AUTOMATIC CONTROL
Signal Flow Graph Solver

Abeer Ahmad “40" | Salma Ahmed “32”

Problem Statement

Given:

Signal flow graph representation of the system. Assume that total number of nodes and numeric branches gains are given.

Required:

1. Graphical interface.
2. Draw the signal flow graph showing nodes, branches, gains, ...
3. Listing all forward paths, individual loops, all combination of n non-touching loops.
4. The values of Δ, Δ1, Δ2... Δm, where m is number of forward paths.
5. Overall system transfer function.

Main Features of The Program

- Given total number of nodes in the graph, with all connecting branches and gains, this program can fully analyse the system by representing all given information as a “Signal-Flow-Graph”.
- Complete analysis of the system includes the following procedures:
    1. Detecting all forward paths, with the corresponding gains.
    2. Detecting all loops, with the corresponding gains.
    3. Detecting and Grouping all non-touching loops, i.e; (single loops, each 2 non-touching loops, each 3 non-touching loops, etc…).
    4. Detecting and Grouping all non-touching loops with all forward paths.
    5. Calculating Δ.
    6. Calculating Δi, where i = 1, 2, …m, as mentioned above.
    7. Calculating the overall system transfer function.

User Guide

- First, you need to enter total number of nodes in the SFG.
- To add any node to your SFG, enter its name in the box labeled “Add New Node”, then click the “Add” button.
- To add any edge (branch) to your SFG, you need to enter 3 information:
    1. The node where it comes from, in the box labeled “Add Edge From”.
    2. The node it goes to, in the box labeled “To”.
    3. The gain of that branch, in the box labeled “Gain”.

  Then click the “Add” button.

- Determine both source and sink nodes in the boxes labeled “Source”, “Sink” correspondingly.
- To solve the SFG and analyse the system, click the “Solve” button.

Data Structures

1. **Array[]** : to store the nodes of the graph, and all data of known size used later in the algorithm.
2. **ArrayList<>** : to store data of unknown size, such as:
   - Forward Paths.
   - Forward Gains.
   - Loops.
   - Loops Gains.
   - Non-Touching Loops.
   - Deltas.
3. **HashMap<>** : to map each node ID in the graph to a fixed integer, representing its index in the array of nodes; for a constant time access.

Algorithms Used

1. **Depth First Search (DFS)** : to detect all forward paths and loops.
2. **Complete Search (Combinations)** : to get all different combinations of non-touching loops.

Main Modules

The project is divided into 3 main packages:

1. **sfg :** which consists of the following classes:
   a. **Node** : representing the node in signal flow graph and holding 2 pieces of information the name of the node and an arraylist of edges connected to that node.
   b. **Edge :** representing the edge in the signal flow graph and holding 2 pieces of information the edge gain and the node to which the edge is connected.
   c. **SFG :** which is responsible for applying DFS algorithm on the graph in order to get all of the forward paths and loops and storing them in arraylists to be sent to the MasonSolver.
   d. **MasonSolver :** which is responsible for all the calculations to be done including calculating different combinations of non touching loops , calculating the delta based on non touching loops to a certain forward path , calculating the total delta and the overall transfer function.
2. **gui :** which consists of mainly following classes :
   a. **UI :** which initializes the user interface with all its components and calls methods to perform different actions whenever different buttons are pressed.
   b. **GraphManager :** which is responsible for managing how the graph is to be drawn according to the no of nodes , the edges and the gains given using the graph stream library.It is also responsible for styling the drawn graph.
3. **tests :** which has 2 different tests used for testing the functionality of the signal flow graph solver independent from the UI.

Sample Runs

1. **An SFG of 7 nodes with the following inputs:**

- **Nodes:** y1, y2, y3, y4, y5, y6, y7.
- **Edges:** y1 → y2 with gain = 1.

  y2 → y3 with gain = 5.

  y2 → y6 with gain = 10.

  y3 → y4 with gain = 10.

  y4 → y3 with gain = -1.

  y4 → y5 with gain = 2.

