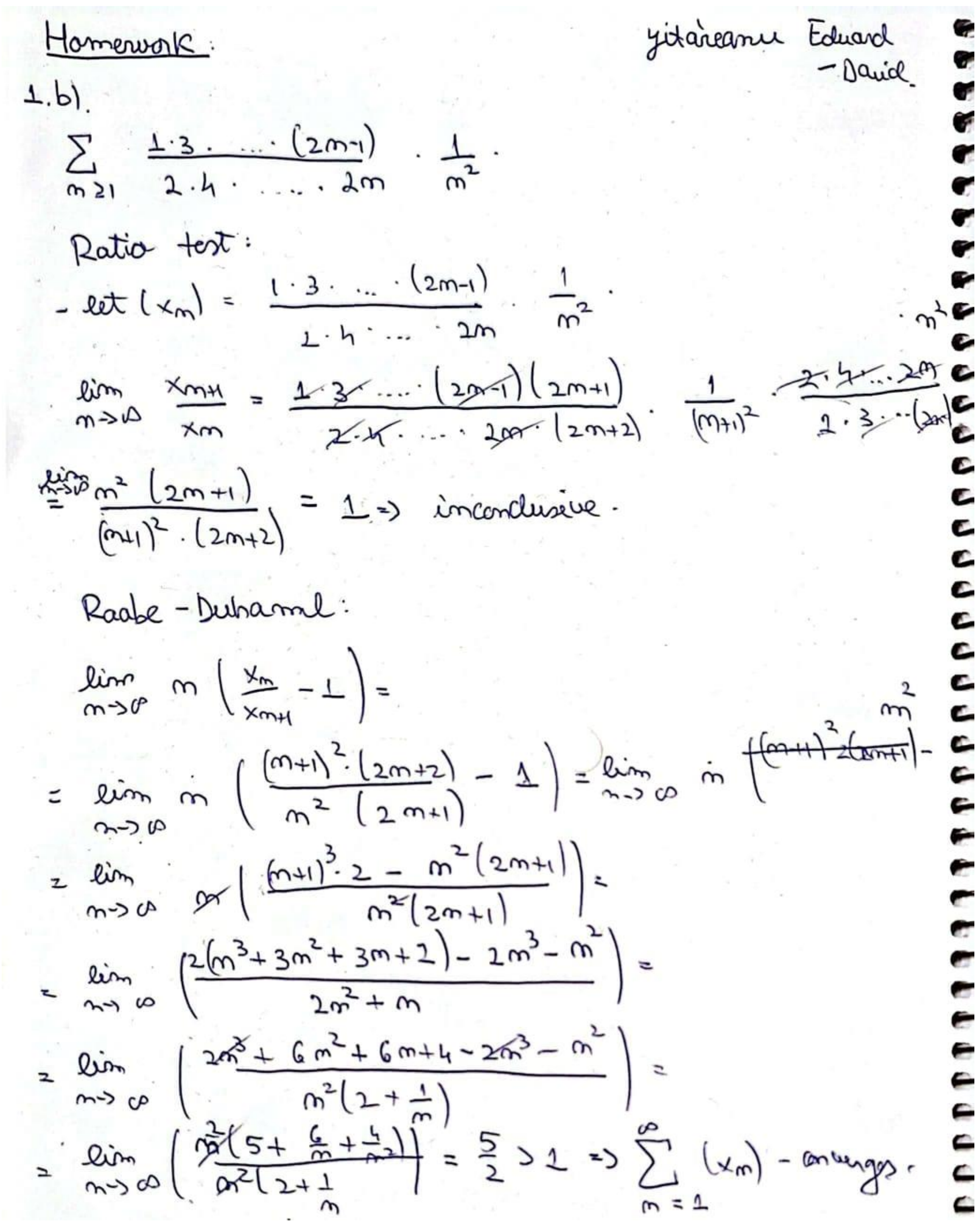
# Homework- S04

Jitareanu Eduard-David, 914



Homework: Jitareanu Eduard -David

1.b).

