Домашняя контрольная работа

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

Желаемая оценка - 10.

Для выполнения заданий я выбрал функцию *тангенс* $\tan(x)$

1. Задача 1

1) Посчитать ошибку вычисления тангенса, как сумму ряда $\tan x = \sum_{n=1}^{\infty} \frac{B_{2n}(-4)^n(1-4^n)}{(2n)!} x^{2n-1}$ **, где** $\begin{cases} B_n = \frac{-1}{n+1} \sum_{k=1}^{n} C_{k+1}^{n+1} B_{n-k} \\ B_0 = 1 \end{cases}$

```
import math


def my_tan(x):
    term = x
    sum = x
    eps = 10 ** (-8)
    B = [1]
    for n in range(150):
        n += 1
        b = 0
        npofact = math.factorial(n + 1)
        kpofact = 1
        for i in range(n):
            i += 1
            kpofact *= (i + 1)
            b += B[n - i] * (-1) * npofact / ((n + 1) * (kpofact * math.
                factorial(n - i)))
        B.append(b)
    n = 1

    while abs(term) > abs(sum * eps) and n < 50:
       n += 1
       term = B[2 * n]/B[2 * n - 2] * (-4) * (1 - 4 ** n)/(1 - 4 ** (n -
          1)) * term * x ** 2 / ((2 * n ) * (2 * n - 1))
       sum += term
    print(n)
    return sum
```

```
while (True):
    try:
        x = float(input())
    except:
        print("Exit")
        break
    if math.tan(x) != 0:
        print(abs((my_tan(x) - math.tan(x)) / (math.tan(x))))
    else:
        print("tan(x) = 0, my_tan(x) = ", my_tan(x))
```

2) Значения

x	$\varepsilon_{\mathbf{sin}(x)}$
0 (0)	0.0/0.0
3.14159 (π)	$4 \cdot 10^{35}$
3.141592654 (π)	$3 \cdot 10^{39}$
6.28318 (2π)	$1 \cdot 10^{65}$
6.283185307 (2π)	$3 \cdot 10^{69}$
1.57 ($\pi/2$)	$1 \cdot 10^{0}$
1	$5 \cdot 10^{-9}$
0.1	$5 \cdot 10^{-13}$
0.01	$5 \cdot 10^{-14}$
0.001	$0 \cdot 10^{0}$

4) Сходится ли алгоритм к правильному ответу при малых x?

Действительно, при малых x алгоритм сходится к правильному ответу.

5) Непонятно

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

13) Увеличение точности с использованием тождества $tan(x) = tan(x + \pi)$

```
import math

def my_tan(x):
    while abs(x) > math.pi/2:
        if x > 0:
```

```
            x -= math.pi
        else:
            x += math.pi
    term = x
    sum = x
    eps = 10 ** (-8)
    B = [1]
    for n in range(150):
        n += 1
        b = 0
        npofact = math.factorial(n + 1)
        kpofact = 1
        for i in range(n):
            i += 1
            kpofact *= (i + 1)
            b += B[n - i] * (-1) * npofact / ((n + 1) * (kpofact * math.
                factorial(n - i)))
        B.append(b)
    n = 1

    while abs(term) > abs(sum * eps) and n < 50:
       n += 1
       term = B[2 * n]/B[2 * n - 2] * (-4) * (1 - 4 ** n)/(1 - 4 ** (n -
          1)) * term * x ** 2 / ((2 * n ) * (2 * n - 1))
       sum += term
    print(n)
    return sum

while (True):
    try:
        x = float(input())
    except:
        print ("Exit")
        break
    if (math.tan(x) != 0):
        print (abs ((my_tan(x) - math.tan(x))/(math.tan(x))), my_tan(x +
            2*math.pi) - my_tan(x))
    else:
        print ("tan(x)␣=␣0,␣my_tan(x)␣=␣", my_tan(x))
```

Точность увеличилась на диапазонах $x \in (\pi/2, +\infty) \cup (-\infty, -\pi/2)$ и стала такой же, как на диапазоне $x \in [-\pi/2, \pi/2]$

14) Где пропадает точность?

```
import math
def my_tan(x):
    term = x
    sum = x
    eps = 10 ** (-8)
    B = [1]
    for n in range(150):
        n += 1
        b = 0
        npofact = math.factorial(n + 1)
        kpofact = 1
        for i in range(n):
            i += 1
            kpofact *= (i + 1)
            b += B[n - i] * (-1) * npofact / ((n + 1) * (kpofact * math.
               factorial(n - i)))
        B.append(b)
    n = 1

    while abs(term) > abs(sum * eps) and n < 50:
       n += 1
       term = B[2 * n]/B[2 * n - 2] * (-4) * (1 - 4 ** n)/(1 - 4 ** (n -
          1)) * term * x ** 2 / ((2 * n ) * (2 * n - 1))
       sum += term
    print(n)
    return sum

x = 0
while (True):
    x+= 0.025
    if (math.tan(x) != 0):
        print ("x␣=␣", x, "␣␣␣␣", abs ((my_tan(x) - math.tan(x))/(math.
           tan(x))))
    else:
        print ("tan(x)␣=␣0,␣my_tan(x)␣=␣", my_tan(x))
```

Алгоритм резко теряет точность где-то при $x \in [1.50, 1.58]$, то есть, в области $\pi/2$. Что значит "перестаёт сходиться" - непонятно.

