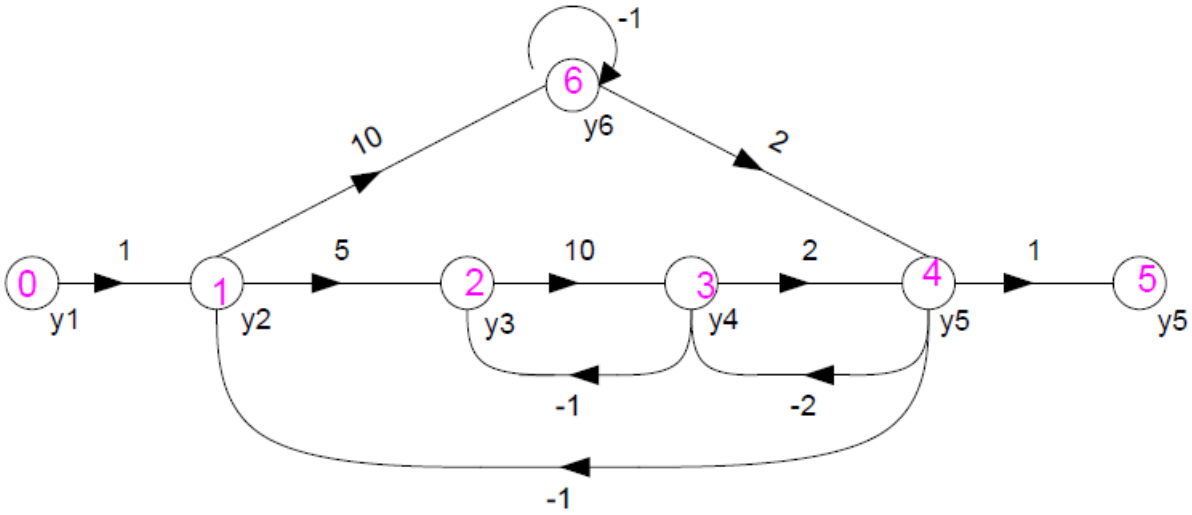
Signal Flow Graph



- **Prepared by :**

1. *Bassent Mostafa Saad Zaghlol (21).*
2. *Mira Samir Ragheb (78).*

• Problem Statement:

⇨ It is required to implement a Signal Flow Graph representation of the system when total number of nodes and numeric branches gains are given.

Requirements:

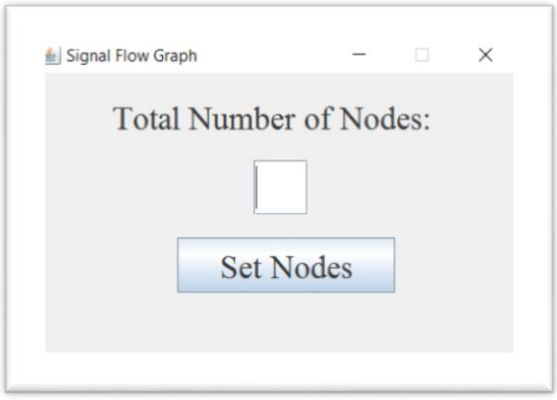
1- Implementing a Graphical user interface (GUI).
2- Drawing the signal flow graph showing nodes, branches, gains, …
3- Listing all forward paths, individual loops, all combination of *n* non-touching loops.
4- The values of Δ , $\Delta 1$, …, Δm where m is number of forward paths.
5- Overall system transfer function.

• Main Features of the Program:

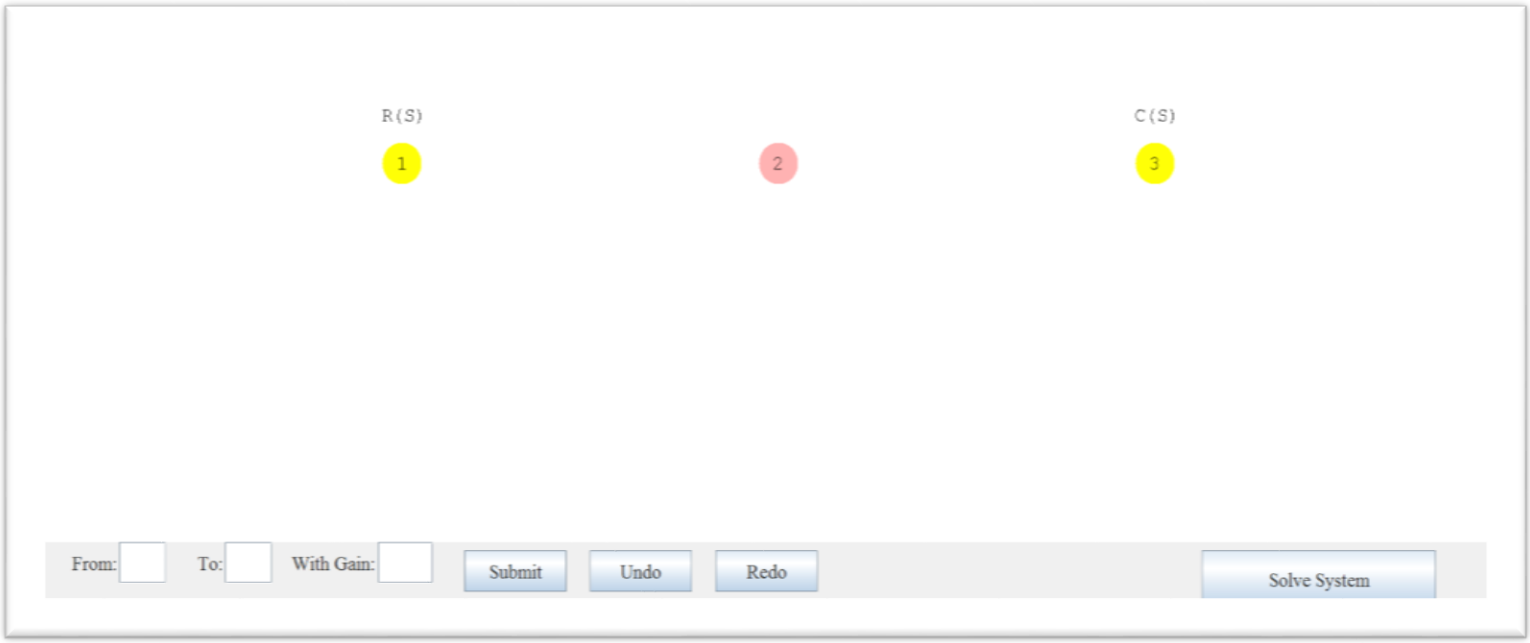
Java Program where:

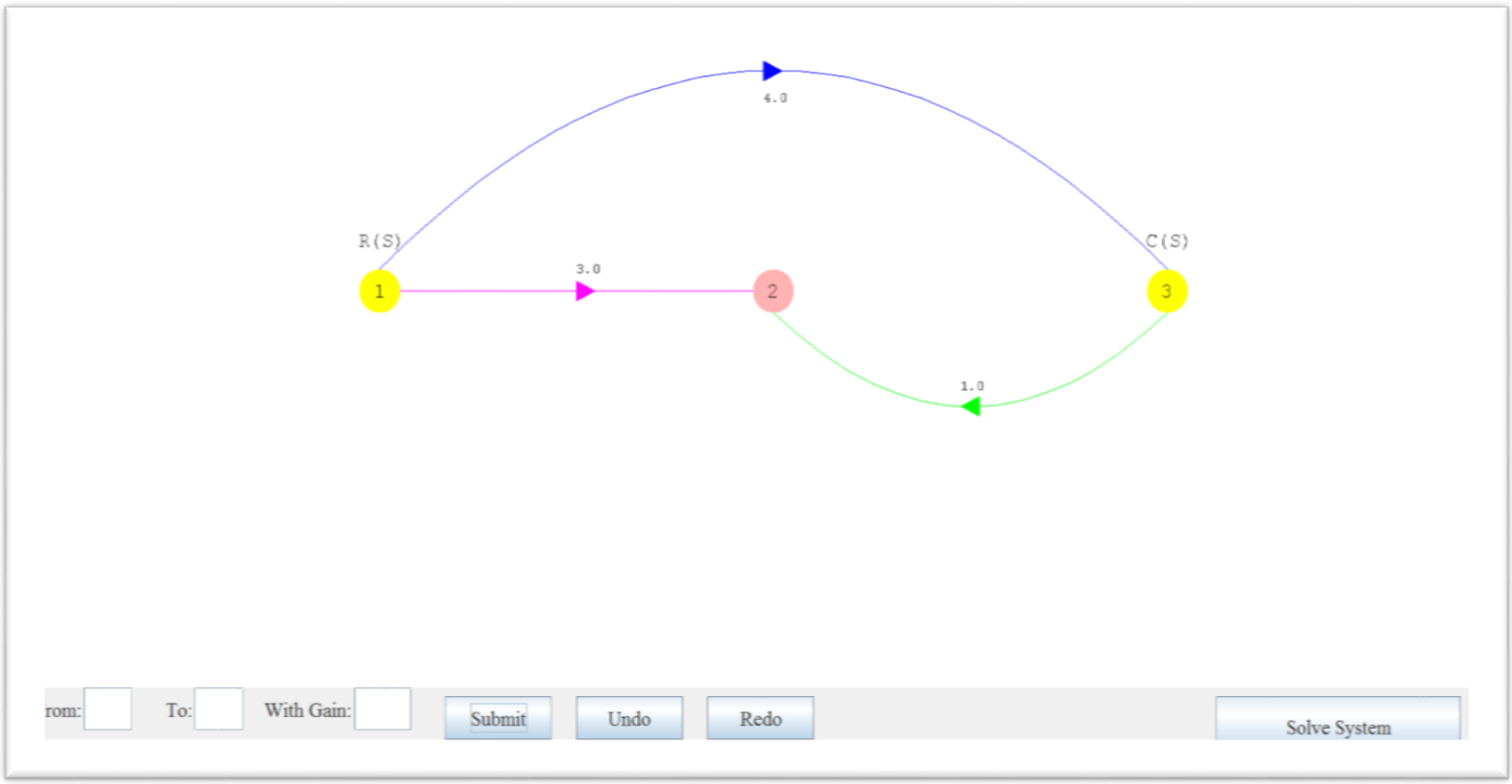
User's able to:

⇨ Draw a Signal Flow Graph for a feedback control system through a User-friendly Interface by specifying the number of nodes to be represented.



⇨ Draw the Signal Flow Graph step by step by adding either forward or feedback branches between specified nodes.

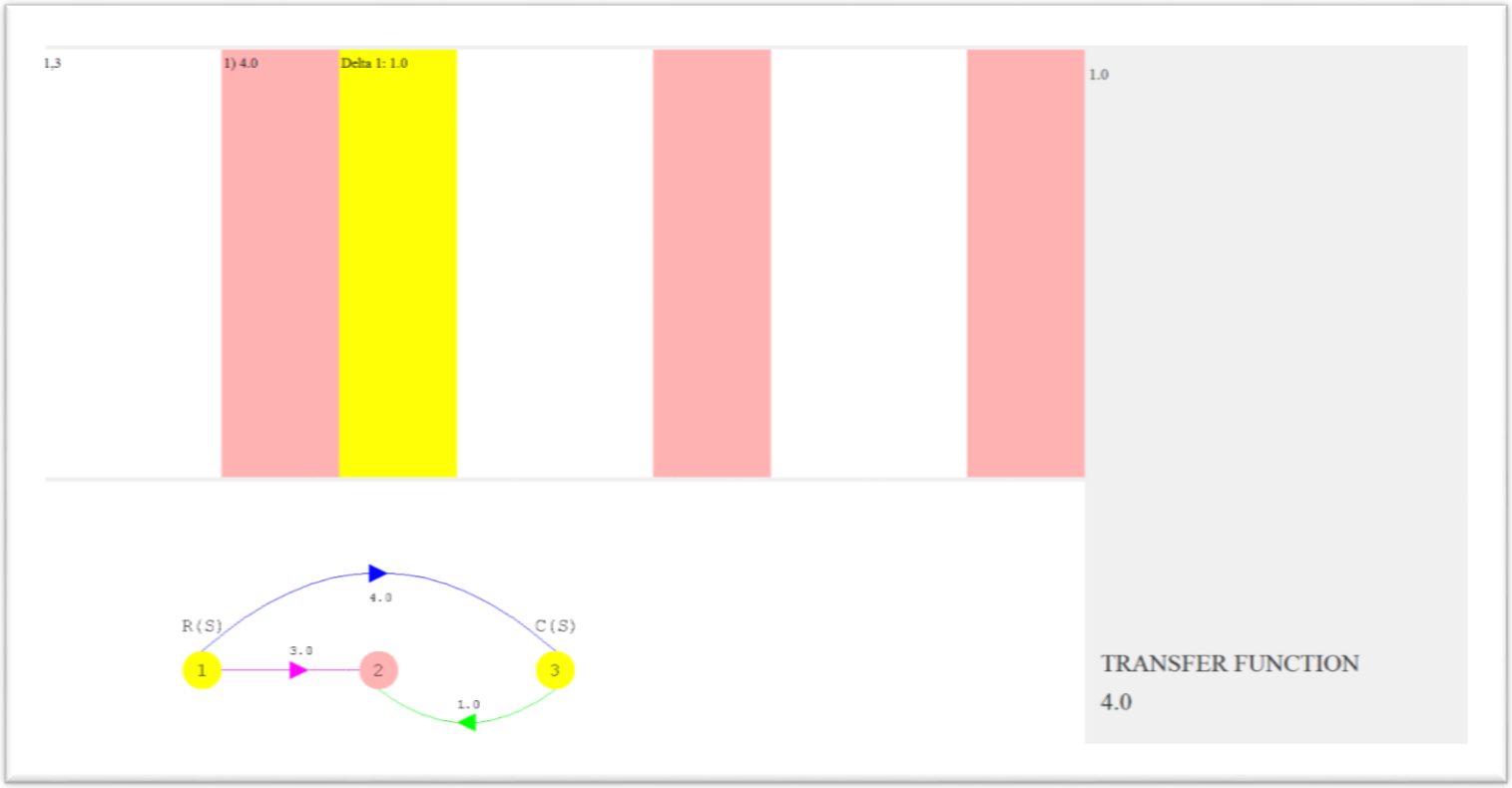




⇨ Solve a given Feedback Control System using Mason's Formula.