$$\sum_{n\geq 1} \frac{1\cdot 3\cdot \ldots \cdot (2n-1)}{2\cdot 4\cdot \ldots \cdot 2n}\cdot \frac{1}{n^2}.$$

Ratio test:

- let $(x_n) = \frac{1\cdot 3\cdot \ldots \cdot(2n-1)}{2\cdot 4\cdot \ldots \cdot 2n}\cdot\frac{1}{n^2}$.

$$\lim_{n\to\infty}\frac{x_{n+1}}{x_n} = \frac{\cancel{1}\cdot\cancel{3}\cdot\ldots\cdot(\cancel{2n-1})(2n+1)}{\cancel{2}\cdot\cancel{4}\cdot\ldots\cdot\cancel{2n}\,(2n+2)}\cdot\frac{1}{(n+1)^2}\cdot n^2\cdot\frac{\cancel{2\cdot 4\cdot\ldots\cdot 2n}}{\cancel{1\cdot 3\cdot\ldots\cdot(2n)}}$$

$$= \lim_{n\to\infty}\frac{n^2(2n+1)}{(n+1)^2\cdot(2n+2)} = 1 \Rightarrow \text{inconclusive.}$$

Raabe - Duhamel:

$$\lim_{n\to\infty} n\left(\frac{x_n}{x_{n+1}} - 1\right) =$$

$$= \lim_{n\to\infty} n\left(\frac{(n+1)^2\cdot(2n+2)}{n^2\,(2n+1)} - 1\right) = \lim_{n\to\infty} n \left(\cancel{\frac{(n+1)^2\, 2(n+1)}{n^2}} - \right.$$

$$= \lim_{n\to\infty} \cancel{n}\left(\frac{(n+1)^3\cdot 2 - n^2(2n+1)}{n^2(2n+1)}\right) =$$

$$= \lim_{n\to\infty}\left(\frac{2(n^3+3n^2+3n+2) - 2n^3 - n^2}{2n^2+n}\right) =$$

$$= \lim_{n\to\infty}\left(\frac{\cancel{2n^3} + 6n^2 + 6n + 4 - \cancel{2n^3} - n^2}{n^2\left(2+\frac{1}{n}\right)}\right) =$$

$$= \lim_{n\to\infty}\left(\frac{\cancel{n^2}\left(5+\frac{6}{n}+\frac{4}{n^2}\right)}{\cancel{n^2}\left(2+\frac{1}{n}\right)}\right) = \frac{5}{2} > 1 \Rightarrow \sum_{n=1}^{\infty}(x_n) \text{ - converges.}$$

6c) $\sum_{m\geq 1} \frac{mx^m}{2^m}$ – radius of convergence.

Power series : $\sum_{m\geq 1} a_m (x-c)^m$

$a_m (x-c)^m = \frac{m\, x^m}{2^m} \Rightarrow$

$\Rightarrow \begin{cases} a_m = \frac{m}{2^m} \\ (x-c)^m = x^m \Rightarrow c = 0 \end{cases}$

$\lim_{m\to\infty} \frac{|a_{m+1}|}{|a_m|} = \lim_{m\to\infty} \frac{m+1}{2^{m+1}} \cdot \frac{2^m}{m} = \lim_{m\to\infty} \frac{m+1}{2m} = \frac{1}{2} \Rightarrow$

$\Rightarrow L = \frac{1}{2} \in [0, +\infty) \Rightarrow R = \frac{1}{\frac{1}{2}} = 2 \Rightarrow R = 2 . \Rightarrow$

