CS 241 : Linear Control Systems

Signal Flow Graph Solver Report

Names:

Amr Mohamed Fathy (45)

Hisham Osama Ghazy (81)

Problem Statement :

It's required to design a **graphical user interface** for a signal flow graph solver application where the user can **freely draw** signal flow graph and **get the numeric result of the overall transfer function** in addition to a **detailed report** containing forward paths, individual loops, all combinations of n non-touching loops and values of deltas of all forward paths.

Main Features :

This signal flow graph solver application enables the user to **freely draw** signal flow graph through a simple graphical user interface using simple buttons and textboxes as follows:

1. **<u>Add Node Button:</u>** This button enables the user to **add a new node** to the signal flow graph in any place with just a mouse click.
   i.e. : The node will be given an initial name that the user can edit later.
2. **<u>Gain Value Textbox:</u>** This textbox enables the user to enter any numeric value for the gain to be used later with the edges.
   i.e. : The default value of the gain will be set to 1 if not specified.
3. **<u>Add Edge Button:</u>** This button enables the user to add an edge from an initial node to a final node where:
   a. The numeric value of the prespecified gain in the textbox will be shown on the edge.
   b. The direction will be shown on the edge.
   c. Two small anchors will be shown on the edge to enable the user to edit the path of the edge (where the user can form curved edges, self loop edges, etc.).
4. **<u>Edit Node Button:</u>** This button enables the user to move any added node freely through any place to come with a good design of the signal flow graph.
5. **<u>Edit Button:</u>** This button enables the user to edit the name of the added nodes.

6. <u>**Solve Button:**</u> This button solves the designed signal flow graph after choosing a **source node** and a **destination node** where a new pop up window will appear with fully organized **detailed report** containing tables showing:
    a. **Forward Paths** and their gains
    b. **Individual Loops** and their gains
    c. **All combination of n non-touching loops** and the loops taking place in this combinations
    d. The value of **deltas** of all forward paths
    e. The numeric value of the overall **transfer function**

Data Structure :

1. **Adjacency List** : to represent graph content , each node has list of edges and each edge has two parts (destination node , gain of edge).
2. **java.util.Map :** to map each node to its given name and access it with its name easily.
3. **Java.util.List :** to store forward paths between source node and destination.
4. **java.util.List :** to store individual loops content and their gain which is independent of input and output nodes.
5. **java.util.List :** to store all combination of n non-touching loops content.
6. **java.util.List :** to store values of delta (∆ , ∆1 , ∆2 , ... ∆M) .

Main Modules :

There are 6 packages in the program :

1. **Comp** : contains all graph components which we need (Node , Edge , Path)
    - **Node** : each node represent variable which has a name and a list of edges represents the children nodes and the gain of the edge.
    - **Edge :** represent an edge between two nodes and contains the gain of edge.
    - **Path :** represents a path through some nodes on graph and the gain of this path which is equal to product of the edges gain on that path.
2. **Signal.flow.graph :** contains the logic of calculations and how to get the paths and we we will discuss that in Algorithms used part
3. **Application :** contains the application launching .
4. **View :** contains a xml file which describe the view of the graphical interface including buttons , labels , …. etc.
5. **Controller :** contains all modules which are responsible of handling user actions and how to respond well to each action like adding a node , dragging a node , adding an edge , editing node name and solve part which is well described in User Guide Part .
6. **Controller.utils :** contains modules which are responsible of representing output data in well organized from in the table .

Algorithms used :

There are two main algorithms we use :

1. **Depth First Search (DFS) Algorithm on Graph** : we use the DFS Algorithm with some modifications to detect the all forward paths between two nodes , Moreover detecting all the loops including self loops in the graph .

   How it works ?

   We start from the source node (input node) and try to visit all the nodes which are connected with it and repeat this step on the next node until we reach the output node (target node) then this is a forward path , if we reach a node which we have already visit it before that means this path is a loop . while doing the previous action we calculate the gain of the current path we reach it till now.

2. **Depth First Search (DFS) Algorithm on Graph :** but this time we use it with other modification in order to get all the non-touching loops .

   How it works ?

