Group 1: William Beck, and Robert Lucio

Homework 8 Results Summary

Part 1:

The pure python implementation for euler_ode took 12.0 seconds on [c001]. Using line_profiler on the code, it was found that the most time taken in the code was running the integration function, as expected. Within the integration function, the longest time was spent calculating each step of the Forward-Euler algorithm. Surprisingly, only 11 out of 41 seconds were spent doing the function evaluation itself. The additional thirty seconds may have been due to moving memory around, and/or python's interpreter. The team added the Numba jit decorator to the int_funct function, and re-ran the code. The new runtime was 9.5 seconds. The time savings were likely due to python no longer having to search for numpy's sine function, since Numba compiled it ahead of time. The team added Numba's jit decorator to the remaining functions, and ran the code again. The execution time was drastically reduced to 0.92 seconds. This may be due to the Numba compilation removing a bunch of function searches, datatype and function compatibility checks, and compiling the for loop in the euler_integration function, as well as already having the int_funct compiled (which in turn may have made compiling and/or running the compiled euler_integration function easier). One thing to note is that before unloading Intel's OneAPI (which was loaded for class), the pure python implementation took longer (it was 20s), and line_profiler was unusable.

Part 2:

The pure python code took 87.6 seconds to complete. Using Numba, the time decreased to 2.4 seconds. Below is a graph showing the speedups and efficiencies for different numbers of Numba parallel workers (ignore the faulty legend):