$\Rightarrow (-R, R) \subseteq C \Rightarrow (-2, 2) \subseteq C.$

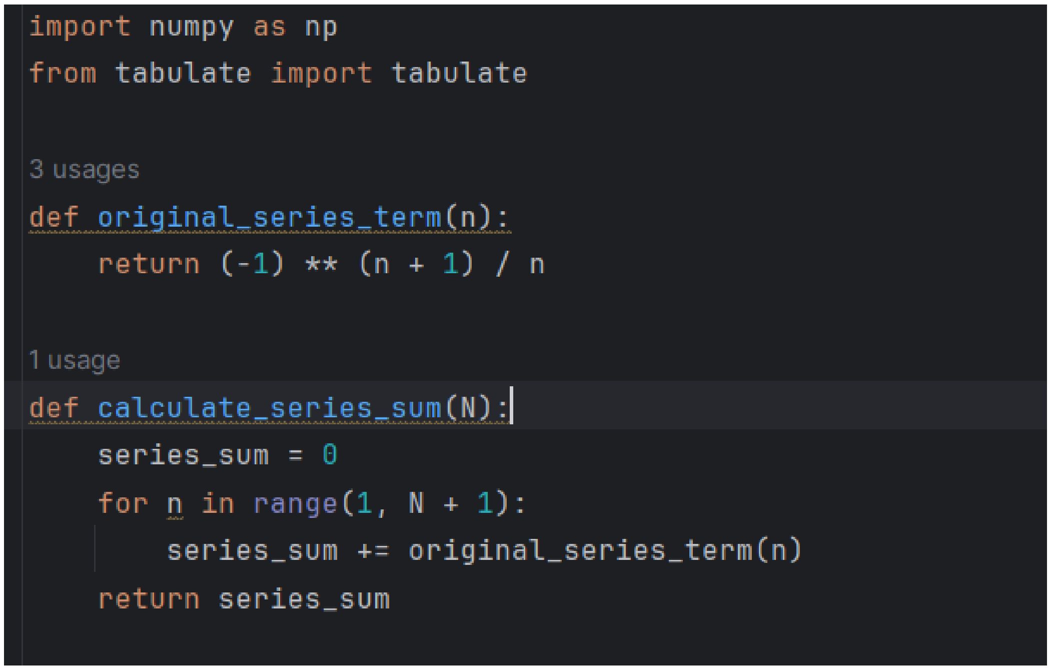
• for ~~$x > 2$~~ $x = -2 \Rightarrow \sum_{m\geq 1} \frac{m}{2^m} \cdot (-2^m) = \sum_{m\geq 1} m\cdot(-$

$\Rightarrow$ the terms dont approach a finite limit (diverge

• same for $x=2 \Rightarrow$ The convergence set is $C = (-2, 2)$.

And Exercise no. 7:

1) We import the required libraries
2) A function named "calculate_series_sum" is implemented which computes the sum, adding each time the loop occurs the appropriate term, using the calls from the "original_series_term"

