Homework 1

Problem 1

Explanation

The folder contains a testbench and a comparison program for comparing the vectorization performance of a matrix x vector and vector x matrix multiplication problem.

How to run

run `cmake -DCMAKE_BUILD_TYPE=Release <path to CMakeLists.txt>` from the preferred build files destination. Then run `make mul_test` to build the test file and `make mul_bench` for the comparison. Run mul_bench using `./mul_bench <no. of particles>.`

Functionality

The test program basically checks if the row vector and column vector multiplications are being done correctly. The values for the matrix and the vector are hard-coded (same as in the example README). The results can be verified.

mul_bench runs the multiplication functions 5 times each for the number of particles and prints out the timing along with the first element of the result vector (as a soft verification tactic). All the values are 0, so the resultant values should be 0 as well.

Results

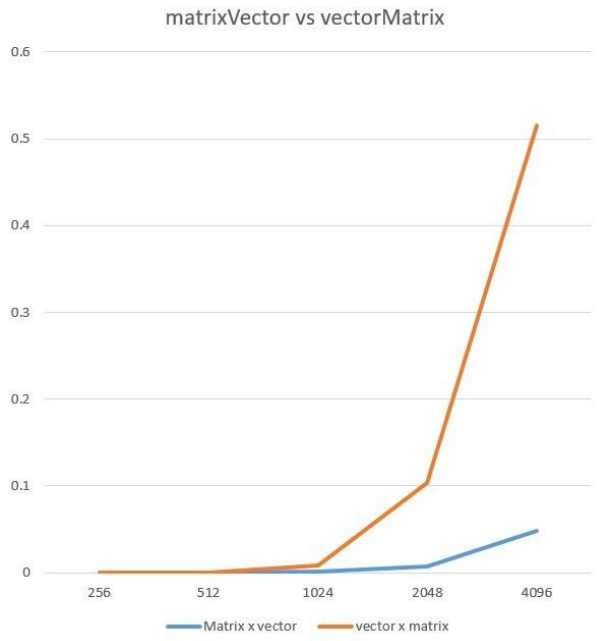
Vectorization

The loops that were succesfully vectorized were the initialisation loop in mul_test.c (trivial) and the matrix x vector multiplication loop from matvec_mul.c. The vector x multiplication loop was not vectorized.

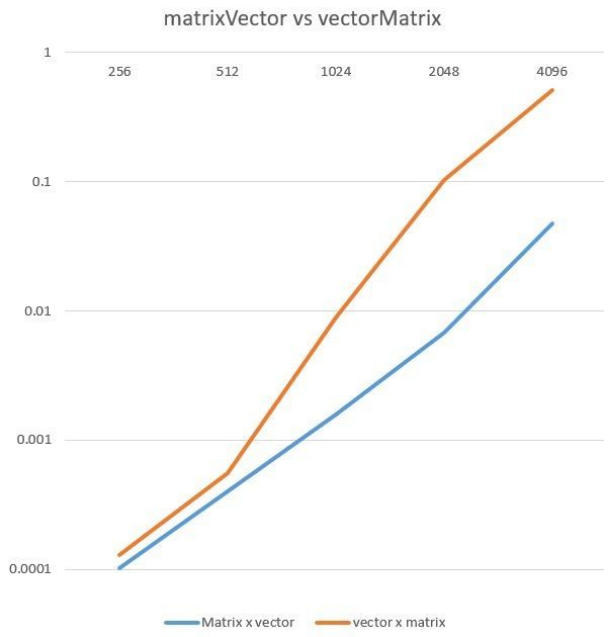
In my opinion, the reason why this loop was able to be vectorized is because the access pattern is pretty simple. More importantly, the elements in the loop are adjacent to each other in memory (all elements of a row are adjacent to each other). This is also the reason why a similar loop in vector x matrix wasn't vectorized was because the loop accesses all the members of a column that are not adjacent in memory, so would be tough to vectorize.

Timing

The vectorized function was predictably faster, as can be seen with the below graph: (y-axis in seconds)



Normal Scale



Logarithmic Scale

Valgrind

Valgrind report clean

Problem 2

Explanation

The folder contains two implementations (array of structs and struct of arrays) to record the movement of particles in a 2D environment.

