Kwame Nkrumah University of Science and Technology

College of Science
Faculty of Physical and Computational Sciences
Department of Mathematics

Project Report - Part 1

MATH 458 - Scientific Computing II

August 12, 2024

Tei Mensah Andrews - 4170120

1 Introduction

1.1 Overview

This report discusses the results, methodology and implementation of the work done in serial programming project for Scientific Computing II using C Programming. The project entails reviewing algorithms that can be used to find the result of a vector-vector multiplication, vector -matrix multiplication and a matrix-matrix multiplication for dense matrices and vectors and implementing them using C programming. Since the *** for these algorithms for dense matrices and vectors involves involves alot of iterative computations, these processes are timed for varying number of rows and columns of a matrix and their results are plotted for analysis.

1.2 Concepts

Vector-vector Multiplication Otherwise know as dot product is the multiplication of two vectors and results in a scaler number that gives details about the behaviour of two vector towards each(eg. the angle between them). Its results is found by pairwise multiplication of elements of both vectors and subsequently adding them.

Matrix-vector Multiplication It is the multiplication of a matrix and a vector and results in a vector of size that is equal to the number of columns of the matrix or the number of elements of the vector. Its results is found by pairwise multiplication of elements of the vectors and column on i-th row and subsequently adding the to be i-th element in the result vector.

Matrix-Matrix Multiplication It is the multiplication of two matrice to form a result matrix which has the same number of rows as the first matrix and the same nuber of columns as the second matrix. To be able to multiply two matrices the number of columns of the first matrix should equal the nuber of rows of the second matrix.

2 Algorithms

2.1 vector-vector multiplication

Mathematical Description

Input data: Two one-dimensional numeric arrays of equal size.

Output data: A single numeric value of the sum of the pairwise products of corresponding elements of both arrays.

For two one-dimensional arrays A[i] and B[i] each of size n, the dot product can be expressed mathematically as:

$$A \cdot B = \sum_{i=1}^{n} A[i]B[i] \tag{1}$$

Complexity

The algorithm has linear complexity, That is $O(N)$ for arrays of length N.

Algorithm 1 Vector-vector Multiplication

Data: $A[N],\ B[N]$
Result: $R \leftarrow 0$
for $i = 0;\ i < N;\ i++$ **do**
    $R \leftarrow R + (A[i] * B[i])$
end for

2.2 Matrix-Vector Multiplication

Mathematical Description

Input data: A two-dimensional array of a size M $\times$ N and a one-dimensional array of size N.

Output data: A one-dimensional array of size.

Given a two-dimensional array A[i][j] of size m $\times$ n, a one-dimensional array B[j] of size n and a one-dimensional results array C[i] the matrix-vector multiplication can be expressed mathematically as:

$$C[i] = \sum_{i=1}^{n} A[i][j]\, B[j], \quad \forall\ i \in [1, m] \tag{2}$$

Complexity

The algorithm has quadratic complexity, that is $O(N^2)$ for arrays of length N.

Algorithm 2 Matrix-vector Multiplication

Data: $A[M][N],\ B[N]$
Result: $C[N]$
for $i = 0;\ i < M;\ i++$ **do**
  $C[i] \leftarrow 0$
  for $j = 0;\ j < N;\ i++$ **do**
    $C[i] \leftarrow C[i] + (A[i][j] * B[j])$
  end for
end for

2.3 Matrix-Matrix Multiplication

Mathematical Description

Input data: Two two-dimensional array of one of size M × N and the other of size N × P.

Output data: A two-dimensional array of size M × P.

Given two two-dimensional array A[i][j] of size m × n, B[i][j] of size n × p and a two-dimensional results array C[i][j] of size m × p the matrix-vector multiplication can be expressed mathematically as:

$$C[i][j] = \sum_{k=1}^{n} A[i][j]\, B[j][i], \quad \forall\ i \in [1, m] \tag{3}$$

Complexity

The algorithm has cubic complexity, that is $O(N^3)$ for arrays of length N.

