Group EDSML1
28th January 2022

Advanced Programming Methods - Group Project

Matrix library, solving the linear system Ax = b

Section 1: Design and Motivation

The group has implemented a Matrix library that solves the linear system Ax = b using two methods:

1. Gaussian elimination with partial pivoting (Direct method)

    - The method first transforms matrix A and vector b in upper-triangular forms (i.e. lower triangle of matrix A is filled with 0), then computes back substitution

2. Jacobi method (Iterative method)

The algorithms return the vector x result to the user. The Matrix class builds upon the Matrix.cpp and Matrix.h files from the Lecture 7 file.

For the purpose of having a more robust user experience, the group has implemented only one class, called *Matrix,* which contains the following functions which build up from lecture 7 files:

- Matrix constructor
- Matrix destructor
- printValues(): print out the values of Matrix in one single row
- printMatrix(): print out the Matrix by using the rows, cols coordinates
- printTextFile(): print the result of vector x into a text file
- gauss_elimination(): function that calculates vector x based on Gaussian elimination with partial pivoting method
- jacobi(): function that calculates vector x based on Jacobi method
- computeResidual(): function that calculates the residual (Ax - b), which can then be compared to the tolerance level for Jacobi method
- testResultJacobi(): function that tests whether Ax - b < tolerance level for Jacobi method
- testResultGauss(): function that tests whether Ax - b ~ 0 for Gaussian method
- matVecMult(): function that multiplies a matrix and a vector
- matMatMult(): function that multiplies two matrices
- matMatAdd(): function that adds to matrices
- matMatSubtract(): function that subtracts one matrix from another matrix

Section 1.1: Assumptions

For the purpose of the implementation, the group has made the following assumptions:

- A is a dense, real positive definite matrix
- In the Jacobi method, maximum number of iterations is 1000
- In the main.cpp file, user needs to include the readDoubles function for the input
- The user must follow clear guidelines for file input of matrix A/vector b
- The user will define vector b and vector x as matrices with cols = 1 (design created for ease-of-use of one single class *Matrix*)

Group EDSML1
28th January 2022

Section 1.2: Algorithm Choice

The group has selected the above-mentioned two algorithms to explore one direct method and one iterative method. The group has implemented partial pivoting within Gaussian elimination in order to avoid division by zero and rounding errors.

Section 2: Validation

The group has validated the algorithms through two methods:

1. Cross-checking the algorithm results of Jacobi method and Gaussian elimination with partial pivoting method against Python results, computed by utilising L5_linear_solvers Python code. Please find more details and example validations in the folder 'Validation' in the GitHub repository. The files called 'method_testx.txt' are the input files, while the files called 'method_testx_output.txt' are the output files. Those output files have been compared against the .ipynb output results (using Python code).
2. Implementation of testResult function in the Matrix class, which computes the remainder of the calculation, as defined:

   Remainder = Ax - b

   The function then asserts whether the remainder is either:

a) Less than the tolerance level (in the case of Jacobi method)
b) Equal to 0, by using rounding (in the case of Gaussian elimination with partial pivoting method)

Section 2.1: Performance

Table 1. Time taken to compute results by matrix size and method in milliseconds:

Matrix Size	Gaussian Elimination	Jacobi
4 x 4	0.066	0.775
6 x 6	0.054	2.582
8 x 8	0.028	46.955
10 x 10	0.063	2.6
14 x 14	0.016	75.049

The Jacobi iterative method works correctly with well-conditioned linear systems. However, If the linear system is ill-conditioned (condition number is >1), the Jacobi method fails to converge (Hansen, 1963). Jacobi method only works if matrix A is **strictly diagonally dominant.** In other words, this condition must be satisfied:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|.$$

Condition numbers can be checked in python with: np.linalg.cond(matrix), however it is not possible to randomly generate a matrix with a predefined condition number. It is possible to lower the condition number (for random matrices only) by multiplying matrix by its transpose:

A = B * B.T

Section 3: User Experience

In the GitHub repository, the group has added a main_test.cpp file, as well as sample test_file.txt and test_file.csv input files, to serve as an example of how the Matrix library can be used.

The user will download Matrix.h, Matrix.cpp and main_test.cpp files in the same location and based on the IDE, they can input the following command to compile the files:

VS Code: *g++ Matrix.cpp main_test.cpp -o main*; to call the library: *./main*

VS Community: add all files in the same project, click on Build Solution and Start Debugging

Please note that the input files (either .txt or .csv) will need to be in the same location as the .h, .cpp files.

Please find below a step-by-step description:

- The user chooses to enter either a text file/csv file or can manually input the matrix and vector (K for keyboard, F for file)
- If choosing the manual input, the user then enters the number of rows and columns, then proceeds to enter the matrix with one row at a time (e.g. 8 4 12 3 4 for row 0) as well as the vector in a similar manner.
- The user then chooses if they wish to use the Gaussian Elimination method or the Jacobi method (G keyboard input for Gaussian, J keyboard input for Jacobi)
- For the Jacobi method, the user can then choose the tolerance level
- In the case of Jacobi, the tolerance can be entered, and the output will be printed
- In the case of Gaussian, the output is directly printed without any further parameter setting
- If the user wishes to, the result can be exported to a text file (Y keyboard input for yes, N keyboard input for no).

The process by which the user should decide on the parameters for the solvers are further described step by step in figure 1 available on the repository readme.md file.

Section 4: Strengths, Opportunities and Future Extensions

The Matrix library is intuitive to use by users. The user can input both .txt (separated by comma) or .csv files. While the group has tested the performance of both algorithms using <ctime> library, the performance can be further enhanced. For future work, the group can explore implementing methods for sparse matrices, as well as other direct method algorithms and/or iterative method algorithms (such as Conjugate Gradient, Gauss Seidel etc.) Additionally, the group can further expand on user experience and functionality of the Matrix class. The group would also like to implement smart pointers as an extension.

Group EDSML1
28th January 2022

Section 5: General Group Information

Group Name: **EDSML1**

GitHub Repository link: https://github.com/acse-2020/group-assignment-edsml1

Team Member	GitHub ID	OS	Compiler	IDE	Worked on
Raluca Gaina	edsml-rig21	macOS Big Sur (Version 11.3.1)	Clang (Version 12.0.5)	Visual Studio Code (Version 1.63.2)	Gaussian elimination method with partial pivoting (gauss_elimination) User experience (main_test_template.cpp file, including the readDoubles function) – implementation of user file or keyboard input Test result functions (testResult) – calculation of remainder and comparing against a tolerance level Algorithm performance calculation using <ctime> Report writing
Olga Kostur	edsml-ok121	Windows 10, 64-bit	C/C++ Optimising Compiler Version 19.30.30706 for x86	Visual Studio Community 2022	Jacobi method (jacobi function) computeResidual function Implementation of tolerance level into the Jacobi method functionality Report writing
Emma Schoenmakers	edsml-de1052e8	Windows 10, 20H2	C/C++ Optimising Compiler	Visual Studio Community 2022	Jacobi method (jacobi function) Validation of algorithm results (writing of Python sample tests and files) Report writing

Section 6: Appendix and References

Gauss Elimination implementation: L5_linear_solvers Lecture (Python code)

Jacobi Method implementation: L5_linear_solvers Lecture (Python code)

Matrix.h and Matrix.cpp files from Lecture 7

Hansen, E.R., 1963. On cyclic Jacobi methods. *Journal of the Society for Industrial and Applied Mathematics*, *11*(2), pp.448-459.