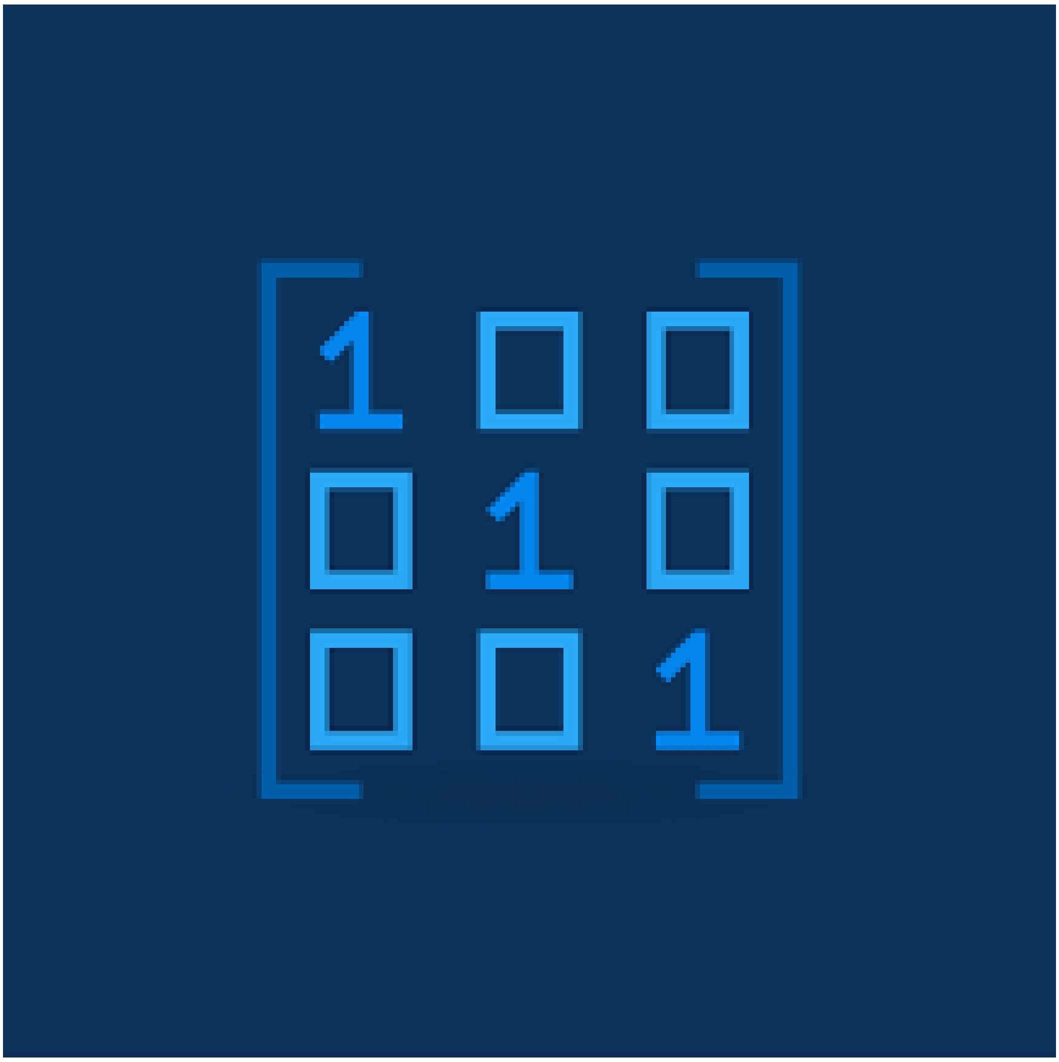
Algorithm to find the Inverse of any 3x3 Invertible Matrix

< Using Elementary Operations >



Sarang Galada, XII | Math Project - Section A | 2020-21

Contents:

Aim: To generalize a standard algorithm (sequence of steps), following which the inverse of any 3x3 invertible matrix can be found.

Concept:

[Notations: 'A'=given matrix, 'I'=unit matrix, LHS=Left hand side, RHS=Right hand side, Eq=Equation]

From Linear Algebra we have the following relations:

1. The inverse of any matrix 'A' is defined by... $\boxed{AA^{-1} = I}$ (Eq.1) [where 'I' is the unit matrix].
2. Any matrix 'A' can be written as... $\boxed{A = IA}$ (Eq.2)

Now, our aim is to convert the form $\boxed{A = IA}$ to $\boxed{I = AA^{-1}}$ so that we can obtain the inverse.

How do we do this?

To be able to convert Eq.2 to the form of Eq.1, we will have to perform Elementary operations. These can be applied either strictly row-wise or strictly column-wise. To maintain equality, the same operations would have to be applied to both LHS and RHS of the equation.

Let us assume that on transforming the 'A' [in LHS of Eq.2] to the unit matrix 'I', the 'I' [in RHS] is transformed into some matrix 'B'. So, $\boxed{A = IA}$ becomes $\boxed{I = BA}$. Comparing this new form to Eq.1, we see that 'B' corresponds to the A^{-1} [in LHS of Eq.1]. Since **there exists only one unique inverse of a given matrix**, we can conclude that matrix 'B' obtained is the inverse of A.

The Procedure's Logical Outline:

1. Write the matrix 'A' as *A = IA.*
2. Use elementary operations (strictly row-wise or column-wise) to convert the matrix 'A' into the unit matrix 'I'.
3. At each above step, apply the exact operations to the 'I' in RHS.
4. The matrix obtained in step 3 is the inverse of A. [ie. A^{-1}]

Having understood the brief outline of the task, we must now devise a sequence of steps to achieve the transformations using elementary procedures – which is clearly the most crucial part.

There are 3 main elementary matrix procedures:

1. Interchanging any 2 rows/columns.
2. Multiplying any row/column by a non-zero scalar quantity.
3. Adding/subtracting any scalar multiple of any other row/column to the given row/column.

