SIGNAL FLOW GRAPH REPORT

CS232 (Control System)

Hossam Fawzy Elsafty (24)
Marwan Morsy(72)

Including User Guide

TABLE OF ONTENTS

PROBLEM STATEMENT

- It is required to implement a graphical user interface for the Signal flow graph representation of the system.
- The input is that total number of nodes and numeric branches gains.
- The graphical user interface includes

1. the signal flow graph showing nodes, branches, gains, …
2. List of all forward paths, individual loops, all combination of *n* non-touching loops.
3. The values of Δ , Δ_1 , …, Δ_m where *m* is number of forward paths.
4. Overall system transfer function.

MAIN FEATURES

Our application has many features , we will represent them in following :

User-Friendly:

1-Our program it very easy to use , we don’t make restriction on user to enter the number of nodes , he can add node as he wish , it depend on graph representation we use dynamic data structure .

2-he can named the node with any string or number to deal with this name .

3- in GUI he can drag the node to arrange the style of representation as he wish .

4-we make user enter only edges and gain and to get transfer function by click on button , then we show him the result and all forward path and cycles and delta and non-touch cycle to make sure for their steps .

Object Oriented :

We use the principle of object oriented to make our application : simplicity , modularity , modifiability ,re-usability and maintainability.

And we divide the code on several modules and we make encapsulated on all module .

DATASTRUCTURE

We use many data Structure to make our program execute in small time and don't need a lot of memory .

Graph

We used this data structure to store the node and edge and gain of the system .when the user add edge we create the nodes with the names that user entered , then we connected between them with this edge and stored the weight of this gain . we represent the graph using hash table for every node and store their adjacent node in hash table.

HashMap

We used this data structure to represent the nodes in graph because in this hash map I can search on node in O(1) , and ever node has inner hash map to store their adjacent node in hash table , so we can search on node and get all adjacent node in O(1) , we can see this as table 2D of Hashing .

LinkedList

We use this data structure to iterate on all adjacent node to make the algorithm of DFS and to represent forward paths and cycle that stored in Linked List ,to make iteration is easy.

Stack

I use the idea on this data structure as recursive function , we call the function till the stop case is checked , the idea is Last input is first output , and this the idea of recursion , this idea is very important for DFS algorithm.

MAIN MODULES

We have 3 main module (search ,DFScycle,Calculate):

Search:

This module to search on forward paths and stored the paths and their gains in linked list .

DFS cycle:

This module to search on all cycle paths and stored the paths and their gains in linked list .

Calculate:

This module divide in several module because the formula calculate by several step, there is module for calculate delta ,and another module for calculate delta of forward paths , and module to get the final result of transfer function .

ALGORITHM USED

Our main algorithm is Depth-First-search to traverser on graph .

Depth-first search (**DFS**) is an algorithm for traversing or searching tree or graph data structures. One starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before backtracking.

Pseudocode

```
procedure DFS(G,v):
    label v as discovered
    for all edges from v to winG.adjacentEdges(v) do
        if vertex w is not labeled as discovered then
            recursively call DFS(G,w)
```

To search on forward paths we edit the algorithm by making condition that the adjacent of node contain the end node ,then we save this path and change the check of the path is unvisited to search on all paths.

To search on cycle paths we edit the algorithm by making condition that the adjacent of node contain the node that visited before that mean we are in loop ,then we save this path and change the check of the path is unvisited to search on all cycle paths.

To calculate the transfer function :

First ,wecalculate Delta by iterate on all cycle paths and sum the gain of non-touch and calculate the formla : delta =1-(sum of gain of individual loops)+(sum of gains of combination of 2 non touch loop)….

