**Report: Solving a System of Linear Equations Using NumPy**

**1. Objective**

The goal of this program is to solve a system of linear equations using NumPy's linalg.solve() function. The user provides the coefficients of the equations and the constants (right-hand side values), and the program computes the solution, if one exists.

**2. Program Workflow**

The program is structured into three main parts:

1. **Input**:
   * Accept the number of variables in the system.
   * Accept the coefficients of each equation and the constants (right-hand side values).
2. **Processing**:
   * Represent the system of equations as matrices: (coefficients) and (constants).
   * Solve the system using NumPy’s np.linalg.solve() method.
3. **Output**:
   * Display the solution if it exists.
   * Handle errors such as invalid input or systems with no unique solution.

**3. Program Components**

**\*\*Function 1: \*\*creating\_matrix(n)**

* **Purpose**: To take input for the coefficient matrix and constants .
* **Input**:
  + n: Number of variables/equations.
* **Processing**:
  + Collect rows of coefficients.
  + Collect constants (right-hand side values).
* **Output**:
  + Returns two NumPy arrays: (coefficient matrix) and (constants vector).

**Key Features**:

* Ensures the user inputs exactly coefficients for each equation.
* Validates that constants are numerical values.

**\*\*Function 2: \*\*solver(A, B)**

* **Purpose**: To solve the system of equations using the matrices and .
* **Input**:
  + : Coefficient matrix.
  + : Constants vector.
* **Processing**:
  + Uses np.linalg.solve(A, B) to solve .
* **Output**:
  + Displays the solution for each variable if a unique solution exists.
  + Handles cases where no unique solution is possible.

**Main Program**

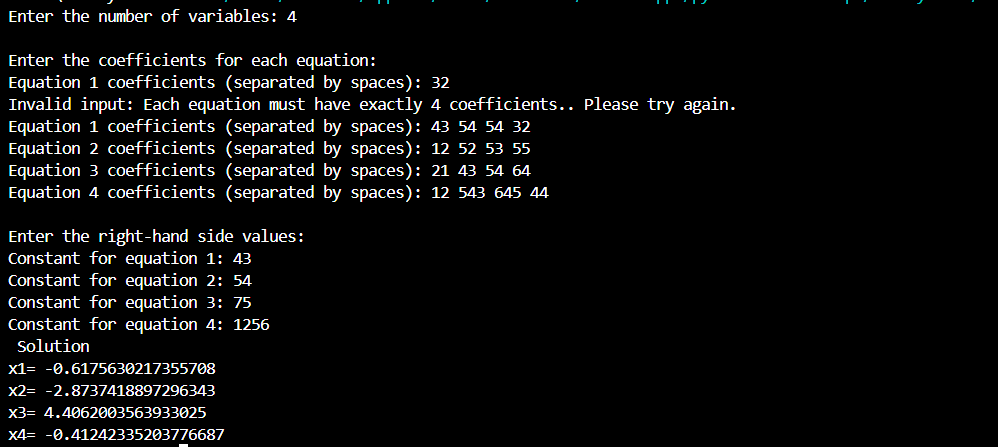
* Prompts the user for the number of variables.
* Calls creating\_matrix() to collect inputs.
* Calls solver() to solve the system and display the solution.
* Handles any exceptions, such as invalid input or singular matrices.

**4. Code**

Here is the full program:



* + 1. **Input/Output Example**

****

**6. Error Handling**

The program includes robust error handling:

1. **Input Validation**:
   * Ensures the user enters valid numbers for coefficients and constants.
   * Checks that the number of coefficients matches the number of variables.
2. **System Solvability**:
   * If the matrix is singular or non-square, an appropriate message is displayed:  
     *"The system has no unique solution."*
3. **Invalid Variable Count**:
   * Prevents the program from proceeding with non-positive numbers for variables.

**7. Applications**

This program can be used in various applications:

* Solving systems of equations in physics, chemistry, and engineering.
* Performing linear algebra calculations in mathematics and data science.
* Automating tasks that involve solving small to medium-sized systems of equations.

**8. Conclusion**

This program successfully solves a system of linear equations using NumPy. It is flexible, robust, and user-friendly, accommodating systems with any number of variables. Proper error handling ensures it can handle invalid inputs and singular systems gracefully.