**Assignment 5 Final Report**

Yiqi Yu Netid : yy757

Haohan Chen Netid: hc937

1. Goal

* Implement sparse matrix solver using direct solutions and iterative solutions with modules that have independent testing.
* Use the condition number to validate the property of the matrix.
* Compare the performance of different solvers.

2. Top down design

* Sparse Matrix Solvers
  + Sparse Matrix Allocation
    - spCreate
    - spClear
  + Sparse Matrix Builder
    - spAddElment
    - spGetElement
  + Sparse Matrix Solver
    - Direct solver
      * Cholesky decomposition
      * LU decomposition
    - Iterative solver
      * Jacobi
      * SOR
  + Sparse Matrix utilities
    - spProductAx
    - spColPermute
    - spRowPermute
    - spRowScale
    - spMinFillinPivot
    - spCountNonzero
    - normVector
    - spConditionNumber
    - spNormCol
    - spNormRow
    - spGetDiagonal
    - spGetInverseDiagonal
  + Sparse Matrix output
  + Sparse Matrix test

3. Data Structure

Defined in spdefs.h.

* Class Sparse Matrix
  + Pointers : ValPtr, RowPtr, ColPtr

At first, I thought about using vector because vector structure can reallocate memory space automatically, which will be of great convenience for sparse matrix builder. However, in order to allocate the memory space by hand instead of doubling the length of vector by machine, I chose to use pointer.

4. Testing

* Unit testing for four components
  + White box testing
* Integration testing
  + White box testing, Black box testing
* System testing
  + Black box testing
* Wilkinson principle

5. Dataset

Suite Sparse Matrix Collection

6. Conclusion

In the last project, finally understand the top down design and how to implement it. Learnt a lot from reading the legacy code of the sparse matrix. Spent a lot of time doing the documentation of the code, which I think is of great importance for future code reuse.

During the software evolution, get a better understanding of the computational software engineering. Bugs can be anywhere, all kinds of testing are essential. Due to the lack of time, haven’t done enough tests for this program. Regression tests and more black box tests are needed.

For direct solvers, the conditions for Cholesky Decomposition and Gaussian elimination are different. For the Cholesky Decomposition, the matrix has to be symmetric, positive definite matrix, but the condition number doesn’t have to be that small. For the Gaussian elimination, the matrix must be well-conditioned, which has a small condition number. Usually, we create it by making it diagonal dominant.

However, I do find something interesting here. Since the Cholesky Decomposition decomposes a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose, it does not need pivoting while in order to avoid fill in and decrease precision, the gaussian elimination needs pivoting. Pivoting costs a lot. I did full search in the project, which makes the computation speed of gaussian elimination much slower than Cholesky Decomposition. However, if we do not use pivoting for gaussian elimination, then it is more efficient than Cholesky.

As for the iterative solver, Jacobi performs better than SOR in terms of speed and accuracy for the specific test case. I used optimized method for both of them, which use element calculation with formula instead of matrix computation.