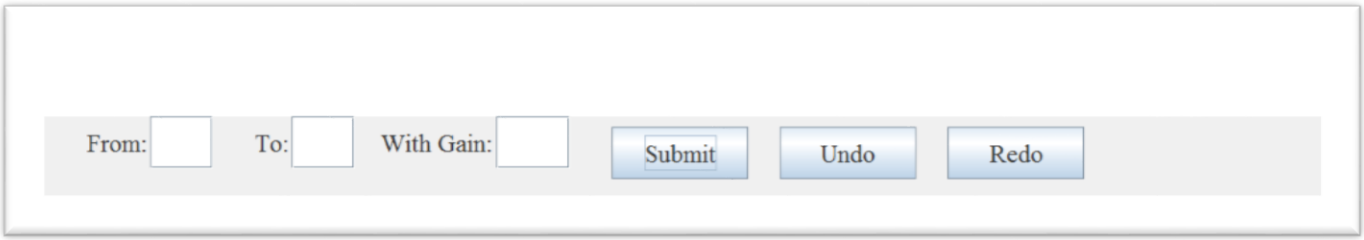
1- A Tabular View representing all forward paths, individual loops, all combinations of the n non-touching loops, values of Δ and big delta together with their gains are listed for the required system to be solved.

2- Total transfer function of the required system to be solved is calculated.

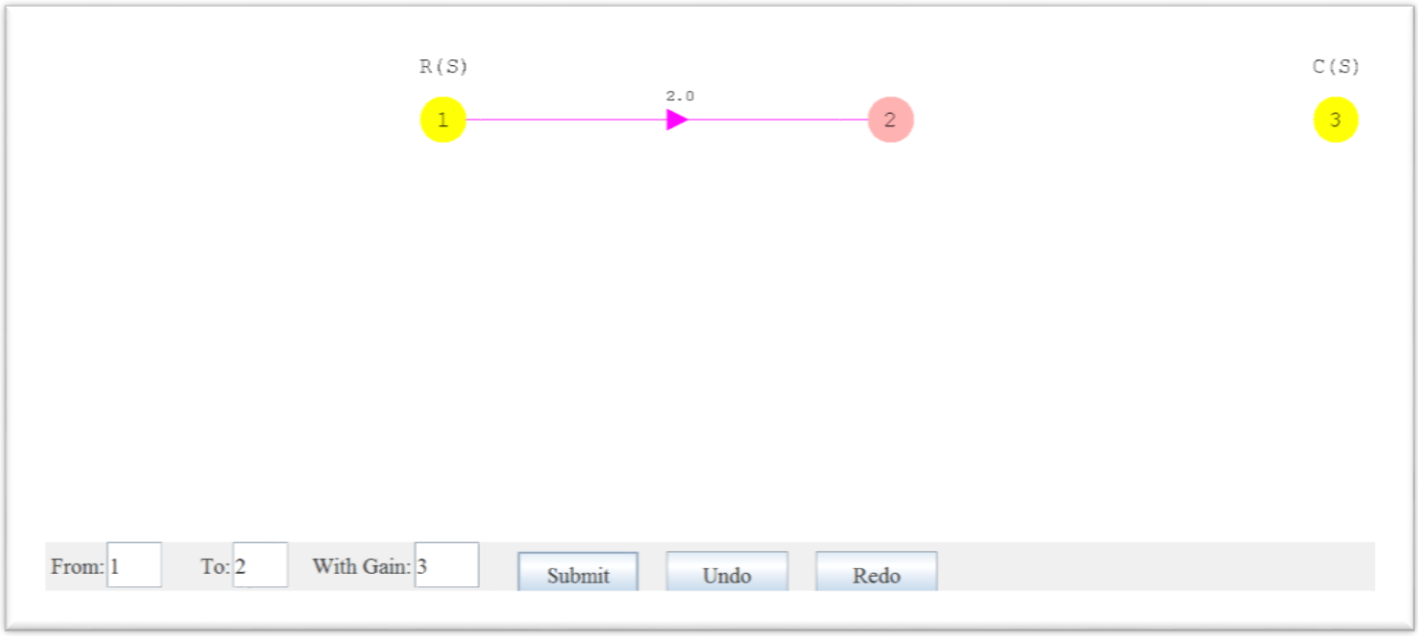


- **Additional Options/ Features:**

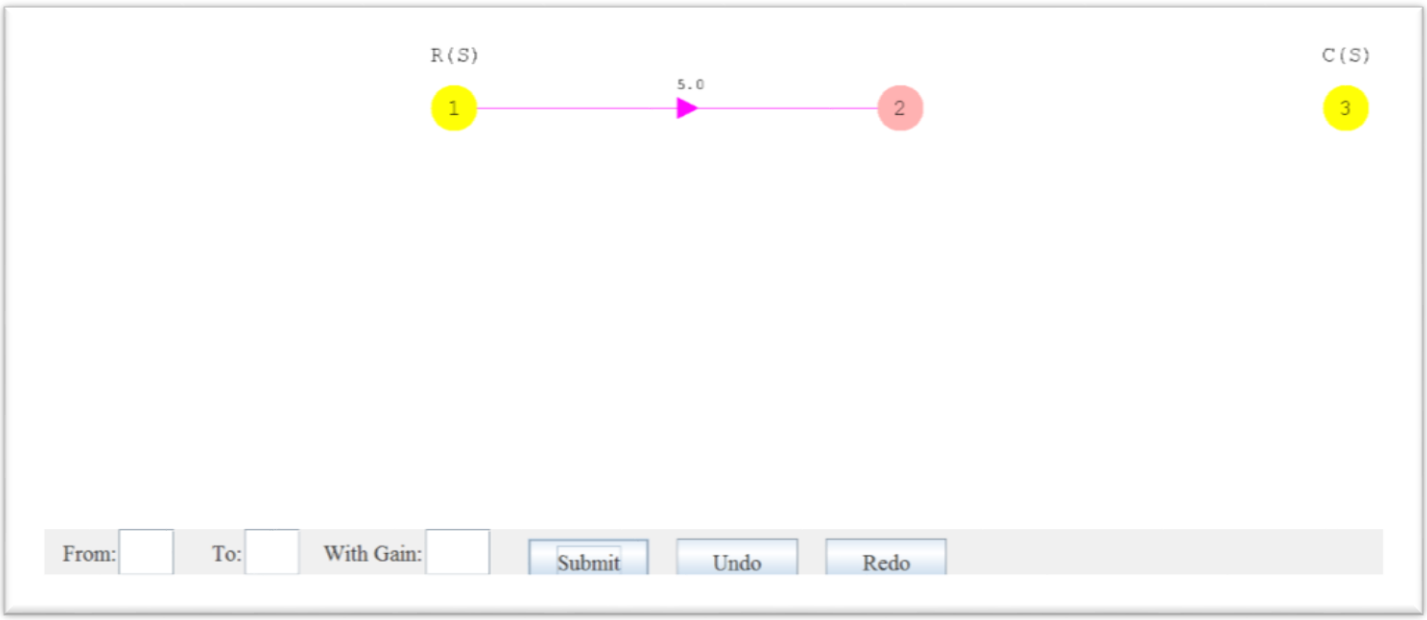
1- User can undo and redo their actions through two buttons.



2- When adding a branch between two nodes with already defined branch, a new branch is added between the two nodes equal to the sum of the two defined branches.

