Programming Assignment

Signal Flow Graphs &

Routh Stability Criterion

Part1

executables and source code:-

https://github.com/Mohamed-Mohamed-Ibrahim/Signal-Flow-Graph.git

1) Problem Statement

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 , ..., Δm where m is number of forward paths.
5. Overall system transfer function.

2) Main Features of the Program and Additional Options

1. Draw the signal flow graph
2. Assignment of values to the branches
3. Assignment of input and output node
4. Solve the signal flow graph to get the transfer function
5. Listing all forward paths, individual loops, all combination of n non-touching loops.

3) Data Structure

- **Lists (`ArrayList`)**: Used for representing the graph, paths, loops, non-touching loops, gains, and other data structures.
- **Stacks**: Utilized for backtracking and maintaining state during loop detection and combination processes.
- **Custom data structure (`MyPair`)**: Represents pairs of integers used to describe edges in the graph.

4) Main Modules

backend ->

- **Initialization (`init()`):** Initializes data structures and prepares the graph for analysis.
- **Finding Paths (`findAllPaths()`):** Finds all paths from the source to the destination node in the graph.
- **Finding Loops (`findLoopsGains()`):** Detects all loops within the graph and calculates their gains.

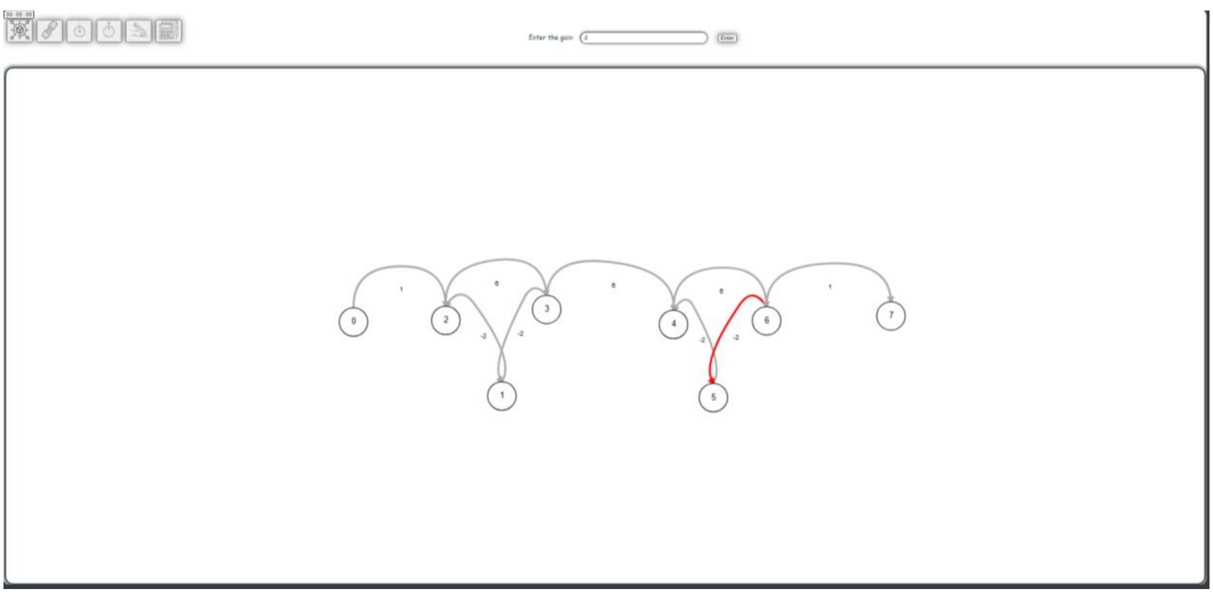
- **Combining Loops (`combine()`):** Combines non-touching loops to analyze higher-order loop interactions.
- **Computing Transfer Function (`overAllTransferFunction()`):** Computes the overall transfer function of the signal flow system.

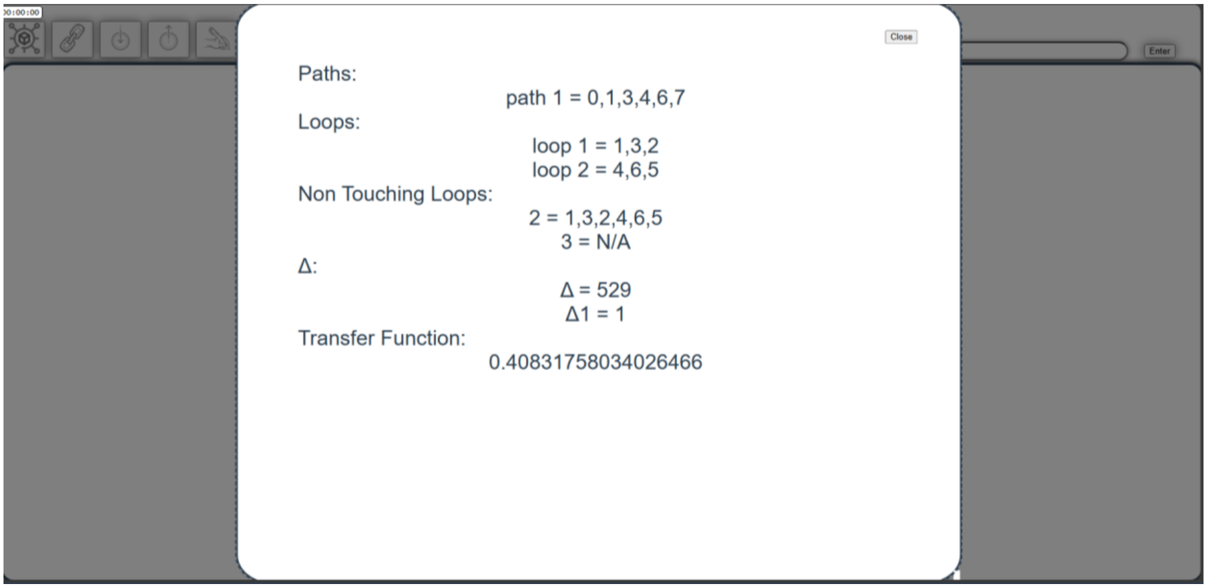
Front end-> This is the GUI of the program. Through it, we can draw the graph and send data to the backend.

