Trabalho de Cálculo II

Orientadora: Adriana Padua Lovatte

Alunos: David de Moura Marques e Magno Macedo de Oliveira

Grupo V - Séries “a e II”

a) $\sum_{n=1}^{\infty} \frac{n}{n^4+1}$

II) $\sum_{n=1}^{\infty} \frac{(-1)^n.n}{10^n}$

1 Verificar se as séries convergem:

1.1 (a) $\sum_{n=1}^{\infty} \frac{n}{n^4+1}$

Seja f(x) = $\sum_{n=1}^{\infty} \frac{n}{n^4+1}$, h(x) = $\sum_{n=1}^{\infty} \frac{n}{n^4}$ e g(x) = $\sum_{n=1}^{\infty} \frac{1}{n^3}$, então f(x) <= h(x) < g(x)

E g(x) converge (P-Série, P=3) logo, pelo Teste da Comparação, F(x) converge.

1.2 (II) $\sum_{n=1}^{\infty} \frac{(-1)^n.n}{10^n}$

Este é um exemplo de uma série alternada, para verificar sua convergência, faremos dois testes:

Se $b_{n+1} <= b_n$

E se $\lim_{n\to\infty} b_n = 0$

Dado que $b_n = \frac{n}{10^n}$ então $b_{n+1} = \frac{n+1}{10^{n+1}}$, logo $b_{n+1} <= b_n$

Calculemos agora $\lim_{n\to\infty} b_n$

$\lim_{n\to\infty} \frac{n}{10^n}$ = $\lim_{n\to\infty} \frac{1}{10^n log(10)} = 0$

Por tanto, a série converge

2 Calcular quantas operações são necessárias para determinar a soma de uma série infinita com uma precisão de 0,00000001 ('e' < 0,0000001)

Métodos a serem utilizados:

2.1 O erro 'e' será obtido, fazendo:

e = $S_n - S_{n-1}$, onde:

S_n : É a enésima soma;

S_{n-1} : É a soma anterior a enésima soma;

2.2 O erro 'e' será obtido utilizando a estimativa do resto para integral ou utilizando o teorema de estimativa de séries alternadas, de acordo com a série.

2.2.1 A estimativa de resto para integral é dada por:

e = $\int_{n+1}^{\infty} f(x)dx$

2.2.2 O teorema de estimativa de séries alternadas:

Se $S = (-1)^{n-1}b_n$ for a soma de uma série alternada que satisfaz: $0 <= b_{n+1} <= b_n$ e $\lim_{n \to \infty} b_n = 0$ então $R_n <= b_{n+1}$

In [24]:

```
import numpy as np
import pandas as pd
from scipy.integrate import quad
import texttable as tt
from sympy import Poly, Symbol, init_printing, latex
from sympy.solvers.inequalities import reduce_rational_inequalities
PRECISAO = 0.00000001
```

In [25]:

```
def serieA(n):
    serie = 0
    for i in range(1,n):
        serie += i/((i**4)+1)
    return serie

def mod(num):
    if num < 0:
        return num * -1
    return num

def calculaErroI(serie):
    erro = 1.0
    quantidadeIteracoes = 1
    sn_menos_um = 0.0
    sn = 0.0

    tabela = tt.Texttable()
    tabela.header(['Nro Op', 'Sn-1', 'Sn', 'erro'])
    tabela.set_cols_dtype(['i','f','f','f'])
    tabela.set_precision(9)
    while erro >= PRECISAO:
        sn_menos_um = mod(serie(quantidadeIteracoes))
        sn = mod(serie(quantidadeIteracoes+1))
        erro = mod(sn - sn_menos_um)
        tabela.add_row([quantidadeIteracoes,sn_menos_um,sn,erro])
        quantidadeIteracoes+=1
    return tabela.draw()
```

2.3 Série a: $\sum_{n=1}^{\infty} \frac{n}{n^4+1}$

2.3.1 Utilizando o método 2.1

In [26]:

```
tabela = calculaErroI(serieA)

# imprime as dez primeiras linhas
print(tabela[0:][0:1218])
print('\t\t\t...')
# imprime as dez ultimas linhas
print(tabela[0:][len(tabela)-1218:])
```

```
+--------+-------------+-------------+-------------+
| Nro Op |    Sn-1     |     Sn      |    erro     |
+========+=============+=============+=============+
| 1      | 0.000000000 | 0.500000000 | 0.500000000 |
+--------+-------------+-------------+-------------+
| 2      | 0.500000000 | 0.617647059 | 0.117647059 |
+--------+-------------+-------------+-------------+
| 3      | 0.617647059 | 0.654232425 | 0.036585366 |
+--------+-------------+-------------+-------------+
| 4      | 0.654232425 | 0.669796627 | 0.015564202 |
+--------+-------------+-------------+-------------+
| 5      | 0.669796627 | 0.677783847 | 0.007987220 |
+--------+-------------+-------------+-------------+
| 6      | 0.677783847 | 0.682409908 | 0.004626060 |
+--------+-------------+-------------+-------------+
| 7      | 0.682409908 | 0.685324146 | 0.002914238 |
+--------+-------------+-------------+-------------+
| 8      | 0.685324146 | 0.687276794 | 0.001952648 |
+--------+-------------+-------------+-------------+
| 9      | 0.687276794 | 0.688648327 | 0.001371533 |
+--------+-------------+-------------+-------------+
| 10     | 0.688648327 | 0.689648227 | 0.000999900 |
+--------+-------------+-------------+-------------+
                       ...
+--------+-------------+-------------+-------------+
| 455    | 0.694170602 | 0.694170612 | 0.000000011 |
+--------+-------------+-------------+-------------+
| 456    | 0.694170612 | 0.694170623 | 0.000000011 |
+--------+-------------+-------------+-------------+
| 457    | 0.694170623 | 0.694170633 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 458    | 0.694170633 | 0.694170644 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 459    | 0.694170644 | 0.694170654 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 460    | 0.694170654 | 0.694170664 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 461    | 0.694170664 | 0.694170675 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 462    | 0.694170675 | 0.694170685 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 463    | 0.694170685 | 0.694170695 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 464    | 0.694170695 | 0.694170705 | 0.000000010 |
+--------+-------------+-------------+-------------+
| 465    | 0.694170705 | 0.694170715 | 0.000000010 |
+--------+-------------+-------------+-------------+
```

