Christopher Almajose
A10463176

Programming Assignment 3

2.3.3: Hadamard and Laplacian (Linear Barrier, n=1024)

Graph:

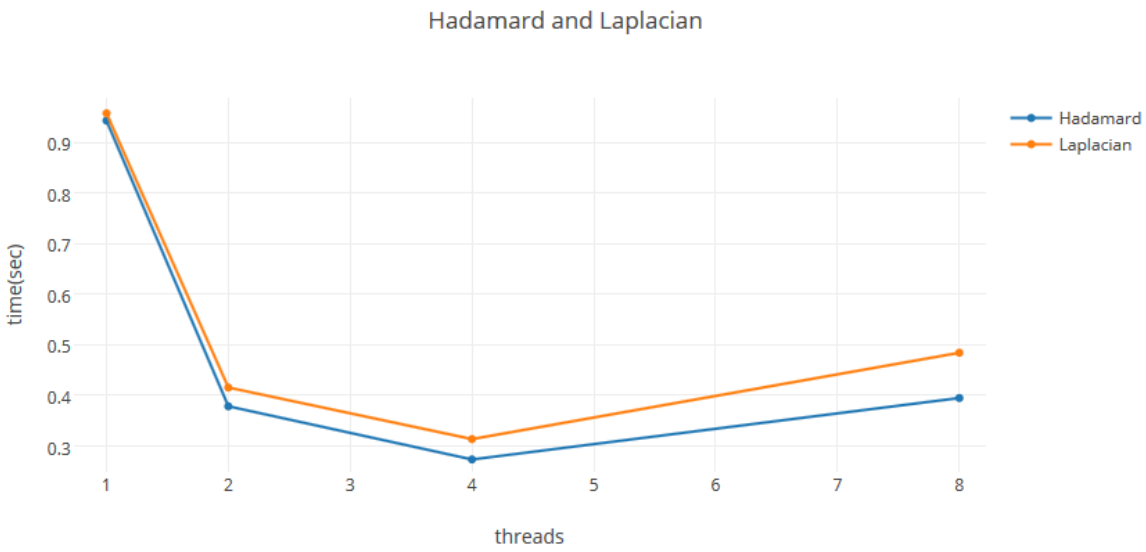
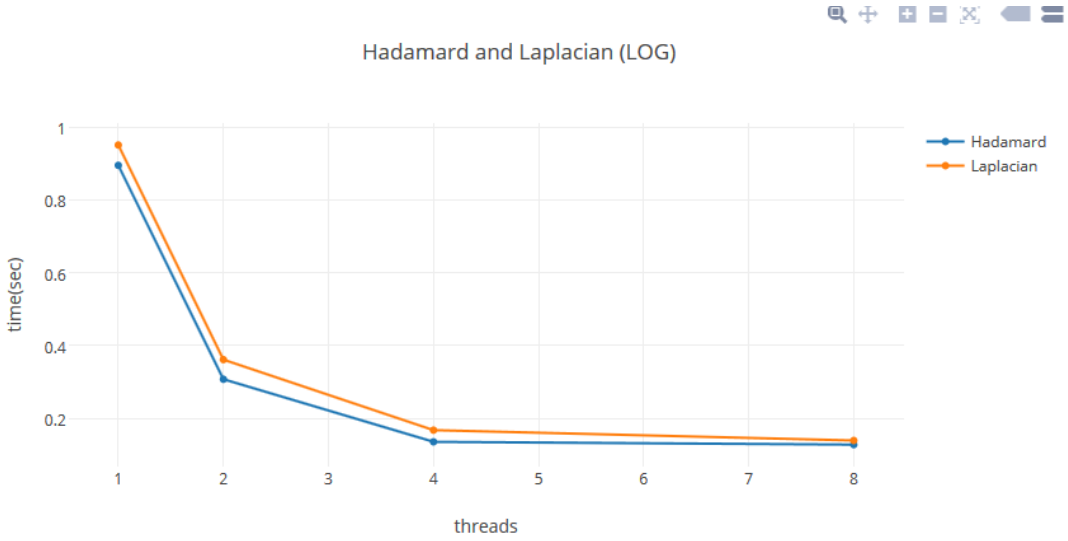


Table:

Hadamard	threads	Laplacian
0.943085	1	0.957427
0.378607	2	0.415721
0.273547	4	0.313767
0.394964	8	0.484376

Overview: There was a speedup in both types of matrices as we parallelized the work. However, there was some slowdown with eight threads. This was due to the the linear barrier starter code implementation as it can only handle four threads effectively.

2.3.4: Hadamard and Laplacian (Log Barrier, n=1024)



Hadamard	threads	Laplacian
0.894901	1	0.950637
0.307624	2	0.361530
0.135845	4	0.167822
0.128018	8	0.139433

Overview: With the implementation of the LOG BARRIER, there was a significant speedup
by a factor of two as the number of threads increases. This was due to the limitations of the linear barrier using mutexes that can handle a load of usually four threads.

Speedup:
Graph

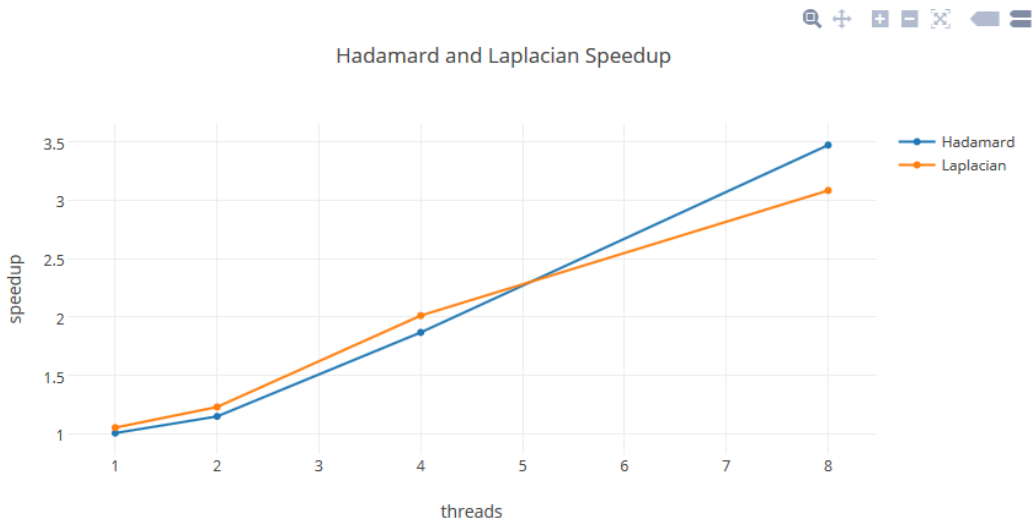


Table:

Hadamard	threads	Laplacian
1.007142579...	1	1.053842827...
1.149893508...	2	1.230745975...
1.869641644...	4	2.013669991...
3.473897857...	8	3.085222390...

Overview: Plotting the speed up, we can see that is linear as we increase the number of threads and versus the linear barrier and the log barrier.

2.3.5: NOTES

As noted from the assignment format, although I implemented Log barrier for the assignment, I was not able to implement the final version properly to be within the times based on the reference library. I implemented it using atomics and atomic_flag provide by the pthread library, but I was somewhat lost in where screwed with the implementation. However, I was able to see the results of a log barrier effects using my current implementation and the plotting of the data.