Since our desired "end-product" in LHS is a unit matrix, we must convert the diagonal elements of matrix A into 1s and all other elements into 0s.

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The above 3 procedures are sufficient for us to do this and so find the inverse of any given matrix.

Devising an Algorithm for the Elementary Operations

[We shall explore the algorithm separately for Row-wise and Column-wise operations]

1. Make the top-left corner element a 1.

   - If any other existing row/column already has a 1 in its 1st position, it can be interchanged with the 1st row/column to make the top-left corner a 1.

eg.

Row-wise

$$\begin{bmatrix} -1 & 1 & 2 \\ 1 & 2 & 3 \\ 3 & 1 & 1 \end{bmatrix} \xrightarrow{R_1 \leftrightarrow R_2} \begin{bmatrix} 1 & 2 & 3 \\ -1 & 1 & 2 \\ 3 & 1 & 1 \end{bmatrix}$$

Column-wise

$$\begin{bmatrix} -1 & 1 & 2 \\ 1 & 2 & 3 \\ 3 & 1 & 1 \end{bmatrix} \xrightarrow{C_1 \leftrightarrow C_2} \begin{bmatrix} 1 & -1 & 2 \\ 2 & 1 & 3 \\ 1 & 3 & 1 \end{bmatrix}$$

   - In the absence of the above, a simple scaling or addition operation can achieve the same. This works for all matrices.

eg.

Row-wise

$$\begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} \xrightarrow{R_1 \to R_1 - \frac{1}{5}R_2} \begin{bmatrix} 1 & -1/5 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix}$$

OR

$$\begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} \xrightarrow{R_1 \to \frac{1}{2}R_1} \begin{bmatrix} 1 & 0 & -1/2 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix}$$

Column-wise

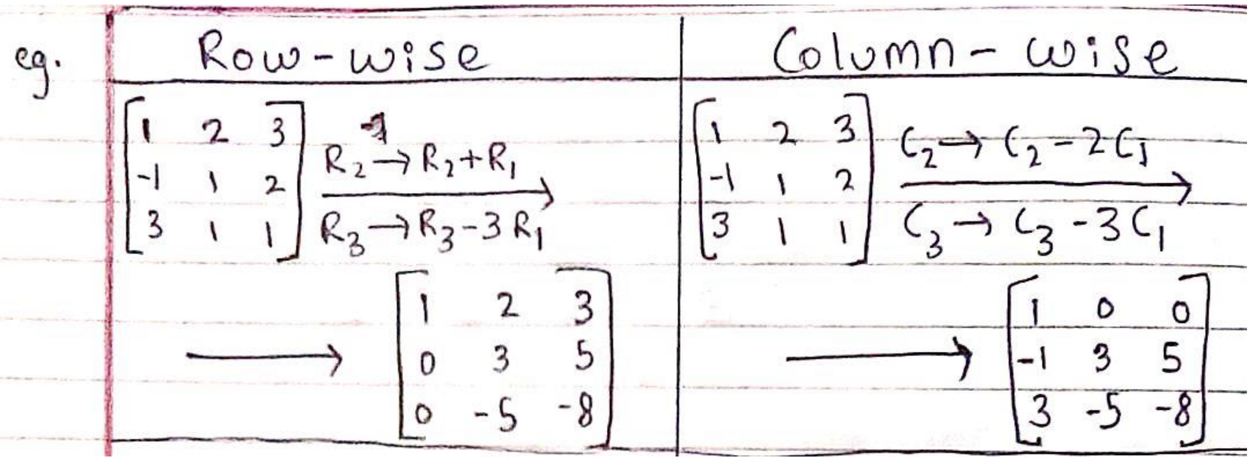
$$\begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} \xrightarrow{C_1 \to C_1 + C_3} \begin{bmatrix} 1 & 0 & -1 \\ 5 & 1 & 0 \\ 3 & 1 & 3 \end{bmatrix}$$

OR

$$\begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} \xrightarrow{C_1 \to 1/2\, C_1} \begin{bmatrix} 1 & 0 & -1 \\ 5/2 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix}$$

2. **Convert the other elements of that row/column into 0s.**

    - This can be done by subtracting 1 times the value of the element (which equals 0), as shown below.



eg.

Row-wise

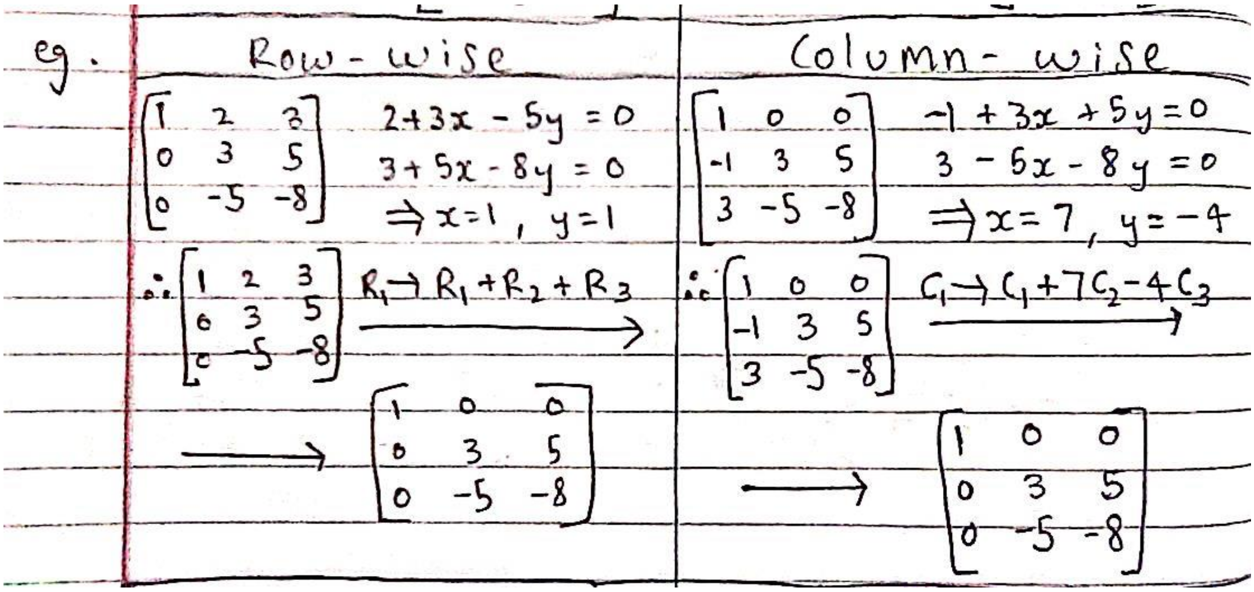
$$\begin{bmatrix} 1 & 2 & 3 \\ -1 & 1 & 2 \\ 3 & 1 & 1 \end{bmatrix} \xrightarrow[R_3 \to R_3 - 3R_1]{R_2 \to R_2 + R_1} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 5 \\ 0 & -5 & -8 \end{bmatrix}$$

