Design Credit Project Presentation

Our Team

Anuj Rajan Lalla B22AI061

Suralkar Pranav Nanasaheb B22ME066

Vaibhav Gupta B22CS058

Problem Statement

Photon correlation for two qubit using 1 dimensional detector

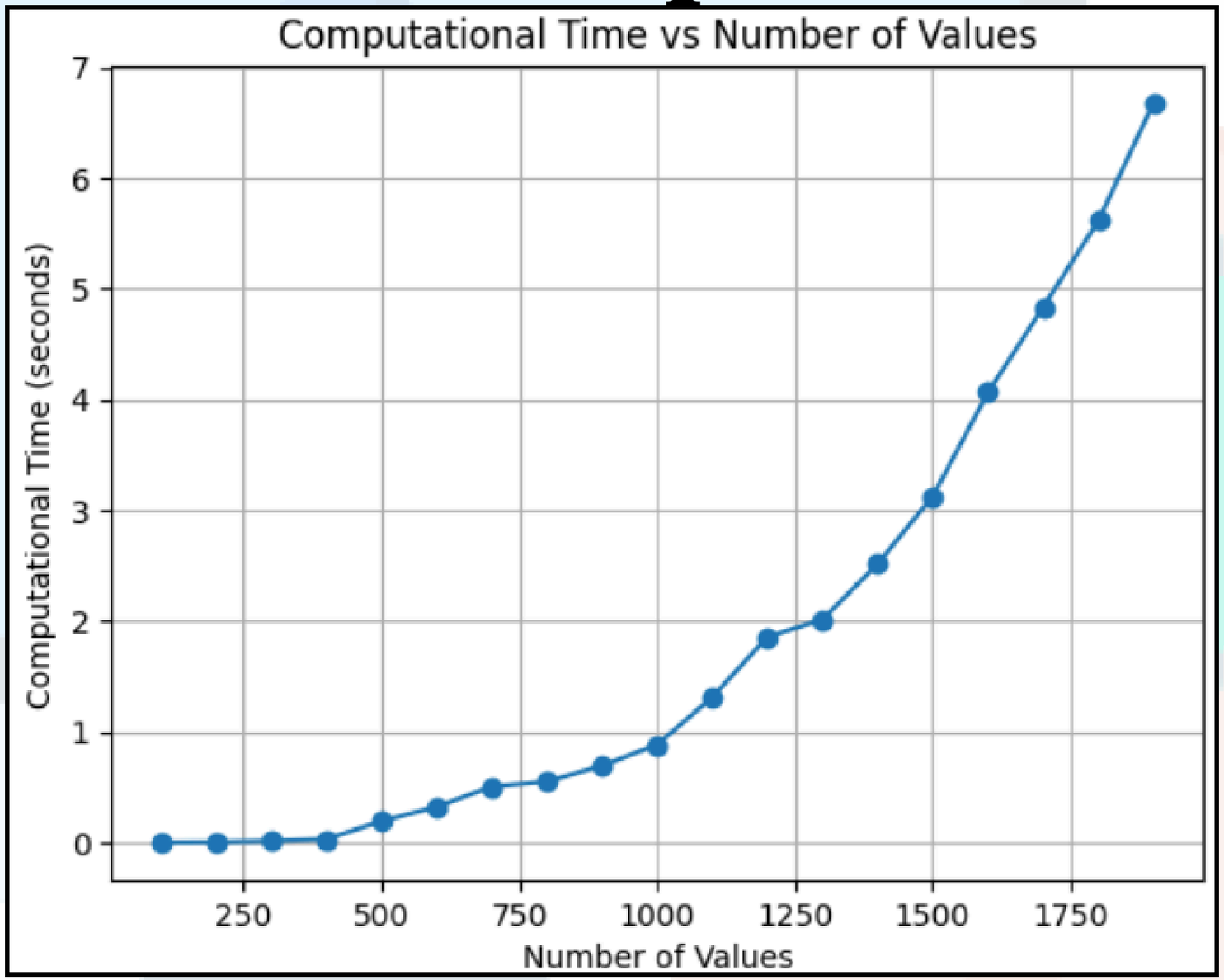
In simple words the problem was that there were 2 input vectors of the same dimension . We want to subtract the vectors element wise and rotate one vector and repeat this step till (size of vector - 1) times and concatenate this. So the dimensions of resultant vector would be (size of vector) x (size of vector)

Link to GitHub : https://github.com/anuj-l22/DC_Project

Inital approach

- **Equal Length Adjustment**: Ensures both input vectors are of equal length by padding the shorter one with zeros.
- **Vector Rotation and Subtraction**:
  - Rotates one vector left and the other right.
  - Subtracts these rotated vectors from each other after each rotation.
- **Repetitive Operation**: Continues rotating and subtracting until each vector has been rotated (number of elements - 1) times.
- **Result Concatenation**: Concatenates all subtraction results into one long vector.
- **Purpose**: Aims to analyze how the computational time is affected by the complexity of these manipulations as the vector size increases.
- **Visualization**: Plots computational times against the number of vector elements to visualize performance trends.

