Process model formulation and solution, 3E4

Assignment 2

Kevin Dunn, dunnkg@mcmaster.ca **October 2010**

Solutions prepared by both Ali and Elliot.

Question 1 [1]

The Gauss elimination method is an effective algorithm for solving a set of linear equations, $Ax = b$. Solve this set of equations by hand, showing the steps taken.

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 0 \\ x_1 - 2x_2 - 2x_3 - 2x_4 = 4 \\ x_1 + 4x_2 - 4x_3 + x_4 = 2 \\ x_1 - 5x_2 - 5x_3 - 3x_4 = -4 \end{cases}$$

Solution

We start by converting the system of equations to the form $Ax = b$.

$$\begin{array}{rcrcrcrcl} x_1 & + & x_2 & + & x_3 & + & x_4 & = & 0 \\ x_1 & - & 2x_2 & - & 2x_3 & - & 2x_4 & = & 4 \\ x_1 & + & 4x_2 & - & 4x_3 & + & x_4 & = & 2 \\ x_1 & - & 5x_2 & - & 5x_3 & - & 3x_4 & = & -4 \end{array} \quad \Longrightarrow \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & -2 \\ 1 & 4 & -4 & 1 \\ 1 & -5 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 2 \\ -4 \end{bmatrix}$$

Forward elimination

Recall that the basic algorithm for the forward elimination step of naive Gauss elimination is:

1. Divide the current row i by the ith diagonal element.
2. Eliminate all elements in the ith column below the ith diagonal element (i.e. rows $j > i$) by subtracting n times the ith row from the jth row (where n is the current value of value in the ith column of the jth row, i.e. element (i , j)). Do not forget to subtract the b vector elements as well.
3. Move to the next row.
4. Repeat steps 1 - 3 until the A matrix is upper triangular (i.e. you have eliminated all elements below the diagonal).

Therefore our first step is to divide row 1 by the it's diagonal element (in this case the first value), which is equal to 1 (easy enough)

$$\begin{bmatrix} \frac{1}{(1)} & \frac{1}{(1)} & \frac{1}{(1)} & \frac{1}{(1)} \\ 1 & -2 & -2 & -2 \\ 1 & 4 & -4 & 1 \\ 1 & -5 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \frac{0}{(1)} \\ 4 \\ 2 \\ -4 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & -2 \\ 1 & 4 & -4 & 1 \\ 1 & -5 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 2 \\ -4 \end{bmatrix}$$

Next we subtract row 1 from rows 2, 3, and 4 using n values equal to the coefficients in the first column of each row. In thise case the all the column coefficients are equal to 1, which makes our lives easy again.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1-(1)(1) & -2-(1)(1) & -2-(1)(1) & -2-(1)(1) \\ 1-(1)(1) & 4-(1)(1) & -4-(1)(1) & 1-(1)(1) \\ 1-(1)(1) & -5-(1)(1) & -5-(1)(1) & -3-(1)(1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 4-(1)(0) \\ 2-(1)(0) \\ -4-(1)(0) \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 3 & -5 & 0 \\ 0 & -6 & -6 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} =$$

Having eliminated all elements in the first column, we move to the second row/column. Once again we start by dividing the current row (row 2) by its diagonal element.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \frac{-3}{(-3)} & \frac{-3}{(-3)} & \frac{-3}{(-3)} \\ 0 & 3 & -5 & 0 \\ 0 & -6 & -6 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{4}{(-3)} \\ 2 \\ -4 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & -5 & 0 \\ 0 & -6 & -6 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -4/3 \\ 2 \\ -4 \end{bmatrix}$$

Next we eliminate all elements below row 2 in column 2 by subtracting 3 times row 2 from row 3 and (-6) times row 2 from row 4.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 3-(3)(1) & -5-(3)(1) & 0-(3)(1) \\ 0 & -6-(-6)(1) & -6-(-6)(1) & -4-(-6)(1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -4/3 \\ 2-(3)(-4/3) \\ -4-(-6)(-4/3) \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & -8 & -3 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Moving to row 3 we notice that the previous elimination step removed all values below the 3rd column diagonal. Therefore all we do is divide row 3 by its diagonal element and move on.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & \frac{-8}{(-8)} & \frac{-3}{(-8)} \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -4/3 \\ \frac{6}{(-8)} \\ -12 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 3/8 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -4/3 \\ -3/4 \\ -12 \end{bmatrix}$$

Similarly, since row 4 is the final row of our matrix, there exist no elements under the diagonal to eliminate so we simply divide through by the row diagonal element and then move on to perform backwards substitution.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 3/8 \\ 0 & 0 & 0 & \frac{2}{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -4/3 \\ -3/4 \\ \frac{-12}{(2)} \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 3/8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -4/3 \\ -3/4 \\ -6 \end{bmatrix}$$

Backward substitution

Recall that the basic algorithm for the backwards substitution step of naive Gauss elimination is:

1. Eliminate all elements in the ith column above the ith diagonal element (i.e. rows $j < i$) by subtracting n times the ith row from the jth row (where n is the current value of value in the ith column of the jth row, i.e. element (i , j)). Do not forget to subtract the b vector elements as well.
2. Move to the next row.
3. Repeat steps 1 - 2 until the A matrix becomes the identity matrix (i.e. you have eliminated all elements above the diagonal).

Therefore we start by subtracting 1 times row 4 from row 1, 1 times row 4 from row 2, and (3/8) times row 4 from row 3.

$$\begin{bmatrix} 1 & 1 & 1 & 1-(1)(1) \\ 0 & 1 & 1 & 1-(1)(1) \\ 0 & 0 & 1 & 3/8-(3/8)(1) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0-(1)(-6) \\ -4/3-(1)(-6) \\ -3/4-(3/8)(-6) \\ -6 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 14/3 \\ 3/2 \\ -6 \end{bmatrix}$$

Moving to next row, in order to eliminate all elements above the 3rd diagonal element we subtract 1 times row 3 from row 1 and 1 times row 3 from row 2.

$$\begin{bmatrix} 1 & 1 & 1-(1)(1) & 0 \\ 0 & 1 & 1-(1)(1) & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6-(1)(3/2) \\ 14/3-(1)(3/2) \\ 3/2 \\ -6 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 9/2 \\ 19/6 \\ 3/2 \\ -6 \end{bmatrix}$$

Finally, we move to row 2 and eliminate all elements above the 2nd diagonal element by subtracting 1 times row 2 from row 1.

$$\begin{bmatrix} 1 & 1-(1)(1) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 9/2-(1)(19/6) \\ 19/6 \\ 3/2 \\ -6 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4/3 \\ 19/6 \\ 3/2 \\ -6 \end{bmatrix}$$

Therefore the solution, arrived at using naive Gauss elimination, is:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4/3 \\ 19/6 \\ 3/2 \\ -6 \end{bmatrix}$$

Question 2 [2]

1. Derive, by hand, the lower-triangular matrix L (with ones on the diagonal), and upper triangular matrix U so that $A = LU$, using the matrix A from question 1. Afterwards, show that $A = LU$.
2. Use MATLAB or Python's built-in functions to solve for L and U. Show your code and show the matrices from the software. Explain any disagreement. See the software tutorial on matrix operations for help.
3. Use **your** matrices for L and U from the first part of this question, and calculate the inverse of A. Compare your inverse with the inverse calculated by MATLAB or Python - does it agree with the software's values? Also, does $AA^{-1} = I$ for your inverse?

Solution

1. The algorithm for determining the LU decomposition of a matrix (without pivoting) is covered on slides 27 - 34 of the *Section B: Linear Algebraic Equations* Notes. We start with the matrix A:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & -2 \\ 1 & 4 & -4 & 1 \\ 1 & -5 & -5 & -3 \end{bmatrix}$$

The basic procedure behind LU decomposition is to perform forward elimination on the matrix A to generate the upper triangular matrix U and, while doing so, also store the row elimination factors used to generate the lower triangular matrix L.

To eliminate the first column of A we add $-\left(\frac{a_{21}}{a_{11}}\right) = -\left(\frac{1}{1}\right)$ times row 1 to row 2, $-\left(\frac{a_{31}}{a_{11}}\right) = -\left(\frac{1}{1}\right)$ times row 1 to row 3, and $-\left(\frac{a_{41}}{a_{11}}\right) = -\left(\frac{1}{1}\right)$ times row 1 to row 4. Also recall that when we collect these multiplication terms into our L we take the negative of the value (i.e. we multiply by negative 1).

$$A^{(1)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1+(-1)(1) & -2+(-1)(1) & -2+(-1)(1) & -2+(-1)(1) \\ 1+(-1)(1) & 4+(-1)(1) & -4+(-1)(1) & 1+(-1)(1) \\ 1+(-1)(1) & -5+(-1)(1) & -5+(-1)(1) & -3+(-1)(1) \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 3 & -5 & 0 \\ 0 & -6 & -6 & -4 \end{bmatrix}$$

$$L^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{21}}{a_{11}} & 1 & 0 & 0 \\ \frac{a_{31}}{a_{11}} & 0 & 1 & 0 \\ \frac{a_{41}}{a_{11}} & 0 & 0 & 1 \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Next we move to column 2. To remove the elements below the diagonal we add $-\left(\frac{a_{32}}{a_{22}}\right) = -\left(\frac{3}{-3}\right)$ times row 2 to row 3 and $-\left(\frac{a_{42}}{a_{22}}\right) = -\left(\frac{-6}{-3}\right)$ times row 2 to row 4. We also add the negative of these factors to the second column of our L matrix.

