Data Structures and Program Design in C

Topic 6: Analysis of Algorithms

Dr Manjusri Wickramasinghe

TAKE
NOTES

Outline

- Why Study Algorithms?
- Unit of Measurement
- Order of Growth
- Big-O Notation
- Analysis of Algorithms

Why Study Algorithms?

- To create efficient programs
- What are efficient programs?
- How does the study of data structures and algorithms enable one to improve a program's efficiency?

Unit of Measurement…(1)

- We have methods invented to measure quantities such as weight, volume etc.

- How can we do this with computers?
  - What are things that needs to be measured to increase program efficiency?

Unit of Measurement…(2)

- Let's consider the execution time as measure of efficiency
  - Many ways this can be measured

  - C
    - Using the time library
    - E.g: `#include <time.h>`

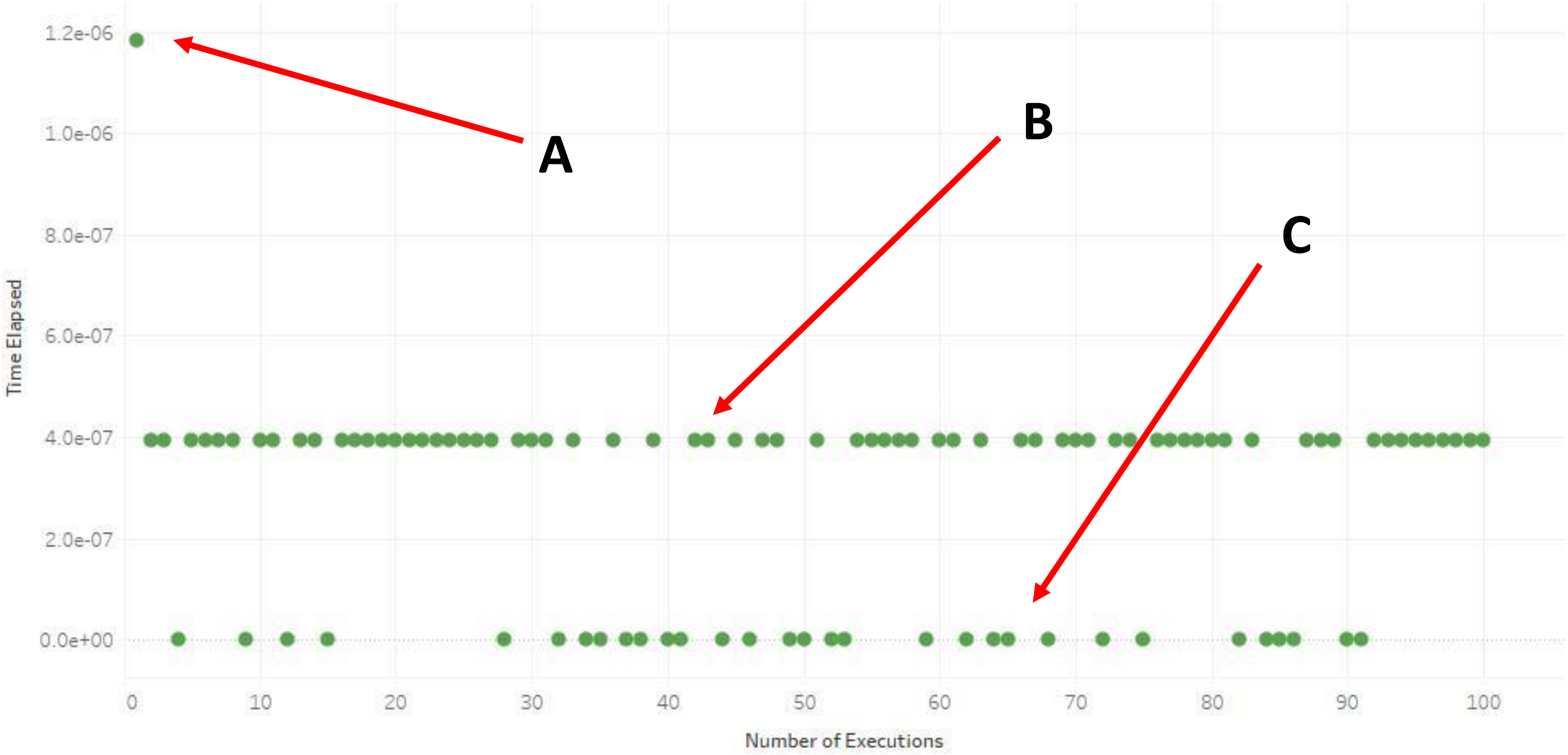
      ```
      A = clock();
      ```

  - Java
    - Using `System.nanotime()`

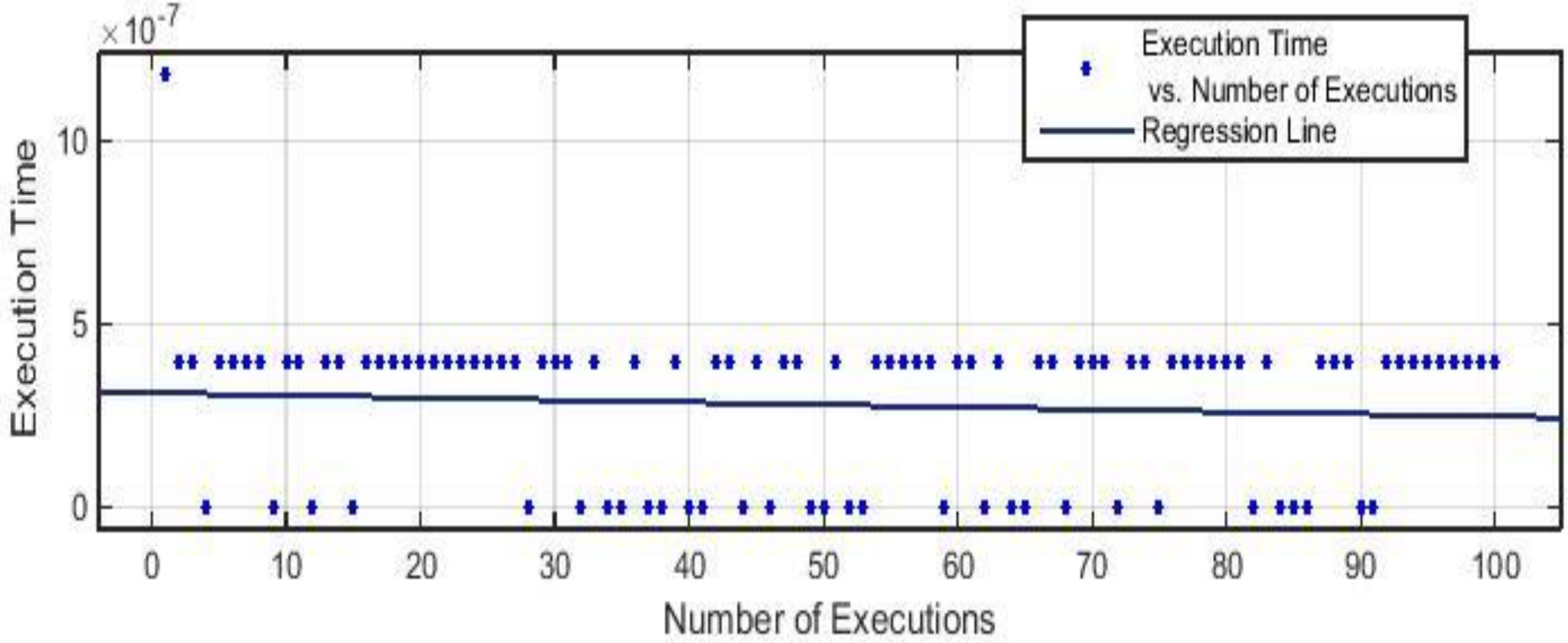
Unit of Measurement…(3)

