High Performance Computing
Anthony Trubiano
Assignment 3 (due April 1, 2019)

Note: All computations were performed with an Intel Core i7-8750H Processor with base frequency 2.20 GHz. The maximum main memory bandwidth is 41.8 GB/s. At 16 double precision operations per cycle, the theoretical max flop rate would be about 35.2 GFlops/s. It has 6 cores and can reach 12 threads through hyper-threading.

1. **Approximating Special Functions Using Taylor Series and Vectorization.** I improved the accuracy of the function sin4_vec() simply by adding more terms to the Taylor series using the Vector class. Including terms up to $O(x^{11})$ gives the same error as the sin4_taylor() code given but runs about 2.5 times faster.

2. **Parallel Scan in OpenMP** Here we parallelize the scan operation using p threads. Let n be the length of the vector. The parallel implementation takes in the number of threads, then assigns each a chunk of size p/n. If there is a remainder, the last threads gets assigned these entries. Each thread then performs the scan operation on its chunk. We then compute a correction in serial by adding the final entry of each chunk to all entries in the following chunk. The speed of the algorithm as a function of number of threads is shown in the following table for $n = 100,000,000$.

Threads	Time (s)
Serial	0.224
1	0.287
2	0.188
3	0.153
4	0.141
5	0.135
6	0.132

Table 1: Time taken (in seconds) for the scan operation on a vector of length $n = 100,000,000$ as a function of thread number.

From this table we see that using 2 threads is enough to get slightly better performance while using 4 threads approximately halves the amount of time taken. We don't see p times speed up because the parallel implementation has to perform the fix calculations that the serial version does not, which takes a non-negligible amount of time. Additionally, if we make n smaller, by a factor of 10 for example, the parallel version is about the same if not slower than the serial version. We need a large enough vector to get noticeable improvements from parallelization.