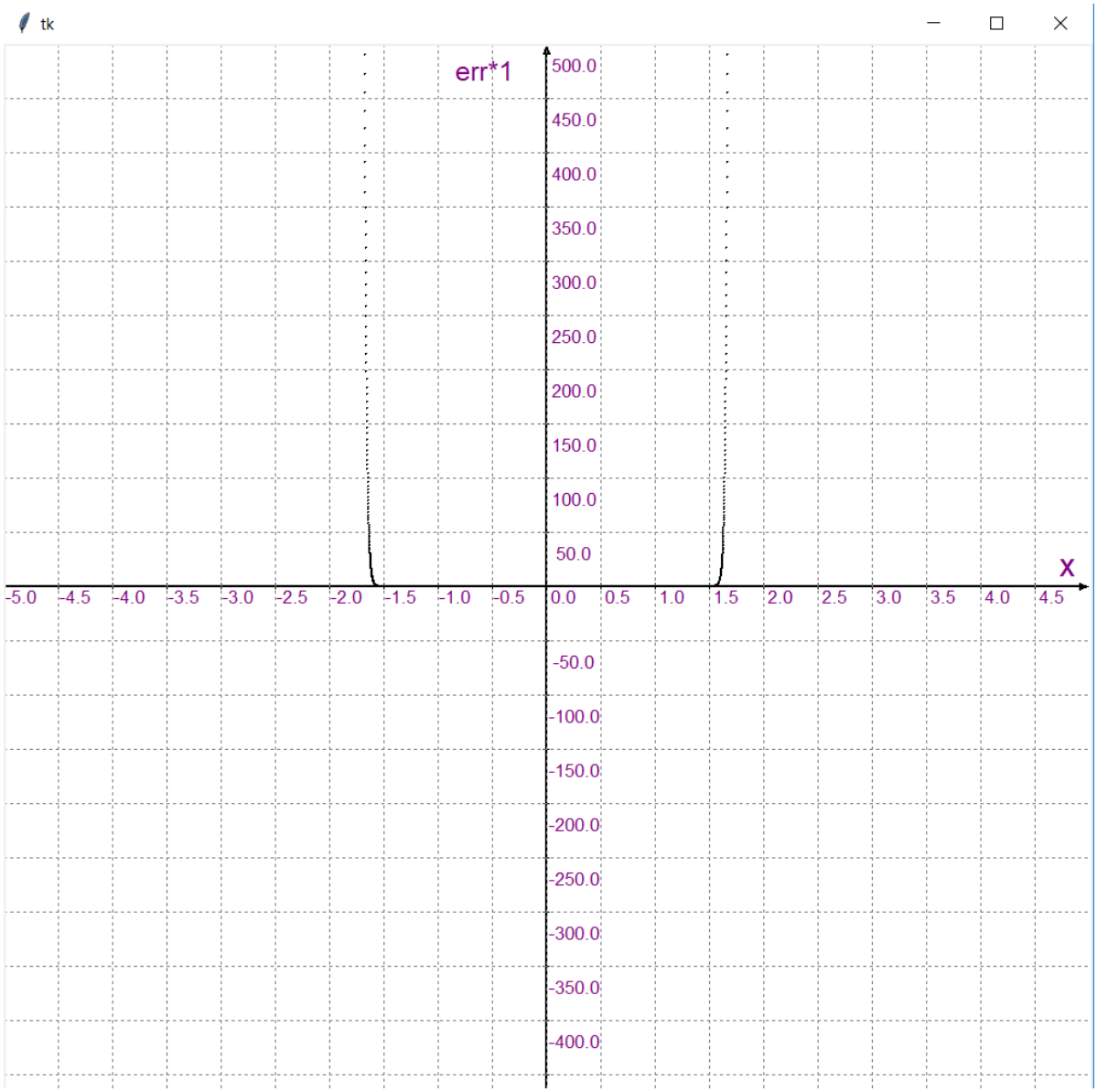
15) Зависимость ошибки от x

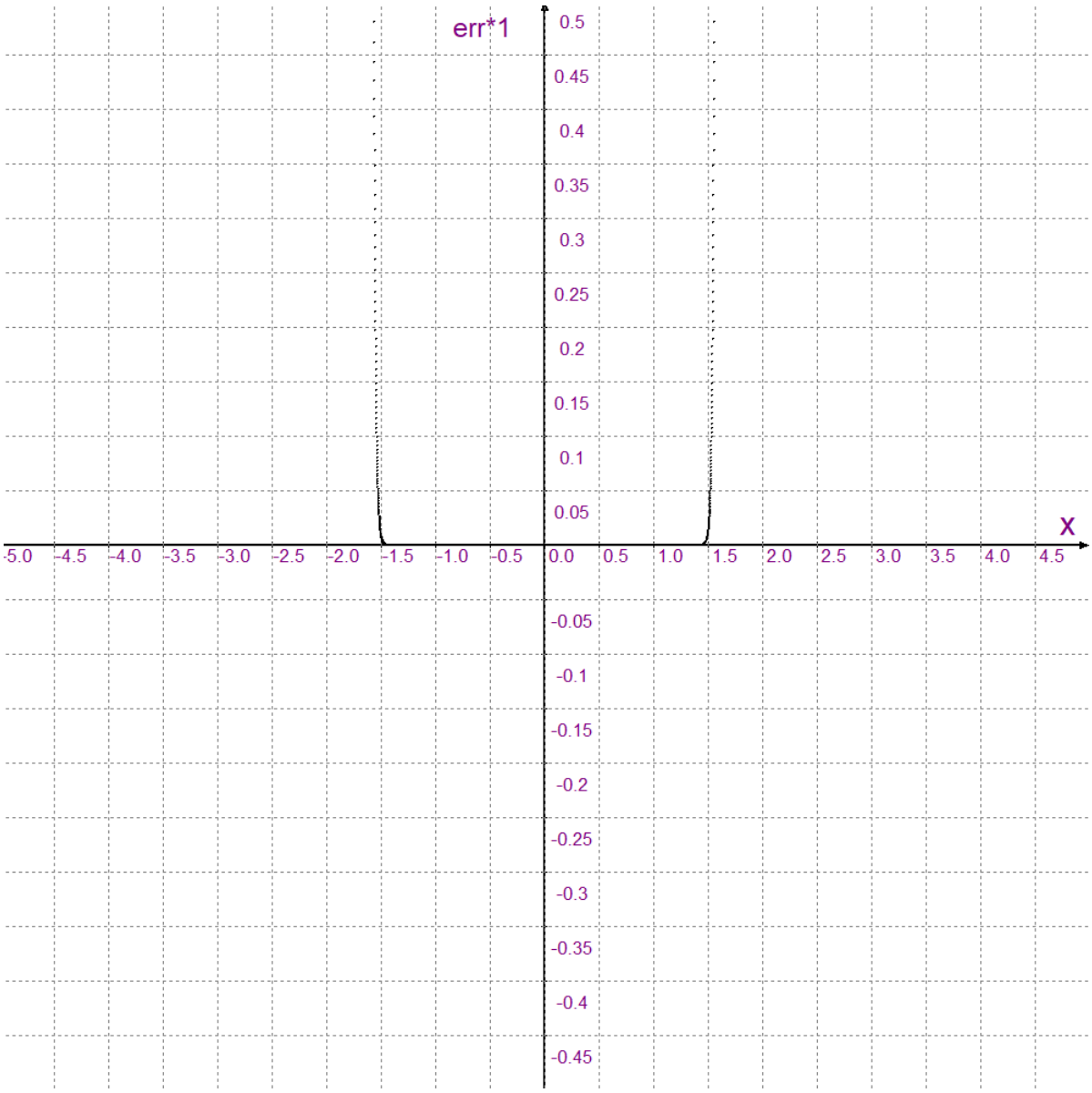
Что такое N? Но можно построить график зависимости $\varepsilon_{\sin(x)}(x)$.

```
from math import *
from tkinter import *

def my_tan(x):
    term = x
    sum = x
    eps = 10 ** (-8)
    B = [1]
    for n in range(150):
        n += 1
        b = 0
        npofact = factorial(n + 1)
        kpofact = 1
        for i in range(n):
            i += 1
            kpofact *= (i + 1)
            b += B[n - i] * (-1) * npofact / ((n + 1) * (kpofact *
               factorial(n - i)))
        B.append(b)
    n = 1

    while abs(term) > abs(sum * eps) and n < 50:
        n += 1
        term = B[2 * n] / B[2 * n - 2] * (-4) * (1 - 4 ** n) / (1 - 4 **
           (n - 1)) * term * x ** 2 / (
               (2 * n) * (2 * n - 1))
        sum += term
    return sum

def err(x):
    if (tan(x) != 0):
        return abs((my_tan(x) - tan(x))/tan(x))
    else:
        return 0



root = Tk()

x_0 = 10**(2)
y_0 = 10**(3)
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



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')
        if i % 1600 == 0:
            print(i/1600)
    except:
        First_x = -500


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

Вот несколько масштабов:


tk
err*1
500.0
450.0
400.0
350.0
300.0
250.0
200.0
150.0
100.0
50.0
X
-5.0
-4.5
-4.0
-3.5
-3.0
-2.5
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
-50.0
-100.0
-150.0
-200.0
-250.0
-300.0
-350.0
-400.0



Видно резкое уменьшение точности
Каждый график программа считала долго, около 10 минут.