Alexandria University,

Faculty of Engineering,

Computer and Systems Engineering Dept.

CS241: Linear Control System

Signal Flow Graph

Solver

Names:

1. Ziad Hisham Ali Abdel-Hameed. (19)
2. Youssef Ahmed Sayed Gabriel. (61)
3. Problem Statement:

It is required to design and implement solver for signal flow graph to get the overall transfer function of linear control system through a definite equation:

**C(s)/R(s) = Σ** Pi **Δ**i / **Δ**

, Where pi: the path of forward path from input node to output node

**Δ**i : the untouched loops with the forward path

1 – (path 1 untouched loops) + (path 2 untouched loops)

– (path 3 untouched loops) + …….

**Δ**: (path 1 untouched loops) + (path 2 untouched loops)

+ (path 3 untouched loops) + …...

i: index of forward path 1, 2, 3,…, n

1. Main Feature:

* It was implemented using JavaScript.
* The signal flow graph is drawn using “Vis Library” through: Add Node & Add Edge
* **Extra Features:** edit Node, edit Edge, Remove Selected, Zoom In,

Zoom out, Shift the graph on the four directions,

Reset button to delete the drawn graph,

Evaluate Button to get the results &

SideView bar to view Home, About the application and

How to use the app.

1. Data Structures:

**Stack**: used to get the Forward Paths present in the graph using the

functions “DFS” & “CalculateForwardPath”

& the loops present in graph using the function “DFSLoops”

**1D Array:** used to fill the Nodes & the Edges of the graph using the

Functions “FillEdges” & “FillNodes”

**2D Array:** used to get the untouched loops present in the graph using the

Function “CalculateUntouched”

& the delta for each forward path present in the graph using

the function “CalculateDeltaIndex”

1. Main Modules:

* **HTML**

For The User Interface, divide the page to groups of (div) present in

“index.html”

* **CSS**

For each Class in the “div” it’s style present in “app.css”

* **JavaScript**

For The Main Algorithm of the app present in “app.js”

& Their Connection with Vis Library and the graph present in

“VisLogic.js”

* FillNodes
* FillEdges
* CalculateForwardPath
* DFSLoops
* CalculateUntouched
* CalculateDeltaIndex