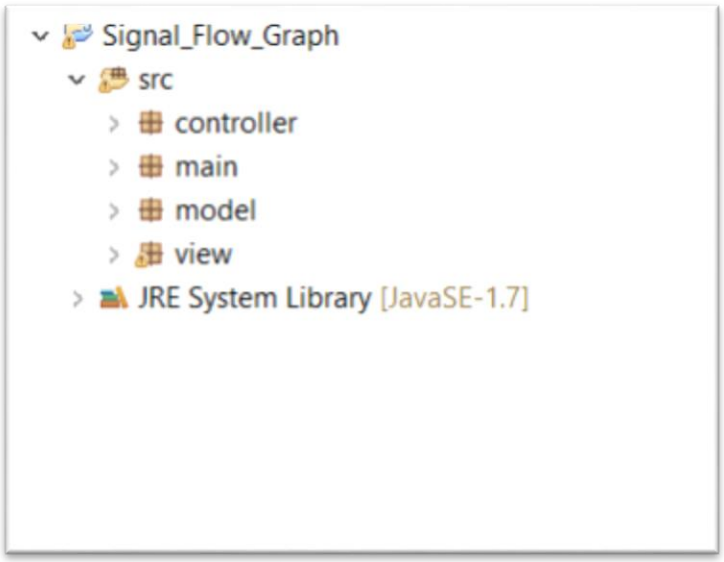


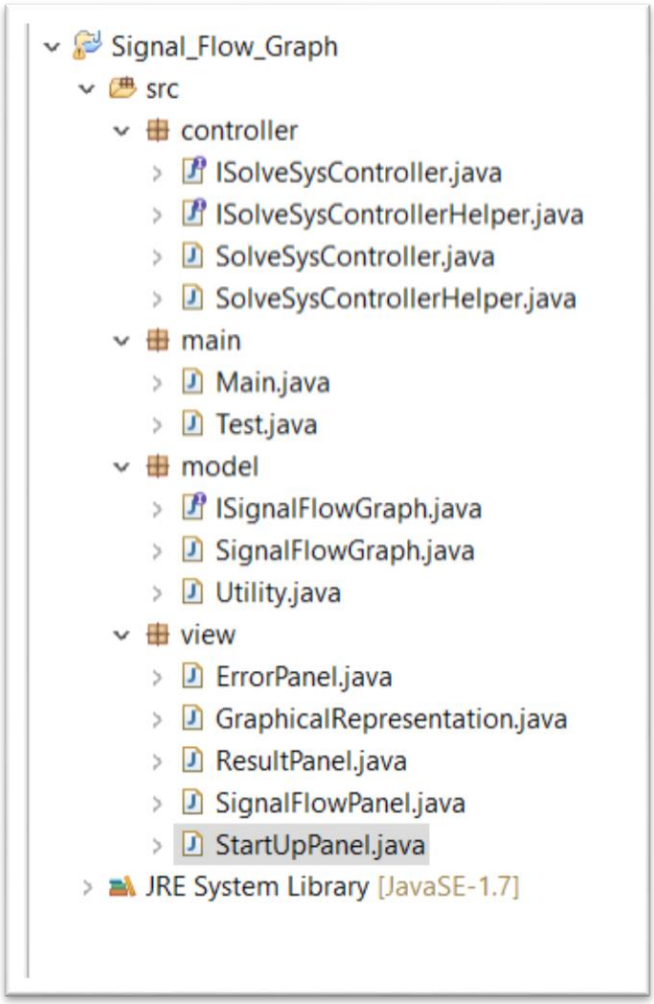
Adding another branch/gain from node 1 to node 2 yields a new branch with gain equal to the sum of the predefined branch's gain and the new introduced branch's gain.



Same goes for feedback branches.

- **Main Modules:** Our program consists of three packages:





1) Controller Package:
   Contains the main classes to help transfer data between the user view and the model side.
2) Main Package:
3) Model Package:
   The classes which holds data of the signal flow graph and computes the total transfer function.
4) View Package:
   The user view panels interacting with the user.
   e.g. The start-up panel, The error window, The signal flow panel and the Results window.

• Data Structures Used:

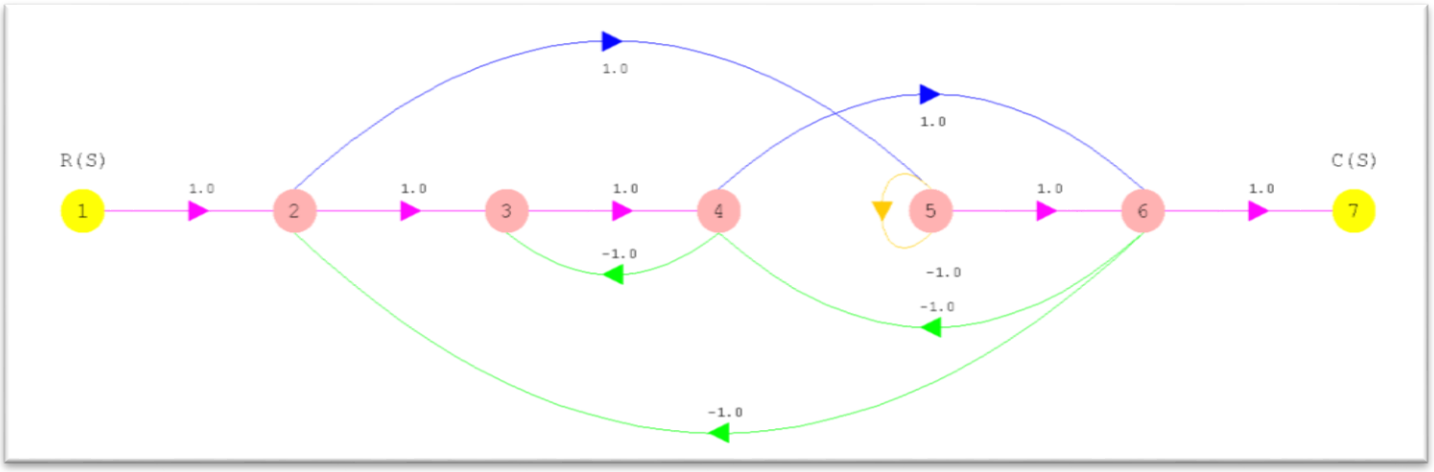
1. **Adjacency list:** In order to represent the Signal Flow Graph using an Arraylist of hashmaps (Key: is the adjacent node, Value: is the weight).
2. Arrays of fixed size to represent different gains and deltas.
3. ArrayLists in order to represent both loops and forward paths.

• Algorithms Used:

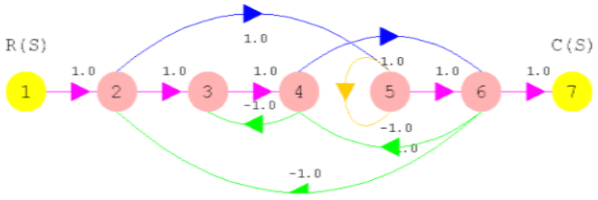
1- **Depth First Search (DFS)** Algorithm to calculate forward paths and loops.
2- *Mason's* algorithm in order to calculate the total Transfer Function of the Signal Flow Graph.