$$A^{(2)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 3+(1)(-3) & -5+(1)(-3) & 0+(1)(-3) \\ 0 & -6+(-2)(-3) & -6+(-2)(-3) & -4+(-2)(-3) \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 0 & -8 & -3 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$L^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & \frac{a_{32}}{a_{22}} & 1 & 0 \\ 1 & \frac{a_{42}}{a_{22}} & 0 & 1 \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$$

Since $A^{(2)}$ is already in upper triangular form we no longer have any reason to continue the forward elimination process. Therefore:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 0 & -8 & -3 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Next we prove that $A = LU$. Since matrix multiplications takes up alot of space we will do this column by column.

$$a_{:1} = \begin{bmatrix} l_{11}u_{11} + l_{12}u_{21} + l_{13}u_{31} + l_{14}u_{41} \\ l_{21}u_{11} + l_{22}u_{21} + l_{23}u_{31} + l_{24}u_{41} \\ l_{31}u_{11} + l_{32}u_{21} + l_{33}u_{31} + l_{34}u_{41} \\ l_{41}u_{11} + l_{42}u_{21} + l_{43}u_{31} + l_{44}u_{41} \end{bmatrix} = \begin{bmatrix} (1)(1) + (0)(0) + (0)(0) + (0)(0) \\ (1)(1) + (1)(0) + (0)(0) + (0)(0) \\ (1)(1) + (-1)(0) + (1)(0) + (0)(0) \\ (1)(1) + (2)(0) + (0)(0) + (1)(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$a_{:2} = \begin{bmatrix} l_{11}u_{12} + l_{12}u_{22} + l_{13}u_{32} + l_{14}u_{42} \\ l_{21}u_{12} + l_{22}u_{22} + l_{23}u_{32} + l_{24}u_{42} \\ l_{31}u_{12} + l_{32}u_{22} + l_{33}u_{32} + l_{34}u_{42} \\ l_{41}u_{12} + l_{42}u_{22} + l_{43}u_{32} + l_{44}u_{42} \end{bmatrix} = \begin{bmatrix} (1)(1) + (0)(-3) + (0)(0) + (0)(0) \\ (1)(1) + (1)(-3) + (0)(0) + (0)(0) \\ (1)(1) + (-1)(-3) + (1)(0) + (0)(0) \\ (1)(1) + (2)(-3) + (0)(0) + (1)(0) \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 4 \\ -5 \end{bmatrix}$$

$$a_{:3} = \begin{bmatrix} l_{11}u_{13} + l_{12}u_{23} + l_{13}u_{33} + l_{14}u_{43} \\ l_{21}u_{13} + l_{22}u_{23} + l_{23}u_{33} + l_{24}u_{43} \\ l_{31}u_{13} + l_{32}u_{23} + l_{33}u_{33} + l_{34}u_{43} \\ l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} + l_{44}u_{43} \end{bmatrix} = \begin{bmatrix} (1)(1) + (0)(-3) + (0)(-8) + (0)(0) \\ (1)(1) + (1)(-3) + (0)(-8) + (0)(0) \\ (1)(1) + (-1)(-3) + (1)(-8) + (0)(0) \\ (1)(1) + (2)(-3) + (0)(-8) + (1)(0) \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ -4 \\ -5 \end{bmatrix}$$

$$a_{:4} = \begin{bmatrix} l_{11}u_{14} + l_{12}u_{24} + l_{13}u_{34} + l_{14}u_{44} \\ l_{21}u_{14} + l_{22}u_{24} + l_{23}u_{34} + l_{24}u_{44} \\ l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} + l_{34}u_{44} \\ l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + l_{44}u_{44} \end{bmatrix} = \begin{bmatrix} (1)(1) + (0)(-3) + (0)(-3) + (0)(2) \\ (1)(1) + (1)(-3) + (0)(-3) + (0)(2) \\ (1)(1) + (-1)(-3) + (1)(-3) + (0)(2) \\ (1)(1) + (2)(-3) + (0)(-3) + (1)(2) \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 1 \\ -3 \end{bmatrix}$$

From the above it is easy to see that

$$LU = [a_{:1}, a_{:2}, a_{:3}, a_{:4}] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & -2 \\ 1 & 4 & -4 & 1 \\ 1 & -5 & -5 & -3 \end{bmatrix} = A$$

Therefore we have proven that $A = LU$.

Extra information - Derivation of Doolittle Method

The LU decomposition of a square matrix A is defined as a set of lower triangular (i.e. no elements above the diagonal) and upper triangular (i.e. no elements below the diagonal) matrices that, when multiplied together, produce A. In the simplest case, it is easy to show that if A contains no zero quantities along it's diagonal during a simple forward elimination process (i.e. partial pivoting is not necessary) then matrices L and U exist for A and are directly calculable from A. In the more general case, it is also easy to show that an LU decomposition exists for any square matrix A if partial pivoting is allowed. In this case the LU decomposition will have an associated permutation matrix P, and really you will have solved the sysem $PA = LU$, but the difference in how these decompositions are used in further calculations (for example to calculate the inverse of a matrix) is minimal (All one must do to use the permuted LU set to solve a linear system is multiply the system by P –> $Ax = b$ –> $PAx = Pb$ –> $LUx = Pb$. The solution of the system will be exactly the same). Finally, if one specifies that either the L **or** U matrix must have all ones along its diagonal (in the Doolittle method, and in this problem..., we specify that L must have a unit diagonal), then the L and U set is unique to A.

So how do we derive the LU decomposition of a matrix A. The answer lies in applying elementary row operations. There are three classic elementary row operations in linear algebra:

- Row Switching:

  Performed by left multiplying a matrix A by an identity matrix that has had its rows i and j switched, where i and j are the rows desired to be switched in A. For example, if A were a 4x4 matrix and you wanted to switch rows 2 and 3, you would left multiply A by the following "row switching" matrix E_1 (i.e. E_1A):

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Multiplication of a row by a constant:

  This one is pretty straightforward. When you multiply a row by some constant c then you multiply every element of that row by that constant on an element by element basis. This may be represented in elementary matrix form by left multiplying A by an elementary matrix E_2 that contains only 1s along its diagonal (i.e. it is an identity matrix) except for the ith element, which contains a value of c (where i is the row we wish to multiply by c). So, for example, if we had an 4x4 matrix A and we wished to multiply row 4 by 5 then we would left multiply A by the following elementary matrix E_2:

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

- Addition of a multiple of one row to another:

  Many will recognize this as the essential operation in solving a linear system of equations. The addition of one row i, multiplied by some constant c, to another row j may be represented in elementary matrix form by an identity matrix with an off diagonal element equal to c in the ith column and the jth row. So, if for example we wanted to add 4 times row 3 to row 1, we would left multiply A by the following elementary matrix E_3 (go ahead, check for yourself, everything works out):

$$E_3 = \begin{bmatrix} 1 & 0 & 4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The final pieces of information we require before we may derive the LU decomposition of A are given below (derivation not provided):

- The inverse of E_1 is equal to E_1^T
- The inverse of E_2 is equal to a matrix $E_{2'}$ whose diagonal elements are equal to the inverse of those in E_2 (i.e. 1 over the elements in E_2)
- The inverse of E_3 is equal to E_3 with all its off-diagonal elements multiplied by (-1)
- A lower triangular matrix multplied by a lower triangular matrix is also lower triangular (the opposite is also true)
- The inverse of a lower triangular matrix is also lower triangular (the opposite is also true)

Now that we have the required background information we may derive the LU decomposition of A. Note that this derivation deals with the simplified case that no intermediate partial pivoting is required during the derivation.

So let us start by asking a simple question. If one wanted to derive an upper triangular matrix from A, how would one do it? The obvious answer would be to perform forward elimination on the matrix until all elements below the diagonal no longer existed. But how would this look in terms elementary row operations? Essentially we would be continuously multiplying by elementary matrices of form E_3. Furthermore, since we are trying to convert matrix A to an upper triangular matrix we would only be concerned with eliminating elements below the diagonal wouldn't we? Therefore all of the elementary matrices applied would be some multiple of row i being added to some row j where $i > j$. Recall that the off-diagonal element of E_3 occurs at location e_{ij}. Therefore the forward elimination step would **always** consist of lower triangular matrices no? So in the case of the first column of A we would simply left multiply A by the following three elementary row operation matrices to eliminate the below diagonal elements wouldn't we? (go ahead, try it out by hand)

$$A^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{a_{41}^{(0)}}{a_{11}^{(0)}} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{a_{31}^{(0)}}{a_{11}^{(0)}} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{a_{21}^{(0)}}{a_{11}^{(0)}} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & -2 \\ 1 & 4 & -4 & 1 \\ 1 & -5 & -5 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 3 & -5 & 0 \\ 0 & -6 & -6 & -4 \end{bmatrix}$$

But we can't exactly claim $A = A^{(1)}$ can we. So what would we need to do to make the equivalence correct? We would need to multiply by the inverse of the elementary row operations.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{21}^{(0)}}{a_{11}^{(0)}} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{a_{31}^{(0)}}{a_{11}^{(0)}} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{a_{41}^{(0)}}{a_{11}^{(0)}} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{a_{41}^{(0)}}{a_{11}^{(0)}} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{a_{31}^{(0)}}{a_{11}^{(0)}} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{a_{21}^{(0)}}{a_{11}^{(0)}} & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