   We first get all the two non touching loops combinations then build a graph (a node in the new graph means a loop) we make edge between two nodes if and only if these two loops ara non touching and so , after we construct the new graph we start at each node and try to add the children to a non touching set with it and so till we find we can't add more children (children mean loops as we said above a node means a loop) otherwise we will make conflict with the non touching set and so till we finish traversing the new graph . after that we will find we have already find all the non touching combinations of size 1 ,2 ,3, … n if they exist .

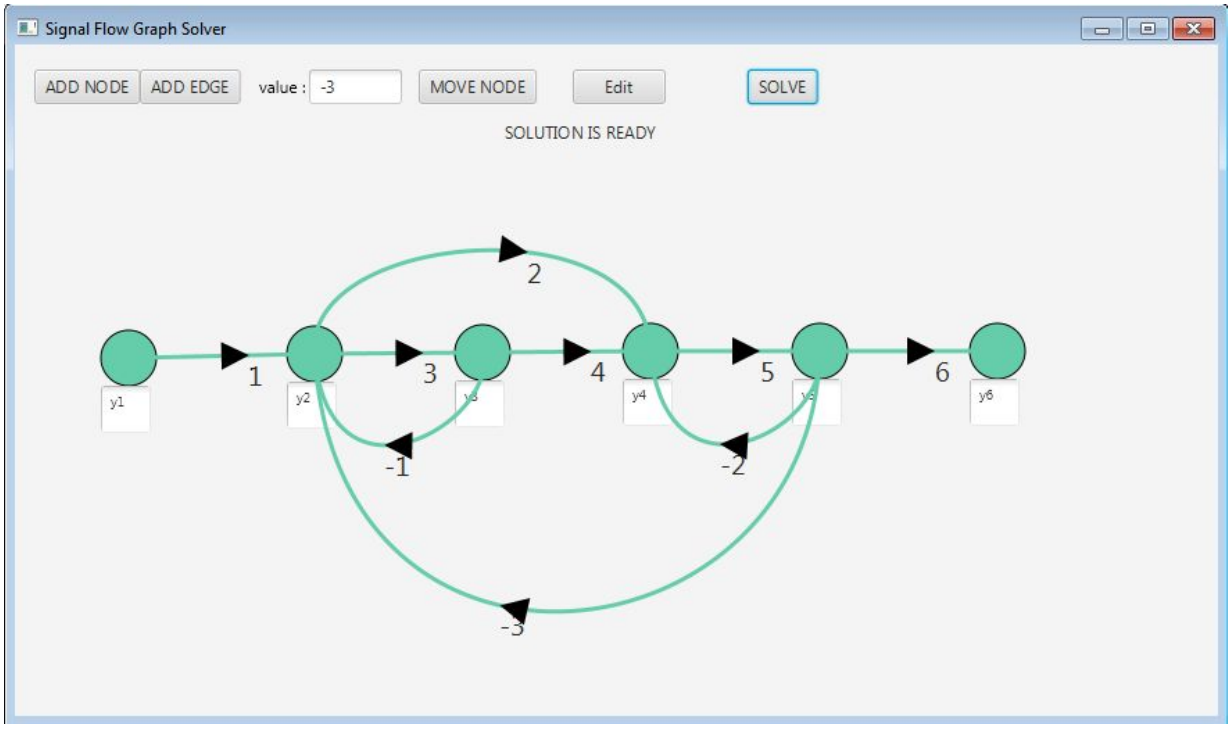
User Guide :

1. Open the application.
2. **Adding a new node**
    a. Press “Add Node”.
    b. Select the place where the node would be placed using the mouse click.
3. Press “Edit“ if you want to change default name of the node.
4. Press “Move Node“ to drag and drop the node to any place if you want to organize the your signal flow graph more.
5. **Adding a new edge**
    a. Write the numeric value of the gain in the text field provided by the application
    b. Press “Add edge”.
    c. Click on the initial node (starting node).
    d. Click on the final node (ending node).
    e. Edit the edge path by using the provided two anchors if you need.
6. Repeat the previous steps until you finish your design.