1. Algorithms :

* **To Get Forward Paths**

Using Stack, pushing the Input Node then Enter While loop as

Stack Length != 0 then pop from the stack and get the popped Node

and iterate on it to get all the Nodes that popped Node go to and push

them in the Stack if they aren’t visited before, Eventually The

Forward Path Gain is got …

Through “DFS” & “CalculateForwardPath” Functions

* **To Get The Loops**

Using Stack, pushing the Input Node then Enter While loop as

Stack Length != 0 then pop from the stack and get the popped Node

and iterate on it to get all the Nodes that popped Node go to and push

them in the Stack if they aren’t visited before, Then removing all the

duplicates loops, Eventually All loops are got …

Through “DFSLoops” Function

* **To Get The Untouched Loops**

Using 2D Array, pushing the all loops in the Zero & First Row

Iterate in Zero Row and Iterate in the Last Row Present in 2D Array

By Comparing their Paths Together ,if they are untouched in any node,

then push their product in the next Row, Eventually 2D Array have in

their First Row 1 untouched Loops, Second Row 2 untouched Loops …

Through “CalculateUntouched”, “GetLoop” & “IsIntersected” Functions

* **To Get The Delta For Each Forward Path**

Using 2D Array, Iterate in the all present Forward Paths, Compare

Each Forward Path with The 2D Array that contain the all loops and

Iterate in this Array Row By Row , if the are untouched in any Node,

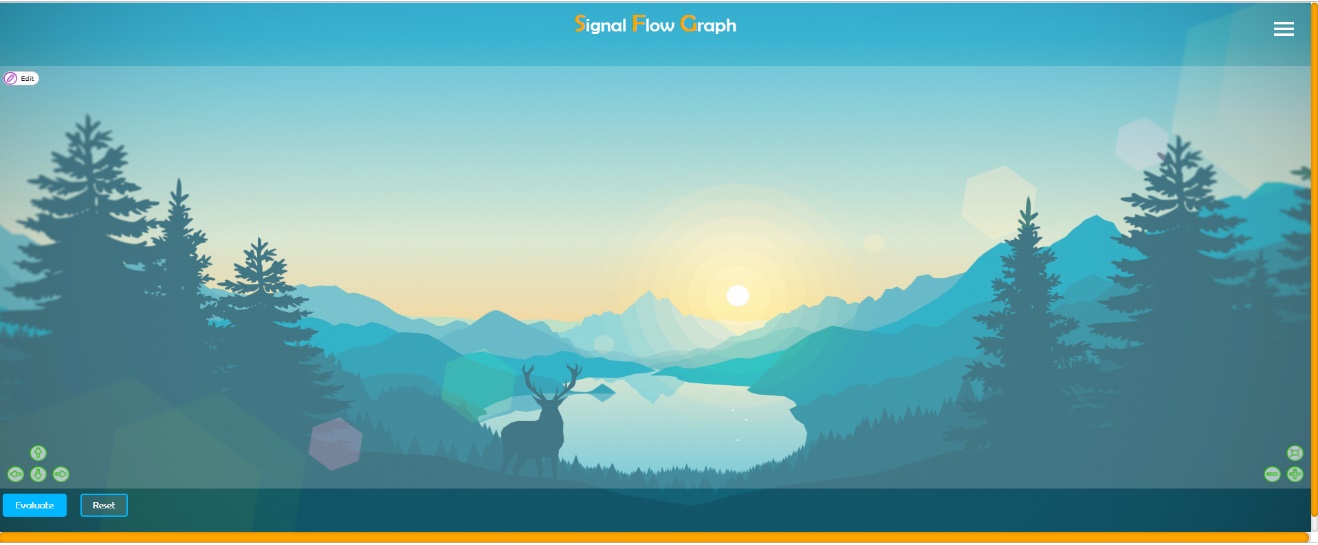
Then push in the Next Row This Untouched Loop, Eventually 2D Array

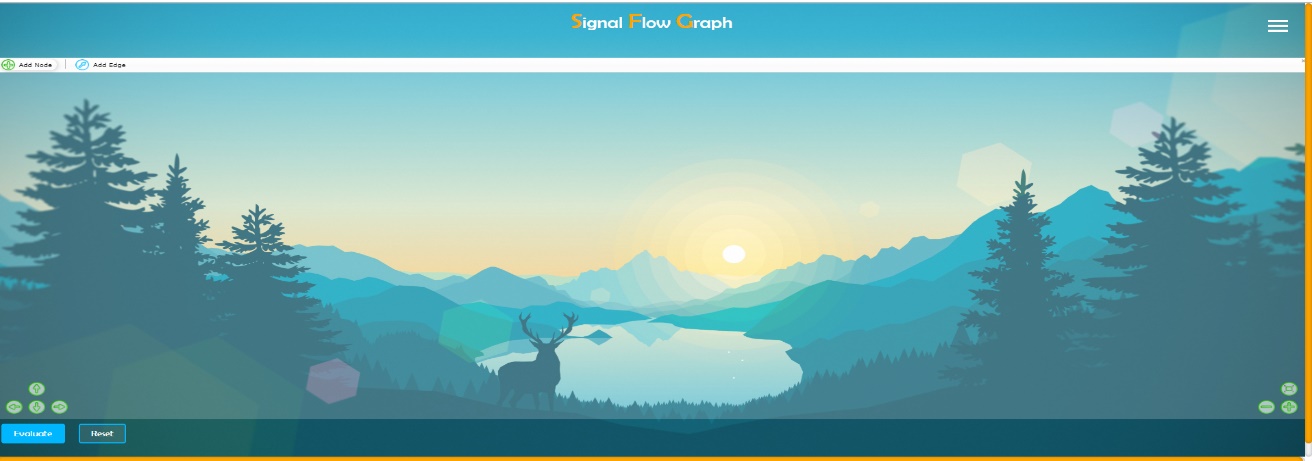
Contain in the First Row the 1 untouched loop with this Forward Path,