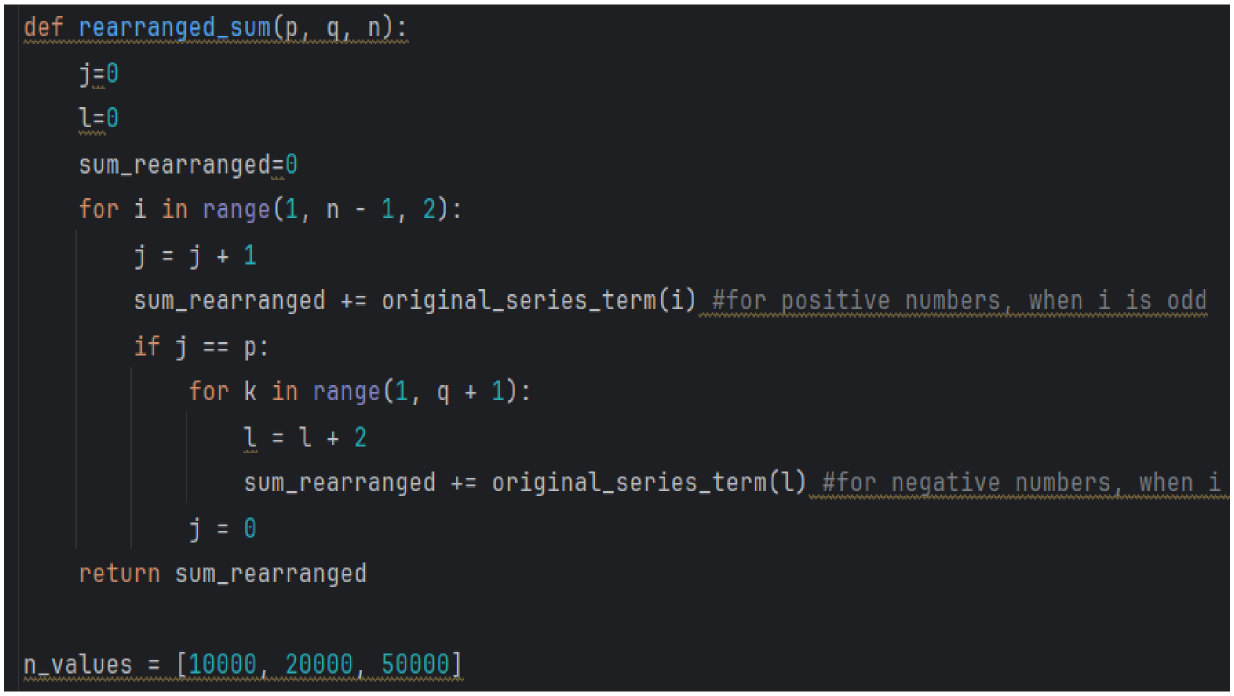


```python
import numpy as np
from tabulate import tabulate

3 usages
def original_series_term(n):
    return (-1) ** (n + 1) / n

1 usage
def calculate_series_sum(N):
    series_sum = 0
    for n in range(1, N + 1):
        series_sum += original_series_term(n)
    return series_sum
```

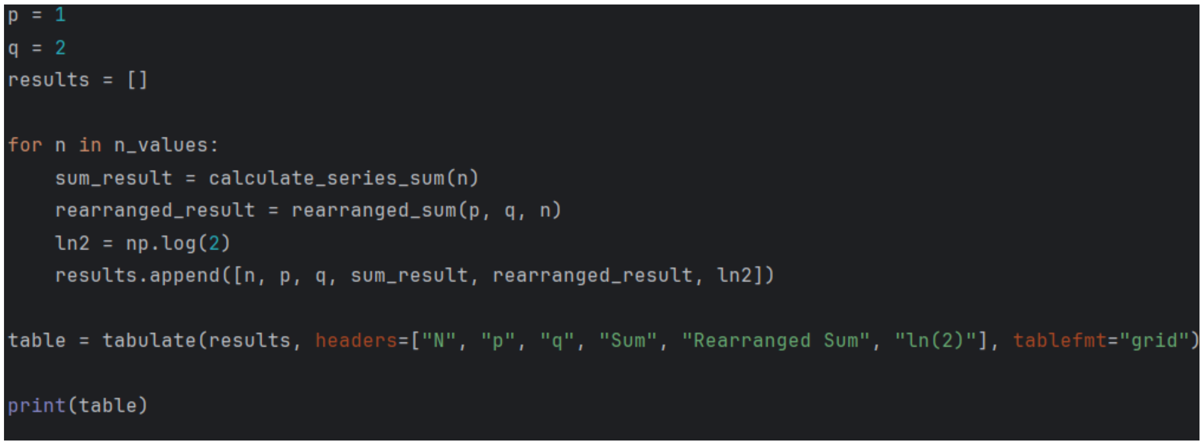
3) the function “rearranged_sum” focuses on adding first p positive numbers, then q negative ones. the “p” positive numbers are determined by a “for” with the pace 2, that only chooses the odd numbers, and after those have been added to the sum, “q” negative numbers are added with another “for”, with an even denominator. In the end, the rearranged sum is returned.



```python
def rearranged_sum(p, q, n):
    j=0
    l=0
    sum_rearranged=0
    for i in range(1, n - 1, 2):
        j = j + 1
        sum_rearranged += original_series_term(i) #for positive numbers, when i is odd
        if j == p:
            for k in range(1, q + 1):
                l = l + 2
                sum_rearranged += original_series_term(l) #for negative numbers, when i
            j = 0
    return sum_rearranged

n_values = [10000, 20000, 50000]
```