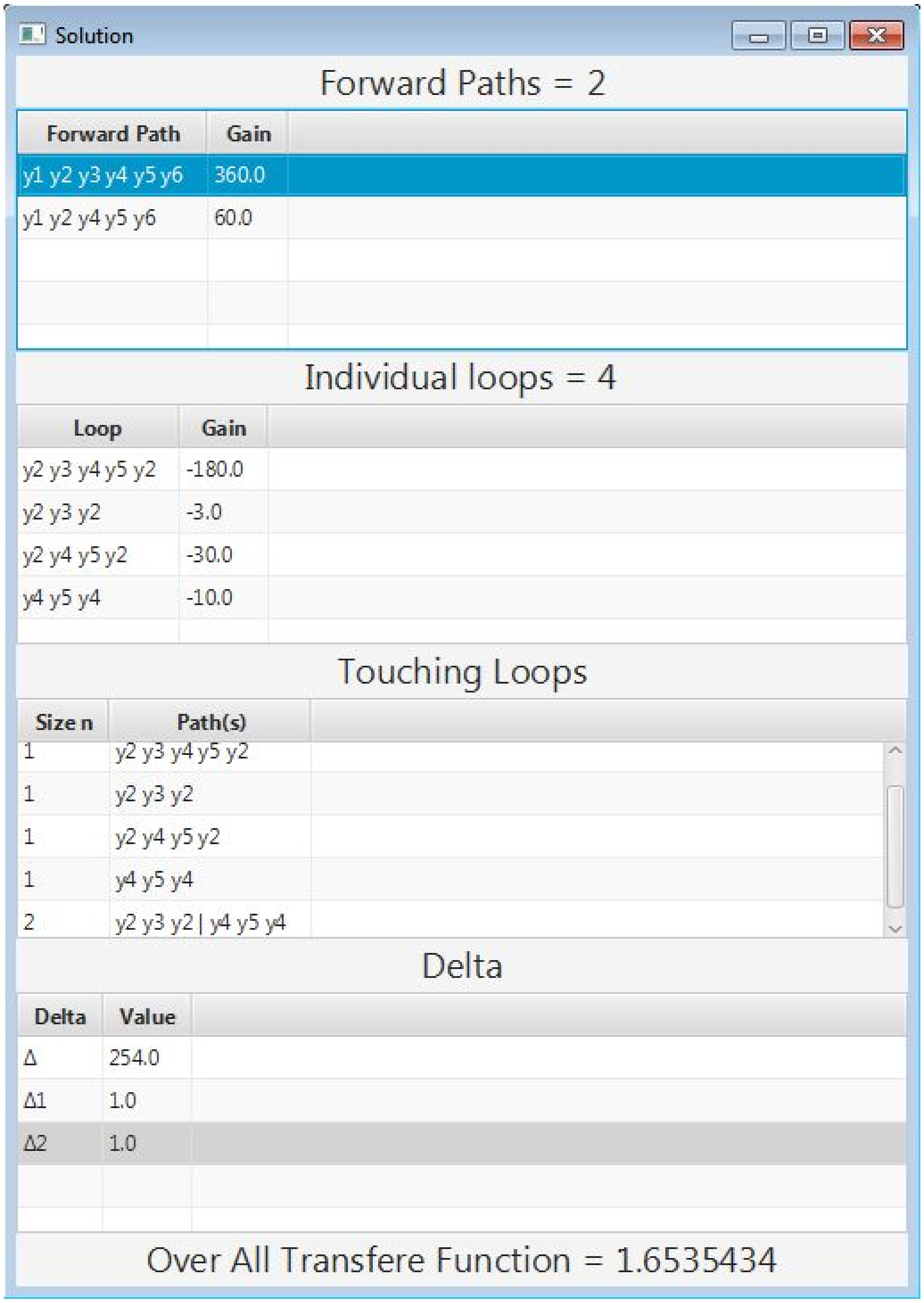
7. **Solving the signal flow graph**
    a. Press “Solve” button
    b. Click on the source node
    c. Click on the destination node
    d. Check the detailed report showing the results of your design

Sample Runs:

Sheet #4 : Example 2 modified with numeric values



Solution for $Y6 \div Y1$



Solution

Forward Paths = 2

Forward Path	Gain
y1 y2 y3 y4 y5 y6	360.0
y1 y2 y4 y5 y6	60.0

Individual loops = 4

Loop	Gain
y2 y3 y4 y5 y2	-180.0
y2 y3 y2	-3.0
y2 y4 y5 y2	-30.0
y4 y5 y4	-10.0

Touching Loops

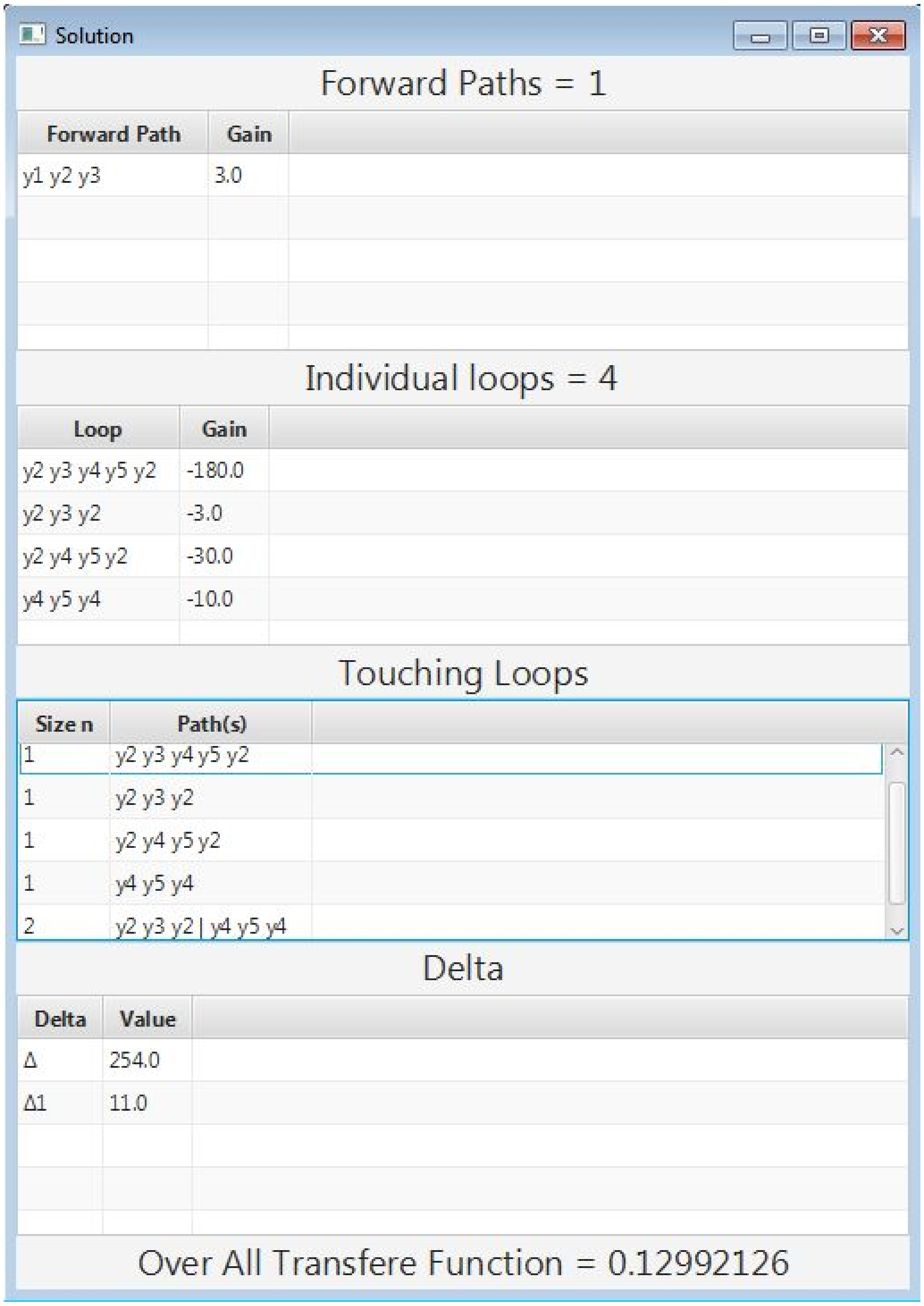
Size n	Path(s)
1	y2 y3 y4 y5 y2
1	y2 y3 y2
1	y2 y4 y5 y2
1	y4 y5 y4
2	y2 y3 y2 | y4 y5 y4