- Lets consider a few cases in a program
  - Simple assignment (e.g. `total = 1;`)
  - Multiplication (e.g. `total = 5*6;`)
  - Programs with multiple execution of the code (e.g. :- loops)

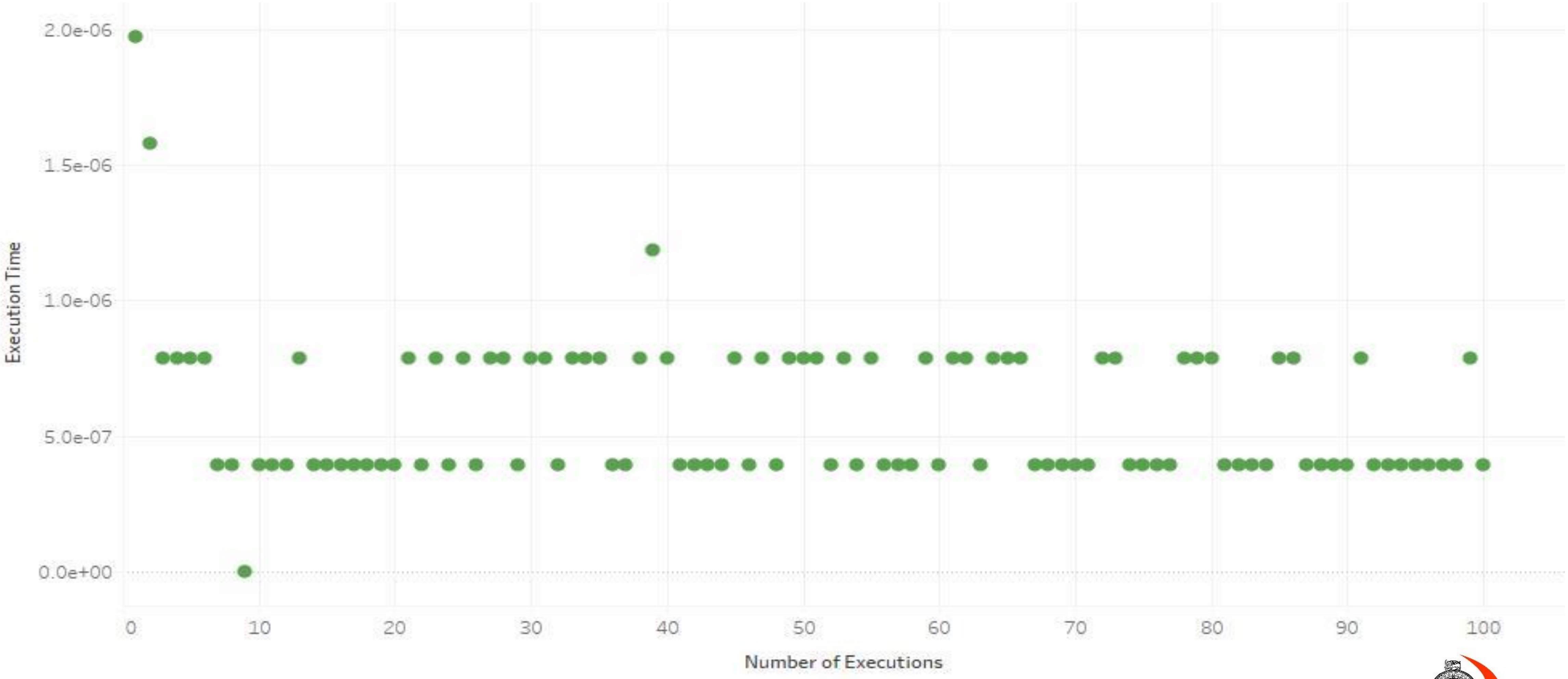
Unit of Measurement…(4)



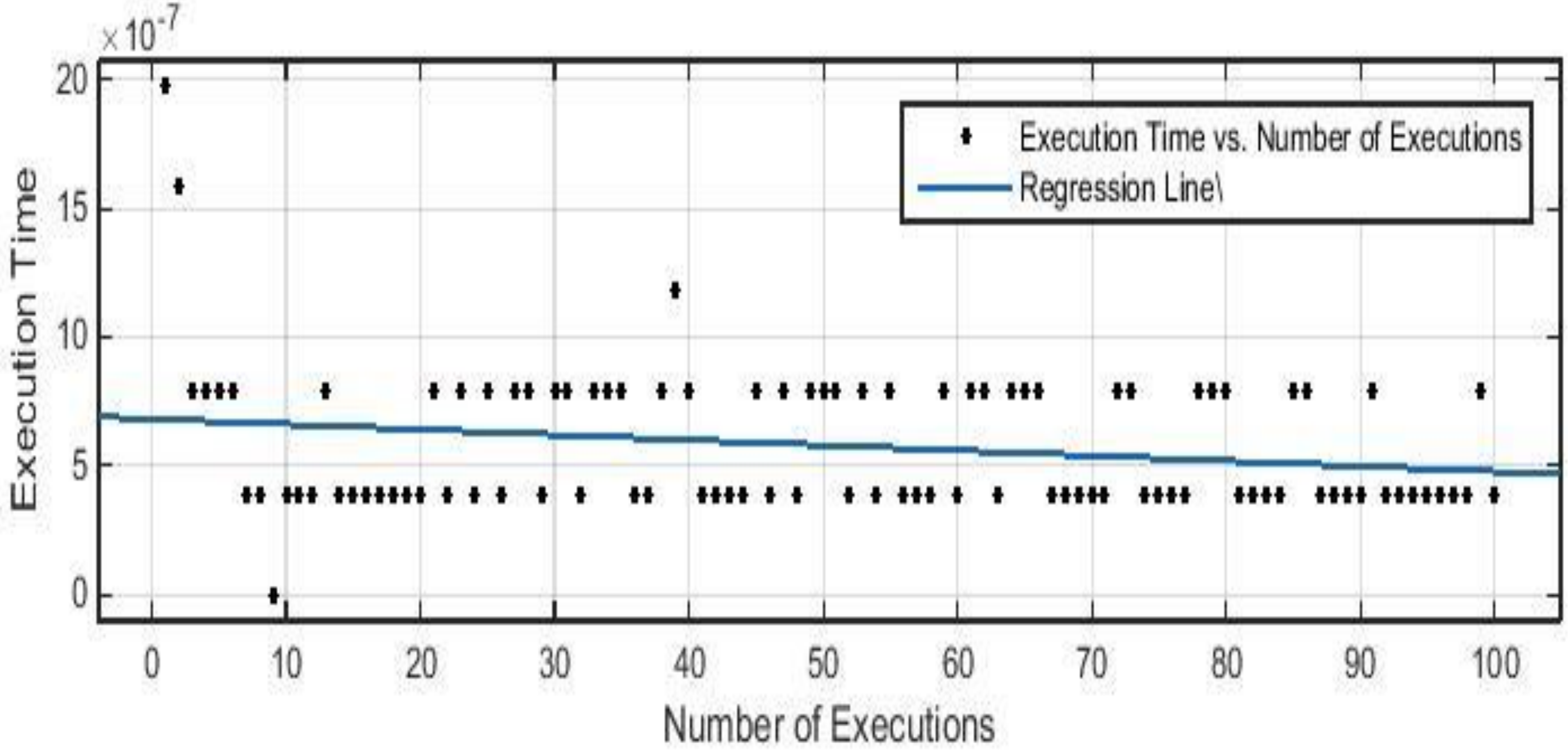
Unit of Measurement…(5)



