Operations Research - Assignment 1 Report

# Alexandria University

**Faculty of Engineering**

**Computer and Systems Department**

# Team members

عبدالرحمن سعيد رمضان جمعه Name: •

* ID: 22010879

ايمن ابراهيم محمد قطب :Name •

* ID: 22010656

فاروق اشرف فاروق Name: •

* ID: 22011012

# 1. Introduction

This report presents the implementation of a Linear Programming (LP) solver using the Simplex Method and its variations. The solver is designed to handle different types of LP problems efficiently and provides a stepby-step solution process.

# 2. Code Flow and Architecture Report

## **2.1. Main Application Flow (main.py)**

The GUI application (main.py) serves as the entry point, built using PyQt5. It provides a user interface for defining LP problems and selecting solution methods.

**Key components**:

**Tabs**: Problem Definition, Solution, Iteration Steps.

**Input Handling**: Collects variables, constraints, objective type (max/min), and method selection. **Solver Dispatch**: Calls appropriate solver based on user selection (Simplex, Big-M, Two-Phase, Goal Programming).

**Data Structures**:

## **2.2 Solver Methods**

The solver was implemented using **Python**.

**2.2.1 Standard Simplex Method (Simplex.py)**

**Purpose**: Solve LP problems with <= constraints and non-negative variables.

**Key Functions**:

simplex\_method(c, A, b, isMax):

Initializes a tableau with slack variables.

Iteratively selects entering/leaving variables using pivot operations.

Returns the optimal solution and iteration steps.

**Steps**:

1. Convert inequalities to equations using slack variables.
2. Iterate via pivoting to maximize/minimize the objective.

### 2.2.2 Big-M Method (Big\_M.py)

**Purpose**: Handle problems with >= or = constraints using artificial variables and a penalty term (M).

**Key Functions**:

big\_m\_method(c, A, b, constraint\_types, isMax, variable\_types):

Splits unrestricted variables into x+ and x-.

Adds artificial variables with large penalty coefficients (M).

Uses simplex iterations to drive artificial variables out of the basis.

**Steps**:

1. Add artificial variables with penalty coefficient M.
2. Solve using simplex, penalizing solutions where artificial variables are non-zero.

### 2.2.3 Two-Phase Method (Two\_phase.py)

**Purpose**: Another way to Handle problems with >= or = constraints using two phases. phase 1 : To minimize the effect of artificial variables and find a feasible solution .

phase 2 : Use the feasible basis to optimize the original objective. **Key Functions**:

**Phase 1**: \_\_execute\_phase1() minimizes the sum of artificial variables.

**Phase 2**: \_\_execute\_phase2() optimizes the original objective after removing artificial variables.

make\_vars\_zeros\_Linearly(): Adjusts the objective row to eliminate basic variables coefficients from it.

### 2.2.4 Goal Programming (Goal\_Programming.py)

**Purpose**: Achieve multiple prioritized goals by minimizing deviations.

**Key Functions**:

goal\_method():

Adds deviation variables (d+, d-) for each goal.

Modifies the tableau to prioritize goals.

Checks goal satisfaction via isDone list.

**Steps**:

1. Convert goals into constraints with deviation variables.
2. construct the objective function by summation of multiplyed priorities by deviation variables responsible for its Goal
3. Optimize goals sequentially based on priority using lexicographic simplex.

# 3. Key Data Structures

**Numpy Arrays**: For constraint matrices (A), objective coefficients (c), RHS values (b) and 2D numpy array representing coefficients, slack/artificial variables, and RHS (tableau).

**Lists**: Track variable types (unrestricted/non-negative), constraint types (<=, =, >=), main\_row: Column headers (e.g., x1, s1, a1) , basic\_var: Current basic variables in the solution and goal priorities.