2.3.2 Utilizando o método 2.2.1

OBS: Para esta série deve-se utilizar o método 2.1 ao invés do 2.2 pois não se trata de uma série alternada

De acordo com 1.1, podemos escrever esta série como $\int_{n+1}^{\infty} \frac{1}{x^3} dx$

Resolvendo esta integral teremos:

$\int_{n+1}^{\infty} \frac{1}{x^3} dx = \lim_{x\to\infty} \int_{n+1}^{x} \frac{1}{x^3} dx = \lim_{x\to\infty} -\frac{1}{2x^2}]_{n+1}^{\infty}$

$\lim_{x\to\infty}[-\frac{1}{2x^2} - (-\frac{1}{2(n+1)^2})] = 0 + \frac{1}{2(n+1)^2} = \frac{1}{2(n+1)^2}$

onde n será nosso erro.

Calculando n:

$\frac{1}{2(n+1)^2} < 0,0000001$

In [27]:

```
def solveIneq():
    n = Symbol('n', real=True)
    return reduce_rational_inequalities([[1/(2*(n+1)**2) < PRECISAO]], n)
```

In [28]:

```
print(solveIneq())
```

```
((-inf < n) & (n < -7072.06781186548)) | ((7070.06781186548 < n) & (n < in
f))
```

Desconsideraremos a parte negativa visto o somatório dar-se para valores positivos. Por tanto o primeiro valor que satisfaz a equação é: **n = 7071**

2.4 Série II: $\sum_{n=1}^{\infty} \frac{(-1)^n.n}{10^n}$

2.4.1 Utilizando o método 2.1

In [29]:

```
def serieII(n):
    serie = 0
    for i in range(1,n):
        serie += ((-1**n)*n)/10**n
    return serie

tabela = calculaErroI(serieII)
print(tabela)
```

```
+--------+-------------+-------------+-------------+
| Nro Op |    Sn-1     |     Sn      |    erro     |
+========+=============+=============+=============+
| 1      | 0.000000000 | 0.020000000 | 0.020000000 |
+--------+-------------+-------------+-------------+
| 2      | 0.020000000 | 0.006000000 | 0.014000000 |
+--------+-------------+-------------+-------------+
| 3      | 0.006000000 | 0.001200000 | 0.004800000 |
+--------+-------------+-------------+-------------+
| 4      | 0.001200000 | 0.000200000 | 0.001000000 |
+--------+-------------+-------------+-------------+
| 5      | 0.000200000 | 0.000030000 | 0.000170000 |
+--------+-------------+-------------+-------------+
| 6      | 0.000030000 | 0.000004200 | 0.000025800 |
+--------+-------------+-------------+-------------+
| 7      | 0.000004200 | 0.000000560 | 0.000003640 |
+--------+-------------+-------------+-------------+
| 8      | 0.000000560 | 0.000000072 | 0.000000488 |
+--------+-------------+-------------+-------------+
| 9      | 0.000000072 | 0.000000009 | 0.000000063 |
+--------+-------------+-------------+-------------+
| 10     | 0.000000009 | 0.000000001 | 0.000000008 |
+--------+-------------+-------------+-------------+
```

2.4.2 Utilizando o método 2.2.2

OBS Aqui utilizaremos o método 2.2.2 ao invés do 2.2.1 pois se trata de uma série alternada. Onde $\varepsilon = b(n+1) = \frac{n+1}{10^{n+1}}$

In [30]:

```
def serieIIErro(n):
    return (n+1)/((10)**(n+1))

def calculaErroAlternada(serieBnMaisUm):
    n = 1
    erro = 1
    while erro >= PRECISAO:
        erro = serieBnMaisUm(n)
        n += 1
    print("Precisão: %.7f" % erro)
    print("Resultado do somatório com %i termos: %.9f" %(n, mod(serieBnMaisUm(n))))
```

In [31]:

```
calculaErroAlternada(serieIIErro)
```

```
Precisão: 0.0000000
Resultado do somatório com 9 termos: 0.000000001
```

Referências

- https://pt.sharelatex.com/learn/Integrals,_sums_and_limits#Integrals (https://pt.sharelatex.com/learn/Integrals,_sums_and_limits#Integrals)
- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#links (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#links)
- https://github.com/foutaise/texttable (https://github.com/foutaise/texttable)
- https://www.symbolab.com/ (https://www.symbolab.com/)
- http://www.wolframalpha.com/ (http://www.wolframalpha.com/)
- http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Typesetting%20Equations.html (http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Typesetting%20Equations.html)
- STEWART, James. Cálculo, Vol. 2, 7a Ediçao. Thomson Learning.