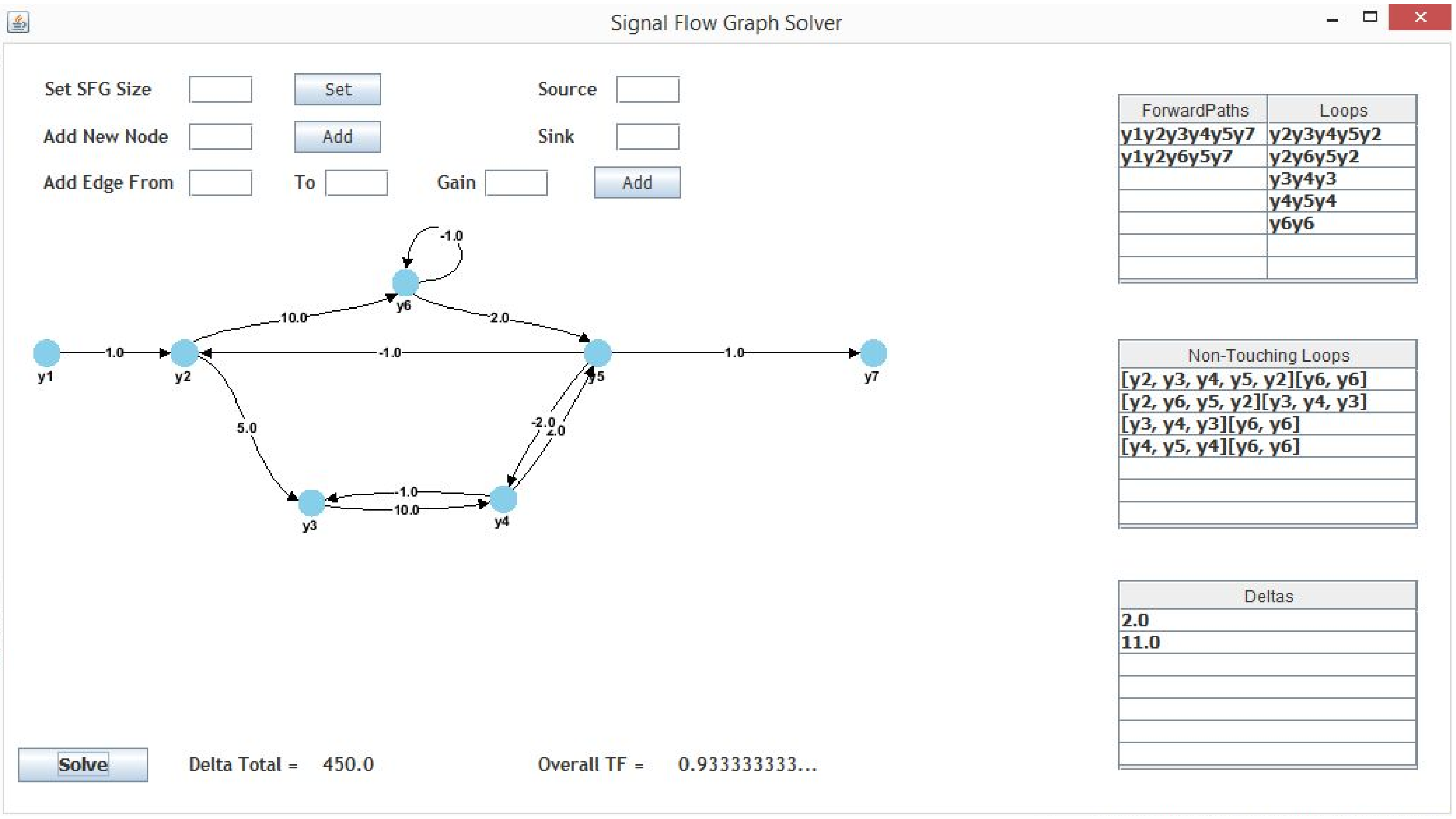
  y5 → y2 with gain = -1.

  y5 → y4 with gain = -2.

  y5 → y7 with gain = 1.

  y6 → y6 with gain = -1.

  y6 → y5 with gain = 2.



ForwardPaths	Loops
y1y2y3y4y5y7	y2y3y4y5y2
y1y2y6y5y7	y2y6y5y2
	y3y4y3
	y4y5y4
	y6y6

Non-Touching Loops
[y2, y3, y4, y5, y2][y6, y6]
[y2, y6, y5, y2][y3, y4, y3]
[y3, y4, y3][y6, y6]
[y4, y5, y4][y6, y6]

Deltas
2.0
11.0

Delta Total = 450.0

Overall TF = 0.933333333...

2. An SFG of 9 nodes with the following inputs:

- **Nodes:** R, x1, x2, x3, x4, x5, x6, C1, C2
- **Edges:** R → x1 with gain = 1.

  x1 → x2 with gain = 1.

  x2 → x3 with gain = 10.

  x2 → C1 with gain = 40.

  x4 → x5 with gain = 20.

  x5 → x6 with gain = 30.