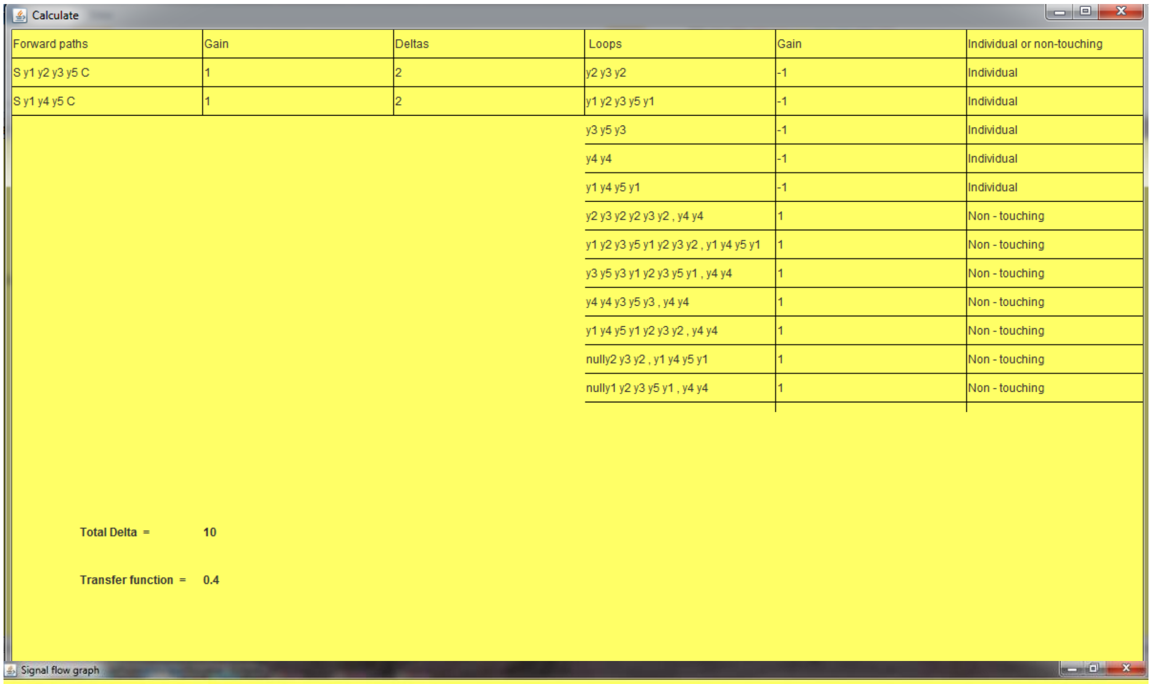
Second ,we calculate delta for all forward paths by iterate all forward paths and cycle to check the non-touch cycle with all forward paths

And make formla :delta(k)=the delta for that part of SGF that is non touching the Kth forward path(1-Lk).