Algorithm 3 Matrix-vector Multiplication

Data: $A[M][N],\ B[N][P]$
Result: $C[M][P]$
for $i = 0;\ i < M;\ i++$ **do**
  for $j = 0;\ j < P;\ j++$ **do**
    $C[i][j] = 0$
    for $k = 0;\ k < N;\ k++$ **do**
      $C[i][j] \leftarrow C[i][j] + (A[i][k] * B[k][j])$
    end for
  end for
end for

---

3 Implementation

Vector-vector Multiplication

The following is the code for a C Programming implementation for a function that does vector-vector multiplication.

```
double vector_vector (int m, int n, double *vecA, double *vecB){
        double results = 0;
        for (int i = 0; i < m; i++){
                results += (vecA[i]*vecB[i]);
        }
        return results;
}
```

Matrix-Vector Multiplication

The following is the code for a C Programming implementation for a function that does Matrix-vector multiplication.

```
void matrix_vector(int m, int n, double *matA, double *vecB, double *matC){
        for (int i = 0; i < m; i++){
                for (int j = 0; j < n; j++){
                        matC[i] += matA[i*n+j]*vecB[j];
                }
        }
}
```

Matrix-Matrix Multiplication

The following is a code for a C Programming implementation for a function that does Matrix-matrix multiplication.

```
void matrix_matrix(int m, int n, double *matA, double *matB, double *matC){
        for (int i = 0; i < m; i++){
                for (int j = 0; j < n; j++){
                        for (int k = 0; k < m; k++){
                                matC[(i*n)+j] += matA[(i*n)+k]*matB[(k*n)+j];
                        }
                }
        }
}
```

4 Test and Results

Arbitrary vectors and square matrices were fed to the code and the run time for varying number of rows and columns. From table 1 we can see that as the number of N increase the run increases linearly for the vector-vector multiplication, quadratically for the matrix-vector multiplication and cubically for the matrix-matrix multiplication which confirms their complexity.

N	Core	V.V/s	M.V/s	M.M/s
1000	1	0.000012	0.008500	9.687999
2000	1	0.000016	0.019900	106.743021
4000	1	0.000018	0.053481	565.105126
5000	1	0.000018	0.079720	1126.728014

Table 1: Run time for processes with different N values

5 Analysis

- Vector-vector multiplication: The computational time for its algorithm for dense matrix is relatively low, hence serial computation is suitable for its algorithm.

- Matrix-vector multiplication: Run time for its algorithm is moderate and lies between those of matrix-matrix and vector-vector. For really dense matrix and vector a serial implementation of the algorithm may be insufficient.

- Matrix-matrix multiplication: Run time for its algorithm increases very quickly for increase size of matrices. Hence for dense matrices parallel implementation for its algorithm is recommended.

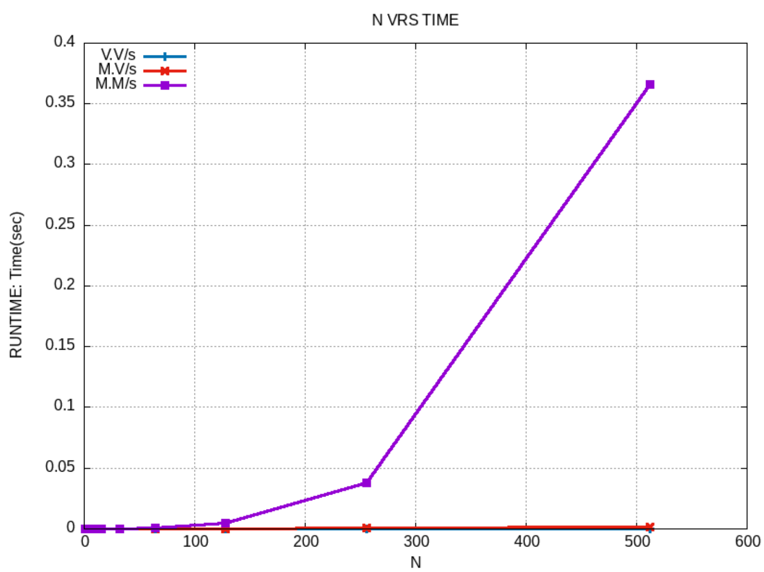

Figure 1: Number of row/col(N) Against Time(sec)

6 Conclusion

The presented run time of the algorithm for Vector-vector, Matrix-vector and Matrix-matrix multilocation shows their efficient in a serial implementation. The results show that the algorithm for matrix-matrix multiplication exhibits a sharp increase in run time for increasing density of matrix hence parallel implementation is best suited for it.