Delta

Delta	Value
Δ	254.0
Δ1	1.0
Δ2	1.0

Over All Transfere Function = 1.6535434

Solution for $Y3 \div Y1$



Forward Paths = 1

Forward Path	Gain
y1 y2 y3	3.0

Individual loops = 4

Loop	Gain
y2 y3 y4 y5 y2	-180.0
y2 y3 y2	-3.0
y2 y4 y5 y2	-30.0
y4 y5 y4	-10.0

Touching Loops

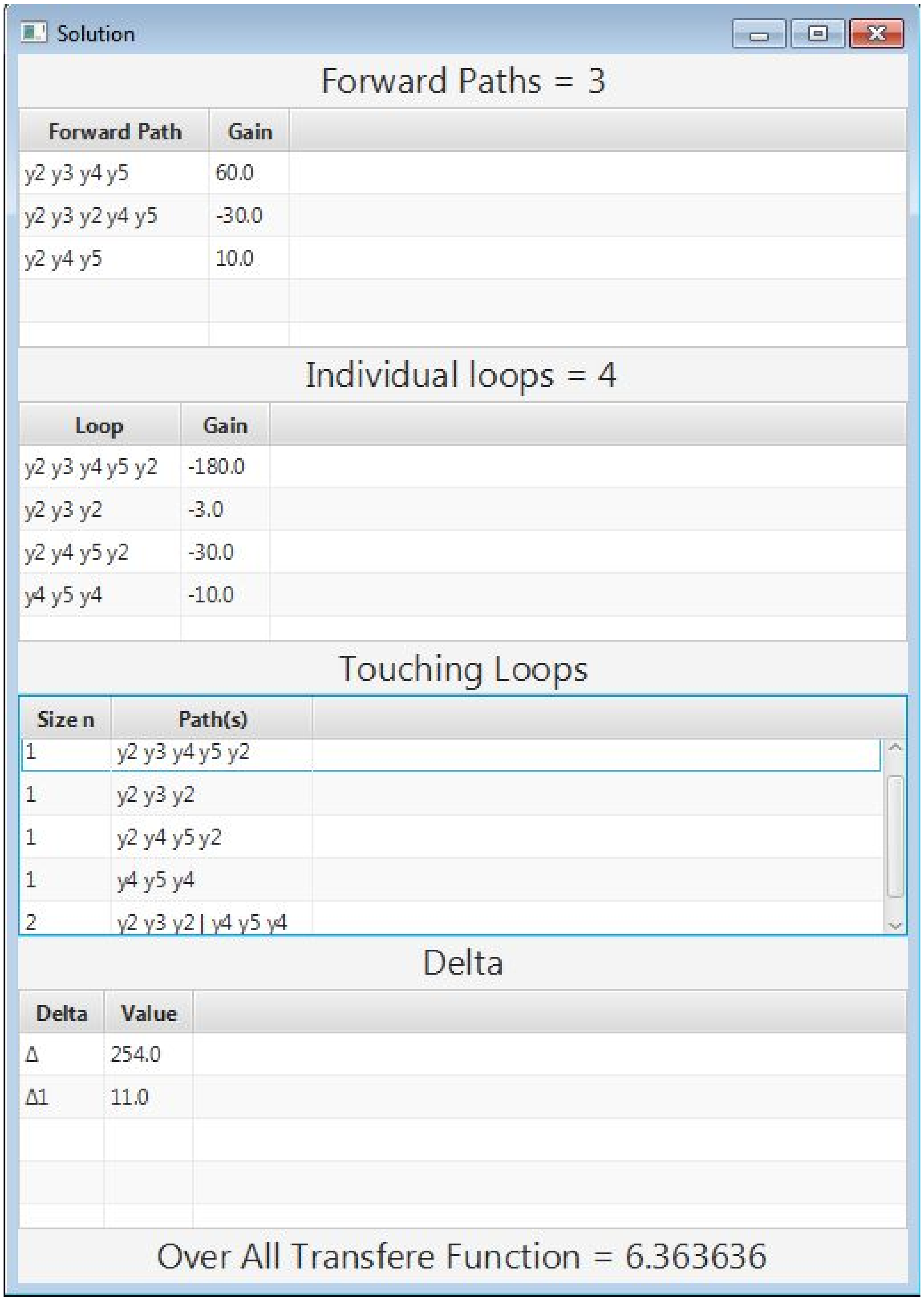
Size n	Path(s)
1	y2 y3 y4 y5 y2
1	y2 y3 y2
1	y2 y4 y5 y2
1	y4 y5 y4
2	y2 y3 y2 | y4 y5 y4