**Dictionaries**: Store solutions and iterations for display.

# 4. Function Interactions

1. **GUI Inputs** → Converted to matrices (A, b, c) and constraint lists.
2. **Solver Selection** → Dispatches to simplex\_method(), big\_m\_method(), etc.
3. **Tableau Initialization** → Built based on variable types and constraints.
4. **Pivoting** → \_\_execute\_simplex() performs iterations across all methods.
5. **Solution Extraction** → Post-processing (e.g., merging x+/x- for unrestricted variables).

# 5. Sample Runs

Below are example cases solved using our program:

## 5.1 Example 1 - Simplex Method

Maximize Z = 5x1 - 4x2 + 6x3 -8x4

Subject to:

x1 + 2x2 +2x3 +4x4 ≤ 40

2x1 – x2 +x3 +2x4 ≤ 8

4x1 – 2x2 +x3 -x4 ≤ 10

4x1 – 2x2 +x3 - x4 ≤ 8

x1, x2, x3, x4 ≥ 0

**Output:**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Optimal Solution: x1 = 0, x2 = 6, x3 = 0, x4 = 7
* Optimal Objective Value: Z = -80

## 5.2 Example 2 - Big-M Method

Maximize Z = x1 + 2x2 +x3

Subject to:

x1 + x2 +x3 = 7

2x1 - 5x2 +x3 ≥ 10

x1, x2 , x3 ≥ 0

A screenshot of a program

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

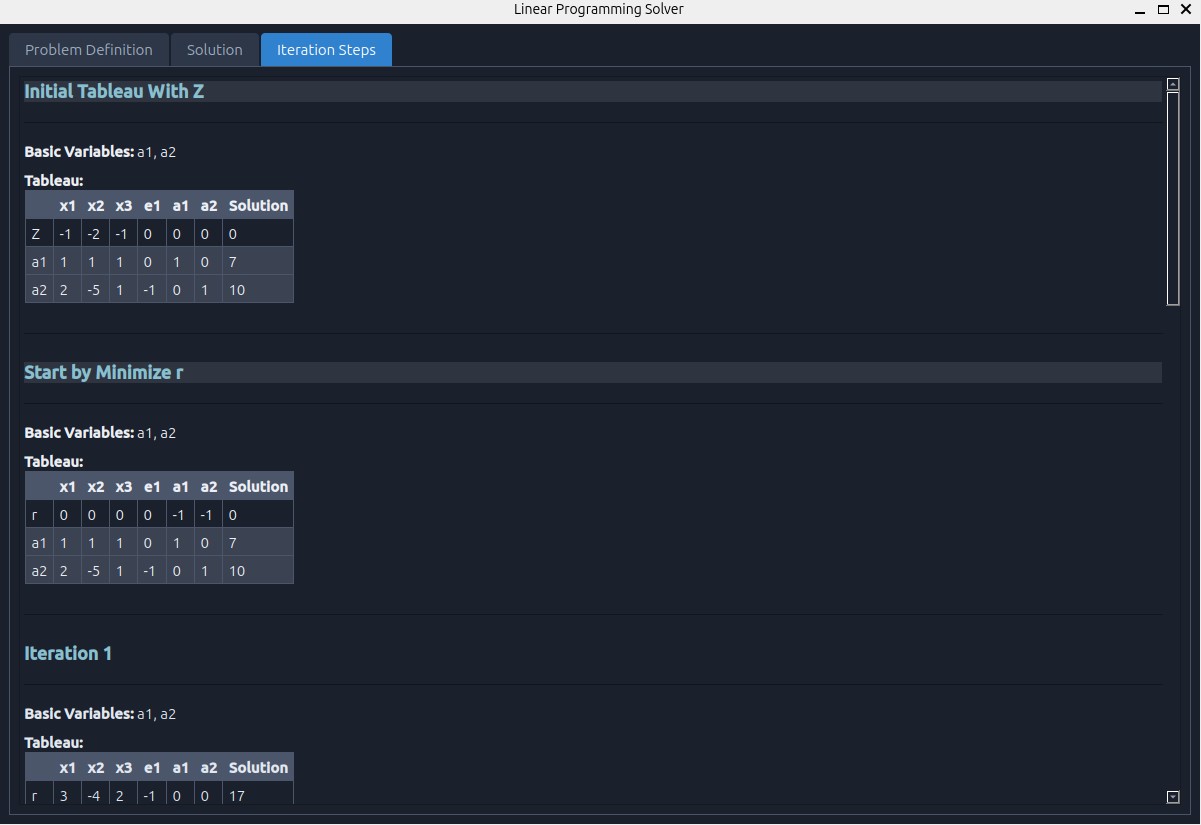
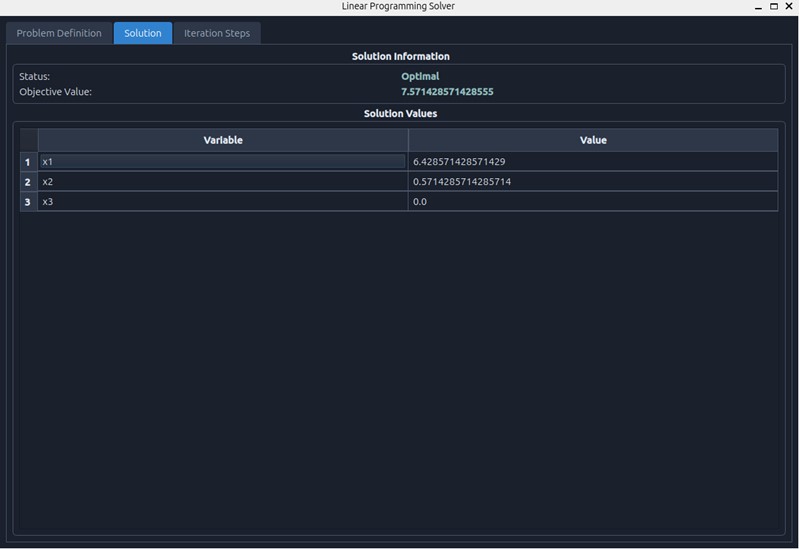
Description automatically generatedA screenshot of a program

