CSC1003 Assignment 2

CSC1003

October 25, 2023

Important Notes

1. The assignment is an individual project, to be finished on one's own effort.
2. The work must be submitted before 6pm Nov. 7th, 2023 (Tuesday), Beijing Time. This is a firm deadline. No late submissions are accepted.
3. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, $30\% - 100\%$ marks will be deducted.
4. This assignment is supposed to be finished after learning arrays. Meanwhile, a draft may be released earlier for students' preparation purpose.
5. Please let the teaching team know for any ambiguity or incorrectness in this draft.

Marking Criterion

1. The full score of the assignment is 100 marks.
2. Two java programs are to be submitted. Each program will be evaluated with several unseen test cases. A submission obtains the full score if and only if both programs pass all test cases.

Running Environment

1. The submissions will be evaluated in the OJ system running Java SDK. It is the students' responsibility to make sure that his/her submissions are compatible with the OJ system.
2. The submission is only allowed to import four packages of (java.lang.*; java.util.*; java.math.*; java.io.*) included in Java SDK. No other packages are allowed.
3. All students will have an opportunity to test their programs in the OJ platform prior to the official submission.

Submission Guidelines

1. You will get your grade only if you submit your code both on OJ and on bb on time. An n-day late submission on bb leads to $n * 10\%$ mark deduction, and any late submission on OJ leads to minimum grade.
2. For bb submission, you need to **directly** upload your java file on bb. That is, your submission should be *GaussianElimination.java*, or *MatrixMultiplication.java*, or both. Wrong submission format will receive 10% mark deduction.
3. Inconsistency with or violation from the guideline leads to marks deduction.

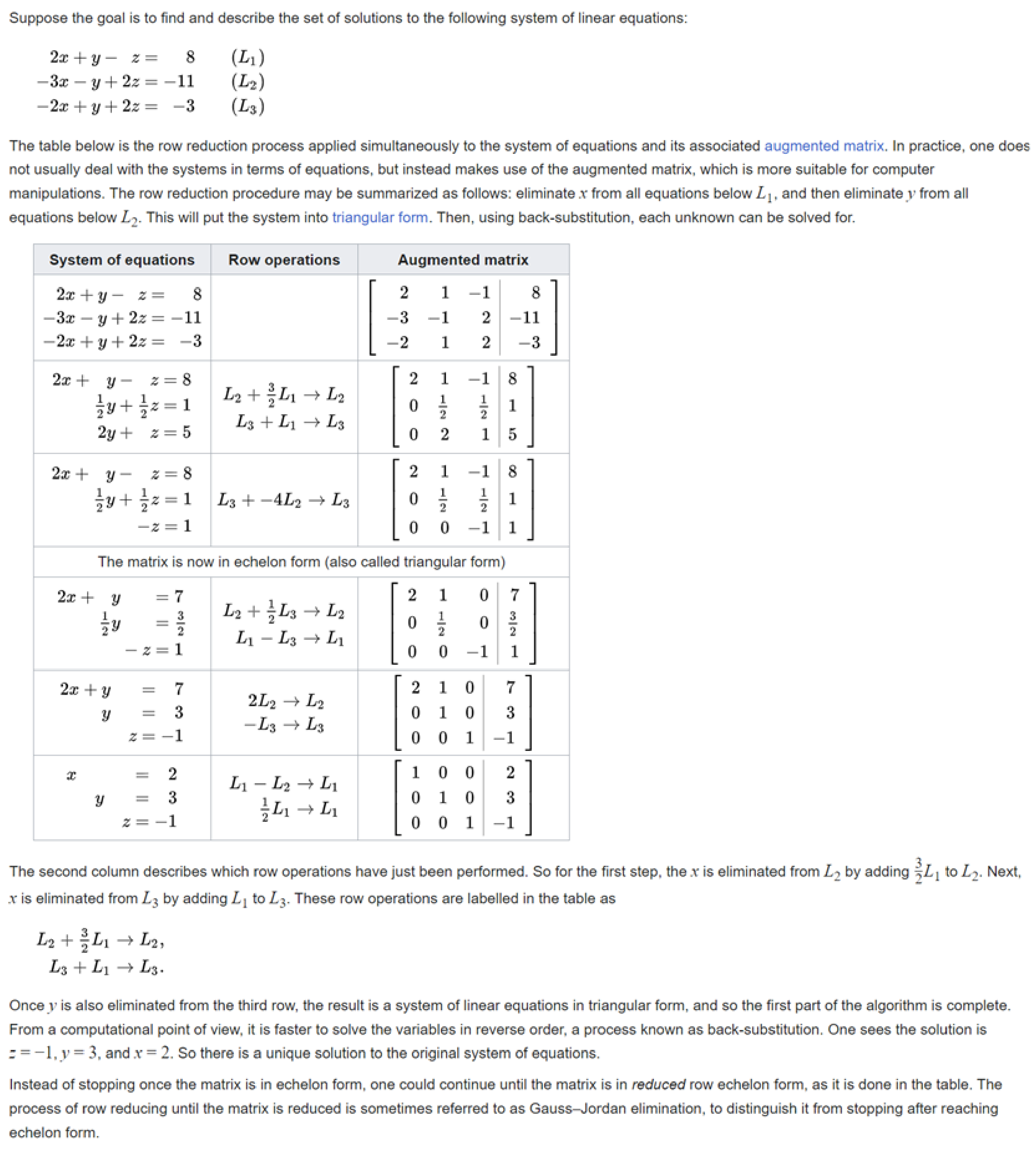
4. All students are reminded to read this assignment document carefully and in detail. No argument will be accepted on issues that have been specified in this document.