Finally we calculate the transfer function

$$T = \frac{C(s)}{R(s)} = \frac{\sum_{i=1}^{n} P_i \Delta_i}{\Delta}$$

SAMPLE RUN

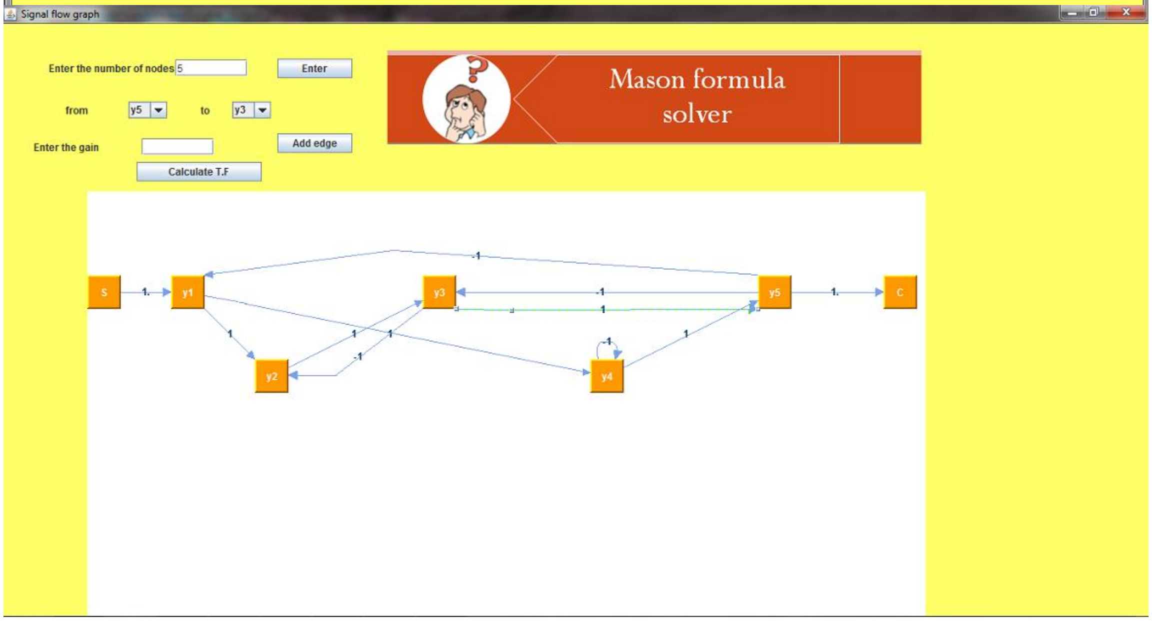


Calculate

Forward paths	Gain	Deltas	Loops	Gain	Individual or non-touching
S y1 y2 y3 y5 C	1	2	y2 y3 y2	-1	Individual
S y1 y4 y5 C	1	2	y1 y2 y3 y5 y1	-1	Individual
			y3 y5 y3	-1	Individual
			y4 y4	-1	Individual
			y1 y4 y5 y1	-1	Individual
			y2 y3 y2 y2 y3 y2 , y4 y4	1	Non - touching
			y1 y2 y3 y5 y1 y2 y3 y2 , y1 y4 y5 y1	1	Non - touching
			y3 y5 y3 y1 y2 y3 y5 y1 , y4 y4	1	Non - touching
			y4 y4 y3 y5 y3 , y4 y4	1	Non - touching
			y1 y4 y5 y1 y2 y3 y2 , y4 y4	1	Non - touching
			nully2 y3 y2 , y1 y4 y5 y1	1	Non - touching
			nully1 y2 y3 y5 y1 , y4 y4	1	Non - touching

