

AMS 209: Project Report

Numerical Linear Algebra

Gaussian Elimination with and without Pivoting for Solving Linear Systems

Geetanjali Rakshit

December 15, 2017

1 Abstract

The aim of this project is to solve a system of linear equations using Gaussian Elimination with and without pivoting. The project has been implemented in *Fortran* and *Python*. A linear algebra solver has been implemented in Fortran, while a run setup, run scheduler, and a data visualizer have been implemented in Python. We solve for 3 sets of linear equations and present our solutions. Also, we compare the results obtained from the Fortran solver with the ones obtained by using the numpy libraries in Python.

2 Method

The pseudo-code for the two algorithms, Gaussian Elimination with and without pivoting, is described in the Reading Material for the project (1). The code has two parts: the Fortran part in the directory **"project/LinAlg"** and the Python part in the directory **"project/pyRun"**.

2.1 Fortran Implementation

The Fortran implementation reads the values of the matrix A and vector b from input files $A_i.dat$ and $b_i.dat$, respectively. It then prints these values to the screen. Next, we apply Gaussian Elimination with or without partial pivoting to decompose the matrix A into lower L and upper U triangular matrices such that $A = LU$, $Ly = b$ and $Ux = y$. After the matrices L and U are computed, we solve $Ly = b$ and $Ux = y$ to obtain the solution x . This solution is then written to the screen and to a file called $x_i.dat$.

The implementation has a main driver routine called **linear_solve** in the directory **"project/LinAlg"**, which calls other subroutines. The code is organized as follows:

- **linear_solve.f90**: This is the main driver routine, which calls the following subroutines:
 - **read_data.f90**: This reads in A and b from $.dat$ files.

- `setup_module.f90`: This reads the parameters *eq_{set}* (can be 1, 2 or 3) and matrix dimension *n* from a file *setup.init*.
- `write_to_screen.f90`: This writes both *A* and *b* to screen.
- `LU_decomp.f90`: This implements Gaussian elimination with and without partial pivoting.
- `forward_solve.f90`: This solves $Ly = b$.
- `backward_solve.f90`: This solves $Ux = y$.
- `write_data.f90`: This outputs the result onto screen as well as to a file, *x_i.dat*.

2.2 Python Implementation

The Python implementation initializes a system of linear equations into 2 variables: a matrix *A* and a vector *b*, and writes them to files *A_i.dat* and *b_i.dat*, respectively, for each *i*. Then, it compiles the Fortran code and runs it. After the Fortran solver runs, we check for its correctness by comparing its solution with the solution from the Python *numpy* library. A threshold value is used to quantify the error between the two solutions. Finally, the implementation produces plots for the matrix *A*, and vectors *x* and *b*.

The **pyRun.py** file is in the directory **project/PyRun**. The code is organized as follows:

- `make_make()`: This function compiles the Fortran code.
- `setValues()`: This function initializes the matrix *A* and the vector *b*, depending on the set of equations (1, 2 or 3), for which the algorithm is to run.
- `createFiles()`: This function writes the matrix *A* and the vector *b* to *A_i.dat* and *b_i.dat* files, for each *i*, and the *setup.init* file with parameters such as the set of equations (1, 2 or 3) and the matrix dimension *n*.
- `runFortranSolver()`: This function runs the Fortran executable file.
- `readFortranSolution()`: This function returns the solution from the Fortran code written in a file *x_i.dat*.
- `checkSolution()`: This function compares the solution of the Fortran code with the one computed by the *numpy* library. It prints PASS if the solutions are the same up to a threshold, else prints FAIL, and outputs the error together with the Fortran and Python solutions to screen.
- `plotData()`: This function plots the matrix *A* and vectors *b* and *x*.
- `runScheduler()`: This function is the scheduler that runs all the steps sequentially, i.e., initialize, run Fortran code, compare with numpy solution and plot the values.