- **Sample runs:**

1)

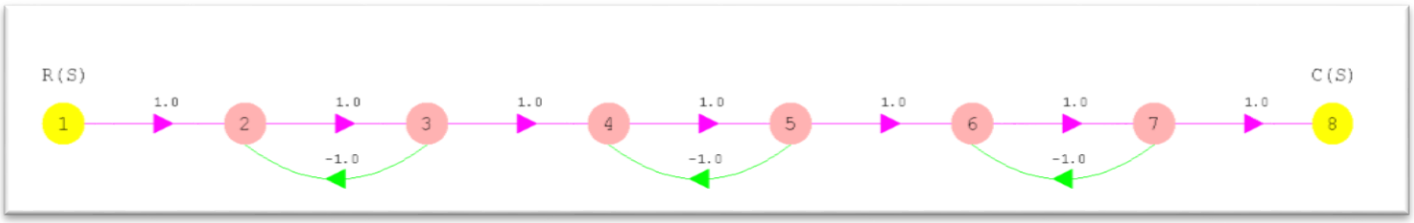


ORWARD PATHS	W/ GAIN	DELTA GAIN	LOOPS	W/ GAIN	NON TOUCHING LOOPS	W/ GAIN	BIG DELTA
) 1,2,3,4,6,7) 1,2,5,6,7	1) 1.0 2) 1.0	Delta 1: 2.0 Delta 2: 2.0	1) 2,3,4,6,2 2) 2,5,6,2 3) 3,4,3 4) 4,6,4 5) 5,5	1) -1.0 2) -1.0 3) -1.0 4) -1.0 5) -1.0	1 non-Touching Loops: 1) 2,3,4,6,2 2) 2,5,6,2 3) 3,4,3 4) 4,6,4 5) 5,5 2 non-Touching Loops: 1) 2,3,4,6,2 / 5,5 2) 2,5,6,2 / 3,4,3 3) 3,4,3 / 5,5 4) 4,6,4 / 5,5	1) -1.0 2) -1.0 3) -1.0 4) -1.0 5) -1.0 1) 1.0 2) 1.0 3) 1.0 4) 1.0	10.0

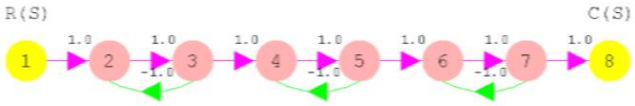


TRANSFER FUNCTION
0.4

2)



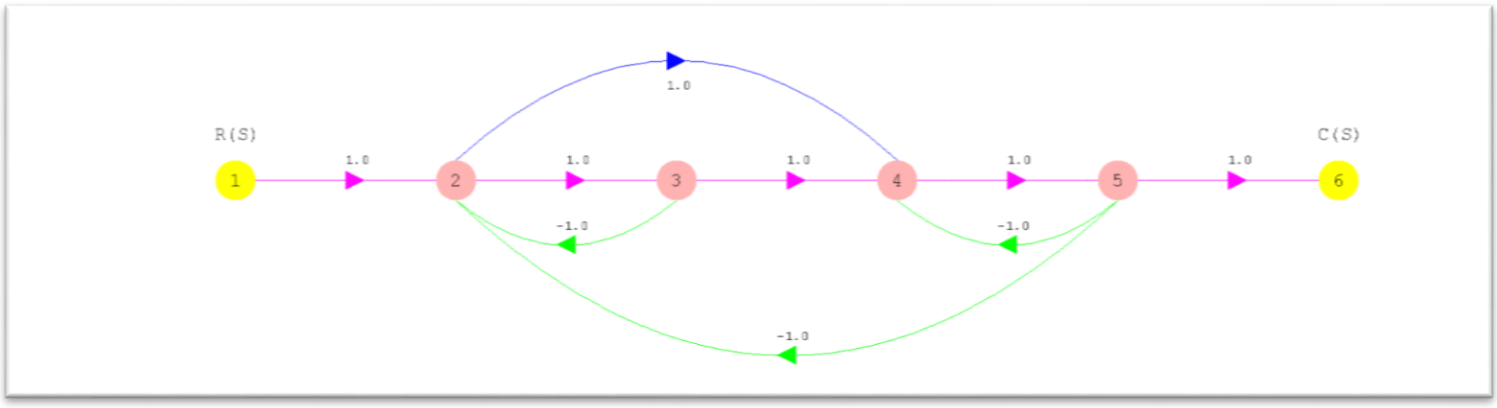
FORWARD PATHS	W/ GAIN	DELTA GAIN	LOOPS	W/ GAIN	NON TOUCHING LOOPS	W/ GAIN	BIG DELTA
) 1,2,3,4,5,6,7,8	1) 1.0	Delta 1: 1.0	1) 2,3,2 2) 4,5,4 3) 6,7,6	1) -1.0 2) -1.0 3) -1.0	1 non-Touching Loops: 1) 2,3,2 2) 4,5,4 3) 6,7,6 2 non-Touching Loops: 1) 2,3,2 / 4,5,4 2) 2,3,2 / 6,7,6 3) 4,5,4 / 6,7,6 3 non-Touching Loops: 1) 2,3,2 / 4,5,4 / 6,7,6	1) -1.0 2) -1.0 3) -1.0 1) 1.0 2) 1.0 3) 1.0 1) -1.0	8.0



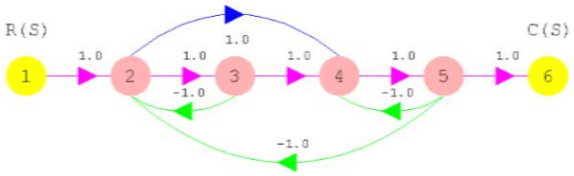
TRANSFER FUNCTION

0.125

3)

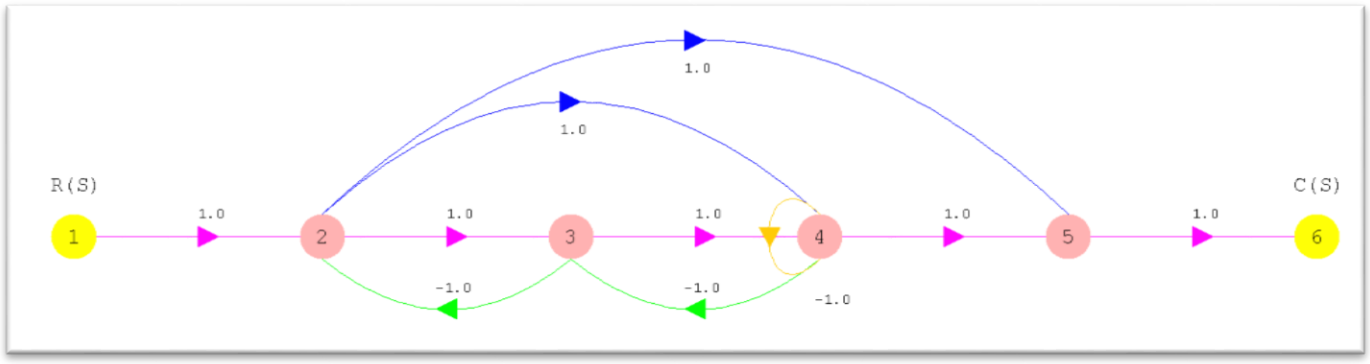


ORWARD PATHS	W/ GAIN	DELTA GAIN	LOOPS	W/ GAIN	NON TOUCHING LOOPS	W/ GAIN	BIG DELTA
) 1,2,3,4,5,6) 1,2,4,5,6	1) 1.0 2) 1.0	Delta 1: 1.0 Delta 2: 1.0	1) 2,3,2 2) 2,3,4,5,2 3) 2,4,5,2 4) 4,5,4	1) -1.0 2) -1.0 3) -1.0 4) -1.0	1 non-Touching Loops: 1) 2,3,2 2) 2,3,4,5,2 3) 2,4,5,2 4) 4,5,4 2 non-Touching Loops: 1) 2,3,2 / 4,5,4	1) -1.0 2) -1.0 3) -1.0 4) -1.0 1) 1.0	6.0

