Matrix Multiplication Using Strassen's Method

Yabaluru Deepthi Naga Ramya Sravani[1], Syed Auron Meera[2]

[1,2] *Undergraduate, Dept. Of Information Technology, Vignan's Lara Institute of Technology and Science, Andhra Pradesh, India*

Abstract - *Matrix can be multiplied in different ways by using different algorithms and gives same result. Matrices are frequently used while doing programming in solving system of linear equations and many more. Lot of research is being done on how to multiply matrices using minimum of operations. After the research, Strassen's algorithm takes less time for the execution. Using this algorithm for n × n matrix multiplication are investigated and compared with the normal algorithm.*

In this paper, it throws insight on the matrix multiplication in C language and also implements an algorithm to reduce the time complexity.

***Key Words*: Multiplication Algorithms, Same Result, Programming Structure, Minimum Operations, Reduce Time Complexity.**

1. INTRODUCTION

A matrix is an arrangement of items into set of rows and columns into the table. Matrix can be represented in a 2-Dimensional array. Operations that perform on matrix are addition, subtraction, multiplication, transpose etc.

Matrix also contains important applications in every field and frequently for large amounts of computer time. An algorithm is an unambiguous specification of how to solve a problem, based on a sequence of specified conditions. For the same problem different types of instructions gives same result even though the time of execution may differ. Some may execute fast, some may be slow, other may occupy the more space etc. Having less execution time and space are the best one.

Small algorithms require a single processor to perform a task within the record time in an efficient way. But single processor is not efficient in all cases. So in complex situations, task will be divided into different number of processors to solve in an effective manner.

In scientific computing the most prevalent operations are done in matrix multiplication. Matrix multiplication can be done between two matrices. To apply an multiplication method, it should satisfy one condition that is, if and only if" the number of columns of first matrix should be equal to the number of rows in the second matrix" Only in this condition the matrix can be multiplied. In other case ,multiplication is not possible.

For Example:

If matrix A[m][n] has, m rows and n columns , the matrix B[p][q] should have its row value q=n. and this matrix multiplication will generate a new matrix C[m][q].

where,
m= number of rows in first matrix and
q= number of columns in second matrix

The following program displays the error until the number of columns of first matrix is equal to the number of rows of second matrix.

1.1 C Program On Matrix Multiplication:

```
#include<stdio.h>
int main()
{
  int a[10][10],b[10][10],
  c[10][10],r1,c1,r2,c2,i,j,k;
 printf("Enter the rows and column for first matrix:");
scanf("%d%d",&r1,&c1);
printf("/nEnter the rows and column for second matrix:");
scanf("%d%d",&r2,&c2);
// Column of first matrix should be equal to column of second matrix
while(c1!=r2)
{
 printf("Error! column of first matrix not equal to row of second.\n\n");
printf("Enter rows and column for first matrix:");
 scanf("%d %d", &r1, &c1);
 printf("Enter rows and column for second matrix: ");
 scanf("%d %d",&r2, &c2);
}
  // Storing elements of first matrix.
  printf("\nEnter elements of matrix 1:\n");
  for(i=0; i<r1; ++i)
    for(j=0; j<c1; ++j)
    {
      printf("Enter elements a%d%d: ",i+1, j+1);
      scanf("%d", &a[i][j]);
    }
  // Storing elements of second matrix.
  printf("\nEnter elements of matrix 2:\n");
  for(i=0; i<r2; ++i)
    for(j=0; j<c2; ++j)
    {
```

```
        printf("Enter elements b%d%d: ",i+1, j+1);
        scanf("%d",&b[i][j]);
      }
// Initializing all elements of result matrix to 0
      for(i=0; i<r1; ++i)
        for(j=0; j<c2; ++j)
        {
         result[i][j] = 0;
        }
   // Multiplying matrices a and b & storing result
   for(i=0; i<r1; ++i)
     for(j=0; j<c2; ++j)
        for(k=0; k<c1; ++k)
        {
           result[i][j]+=a[i][k]*b[k][j];
        }
    // Displaying the result
   printf("\nOutput Matrix:\n");
   for(i=0; i<r1; ++i)
     for(j=0; j<c2; ++j)
    {
        printf("%d  ", result[i][j]);
        if(j == c2-1)
           printf("\n\n");
     }
return 0;
}
```

Output:

run

input output

```
try@programiz:/home/try$ gcc -w df11f.c -o df11f -lm && df11f || echo
Enter rows and column for first matrix: 3 2
Enter rows and column for second matrix: 3 2
Error! column of first matrix not equal to row of second.

Enter rows and column for first matrix: 2 3
Enter rows and column for second matrix: 3 2

Enter elements of matrix 1:
Enter elements a11: 3
Enter elements a12: -2
Enter elements a13: 5
Enter elements a21: 3
Enter elements a22: 0
Enter elements a23: 4

Enter elements of matrix 2:
```

Fig (a): The output shows an error when column of first matrix is not equal to row of second,for first input matrix and second input matrix is given as per the rule.

run

input output

```
Enter elements a23: 4

Enter elements of matrix 2:
Enter elements b11: 2
Enter elements b12: 3
Enter elements b21: -9
Enter elements b22: 0
Enter elements b31: 0
Enter elements b32: 4

Output Matrix:
24  29

6  25

try@programiz:/home/try$ exit
```

Fig (b): Matrix Multiplication is done and it shows in output matrix.

[A] Related Works:

The related works of matrix multiplication is done briefly. In[2], the author proposed a new parallel implementation of Strassen's matrix multiplication algorithm. This proposed algorithm has a special conflict-free routing pattern for better scalability. And finally, they achieved the speedup result as nearly 21.7%.