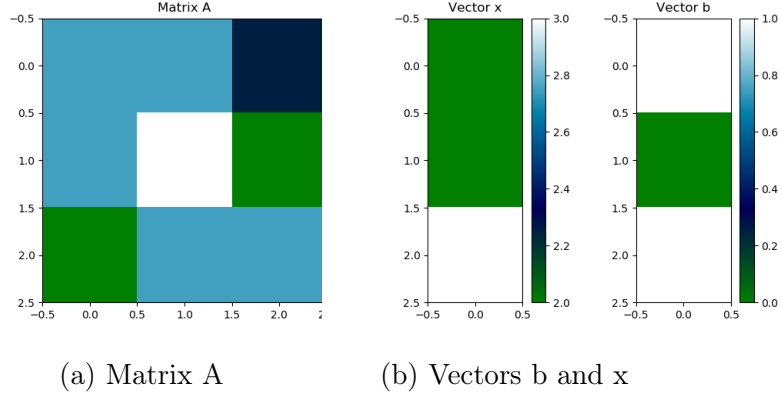


Figure 1: 1st set of equations

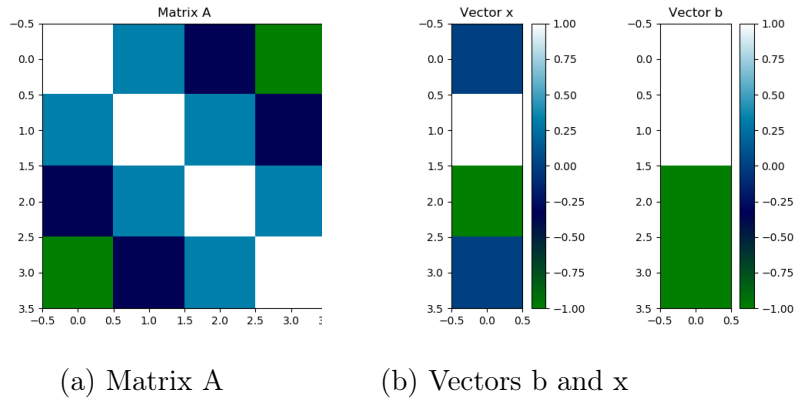


Figure 2: 2nd set of equations

3 Results

We solve for 3 sets of equations, as provided in the Project Handout (2). The equations and the obtained solutions (value of x 's) are listed in equations 1, 2 and 3.

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, x = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} \quad (1)$$

$$A = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}, x = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \quad (2)$$

$$A = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 3 & -3 & 3 \\ 2 & -4 & 7 & -7 \\ -3 & 7 & -10 & 14 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 2 \\ -2 \\ 8 \end{bmatrix}, x = \begin{bmatrix} 1 \\ 1 \\ -3 \\ -3 \end{bmatrix} \quad (3)$$

We plot the matrix A and the vectors b and x for equations 1, 2 and 3 in Figures 1, 2 and 3, respectively.

Gaussian elimination with and without pivoting produces the same results. These values of x were the same as those calculated by the numpy library in Python and shown as x in the equations 1, 2 and 3.

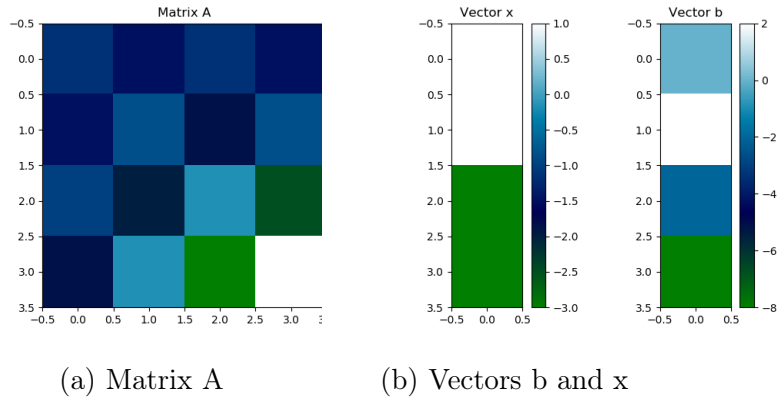


Figure 3: 3rd set of equations

4 Findings and Comments

Gaussian elimination without pivoting worked fine for the system of equations on which we tested. This method doesn't work when a diagonal element of the matrix becomes zero. Gaussian elimination with partial pivoting is a more robust method due to the row permutations.

5 Conclusion

We successfully solved the set of linear equations provided to us using the Fortran linear solver, both with and without pivoting using the Gaussian Elimination method. These results were the same as those produced by the Python numpy library.

6 References

1. Reading material: https://users.soe.ucsc.edu/~dongwook/wp-content/uploads/2017/ams209/lectureNote/_build/html/project/finalproject.html#final-term-projects
2. Project Handout: https://users.soe.ucsc.edu/~dongwook/wp-content/uploads/2017/ams209/lectureNote/_build/html/project/finalproject.html#final-term-projects