Program

You need to choose either "Program A" or "Program B" to finish. If you finish both, the higher score will be included in the overall grade.

A

In mathematics, Gaussian elimination, also known as row reduction, is an algorithm for solving systems of linear equations. It consists of a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to compute the rank of a matrix, the determinant of a square matrix, and the inverse of an invertible matrix. The method is named after Carl Friedrich Gauss (1777–1855). (see, https://en.wikipedia.org/wiki/Gaussian_elimination)



Suppose the goal is to find and describe the set of solutions to the following system of linear equations:

$$\begin{aligned} 2x + y - z &= 8 & (L_1) \\ -3x - y + 2z &= -11 & (L_2) \\ -2x + y + 2z &= -3 & (L_3) \end{aligned}$$

The table below is the row reduction process applied simultaneously to the system of equations and its associated augmented matrix. In practice, one does not usually deal with the systems in terms of equations, but instead makes use of the augmented matrix, which is more suitable for computer manipulations. The row reduction procedure may be summarized as follows: eliminate x from all equations below L_1, and then eliminate y from all equations below L_2. This will put the system into triangular form. Then, using back-substitution, each unknown can be solved for.

System of equations	Row operations	Augmented matrix
$\begin{aligned} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned}$		$\left[\begin{array}{rrr\|r} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$
$\begin{aligned} 2x + y - z &= 8 \\ \tfrac{1}{2}y + \tfrac{1}{2}z &= 1 \\ 2y + z &= 5 \end{aligned}$	$L_2 + \tfrac{3}{2}L_1 \to L_2$ $L_3 + L_1 \to L_3$	$\left[\begin{array}{rrr\|r} 2 & 1 & -1 & 8 \\ 0 & \tfrac{1}{2} & \tfrac{1}{2} & 1 \\ 0 & 2 & 1 & 5 \end{array}\right]$
$\begin{aligned} 2x + y - z &= 8 \\ \tfrac{1}{2}y + \tfrac{1}{2}z &= 1 \\ -z &= 1 \end{aligned}$	$L_3 + -4L_2 \to L_3$	$\left[\begin{array}{rrr\|r} 2 & 1 & -1 & 8 \\ 0 & \tfrac{1}{2} & \tfrac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$
The matrix is now in echelon form (also called triangular form)		
$\begin{aligned} 2x + y &= 7 \\ \tfrac{1}{2}y &= \tfrac{3}{2} \\ -z &= 1 \end{aligned}$	$L_2 + \tfrac{1}{2}L_3 \to L_2$ $L_1 - L_3 \to L_1$	$\left[\begin{array}{rrr\|r} 2 & 1 & 0 & 7 \\ 0 & \tfrac{1}{2} & 0 & \tfrac{3}{2} \\ 0 & 0 & -1 & 1 \end{array}\right]$
$\begin{aligned} 2x + y &= 7 \\ y &= 3 \\ z &= -1 \end{aligned}$	$2L_2 \to L_2$ $-L_3 \to L_3$	$\left[\begin{array}{rrr\|r} 2 & 1 & 0 & 7 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$
$\begin{aligned} x &= 2 \\ y &= 3 \\ z &= -1 \end{aligned}$	$L_1 - L_2 \to L_1$ $\tfrac{1}{2}L_1 \to L_1$	$\left[\begin{array}{rrr\|r} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$