Column-wise

$$\begin{bmatrix} 1 & 2 & 3 \\ -1 & 1 & 2 \\ 3 & 1 & 1 \end{bmatrix} \xrightarrow[C_3 \to C_3 - 3C_1]{C_2 \to C_2 - 2C_1} \begin{bmatrix} 1 & 0 & 0 \\ -1 & 3 & 5 \\ 3 & -5 & -8 \end{bmatrix}$$

3. **Convert the other elements perpendicular to the corner 1 into 0s.**

    - Form the following equations to solve for the scaling factors using which the other 2 elements can be made into 0s. Apply the operations as shown.



eg.

Row-wise

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 5 \\ 0 & -5 & -8 \end{bmatrix} \quad \begin{aligned} 2 + 3x - 5y &= 0 \\ 3 + 5x - 8y &= 0 \\ \Rightarrow x = 1, \ y &= 1 \end{aligned}$$

$$\therefore \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 5 \\ 0 & -5 & -8 \end{bmatrix} \xrightarrow{R_1 \to R_1 + R_2 + R_3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 5 \\ 0 & -5 & -8 \end{bmatrix}$$

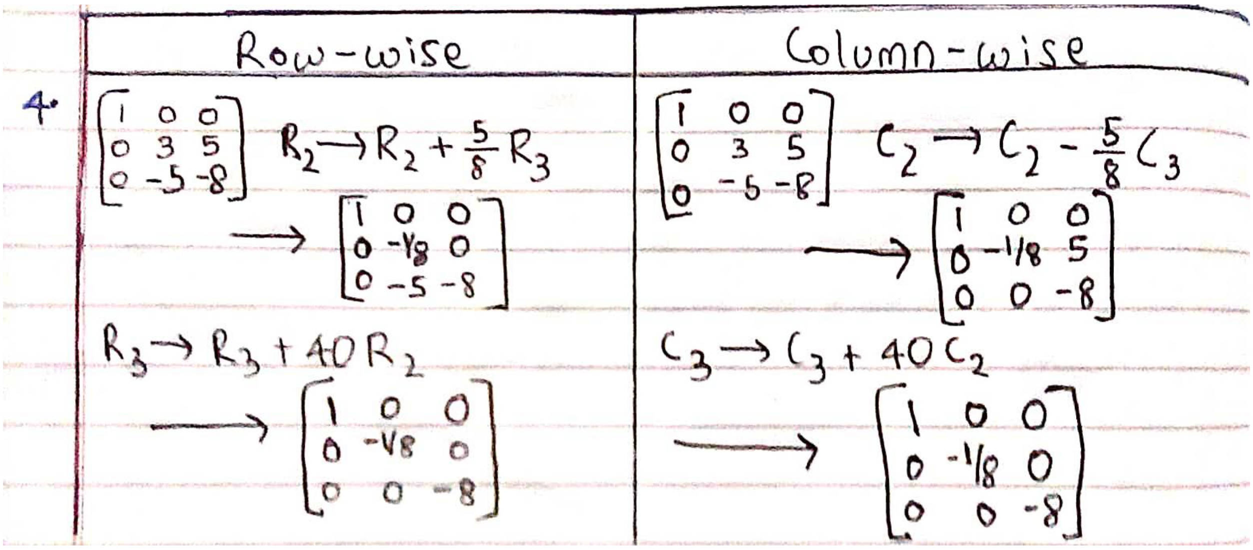
Column-wise

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 3 & 5 \\ 3 & -5 & -8 \end{bmatrix} \quad \begin{aligned} -1 + 3x + 5y &= 0 \\ 3 - 5x - 8y &= 0 \\ \Rightarrow x = 7, \ y &= -4 \end{aligned}$$

$$\therefore \begin{bmatrix} 1 & 0 & 0 \\ -1 & 3 & 5 \\ 3 & -5 & -8 \end{bmatrix} \xrightarrow{C_1 \to C_1 + 7C_2 - 4C_3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 5 \\ 0 & -5 & -8 \end{bmatrix}$$

4. **Convert the remaining non-diagonal elements into 0s.**

   - To do this, subtract the scaled version of the adjacent row/column such that the resulting value equals 0 as shown.

