MINI PROJECT

GAUSS ELIMINATION AND ITS APPLICATIONS ,CODED IN JAVA AND DEMONSTRATED THOUGHTFULLY

SHASHANK SINGH DATE:16TH SEPTEMBER 2024

Abstract

Gauss elimination is a method for solving a system of linear equations given by Carl Freidrich Gauss , he implemented the method basically using row reduction to such perfection such that it is called gauss elimination now.

Further the method of Row reduction was also used to derive the gauss-jordan elimination method to find the inverse of a matrix.

The Method

In Gauss Jordan elimination we take a system of linear equations , the system can be a homogeneous or nonhomogeneous system .

We take the system of equations , form a coefficient matrix using only the coefficients of the equations .

Example

A11x1 + A12x2 = b1

A21x1 + A12x3 = b2

Where the A11,A12,A21,A22 are the coefficients of the variable x1 , x2 and b1,b2 are the equivalents of the solutions.

We create a coefficient matrix of the coefficients and the equivalents .

A11 A21 | b1

A21 A22 |b2

We use elementary row operations to create an upper triangular matrix from the coefficient matrix where the elements below the right diagonal are zero .

After we have created the upper triangular matrix , the matrix we get is called the Row-Echelon form .

From the row Echelon Form we can use back substitution to get the values of the variables .

There are three basic row operations:

1.Interchange of Two rows.

2.Addition of a constant multiple of one row to another row.

3.Multiplication of one row by a nonzero constant C.

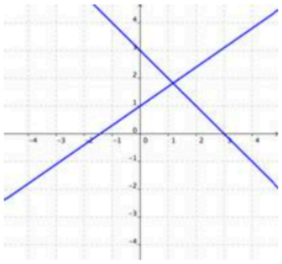
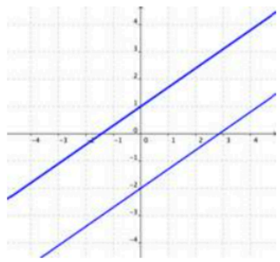
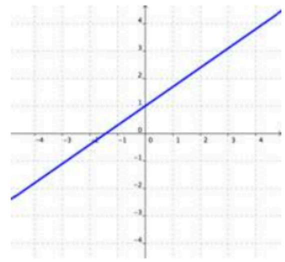
Using the above listed row operations we can make the coefficient matrix into an upper triangular matrix and get the row echelon form and after that we can simply use back substitution to get the values of the variables.

In the gauss jordan elimination , we can encounter three types of solutions i.e

1.single solution

2.No solutions

3.Infinite solutions

One Solution	No Solutions	Infinite Solutions
		
If the graphs of the equations intersect, then there is one solution that is true for both equations.	If the graphs of the equations do not intersect (for example, if they are parallel), then there are no solutions that are true for both equations.	If the graphs of the equations are the same, then there are an infinite number of solutions that are true for both equations.

SINGULAR MATRICES AND NON-SQUARE MATRICES

Gauss elimination cannot be done with singular matrices since with singular matrices the determinant of the matrices is zero and hence the upper triangular form of the matrix cannot be obtained hence the gauss method to find the values is not possible .

Secondly , gauss elimination only works with square matrices and not with the non singular ones.