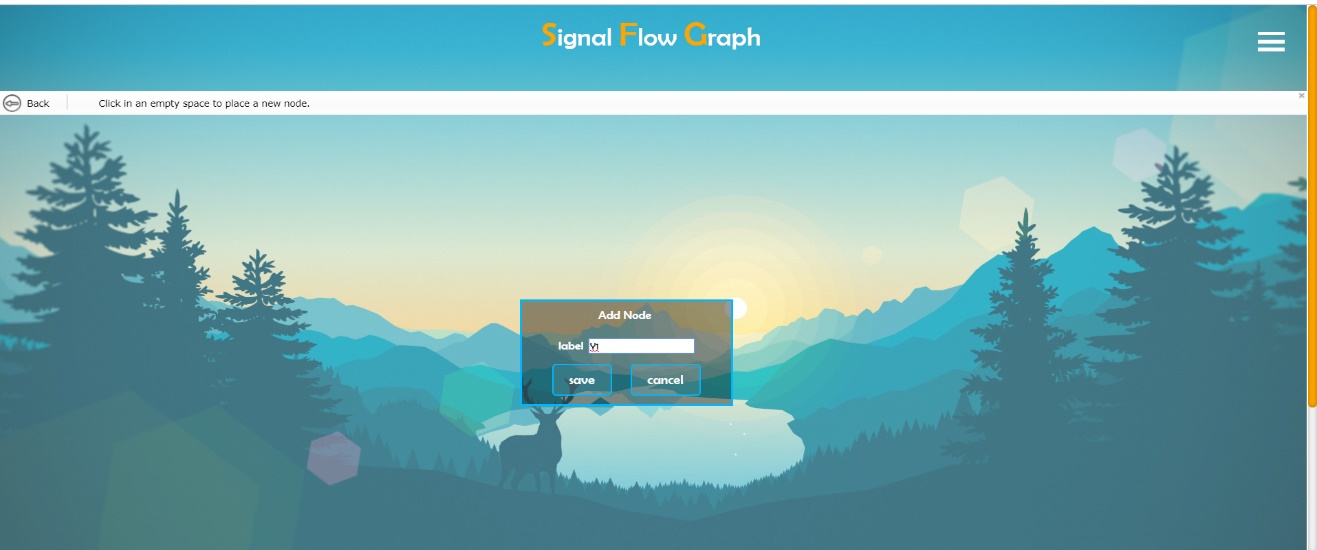
Second Row 2 Untouched With this Forward Path ……

