоритм сходится к правильному ответу.

5)

В пропущенных пунктах я не понял задание. Но, думаю, это сделать не сложно (если знать, что надо в итоге).

13)

```
import math
def my_sin(x):
    while (abs(x) > math.pi):
        x -= 2*math.pi
    term = x
    sum = x
    eps = 10**(-8)
    n = 1
    while (abs(term) > abs(sum * eps)):
        n += 1
        term = -term * x**2 /( (2*n-1) * (2*n-2) )
        sum = sum + term
    return sum
while (True):
    try:
        x = float(input())
```

```
    except:
        print ("Exit")
        break
    if (math.sin(x) != 0):
        print (abs ((my_sin(x) - math.sin(x))/(math.sin(x))), my_sin(x +
            2*math.pi) - my_sin(x))
    else:
        print ("sin(x) = 0, my_sin(x) = ", my_sin(x))
```

Точность увеличилась на диапазоне $|x| \in (\pi, +\infty)$ и стала такой же, как на диапазоне $|x| \in [0, \pi]$

13)

```
import math
def my_sin(x):
    term = x
    sum = x
    eps = 10**(-8)
    n = 1
    while (abs(term) > abs(sum * eps)):
        n += 1
        term = -term * x**2 /( (2*n-1) * (2*n-2) )
        sum = sum + term
    return sum
x = 0
while (True):
    x+= 0.25
    if (math.sin(x) != 0):
        print ("x = ", x, " ", abs ((my_sin(x) - math.sin(x))/(math.sin(x
            ))), my_sin(x + 2*math.pi) - my_sin(x))
    else:
        print ("sin(x) = 0, my_sin(x) = ", my_sin(x))
```

Алгоритм резко теряет точность где-то при $x \in [30 - 34]$. Что значит "перестаёт сходиться" - непонятно.

15)

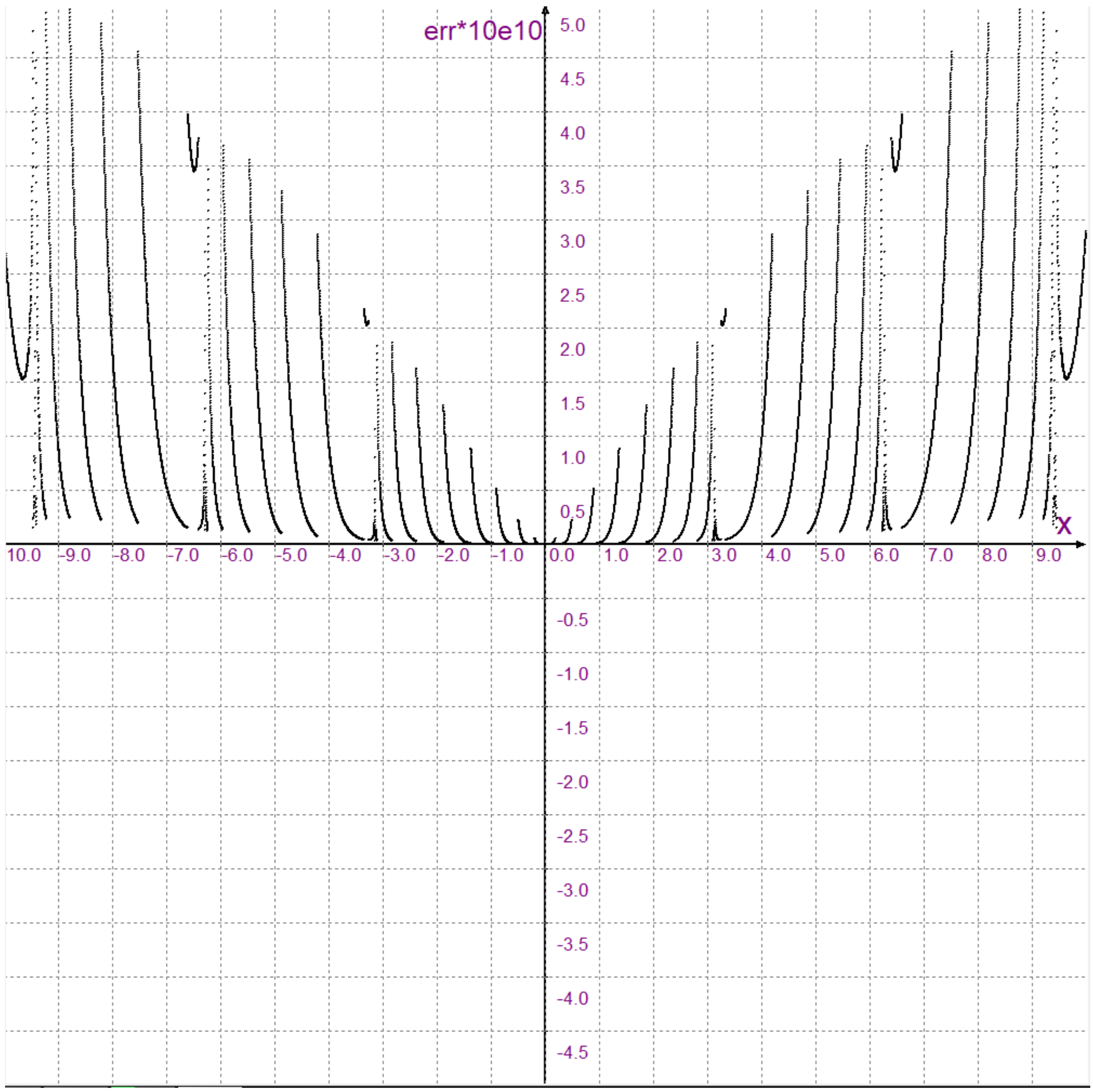
Что такое N? Но можно построить график зависимости $\varepsilon_{\sin(x)}(x)$.

```
from math import *
from tkinter import *
```