Total Delta = 10

Transfer function = 0.4

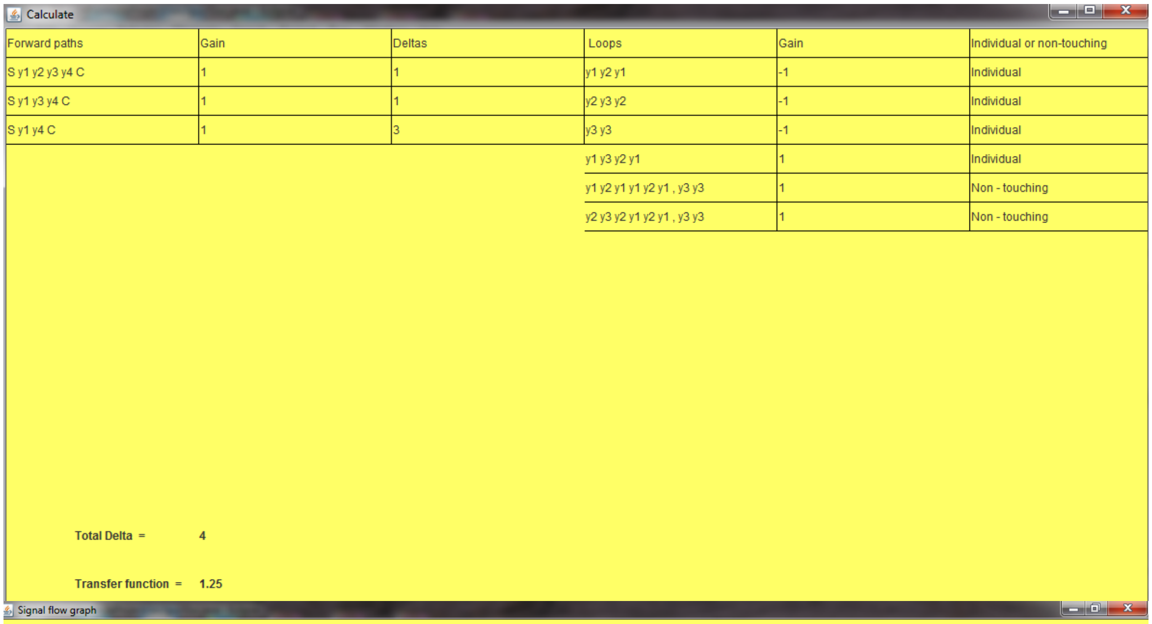


Signal flow graph

Enter the number of nodes 5

from y5 to y3

Enter the gain

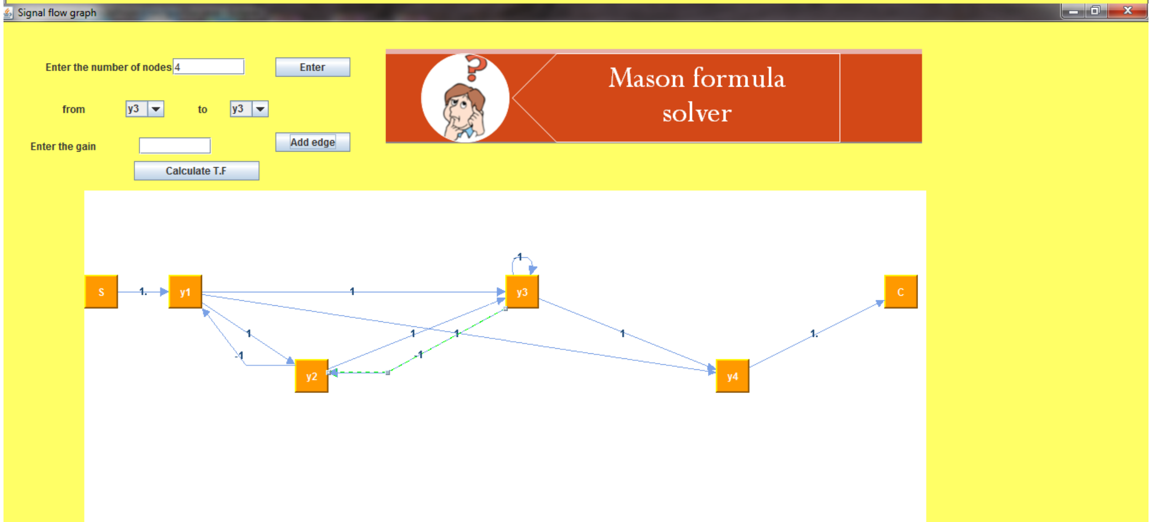


Calculate

Forward paths	Gain	Deltas	Loops	Gain	Individual or non-touching
S y1 y2 y3 y4 C	1	1	y1 y2 y1	-1	Individual
S y1 y3 y4 C	1	1	y2 y3 y2	-1	Individual
S y1 y4 C	1	3	y3 y3	-1	Individual
			y1 y3 y2 y1	1	Individual
			y1 y2 y1 y1 y2 y1 , y3 y3	1	Non - touching
			y2 y3 y2 y1 y2 y1 , y3 y3	1	Non - touching

