Исследование функции 1/(1+x^2).

Давайте исследуем следующую интересную функцию – f(x)=1/(1+x^2). Для вычисления её значения сперва воспользуемся разложением этой функции в ряд Тейлора в окрестности точки x=0. Данное разложение задаётся следующей формулой:

$$\frac{1}{1+x^2} = \sum_{k=0}^{\infty} (-1)^k x^{2k}$$

Создадим следующую программу в Python 3:

```
x=float(input("Введите x: "))
f=1/(1+(x**2))
s=1
t=1
k=1
eps=10**(-8)
while abs(t)>eps:
    t=t*(-1)*(x**2)
    s=s+t
    k=k+1
print(s,abs((s-f)/f),k)
```

Результаты вычислений представлены в таблице (x – аргумент функции, s – вычисленная сумма ряда, k – количество суммируемых членов, |s-f|/s – относительная погрешность вычисления):

x	s	k	|s-f|/s
0.1	0.9900990099	6	9.999978622943218e-13
0.3	0.9174311930159822	9	3.874206566667482e-10
0.5	0.8000000007450581	15	9.313225191043273e-10
0.7	0.6711409424953789	27	4.31811454926212e-09
0.9	0.5524861918034165	89	7.164183833952009e-09
1	программа не работает		
2	программа не работает		

Как видно из таблицы, точность вычислений при |x|<1 достаточно высокая, но при x=1 и больших значениях программа перестаёт работать! Точнее, она входит в бесконечный цикл while, т.к. при x>1 общий член соответствующего ряда Тейлора стремится к бесконечности.

В чём же дело? Почему ряд Тейлора везде определённой, непрерывной и бесконечное число раз непрерывно дифференцируемой функции f(x)=1/(1+x^2) ведёт себя так отвратительно вне интервала (-1,1)? Причина кроется в аналитических свойствах ассоциированной с ней комплекснозначной функции f(z)=1/(1+z^2), имеющей полюсы первого порядка в точках i и -i.

Для того, чтобы решить данную проблему, будем раскладывать f(x) в окрестностях точек, лежащих в радиусе ½ от аргумента: таким образом мы никогда не окажемся вне радиуса сходимости. Воспользуемся следующей формулой:

$$\frac{1}{1+x^2} = \frac{1}{2i}\left(\frac{1}{n-i}\sum_{k=0}^{\infty}(-1)^k\left(\frac{x-n}{n-i}\right)^k - \frac{1}{n+i}\sum_{k=0}^{\infty}(-1)^k\left(\frac{x-n}{n+i}\right)^k\right)$$

Создадим следующую программу в Python 3:

```
import math, cmath
x=complex(float(input("Введите x: ")),0)
f=1/(1+(x**2))
s1=complex(1,0)
s2=complex(1,0)
t1=complex(1,0)
t2=complex(1,0)
k=1
eps=10**(-8)
n=complex((math.floor(2*(x.real)))/2,0)
j=complex(0,1)
while abs(t1)>eps or abs(t2)>eps:
    t1=t1*(-1)*(x-n)/(n-j)
    t2=t2*(-1)*(x-n)/(n+j)
    s1=s1+t1
    s2=s2+t2
    k=k+1
s=((1/(2*j))*((1/(n-j))*s1-(1/(n+j))*s2)).real
print(s,abs((s-f)/f),k)
```