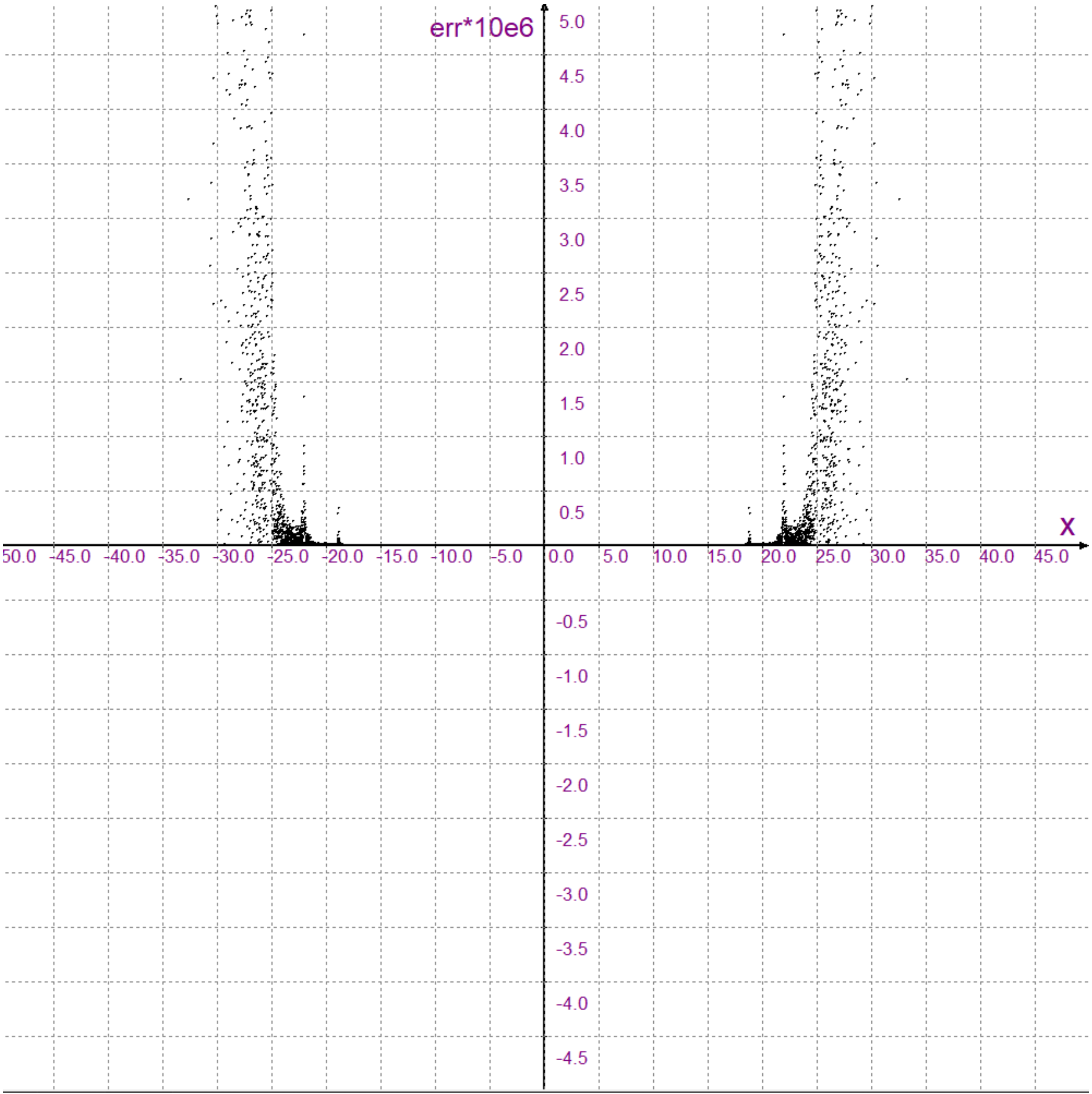
```
def my_sin(x):
    term = x
    sum = x
    eps = 10**(-8)
    n = 1
    while (abs(term) > abs(sum * eps)):
        n += 1
        term = -term * x**2 /( (2*n-1) * (2*n-2) )
        sum = sum + term
    return sum


def err(x):
    if (sin(x) != 0):
        return abs((my_sin(x) - sin(x))/sin(x))
    else:
        return 0



root = Tk()

x_0 = 100
y_0 = 10**(12)
x_sc = 10**(10)

canv = Canvas(root, width = 1000, height = 1000, bg = "white")
canv.create_line(500, 1000, 500, 0, width = 2, arrow = LAST)
canv.create_line(0, 500, 1000, 500, width = 2, arrow = LAST)
canv.create_text(980, -20 +500, font = ("Purisa", 18), text = "x", fill
   = "purple")
canv.create_text(-57 + 500, 25, font = ("Purisa", 15), text = "err*" +
   str(x_sc), fill = "purple")

First_x = -500



for i in range(16000):
    if (i % 800 == 0):
        k = First_x + (1 / 16) * i
        canv.create_line(k + 500, -3 + 500, k + 500, 3 + 500, width=0.5,
           fill='black')
```

