GAUSSIAN CALCULATOR

Midterm Exam

DESCRIPTION OF THE PROBLEM

The example program attempts to handle the computational difficulty of solving systems of linear equations using the Gaussian elimination approach. It is intended to convert a system of equations into an augmented matrix format, conduct row operations methodically to produce an upper triangular form, and then back substitute to get the values of the unknowns. To improve numerical stability, the method begins with selecting the row with the greatest pivot element, then deleting the lower rows. Finally, the solve function accepts equations via a web interface, calculates the solution with the GaussianElimination function, and dynamically displays the result on the webpage.

ALGORITHM DESIGN

The Gaussian elimination algorithm is based on the transform and conquer strategy, where the original problem (solving a system of linear equations) is transformed into a simpler version of the same problem (an upper triangular matrix) that can be easily solved.

ALGORITHM DESIGN

1. *Forward Elimination:*
   - *Find the maximum element in each column (pivot) to avoid numerical instability.*
   - *Swap the maximum element's row with the current row.*
   - *For each row below the current one, subtract a multiple of the current row to make the elements below the current pivot element zero. This creates an upper triangular matrix.*
2. *Back Substitution:*
   - *Starting from the last row, calculate the value of each variable.*
   - *Substitute the calculated values into the rows above to find the values of the remaining variables.*

CODE IMPLEMENTATION

```
function gaussianElimination(matrix) {
    let n = matrix.length;

    for (let i = 0; i < n; i++) {
        let maxEl = Math.abs(matrix[i][i]),
            maxRow = i;
        for (let k = i + 1; k < n; k++) {
            if (Math.abs(matrix[k][i]) > maxEl) {
                maxEl = Math.abs(matrix[k][i]);
                maxRow = k;
            }
        }

        for (let k = i; k < n + 1; k++) {
            let tmp = matrix[maxRow][k];
            matrix[maxRow][k] = matrix[i][k];
            matrix[i][k] = tmp;
        }
        for (let k = i + 1; k < n; k++) {
            let c = -matrix[k][i] / matrix[i][i];
            for (let j = i; j < n + 1; j++) {
                if (i === j) {
                    matrix[k][j] = 0;
                } else {
                    matrix[k][j] += c * matrix[i][j];
                }
            }
        }
    }

    let x = new Array(n).fill(0);
    for (let i = n - 1; i > -1; i--) {
        x[i] = matrix[i][n] / matrix[i][i];
        for (let k = i - 1; k > -1; k--) {
            matrix[k][n] -= matrix[k][i] * x[i];
        }
    }

    return x;
}
```

CODE IMPLEMENTATION

```
function solve() {
    let matrix = [];
    for (let i = 1; i <= 3; i++) {
        let row = [];
        for (let j = 1; j <= 3; j++) {
            row.push(parseFloat(document.getElementById(`a${i}${j}`).value));
        }
        row.push(parseFloat(document.getElementById(`b${i}`).value));
        matrix.push(row);
    }

    let solution = gaussianElimination(matrix);
    document.getElementById("solution").innerHTML = `<p>Solution: ${solution.map(x => isNaN(x)
        ? 'No solution' : x.toFixed(2)).join(', ')}</p>`;
}
```

EXPLANATION OF CODE

Forward Elimination Process:

- *Iterate through each column of the matrix. For the current column i, find the row k below it (including itself) that has the maximum absolute value in column i. This is to reduce numerical errors and improve stability.*
- *Swap the current row i with the row k found above to ensure the pivot element has the largest absolute value.*
- *For each row k below the current row i, calculate a factor c that when multiplied with the current row and subtracted from row k will eliminate the element below the pivot, making the matrix closer to upper triangular form.*

EXPLANATION OF CODE

Back Substitution Process:

- *After forward elimination, the matrix is in upper triangular form. Starting from the bottom row, calculate the value of each variable by dividing the rightmost element of the row by the pivot element (diagonal element of the row).*
- *For each row above, update the rightmost element to eliminate the already solved variables, effectively solving for one variable at a time in reverse order.*

The solve function collects input from a web page, constructs the augmented matrix, calls gaussianElimination to solve it, and then updates the webpage with the solution.

CODE OUTPUT

Gaussian Elimination Calculator

Enter your equations:

1	X_1 +	5	X_2 +	3	X_3 =	52
2	X_1 +	4	X_2 +	1	X_3 =	24
2	X_1 +	4	X_2 +	12	X_3 =	36

Solve

Gaussian Elimination Calculator

Enter your equations:

1	X_1 +	5	X_2 +	3	X_3 =	52
2	X_1 +	4	X_2 +	1	X_3 =	24
2	X_1 +	4	X_2 +	12	X_3 =	36

Solve

Solution: -13.39, 12.42, 1.09

TIME COMPLEXITY

- *Forward Elimination: The outer loop runs n times, where n is the number of variables (rows) in the matrix. For each iteration, the search for the maximum element and the row subtraction operation both have a time complexity of O(n^2), as they involve going through the elements of the matrix rows. Therefore, the forward elimination has a time complexity of O(n^3).*
- *Back Substitution: This process has a time complexity of O(n^2), as it involves iterating through each row (n iterations) and updating the solution vector and the right-hand side of the matrix, which involves going through up to n elements in each iteration.*
- *Overall: The total time complexity of the Gaussian elimination algorithm is O(n^3) due to the forward elimination phase being the most computationally intensive part.*

LEARNING REFLECTION

During the midterm period, I worked actively with numerous algorithms, with a particular focus on the transform and conquer technique, which was both difficult and useful. Understanding the Gaussian elimination approach for solving systems of linear equations required first translating the problem into a more understandable form. This approach, as demonstrated in the accompanying code, stood out because of its practical use and complexity. The process of determining the maximum pivot element for numerical stability and conducting row operations to reduce the matrix to an upper triangular form was originally difficult. However, by breaking down the steps, practicing comparable tasks, and grasping the underlying mathematical ideas, I gradually overcame these obstacles. A key takeaway was the importance of problem transformation in algorithm design, which not only simplifies complex problems but also significantly improves computational efficiency. This approach, as demonstrated through Gaussian elimination, highlights the power of algorithmic strategies in solving real-world problems effectively.