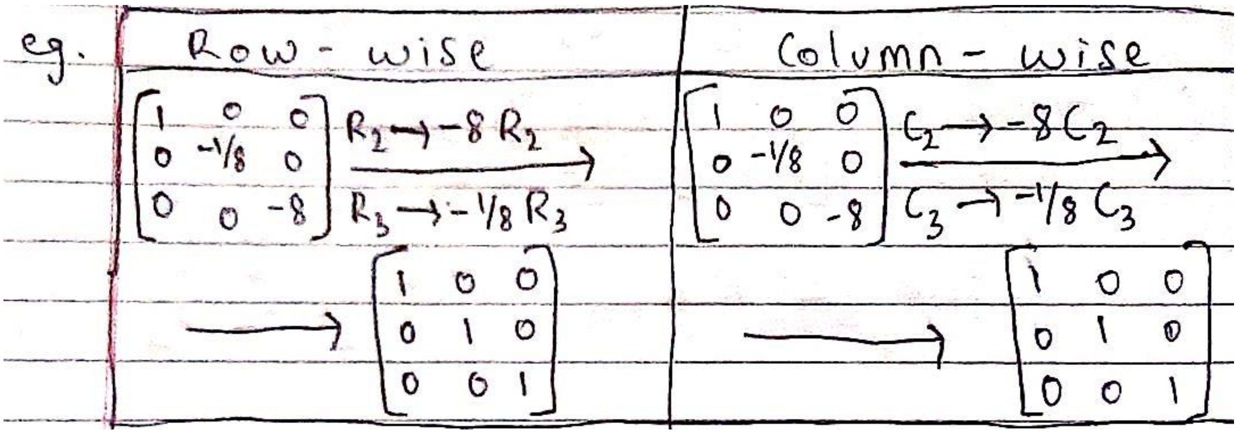


4.

Row-wise	Column-wise
$\begin{bmatrix}1&0&0\\0&3&5\\0&-5&-8\end{bmatrix}$ $R_2 \to R_2 + \frac{5}{8}R_3$	$\begin{bmatrix}1&0&0\\0&3&5\\0&-5&-8\end{bmatrix}$ $C_2 \to C_2 - \frac{5}{8}C_3$
$\longrightarrow \begin{bmatrix}1&0&0\\0&-1/8&0\\0&-5&-8\end{bmatrix}$	$\longrightarrow \begin{bmatrix}1&0&0\\0&-1/8&5\\0&0&-8\end{bmatrix}$
$R_3 \to R_3 + 40R_2$	$C_3 \to C_3 + 40C_2$
$\longrightarrow \begin{bmatrix}1&0&0\\0&-1/8&0\\0&0&-8\end{bmatrix}$	$\longrightarrow \begin{bmatrix}1&0&0\\0&-1/8&0\\0&0&-8\end{bmatrix}$

5. **Convert the remaining diagonal elements into 1s.**

   - Multiply the row/column with the reciprocal of the diagonal element's value.

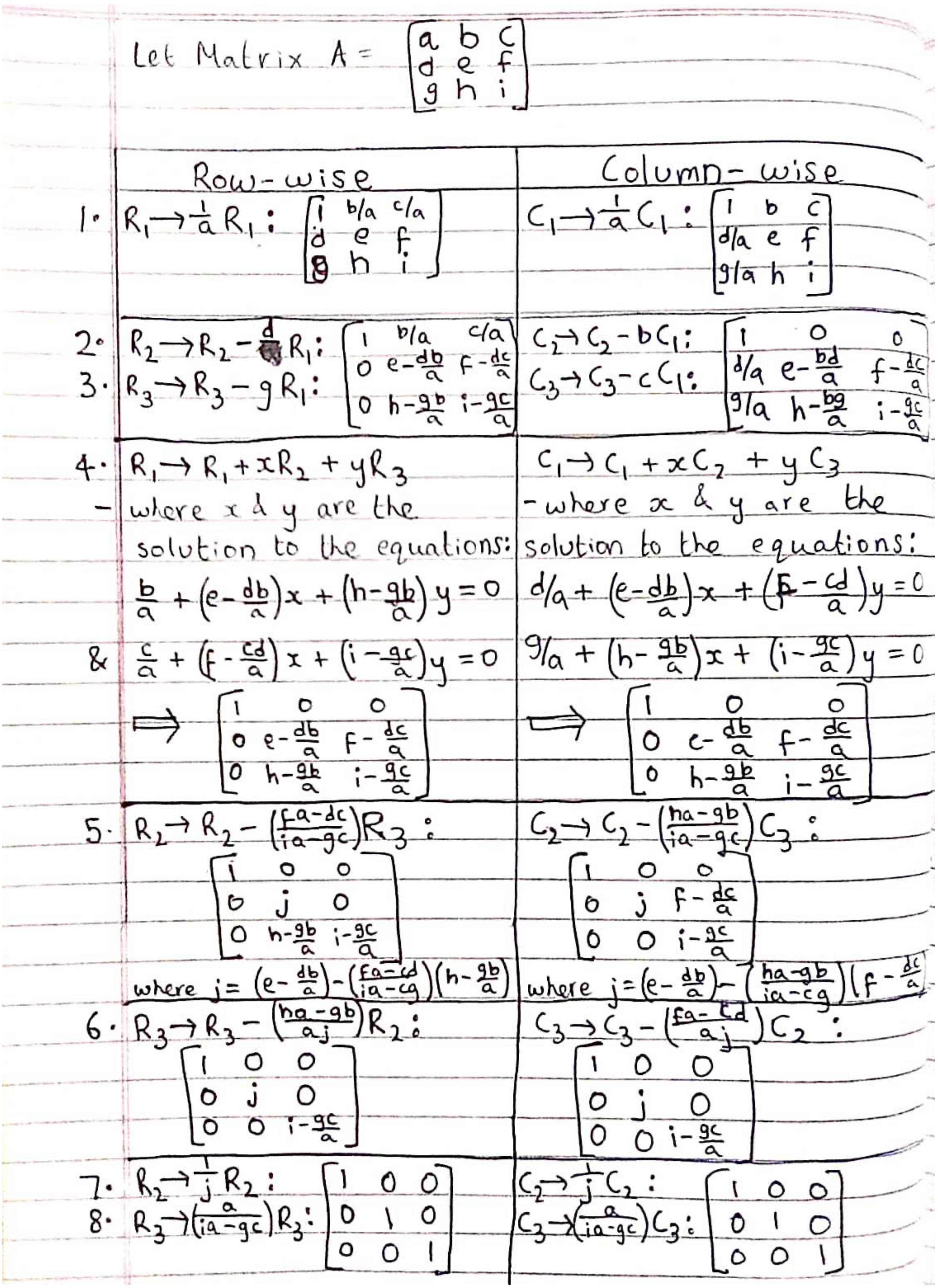


eg.

