ACSE-5: Advanced Programming Assignment

Team Name: Team Inheritor

Date:07/02/2021

1. Code structure

1.1. Description

In this programe, four algorithms are implemented for dense matrix stored in column major ordering, which are Jacobi, Gauss-Seidel, LU factorization and Biconjugate gradient stabilized method respectively. And Gauss_Seidel for CSR matrix and Jacobi for CSR matrix are implemented for the sparse matrix stored in CSR format as comparison. To implement these algorithms and construct or read matrices, methods are added to the original Matrix and CSRMatrix classes, and Interface.cpp Solver.cpp and their corresponding header files are built. The following is a detailed description of the program.

- **'Matrix.h', 'Matrix.cpp'**

The purpose of this file is to generate a class template for dense matrix. This class includes functions to print out values in matrix, generate random positive definite matrix, do matrix-matrix and matrix-vector multiplication. A copy constructor for Matrix<T> is constructed in the original Matrix.cpp to copy the existing matrix. And a void method was added in the class to generate positive definite matrix.

- **'CSRMatrix.h', 'CSRMatrix.cpp'**

Based on the existing CSRMatrix class, sparse matrix with csr format is constructed and calculated. Besides, some methods are added, including the method to convert matrices from CSR format to dense format, generate random positive definite matrix, matrix-matrix mult and a constructor for deep copy.

- **'Solver.h', 'Solver.cpp'**

Solver class includes the detailed implementation of the above algorithms. It has a parameter-free constructor and a destructor. Every method requires the input of matrix A of Matrix<T> type, vector RHS of array type and an initialized solution x which equals to the length of RHS as parameters. And the final results will be stored in array x. For dense matrix, solvers such as dense_Jacobi, dense_Gauss_Seidel, dense_LU, BiCGTAB_are_implemented. For sparse matrix, solvers such as sparse_Gauss_Seidel, sparse_Jacobi are implemented. The purpose of this file is to solve the linear system using different methods.

- **'Interface.h','Interface.cpp'**

The purpose of these files is to generate an interface for users to implement an algorithm to solve the linear system Ax=b. In the interface, users are able to generate or load matrices and choose different methods to solve the corresponding linear systems.

- **'cblas.h', 'cblas_mangling.h'**

These files contains the configuration code for CBLAS.

1.2. Strength and weakness

- **Strength**
  - All the matrices are traversed by column major, achieving faster calculation.
  - Varied solvers are provided. In this way, users can choose any solver depending on their preference.
  - The input can be the positive define matrices randomly generated by our program or read from a '.txt' file.
  - Users can run a single algorithm or multiple algorithms simultaneously on a set of generated or known data to gain feedback.
  - The level-1 function in CBLAS package is used to optimize the algorithm and accelerate the calculation speed.

- **Weakness**
  - Occassion that matrix is stored banded is not be implemented.
  - There are strict formatting requirements for users to choose to read .txt files.

2. Input and output

2.1. Input

The input can be the positive define matrices generated by our program or loaded from a '.txt' file. The matrices can be stored densely or sparsely depending on users' preference.

2.2. Output

To analyze the output, matrices of different sizes are generated as input. A solver is considered 'well-performed' if the operating time is short and the absolute L2 error is small. Below are the tables of output.

For dense matrix:

size	Gauss-seidal (time, error)	BiCGSTAB (time, error)	LU-factorization (time, error)	Jacobi (time, error)
20 * 20	0.051, 3.1e-6	0.330, 2.7e-13	0.026, 7.98e-26	0.038, 1.4e-6
50 * 50	0.162, 3.1e-7	0.957, 7.7e-13	0.255, 7.5e-25	0.129, 9.7e-7
100 * 100	0.972, 4.3e-7	6.8, 7.5e-13	1.49, 7.5e6	0.659, 3.7e-6
200 * 200	8.864, 3.5e-6	60.205, 7.2e-13	13.117, 1.7e7	10.484, 1.6e-5
500 * 500	738, 4e68	2087,17.1	197, 4e7	517, nan

Here are some key comments from the table:

- Generally speaking, Jacobi method can solve a linear system in the shortest time.
- As for accuracy, BiCGSTAB method performs the best since it has smallest errors. However, this method is the most time-consuming.
- Sometimes, LU-factorization can provide the most accurate solutions. But its performance is quite unstable.
- All the method are suitable for matrix A whose size is smaller than 50 * 50.

For sparse matrix:

size	Gauss-seidal (time, error)	Jacobi (time, error)
20 * 20	0.211, 2.5e-6	0.229, 2.8e-5

50 * 50	0.231, 6.6e-6	0.246, 7.1e-5
100 * 100	0.247, 1.2e-6	0.275, 8.5e-5
200 * 200	0.240, 2.7e-5	0.265, 1.1e-4
500 * 500	0.338, 7.2e-6	0.388, 1.1e-4

Here are some key comments from the table:

- Generally speaking, Gauss-seidal method performs better since its time is shorter and error is smaller.
- Compared to the solvers for dense matrix, solvers for sparse matrix are less time consuming and suitable for larger matrices.

3. Execution

To execute the program, please refer to the guide in 'ReadMe.md'.

4. Other information

Github Link: https://github.com/acse-2020/group-project-team-inheritor

Members: Jin Yu, Xuyuan Chen, Yuchen Wang

Breakdown of work:

Jin Yu: solver coding, matrix generating, interface coding, 'ReadMe.md' writing

Xuyuan Chen: solver coding, interface coding, report writing

Yuchen Wang: solver coding, report writing

5. Reference

Van der Vorst, H. A. (1992). "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems". *SIAM J. Sci. Stat. Comput.* **13** (2): 631–644. doi:10.1137/0913035. hdl:10338.dmlcz/104566.