4)we choose p=1 and q=2 ro rearrange the sum, then we call the appropriate functions, and we create a table to print the results.



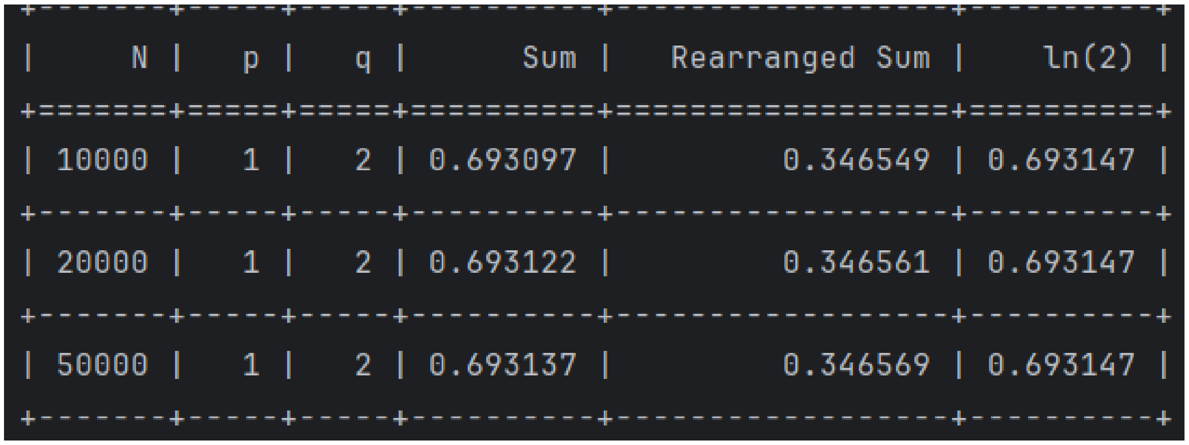
```python
p = 1
q = 2
results = []

for n in n_values:
    sum_result = calculate_series_sum(n)
    rearranged_result = rearranged_sum(p, q, n)
    ln2 = np.log(2)
    results.append([n, p, q, sum_result, rearranged_result, ln2])

table = tabulate(results, headers=["N", "p", "q", "Sum", "Rearranged Sum", "ln(2)"], tablefmt="grid")

print(table)
```

With the results:



```
+-------+-----+-----+----------+------------------+----------+
|     N |   p |   q |      Sum |   Rearranged Sum |    ln(2) |
+=======+=====+=====+==========+==================+==========+
| 10000 |   1 |   2 | 0.693097 |         0.346549 | 0.693147 |
+-------+-----+-----+----------+------------------+----------+
| 20000 |   1 |   2 | 0.693122 |         0.346561 | 0.693147 |
+-------+-----+-----+----------+------------------+----------+
| 50000 |   1 |   2 | 0.693137 |         0.346569 | 0.693147 |
+-------+-----+-----+----------+------------------+----------+
```

## Conclusion!

The implementation of problem number 7 is built of two parts: computing the sum and rearranging the elements by p positive numbers and q negative ones. As we see in the column of “Sum”, the larger the value to compute the sum for, the closer it gets to ln 2, approaching it. A similar thing can be said about the rearranged sum too. Having chosen p=1 and q=2, the sum is getting closer to what was proven in the Lecture 04 as ½(ln2), approaching it.