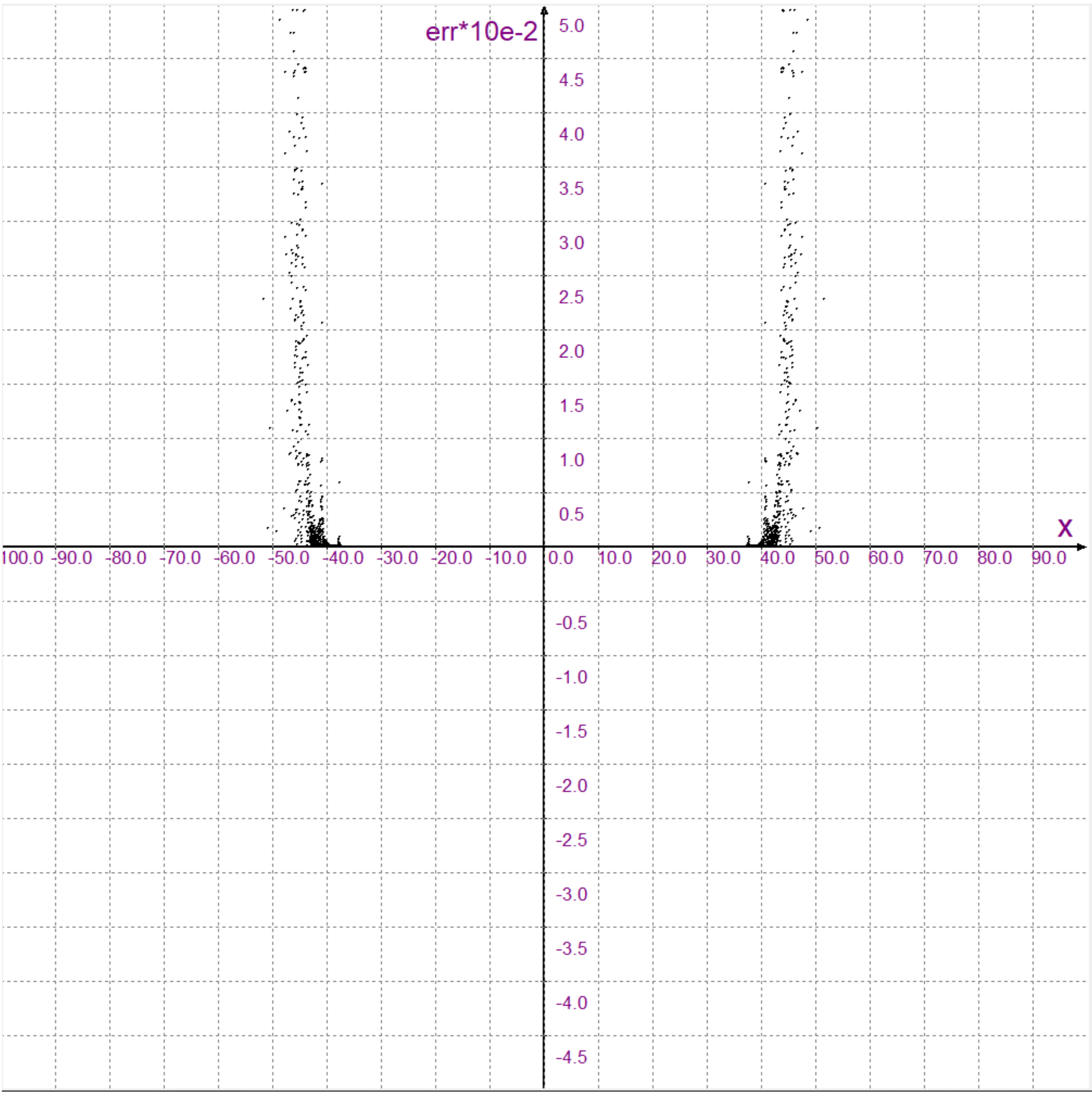
```
        canv.create_line(k + 500, 0, k + 500, 1000, width=0.1, fill='grey
            ', dash=(1, 1))
        canv.create_text(k + 515, 10 + 500, font = ("Purisa", 10), text=
            str(k/x_0), fill="purple")
        if (k != 0):
            canv.create_line(-3 + 500, k + 500, 3 + 500, k + 500, width
                =0.5, fill='black')
            canv.create_line(0, k + 500, 1000, k + 500, width=0.1, fill='
                grey', dash=(1, 1))
            canv.create_text(25 + 500, k + 500 + 20, font = ("Purisa",
                10), text=str(-k/y_0*x_sc), fill="purple")
    try:
        x = First_x + (1 / 16) * i
        y = -err(x/x_0)*y_0 + 499
        x += 499
        canv.create_oval(x, y, x + 1, y + 1, fill='black')
    except:
        First_x = -500


canv.pack()
root.mainloop()
```

