

### Practical No. 3

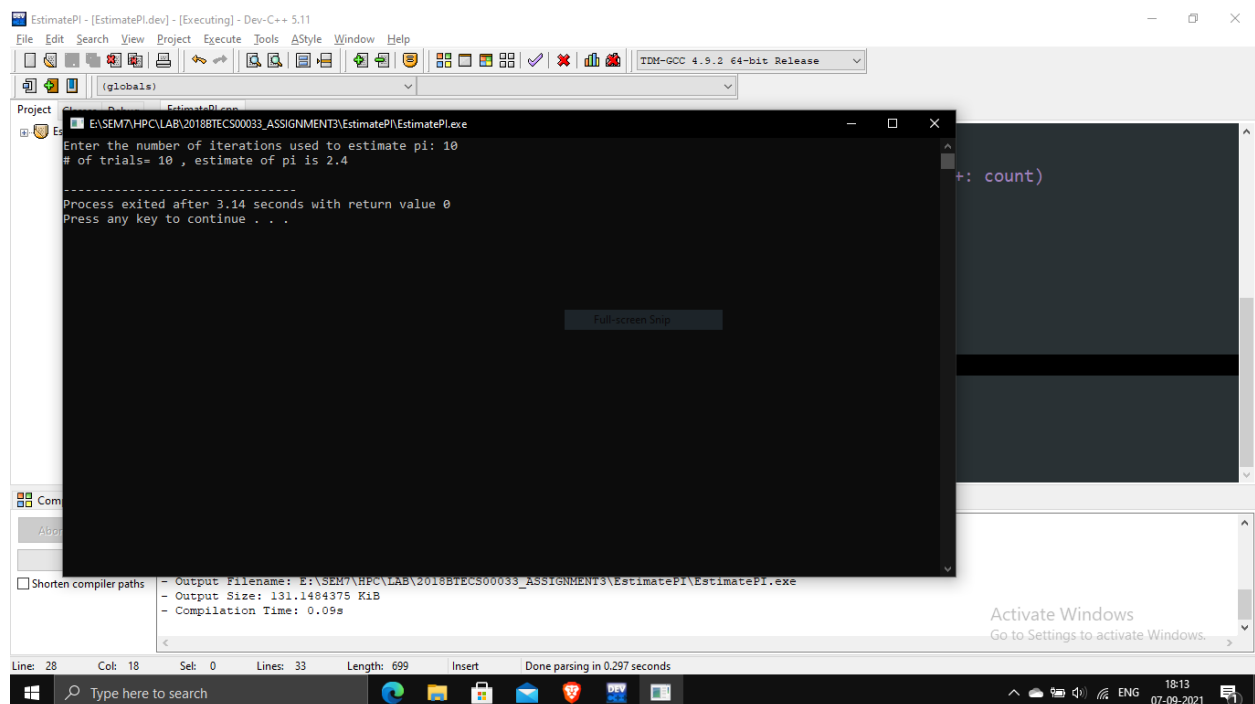
**Exam Seat No: 2018BTECS00033**

1. 2018BTECS00033- Mahendra Bhimrao Gcharge

#### Problem Statement 1:

- a. Estimation of value of PI using OpenMP
- b. Matrix Vector multiplication using OpenMP
- c. Matrix Matrix addition using OpenMP

#### Screenshot 1:



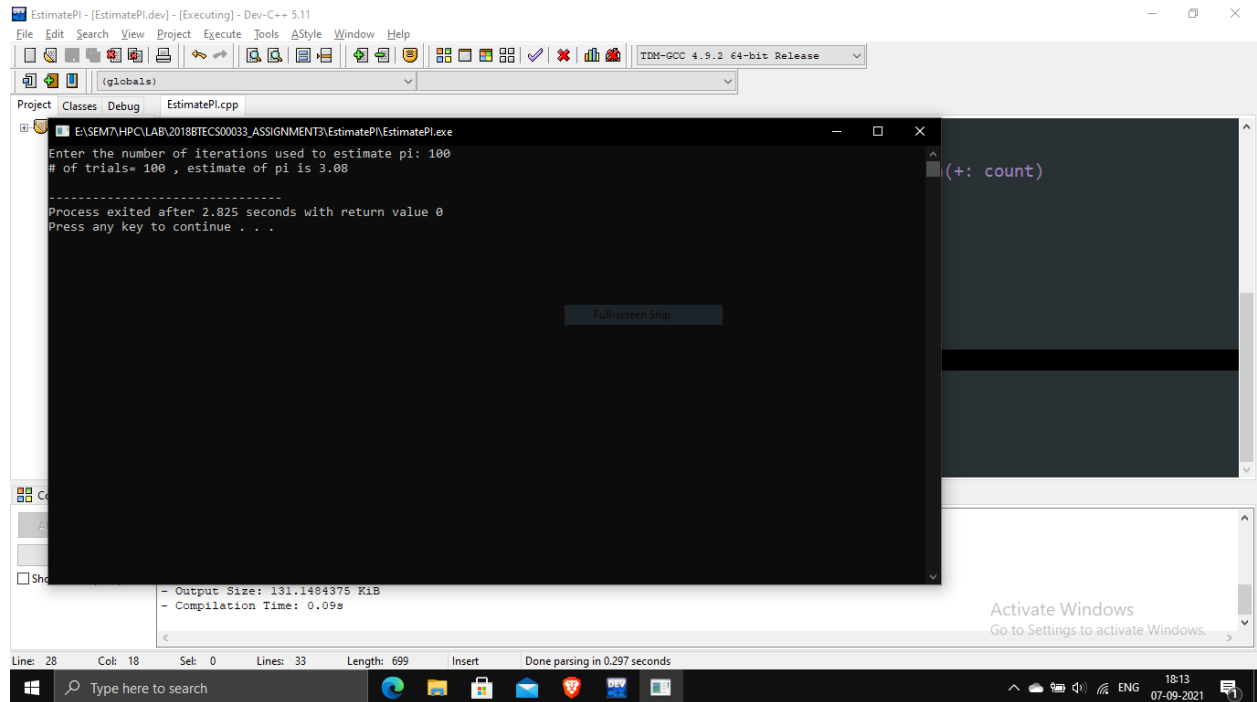
```
EstimatePI - [EstimatePI.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project: EstimatePI
E:\SEM7\HPC\LAB\2018BTECS00033_ASSIGNMENT3\EstimatePI\EstimatePI.exe
Enter the number of iterations used to estimate pi: 10
# of trials= 10 , estimate of pi is 2.4
.....
Process exited after 3.14 seconds with return value 0
Press any key to continue . . .
Full-screen Snap
- Output Filename: E:\SEM7\HPC\LAB\2018BTECS00033_ASSIGNMENT3\EstimatePI\EstimatePI.exe
- Output Size: 131.1484375 KiB
- Compilation Time: 0.09s
Line: 28 Col: 18 Sel: 0 Lines: 33 Length: 699 Insert Done parsing in 0.297 seconds
Type here to search 18:13 07-09-2021
```