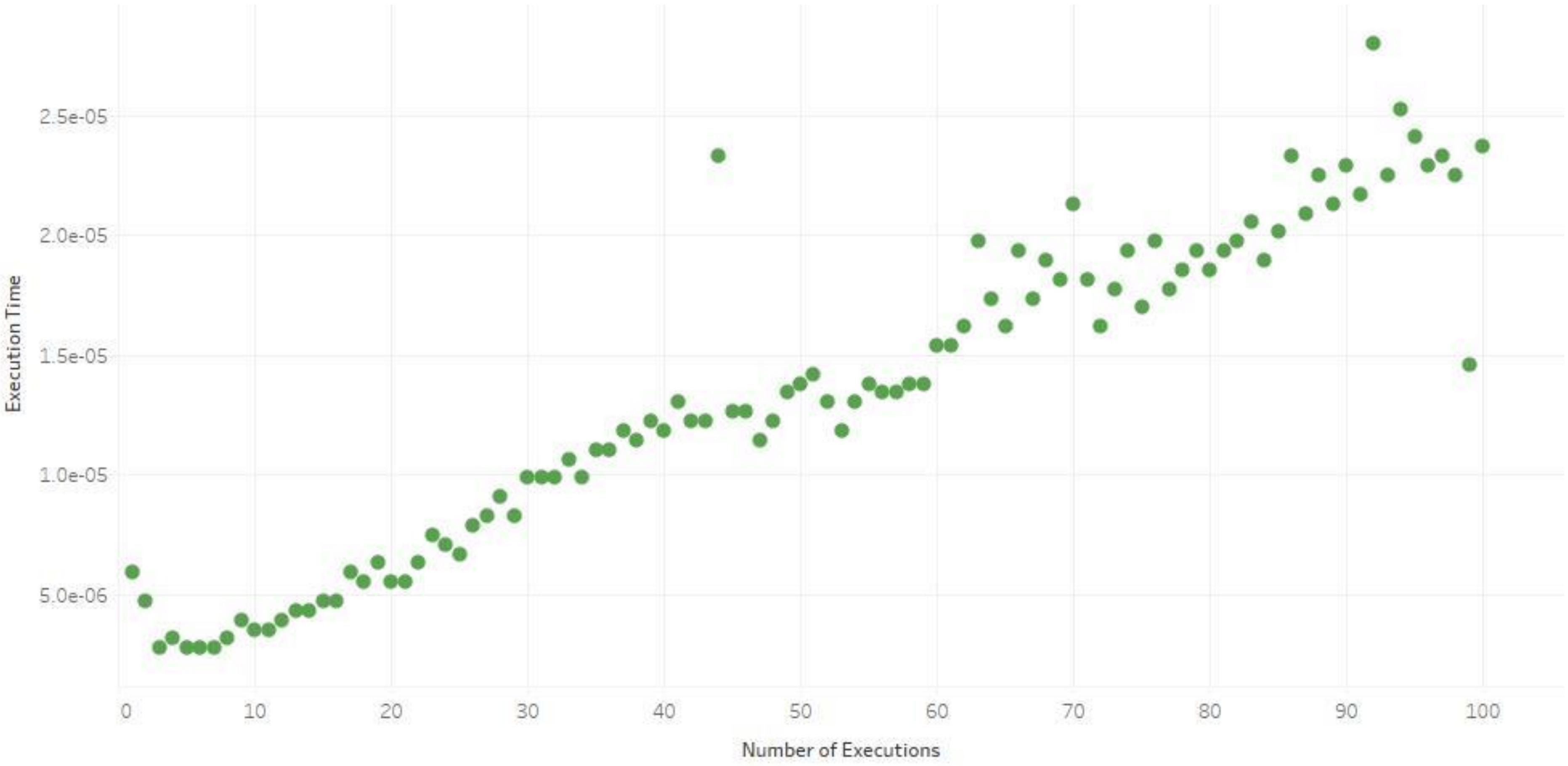
Unit of Measurement…(6)



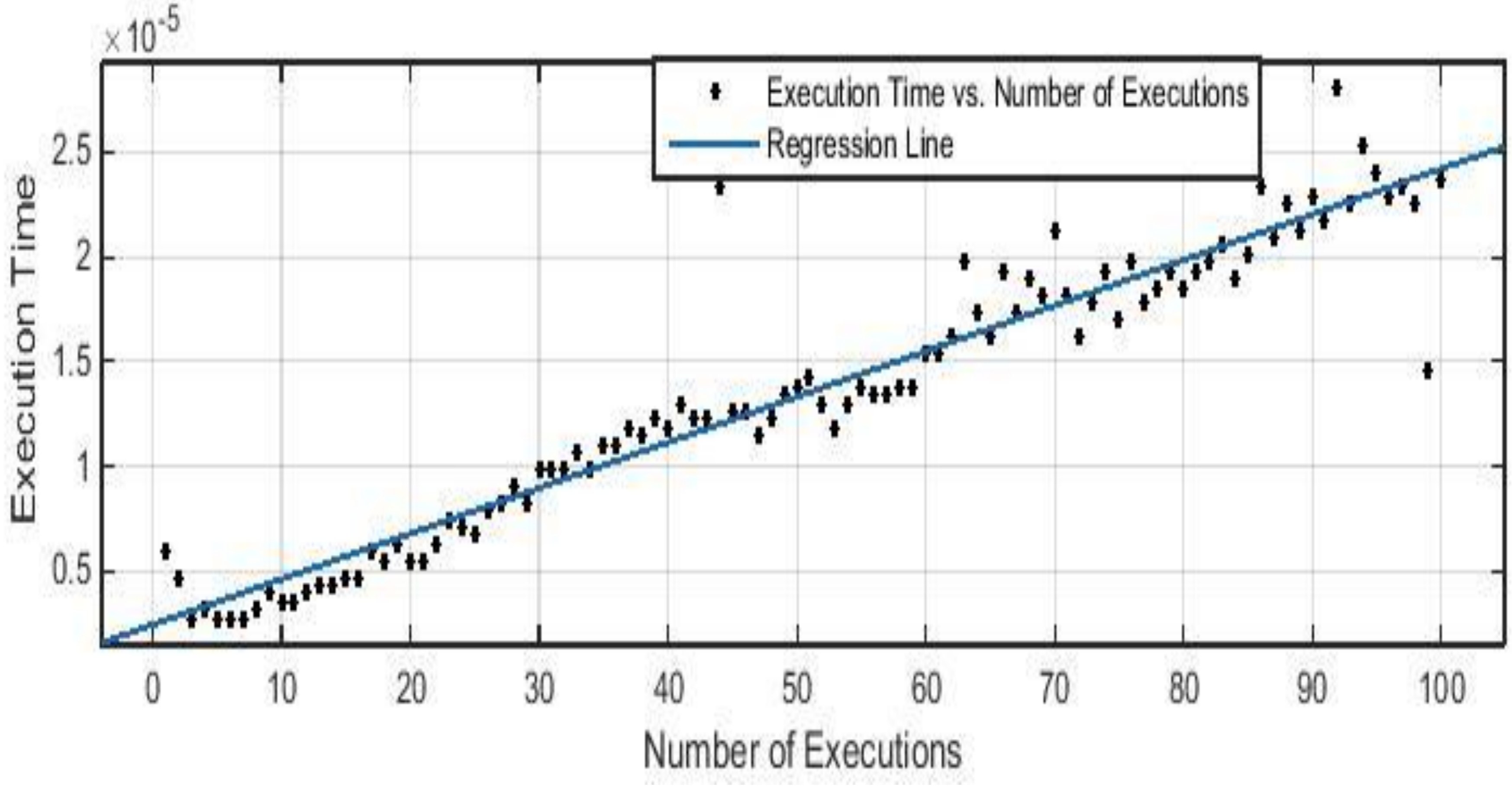
Unit of Measurement…(7)