In[3], the authors analyzed the scalability of number of parallel formulations of the matrix multiplication algorithm and the performance and predict the conditions under which each formulation is better than others.

In[4], the author tells about the main factors of determining an algorithm how much time it takes to complete its work. Matrix multiplication is one of the core components of many scientific computations. So to find the best method of matrix multiplication, the execution time of all methods will be calculated.

In[5], the author explained about the Strassen's algorithm for matrix multiplication to reduce the complexity. To implement the algorithm efficiently that makes it a challenging task which is done in the hierarchical memory system.

In[6], the authors on configurable hardware they develop new architectures and algorithms for matrix multiplications. These algorithms improve the previous design in terms of area.

In[7], the author discuss an interesting activity which involves in the problem in an matrix multiplication. They are well aware of increasing importance of matrix applications to several day-to-day real life problems.

2.Why Strassen's Method?

Matrix multiplication or the matrix product in a binary operation that produces a matrix from two matrices. So upto now, you have seen the code for matrix multiplication and the brief introduction of algorithms. It is easy to write logic of multiplication as we have seen before in the C program. But the time complexity is more in the normal code. Matrix multiplication contain many algorithms to perform the task. But the main aim is to choose the best algorithm which can reduce the time complexity. So, Strassen's method is the best method to implement for this purpose.

Generally, in the standard matrix multiplication the time complexity is $O(n^3)$ operations. When we apply the Strassen's algorithm the time complexity is $O(n^{2.81})$. This is the main feature and advantage of this algorithm.

2.2 Strassen's Algorithm:

The Strassen's method of matrix multiplication is a typical divide and conquer algorithm. We have seen so far some divide and conquer method as merge sort, karatsuba's fast multiplication of large numbers. This method is introduced to reduce the complexity. Without communications the addition and subtraction of matrices can be computed in linear time. So, this method is better than the standard matrix multiplication. However, let's again go behind the divide and conquer approach.

Generally, a product of 2x2 matrices requires 8 multiplications to get resultant matrix. But the main scheme of this method is, it reduces to 7 multiplications.

Example:

Basic multiplication method,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} X \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

A B C

Using Strassen's method, the following are the 7 operations or formula's for multiplication.

$$p1 = a(f - h) \qquad p2 = (a + b)h$$
$$p3 = (c + d)e \qquad p4 = d(g - e)$$
$$p5 = (a + d)(e + h) \qquad p6 = (b - d)(g + h)$$
$$p7 = (a - c)(e + f)$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} X \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} p5 + p4 - p2 + p6 & p1 + p2 \\ p3 + p4 & p1 + p5 - p3 - p7 \end{bmatrix}$$

A B C

Resultant matrix

Code:

```
p1= A[0][0]*(B[0][1]-B[1][1]);
p2= (A[0][0]+A[0][1])*B[1][1];
p3= (A[1][0]+A[1][1])*B[0][0];
p4= A[1][1]*(B[1][0]-B[0][0]);
p5= (A[0][0] + A[1][1])*(B[0][0]+B[1][1]);
p6= (A[0][1]-A[1][1])*(B[1][0]+B[1][1]);
p7= (A[1][0]-A[0][0])*(B[0][0]+B[0][1]);

C[0][0]=p5+p4-p2+p6;
C[0][1]=p1+p2;
C[1][0]=p3+p4;
C[1][1]=p1+p5-p3-p7;
```

This is the main logic of Strassen's multiplication.

3. CONCLUSIONS

1. To write a program on any concept or to solve any problem, then try to solve it with less complexity.
2. After some researches we found that Strassen's method is the best and the fast method for multiplication.
3. Approximately, the time complexity of this algorithm is $O(n^{2.81})$.
4. But it would be slower than the fastest known algorithms 'for extremely large matrices'.

REFERENCES

[1]. V. Yegnanarayanan (2013)" An Application of Matrix Multiplication".Indian Academy of Sciences.

[2]. Ohtaki , Y .(2004)." Parallel Implementation of Strassen's Matrix Multiplication Algorithm for Heterogeneous Clusters". IEEE. 18th International Parallel and Distributed Processing Symposium (IPDPS'04).

[3]. Gupta ,A and ICumar ,V.(1993)."Scalability of Parallel Algorithms for Matrix Multiplication". 1 993 International Conference on Parallel Processing.

[4]. Khaled Thabet ,Sumaia AL-Ghuribi(2014)."Matrix Multiplication Algorithms" IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.2

[5]. Drevet , C , Islam , M and Schost , r.(2011).” Optimization techniques for small matrix multiplication”. ScienceDirect .Theoretical Computer Science 412 (2011) 2219–2236.

[6]. Thottethodi , M , Chatterjee , S and Lebeck , A.(1998).”Tuning Strassen's Matrix Multiplication for Memory Efficiency”. ACM/IEEE SC98 Conference (SC’98).

[7]. Ju-wook Jang, V.K.K. Prasanna, Seonil Choi ” Area and time efficient implementations of matrix multiplication on FPGAs”

[8].Taras Tkachuk(2016) “Two methods to determining the time complexity of algorithm.IEEE.13th International Conference.

[9]. S. Huss-Lederman, E.M. Jacobson, J.R. Johnson(1996) “Implementation of Strassen's Algorithm for Matrix Multiplication”.IEEE

[10]. Jorge F. Fabeiro, Diego Andrade, Basilio B. Fraguela(2016)” Writing a performance-portable matrix multiplication”.Journal Parllel Computing.