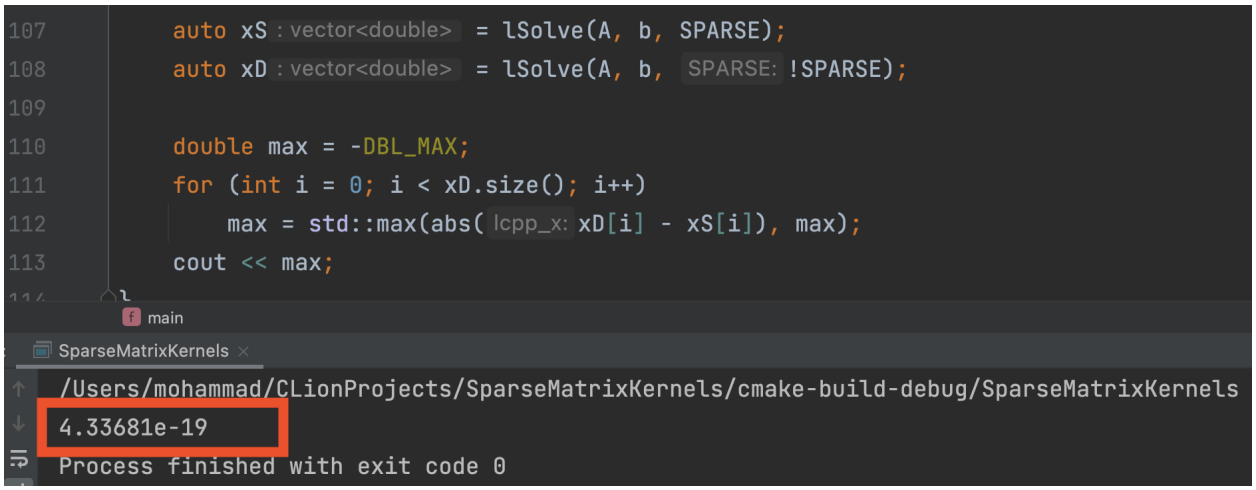
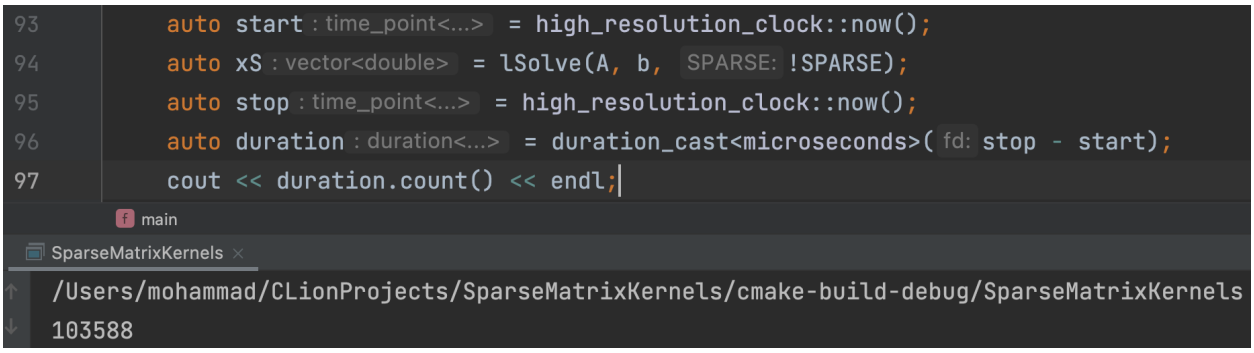
Sparse Matrix Kernels

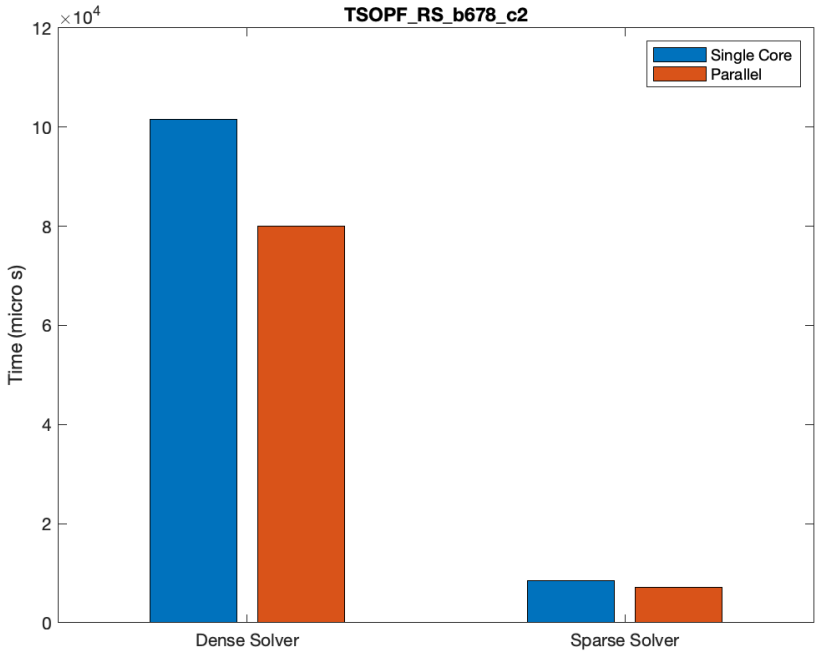
In order to solve the sparse systems for *TSOPF_RS_b678_c2* and *torso1* matrices, we first use naive implementation of the lower triangular solver and then improve it by taking advantage of Algorithm 3.2 described in "A survey of direct methods for sparse linear systems", page 10. The Sparse Solver finds the set $Reach_L(B)$ and solves the equation just for these nonzero entries of vector *x*. It does so using *topologicalSort()*.
First, we convert the matrix market format to CSC using *csc_formatter.h* and then pass the matrices to the solver.
Correctness of sparse solver approved by comparing the answer to the naive solver using $||xS - xD||_\infty < \varepsilon$, where *xS* is the solution of Sparse Solver, and *xD* is the solution of the Dense Solver. You can see the test for *TSOPF_RS_b678_c2:*