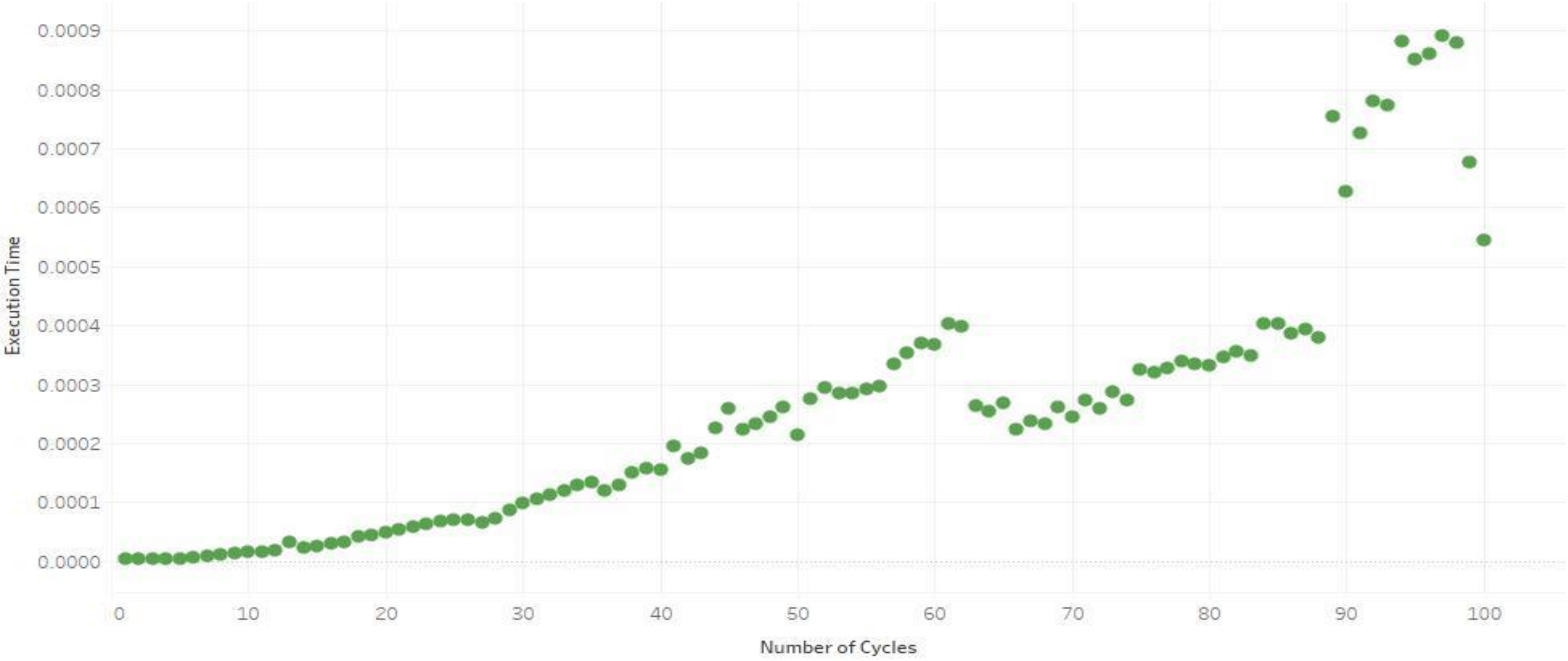
Unit of Measurement…(8)



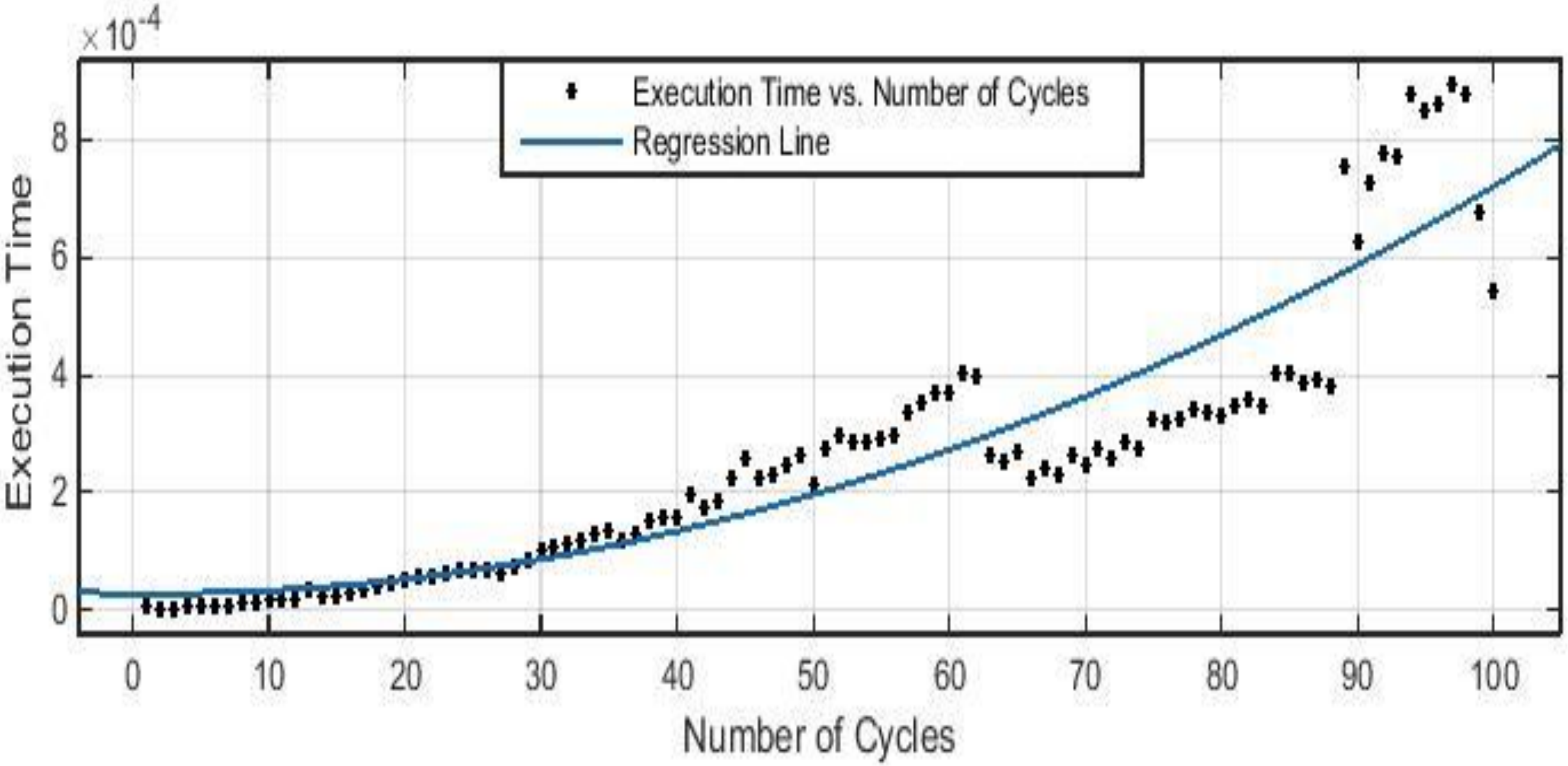
Unit of Measurement…(9)