Delta

Delta	Value
Δ	254.0
Δ1	11.0

Over All Transfere Function = 0.12992126

Solution for $Y5 \div Y2$



Solution

Forward Paths = 3

Forward Path	Gain
y2 y3 y4 y5	60.0
y2 y3 y2 y4 y5	-30.0
y2 y4 y5	10.0

Individual loops = 4

Loop	Gain
y2 y3 y4 y5 y2	-180.0
y2 y3 y2	-3.0
y2 y4 y5 y2	-30.0
y4 y5 y4	-10.0

Touching Loops

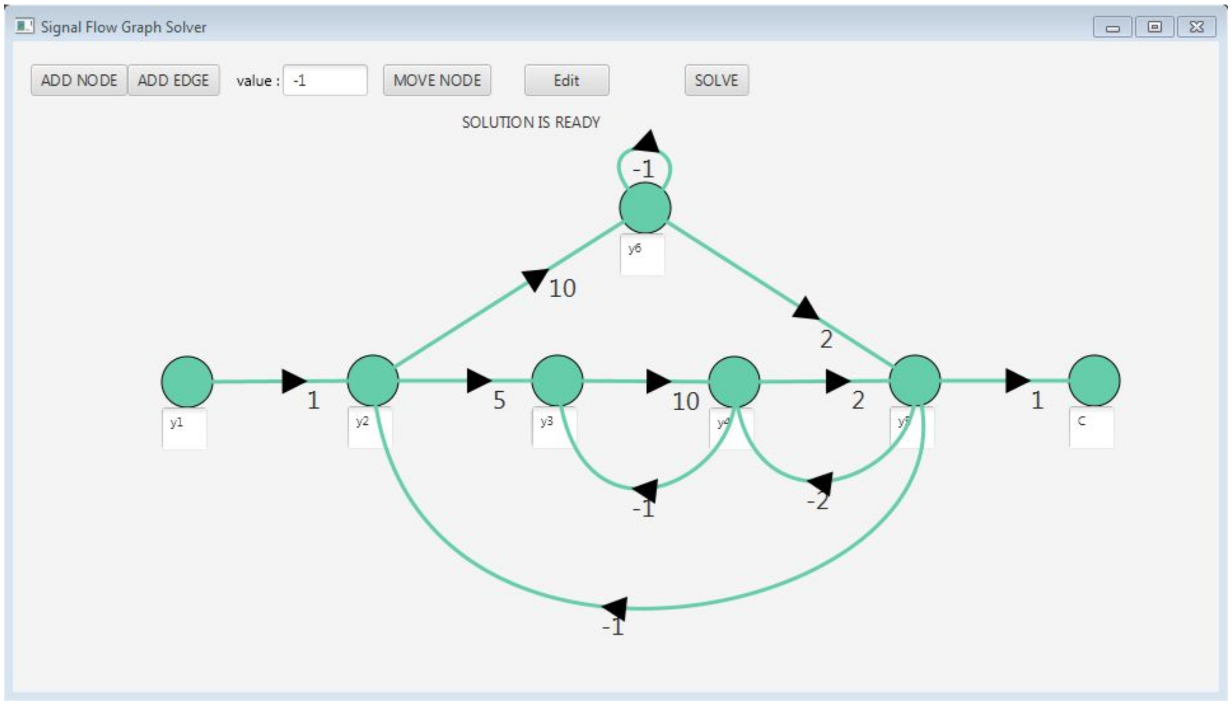
Size n	Path(s)
1	y2 y3 y4 y5 y2
1	y2 y3 y2
1	y2 y4 y5 y2
1	y4 y5 y4
2	y2 y3 y2 | y4 y5 y4