Description automatically generated

**Output:**

* Optimal Solution: x1 = 45/7, x2 = 4/7
* Optimal Objective Value: Z = 53/7

## Example 3 - Two-Phase Method



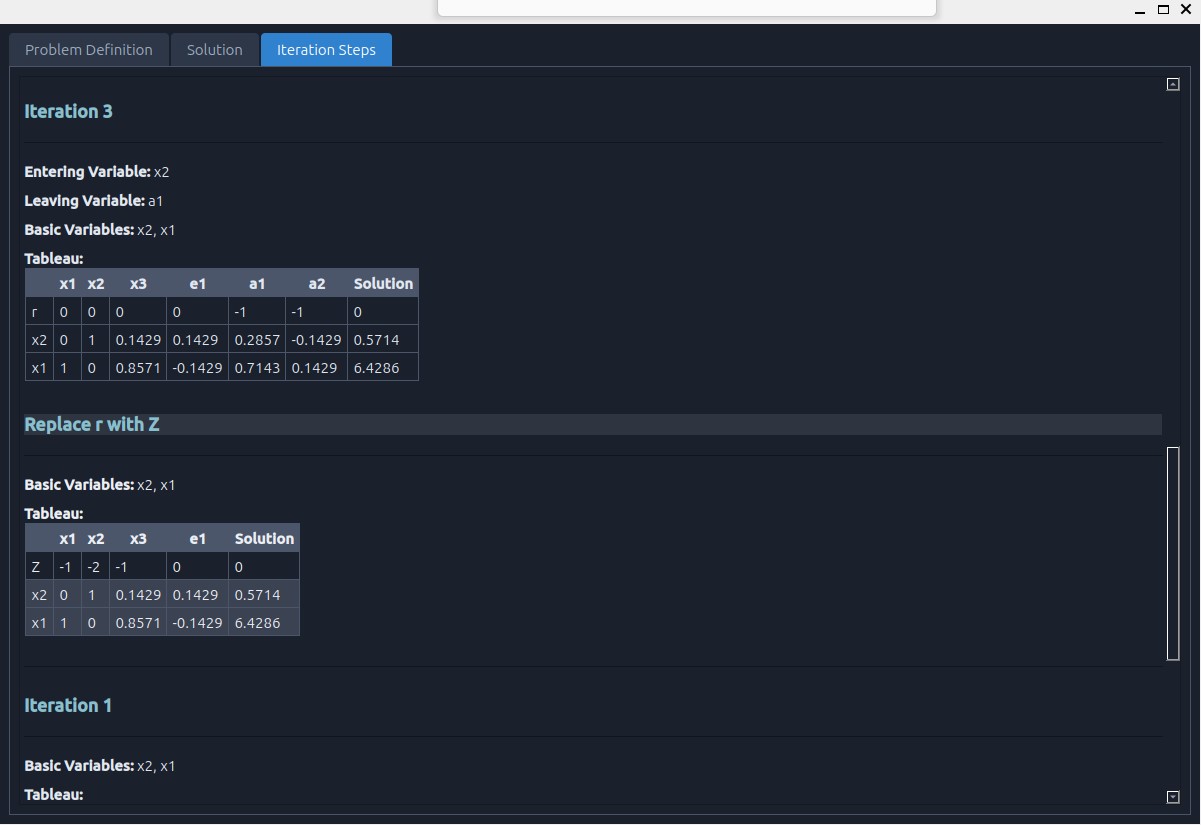
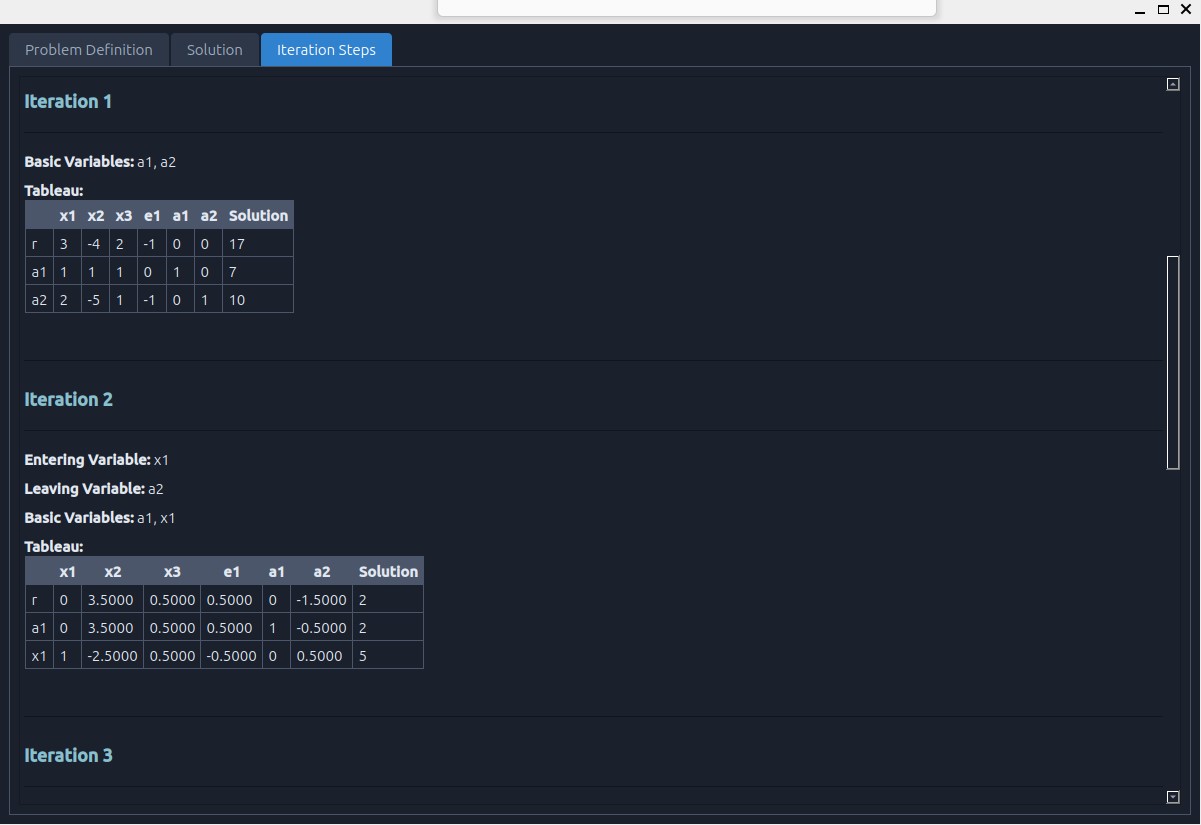
Maximize Z = x1 +2x2+ x3