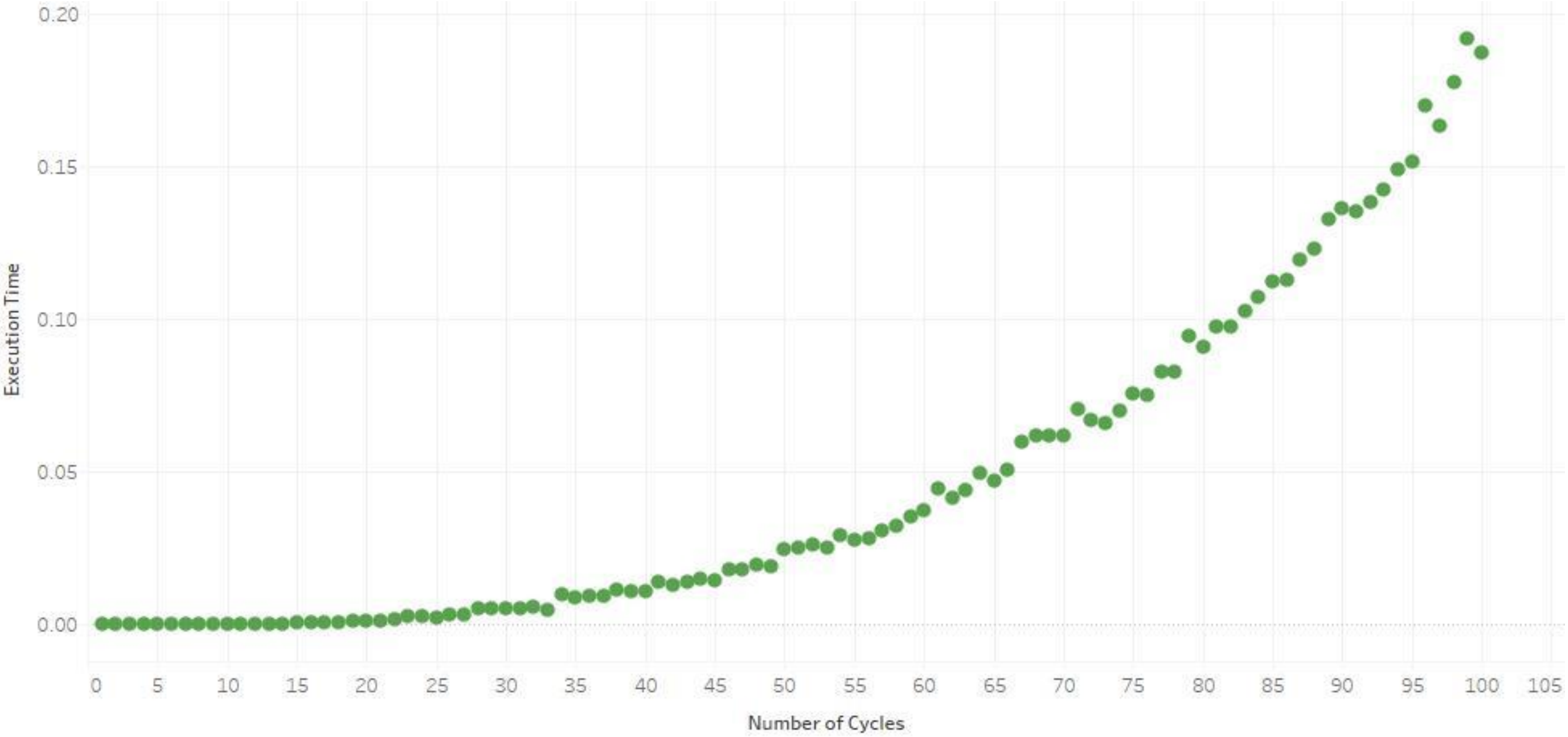
Unit of Measurement…(10)



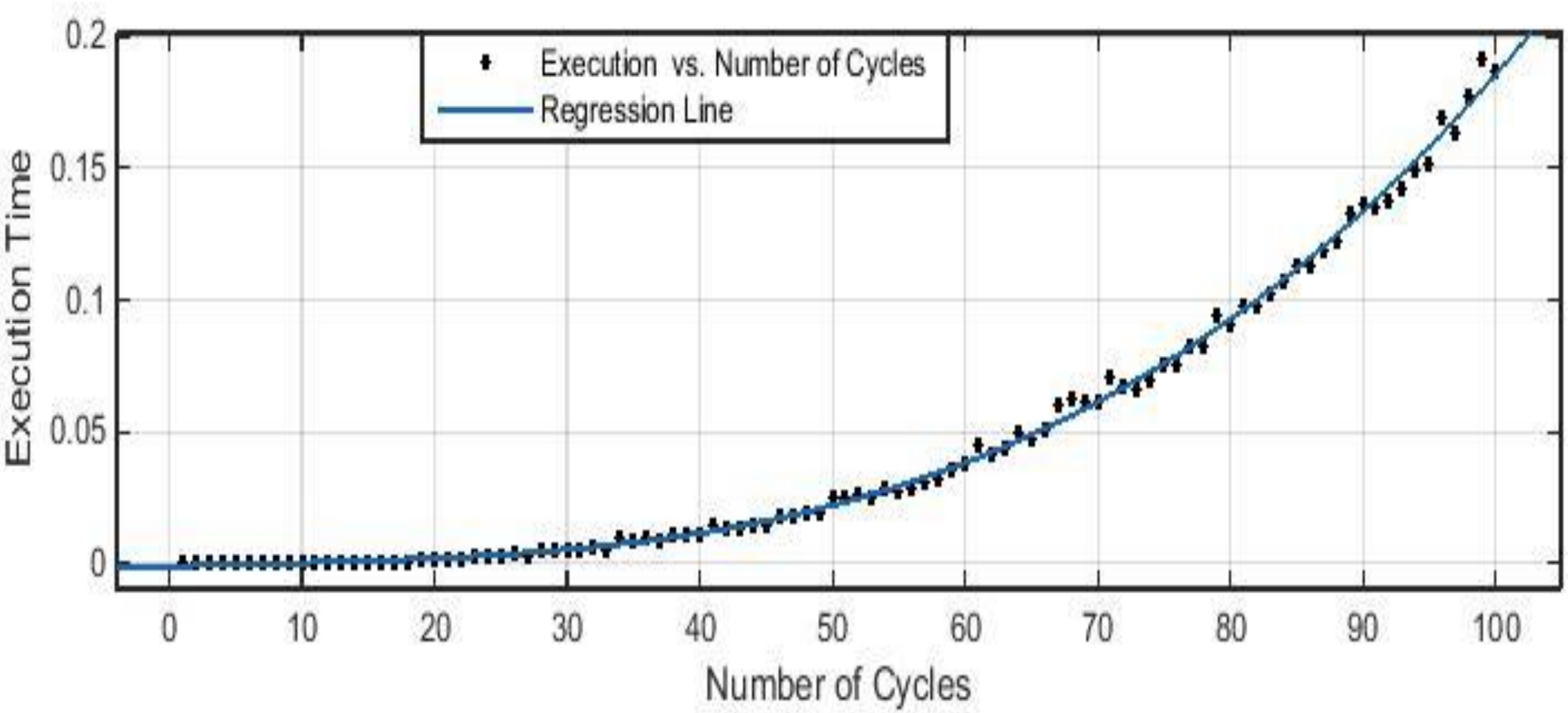
Unit of Measurement…(11)