At this point it should be stated that when multiplying elementary matrices of type E_3 where no off-diagonal elements occur in the same position (always the case when performing forward elimination) the result of the multiplication is simply a matrix containing **all** of the off-diagonal elements in the exact same positions they occurred separately (a prove will not be given but the reader is encouraged to investigate the validity of this statement on their own). Therefore if we conduct the multiplication of the row operations and inverse operations respectively then it is easy to see the system simplifies to:

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{21}^{(0)}}{a_{11}^{(0)}} & 1 & 0 & 0 \\ \frac{a_{31}^{(0)}}{a_{11}^{(0)}} & 0 & 1 & 0 \\ \frac{a_{41}^{(0)}}{a_{11}^{(0)}} & 0 & 0 & 1 \end{bmatrix}}_{(L^{(1)})^{-1}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{a_{21}^{(0)}}{a_{11}^{(0)}} & 1 & 0 & 0 \\ -\frac{a_{31}^{(0)}}{a_{11}^{(0)}} & 0 & 1 & 0 \\ -\frac{a_{41}^{(0)}}{a_{11}^{(0)}} & 0 & 0 & 1 \end{bmatrix}}_{L^{(1)}} \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & -2 & -2 \\ 1 & 4 & -4 & 1 \\ 1 & -5 & -5 & -3 \end{bmatrix}}_{A} = (L^{(1)})^{-1} A^{(1)}$$

Now what if we wanted to eliminate the second column of A (also $A^{(1)}$)? Well naturally we would follow the exact same process (i.e. we would multiply $A^{(1)}$ by the elementary row operation matrices required to eliminate all elements below the second diagonal, remembering to also multiply by the inverse of the operations to maintain the equivalence to A).

$$\begin{aligned} A &= (L^{(1)})^{-1}(L^{(2)})^{-1}L^{(2)}A^{(1)} \\ A &= (L^{(1)})^{-1}(L^{(2)})^{-1}A^{(2)} \end{aligned}$$

Are you starting to see the pattern now? If we were to repeat this process $n-1$ times (where n is the dimension of A) then the forward elimination would be complete and therefore $A^{(n-1)}$ would **have to be upper triangular** (as a quick aside, we repeat the process $n-1$ times and not n times because we do not have to eliminate the first row).

$$\begin{aligned} A &= (L^{(1)})^{-1}(L^{(2)})^{-1}\cdots(L^{(n-2)})^{-1}(L^{(n-1)})^{-1}L^{(n-1)}L^{(n-2)}\ldots L^{(2)}L^{(1)}A \\ A &= (L^{(1)})^{-1}(L^{(2)})^{-1}\cdots(L^{(n-2)})^{-1}(L^{(n-1)})^{-1}A^{(n-1)} \\ A &= (L^{(1)})^{-1}(L^{(2)})^{-1}\cdots(L^{(n-2)})^{-1}(L^{(n-1)})^{-1}U \end{aligned}$$

Now recall the the inverse of a lower triangular matrix is also a lower triangular matrix and the multiplication of two lower triangular matrices is also lower triangular. Therefore all of the terms to the left of U collect to form one lower triangular matrix L.

$$A = LU$$

But this is exactly what we wanted when we started the LU decomposition process. Furthermore, knowing the simplified multiplication rule for E_3 type matrices presented earlier, it is easy to see that L equals:

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{a_{21}^{(0)}}{a_{11}^{(0)}} & 1 & 0 & \cdots & 0 & 0 & 0 \\ \frac{a_{31}^{(0)}}{a_{11}^{(0)}} & \frac{a_{32}^{(1)}}{a_{22}^{(1)}} & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{a_{n-2,1}^{(0)}}{a_{11}^{(0)}} & \frac{a_{n-2,2}^{(1)}}{a_{22}^{(1)}} & \frac{a_{n-2,3}^{(2)}}{a_{33}^{(2)}} & \cdots & 1 & 0 & 0 \\ \frac{a_{n-1,1}^{(0)}}{a_{11}^{(0)}} & \frac{a_{n-1,2}^{(1)}}{a_{22}^{(1)}} & \frac{a_{n-1,3}^{(2)}}{a_{33}^{(2)}} & \cdots & \frac{a_{n-1,n-2}^{(n-2)}}{a_{n-2,n-2}^{(n-2)}} & 1 & 0 \\ \frac{a_{n,1}^{(0)}}{a_{11}^{(0)}} & \frac{a_{n,2}^{(1)}}{a_{22}^{(1)}} & \frac{a_{n,3}^{(2)}}{a_{33}^{(2)}} & \cdots & \frac{a_{n,n-2}^{(n-2)}}{a_{n-2,n-2}^{(n-2)}} & \frac{a_{n,n-1}^{(n-1)}}{a_{n-1,n-1}^{(n-1)}} & 1 \end{bmatrix}$$

This is exactly the general form of L presented in the notes. This is where that form comes from.

2. To perform LU decomposition in MATLAB we use the `lu` function. The two most common methods of using `lu` are:

   - `[L,U] = lu(A)` : this will return the L and U matrices with the L matrix multiplied by a permutation matrix P (if partial pivoting occurred during the solution procedure).
   - `'[L,U,P] = lu(A)` : this will return the L and U matrices with the implicit understanding that the system solved was PA.

   To perform LU decomposition in Python we use the `lu_factor` function.

   MATLAB code

```
A = [1, 1, 1, 1;
     1,-2,-2,-2;
     1, 4,-4, 1;
     1,-5,-5,-3];

[L,U] = lu(A)
[L,U,P] = lu(A)
```

```
% Running the above script results in the following output

L =
    1.0000         0         0         0
    1.0000    0.5000         0    1.0000
    1.0000   -0.5000    1.0000         0
    1.0000    1.0000         0         0

U =
     1     1     1     1
     0    -6    -6    -4
     0     0    -8    -2
     0     0     0    -1


L =
    1.0000         0         0         0
    1.0000    1.0000         0         0
    1.0000   -0.5000    1.0000         0
    1.0000    0.5000         0    1.0000


U =
     1     1     1     1
     0    -6    -6    -4
     0     0    -8    -2
     0     0     0    -1


P =
     1     0     0     0
     0     0     0     1
     0     0     1     0
     0     1     0     0
```

Python code

```
import numpy as np
from numpy.linalg import *
from scipy.linalg import *

A = np.array([[1,1,1,1],
              [1,-2,-2,-2],
              [1,4,-4,1],
              [1,-5,-5,-3]])

LU, P = lu_factor(A)

print('LU\n %s\n P\n %s' % (LU, P))
"""
Running the above code results in the following output

LU
[[ 1.   1.   1.   1. ]
 [ 1.  -6.  -6.  -4. ]
 [ 1.  -0.5 -8.  -2. ]
 [ 1.   0.5  0.  -1. ]]
P
[0 3 2 3]
```

```
Remember that python outputs L and U in a compressed format

The separated L and U matrices would be

L
[[ 1.   0.   0.   0. ]
 [ 1.   1.   0.   0. ]
 [ 1.  -0.5  1.   0. ]
 [ 1.   0.5  0.   1. ]]

U
[[ 1.   1.   1.   1. ]
 [ 0.  -6.  -6.  -4. ]
 [ 0.   0.  -8.  -2. ]
 [ 0.   0.   0.  -1. ]]
"""
```

We note that the results are different from those we calculated above. This is because MATLAB and Python perform LU decomposition with partial pivoting. Thus the system solved was actually $PA = LU$.

3. To calculate the inverse of a matrix using its LU decomposition we realize that the system $AX = B$, where X and B are matrices, may be solved by splitting X and B into their individual column pairs (i.e. column 1 of X goes with column 1 of B, column 2 with column 2, etc.) and solving each system of linear equations separately. In this case we know a matrix multiplied by its inverse yields the identity matrix. Therefore:

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, we recall that the LU decomposition of a matrix may be used to simplify the solution process by splitting the system into two separate systems that are easily solved.