Total Delta = 4

Transfer function = 1.25

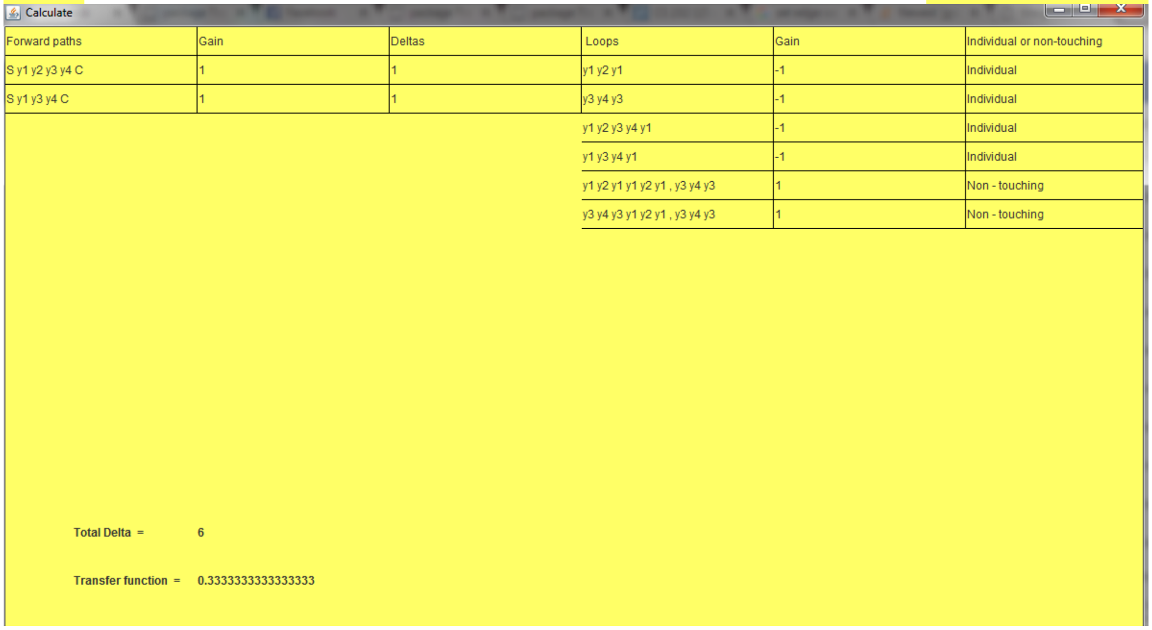


Signal flow graph

Enter the number of nodes 4

from y3 to y3

Enter the gain

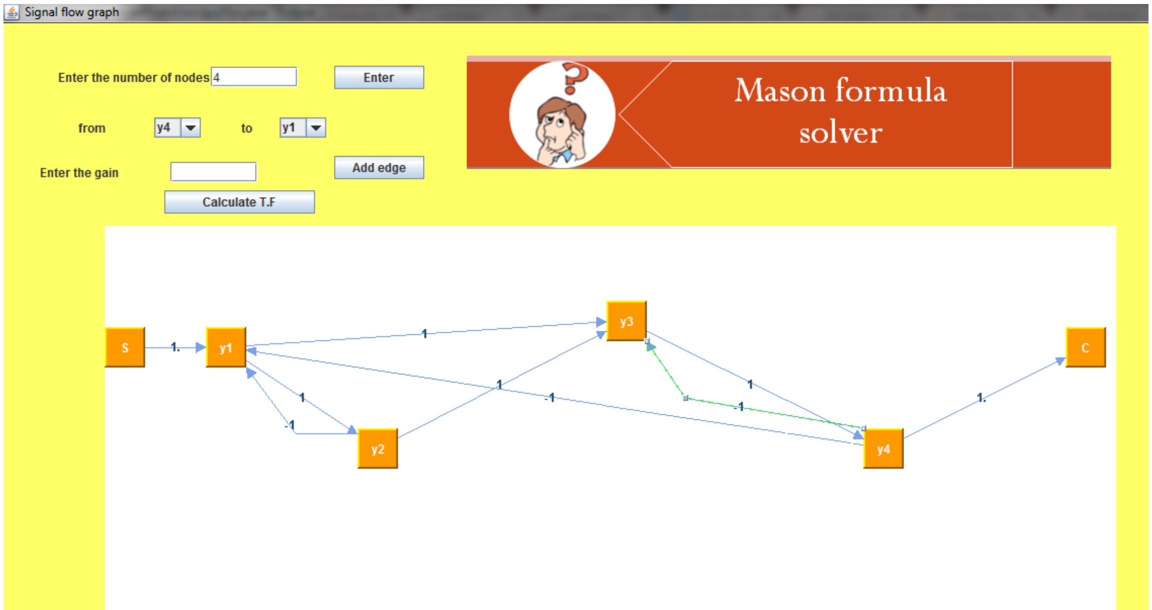


Calculate

Forward paths	Gain	Deltas	Loops	Gain	Individual or non-touching
S y1 y2 y3 y4 C	1	1	y1 y2 y1	-1	Individual
S y1 y3 y4 C	1	1	y3 y4 y3	-1	Individual
			y1 y2 y3 y4 y1	-1	Individual
			y1 y3 y4 y1	-1	Individual
			y1 y2 y1 y1 y2 y1 , y3 y4 y3	1	Non - touching
			y3 y4 y3 y1 y2 y1 , y3 y4 y3	1	Non - touching