**Information 1: Calculated the PI using 10 iterations as was found to be 2.4**

#### Screenshot 2:

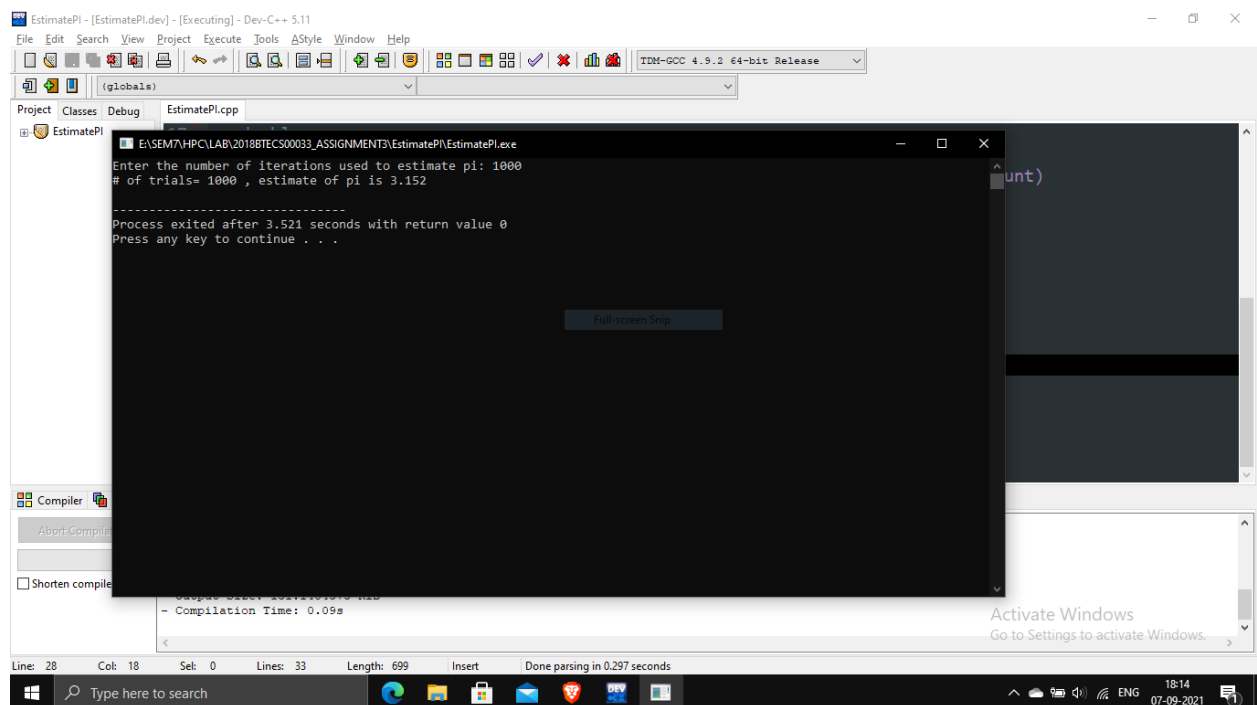
# Walchand College of Engineering, Sangli

## Department of Computer Science and Engineering



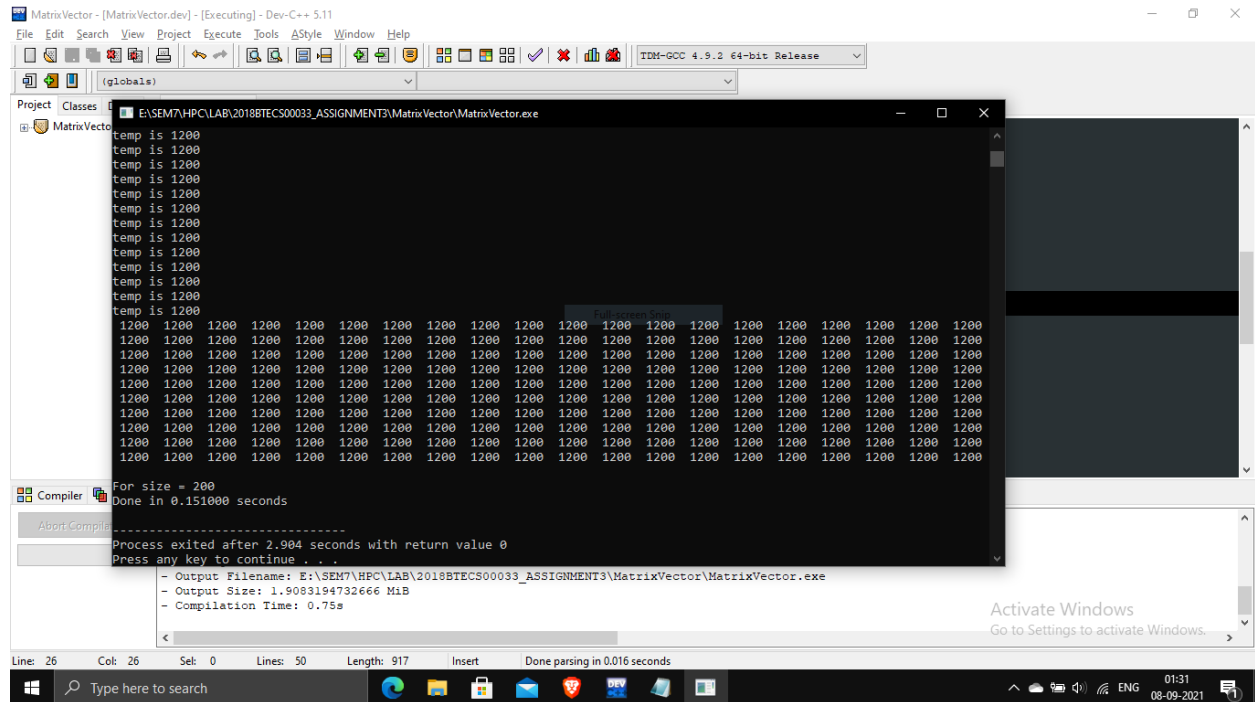
**Information 2: Calculated the PI using 100 iterations as was found to be 3.08**

