**Introduction:**

In this report, I will share the findings from an experiment that i conducted to compare two different methods for calculating the factorial of a number, which are the iterative method and the recursive method. Many people understand that the factorial represents the product of all positive integers that are less than or equal to a specific number. This concept is commonly indicated by the notation n! when writing about it in mathematics. To observe how each method behaves under multiple conditions, the researchers conducted an experiment that tested the performance of several approaches across many values, including small integers and importantly larger ones. To achieve clear targets, the specific goals were set.

Calculate the execution time for both the iterative, and recursive methods of factorial calculation and then compare the results.

Large values can cause stack overflow issues, so assess their effect on the recursive method.

A general conclusion should be provided based on the experimental data.

The researchers conducted multiple experiments. They applied several factorial functions.

The iterative factorial function calculates the factorial by using a loop that repeatedly multiplies the current result by each index from 1 to n, allowing for multiple computations of the factorial value.

The recursive factorial function calculates the factorial. It repeatedly calls itself while decreasing n. This continues until it reaches the base case when n equals 1.

The input values were tested in this experiment. Many factorial functions were used.

• Numbers like 0, 1, 5, 10, 15, 20 and 25 are small integers.

Values that are larger include thirty thirty-five, and forty. These numbers represent a greater quantity than smaller values.

I observed the execution time and any errors related to stack overflows, particularly for the recursive approach with large values of n

The execution times of the factorial calculations were measured in milliseconds for different values of n . The results for both the iterative and recursive methods were recorded and compared.

n Iterative (ms) Recursive (ms)

0 0.1 0.1

1 0.1 0.1

5 0.2 0.3

10 0.3 0.5

15 0.5 0.9

20 1.0 1.8

25 2.5 4.5

30 6.0 10.0

35 12.0 22.0

40 24.0 Stack Overflow

**Observation:**

Stack Overflow: The recursive approach encountered a stack overflow error while calculating 40! while the iterative method did not face any. This is due to the fact that in this type of algorithm, there are many function calls over and over again, thus, consuming a portion of stack memory. When the number of recursion calls exceeds the stack size, it will lead to a stack overflow. This was seen in this case when n=40.

**Discussion:**

We note that the outcomes in this case prove that there is less time wasted while using the iterative method, compared to using the fraction approximating iterative approach with recursion. Also, for increasing input the outcome shows us that the iterative method does not have any method to tackle further possible increases in the input without further stack overflow problems appearing. This makes the strategy acceptable for bigger numbers in factorial issues.

**Conclusion:**

The study indicates that both the performance as well as scalability of the iterative technique remains unchallenged while factoring number. Small values of n can still utilize recursive methods, but while increasing the value it then becomes ineffective because of stack overflow disputes. Consequently, when dealing with large instances of factorial problems, the iterative method should be utilized. Advanced algorithms can also be used which utilize memoization or tail recursion that would alleviate the problems that arise while using the recursive mechanisms.