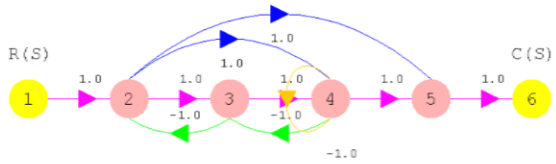


TRANSFER FUNCTION
0.3333333333333333

4)



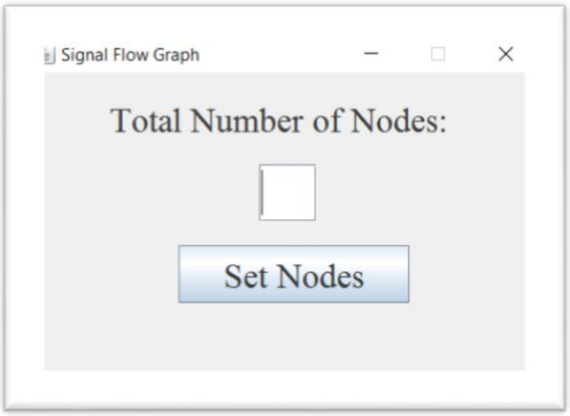
ORWARD PATHS	W/ GAIN	DELTA GAIN	LOOPS	W/ GAIN	NON TOUCHING LOOPS	W/ GAIN	BIG DELTA
) 1,2,3,4,5,6) 1,2,4,5,6) 1,2,5,6	1) 1.0 2) 1.0 3) 1.0	Delta 1: 1.0 Delta 2: 1.0 Delta 3: 3.0	1) 2,3,2 2) 2,4,3,2 3) 3,4,3 4) 4,4	1) -1.0 2) 1.0 3) -1.0 4) -1.0	1 non-Touching Loops: 1) 2,3,2 2) 2,4,3,2 3) 3,4,3 4) 4,4 2 non-Touching Loops: 1) 2,3,2 / 4,4	1) -1.0 2) 1.0 3) -1.0 4) -1.0 1) 1.0	4.0



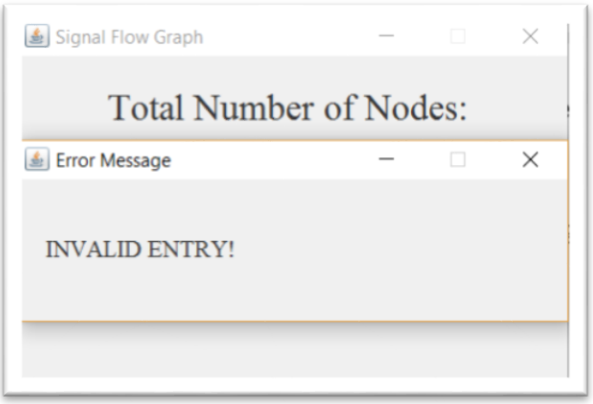
TRANSFER FUNCTION
1.25

- ## Simple User Guide:

1- On running the program, a small window appears asking the user to enter the required number of nodes to represent.
It should be noticed that the user will have to stick to the number of nodes he entered at the beginning during the whole program, and to change the number of nodes: user will have to rerun the program.

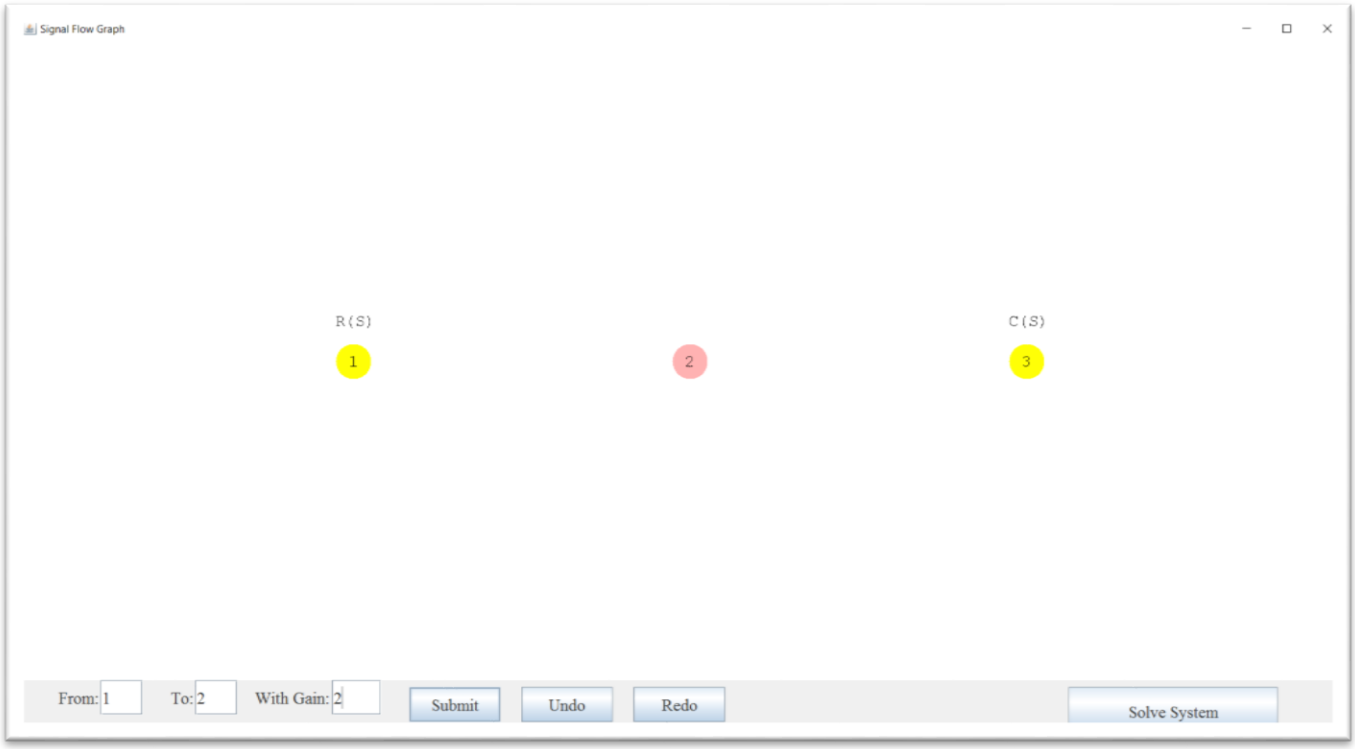


Only non-zero positive integer numbers are valid.
On entering an invalid input, an error message appears with the following message and user must reenter a valid number of nodes to start with.



2- User will be redirected to a new window, with the following features:

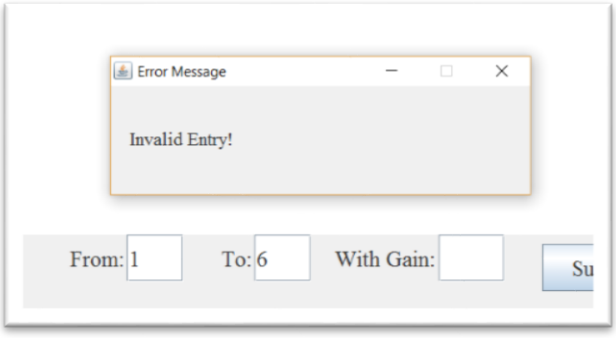
- From, To and gain text boxes where user can enter a valid node number according to the node labels represented to specify a branch between them with a gain.



