Report

Modular Design and Test Stratigies

1. Iterative_solver
   I choose to implement a Jacobi iterative matrix solver. For an *"Iterative_solver"* object, there are two fields: a sparse matrix A and a vector b. There are mainly three different modules in this class. First, I create a method named *"getDiagonal()"* to get the diagonal matrix of the matrix A. Then, I also need to get the inverse of the diagonal matrix. Thus, I create the method *"inverseD (SparseMatrix D)"*. Finally, I create a method *"iterate()"* to the iteration using all the methods I have created. For the purpose of efficiency, I also implement two helper methods *"specialProductAB (SparseMatrix A, SparseMatrix B)"* and *"specialProductAX (SparseMatrix A, Vector x)"*. These first method is used to get the product of two matrices when matrix A is a diagonal matrix. And the other one is used to get the product of a diagonal matrix A and a vector x.

2. Tests
   Before we use the *"Iterative_solver"* class, we need to do some test to make sure that all the modules in the class can return expected results. I include all the test methods in the class *"Tests"*.
   The first test is "known matrix test", which means we use the iterative solver to solve an equation with known answer. If the second norm of the residual vector is less than 10^{-7}, the method will return "Pass", otherwise, the method will return "Fail".
   The second test is for the method *"specialProductAX (SparseMatrix A, Vector x)"*. This test method will get a diagonal matrix D as input. Then, it will compute the product of D and a vector x_0 with all ones. The result should be a vector with all the diagonal values. We will check each value of the result we get from *"specialProductAX (SparseMatrix A, Vector x)"*. If any value of the resulting vector is not equal to the corresponding diagonal value, this test method will return "Fail", otherwise, return "Pass".
   Similar to the second test method, we create the third test function to test the method *"specialProductAX (SparseMatrix A, Vector x)"*. The input of this test method is a diagonal matrix D. Then, it will compute the product of D and a matrix with all one's. The result should still be the matrix D. Thu, if there are any difference between the result matrix and the matrix D, this test method will return "Fail", otherwise, it will return "Pass".
   The forth test is for the method *"getDiagonal()"*. The input of this test method is a matrix A and its diagonal matrix D which is returned by the method *"getDiagonal()"*. The result of A-D should be a matrix whose values on the diagonal are all zero's. We compute the second norm, and if the result is less than 10^{-7}, the test method will return "Pass", otherwise, it will return "Fail".
   The final test is for the method *"inverseD (SparseMatrix D)"*. The input of this test methods are a diagonal matrix D and its inverse DI which is returned by the method *"inverseD (SparseMatrix D)"*. The product of D and DI should be a unit diagonal matrix. Thus, we compute the product and the second norm. If the result is less than 10^{-7}, the test method will return "Pass", otherwise, it will return "Fail".

Discussion

We set the threshold of convergence to three different values: $10^{-7}, 10^{-8}$ and 10^{-9}. And the corresponding outputs are in three different files: “output1”, “output2” and “output3”. What we can observe from the results is as following. First, as the threshold getting smaller, we need to iterate more times to reach the final result, which also means it will take more time to get the result. On the other hand, the result is more and more precise (the second norm of the residual vector is smaller and smaller), as the threshold getting smaller.
I also analyze the reasons to these observations. After each iteration, the result should be closer to the correct result. Thus, if we iterate more times, the precision of our result will increase. But each iteration will take some, so the time used to get the final result will also increase when we choose a smaller convergence threshold. I also notice that, after each iteration, the difference between the result of this iteration and the result of the last iteration is getting smaller and smaller. This can explain why we need more iterations to get the final result if we choose a smaller convergence threshold.