### Screenshot 3:



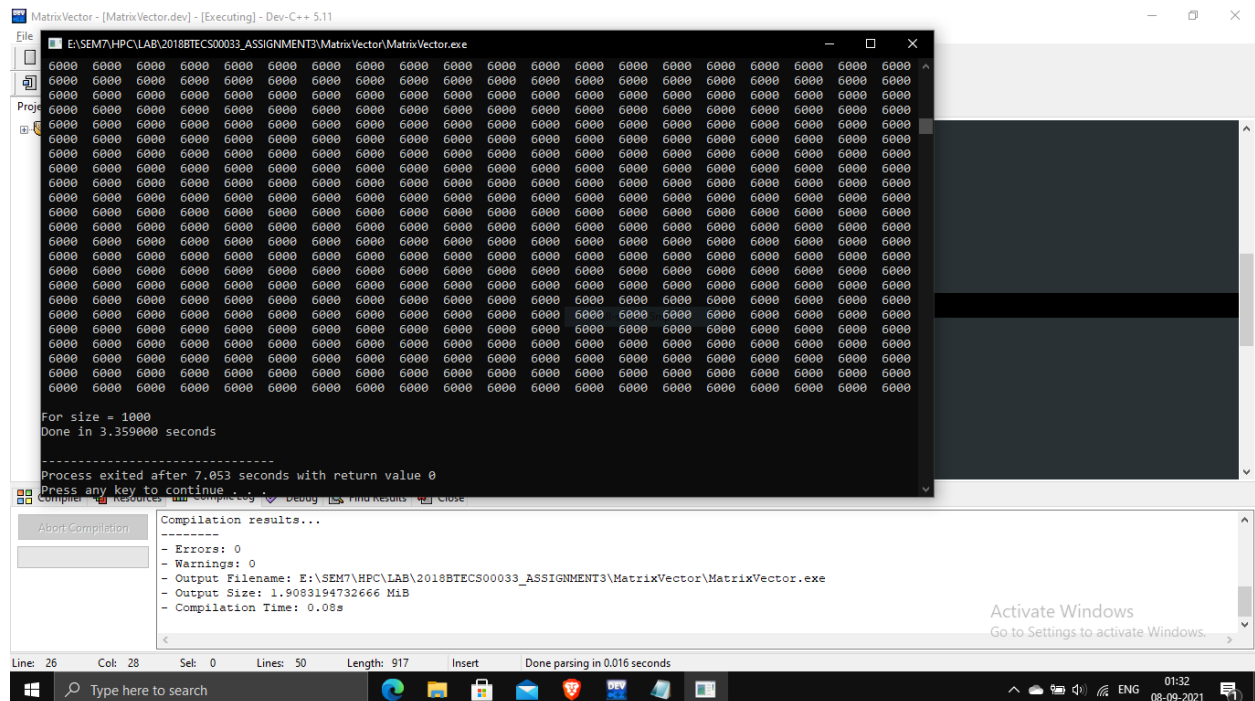
**Information 3: Calculated the PI using 1000 iterations as was found to be 3.15**

### Screenshot 4:



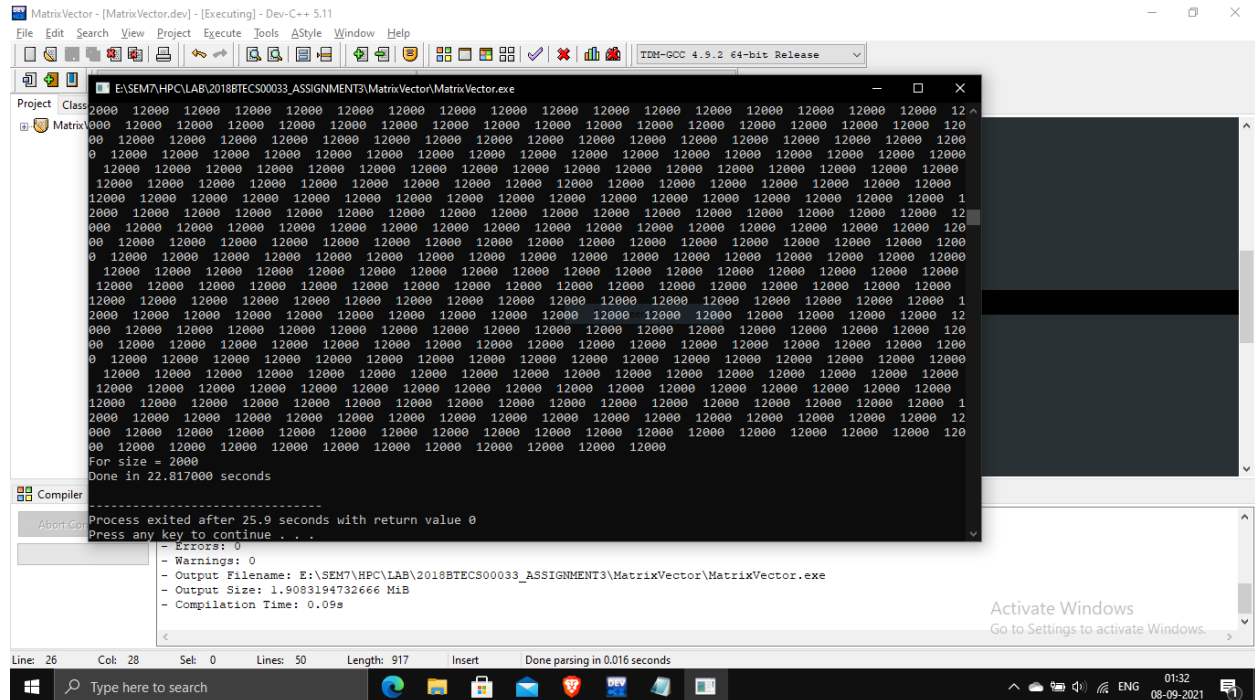
**Information 4 Calculated matrix vector multiplication using 200 as size and 1 thread**