Row-wise	Column-wise
$\begin{bmatrix}1&0&0\\0&-1/8&0\\0&0&-8\end{bmatrix} \xrightarrow[R_3 \to -1/8 R_3]{R_2 \to -8R_2}$	$\begin{bmatrix}1&0&0\\0&-1/8&0\\0&0&-8\end{bmatrix} \xrightarrow[C_3 \to -1/8 C_3]{C_2 \to -8C_2}$
$\longrightarrow \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$	$\longrightarrow \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$

And thus we have converted the given matrix into a unit matrix. Applying the exact procedures to the unit matrix in RHS of Eq.2, we can obtain the inverse of the matrix.

The Generalised Algorithm:

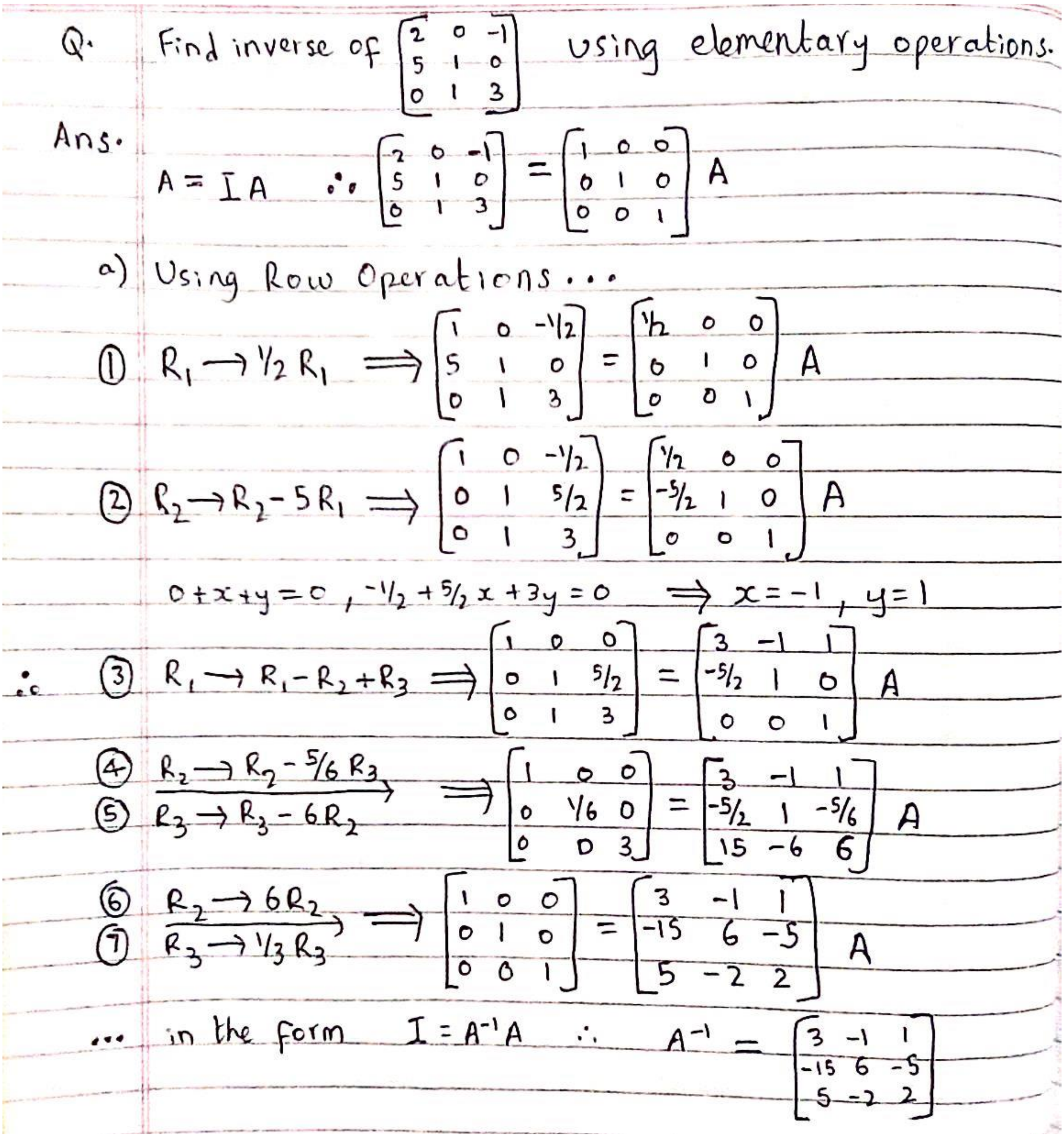


Let Matrix $A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$

	Row-wise	Column-wise
1.	$R_1 \to \frac{1}{a} R_1$: $\begin{bmatrix} 1 & b/a & c/a \\ d & e & f \\ g & h & i \end{bmatrix}$	$C_1 \to \frac{1}{a} C_1$: $\begin{bmatrix} 1 & b & c \\ d/a & e & f \\ g/a & h & i \end{bmatrix}$
2. 3.	$R_2 \to R_2 - \frac{d}{a} R_1$: $R_3 \to R_3 - g R_1$: $\begin{bmatrix} 1 & b/a & c/a \\ 0 & e-\frac{db}{a} & f-\frac{dc}{a} \\ 0 & h-\frac{gb}{a} & i-\frac{gc}{a} \end{bmatrix}$	$C_2 \to C_2 - bC_1$: $C_3 \to C_3 - cC_1$: $\begin{bmatrix} 1 & 0 & 0 \\ d/a & e-\frac{bd}{a} & f-\frac{dc}{a} \\ g/a & h-\frac{bg}{a} & i-\frac{gc}{a} \end{bmatrix}$
4. -	$R_1 \to R_1 + xR_2 + yR_3$ where x & y are the solution to the equations: $\frac{b}{a} + \left(e-\frac{db}{a}\right)x + \left(h-\frac{gb}{a}\right)y = 0$ & $\frac{c}{a} + \left(f-\frac{cd}{a}\right)x + \left(i-\frac{gc}{a}\right)y = 0$ $\Longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & e-\frac{db}{a} & f-\frac{dc}{a} \\ 0 & h-\frac{gb}{a} & i-\frac{gc}{a} \end{bmatrix}$	$C_1 \to C_1 + xC_2 + yC_3$ - where x & y are the solution to the equations: $d/a + \left(e-\frac{db}{a}\right)x + \left(f-\frac{cd}{a}\right)y = 0$ $g/a + \left(h-\frac{gb}{a}\right)x + \left(i-\frac{gc}{a}\right)y = 0$ $\Longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & e-\frac{db}{a} & f-\frac{dc}{a} \\ 0 & h-\frac{gb}{a} & i-\frac{gc}{a} \end{bmatrix}$