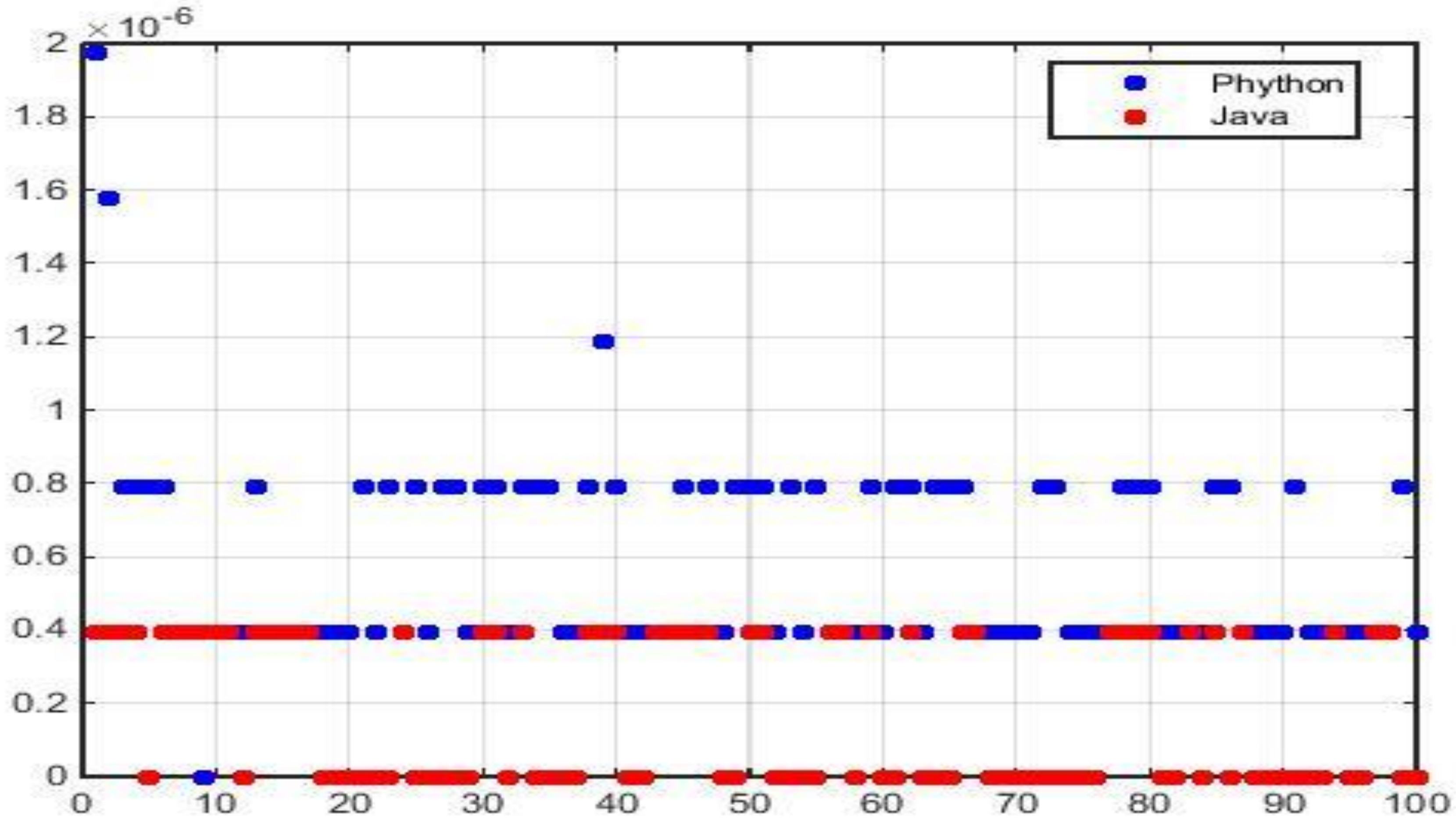
Unit of Measurement…(12)



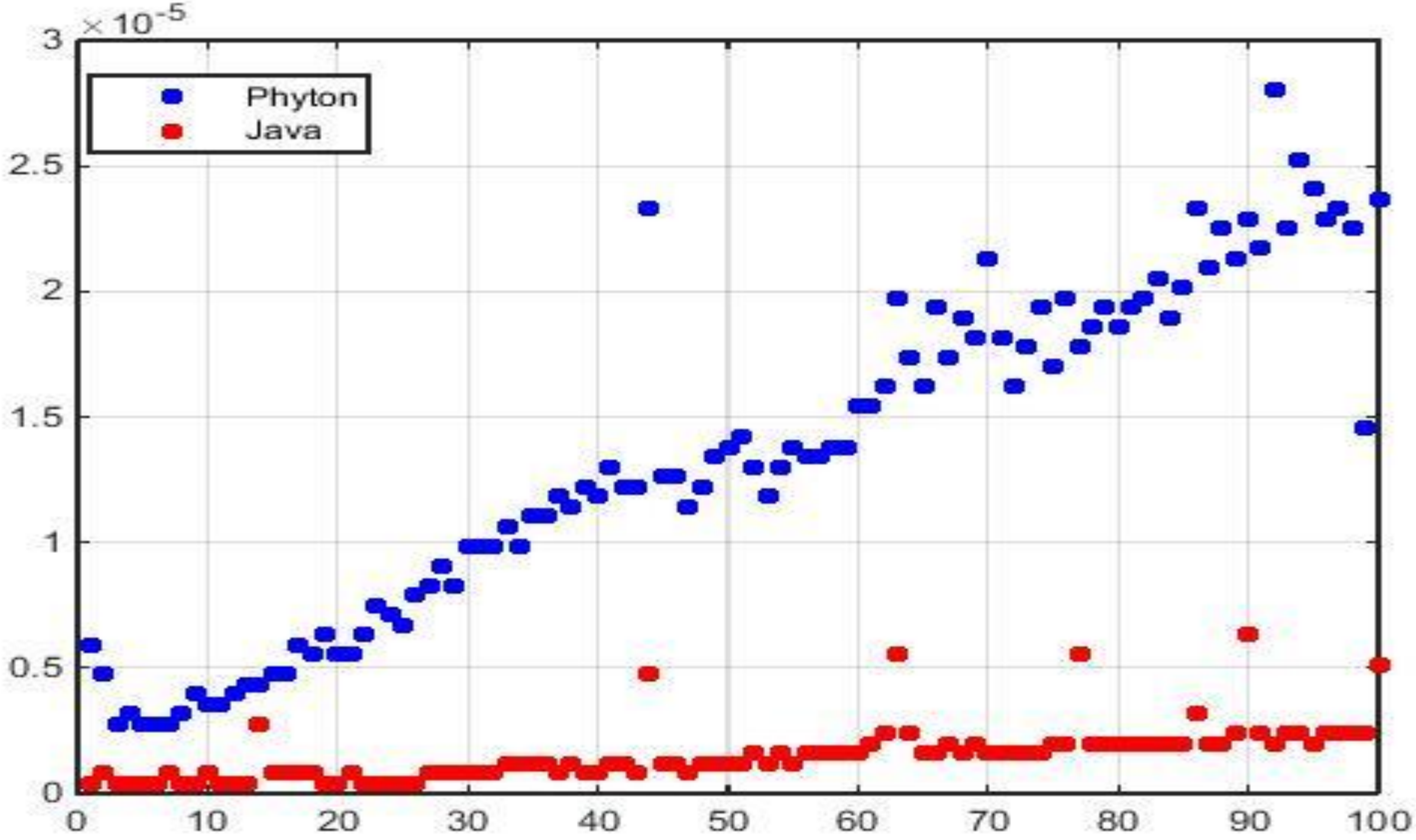
Unit of Measurement…(13)