- $Ly = b$
- $Ux = y$

Invert column 1

We start with the subsystem $Ly = b$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We eliminate the elements in the first column under the diagonal (pivot) element by adding (-1) times row 1 to row 2, (-1) times row 1 to row 3, and (-1) times row 1 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1+(-1)(1) & 1+(-1)(0) & 0+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & -1+(-1)(0) & 1+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & 2+(-1)(0) & 0+(-1)(0) & 1+(-1)(0) \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} = \begin{bmatrix} 1 \\ 0+(-1)(1) \\ 0+(-1)(1) \\ 0+(-1)(1) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix}$$

We eliminate the elements in the second column under the diagonal (pivot) element by adding (1) times row 2 to row 3, and (-2) times row 2 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1+(1)(1) & 1+(1)(0) & 0+(1)(0) \\ 0 & 2+(-2)(1) & 0+(-2)(0) & 1+(-2)(0) \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1+(1)(-1) \\ -1+(-2)(-1) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} = \begin{bmatrix}$$

Now that we have solved for y we move on to solve the second subsystem $Ux = y$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 0 & -8 & -3 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -2 \\ 1 \end{bmatrix}$$

We start by dividing row 4 by its diagonal element and then adding 3 times row 4 to row 3, 3 times row 4 to row 2, and (-1) times row 4 to row 1.

$$\begin{bmatrix} 1 & 1 & 1 & 1+(-1)\left(\frac{2}{(2)}\right) \\ 0 & -3 & -3 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & -8 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & 0 & \frac{2}{(2)} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 1+(-1)\left(\frac{1}{(2)}\right) \\ -1+(3)\left(\frac{1}{(2)}\right) \\ -2+(3)\left(\frac{1}{(2)}\right) \\ \frac{1}{(2)} \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & -3 & -3 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix}$$

Next we divide row 3 by its diagonal element and then add 3 times row 3 to row 2, and (-1) times row 3 to row 1.

$$\begin{bmatrix} 1 & 1 & 1+(-1)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & -3 & -3+(3)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & 0 & \frac{-8}{(-8)} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 1/2+(-1)\left(\frac{-1/2}{(-8)}\right) \\ 1/2+(3)\left(\frac{-1/2}{(-8)}\right) \\ \frac{-1/2}{(-8)} \\ 1/2 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 7/16 \\ 11/16 \\ 1/16 \\ 1/2 \end{bmatrix}$$

Finally we divide row 2 by its diagonal element and then add (-1) times row 2 to row 1.

$$\begin{bmatrix} 1 & 1+(-1)\left(\frac{-3}{(-3)}\right) & 0 & 0 \\ 0 & \frac{-3}{(-3)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 7/16+(-1)\frac{11/16}{(-3)} \\ \frac{11/16}{(-3)} \\ 1/16 \\ 1/2 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 32/48 \\ -11/48 \\ 3/48 \\ 24/48 \end{bmatrix}$$

Therefore the first column of the inverse of A is.

$$\begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 32/48 \\ -11/48 \\ 3/48 \\ 24/48 \end{bmatrix}$$

Invert column 2

We start with the subsystem $Ly = b$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{12} \\ y_{22} \\ y_{32} \\ y_{42} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

We eliminate the elements in the first column under the diagonal (pivot) element by adding (-1) times row 1 to row 2, (-1) times row 1 to row 3, and (-1) times row 1 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1+(-1)(1) & 1+(-1)(0) & 0+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & -1+(-1)(0) & 1+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & 2+(-1)(0) & 0+(-1)(0) & 1+(-1)(0) \end{bmatrix} \begin{bmatrix} y_{12} \\ y_{22} \\ y_{32} \\ y_{42} \end{bmatrix} = \begin{bmatrix} 0 \\ 1+(-1)(0) \\ 0+(-1)(0) \\ 0+(-1)(0) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

We eliminate the elements in the second column under the diagonal (pivot) element by adding (1) times row 2 to row 3, and (-2) times row 2 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1+(1)(1) & 1+(1)(0) & 0+(1)(0) \\ 0 & 2+(-2)(1) & 0+(-2)(0) & 1+(-2)(0) \end{bmatrix} \begin{bmatrix} y_{12} \\ y_{22} \\ y_{32} \\ y_{42} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0+(1)(1) \\ 0+(-2)(1) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{12} \\ y_{22} \\ y_{32} \\ y_{42} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ -2 \end{bmatrix}$$

Now that we have solved for y we move on to solve the second subsystem $Ux = y$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 0 & -8 & -3 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ -2 \end{bmatrix}$$

We start by dividing row 4 by its diagonal element and then adding 3 times row 4 to row 3, 3 times row 4 to row 2, and (-1) times row 4 to row 1.

$$\begin{bmatrix} 1 & 1 & 1 & 1+(-1)\left(\frac{2}{(2)}\right) \\ 0 & -3 & -3 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & -8 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & 0 & \frac{2}{(2)} \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 0+(-1)\left(\frac{-2}{(2)}\right) \\ 1+(3)\left(\frac{-2}{(2)}\right) \\ 1+(3)\left(\frac{-2}{(2)}\right) \\ \frac{-2}{(2)} \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & -3 & -3 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ -2 \\ -1 \end{bmatrix}$$

Next we divide row 3 by its diagonal element and then add 3 times row 3 to row 2, and (-1) times row 3 to row 1.

$$\begin{bmatrix} 1 & 1 & 1+(-1)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & -3 & -3+(3)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & 0 & \frac{-8}{(-8)} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 1+(-1)\left(\frac{-2}{(-8)}\right) \\ -2+(3)\left(\frac{-2}{(-8)}\right) \\ \frac{-2}{(-8)} \\ -1 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 3/4 \\ -5/4 \\ 1/4 \\ -1 \end{bmatrix}$$

Finally we divide row 2 by its diagonal element and then add (-1) times row 2 to row 1.

$$\begin{bmatrix} 1 & 1+(-1)\left(\frac{-3}{(-3)}\right) & 0 & 0 \\ 0 & \frac{-3}{(-3)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 3/4+(-1)\frac{-5/4}{(-3)} \\ \frac{-5/4}{(-3)} \\ 1/4 \\ -1 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 5/12 \\ 1/4 \\ -1 \end{bmatrix}$$

Therefore the first column of the inverse of A is.

$$\begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 16/48 \\ 20/48 \\ 12/48 \\ -48 \end{bmatrix}$$

Invert column 3

We start with the subsystem $Ly = b$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{13} \\ y_{23} \\ y_{33} \\ y_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

We eliminate the elements in the first column under the diagonal (pivot) element by adding (-1) times row 1 to row 2, (-1) times row 1 to row 3, and (-1) times row 1 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1+(-1)(1) & 1+(-1)(0) & 0+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & -1+(-1)(0) & 1+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & 2+(-1)(0) & 0+(-1)(0) & 1+(-1)(0) \end{bmatrix} \begin{bmatrix} y_{13} \\ y_{23} \\ y_{33} \\ y_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 0+(-1)(0) \\ 1+(-1)(0) \\ 0+(-1)(0) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} \end{bmatrix}$$

We eliminate the elements in the second column under the diagonal (pivot) element by adding (1) times row 2 to row 3, and (-2) times row 2 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1+(1)(1) & 1+(1)(0) & 0+(1)(0) \\ 0 & 2+(-2)(1) & 0+(-2)(0) & 1+(-2)(0) \end{bmatrix} \begin{bmatrix} y_{13} \\ y_{23} \\ y_{33} \\ y_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1+(1)(0) \\ 0+(-2)(0) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{13} \\ y_{23} \\ y_{33} \\ y_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Now that we have solved for y we move on to solve the second subsystem $Ux = y$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 0 & -8 & -3 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

We start by dividing row 4 by its diagonal element and then adding 3 times row 4 to row 3, 3 times row 4 to row 2, and (-1) times row 4 to row 1.

$$\begin{bmatrix} 1 & 1 & 1 & 1+(-1)\left(\frac{2}{(2)}\right) \\ 0 & -3 & -3 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & -8 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & 0 & \frac{2}{(2)} \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 0+(-1)\left(\frac{0}{(2)}\right) \\ 0+(3)\left(\frac{0}{(2)}\right) \\ 1+(3)\left(\frac{0}{(2)}\right) \\ \frac{0}{(2)} \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & -3 & -3 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Next we divide row 3 by its diagonal element and then add 3 times row 3 to row 2, and (-1) times row 3 to row 1.

$$\begin{bmatrix} 1 & 1 & 1+(-1)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & -3 & -3+(3)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & 0 & \frac{-8}{(-8)} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 0+(-1)\left(\frac{1}{(-8)}\right) \\ 0+(3)\left(\frac{1}{(-8)}\right) \\ \frac{1}{(-8)} \\ 0 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 1/8 \\ -3/8 \\ -1/8 \\ 0 \end{bmatrix}$$

Finally we divide row 2 by its diagonal element and then add (-1) times row 2 to row 1.

$$\begin{bmatrix} 1 & 1+(-1)\left(\frac{-3}{(-3)}\right) & 0 & 0 \\ 0 & \frac{-3}{(-3)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 1/8+(-1)\frac{-3/8}{(-3)} \\ \frac{-3/8}{(-3)} \\ -1/8 \\ 0 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 1/8 \\ -1/8 \\ 0 \end{bmatrix}$$

Therefore the first column of the inverse of A is.

$$\begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 6/48 \\ -6/48 \\ 0 \end{bmatrix}$$

Invert column 4

We start with the subsystem $Ly = b$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{14} \\ y_{24} \\ y_{34} \\ y_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

We eliminate the elements in the first column under the diagonal (pivot) element by adding (-1) times row 1 to row 2, (-1) times row 1 to row 3, and (-1) times row 1 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1+(-1)(1) & 1+(-1)(0) & 0+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & -1+(-1)(0) & 1+(-1)(0) & 0+(-1)(0) \\ 1+(-1)(1) & 2+(-1)(0) & 0+(-1)(0) & 1+(-1)(0) \end{bmatrix} \begin{bmatrix} y_{14} \\ y_{24} \\ y_{34} \\ y_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0+(-1)(0) \\ 0+(-1)(0) \\ 1+(-1)(0) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

We eliminate the elements in the second column under the diagonal (pivot) element by adding (1) times row 2 to row 3, and (-2) times row 2 to row 4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1+(1)(1) & 1+(1)(0) & 0+(1)(0) \\ 0 & 2+(-2)(1) & 0+(-2)(0) & 1+(-2)(0) \end{bmatrix} \begin{bmatrix} y_{14} \\ y_{24} \\ y_{34} \\ y_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0+(1)(0) \\ 1+(-2)(0) \end{bmatrix} \implies \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{14} \\ y_{24} \\ y_{34} \\ y_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Now that we have solved for y we move on to solve the second subsystem $Ux = y$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -3 & -3 & -3 \\ 0 & 0 & -8 & -3 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