5.	$R_2 \to R_2 - \left(\frac{fa-dc}{ia-gc}\right)R_3$: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & j & 0 \\ 0 & h-\frac{gb}{a} & i-\frac{gc}{a} \end{bmatrix}$ where $j = \left(e-\frac{db}{a}\right) - \left(\frac{fa-cd}{ia-cg}\right)\left(h-\frac{gb}{a}\right)$	$C_2 \to C_2 - \left(\frac{ha-gb}{ia-gc}\right)C_3$: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & j & f-\frac{dc}{a} \\ 0 & 0 & i-\frac{gc}{a} \end{bmatrix}$ where $j = \left(e-\frac{db}{a}\right) - \left(\frac{ha-gb}{ia-cg}\right)\left(f-\frac{dc}{a}\right)$
6.	$R_3 \to R_3 - \left(\frac{ha-gb}{aj}\right)R_2$: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & j & 0 \\ 0 & 0 & i-\frac{gc}{a} \end{bmatrix}$	$C_3 \to C_3 - \left(\frac{fa-cd}{aj}\right)C_2$: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & j & 0 \\ 0 & 0 & i-\frac{gc}{a} \end{bmatrix}$
7. 8.	$R_2 \to \frac{1}{j} R_2$: $R_3 \to \left(\frac{a}{ia-gc}\right)R_3$: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$C_2 \to \frac{1}{j} C_2$: $C_3 \to \left(\frac{a}{ia-gc}\right)C_3$: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Solving an Example:

[A]: Row-wise:



Q. Find inverse of $\begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix}$ using elementary operations.

Ans.

$A = IA \quad \therefore \begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} A$

a) Using Row Operations . . .

① $R_1 \to 1/2\, R_1 \implies \begin{bmatrix} 1 & 0 & -1/2 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} A$

② $R_2 \to R_2 - 5R_1 \implies \begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 1 & 5/2 \\ 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 0 \\ -5/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} A$

$0 + x + y = 0 \;,\; -1/2 + 5/2\, x + 3y = 0 \implies x = -1 \;,\; y = 1$

$\therefore$ ③ $R_1 \to R_1 - R_2 + R_3 \implies \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 5/2 \\ 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 1 \\ -5/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} A$