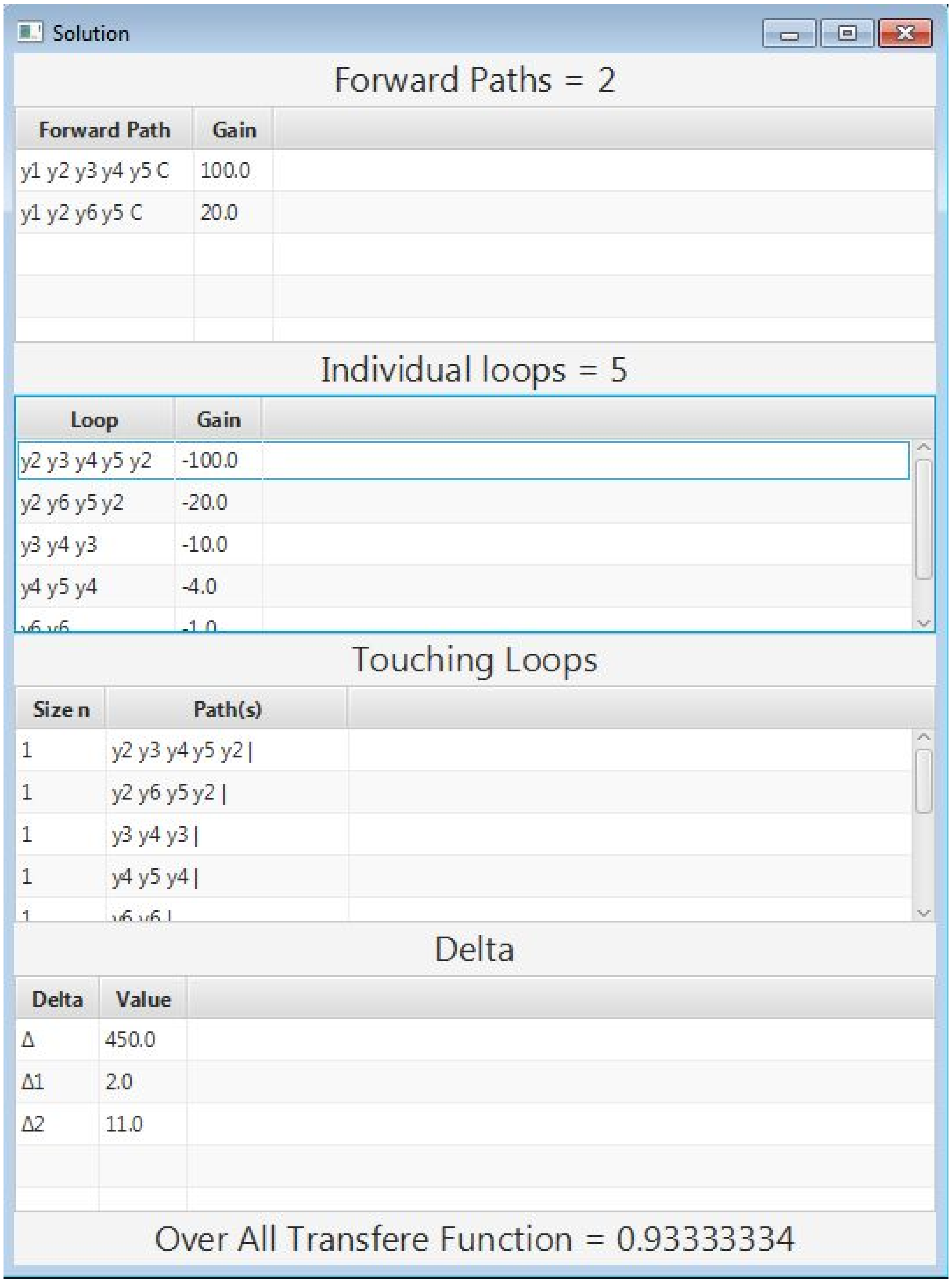
Delta

Delta	Value
Δ	254.0
Δ1	11.0

Over All Transfere Function = 6.363636

Sheet #4 : Example 3


Signal Flow Graph Solver
ADD NODE
ADD EDGE
value : -1
MOVE NODE
Edit
SOLVE
SOLUTION IS READY
-1
y6
10
2
1
5
10
2
1
y1
y2
y3
y4
y5
C
-1
-2
-1



Forward Paths = 2

Forward Path	Gain
y1 y2 y3 y4 y5 C	100.0
y1 y2 y6 y5 C	20.0

Individual loops = 5

Loop	Gain
y2 y3 y4 y5 y2	-100.0
y2 y6 y5 y2	-20.0
y3 y4 y3	-10.0
y4 y5 y4	-4.0
y6 y6	-1.0

Touching Loops

Size n	Path(s)
1	y2 y3 y4 y5 y2 |
1	y2 y6 y5 y2 |
1	y3 y4 y3 |
1	y4 y5 y4 |
1	y6 y6 |

Delta

Delta	Value
Δ	450.0
Δ1	2.0
Δ2	11.0

Over All Transfere Function = 0.93333334