Вот несколько масштабов:



err*10e10
X
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
10.0
-9.0
-8.0
-7.0
-6.0
-5.0
-4.0
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0


err*10e6
X
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
50.0 -45.0 -40.0 -35.0 -30.0 -25.0 -20.0 -15.0 -10.0 -5.0 0.0 5.0 10.0 15.0 20.0 25.0 30.0 35.0 40.0 45.0



Видно резкое уменьшение точности

5. Задача 5

a)

$ax^2 + bx + c = 0$

$x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a} = \frac{2c}{-b \mp \sqrt{b^2-4ac}} = \frac{b(-1 \pm sign(b)\sum_{n=0}^{\infty} \frac{(-1)^n (2n)!}{(1-2n) n!^2 r^n} x^n)}{2a} = \frac{2c}{b(-1 \mp sign(b)\sum_{n=0}^{\infty} \frac{(-1)^n (2n)!}{(1-2n) n!^2 r^n} x^n)}$

Третий и четвёртый способ вычисления даёт выигрыш в точности при вычисле-

нии x_1, а первый и второй - при вычислении x_2 при $a = b = 1$, $c = 10^{-n}, n > 0$
$a = 1$
$b = 1$
$c = 0.0001$

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$
-0.00010001000200049459	-0.9998999899979994 (1)
-0.00010001000200049459	-0.9998999899979994 (2)
-0.00010001000200050015	-0.9998999899980551 (3)
-0.00010001000200050015	-0.9998999899980551 (4)

```
from math import *

def solve(a, b, c):
    if (b**2 >= 4*a*c) and (a != 0) and (c != 0):
        print ((-b + sqrt(b**2-4*a*c))/(2*a), (-b - sqrt(b**2-4*a*c))/(2*
            a))
        print ((2*c)/(-b - sqrt(b**2-4*a*c)), (2*c)/((-b + sqrt(b**2-4*a*
            c))))
        sum = 0
        x = -4*a*c/b**2
        f = 1
        term = 1
        sum = 1
        eps = 10 ** (-40)
        n = 0
        while (abs(term) > abs(sum * eps)):
            term = term * x * (1-2*n)/(2*(n+1))
            sum = sum + term
            n += 1
        print (b*(-1+b/abs(b)*sum)/(2*a), b*(-1-b/abs(b)*sum)/(2*a))
        print ((2*c)/(b*(-1-b/abs(b)*sum)), (2*c)/(b*(-1+b/abs(b)*sum)))

    else:
        if (a == 0) and ( b != 0):
            print ((-c/b), (-c/b))
        else:
            if (a == 0) and ( b == 0):
                if (c == 0):
                    print ("x in (-inf, +inf)")
                else:
                    if ( c != 0):
                        print("NaN, NaN")
                    else:
```

```
                    print("Error: D < 0")


while (True):
    try:
        a = float(input("a = "))
        b = float(input("b = "))
        c = float(input("c = "))
    except:
        print ("Exit")
        break
    solve(a, b, c)
```

6. Задача 5

a)

```
from math import *

def s_1(N):
    n = 1
    sum = -0.5
    term = -0.5
    while (n < 2*N):
        term = term * (-1)*((n+1)**2)/(n*(n+2))
        sum += term
        n += 1
    return sum

def s_2(N):
    n = 1
    sum = 0
    while (n < N):
        sum += -(2*n-1)/(2*n) + (2*n)/(2*n+1)
        n += 1
    return sum

def s_3(N):
    n = 1
    sum = 1/6
    term = 1/6
```

```
    while (n < N):
        term = term * (2*n*(2*n+1))/((2*n+2)*(2*n+3))
        sum += term
        n += 1
    return sum
```

b)