Total Delta = 6

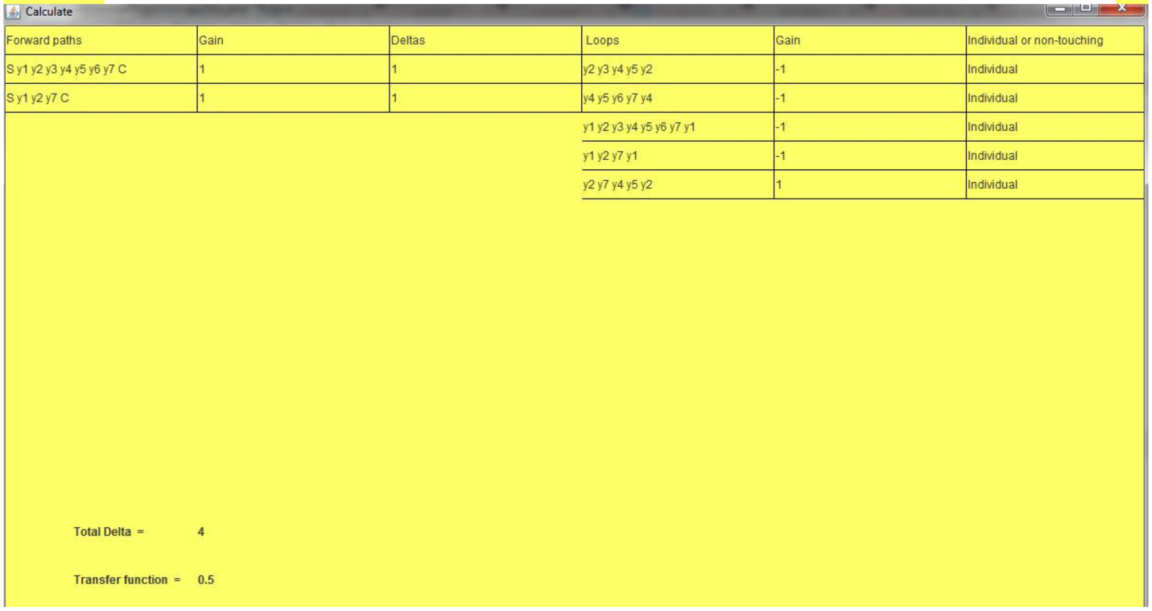
Transfer function = 0.3333333333333333



Enter the number of nodes 4

from y4 to y1

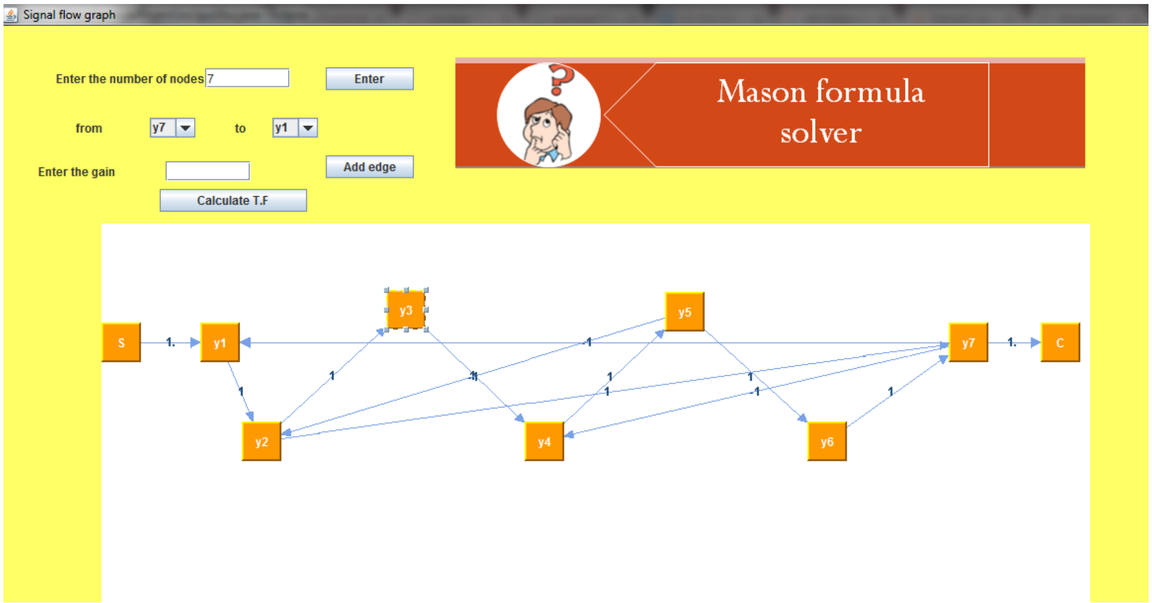
Enter the gain



Forward paths	Gain	Deltas	Loops	Gain	Individual or non-touching
S y1 y2 y3 y4 y5 y6 y7 C	1	1	y2 y3 y4 y5 y2	-1	Individual
S y1 y2 y7 C	1	1	y4 y5 y6 y7 y4	-1	Individual
			y1 y2 y3 y4 y5 y6 y7 y1	-1	Individual
			y1 y2 y7 y1	-1	Individual
			y2 y7 y4 y5 y2	1	Individual

Total Delta = 4

Transfer function = 0.5



Enter the number of nodes 7

from y7 to y1

Enter the gain

USER GUIDE

1. First the user enters the number of nodes will be in the graph excluding the input node "S" and the output node "C" by clicking "Enter" button.
2. Then in the combo box will appear the node names from y_1 to y_n the user select the start of the edge and its end then enter the gain by clicking on the "add edge" button .
3. After that when the user finishes entering the input and clicks on "calculate T.F " button.
4. Finally another window will appear .This window contains a table of all forward paths, individual loops, all combination of *n* non-touching loops. and the values of Δ , Δ_1 , …, Δ_m deltas. At the end of it the total delta and Overall system transfer function appear .