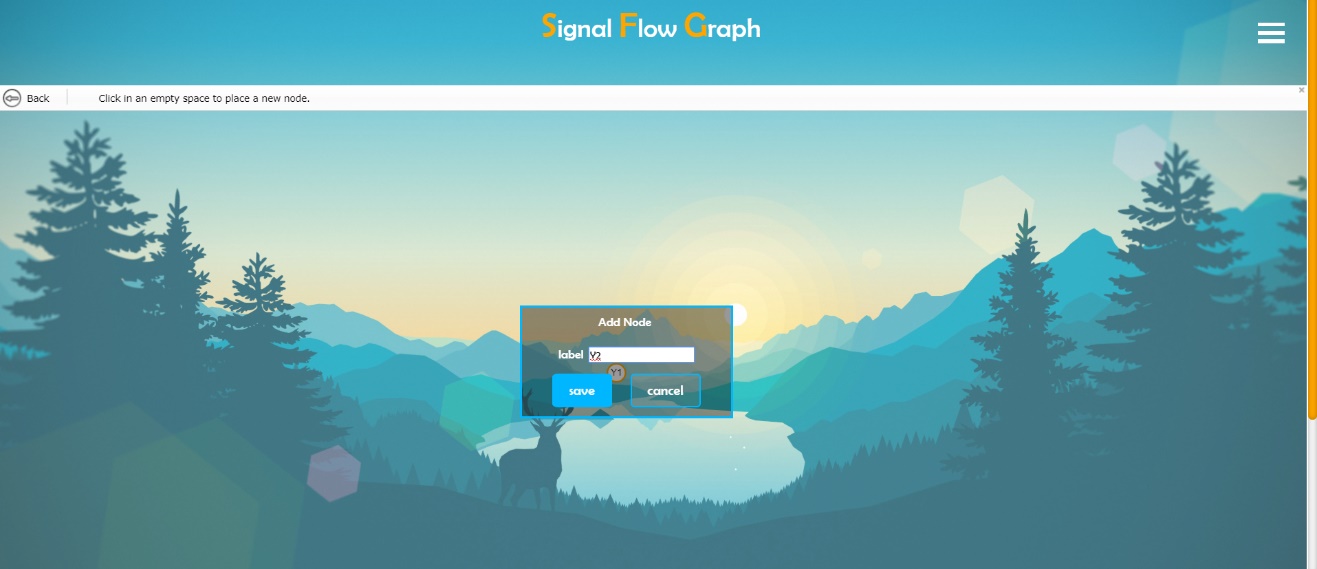
Through “CalculateDeltaIndex” Function

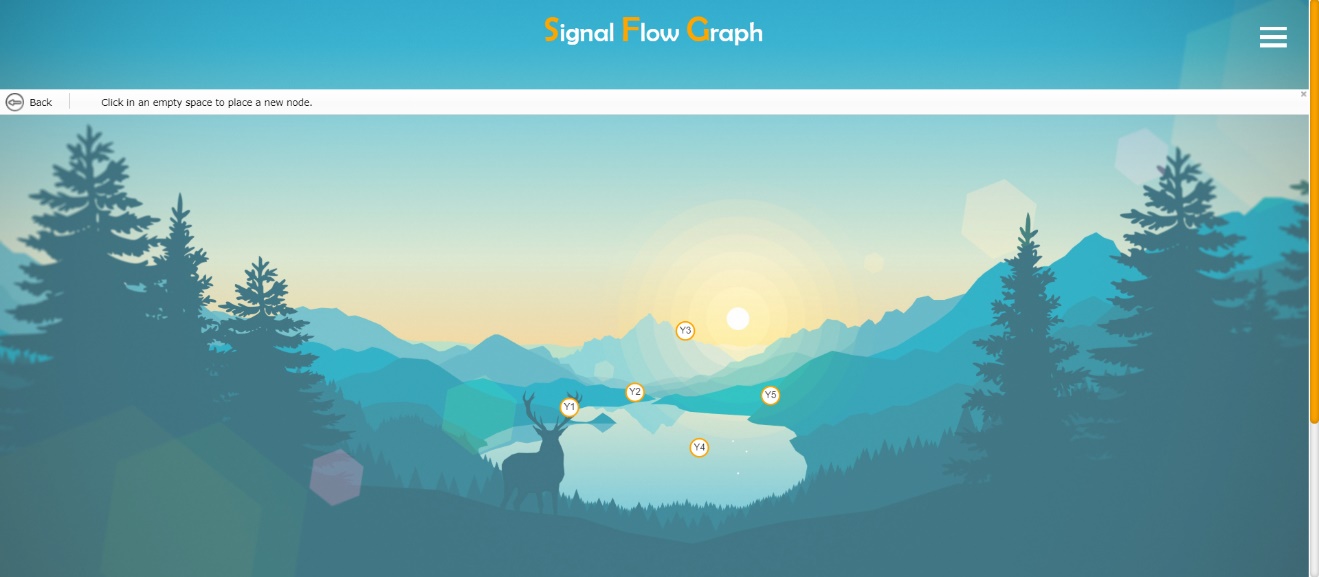
1. Sample Runs:



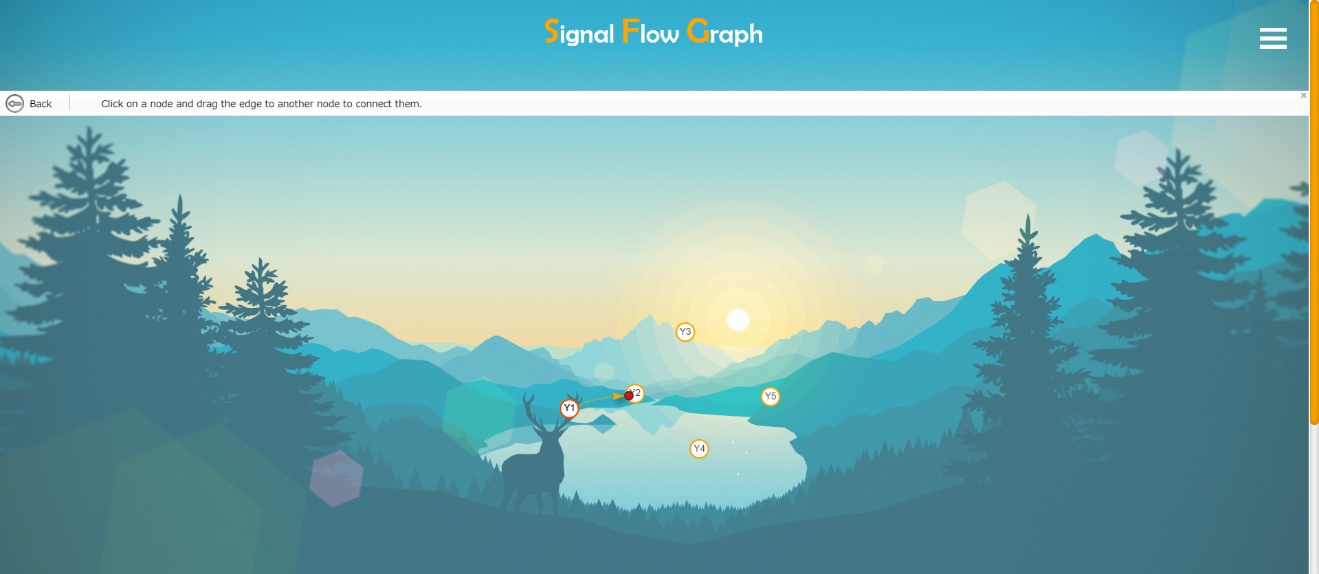


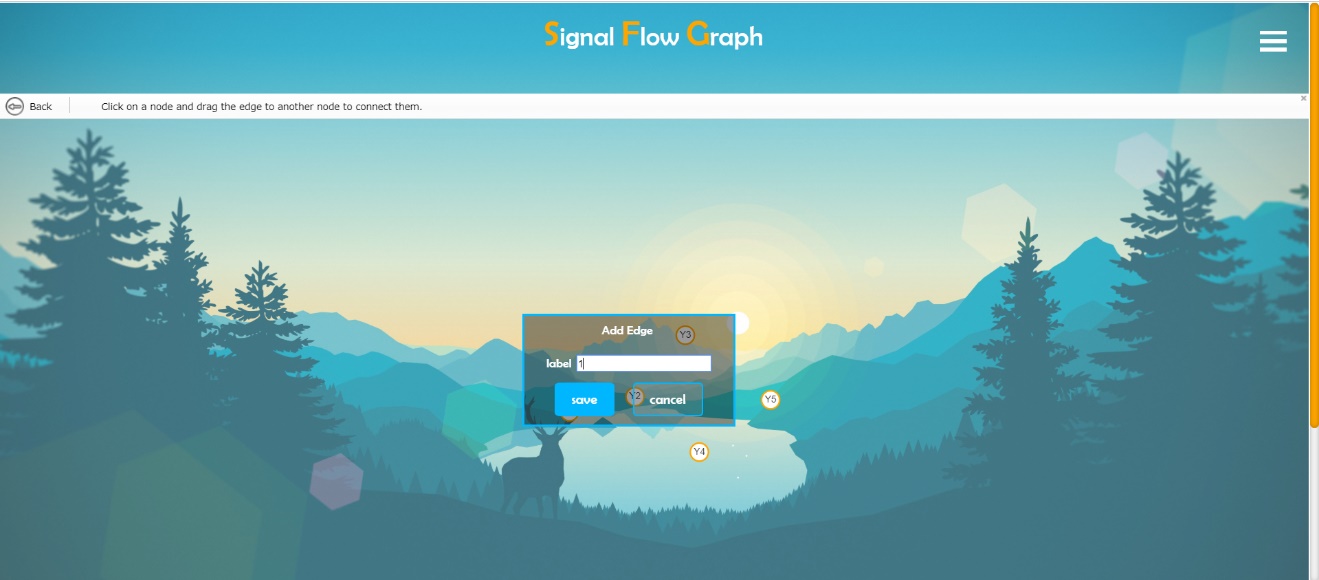


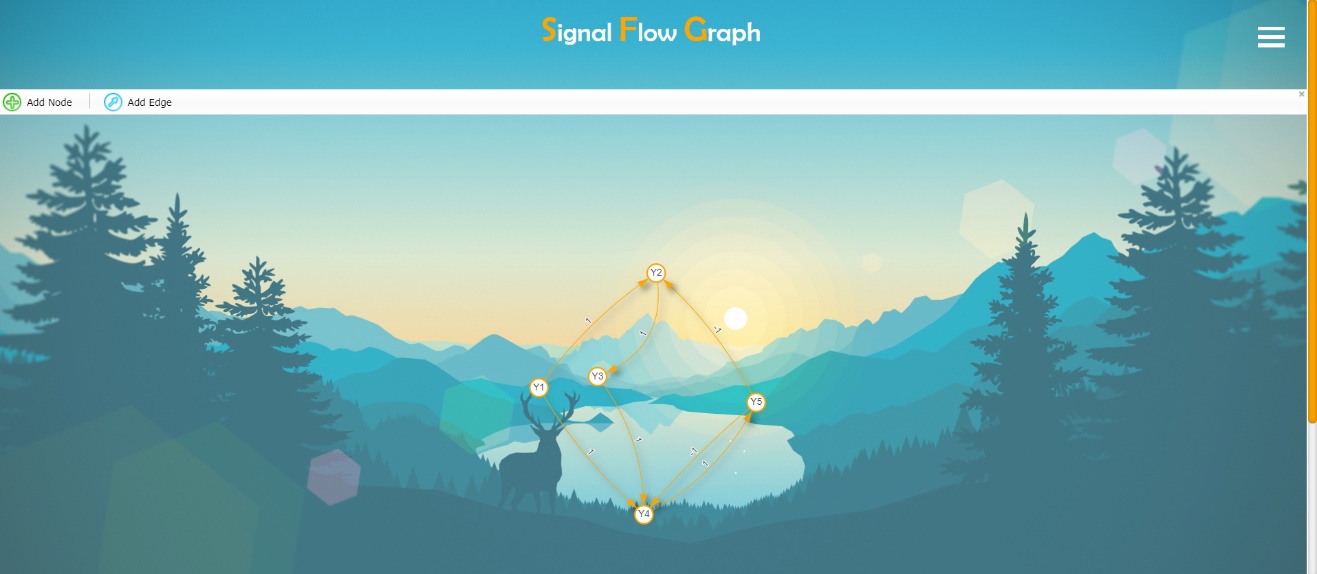


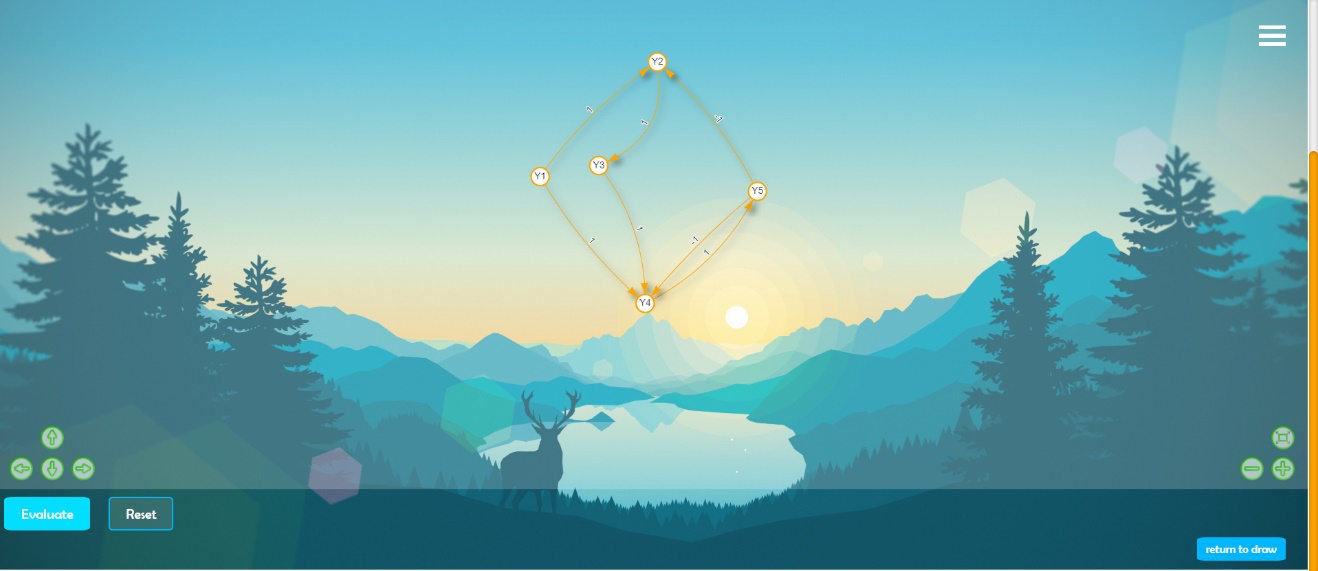


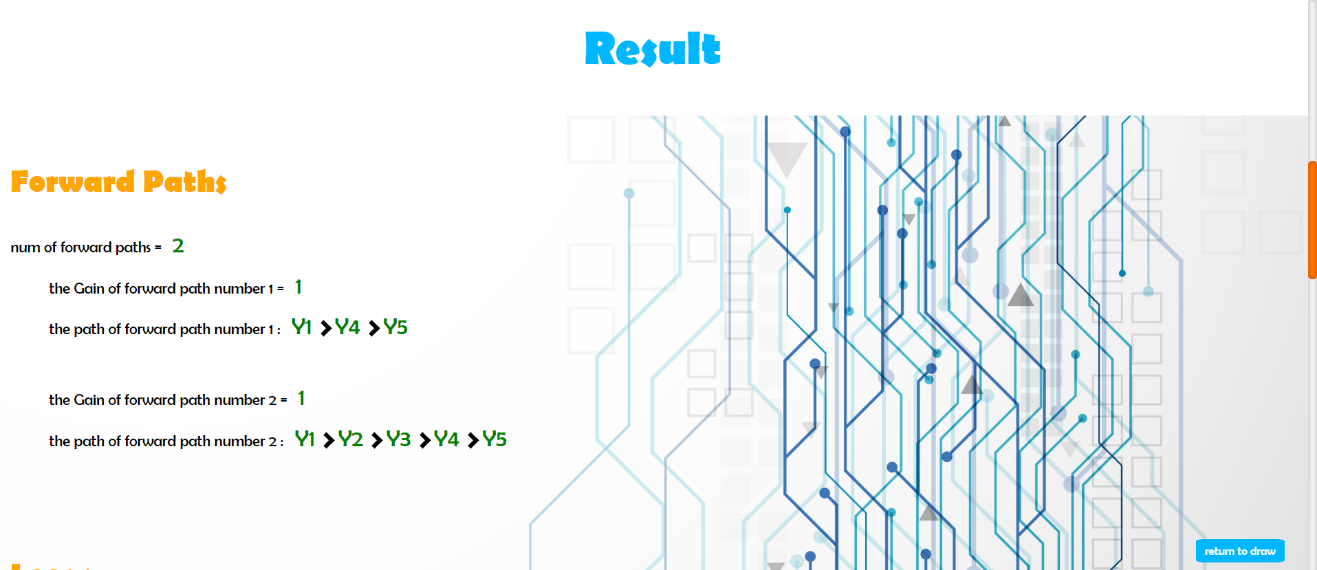






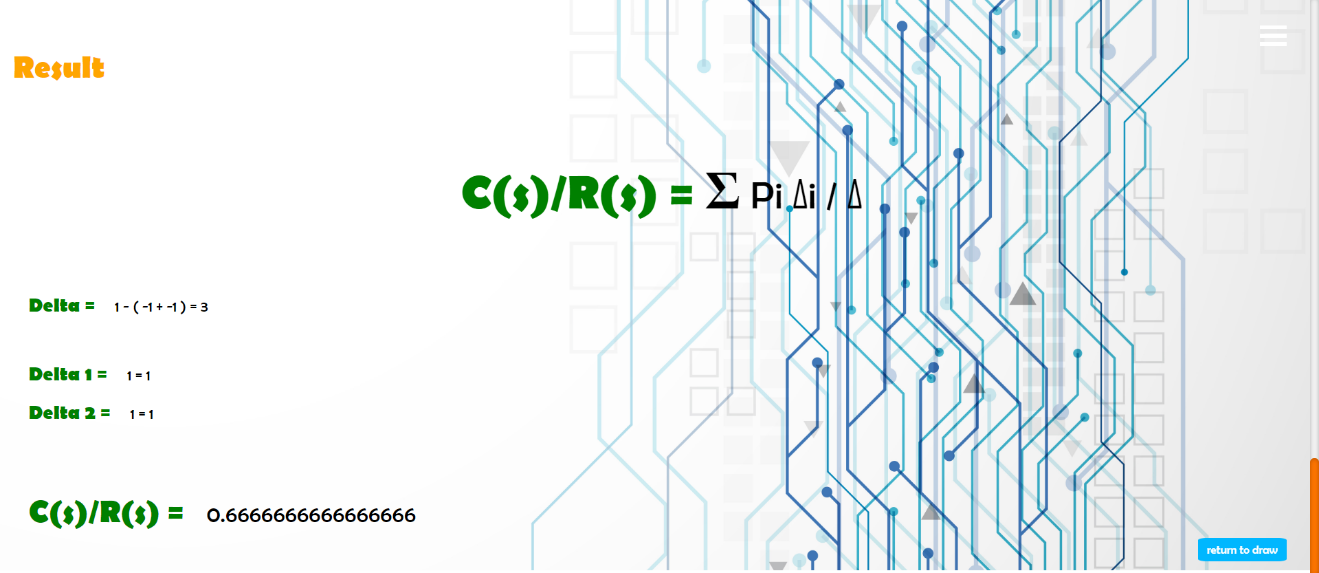