We start by dividing row 4 by its diagonal element and then adding 3 times row 4 to row 3, 3 times row 4 to row 2, and (-1) times row 4 to row 1.

$$\begin{bmatrix} 1 & 1 & 1 & 1+(-1)\left(\frac{2}{(2)}\right) \\ 0 & -3 & -3 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & -8 & -3+(3)\left(\frac{2}{(2)}\right) \\ 0 & 0 & 0 & \frac{2}{(2)} \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} 0+(-1)\left(\frac{1}{(2)}\right) \\ 0+(3)\left(\frac{1}{(2)}\right) \\ 0+(3)\left(\frac{1}{(2)}\right) \\ \frac{1}{(2)} \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & -3 & -3 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} -1/2 \\ 3/2 \\ 3/2 \\ 1/2 \end{bmatrix}$$

Next we divide row 3 by its diagonal element and then add 3 times row 3 to row 2, and (-1) times row 3 to row 1.

$$\begin{bmatrix} 1 & 1 & 1+(-1)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & -3 & -3+(3)\left(\frac{-8}{(-8)}\right) & 0 \\ 0 & 0 & \frac{-8}{(-8)} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} -1/2+(-1)\left(\frac{3/2}{(-8)}\right) \\ 3/2+(3)\left(\frac{3/2}{(-8)}\right) \\ \frac{3/2}{(-8)} \\ 1/2 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} -5/16 \\ 15/16 \\ -3/16 \\ 1/2 \end{bmatrix}$$

Finally we divide row 2 by its diagonal element and then add (-1) times row 2 to row 1.

$$\begin{bmatrix} 1 & 1+(-1)\left(\frac{-3}{(-3)}\right) & 0 & 0 \\ 0 & \frac{-3}{(-3)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} -5/16+(-1)\frac{15/16}{(-3)} \\ \frac{15/16}{(-3)} \\ -3/16 \\ 1/2 \end{bmatrix} \implies \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ -15/48 \\ -3/16 \\ 1/2 \end{bmatrix}$$

Therefore the first column of the inverse of A is.

$$\begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ -15/48 \\ -9/48 \\ 24/48 \end{bmatrix}$$

Therefore the inverse of A is

$$A^{-1} = [x_{:1}, x_{:2}, x_{:3}, x_{:4}] = \begin{bmatrix} 32/48 & 16/48 & 0 & 0 \\ -11/48 & 20/48 & 6/48 & -15/48 \\ 3/48 & 12/48 & -6/48 & -9/48 \\ 24/48 & -48/48 & 0 & 24/48 \end{bmatrix} = \begin{bmatrix} 0.6667 & 0.3333 & 0 & 0 \\ -0.2292 & 0.4167 & 0.1250 & -0.3125 \\ 0.0625 & 0.2500 & -0.1250 & -0.1875 \\ 0.5000 & -1.0000 & 0 & 0.5000 \end{bmatrix}$$

Check the inverse

To check the inverse of A in MATLAB and Python we use the `inv()` function.

MATLAB code

```
A = [1, 1, 1, 1;
     1,-2,-2,-2;
     1, 4,-4, 1;
     1,-5,-5,-3];

A_inv = inv(A)
```

```
% Running the above script results in the following output
A_inv =
    0.6667    0.3333         0   -0.0000
   -0.2292    0.4167    0.1250   -0.3125
    0.0625    0.2500   -0.1250   -0.1875
    0.5000   -1.0000         0    0.5000
```

Python code

```
import numpy as np
from numpy.linalg import *
from scipy.linalg import *

A = np.array([[1,1,1,1],
              [1,-2,-2,-2],
              [1,4,-4,1],
              [1,-5,-5,-3]])
print ('A inverted = \n%s' % inv(A))

"""
Running the above code results in the following output

A inverted =
[[  6.66666667e-01   3.33333333e-01   0.00000000e+00  -2.77555756e-17]
 [ -2.29166667e-01   4.16666667e-01   1.25000000e-01  -3.12500000e-01]
 [  6.25000000e-02   2.50000000e-01  -1.25000000e-01  -1.87500000e-01]
 [  5.00000000e-01  -1.00000000e+00   0.00000000e+00   5.00000000e-01]]
"""
```

Finally we show that $AA^{-1} = I$. Since matrix multiplication takes up alot of space we will do this column by column.

$$id_{:1} = \begin{bmatrix} a_{11}a_{11}^{-1} + a_{12}a_{21}^{-1} + a_{13}a_{31}^{-1} + a_{14}a_{41}^{-1} \\ a_{21}a_{11}^{-1} + a_{22}a_{21}^{-1} + a_{23}a_{31}^{-1} + a_{24}a_{41}^{-1} \\ a_{31}a_{11}^{-1} + a_{32}a_{21}^{-1} + a_{33}a_{31}^{-1} + a_{34}a_{41}^{-1} \\ a_{41}a_{11}^{-1} + a_{42}a_{21}^{-1} + a_{43}a_{31}^{-1} + a_{44}a_{41}^{-1} \end{bmatrix} = \begin{bmatrix} (1)(32/48) + (1)(-11/48) + (1)(3/48) + (1)(24/48) \\ (1)(32/48) + (-2)(-11/48) + (-2)(3/48) + (-2)(24/48) \\ (1)(32/48) + (4)(-11/48) + (-4)(3/48) + (1)(24/48) \\ (1)(32/48) + (-5)(-11/48) + (-5)(3/48) + (1)(24/48) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$id_{:2} = \begin{bmatrix} a_{11}a_{12}^{-1} + a_{12}a_{22}^{-1} + a_{13}a_{32}^{-1} + a_{14}a_{42}^{-1} \\ a_{21}a_{12}^{-1} + a_{22}a_{22}^{-1} + a_{23}a_{32}^{-1} + a_{24}a_{42}^{-1} \\ a_{31}a_{12}^{-1} + a_{32}a_{22}^{-1} + a_{33}a_{32}^{-1} + a_{34}a_{42}^{-1} \\ a_{41}a_{12}^{-1} + a_{42}a_{22}^{-1} + a_{43}a_{32}^{-1} + a_{44}a_{42}^{-1} \end{bmatrix} = \begin{bmatrix} (1)(16/48) + (1)(20/48) + (1)(12/48) + (1)(-48/48) \\ (1)(16/48) + (-2)(20/48) + (-2)(12/48) + (-2)(-48/48) \\ (1)(16/48) + (4)(20/48) + (-4)(12/48) + (1)(-48/48) \\ (1)(16/48) + (-5)(20/48) + (-5)(12/48) + (1)(-48/48) \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$$id_{:3} = \begin{bmatrix} a_{11}a_{13}^{-1} + a_{12}a_{23}^{-1} + a_{13}a_{33}^{-1} + a_{14}a_{43}^{-1} \\ a_{21}a_{13}^{-1} + a_{22}a_{23}^{-1} + a_{23}a_{33}^{-1} + a_{24}a_{43}^{-1} \\ a_{31}a_{13}^{-1} + a_{32}a_{23}^{-1} + a_{33}a_{33}^{-1} + a_{34}a_{43}^{-1} \\ a_{41}a_{13}^{-1} + a_{42}a_{23}^{-1} + a_{43}a_{33}^{-1} + a_{44}a_{43}^{-1} \end{bmatrix} = \begin{bmatrix} (1)(0) + (1)(6/48) + (1)(-6/48) + (1)(0) \\ (1)(0) + (-2)(6/48) + (-2)(-6/48) + (-2)(0) \\ (1)(0) + (4)(6/48) + (-4)(-6/48) + (1)(0) \\ (1)(0) + (-5)(6/48) + (-5)(-6/48) + (1)(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$id_{:4} = \begin{bmatrix} a_{11}a_{14}^{-1} + a_{12}a_{24}^{-1} + a_{13}a_{34}^{-1} + a_{14}a_{44}^{-1} \\ a_{21}a_{14}^{-1} + a_{22}a_{24}^{-1} + a_{23}a_{34}^{-1} + a_{24}a_{44}^{-1} \\ a_{31}a_{14}^{-1} + a_{32}a_{24}^{-1} + a_{33}a_{34}^{-1} + a_{34}a_{44}^{-1} \\ a_{41}a_{14}^{-1} + a_{42}a_{24}^{-1} + a_{43}a_{34}^{-1} + a_{44}a_{44}^{-1} \end{bmatrix} = \begin{bmatrix} (1)(0) + (1)(-15/48) + (1)(-9/48) + (1)(24/48) \\ (1)(0) + (-2)(-15/48) + (-2)(-9/48) + (-2)(24/48) \\ (1)(0) + (4)(-15/48) + (-4)(-9/48) + (1)(24/48) \\ (1)(0) + (-5)(-15/48) + (-5)(-9/48) + (1)(24/48) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

From the above it is easy to see that

$$AA^{-1} = [id_{:1}, id_{:2}, id_{:3}, id_{:4}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I$$

Therefore we have proven that $AA^{-1} = I$.