Я записал точки в файл и построил логарифмы в GnuPlot, потому что ~~мне было лень думать, как построить логарифмические координаты в своей программе~~ мне хотелось попробовать запись в файл на python'e.

```
from math import *
from tkinter import *

def s_1(N):
    n = 1
    sum = -0.5
    term = -0.5
    while (n < 2*N):
        term = term * (-1)*((n+1)**2)/(n*(n+2))
        sum += term
        n += 1
    return sum

def s_2(N):
    n = 1
    sum = 0
    while (n < N):
        sum += -(2*n-1)/(2*n) + (2*n)/(2*n+1)
        n += 1
    return sum

def s_3(N):
    n = 1
    sum = 1/6
    term = 1/6
    while (n < N):
        term = term * (2*n*(2*n+1))/((2*n+2)*(2*n+3))
        sum += term
        n += 1
    return sum

def err(x):
```

```
    try:
        return abs((s_1(x)-s_3(x))/s_3(x))
    except:
        return 1


root = Tk()

x_0 = 10**(0)
y_0 = 10**(2)
x_sc = 10**(0)

canv = Canvas(root, width=1000, height=1000, bg="white")
canv.create_line(500, 1000, 500, 0, width=2, arrow=LAST)
canv.create_line(0, 500, 1000, 500, width=2, arrow=LAST)
canv.create_text(980, -20 + 500, font=("Purisa", 18), text="x", fill="
    purple")
canv.create_text(-57 + 500, 25, font = ("Purisa", 15), text="err*"+ str(
    x_sc), fill="purple")

First_x = -500;


my_file = open("lg.dat", "w")

for i in range(18000):
    if (i % 1800 == 0):
        k = First_x + (1 / 18) * i
        canv.create_line(k + 500, -3 + 500, k + 500, 3 + 500, width=0.5,
            fill='black')
        canv.create_line(k + 500, 0, k + 500, 1000, width=0.1, fill='grey
            ', dash=(1, 1))
        canv.create_text(k + 515, 10 + 500, font = ("Purisa", 10), text=
            str(k/x_0), fill="purple")
        if (k != 0):
            canv.create_line(-3 + 500, k + 500, 3 + 500, k + 500, width
                =0.5, fill='black')
            canv.create_line(0, k + 500, 1000, k + 500, width=0.1, fill='
                grey', dash=(1, 1))
            canv.create_text(25 + 500, k + 500 + 20, font = ("Purisa",
                10), text=str(-k/y_0*x_sc), fill="purple")
    try:
        x = First_x + (1 / 18) * i
        y = -err(x/x_0)*y_0 + 499
```

```
        x += 499
        canv.create_oval(x, y, x + 1, y + 1, fill='black')
        my_file.write(str(x - 499) + " " + str(499-y) + "\n")
    except:
        First_x = -500
my_file.close()
canv.pack()
root.mainloop()
```

