

## Lab 2

Brendon Swanepoel - 601949,  
Anita de Mello Koch - 1371116,  
Nicholas Kastanos - 1393410

### 1 Comparisons between openmp and Pthreads

| $N_0 = N_1$ | Basic                   | Pthread - Diagonal | Pthreads - Blocked | OpenMP - Naive | OpenMP - Diagonal | OpenMP - Blocked |
|-------------|-------------------------|--------------------|--------------------|----------------|-------------------|------------------|
| 128         | $5.5828 \times 10^{-5}$ | 0.0057640          |                    | 0.0054073      | 0.0017463         | 0.0026457        |
| 1024        | 0.0037808               | 0.051978           |                    | 0.0028294      | 0.0040702         | 0.060210         |
| 2048        | 0.021078                | 0.18114            |                    | 0.0090414      | 0.0098804         | 0.21979          |
| 4096        | 0.088854                | 0.63340            |                    | 0.033679       | 0.039016          | 0.92435          |
| 16384       | 1.7036                  | 9.5218             |                    | 1.1346         | 0.96734           | 15.318           |

### 2 Pseudocode

**input** : Two 2D square matrices

**output**: Calculating the addition of two 2D square matrices

initialise results matrix;

**for** *Each row of the matrix* **do**

**for** *Each column element in the current row* **do**  
        Sum the corresponding row and column elements of  
        both matrices;  
        Store sum value in result matrix;

**end**

**end**

Return results matrix;

**Algorithm 1:** 2D Addition Algorithm

**input** : Two 3D cubic matrices

**output**: Calculating the addition of two 3D cubic matrices

initialise results matrix;

**for** *Each depth row of the cube* **do**

**for** *Each row of the cube at the current depth* **do**  
        **for** *Each column element at the current depth and row* **do**  
            Sum the corresponding row and column elements of  
            both matrices at the current depth;  
            Store sum value in result matrix;

**end**

**end**

**end**

Return results matrix;

**Algorithm 2:** 3D Addition Algorithm

**input** : Two 2D square matrices

**output**: Calculating the multiplication of two 2D square matrices

initialise results matrix;

**for** *Each row of the square* **do**

**for** *Each column of the square at the current row* **do**  
        **for** *Each element in the row and column of the corresponding matrices* **do**  
            Mutiply and sum the corresponding row and column elements;

**end**

        Store sum value in result matrix;

**end**

**end**

Return results matrix;

**Algorithm 3:** 2D Multiplication Algorithm

**input** : Matrix A and B, two 3D cubic matrices  
**output**: Matrix C, the multiple of two 3D matrices

```

for each row in matrix C do
  for each column in matrix C do
    for each depth in matrix C do
      Get corresponding row at depth from matrix A;
      Get corresponding column at depth from matrix B;
      Multiply the obtained row and column to get the value of matrix C at the current row,
      column and depth.
    end
  end
end

```

**end**

**Algorithm 4:** 3D Multiplication Algorithm