### Screenshot 5:



## Information 5 Calculated matrix vector multiplication using 1000 as size and 1 thread

### Screenshot 6:



The screenshot displays the Dev-C++ IDE interface. The main window shows a C++ program for matrix-vector multiplication. The code includes headers for `<iostream>`, `<vector>`, and `<thread>`. It defines a `Matrix` struct with `size` and `data` members. The `main` function initializes a `Matrix` object with `size = 1000` and `data` as a vector of 1000 elements. It then creates a `thread` object to perform the multiplication. The output window shows the execution time as 22.817000 seconds. The compiler window shows the output filename as `E:\SEM7\HPC\LAB\2018BTECS00033_ASSIGNMENT3\MatrixVector\MatrixVector.exe` and the output size as 1.9083194732666 MiB.

## Information 6 Calculated matrix vector multiplication using 2000 as size and 1 thread

### Screenshot 7:

# Walchand College of Engineering, Sangli

## Department of Computer Science and Engineering

The screenshot shows the Dev-C++ IDE with a project named 'MatrixVector'. The main window displays the output of the program, which is a large grid of numbers representing the result of matrix multiplication. The numbers are arranged in a pattern that suggests a 3000x3000 matrix. The output is displayed in a black window titled 'E:\SEM7\HPC\LAB\2018BTECS00033\_ASSIGNMENT3\MatrixVector\MatrixVector.exe'. The status bar at the bottom indicates 'Line: 26 Col: 28 Sel: 0 Lines: 50 Length: 917 Insert Done parsing in 0.016 seconds'. The Windows taskbar at the bottom shows the date as 08-09-2021 and the time as 01:37.

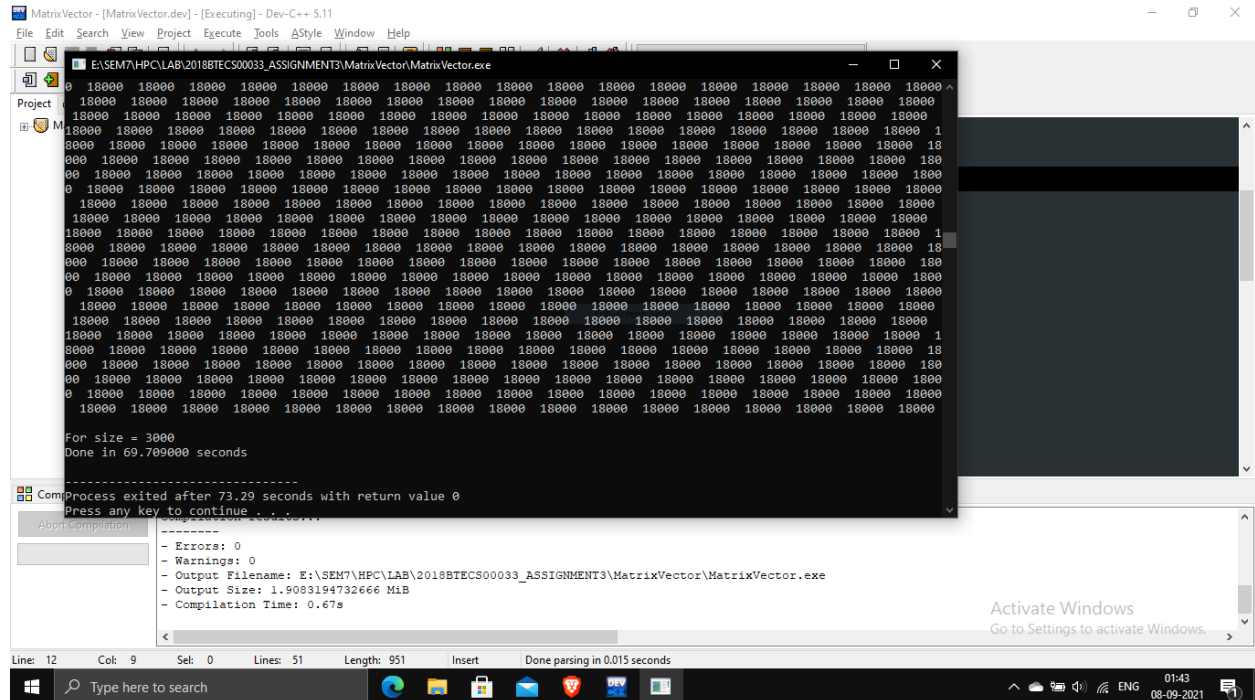
**Information 7** Calculated matrix vector multiplication using 3000 as size and 1 thread It was found that as we increase the size, the time also increases. for a single thread.

### Screenshot 8 :

The screenshot shows the Dev-C++ IDE with a project named 'MatrixVector'. The main window displays the output of the program, which is a large grid of numbers representing the result of matrix multiplication. The numbers are arranged in a pattern that suggests a 3000x3000 matrix. The output is displayed in a black window titled 'E:\SEM7\HPC\LAB\2018BTECS00033\_ASSIGNMENT3\MatrixVector\MatrixVector.exe'. The status bar at the bottom indicates 'Line: 26 Col: 26 Sel: 0 Lines: 50 Length: 917 Insert Done parsing in 0 seconds'. The Windows taskbar at the bottom shows the date as 08-09-2021 and the time as 01:38.