How to run

run `cmake -DCMAKE_BUILD_TYPE=Release <path to CMakeLists.txt>` from the preferred build files destination. Then run `make particle_aos` to build the array of structs file and `make particle_soa` for the struct of arrays. Run either using `./particle_soa [no. of particles timesteps]`. The program will run fine without any arguments.

Functionality

The programs both start off by assigning a random position and velocity to every particle on the plane and then tracks their movements for the given number of timesteps. The .bmp files visualising this movement can be found in the bmp folder.

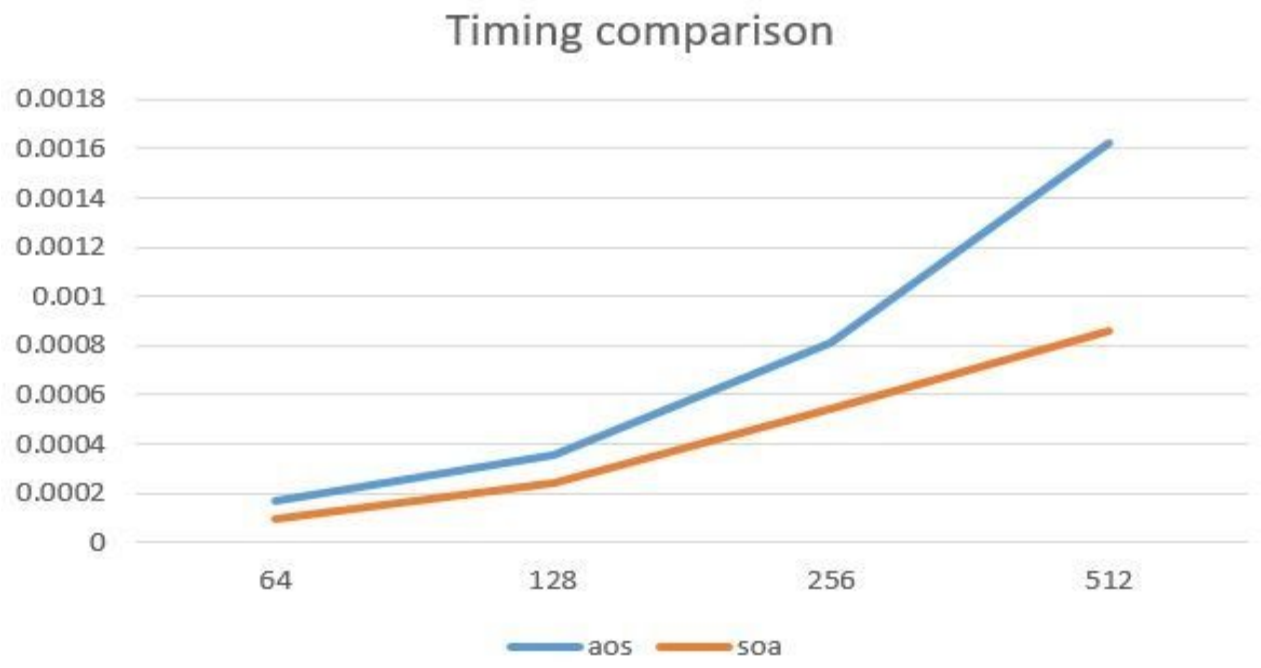
Results

Vectorization

The optimization reports for both the programs were similar. Both the array of structs and struct of array loops were vectorized as per the compilation messages. I was not expecting the array of structs version to vectorize because the data members are not technically adjacent to each other, which is why it probably works in the struct of arrays version. However, I think the fact that the individual structs themselves are adjacent to each other and the data members are at a fixed position inside the structs could have helped with vectorization in the first case.

Timing

Both the versions had comparable results. However, with 512 particles and 128 timesteps, the struct of arrays version was consistently faster at 0.000217 seconds on average, while the array of structs version took 0.000360 seconds.



Valgrind

Valgrind report clean