C++ Simplex Method - First Iteration

 # Step-by-Step Explanation

1. Header Files and Namespace

```cpp
#include <iostream>
#include <vector>
#include <iomanip>
#include <limits>
using namespace std;
```

- `<iostream>`: Input/output operations.
- `<vector>`: Dynamic array for storing the tableau.
- `<iomanip>`: Formatting output with decimal precision.
- `<limits>`: Provides numeric limits (like infinity()).

2. Printing the Tableau

```cpp
void printTableau(const vector<vector<double>> &tableau) {
    for (const auto &row : tableau) {
        for (double value : row) {
            cout << setw(8) << fixed << setprecision(2) << value << " ";
        }
        cout << "\n";
    }
}
```

Formats and prints the tableau.

3. Finding the Pivot Column

```cpp
int findPivotColumn(const vector<vector<double>> &tableau) {
    int pivotCol = -1;
    double minValue = 0.0;
    for (int j = 0; j < tableau[0].size() - 1; ++j) {
        if (tableau.back()[j] < minValue) {
            minValue = tableau.back()[j];
            pivotCol = j;
        }
    }
    return pivotCol;
}
```

```
```

Finds the most negative coefficient in the objective function row.

4. Finding the Pivot Row (Handling Degeneracy)

```cpp
int findPivotRow(const vector<vector<double>> &tableau, int pivotCol) {
    int pivotRow = -1;
    double minRatio = numeric_limits<double>::infinity();

    for (int i = 0; i < tableau.size() - 1; ++i) {
        double value = tableau[i][pivotCol];
        if (value > 0) {
            double ratio = tableau[i].back() / value;
            if (ratio < minRatio || (ratio == minRatio && pivotRow == -1)) {
                minRatio = ratio;
                pivotRow = i;
            }
        }
    }
    return pivotRow;
}
```

Ensures degeneracy handling by picking the first valid row in case of ties.

5. Performing the Pivot Operation

```cpp
void pivot(vector<vector<double>> &tableau, int pivotRow, int pivotCol) {
    double pivotValue = tableau[pivotRow][pivotCol];
    for (double &value : tableau[pivotRow]) {
        value /= pivotValue;
    }
    for (int i = 0; i < tableau.size(); ++i) {
        if (i != pivotRow) {
            double factor = tableau[i][pivotCol];
            for (int j = 0; j < tableau[0].size(); ++j) {
                tableau[i][j] -= factor * tableau[pivotRow][j];
            }
        }
    }
}
```

Converts the pivot element to 1 and clears the pivot column.

6. Defining the Tableau

```cpp
```

```
vector<vector<double>> tableau = {
    {2, 3, 1, 0, 0, 100},
    {4, 1, 0, 1, 0, 80},
    {3, 2, 0, 0, 1, 60},
    {-3, -5, 0, 0, 0, 0}
};
```

The last row is the objective function, and the last column is RHS.

7. Main Execution Loop

```cpp
int main() {
    vector<vector<double>> tableau = {
        {2, 3, 1, 0, 0, 100},
        {4, 1, 0, 1, 0, 80},
        {3, 2, 0, 0, 1, 60},
        {-3, -5, 0, 0, 0, 0}
    };

    cout << "Initial Tableau:\n";
    printTableau(tableau);

    while (true) {
        int pivotCol = findPivotColumn(tableau);
        if (pivotCol == -1) {
            cout << "Optimal solution found.\n";
            break;
        }

        int pivotRow = findPivotRow(tableau, pivotCol);
        if (pivotRow == -1) {
            cout << "Problem is unbounded.\n";
            break;
        }

        pivot(tableau, pivotRow, pivotCol);
        cout << "Updated Tableau:\n";
        printTableau(tableau);
    }
}
```

Runs multiple iterations until the optimal solution is found.

Final Notes:

- The program iterates through Simplex steps until an optimal solution is reached.
- Handles degeneracy to prevent cycling.
- Provides formatted output for clarity.