TRINITY COLLEGE DUBLIN, SCHOOL OF MATHEMATICS

Assignment #2, Module: MA5612

Gustavo Ramirez

October 28, 2016

1 PROBLEM DESCRIPTION

1. Write a C program that multiplies two matrices. The sizes of the matrices should again be given to the program as command line arguments and be filled with values from a random number generator.

2. Use the gettimeofday() function (or some other appropriate timing routine) to measure the time taken to calculate the matrix product for various sizes of matrices. Plot a graph of your timings using gnuplot and generate a PostScript file with the graph. What conclusions, if any, can you draw about the performance of your code.

3. Play around with various compiler options for optimizing the execution of your code. Compare the performance against the unoptimized (-O0) version timed in Task 2. Which combination of flags gives the best performance?

4. (bonus marks) Read about the BLAS library and see if you can modify your code to use this library to get better performance. You will want to look at the DGEMM function.

1.1

Following is the tree of files and directories for this assignment:

```
├─ assignment_2.pdf
├─ assignment_2.tex
├─ task1
│  ├─ matmul
│  └─ matmul.c
├─ task2
│  ├─ exec_times_curve_opt0.png
│  ├─ exec_times_curve.ps
│  ├─ execution_times.dat
│  ├─ gnp_script.gp
│  └─ program.py
├─ task3
│  ├─ exec_times_curve_opt0.png
│  ├─ exec_times_curve.ps
│  ├─ execution_times.dat
│  ├─ gnp_script.gp
│  └─ program.py
└─ task4
   └─ task4.txt
```

Three parts deserve some explanation:

1. in the directory task1/, there is a matmul.c code where matrix multiplication is implemented. The program compiled from this code, is to be called later multiple times with the use of a Python script, to test the performance of this matrix multiplication implementation

2. in the task2/ directory, the most important file is the program.py script, from which the program task1/matmul is called multiple times, for different matrix sizes, to measure time of execution for matrix multiplication versus matrix size. After that, the gnp_script.gp

can be called with the gnuplot command to give output exec_times_curve.png, with a plot of time versus size (clearly, this sizes are symbolic; to understand better the range of sizes used, brief explanations are given within the program.py file, but basically, the range of matrix sizes are: $5 \times 15 \cdot 15 \times 5 \rightarrow 99 \times 15 \cdot 15 \times 99$)

3. in task3/ directory, figures can be found for different performance tests, for several used flags

1.2

For this part, a Python script was written for the generation of the data associated to the execution times for matrix multiplication at different sizes.
Also, a .gp file was used for the images generation with the use of Gnuplot.
For the implemented serial matrix multiplication, from the task2/exec_times_curve_opt0.png file, a simple fitting gives a resulting complexity of $\mathcal{O}(n^2)$.

1.3

As can be seen, in the directory task3/ are seven images of different optimizations employed in the compilation of the *matmul* program for matrix multiplication.
To check which flag corresponds to which image, the flag is attached at the end of the name of the file; e.g. exec_times_curve_opt_fast.png means than the -Ofast flag was used.
The same range of axis were set in all the plots, from which can be seen that the -O2 flag results in the best performance.