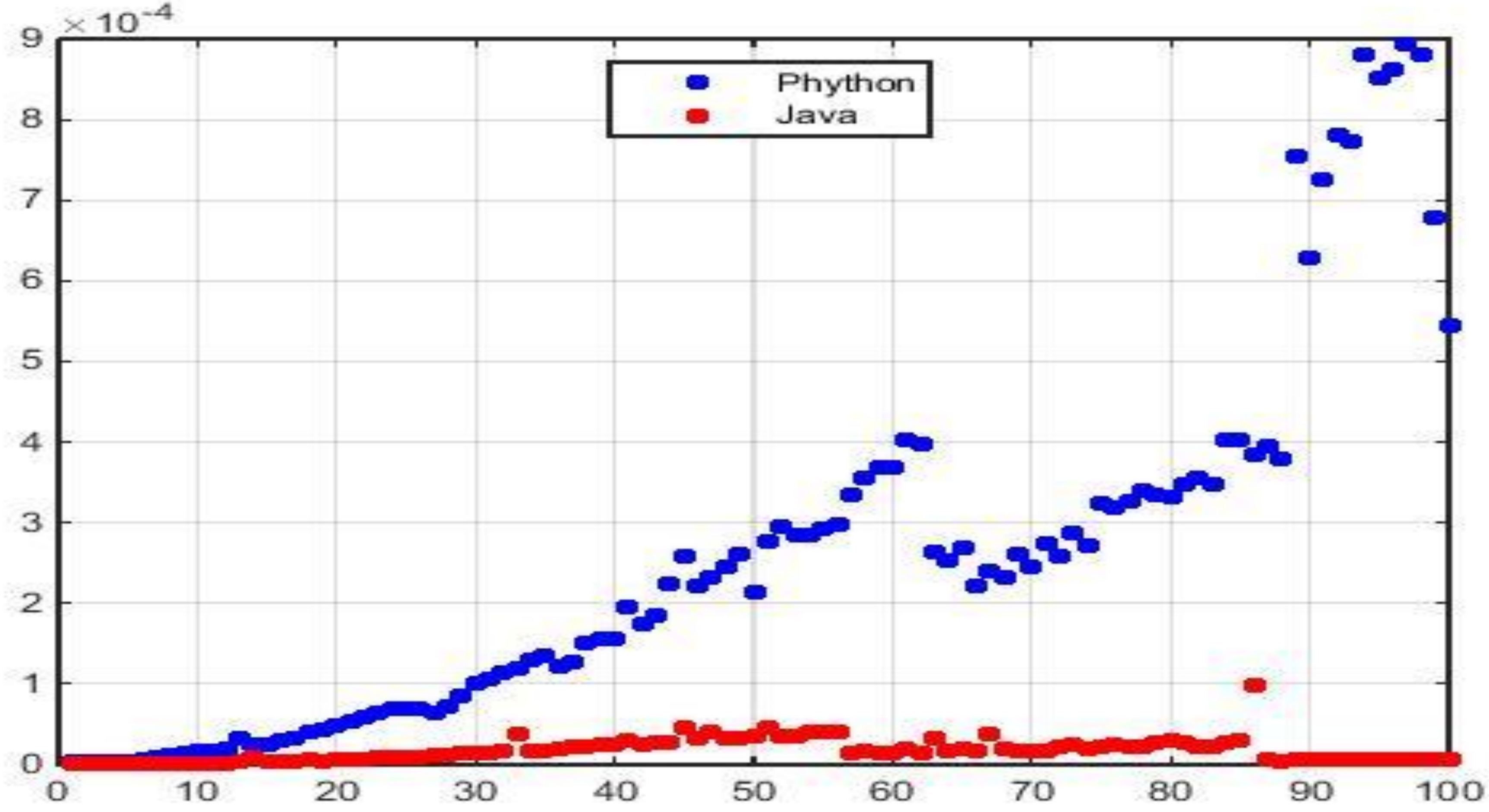
Unit of Measurement…(15)



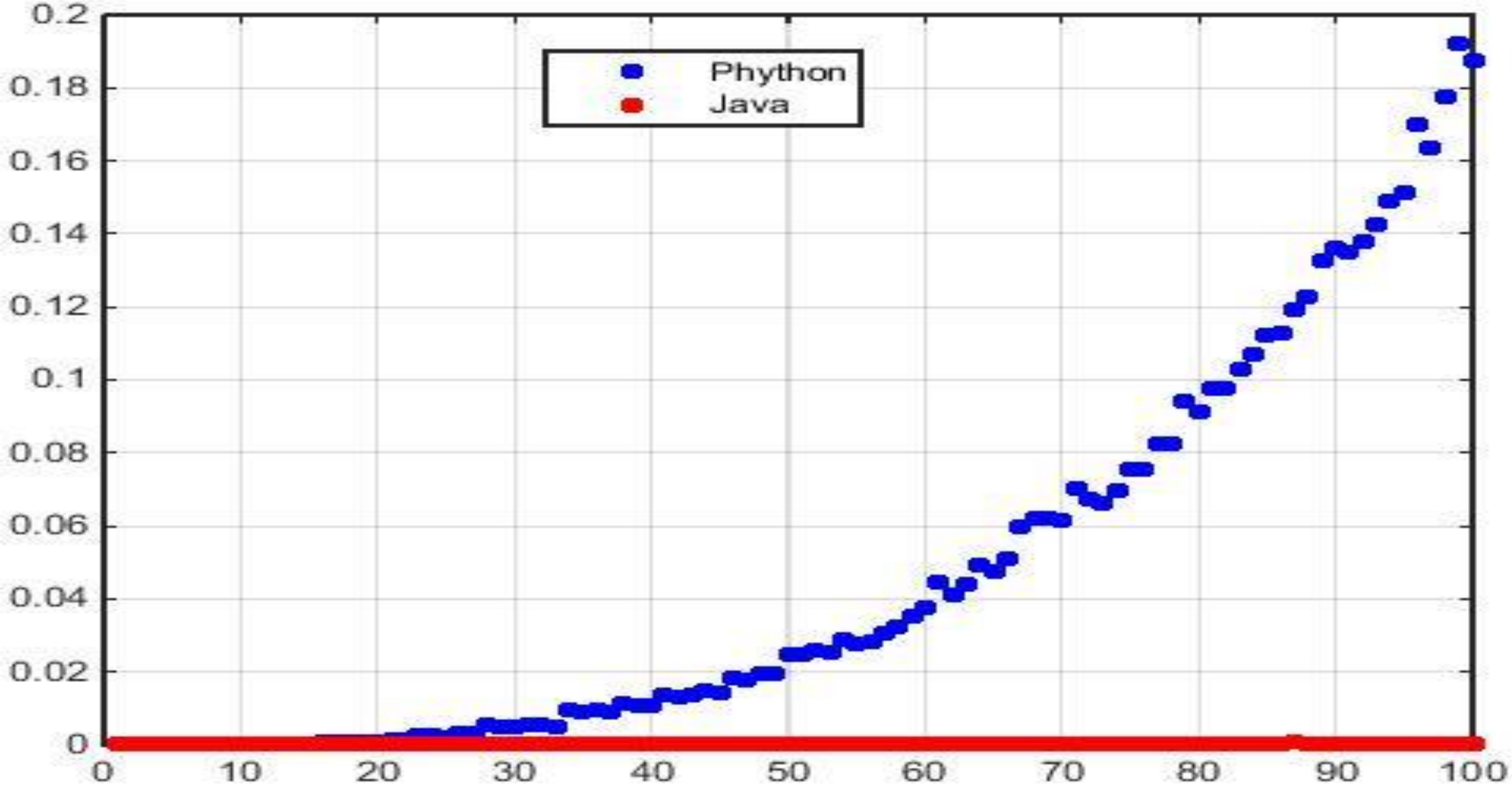
Unit of Measurement…(16)