④ $R_2 \to R_2 - 5/6\, R_3$

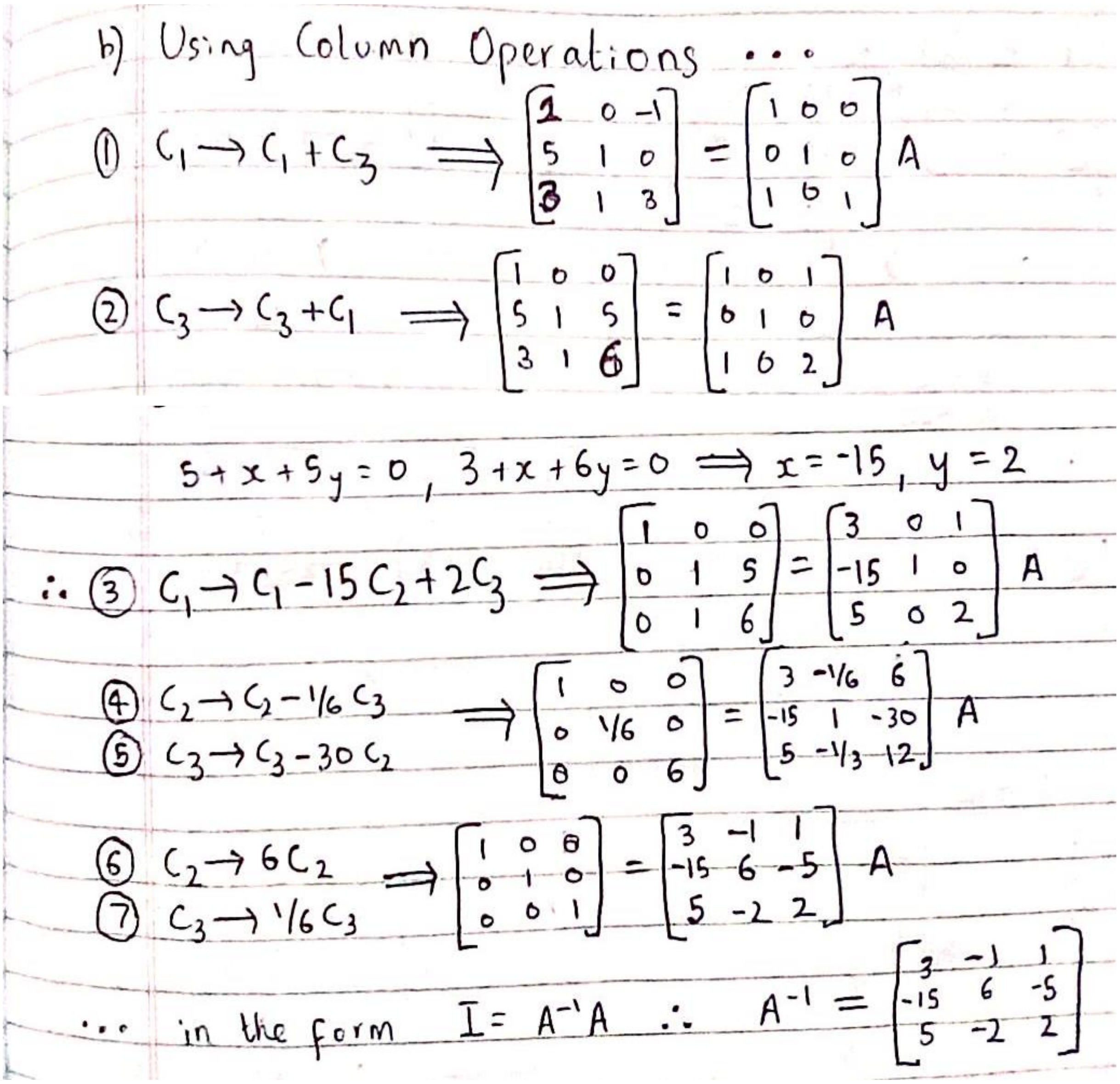
⑤ $R_3 \to R_3 - 6R_2 \implies \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/6 & 0 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 1 \\ -5/2 & 1 & -5/6 \\ 15 & -6 & 6 \end{bmatrix} A$

⑥ $R_2 \to 6R_2$

⑦ $R_3 \to 1/3\, R_3 \implies \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 1 \\ -15 & 6 & -5 \\ 5 & -2 & 2 \end{bmatrix} A$

$\because$ in the form $I = A^{-1}A \quad \therefore \quad A^{-1} = \begin{bmatrix} 3 & -1 & 1 \\ -15 & 6 & -5 \\ 5 & -2 & 2 \end{bmatrix}$

[B]: <u>Column-wise</u>:



b) Using Column Operations ...

① $C_1 \rightarrow C_1 + C_3 \implies \begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 3 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} A$

② $C_3 \rightarrow C_3 + C_1 \implies \begin{bmatrix} 1 & 0 & 0 \\ 5 & 1 & 5 \\ 3 & 1 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix} A$

$5 + x + 5y = 0, \; 3 + x + 6y = 0 \implies x = -15, \; y = 2$

$\therefore$ ③ $C_1 \rightarrow C_1 - 15C_2 + 2C_3 \implies \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 1 & 6 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 1 \\ -15 & 1 & 0 \\ 5 & 0 & 2 \end{bmatrix} A$

④ $C_2 \rightarrow C_2 - 1/6\, C_3$

⑤ $C_3 \rightarrow C_3 - 30\, C_2$

$\implies \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/6 & 0 \\ 0 & 0 & 6 \end{bmatrix} = \begin{bmatrix} 3 & -1/6 & 6 \\ -15 & 1 & -30 \\ 5 & -1/3 & 12 \end{bmatrix} A$

⑥ $C_2 \rightarrow 6C_2$

⑦ $C_3 \rightarrow 1/6\, C_3$

$\implies \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 1 \\ -15 & 6 & -5 \\ 5 & -2 & 2 \end{bmatrix} A$

... in the form $I = A^{-1}A \;\therefore\; A^{-1} = \begin{bmatrix} 3 & -1 & 1 \\ -15 & 6 & -5 \\ 5 & -2 & 2 \end{bmatrix}$

Java Computer Code:

The following Java code implements the above algorithm and calculates and displays the inverse of a user-defined matrix.

The code: [can be executed in any Command-line interface or basic Java IDE]

```
import java.util.Scanner;
public class InverseMat {
    Scanner input=new Scanner(System.in);
    float[][] a=new float[3][3]; float[][] I={{1,0,0},{0,1,0},{0,0,1}};
    void getMat(){
        for(int i=0;i<a.length;i++){
            for(int j=0;j<a[i].length;j++){
                System.out.print("Enter element "+(j+1)+" of row "+(i+1)+":  ");
                a[i][j] = input.nextFloat();
            }
        }
    }
    void Inverse(){
        float x=(a[1][2]*a[0][1]-a[1][1]*a[0][2])/(a[2][2]*a[1][1]-a[2][1]*a[1][2]);
        float y=(a[2][1]*a[0][2]-a[2][2]*a[0][1])/(a[2][2]*a[1][1]-a[2][1]*a[1][2]);
        for(int j=0;j<3;j++){
            I[0][j]+=(y*I[1][j]+x*I[2][j]);
            a[0][j]+=(y*a[1][j]+x*a[2][j]);
        }
        for(int j=0;j<3;j++){
            I[0][j]/=a[0][0];
        }
        for(int j=0;j<3;j++){
            a[0][j]/=a[0][0];
        }
        for(int i=1;i<3;i++){
            for(int j=0;j<3;j++){
                I[i][j]-=a[i][0]*I[0][j];
            }
        }
        for(int i=1;i<3;i++){
                for(int j=0;j<3;j++){
                    a[i][j]-=a[i][0]*a[0][j];
                }
            }
            for(int j=0;j<3;j++){
                I[1][j]-=(a[1][2]/a[2][2])*I[2][j];
            }
            for(int j=0;j<3;j++){
                a[1][j]-=(a[1][2]/a[2][2])*a[2][j];
            }
            for(int j=0;j<3;j++){
                I[2][j]-=(a[2][1]/a[1][1])*I[1][j];
            }
            for(int j=0;j<3;j++){
                a[2][j]-=(a[2][1]/a[1][1])*a[1][j];
            }
            for(int i=1;i<3;i++){
                for(int j=0;j<3;j++){
                    I[i][j]/=a[i][i];
                }
            }
        }
        void display(float[][] m){
            System.out.println();
            for(int i=0;i<3;i++){
                for(int j=0;j<3;j++){
                    System.out.print(m[i][j]+"  ");}
                System.out.println();
            }
        }
        public static void main(String[] args) {
            InverseMat im=new InverseMat();
            im.getMat();im.Inverse();
            System.out.println("Inverse of your matrix is...");im.display(im.I);
        }
    }
```