The second column describes which row operations have just been performed. So for the first step, the x is eliminated from L_2 by adding $\tfrac{3}{2}L_1$ to L_2. Next, x is eliminated from L_3 by adding L_1 to L_3. These row operations are labelled in the table as

$$\begin{aligned} L_2 + \tfrac{3}{2}L_1 &\to L_2, \\ L_3 + L_1 &\to L_3. \end{aligned}$$

Once y is also eliminated from the third row, the result is a system of linear equations in triangular form, and so the first part of the algorithm is complete. From a computational point of view, it is faster to solve the variables in reverse order, a process known as back-substitution. One sees the solution is $z = -1$, $y = 3$, and $x = 2$. So there is a unique solution to the original system of equations.

Instead of stopping once the matrix is in echelon form, one could continue until the matrix is in *reduced* row echelon form, as it is done in the table. The process of row reducing until the matrix is reduced is sometimes referred to as Gauss–Jordan elimination, to distinguish it from stopping after reaching echelon form.

You are required to write a Java program that reads the input of an augmented matrix, solve the associated system of linear equations, and then output the solution vector. For the example in Fig. 1, the corresponding input and output are given in the following table.

An example of console input	Expected console output

3
 2.0 1.0 -1.0 8.0
-3.0 -1.0 2.0 -11.0
-2.0 1.0 2.0 -3.0
	2.000 3.000 -1.000

Note

1. The first line is a positive integer, giving the number of unknown variables of the linear equations. Let the number be n.
2. Starting from the second line, each line of the input consists of n double number separated by spaces.
3. The output is expected to be n floating point number rounded to three decimal places, giving the solution of unknown variables.
4. The template for reading input and output has been provided in *GaussianElimination.java*. Please strictly follow the template if you are not familiar with the output format. We have also provided the sample input and output separately in *in_A.txt* and *in_A.txt*. You can refer to them and use them to test your program. For command line arguments, just type
java GaussianElimination < in_A.txt
5. We promise that the input linear equations have a unique solution. That is, you don't need to consider the situation the matrix doesn't have or have multiple solutions. Also, we promise that you don't need to do any row interchange.

B

Write a java program named "MatrixMultiplication.java" to calculate the product of two matrices of sizes m×n and n×p respectively. Each element of the input matrices is a complex number. (For def. of complex number, see https://brilliant.org/wiki/complex-numbers/) Therefore, the result matrix is an m×p matrix with complex elements.
In this program, each input complex number is in the form $a + b * i$, where a, b are two positive integer smaller than 1000. For example, $3 + 89 * i, 0 + 2 * i, 3 + 0 * i$.

An example of console input	Expected console output
2 3 5 1+2*i 2+3*i 3+4*i 1+3*i 2+6*i 5+2*i 3+4*i 7+2*i 5+4*i 3+8*i 6+3*i 5+4*i 3+1*i 7+3*i 7+7*i 9+3*i 6+3*i 3+7*i 8+3*i 6+3*i 4+4*i	-1+66*i -13+60*i 14+82*i -14+82*i 5+76*i 1+78*i 2+84*i 23+98*i -25+100*i 9+109*i

Note

1. The first line of the input is the size of the two input matrices. In this example, $m = 2, n = 3, p = 5$, each separated by a space. The following m lines give the elements of the first matrix, and each line has n space-separated complex numbers, which form the first input matrix. Then the following n lines give the elements of the second matrix, and each line has p complex numbers, which form the second input matrix.

2. The output contains m lines, each line having p space-separated complex numbers, giving the elements of the result matrix. We have provided the sample input and output separately in *in_B.txt* and *out_B.txt*. You can refer to them and use them to test your program. For command line arguments, just type $java\ MatrixMultiplication\ <\ in\_B.txt$

3. The definition of matrix and matrix multiplication operation can be found in any linear algebra textbook, or from the following link: https://en.wikipedia.org/wiki/Matrix_multiplication

4. When testing the assignment, the values of m, n, p are less than 500.

OJ Access Code

8uh&2s9plm