JAVA CODE FOR THE GAUSS ELIMINATION METHOD

```
public class gausselimination {
    private static final double EPSILON = 1e-10;  1 usage

    // The following is a program for Gaussian elimination using the partial pivoting method
    public static double[] solution(double[][] A, double[] b) {  1 usage
        // Inputting the coefficient matrix and the column vector of solutions
        int n = b.length;
```

```
for (int p = 0; p < n - 1; p++) {
    // Finding the pivot row and swapping
    int max = p;
    for (int i = p + 1; i < n; i++) {
        if (Math.abs(A[i][p]) > Math.abs(A[max][p])) {
            max = i;
        }
    }

    // Swap rows in matrix A
    double[] temp = A[p];
    A[p] = A[max];
    A[max] = temp;

    // Swap corresponding values in vector b
    double t = b[p];
    b[p] = b[max];
    b[max] = t;

    // If the matrix is singular or near-singular
    if (Math.abs(A[p][p]) <= EPSILON) {
        throw new ArithmeticException("Matrix is partially or completely singular");
    }

    // Perform row elimination
    for (int i = p + 1; i < n; i++) {
        double alpha = A[i][p] / A[p][p];
        b[i] -= alpha * b[p];
        for (int j = p; j < n; j++) {
            A[i][j] -= alpha * A[p][j];
```

```java
                }
            }
        }

        // Code for back substitution
        double[] x = new double[n];
        for (int i = n - 1; i >= 0; i--) {
            double sum = 0.0;
            for (int j = i + 1; j < n; j++) {
                sum += A[i][j] * x[j];
            }
            x[i] = (b[i] - sum) / A[i][i];
        }

        return x;
    }

    // Sample input for the method
    public static void main(String[] args) {
        int n = 3;  // Corrected matrix size
        double[][] A = { {0, 1, 1}, {2, 4, -2}, {0,3,15 }};
        double[] b = {4,2,36};

        // Printing results
        double[] x = solution(A, b);
        for (int i = 0; i < n; i++) {
            System.out.println(x[i]);
        }
    }
}
```

OUTPUT:

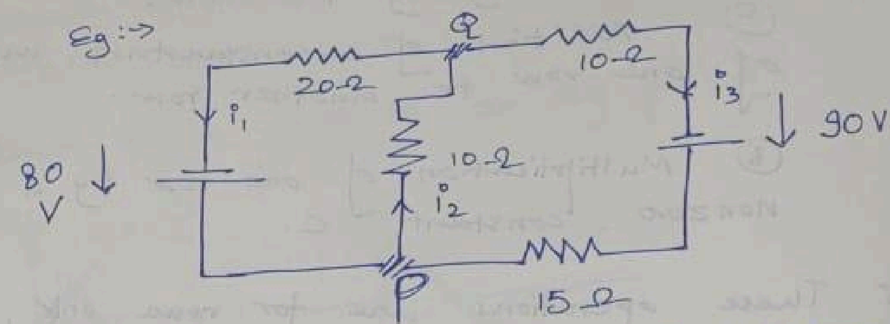
```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Jet
-1.0
2.0
2.0

Process finished with exit code 0
```

SOME APPLICATIONS OF GAUSS ELIMINATION:

1. To solve electrical networks and find values of currents.
2. To balance complex chemical equations.
3. To solve market models .

Gauss Elimination in Electrical Networks.

Eg:->



To derive the equations we need to know two laws

① Kirchoff's Current Law (KCL)

At any point in a circuit, sum of inflowing currents is equal to sum of outflowing currents.

② Kirchoff's Voltage Law (KVL)

In any closed loop, sum of all voltage drops is equivalent to applied electromotive force.

So, According to KCL and KVL

in Node P : $i_1 + i_3 = i_2$

$\Rightarrow i_1 - i_2 + i_3 = 0$

in Node Q : $i_2 = i_1 + i_3$

$\Rightarrow i_2 - i_1 - i_3 = 0$

Right loop $10 i_2 + 25 i_3 = 90$

Note :-> Ohm (Ω) is unit of electrical resistance
A (Ampere) is unit of current.
Volts (V.) is unit of electrical voltage.

Left Loop : $20 i_1 + 10 i_2 = 80$

So creating a system of equations from the above equations derived using the Laws of KCL and KVL.

$$\begin{bmatrix} i_1 & -i_2 & i_3 \\ -i_1 & +i_2 & -i_3 \\ 0 & 10 i_2 & 25 i_3 \\ 20 i_1 & 10 i_2 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 90 \\ 80 \end{bmatrix}$$

$$\Rightarrow \left[\begin{array}{ccc|c} i_1 & -i_2 & i_3 & 0 \\ -i_1 & i_2 & -i_3 & 0 \\ 0 & 10 i_2 & 25 i_3 & 90 \\ 20 i_1 & 10 i_2 & 0 & 80 \end{array}\right]$$

operation
Row 2 + Row 1
Row 4 − 20Row 1

$$\Rightarrow \left[\begin{array}{ccc|c} i_1 & -i_2 & i_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 10 i_2 & 25 i_3 & 90 \\ 0 & 30 i_2 & -20 i_3 & 80 \end{array}\right]$$

Now we can use back substitution to find the values of i_1 i_2 i_3 easily.