**Information 8 Calculated matrix vector multiplication using 3000 as size and 2 thread**

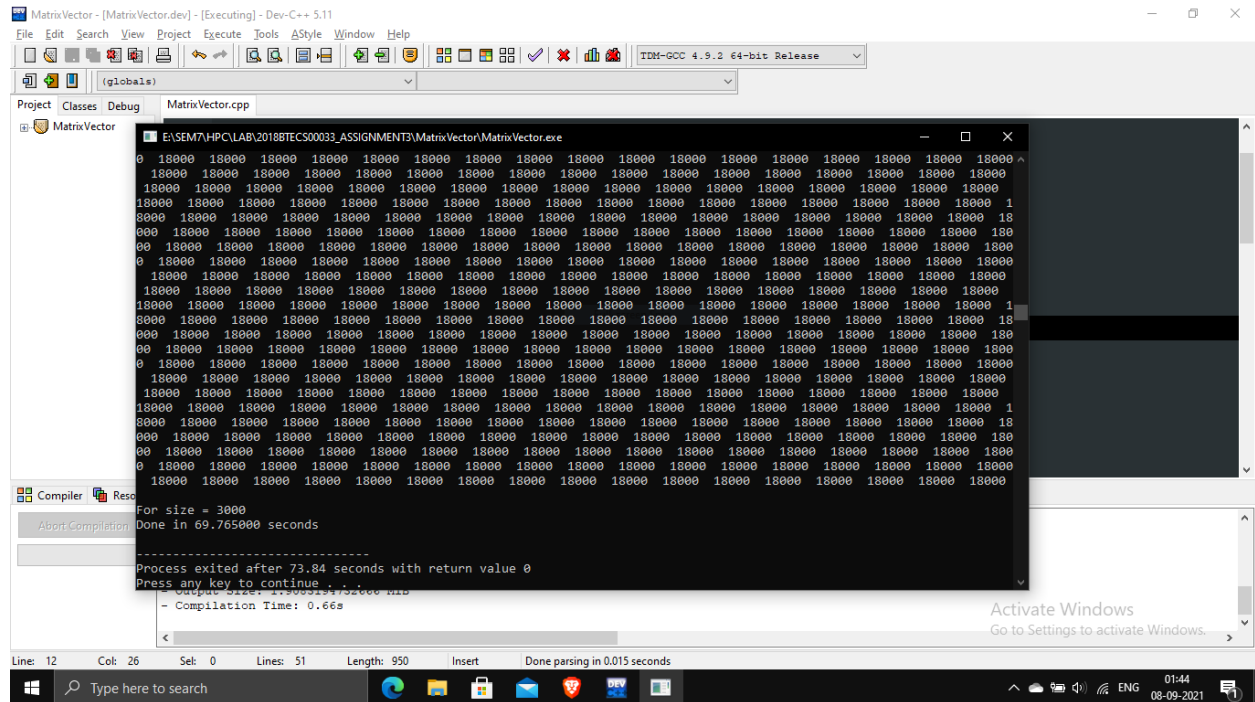
### Screenshot 9:



**Information 9 Calculated matrix vector multiplication using 3000 as size and 4 thread**

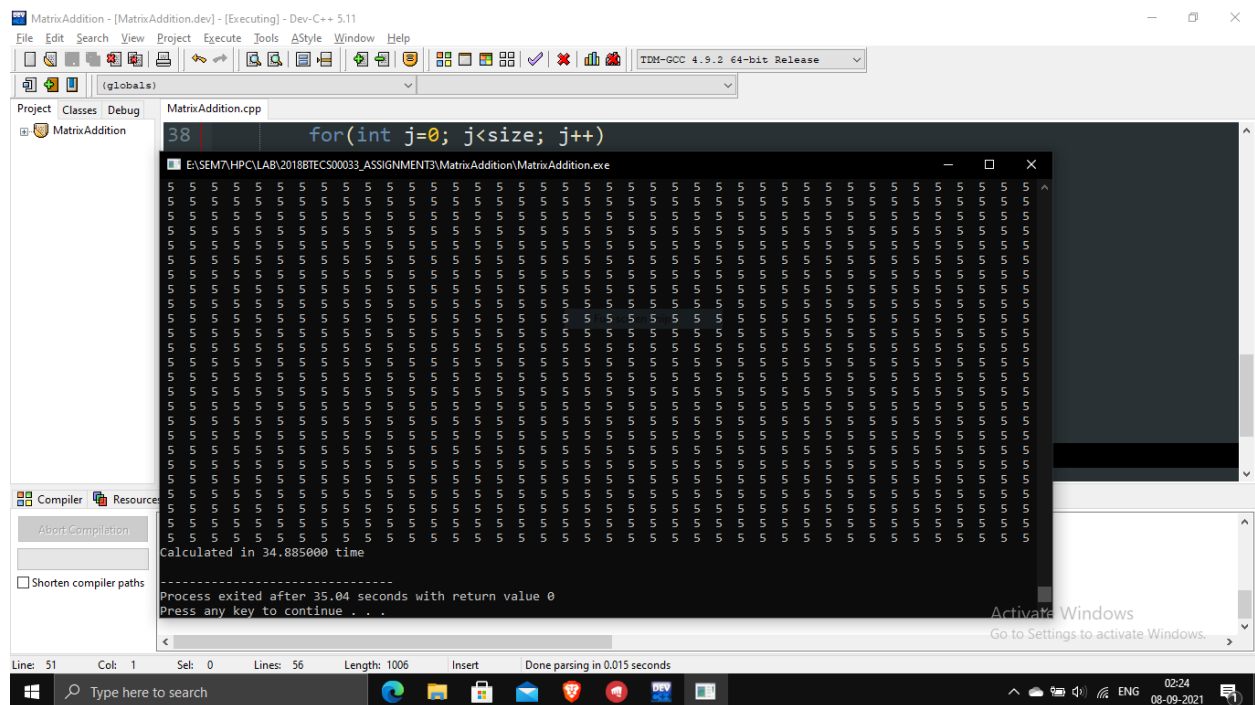
**Screenshot 10:**





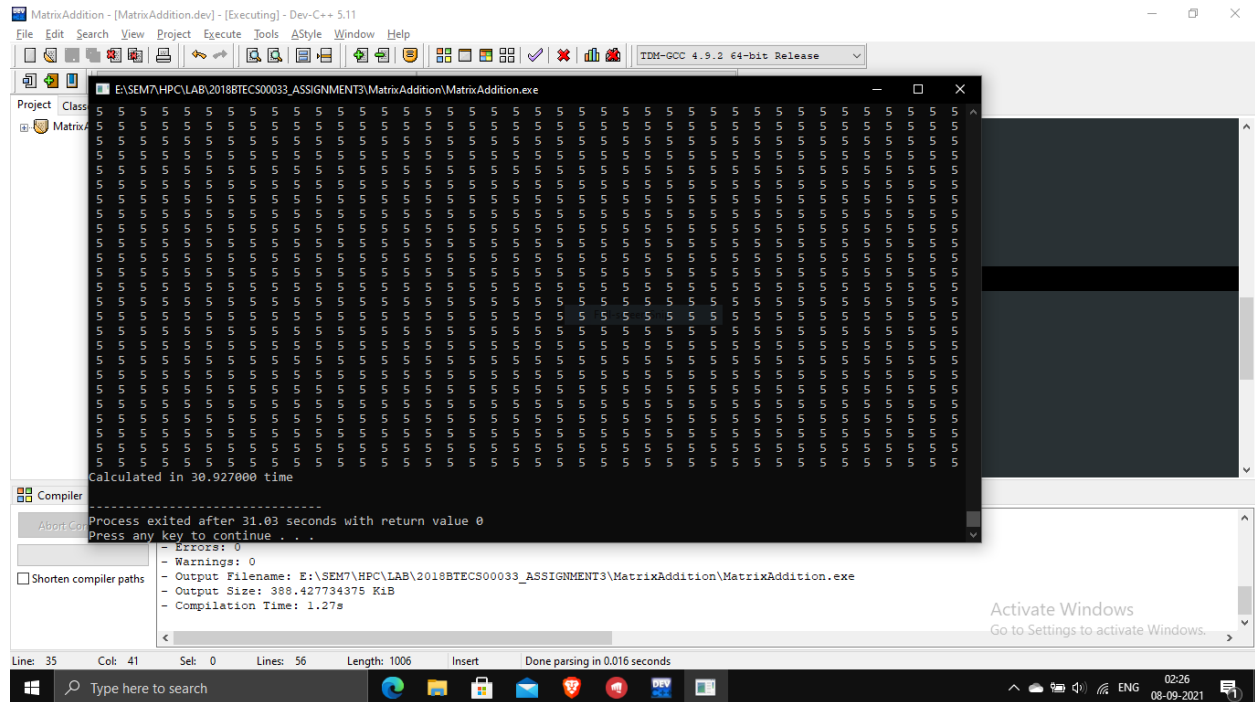
**Information 10** Calculated matrix vector multiplication using 3000 as size and 8 thread It was found that speed slightly increases and then decreases.

**Screenshot 11:**



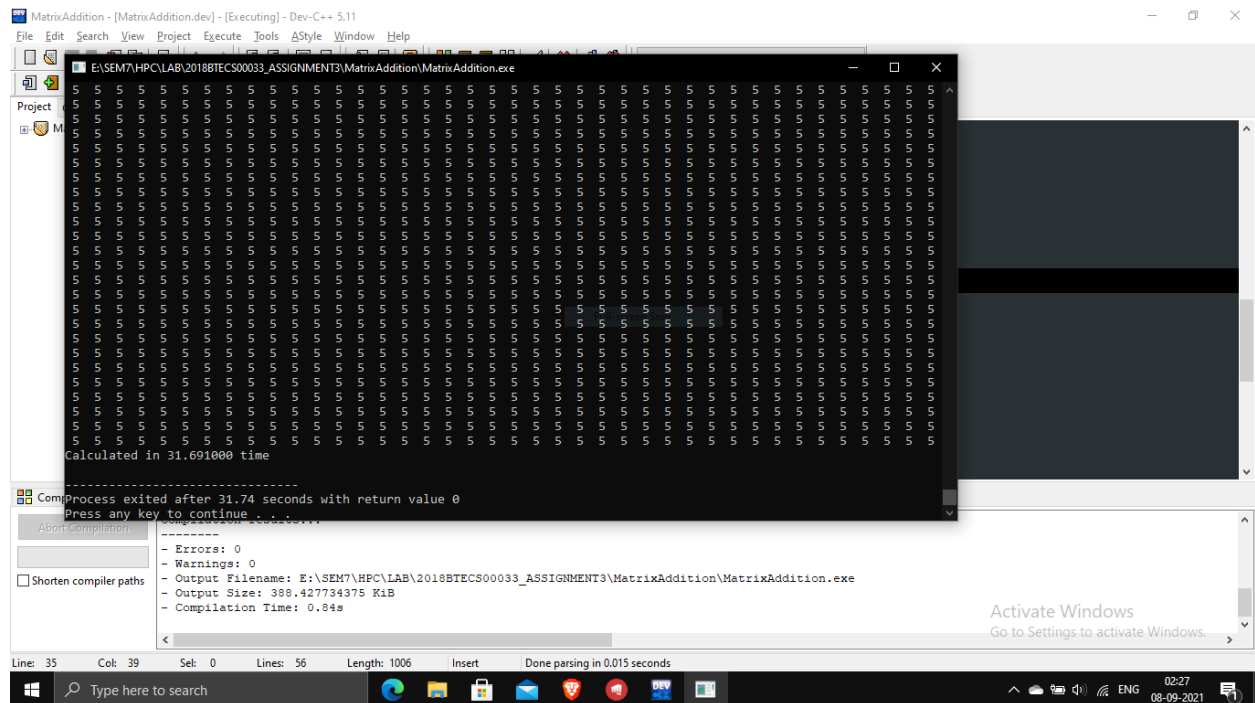
**Information 11** Calculated addition of two matrices using static schedule with chunk of 500