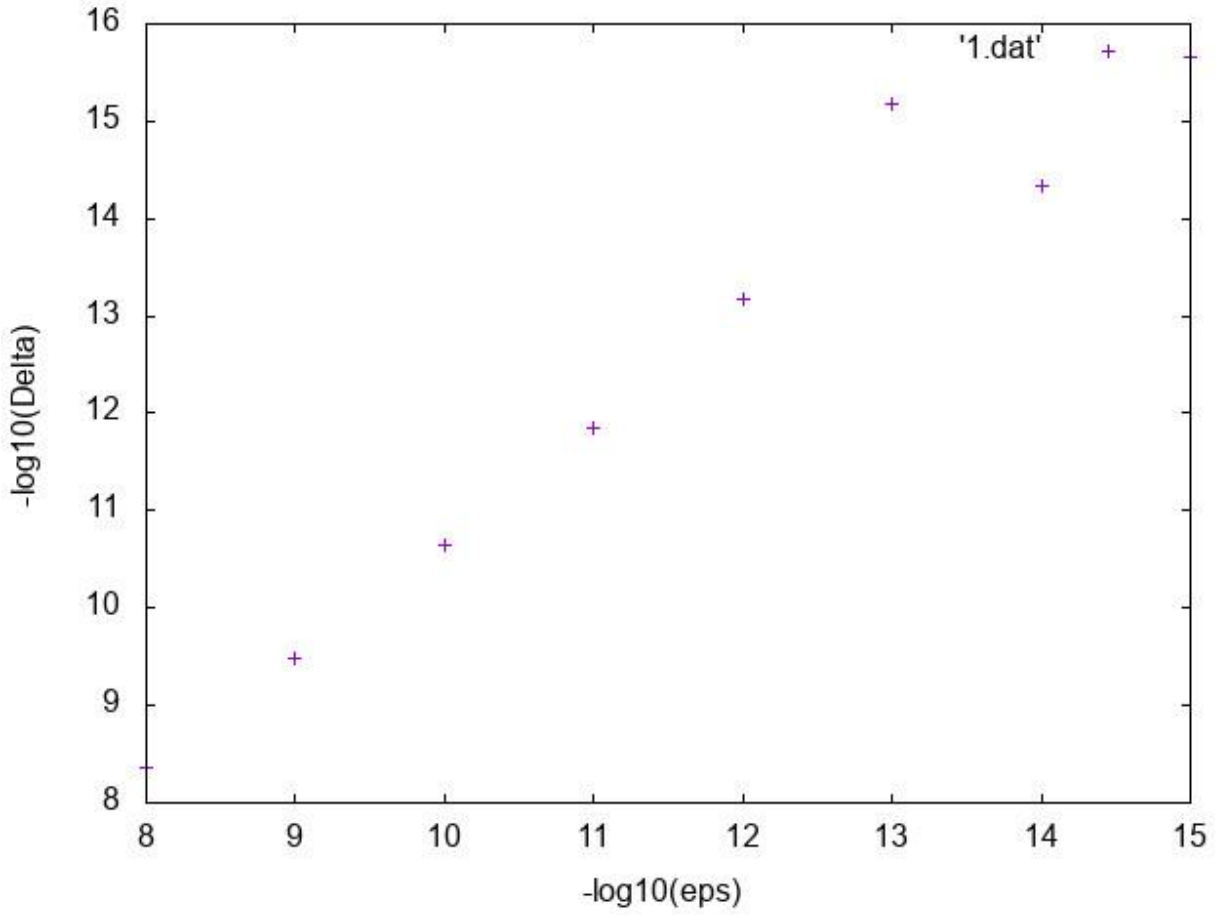
Несмотря на громоздкость формулы и использование аппарата комплексных чисел, данная программа демонстрирует высокую точность для любых значений аргумента x:

x	s	k	|s-f|/s
0.1	0.99009901	10	1.0000001049448315e-10
0.5	0.8	2	0.0
0.9	0.5524861881324554	19	5.197442598525725e-10
1	0.5	2	0.0
2	0.2	2	0.0
25	0.0015974440894568692	2	1.3574211199518516e-16
2018.2018	2.455108641989759e-07	3	4.00015602564191e-12
-100.500	9.899764880584087e-05	2	0.0
133.333	5.62471173349183e-05	5	5.914023594581582e-13

Посмотрим на зависимость относительной погрешности от точности вычисления f(0.999):

eps	s	k	|s-f|/s
10^{-8}	0.5005002478078219	24	4.379724053233724e-09
10^{-9}	0.500500249834849	27	3.2972187695945454e-10
10^{-10}	0.5005002499883313	30	2.3064012252826948e-11
10^{-11}	0.5005002499991598	33	1.4287598261164547e-12
10^{-12}	0.500500249999841	36	6.765591475943467e-14
10^{-13}	0.5005002499998745	39	6.654680140272262e-16
10^{-14}	0.5005002499998772	41	4.658276098190584e-15
10^{-15}	0.500500249999875	44	2.2182267134240875e-16
10^{-16}	0.5005002499998749	47	0.0

Следующий график иллюстрирует зависимость логарифма относительной погрешности от логарифма точности:



Мы видим, что в среднем зависимость линейная, однако имеется небольшая аномалия при подходе к максимальной точности, которую я пока не могу объяснить.