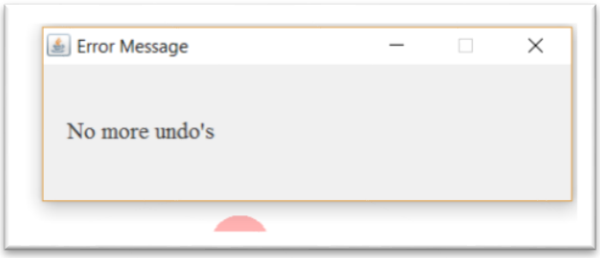
Gain must be a numeric, maybe floating point, entry.

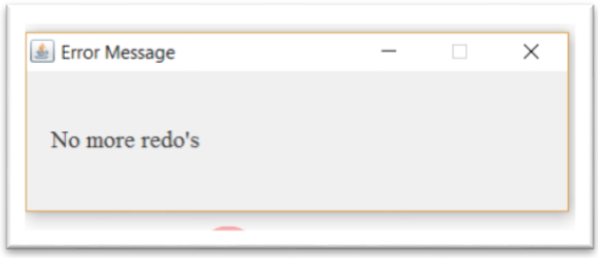
Notes:

- *Nodes are labeled one based.*
- *Unspecifying a gain value to a branch generates an error message.*

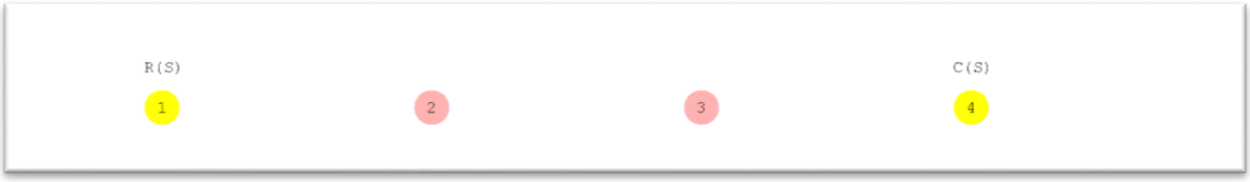


- *Undo and redo actions are permitted through two buttons, reaching the maximum number of undo's, or redo's generates an error message.*

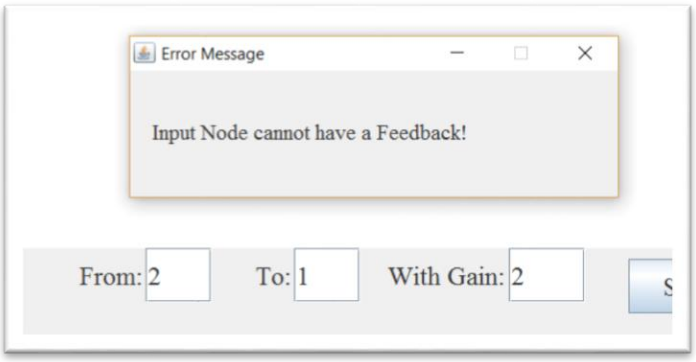




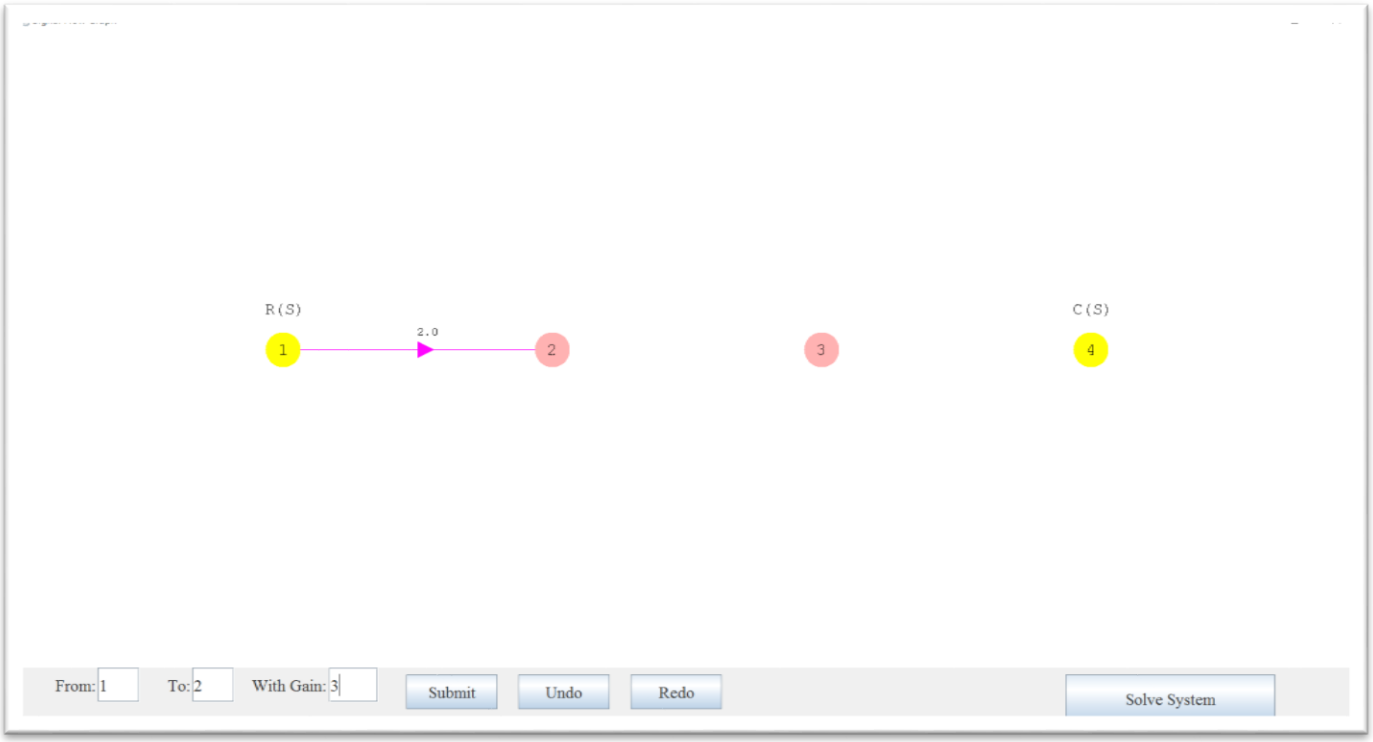
- *Input and output nodes are specified with R(s) and C(s) labels respectively, also they are yellow-colored nodes.*