**Screenshot 12:**



**Information 12** Calculated addition of two matrices using static schedule with chunk of 200

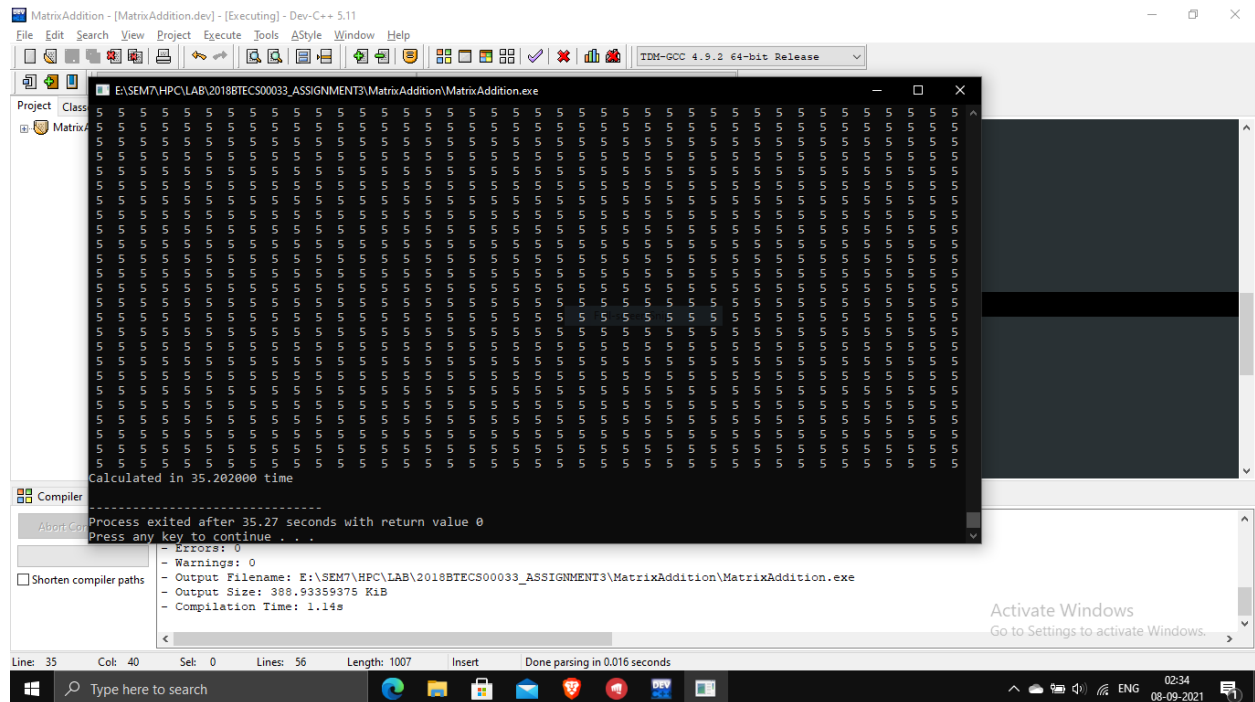
**Screenshot 13 :**





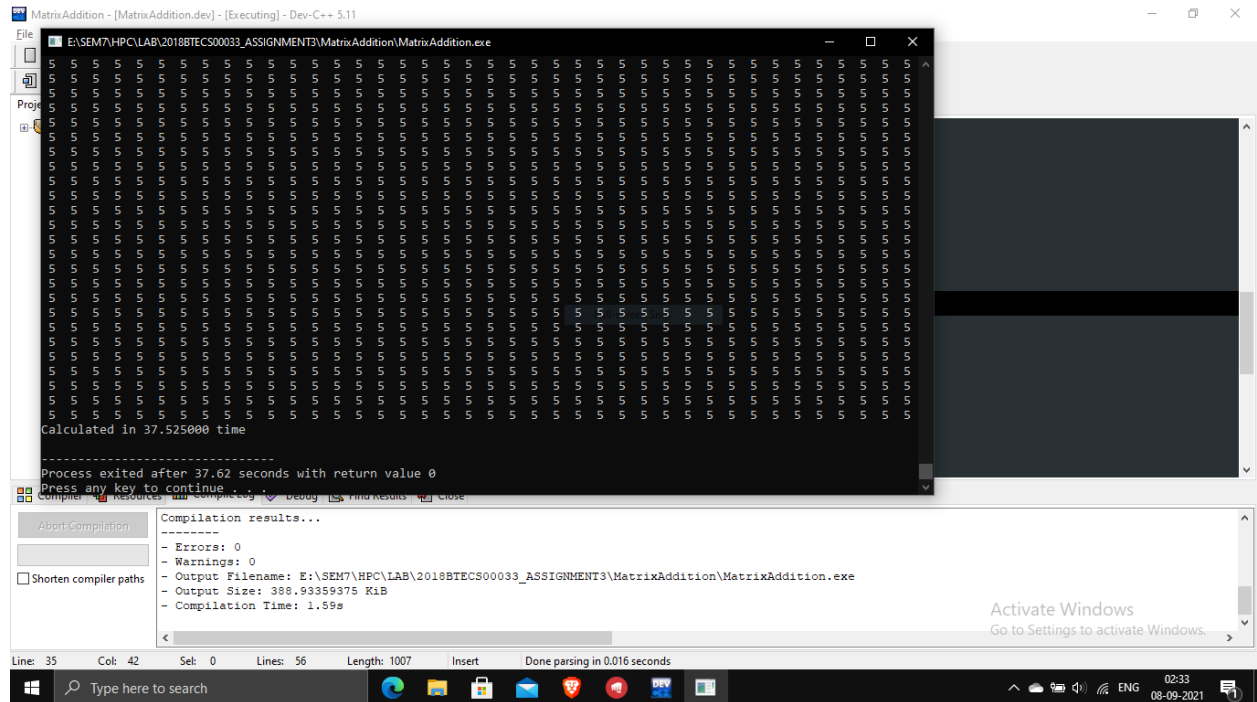
**Information 13** Calculated addition of two matrices using a static schedule with a chunk of 100. For a static schedule it was found that as I decrease the chunk size for 4 threads, the time decreases.

