To: Christina Taylor
From: Eli Case
Date: May 5, 2023
Subject: CAAM 420/520 – Homework 6

Problem 1 (a) Using the MatLab curve fitting toolbox, a value of $p = 0.6859$ was determined within a 95 % confidence interval. The generated plot with its corresponding data is shown below in Figure 1.

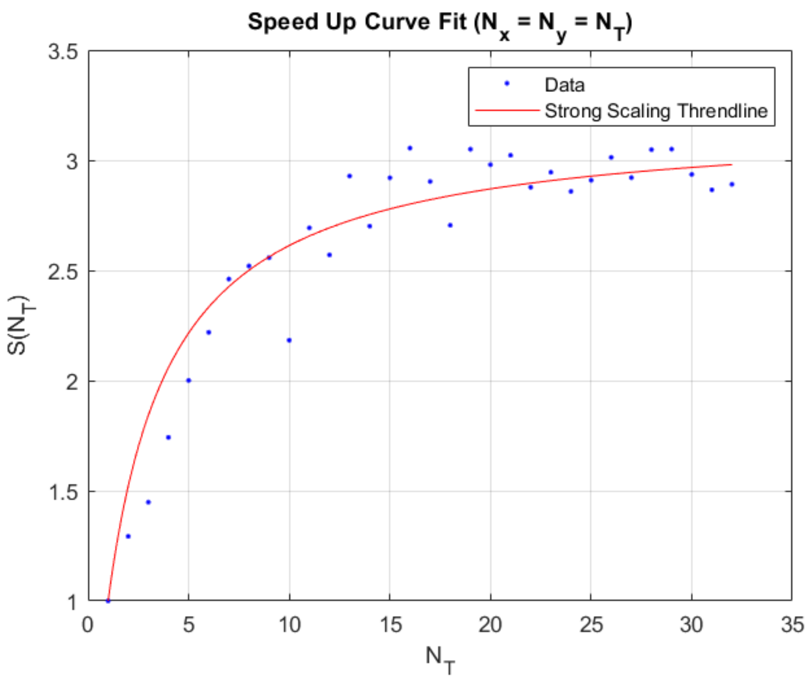

Figure 1. Strong Scaling Curve Fit for $N_x = N_y = N_T$

(b) Using the MatLab curve fitting toolbox, a value of $p = 0.8306$ was determined within a 95 % confidence interval. The generated plot with its corresponding data is shown below in Figure 2.

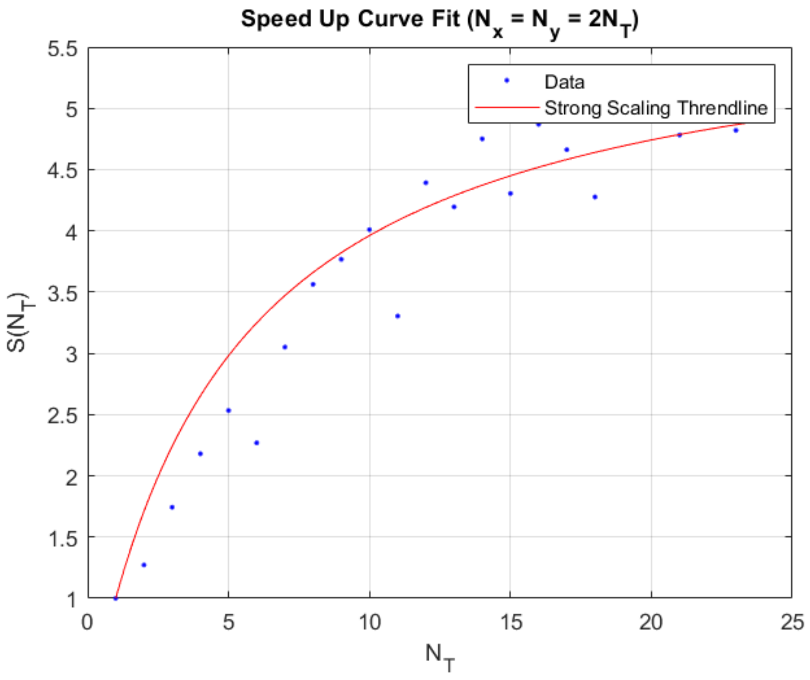

Figure 2. Strong Scaling Curve Fit for $N_x = N_y = 2N_T$

(c) The performance deviated from theoretical asymptotic behavior when more than 24 threads are used for $N_x, N_y > N_T$ because the fully spun up region has more blocks than threads that are able to be active. This configuration causes some threads to have to be active more than once per diagonal, adding additional parallelism overhead time unaccounted for in the theoretical strong scaling when a single wave is processed. In the case of $N_T > 24$, the additional overhead time of the parallelism dominates any potential time saved by having additional blocks. Thus, the aggregate time delay from having more parallelism overhead introduced when more blocks than threads are active in the fully spun-up region causes performance degredation when $N_T > 24$.

(e) $p \neq 1$ for the function `wavefront520` because of the overhead cost of parallelism. Even in a fully parallel region, a given thread will need to devote time to, for example, declare local variables, update local variables, synchronize threads, and read and write to shared memory, operations which take approximately the same amount of time regardless of the number of threads used. As a result, even in a region that is fully within a code block of operation executed in parallel, the overhead cost of parallelism that is independent of the amount of threads used will always result in $p < 1$ when measured from actual timing data. The consequences of this result is that even a fully parallel region will always have some degree of code whose runtime does not scale as the problem is parallelized further. To make p closer to 1 for a measured fully parallel region, one could declare variables outside of the parallel region, reduce the number of reads and write to shared memory inside the parallel region, and find different problem implementations to reduce the instances of thread synchronization required for the problem, for example.