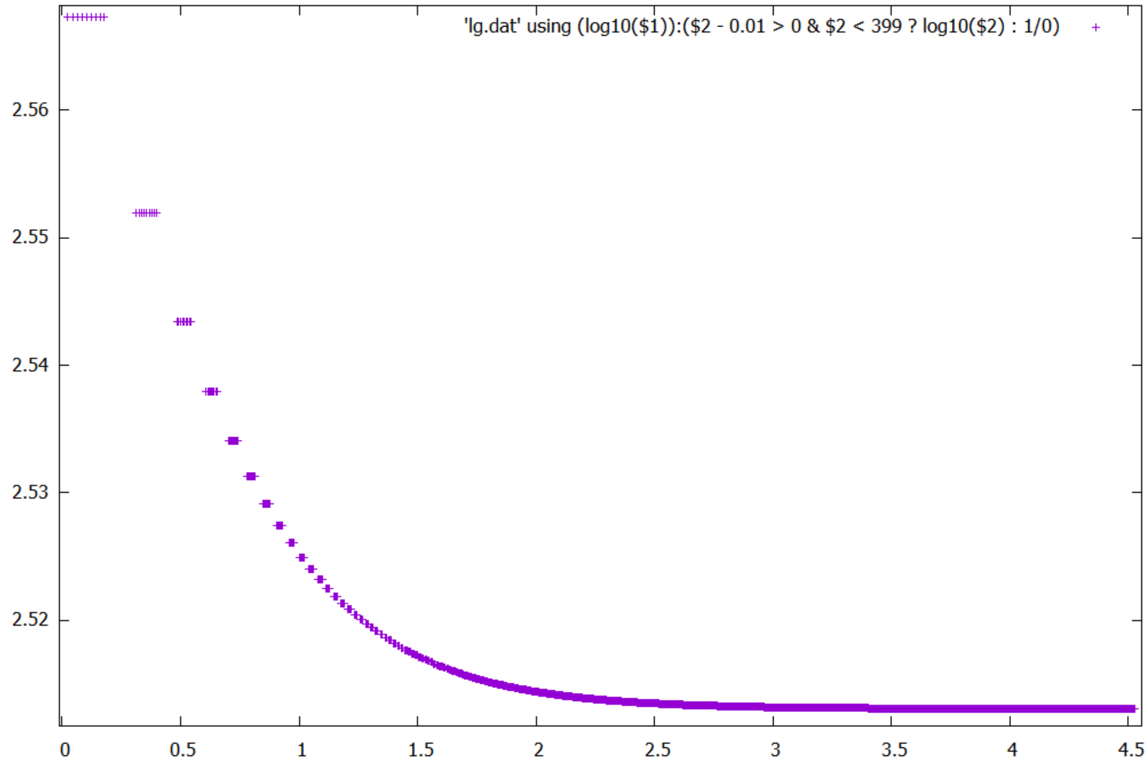
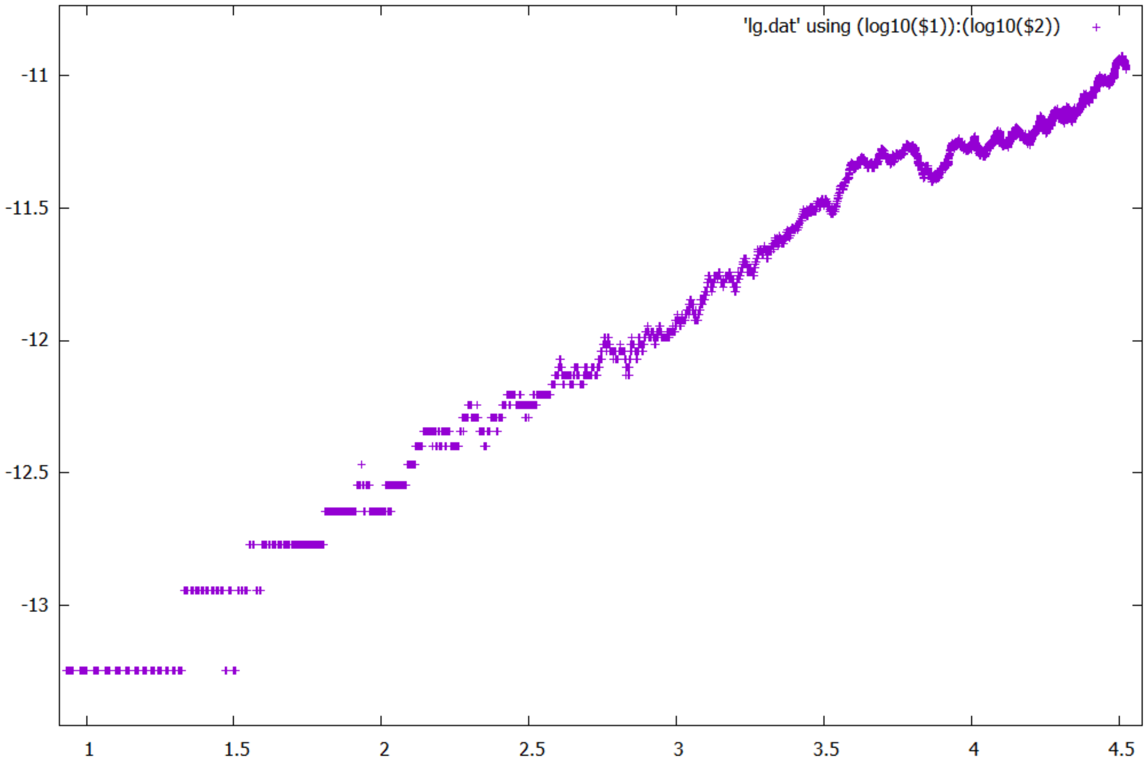


График показывает зависимость $\lg|(S_N^{(1)} - S_N^{(3)})/S_N^{(3)}|$ от $\lg(N)$. За много часов работы программы (из-за этого я не успел в дедлайн) было получено 600 000 точек. Видно, что при увеличении N зависимость становится приближенно линейной.

Но на этом же графике, в области отрицательных чисел есть ещё точки:


'lg.dat' using (log10($1)):(log10($2))
-11
-11.5
-12
-12.5
-13
1
1.5
2
2.5
3
3.5
4
4.5