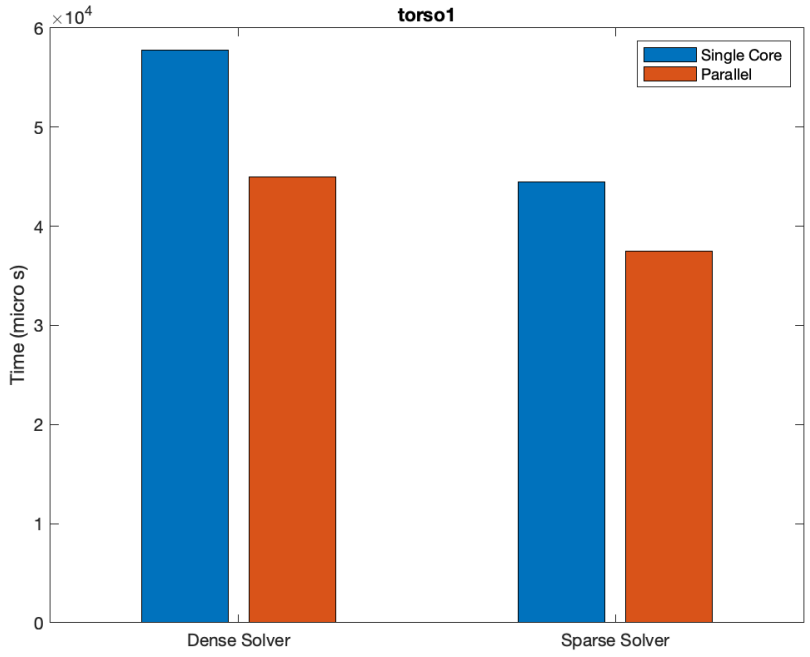


The execution time for each solver is measured using *high_resolution_clock.* You can see a test for *TSOPF_RS_b678_c2:*



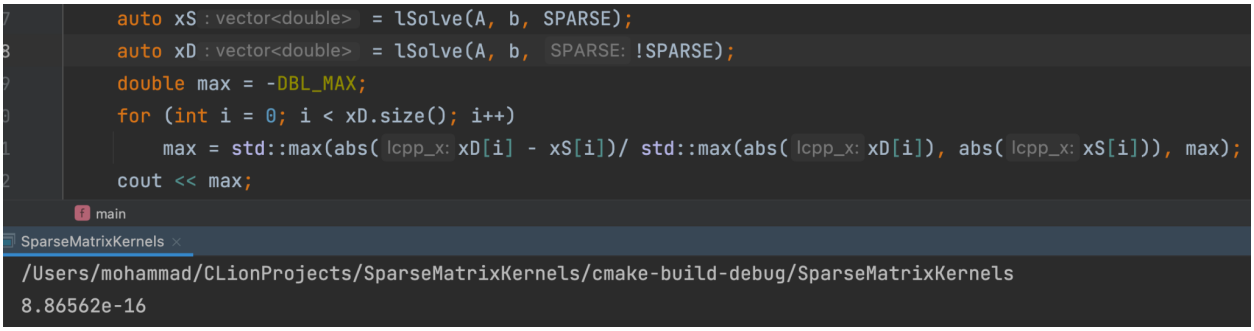
All tests are done on MacBook Pro, Quad-Core Intel Core i5, 8 GB Memory. Now we compare running times for variation of these methods for two systems:





The effect of parallelism for Dense Solver is more than Sparse Solver. That is because the Sparse Solver uses *topologicalSort,* which OpenMP could not optimize since it has to preserve the order of steps.

Also, for ill condition systems, we provide another useful test that measures the relative error of two answer vectors. You can see an example here:



All codes are written in a clean and well-commented way, which you can follow the algorithms and steps easily.