Graph


Computational Time vs Number of Values
Computational Time (seconds)
0
1
2
3
4
5
6
7
250
500
750
1000
1250
1500
1750
Number of Values

Drawbacks of this code

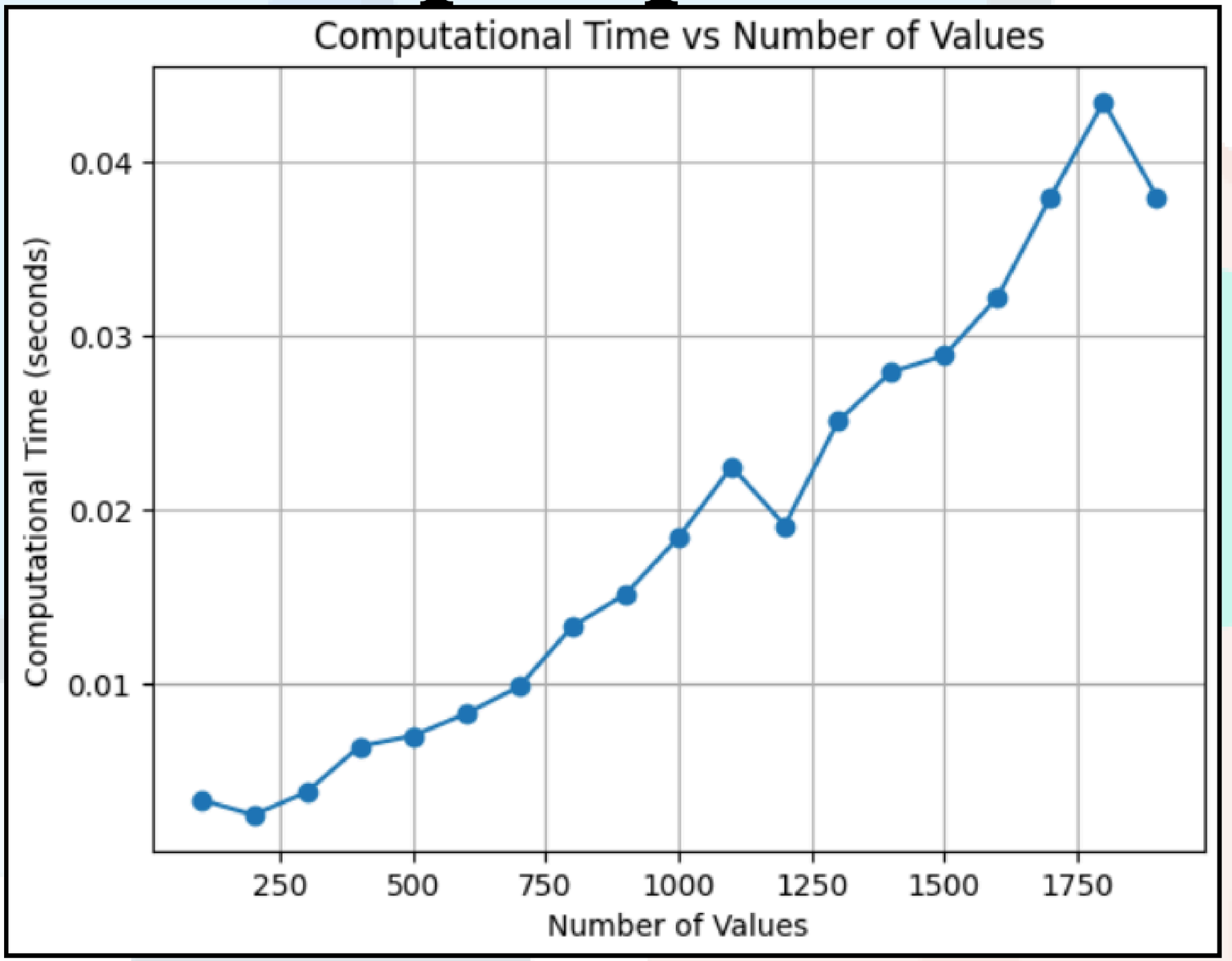
1. **Quadratic Complexity**: The shape of the curve suggests that the computational time increases quadratically with the number of values. Given the operations in the code—specifically, the repeated rotations and subtractions for each element—the computational complexity is likely exponential

2. **Scalability Issues**: The steep increase in computational time as the vector size increases indicates poor scalability of the code. For very large vector sizes, the performance would degrade, making the approach impractical for large-scale applications.

Updated approach

- Chunked Processing: Utilizes smaller batches (chunk_size) for data handling, enhancing memory management and reducing overhead.
- Linear Time Complexity: Achieves linear time complexity, improving consistency and performance compared to the quadratic complexity of the previous version.
- Memory Optimization: Employs float32 data types, reducing the memory footprint significantly.
- Scalability: Improved scalability with increasing data sizes due to more efficient processing methods.
- Data Sorting and Exporting: Final results are sorted in ascending order and saved to a CSV file

Graph updated



Results obtained

The time taken has been reduced significantly in the updated approach. The time taken for plain computation for result vector for the maximum number of elements is 118921 is 3 minutes
But the overhead is to convert the result_vector obtained to csv file . It takes a lot of time
Overall the main overhead of time has been optimized , and even memory has improved

Thank You