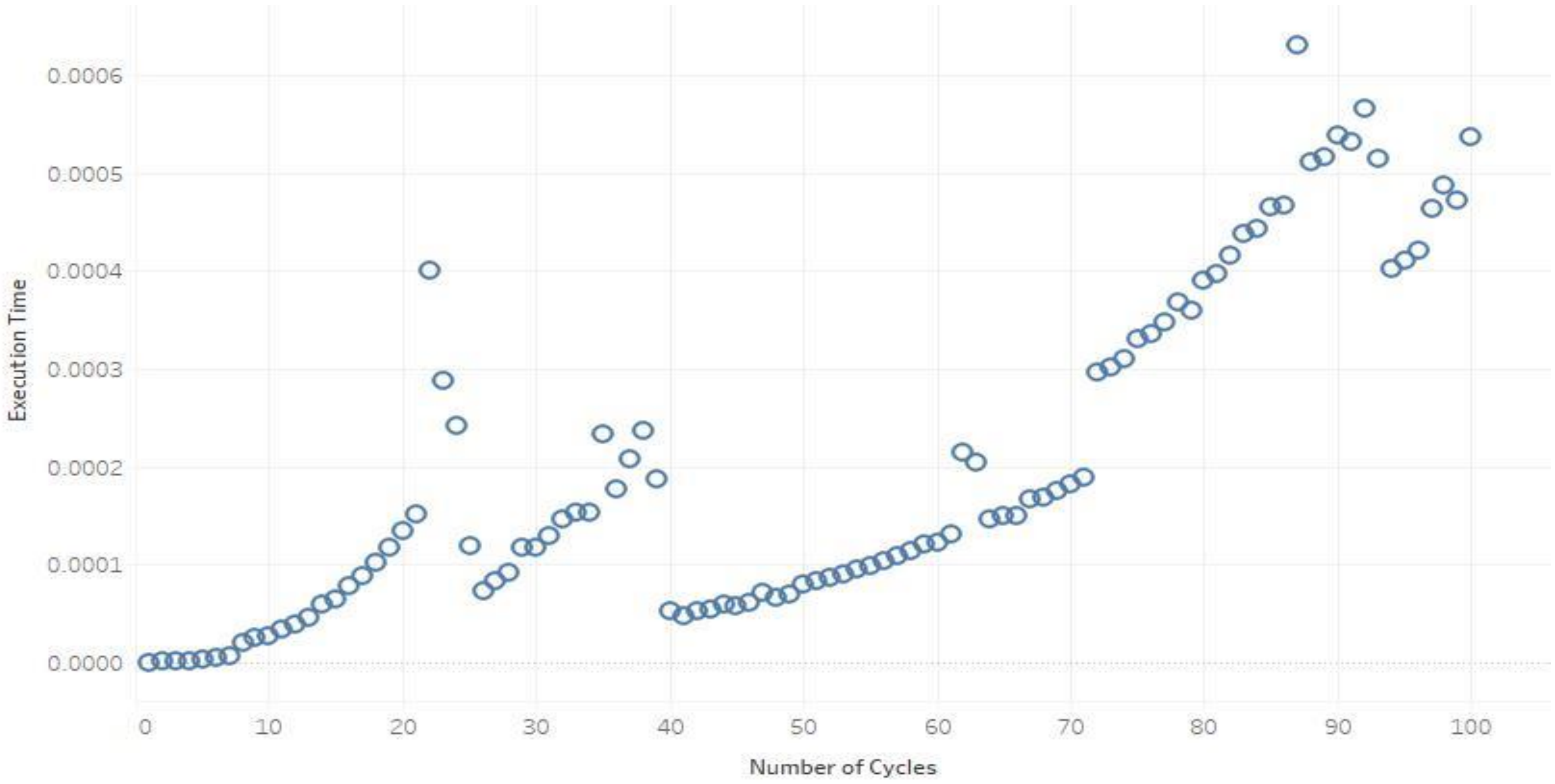
Unit of Measurement…(17)



Unit of Measurement…(18)



Unit of Measurement…(19)



Unit of Measurement…(20)

- What are the problems associated developing a measuring unit for computers?

- What are the properties that can be observed to measure the efficiency of the program?
  - What are the qualities of these properties?
  - Does the value really matter?

Order of Growth

- Order of growth represents how the computation time increases with the number of inputs.

- The order of growth is a rough estimate of the size of the input (file) and the time.
  - Exact order of growth is irrelevant and only a rough estimate is good enough.

- E.g.:- $f(n) = n^2 + 100n + 10^4$

Big-O Notation ...(1)

"$f(n)$ is $O(g(n))$ if there exist positive number c and N such that $f(n) \leq c.g(n)$ for all $n \geq N$."

- This is the most common notation used for estimating the rate of function growth.

Big-O Notation …(2)

- Big-O is inherently imprecise, hence the smallest possible function g(n) is selected.

- Big-O is transitive, if f(n) is O(g(n)) and g(n) is O(h(n)) then f(n) is O(h(n)).

- if f(n) is O(h(n)) and g(n) is O(h(n)) then f(n) + g(n) is O(h(n)).

- The function an^k is O(n^k).

Big-O Notation …(3)

- If f(n) = c.g(n) then f(n) is O(g(n)).

- The function $\log_a$ n is O($\log_b$ n)

- The Big-O notation describes the upper bound on the efficiency of the program.

Big-O Notation …(4)

- Find the Big-O or prove the following

1. $T(n) = a_k n^k + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + a_1 n + a_0$
2. $1000n^2 + 50n$ is $O(n^2)$
3. $g(n) = 2n^3 + 4n$ is not $O(n^2)$

Analysis of Algorithms…(1)

- Algorithms analysis is the process of estimating the efficiency (aka:- complexity) of a given computer program.

- The efficiency of a program can be computed for various aspects of a program. However, in most cases the fundamental interest is in the time complexity and to lesser extent memory complexity.

- In order to compute the time complexity of a program, the number of operations (e.g. assignments, comparisons etc.) are measured.

Analysis of Algorithms…(2)

- What is the complexity of the following pseudocode programs

1.)

```
        a = 5;
        b = 7;
        print(a*b);
```

Analysis of Algorithms…(3)

<u>2.)</u>

```
~~for i in range(100):~~
    a =  5;
    b = 7;
    print(a*b);
```

What would happen if the for loop is replaced with the following

```
N = input(‘Enter integer’);
for i in range(int(N)):
```

Analysis of Algorithms…(4)

3.)

```
N = input('Enter number: ')
j = int(N);
for i in range(j):
  for k in range(j):
    a = 5;
    b = 7;
    print(a*b);
```

Analysis of Algorithms…(5)

4.)

```
N = input('Enter number: ')
j = int(N);
for i in range(j):
    for k in range(j):
        for l in range(j):
            a = 5;
            b = 7;
            print(a*b);
```

Analysis of Algorithms…(6)

5.)

```
N = input('Enter number: ')
j = int(N);
for i in range(j):
    for k in range(j):
        a = 5;
        b = 7;
        print(a*b);
        break;
```

Analysis of Algorithms…(7)

6.)

```
iter = input('Enter iterations: ')
k = 0;
j = int(iter);
for i in range(j):
     for l in range(j):
          k +=1;
          print(k);
          if(i<j):
               break;
```

Analysis of Algorithms…(8)

7.)

```
for(int i = 1; i < n; i  = i*2)
{
    total = 5*6;
}
```

Analysis of Algorithms…(9)

8.)

```
int j = 0;
for(int i = 1; j <= n; i++)
{
    j = j + i;
}
```

Analysis of Algorithms…(10)

9.)

```
for(int i = 0; i < n; i++)
{
    for(int j = 1; j < n; j = j*2)
    {
        code;
    }
}
```

Analysis of Algorithms…(11)

10.)

```
for(int i = n; i > 1; i = i/2)
{
    code;
}
```

Questions?