**Screenshot 14:**



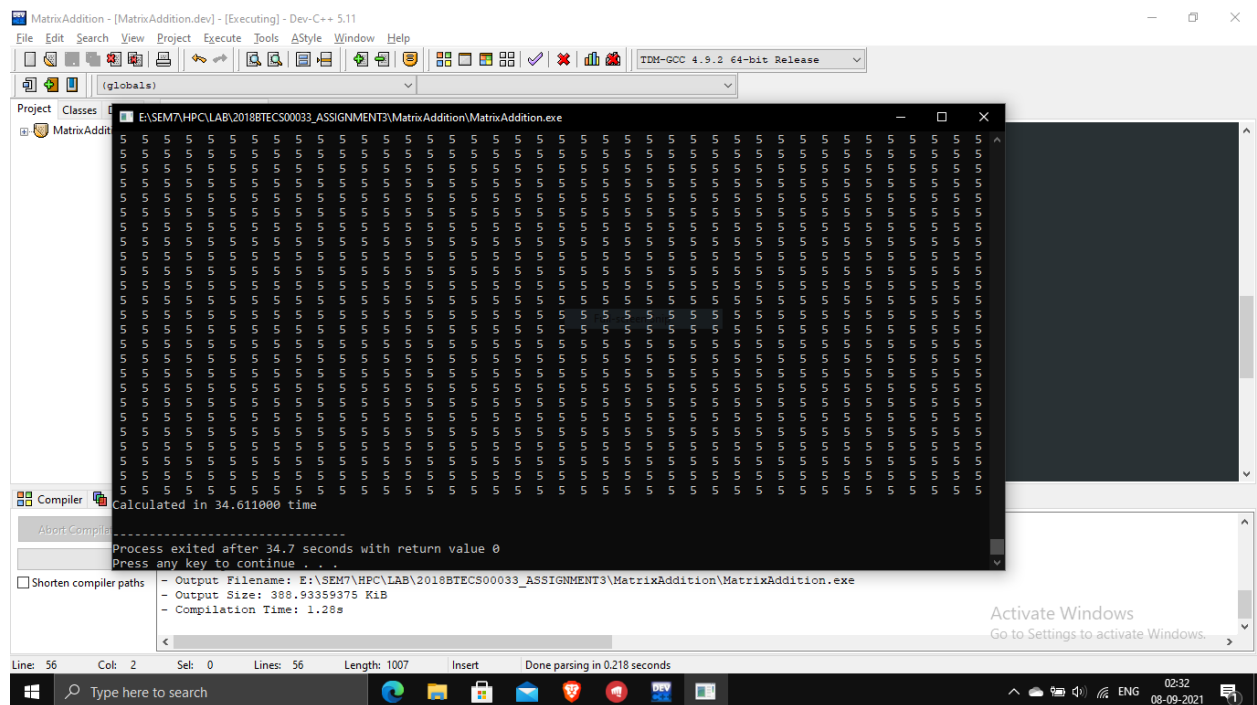
**Information 14** Calculated addition of two matrices using a dynamic schedule with a chunk of 500

**Screenshot 15:**



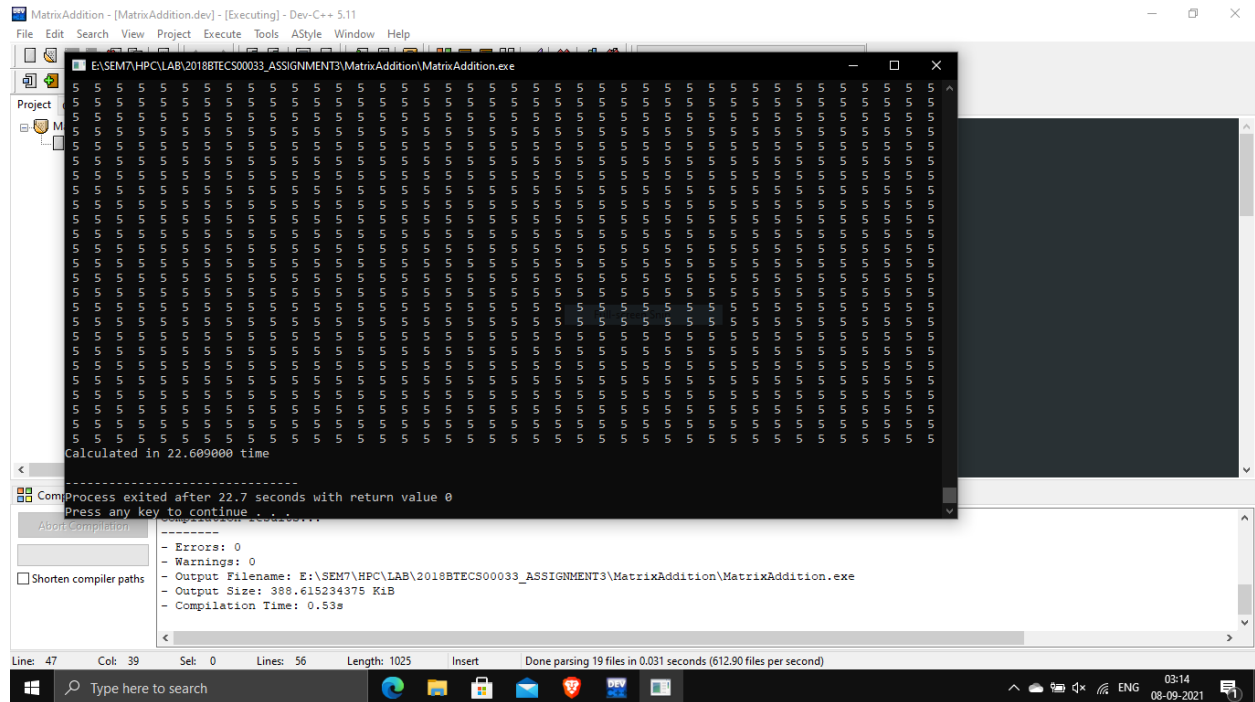
**Information 15** Calculated addition of two matrices using a dynamic schedule with a chunk of 200

**Screenshot 16:**



**Information 16** Calculated addition of two matrices using a dynamic schedule with a chunk of 100. There was no significant improvement found.

**Screenshot 17:**



**Information 17** Calculated addition of two matrices using a dynamic schedule with a chunk of 500. After using `nowait`, a significant improvement was seen.

**Github Link:** [https://github.com/g-mahendra/HPC\\_LAB\\_ASSIGNMENTS](https://github.com/g-mahendra/HPC_LAB_ASSIGNMENTS)