Subject to:

x1 + x2 +x3 = 7

2x1 - 5x2 +x3 ≥ 10

x1, x2 , x3 ≥ 0



A screenshot of a computer

Description automatically generated

**Output:**

* Optimal Solution: x1 = 45/7, x2 = 4/7
* Optimal Objective Value: Z = 53/7

### 5.4 Example 4 - Goal-programming Method

Goals:   
200x1>=1000 priority:1

100x1+ 400x2 >=1200 priority:2

250x2 >=800 priority:1

Constraints:

1500x1 +3000x2 ≤ 15000

A screenshot of a computer

Description automatically generatedA screenshot of a video editing program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Output:**

Optimal Solution: x1 = 5, x2 = 2.5

Satisfying Goals : G1 and G2

### 5.5 Example 5 – Unrestricted variables

Constraints:   
x1+x2 >=7

2x1+ x2 <=4

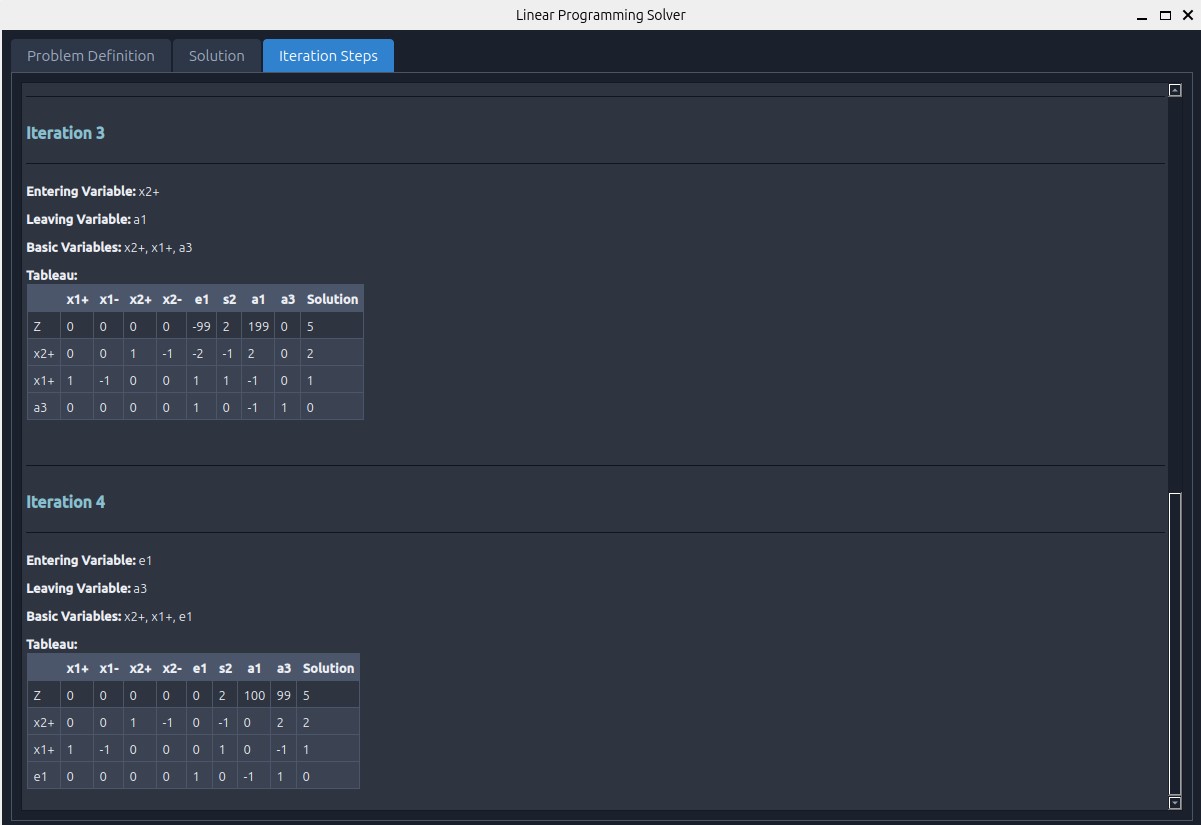
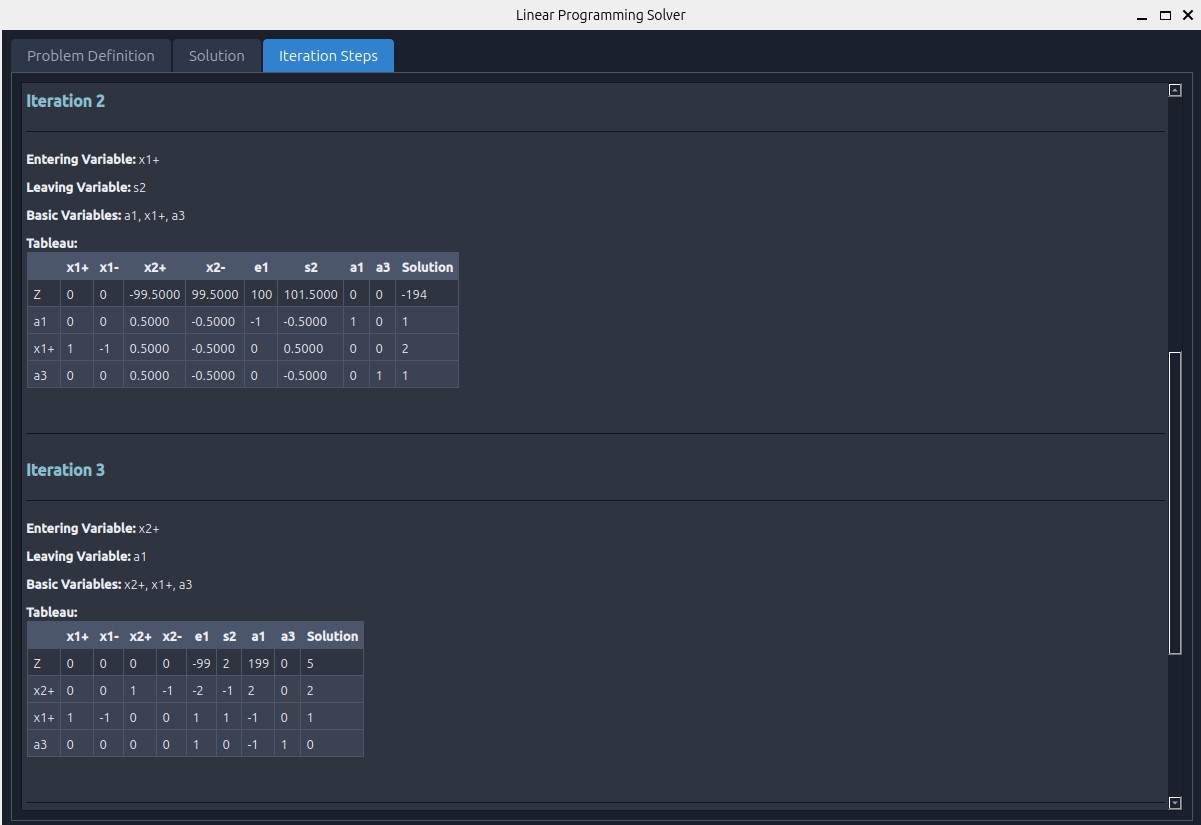
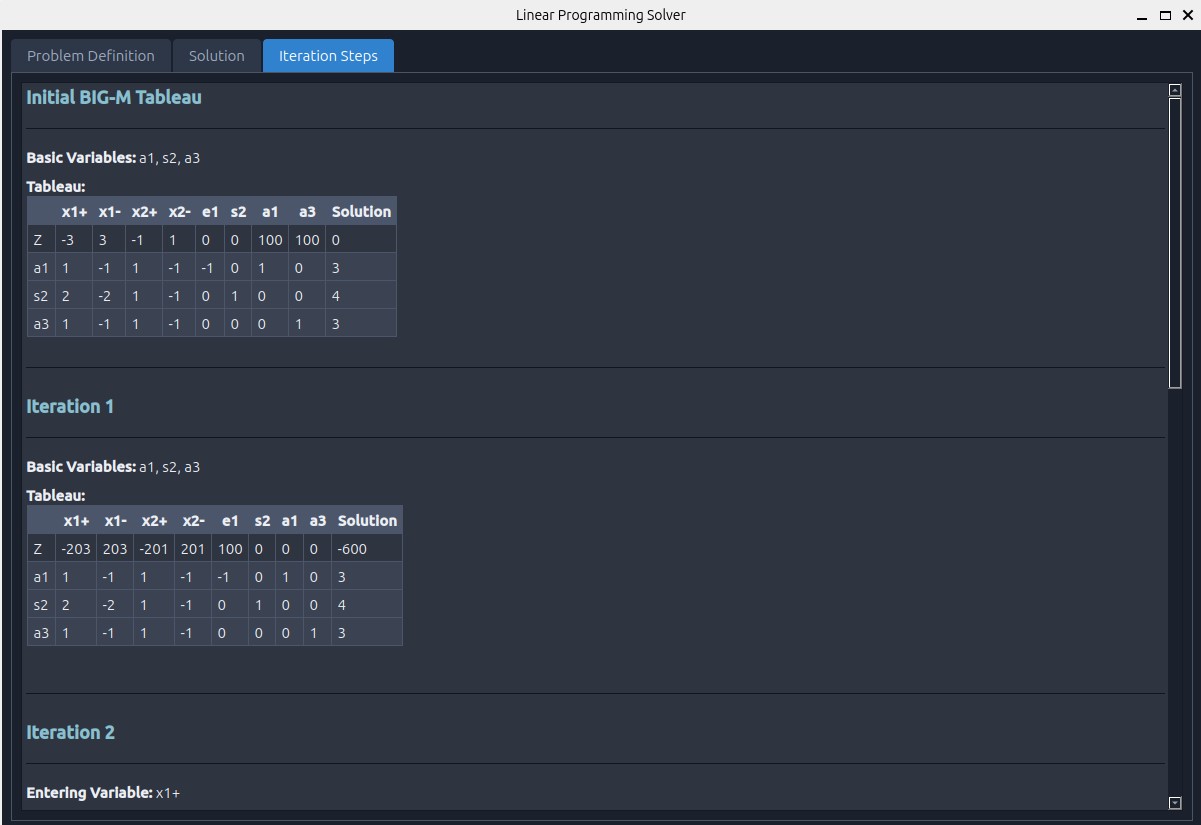
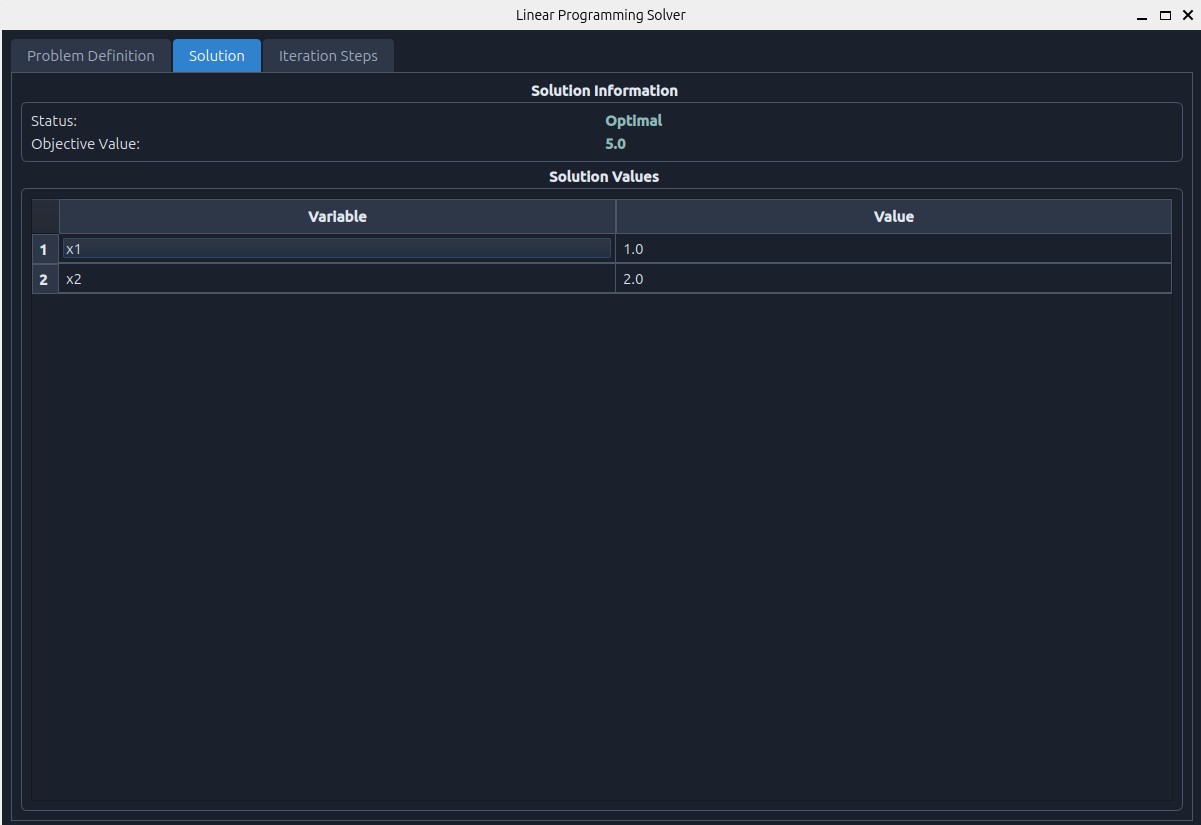
x1 + x2=800

Maximize Z =3 x1 +x2

x1 & x2 are unrestricted variables.

A screenshot of a computer program

Description automatically generated



### 5.6 Example 6 – Unbounded solution

Constraints:   
x1+x2 -x3 <=5

6x1+ 5x2 – x4 <=10

Maximize Z =36 x1 +30x2 -3 x3 -4x4

x1 , x2 , x3 , x4 >=0

A screenshot of a computer

Description automatically generated

# A screenshot of a computer Description automatically generated

A screenshot of a computer

Description automatically generated

### 5.7 Example 7 – infeasible solution

Constraints:   
x1+x2 >=5

x1+ x2 <=5

Maximize Z =36 x1 +30x2

x1 , x2 >=0

A screenshot of a program

Description automatically generated

# 6. Bonus Feature

We developed a **user-friendly interface** by pyQt5 in python that allows users to input LP problems easily and view the solution process interactively.

# 7. Conclusion

This project provided hands-on experience in solving LP problems using different methods. The solver successfully handles various constraints and outputs detailed step-by-step solutions. Future improvements include extending support for additional optimization techniques and graphical visualization of results.