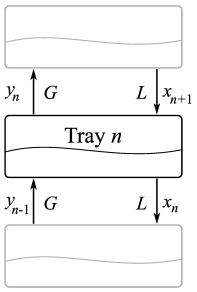
Question 3 [4]

Note: This is an example of the type of question you will get in the take-home midterm.

An impure gas stream can be cleaned, by absorbing the impurity into the liquid phase of an appropriate liquid solvent. For example, carbon dioxide can be removed from flue gas by scrubbing it with monoethanolamine; or NO_2 can be absorbed into water to produce nitric acid.

In this question we will derive the model equations for a sequence of absorber stages (trays), in which a component in the gas phase is absorbed into the liquid phase.

First consider the n^{th} stage in an absorber:



where the molar gas flow is G and the liquid molar flow is L, and assume these are constant throughout the column. Let the number of moles held-up in the liquid phase of the tray be H_n, the hold-up. And let's only consider a binary system, where the species we are interested in is called A, and the absorbing liquid is species B.

The mol **fraction** of species A, in the gas phase, leaving the tray is $y_{A,n}$, and we also have $y_{B,n}$, the mol fraction of B leaving: this gives of course $y_{A,n} + y_{B,n} = 1$. But since we are interested only in species A in this question, and because we can always calculate B by subtraction from 1.0, we will drop the A and B subscripts, and only write – for example – y_n to denote the gas mol fraction of A leaving the n^{th} stage of the absorber.

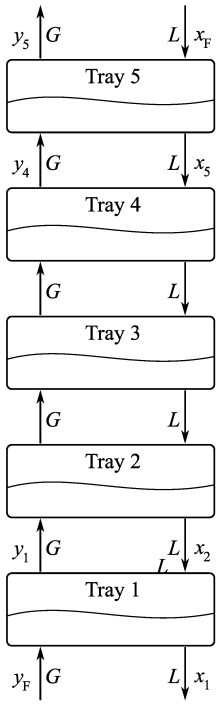
Similarly, the mol fraction of A, in the liquid phase, leaving the stage is x_n. And we will denote the liquid mol fraction of A entering as x_{n+1}, while the gas-phase mol fraction of A entering is defined as y_{n-1}.

Using Raoult's Law, and assuming fairly ideal conditions, and that the liquid and vapour phase are well-mixed and in equilibrium, we may relate the concentrations in the gas and liquid phase as:

$$y_n P_{\text{total}} = x_n P_{\text{A}} \gamma_A$$
$$y_n = \beta x_n$$

where P_{total} is the total pressure in the absorber, and P_{A} is the pure-component vapour pressure of A, at the given conditions, and γ_A is an activity coefficient that corrects for non-idealities (cross-reference your thermodynamics course for more background on this equation). In this question you may assume that β is constant throughout the absorber.

1. Write down a **dynamic** mass balance for species A, in terms of the nomenclature above, for this n^{th} stage in the absorber tower.
2. Now write down the **steady-state** mass-balance equation for the above, but only in terms of liquid-phase mol fractions. Also divide the steady-state tray balance equations through by βG and simplify them using the term $\delta = \dfrac{L}{\beta G}$
3. Now repeat this previous step, and write the steady-state mass-balance equations for the five-stage system shown here.



4. Rewrite these mass-balance equations in matrix form, $Ax = b$, where:

$$x = \begin{bmatrix} y_F \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_F \\ y_5 \end{bmatrix}$$

5. Write a function that will generate the A matrix and b vector, given the variables in the system. Use this function and solve the system of equations for the two cases below:

Case	G	L	β	x_F	y_F
Case I	100	150	0.6	0.05	0.35
Case II	100	50	0.6	0.05	0.35

In each case, we are obviously interested in the liquid-phase concentration profile along the column, x_F down to x_1. Report these values and explain whether your answers make physical sense, or not, for the two cases.

Solution

1. The dynamic mass balance for the n^{th} tray:

$$\frac{H dx_n}{dt} = Lx_{n+1} + Gy_{n-1} - Lx_n - Gy_n$$
$$H\frac{dx_n}{dt} = L(x_{n+1} - x_n) + G(y_{n-1} - y_n)$$

2. At steady state we have $\dfrac{dx_n}{dt} = 0$. Further, using that $y_n = \beta x_n$, and the definition of $\delta = \dfrac{L}{G\beta}$, we can simplify

the previous equation to:

$$0 = L(x_{n+1} - x_n) + G\beta(x_{n-1} - x_n)$$
$$0 = x_{n-1} - \delta x_n - x_n + \delta x_{n+1}$$
$$0 = x_{n-1} - (\delta + 1)x_n + \delta x_{n+1}$$

3. For the system with 5 trays, we have:
   (a) $0 = \frac{y_F}{\beta} - (\delta + 1)x_1 + \delta x_2$
   (b) $0 = x_1 - (\delta + 1)x_2 + \delta x_3$
   (c) $0 = x_2 - (\delta + 1)x_3 + \delta x_4$
   (d) $0 = x_3 - (\delta + 1)x_4 + \delta x_5$
   (e) $0 = x_4 - (\delta + 1)x_5 + \delta x_F$
4. We require a square matrix for A, so we add 3 other equations to the above 5 tray balances. These three equations correspond to the known fractions of A in the inlet gas and liquid streams, as well as the equation that relates y_5 to x_5.

$$Ax = b$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\beta} & -(\delta+1) & \delta & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -(\delta+1) & \delta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -(\delta+1) & \delta & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -(\delta+1) & \delta & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -(\delta+1) & \delta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\beta & 0 & 1 \end{pmatrix} \begin{pmatrix} y_F \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_F \\ y_5 \end{pmatrix} = \begin{pmatrix} y_F \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ x_F \\ 0 \end{pmatrix}$$

Note: Assuming $y_5 = 0$ (complete separation) instead of making the equilibrium assumption $y_5 - \beta x_5 = 0$ will change the last row of the above matrix as having zero element instead of $-\beta$. This approach only affects the result of y_5 itself because y_5 does not participate in any other equations. **Both solutions will be accepted**.

1. The code below defines functions for the A matrix and the b vector, and then solves the system for the two cases. The liquid concentration profiles through the column are obtained as follow:

Case I ($L = 150$): Impurity [%] in the liquid stream, top to bottom: 5.00, 5.33, 6.15, 8.21, 13.3, 26.2; $y_5 = 3.20$

Case II ($L = 50$): Impurity [%] in the liquid stream, top to bottom: 5.00, 18.4, 29.5, 38.8, 46.5, 53.0; $y_5 = 11.0$

Discussion: The only difference between Case I and case II is the amount of the liquid solvent L used in the absorber. One expects that if a lower flow rate of solvent is used, yet the hold-up on each tray is the same, that the concentrations of the impurity in the liquid phase will be higher on each tray. This is indeed shown in the results. However, the outlet concentration of impurity in the gas phase is going to be higher (less impurity is removed) at the lower liquid flow rate, since absorption of the impurity into the liquid phase is inversely proportional to the impurity already in the liquid phase (due to concentration gradients).

You can verify for yourself that as the liquid flow rate is increased, keeping the gas flow constant, that the limiting outlet concentration in the gas phase is 3.0%, the amount that is in equilibrium with an "infinite" liquid composition of 5%. Also, the outlet liquid-phase concentration tends to 5% across all trays as $L \to \infty$.

MATLAB code

```
% Define functions for A and b.  Note: this form of defining a
% function is called an anonymous function.  The result of the
% statement on the right, assigned to the variable on the left,
% A_func, is what is known as a function handle.  Type
```

```
%   >> help function_handle for more information.
%
% You could have also defined a function, in a separate MATLAB
% file for this question.

A_func = @(b,d) [1 zeros(1,7);                ...
                 1/b -(d+1) d zeros(1,5);     ...
                 0 1 -(d+1) d zeros(1,4);     ...
                 0 0 1 -(d+1) d zeros(1,3);   ...
                 zeros(1,3) 1 -(d+1) d 0 0;   ...
                 zeros(1,4) 1 -(d+1) d 0;     ...
                 zeros(1,6) 1 0;              ...
                 zeros(1,5) -b 0 1];

% y_F, x_1, x_2, x_3, x_4, x_5, x_Feed, y_5 (outlet gas conc)
b_func = @(y_F,x_F) [y_F, 0, 0, 0, 0, 0, x_F, 0]';

% Case I:
G = 100.0;
L = 150.0;      % or use 50.0
beta = 0.6;
d = L/(G*beta);
y_F = 0.35;
x_F = 0.05;
A = A_func(beta, d);
b = b_func(y_F,x_F);
x = A\b;

% Check: norm(A*x - b)  = 0.0
x = x * 100;
fprintf('Gas_exit = %f\n', x(8))
fprintf ('Gas_f = %f\n', x(1))
fprintf(['Liquid concentration profile: top to bottom: %f, ' ...
         '%f, %f, %f, %f, %f\n'], x(7), x(6), x(5), x(4), x(3), x(2))
```

Python code

```
import numpy as np

G = 100.0
L = 150.0  # or use 50.0
b = 0.6
d = L/(G*b)
y_F = 0.35
x_F = 0.05

A = np.eye(8)
A[1,1] = A[2,2] = A[3,3] = A[4,4] = A[5,5] = -(d+1)
A[1,2] = A[2,3] = A[3,4] = A[4,5] = A[5,6] = d
A[2,1] = A[3,2] = A[4,3] = A[5,4] = 1.0
A[1,0] = 1/b
A[7,5] = -b

# y_F, x_1, x_2, x_3, x_4, x_5, x_Feed, y_5 (outlet gas conc)
b = np.array([y_F, 0, 0, 0, 0, 0, x_F, 0])
x = np.linalg.solve(A, b)
# np.linalg.norm(np.dot(A,x) - b)
x *= 100.0
print(x)
print('Gas_exit = %f' % x[7])
```

```
print('Gas_feed = %f' % x[0])
print(('Liquid concentration profile: top to bottom: %f, %f'
       '%f, %f, %f, %f' % (x[6], x[5], x[4], x[3], x[2], x[1])))
```

Question 4 [3]

Write a MATLAB or Python function that implements the Jacobi (simultaneous update) method for solving a general system of equations, $Ax = b$. The function should accept the following inputs:

- A: any $N \times N$ matrix.
- b: any $N \times 1$ vector.
- x_0: any $N \times 1$ vector that is the initial guess for x.
- tol: a small number, so that we will stop the algorithm when the relative difference between successive iterations is small: i.e. stop when $\frac{||x^{(k)} - x^{(k-1)}||}{||x^{(k)}||} <$ tol, where the norms are assumed to be the Euclidean norm.
- maxiter: an integer, for example, 200, representing an upper bound on the number of iterations we are willing to perform.