Test Run:

User input

```
Enter element 1 of row1: 2
Enter element 2 of row1: 0
Enter element 3 of row1: -1
Enter element 1 of row2: 5
Enter element 2 of row2: 1
Enter element 3 of row2: 0
Enter element 1 of row3: 0
Enter element 2 of row3: 1
Enter element 3 of row3: 3
Inverse of your matrix is...
```

$$\begin{bmatrix} 2 & 0 & -1 \\ 5 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix}$$

```
3.0000007  -1.0000002  1.0000002
-15.000004  6.000001  -5.000001
5.0000014  -2.0000002  2.0000002
```

$$\begin{bmatrix} 3 & -1 & 1 \\ -15 & 6 & -5 \\ 5 & -2 & 2 \end{bmatrix}$$

...Which is the same as what we calculated previously

[Note: the miniscule decimal values in the output exist due to the decimal rounding-off that java executes in each calculation. These values get carried over to the final output; hence their presence.]

Higher Order Matrices:

According to our algorithm, the general procedure would be:

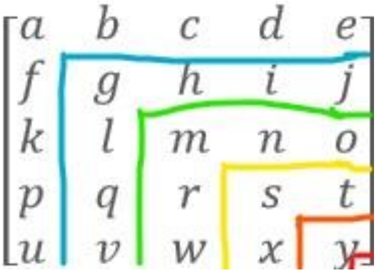
1. **Make top right element a 1.** => 1 operation
2. **Convert all elements below it into 0s**. => (n-1) operations
3. **Convert all elements to its right into 0s using simultaneous linear equations.** => 1 operation

As shown above, it takes (n+1) operations to modify the first row and column into desired 1s and 0s. Similarly, the 2nd (inner) row and column will take ((n-1)+1) operations; the 3rd ((n-2)+1) and so on until it reaches the bottom-left element which only requires a single operation.

Thus, we see that the algorithm is essentially recursive and keeps repeating the above three steps over and over again as it proceeds to inner rows and columns and until finally reaching the bottom-left element. The number of operations required for inverting an 'n' order matrix would be: **n+1+(number of operations for inner rows and columns)**. Expanding this by applying recursion we get: **n+1+((n-1)+1+((n-2)+1+(...+(2+1+(1)))))** which simplifies to give $\boldsymbol{\frac{n^2+3n-2}{2}}$ steps.

Importantly, even though the number of operations may be a high number depending on 'n', the **steps required are only 3**, which are repeated over and over again recursively.

Let us take the case of a 5x5 matrix. The outermost layer (blue and anchored by 'a') would require 6 operations; the green layer 5; yellow 4; orange 3 and element y - only 1. Hence, totally 19 steps would be required, which follows $\boldsymbol{\frac{n^2+3n-2}{2}}$.



$$\begin{bmatrix} a & b & c & d & e \\ f & g & h & i & j \\ k & l & m & n & o \\ p & q & r & s & t \\ u & v & w & x & y \end{bmatrix}$$

Since recursion is well-suited to computer coding, I hope to apply this algorithm and improve the Java program I made earlier to be able to invert matrices (invertible) of any order, with far more efficiency.

How does this algorithm compare with existing ones?:

From what we have just seen, our algorithm requires only $\frac{n^2+3n-2}{2}$ operations to invert a matrix. Gauss-Jordan Elimination (or GJ Method) - the existing method for inverting a matrix using elementary operations - proposes the use of an individual operation to convert each element into a 0 or 1, as the case may be. This means that an 'n' order matrix would need n^2 operations to invert.

Let us compare the two methods by equating the formula for maximum number of operations required:

$$\frac{n^2+3n-2}{2} \geq n^2 \quad \Rightarrow \quad (n-1)(n-2) \leq 0 \quad \Rightarrow \quad n \in [1, 2]$$

Since 'n' is an integer greater than 1, the two methods require same number of operations (ie. 4) only for n=2. For orders greater or equal to 3, the recursive algorithm developed in this project is shorter than the GJ Method. Further, due to the use of recursion, only 3 fundamental steps are needed, which just need to be applied over and over again for larger matrices.

Both the methods have the same basis, but unlike the GJ Method which requires one operation per element; we make use of simultaneous linear equations to convert multiple elements of a row into desired 0s.

*The Gauss Jordan Elimination Method is sometimes used to refer to the method of inverting a matrix using elementary operations. In that case, this project makes use of it but goes further and defines a precise algorithm to find the inverse.

~~~~~~
~~~~~~

Bibliography & Resources

1. **ISC Mathematics Class XII** by OP Malhotra, SK Gupta, Anubhuti Gangal

2. **Netbeans IDE** - software for running the java program.

3. https://www.mathsisfun.com/algebra/matrix-inverse-row-operations-gauss-jordan.html

4. https://en.wikipedia.org/wiki/Gaussian_elimination