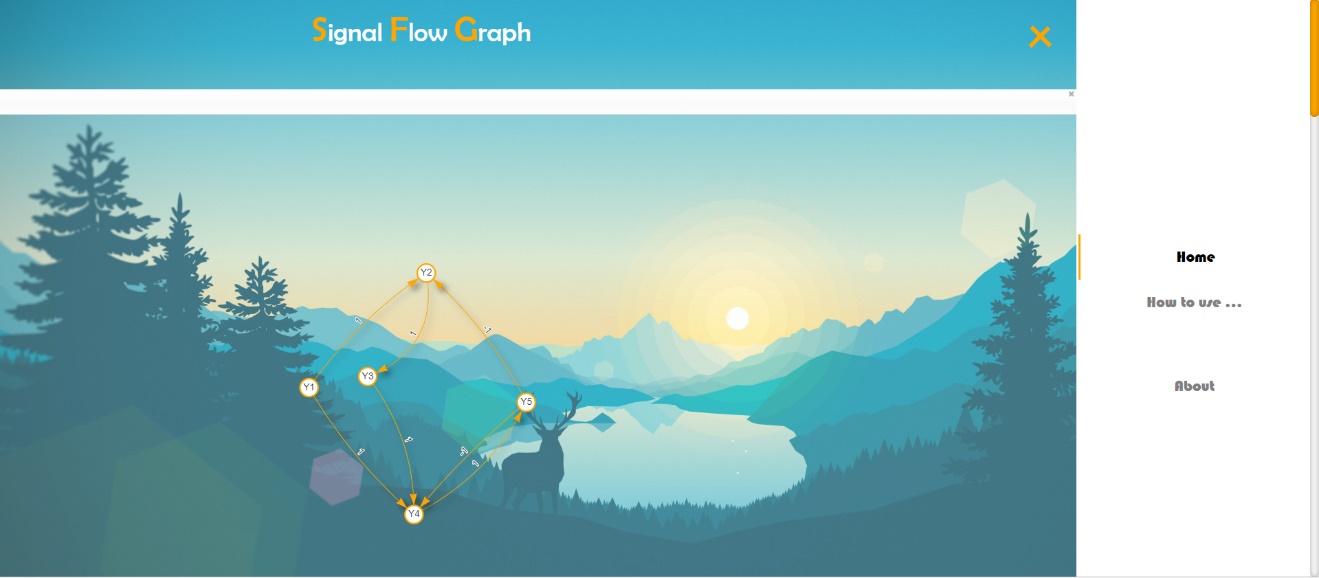












1. Simple User Guide:

* To open the app from source code open “index.html” file in any browser.
* Add The Nodes Of the graph through “Add Node” Button and give them labels.
* Add The Edges Of the graph through “Add Edges” Button and give them labels.
* Click “Evaluate” Button.
* The Overall Transfer Function Result will be calculated.
* To Draw another Graph, Click “return to draw” Button.
* Click On “Reset” Button And Repeat The previous steps …

1. Assumptions:

* The first node is the input node and the last added node is the output node.
* Nodes colors:
  + Input node: Orange with White border.
  + Output node: White with Black border.
  + Other nodes: White with Orange border.
* The label of an edge is its gain and the app is dealing with numerical values only.