The function does not have to perform pivoting to achieve diagonal dominance. The function should calculate and print out, at every iteration, the relative difference between the k^{th} and $k-1^{\text{th}}$ iteration.

Consider this system of equations:

$$\begin{aligned} y_F &= 0.5 \\ G/Ly_F - (\delta+1)x_1 + x_2 &= 0 \\ \delta x_1 - (\delta+1)x_2 + x_3 &= 0 \\ \delta x_2 - (\delta+1)x_3 + x_F &= 0 \\ x_F &= 0.1 \end{aligned}$$

where $G/L = 1.6$, and $\delta = 0.8$

Your answer should include:

1. A print out of the function
2. The relative difference between the estimates for the first 5 iterations of solving the system of **five** equations, start from $y_F^{(0)} = x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = x_F^{(0)} = 0$.
3. The number of iterations required, and the final solution, using a termination tolerance of $\epsilon = 1 \times 10^{-9}$.
4. What is the condition number of A?

Solution

Given $Ax = b$, the Jacobi method can be derived as shown in class, or an alternative derivation is given here, which leads to a slightly cleaner implementation. Let $A = D + R$ where D is a diagonal matrix containing diagonal elements of A. Then $x^{k+1} = D^{-1}(b - Rx^k)$. This procedure is generically implemented as shown in the MATLAB code below, and gives exactly the same solution as the Python code (the Python code implements the version shown in the course notes).

Note the termination criteria is for either exceeding the maximum iterations **or** achieving the desired tolerance. Now for the given system of 5 equations, we can simply call the function we have written, and provide appropriate input values as given in the problem:

MATLAB code

```
% This code goes in a file called: jacobi.m

function [x, rel_error] = jacobi(A, b, x0, tol, maxiter)

% Implements the Jacobi method to solve Ax = b
% A = D + R where D is a diagonal matrix containing
% diagonal elements of A.
%
% NOTE: this an alternative algorithm to what was presented
% ----  in the course notes.  However, the course notes
%       algorithm gives the same values of x at each
%       iteration.

D = diag(diag(A));      % construct the diagonal matrix
R = A - D;              % Construct R
iter = 1;               % iteration counter
%x = inv(D)*(b - R*x0);  % Perform the first iteration
x = x0;
rel_error = tol * 2;% norm(x - x0)/norm(x);
%fprintf('\n Iteration %i: relative error =%d ',iter, rel_error)
%iter = iter + 1;
while (rel_error>tol && iter <= maxiter)

    xprev = x;
    x = inv(D)*(b - R*xprev);
    rel_error = norm(x - xprev)/norm(x);

    fprintf('\n Iteration %i: Relative error =%d ',iter, rel_error)
    iter = iter + 1;
end

% The rest of this code goes in a file called anything else,
% for example, question4_assign2.m - it uses the jacobi.m file
GL = 1.6;
d = 0.8;

A = [1.0,      0,      0,      0,   0; ...
      GL, -(d+1),    1.0,      0,   0; ...
       0,      d, -(d+1),    1.0,   0; ...
       0,      0,      d, -(d+1), 1.0; ...
       0,      0,      0,      0, 1.0];
b = [0.5,      0,      0,      0, 0.1]';

% Solver parameters
tol = 1e-9;
maxiter = 200;
x0 = zeros(5,1);

fprintf('\nCondition number of A=%d\n',cond(A))

% Solve the system, using our jacobi.m function file
[x, rel_error] = jacobi(A, b, x0, tol, maxiter);

fprintf('\nJacobi function: x=[%f %f %f %f %f]\n rel error=%d',...
    x, rel_error)
fprintf('norm(A*x-b)=%d',norm(A*x-b))   % 1.081579e-09
x=A\b;                                  % MATLAB's solver
fprintf('\nMATLAB built-in solver: x=[%f %f %f %f %f]\n',x)
```

Python code

```
import numpy as np
from numpy.linalg import *

def jacobi(A, b, x0, tol, maxiter=200):
    """
    Performs Jacobi iterations to solve the line system of
    equations, Ax=b, starting from an initial guess, ''x0''.

    Terminates when the change in x is less than ''tol'', or
    if ''maxiter'' [default=200] iterations have been exceeded.

    Returns 3 variables:
        1.  x, the estimated solution
        2.  rel_diff, the relative difference between last 2
            iterations for x
        3.  k, the number of iterations used.  If k=maxiter,
            then the required tolerance was not met.
    """
    n = A.shape[0]
    x = x0.copy()
    x_prev = x0.copy()
    k = 0
    rel_diff = tol * 2

    while (rel_diff > tol) and (k < maxiter):

        for i in range(0, n):
            subs = 0.0
            for j in range(0, n):
                if i != j:
                    subs += A[i,j] * x_prev[j]

            x[i] = (b[i] - subs ) / A[i,i]
        k += 1

        rel_diff = norm(x - x_prev) / norm(x)
        print(x, rel_diff)
        x_prev = x.copy()

    return x, rel_diff, k

# Main code starts here
# ---------------------
GL = 1.6
d = 0.8
A = np.array([
    [1.0,      0,      0,      0,   0],
    [ GL, -(d+1),    1.0,      0,   0],
    [  0,      d, -(d+1),    1.0,   0],
    [  0,      0,      d, -(d+1), 1.0],
    [  0,      0,      0,      0, 1.0]])
b = [0.5,      0,      0,      0, 0.1]
x0 = np.zeros(5);

tol = 1E-9
maxiter = 200
x, rel_diff, k = jacobi(A, b, x0, tol, maxiter)
if k == maxiter:
```

```
    print(('WARNING: the Jacobi iterations did not '
           'converge within the required tolerance.'))
print(('The solution is %s; within a tolerance of %g, '
       'using %d iterations.' % (x, rel_diff, k)))
print('Solution error = norm(Ax-b) = %g' % \
           norm(np.dot(A,x)-b))
print('Condition number of A = %0.5f' % cond(A))
print('Solution from built-in functions = %s' % solve(A, b))
```

The relative difference $\frac{||x^{(k)} - x^{(k-1)}||}{||x^{(k)}||}$ and the first five iterations is given as:

```
         X-vector updates                                          Tolerance
Iter 0 : [ 0.5,  0.        , -0.        , -0.        ,  0.1], 1.0
Iter 1 : [ 0.5,  0.44444444, -0.        ,  0.05555556,  0.1], 0.65995
Iter 2 : [ 0.5,  0.44444444,  0.22839506,  0.05555556,  0.1], 0.31895
Iter 3 : [ 0.5,  0.57133059,  0.22839506,  0.15706447,  0.1], 0.19952
Iter 4 : [ 0.5,  0.57133059,  0.34118275,  0.15706447,  0.1], 0.13224
```

It is seen that the relative difference is decreasing. The number of iterations required is 58 to achieve a relative error of 8.604×10^{-10}, less than the required 1×10^{-9}.

The solution at the 58th iteration is $x = [0.500000, 0.695122, 0.451220, 0.256098, 0.100000]$, which agrees with the solution found using built-in functions.

The condition number of A can be obtained by computing $||A||||A^{-1}||$ or using MATLAB's built-in function as cond(A). The result is $||A||||A^{-1}|| = 8.7259$. Note that it is the 2-norm condition number.

Question 5 [2]

Note: From a previous exam, 5 marks out of 50, 3 hours. This question should be solved by hand, and is a typical question you can expect in the midterm and final.

For the following system of linear algebraic equations:

$$\begin{aligned} -5x_1 + 0x_2 + 12x_3 &= 80 \\ 4x_1 - x_2 - x_3 &= -2 \\ 6x_1 + 8x_2 + 0x_3 &= 45 \end{aligned}$$

1. Use the Jacobi method to compute $x^{(1)}$ starting from $x^{(0)} = \mathbf{0}$.
2. Use the Gauss-Seidel method to compute $x^{(1)}$ starting from $x^{(0)} = \mathbf{0}$.
3. Use the Gauss-Seidel method with relaxation, $\omega = 0.9$, to compute $x^{(1)}$ starting from $x^{(0)} = \mathbf{0}$.

