Spring 2019: Advanced Topics in Numerical Analysis: High Performance Computing Assignment 3 (due Apr. 1, 2019)

Terrence Alsup

1. **Approximating Special Functions Using Taylor Series & Vectorization.**

   To evaluate $\sin(x)$ to 12 digits of accuracy we only need to sum up to the 11th order terms since

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + O(x^{13})$$

   We improve the accuracy to 12 digits for the function `sin4_vec()` by simply adding the higher order terms. Using a CIMS desktop Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz with 4 cores the time taken to sum this series with serial code is about 1.835s, whereas in parallel using `sin4_vec()` it is only about 0.846, which is roughly twice as fast.

2. **Parallel Scan in OpenMP.**

   To parallelize the scan we break the vector into p chunks, where p is the number of threads we are using. To balance the loads between each thread we split the vector into even chunks, except for possibly the last one which must handle any remaining pieces (due to integer division). We run the scan on a vector of length $n = 10^8$ for different numbers of threads. The machine used was a CIMS desktop which is Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz with 4 cores. The table below shows the results. 4 threads seems to be optimal which is how many cores the machine has. Too few threads and we have worse performance than the serial code due to the overhead cost of setting up the parallelism. Too many threads and we can no longer see any speed-up.

Threads	Time (s)
1	0.485
2	0.286
3	0.221
4	0.189
6	0.198
8	0.215
12	0.209

Table 1: The wall-clock time (s) to scan a vector of length 10^8 with different threads. For reference, the serial code takes 0.374s.