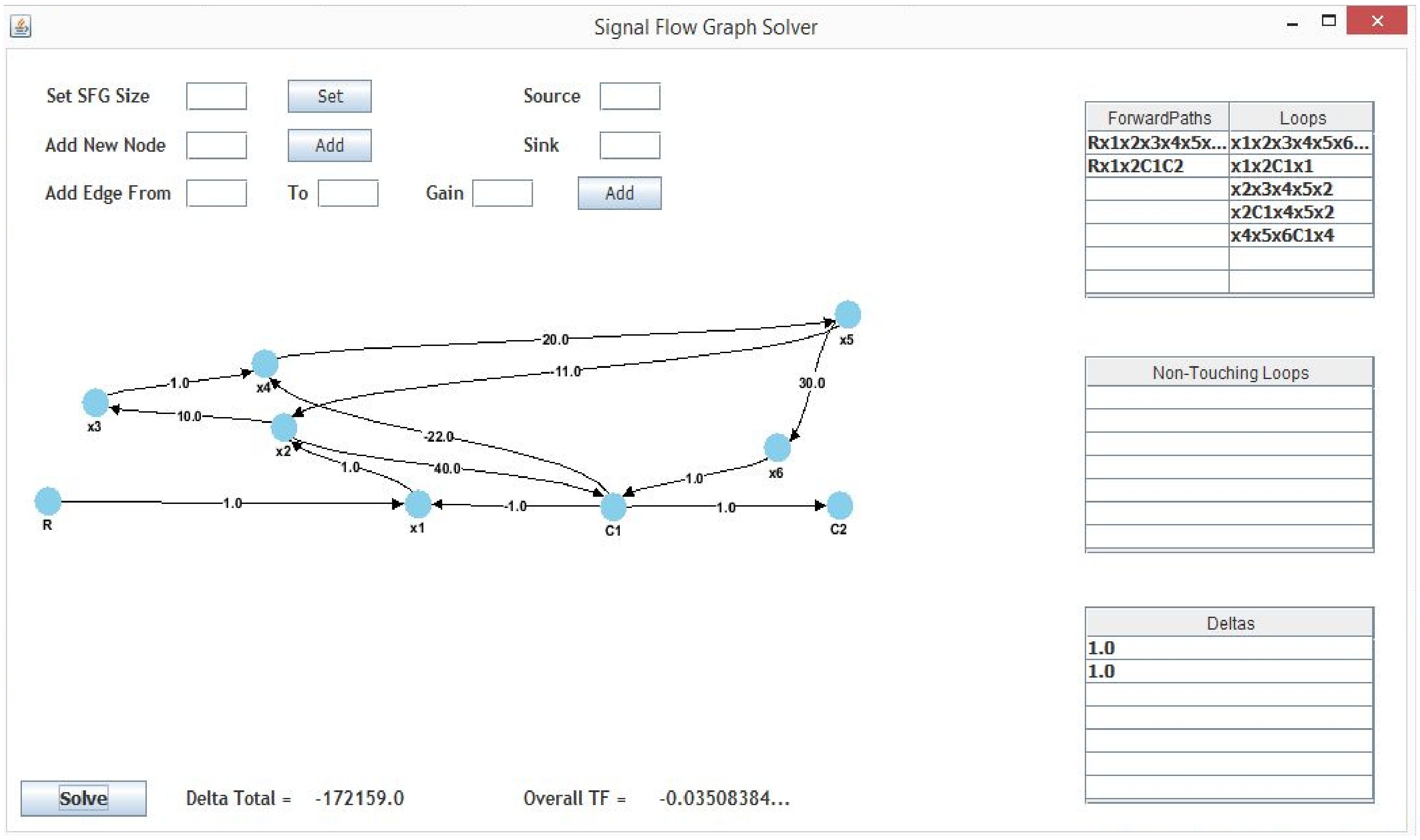
  x5 → x2 with gain = -11.

  x6→ C1 with gain = 1.

  C1 → C2 with gain = 1.

  C1 → x4 with gain = -22.

  C1 → x1 with gain = -1.



ForwardPaths	Loops
Rx1x2x3x4x5x...	x1x2x3x4x5x6...
Rx1x2C1C2	x1x2C1x1
	x2x3x4x5x2
	x2C1x4x5x2
	x4x5x6C1x4

Non-Touching Loops

Deltas
1.0
1.0

Delta Total = -172159.0

Overall TF = -0.03508384...

Credits:

- **Graphstream library for drawing graphs (http://graphstream-project.org/)**