Solution

We start this problem by converting the system to the form $Ax = b$.

$$\begin{bmatrix} -5 & 0 & 12 \\ 4 & -1 & -1 \\ 6 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 80 \\ -2 \\ 45 \end{bmatrix}$$

Next we rearrange the system so that it is diagonally dominant (i.e. $|a_{ii}| \geq \sum_{j=1,j\neq i}^{n} |a_{ij}|$ where ever possible). To achieve this we rearrange the equation order to equation 2, equation 3, equation 1.

$$\begin{bmatrix} 4 & -1 & -1 \\ 6 & 8 & 0 \\ -5 & 0 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 45 \\ 80 \end{bmatrix}$$

Now we are ready to attack this problem.

1. The Jacobi method uses the following update algorithm:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1,j\neq i}^{n} a_{ij} x_j^{(k)} \right]$$

   Using a starting value of $x^{(0)} = [0, 0, 0]$ we calculate the next iterate using the above equation.

$$\begin{aligned} x_1^{(1)} &= \tfrac{1}{a_{11}} \left[b_1 - \left(a_{12} x_2^{(0)} + a_{13} x_3^{(0)} \right) \right] \\ x_1^{(1)} &= \tfrac{1}{(4)} \left[(-2) - ((-1)(0) + (-1)(0)) \right] \\ x_1^{(1)} &= -0.5 \end{aligned}$$

$$\begin{aligned} x_2^{(1)} &= \tfrac{1}{a_{22}} \left[b_2 - \left(a_{21} x_1^{(0)} + a_{23} x_3^{(0)} \right) \right] \\ x_2^{(1)} &= \tfrac{1}{(8)} \left[(45) - ((6)(0) + (0)(0)) \right] \\ x_2^{(1)} &= 5.625 \end{aligned}$$

$$\begin{aligned} x_3^{(1)} &= \tfrac{1}{a_{33}} \left[b_3 - \left(a_{31} x_1^{(0)} + a_{32} x_2^{(0)} \right) \right] \\ x_3^{(1)} &= \tfrac{1}{(12)} \left[(80) - ((-5)(0) + (0)(0)) \right] \\ x_3^{(1)} &= 6.6667 \end{aligned}$$

   Therefore $x^{(1)} = [-0.5, 5.625, 6.6667]$

2. The Gauss-Seidel method uses the following update algorithm:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right]$$

   Using a starting value of $x^{(0)} = [0, 0, 0]$ we calculate the next iterate using the above equation.

$$\begin{aligned} x_1^{(1)} &= \tfrac{1}{a_{11}} \left[b_1 - \left(a_{12} x_2^{(0)} + a_{13} x_3^{(0)} \right) \right] \\ x_1^{(1)} &= \tfrac{1}{(4)} \left[(-2) - ((-1)(0) + (-1)(0)) \right] \\ x_1^{(1)} &= -0.5 \end{aligned}$$

$$\begin{aligned} x_2^{(1)} &= \tfrac{1}{a_{22}} \left[b_2 - \left(a_{21} x_1^{(1)} + a_{23} x_3^{(0)} \right) \right] \\ x_2^{(1)} &= \tfrac{1}{(8)} \left[(45) - ((6)(-0.5) + (0)(0)) \right] \\ x_2^{(1)} &= 6 \end{aligned}$$

$$\begin{aligned} x_3^{(1)} &= \tfrac{1}{a_{33}} \left[b_3 - \left(a_{31} x_1^{(1)} + a_{32} x_2^{(1)} \right) \right] \\ x_3^{(1)} &= \tfrac{1}{(12)} \left[(80) - ((-5)(-0.5) + (0)(6)) \right] \\ x_3^{(1)} &= 6.4583 \end{aligned}$$

   Therefore $x^{(1)} = [-0.5, 6, 6.4583]$

3. The Gauss-Seidel method with relaxation uses the following update algorithm:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right]$$

Using a starting value of $x^{(0)} = [0,0,0]$ we calculate the next iterate using the above equation.

$$\begin{aligned} x_1^{(1)} &= (1-\omega)x_1^{(0)} + \tfrac{\omega}{a_{11}}\left[b_1 - \left(a_{12}x_2^{(0)} + a_{13}x_3^{(0)}\right)\right] \\ x_1^{(1)} &= (1-(0.9))(0) + \tfrac{(0.9)}{(4)}\left[(-2) - ((-1)(0) + (-1)(0))\right] \\ x_1^{(1)} &= -0.45 \end{aligned}$$

$$\begin{aligned} x_2^{(1)} &= (1-\omega)x_2^{(0)} + \tfrac{\omega}{a_{22}}\left[b_2 - \left(a_{21}x_1^{(1)} + a_{23}x_3^{(0)}\right)\right] \\ x_2^{(1)} &= (1-(0.9))(0) + \tfrac{(0.9)}{(8)}\left[(45) - ((6)(-0.45) + (0)(0))\right] \\ x_2^{(1)} &= 5.36625 \end{aligned}$$

$$\begin{aligned} x_3^{(1)} &= (1-\omega)x_3^{(0)} + \tfrac{\omega}{a_{33}}\left[b_3 - \left(a_{31}x_1^{(1)} + a_{32}x_2^{(1)}\right)\right] \\ x_3^{(1)} &= (1-(0.9))(0) + \tfrac{(0.9)}{(12)}\left[(80) - ((-5)(-0.45) + (0)(5.36625))\right] \\ x_3^{(1)} &= 5.83125 \end{aligned}$$

Therefore $x^{(1)} = [-0.45, 5.36625, 5.83125]$

Bonus question [1]

Use the pseudo code developed in the course notes to write a MATLAB or Python function that implements Gauss elimination, without pivoting. The function should take A and b as inputs, and return vector x.

Use your function to solve this set of equations from question 1, and print the update A matrix and b vector after every pivot step.

Solution

A function that implements the Gauss elimination without pivoting is provided below.

MATLAB code

```
function x = gauss(A,b)
% This function performs the Gauss elimination without pivoting
%
% x = GAUSS(A, b)

[n,n] = size(A);

% Check for zero diagonal elements
if any(diag(A)==0)
    error('Division by zero will occur; pivoting not supported')
end

% Forward elimination
for row=1:n-1
    for i=row+1:n
        factor = A(i,row) / A(row,row);
```

```
        for j = row:n
            A(i,j) = A(i,j) - factor*A(row,j);
        end
        b(i) = b(i) - factor*b(row);
    end
  A_and_b =  [A b]
end

% Backward substitution
x(n) = b(n) / A(n,n);
for row = n-1:-1:1
    sums = b(row);
    for j = row+1: n
        sums = sums - A(row,j) * x(j);
    end
    x(row) = sums / A(row,row);
end

%Now run this in the command window or an M-file:
% >> A = [1 1 1 1;1 -2 -2 -2;1 4 -4 1;1 -5 -5 -3];
% >> b = [0 4 2 -4]';
% >> x = gauss(A,b)
```

Python code

```
import numpy as np

def forward_elimination(A, b, n):
    """
    Calculates the forward part of Gaussian elimination.
    """
    for row in range(0, n-1):
        for i in range(row+1, n):
            factor = A[i,row] / A[row,row]
            for j in range(row, n):
                A[i,j] = A[i,j] - factor * A[row,j]

            b[i] = b[i] - factor * b[row]

        print('A = \n%s and b = %s' % (A,b))
    return A, b

def back_substitution(a, b, n):
    """"
    Does back substitution, returns the Gauss result.
    """
    x = np.zeros((n,1))
    x[n-1] = b[n-1] / a[n-1, n-1]
    for row in range(n-2, -1, -1):
        sums = b[row]
        for j in range(row+1, n):
            sums = sums - a[row,j] * x[j]
        x[row] = sums / a[row,row]
    return x

def gauss(A, b):
    """
    This function performs Gauss elimination without pivoting.
    """
    n = A.shape[0]
```

```
    # Check for zero diagonal elements
    if any(np.diag(A)==0):
        raise ZeroDivisionError(('Division by zero will occur; '
                                 'pivoting currently not supported'))

    A, b = forward_elimination(A, b, n)
    return back_substitution(A, b, n)

# Main program starts here
if __name__ == '__main__':
    A = np.array([[1,   1,   1, 1],
                  [1, -2,  -2, -2],
                  [1,   4, -4, 1],
                  [1, -5,  -5, -3]])
    b = np.array([0, 4, 2, -4])
    x = gauss(A, b)
    print('Gauss result is x = \n %s' % x)
```

Now we use this function to solve the system of equations in Question 1. The commented lines in the MATLAB code show how the function is used. The results are: $x = [1.3333, 3.1667, 1.5000, -6.0000]$. Also, the A and b (denoted as $C = [A \quad b]$ below) are updated as:

$$C^1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & -3 & -3 & -3 & 4 \\ 0 & 3 & -5 & 0 & 2 \\ 0 & -6 & -6 & -4 & -4 \end{bmatrix}$$

$$C^2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & -3 & -3 & -3 & 4 \\ 0 & 0 & -8 & -3 & 6 \\ 0 & 0 & 0 & 2 & -12 \end{bmatrix}$$

$$C^3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & -3 & -3 & -3 & 4 \\ 0 & 0 & -8 & -3 & 6 \\ 0 & 0 & 0 & 2 & -12 \end{bmatrix}$$

The final result for $x = [1.333, 3.167, 1.5, -6]$.

END