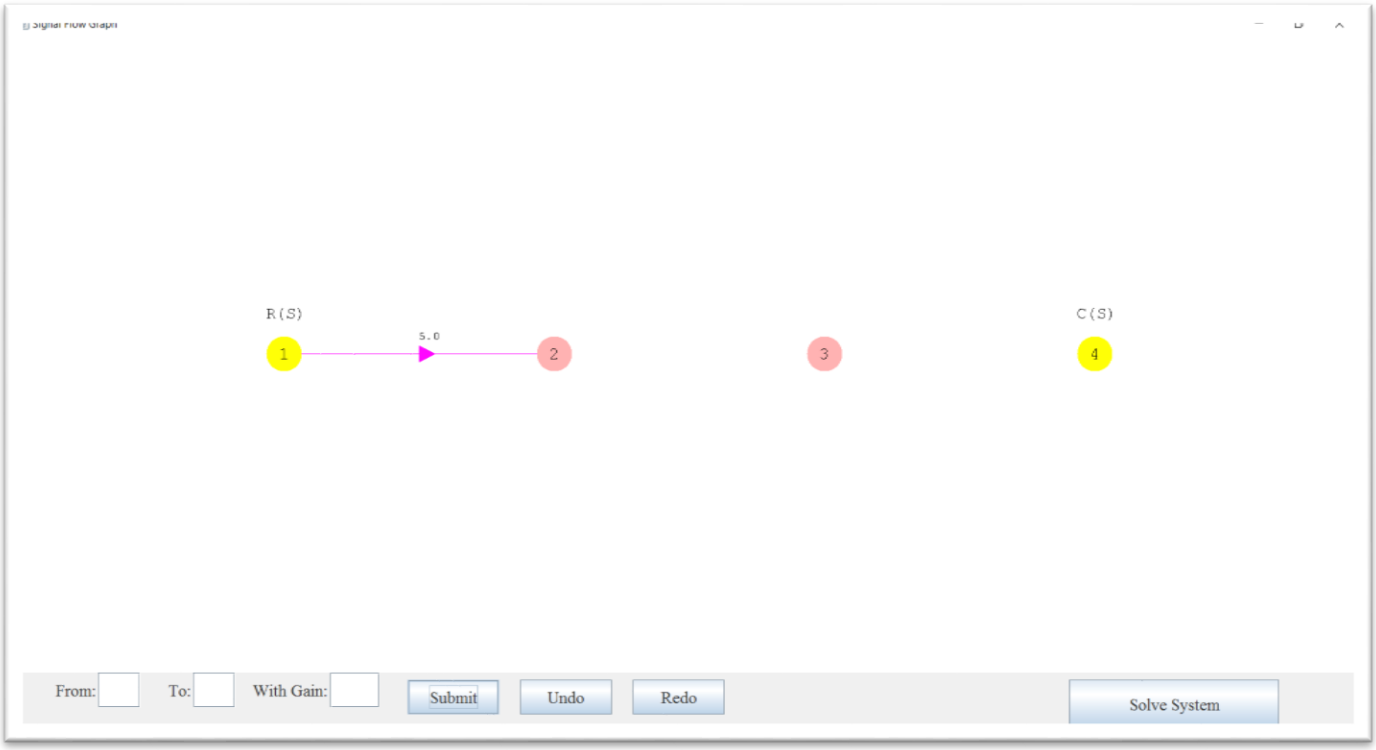


- *Input nodes can't have a feedback branch back to them, on trying to add a feedback branch an error message is generated.*



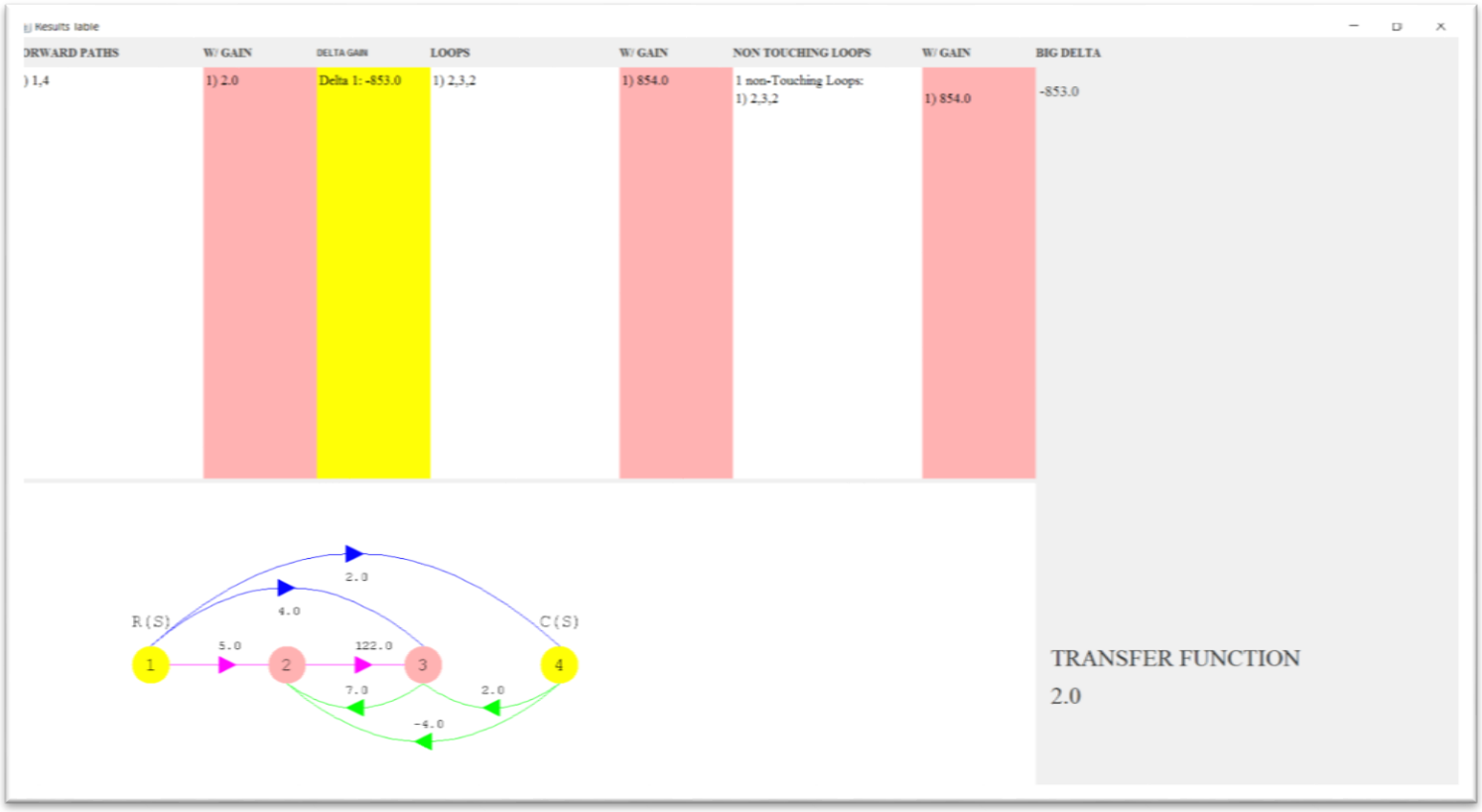
3- User can add more than one branch between two specified node, their summation is the new gain of the branch between the two nodes.
Negative numbers are included.
User can either undo or redo an addition to a given branch and maintain the previous value.





After adding another branch between nodes 1 and 2 with a gain equals to 3.

4- After specifying the whole signal flow graph, User presses the “Solve System Button” for a full answer.
User will be redirected to a new window, with a tabular form of the full information.



Tabular view consists of: 1- Forward Paths, 2- Forward Paths’ gain, 3- Delta of given Forward Paths, 3- Loops, 4- Loops Gain, 5- Non-touching Loops, 6- Non touching loops gains, 6- Big delta.