**Report on Simulation using FUG Method**

**1. Introduction**

This report discusses the implementation of a **Fractionation Tower** simulation, a critical unit operation in chemical engineering used to separate mixtures based on the volatility of components. The simulation employs the **Fenske-Underwood-Gilliland (FUG) method**, a shortcut approach used in distillation column design. This method estimates the **minimum number of theoretical stages, minimum reflux ratio, and actual operating stages** required for efficient separation.

**2. Chemical Engineering Concepts**

**2.1 Fractionation Tower (Distillation Column)**

A fractionation tower separates a mixture based on **differences in boiling points (volatility)**. It consists of the following key components:

* **Feed**: The mixture to be separated, introduced at an intermediate stage.
* **Distillate (Overhead)**: The lighter components collected at the top of the column.
* **Bottoms (Residue)**: The heavier components collected at the bottom of the column.

**2.2 FUG Method**

The **FUG method** is a widely used shortcut model for estimating key design parameters:

1. **Fenske Equation** – Determines the **minimum number of theoretical stages** needed for separation under total reflux conditions.
2. **Underwood Equation** – Computes the **minimum reflux ratio**, representing the minimum energy required for separation.
3. **Gilliland Correlation** – Estimates the **actual number of stages** under practical operating conditions based on the reflux ratio.

**3. C++ Code Implementation**

**3.1 Class: FractionationTower**

The **FractionationTower** class models a distillation column with key attributes and methods for analysis.

**3.2 Private Data Members**

1. **feed\_composition**: A map<string, double> representing the composition of the feed mixture.
2. **distillate**: A map<string, double> storing the composition of the overhead product.
3. **bottoms**: A map<string, double> representing the composition of the bottom product.
4. **component\_volatility**: A map<string, double> storing the relative volatilities of components.
5. **rel\_volatility**: A double value representing the relative volatility between key components (e.g., iC5 and nC5), essential for the Fenske equation.

**3.3 Private Methods**

1. **sum\_components()** – Computes the total flow rate of a given mixture.
2. **calculate\_key\_volatility()** – Determines the relative volatility of key components, essential for separation calculations.
3. **solve\_underwood\_equation()** – Solves for the Underwood root (θ), used to compute the minimum reflux ratio. The function iterates using a **bisection method** to find the value of θ that satisfies the Underwood equation. This iterative approach ensures convergence to an accurate solution by narrowing down the possible values of θ within a defined tolerance range.

**3.4 Public Methods**

1. **compute\_fenske\_stages()** – Calculates the minimum number of theoretical stages required for separation.
2. **analyze\_secondary\_components()** – Evaluates how non-key components are distributed in distillate and bottoms.
3. **calculate\_underwood\_reflux()** – Computes the minimum reflux ratio required for efficient separation.
4. **estimate\_operating\_stages()** – Uses the Gilliland correlation to determine the actual number of stages in practical operation. The **Gilliland correlation graph** is utilized in this function to estimate the actual number of trays based on the **minimum reflux ratio and operating reflux ratio**. The function uses an empirical equation derived from the Gilliland chart to approximate the **fraction of the difference between actual and minimum stages**.
5. **perform\_analysis()** – Executes all calculations and outputs the simulation results.

**3.5 Main Function**

The main() function:

* Initializes feed, distillate, bottoms, and component volatility data.
* Creates an instance of FractionationTower.
* Calls perform\_analysis() to execute the FUG method and display results.

**4. Significance of Each Function**

**4.1 Private Methods**

* **sum\_components()** – Ensures mass balance and normalizes compositions.
* **calculate\_key\_volatility()** – Determines how easily components can be separated.
* **solve\_underwood\_equation()** – Essential for computing the **minimum reflux ratio** by iterating to find a θ value that satisfies the Underwood equation.

**4.2 Public Methods**

* **compute\_fenske\_stages()** – Establishes the baseline for column design by estimating the **minimum number of stages**.
* **analyze\_secondary\_components()** – Helps optimize separation efficiency by understanding non-key component behavior.
* **calculate\_underwood\_reflux()** – Determines the minimum energy input required for effective separation.
* **estimate\_operating\_stages()** – Uses the Gilliland correlation to approximate the actual number of trays based on the reflux ratio.
* **perform\_analysis()** – Integrates all computations into a single workflow.

**5. Example Output**

Given the input:

* **Feed Composition:** iC4, nC4, iC5, nC5, C6, C7, C8, C9
* **Distillate (Overhead):** iC4, nC4, iC5, nC5
* **Bottoms (Residue):** nC4, iC5, nC5, C6, C7, C8, C9
* **Component Volatility:** iC4 (2.5), nC4 (2.1), iC5 (1.0), nC5 (0.83), C6 (0.5), C7 (0.20), C8 (0.10), C9 (0.08)

The program outputs:

1. **Relative volatility between key components**.
2. **Minimum theoretical stages** (Fenske equation).
3. **Distribution of secondary components**.
4. **Minimum reflux ratio** (Underwood equation).
5. **Estimated actual stages** (Gilliland correlation).

**6. Applications**

This simulation is useful for:

* **Preliminary distillation column design** before rigorous simulations.
* **Optimizing operating conditions** to minimize energy consumption.
* **Understanding component distribution** within the column.

**7. Conclusion**

The C++ implementation of the **Fractionation Tower** using the **FUG method** effectively simulates distillation column design. By leveraging shortcut equations, it provides quick estimates for **minimum stages, reflux ratio, and actual operating stages**, which are crucial for industrial applications. The developed program serves as a valuable tool for distillation analysis and optimization.