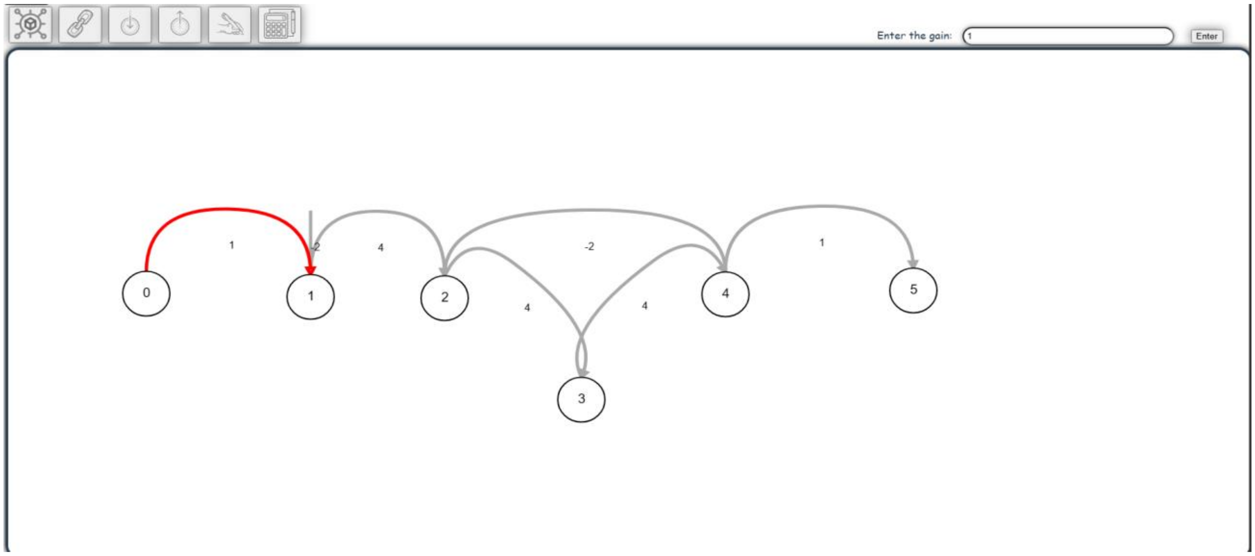
5) Algorithms Used

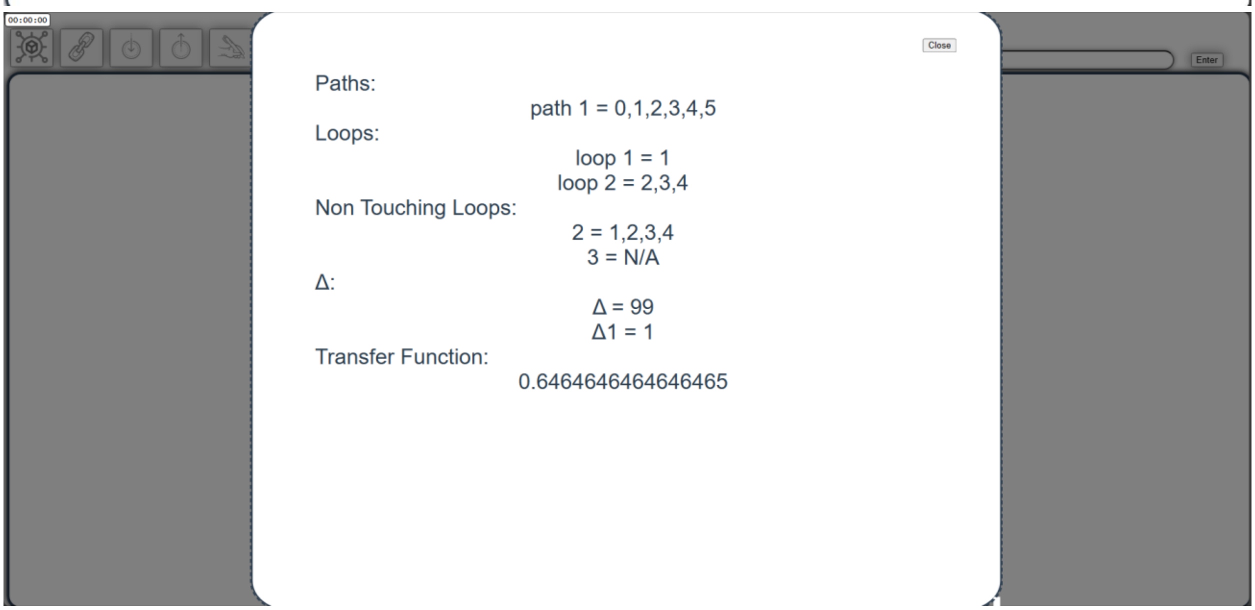
- **Depth-First Search (DFS):** Used for finding all paths in the graph and detecting loops.
- **Backtracking:** Utilized in combination with DFS for loop detection and non-touching loop combination.
- **Combinatorial Techniques:** Employed to combine non-touching loops efficiently.

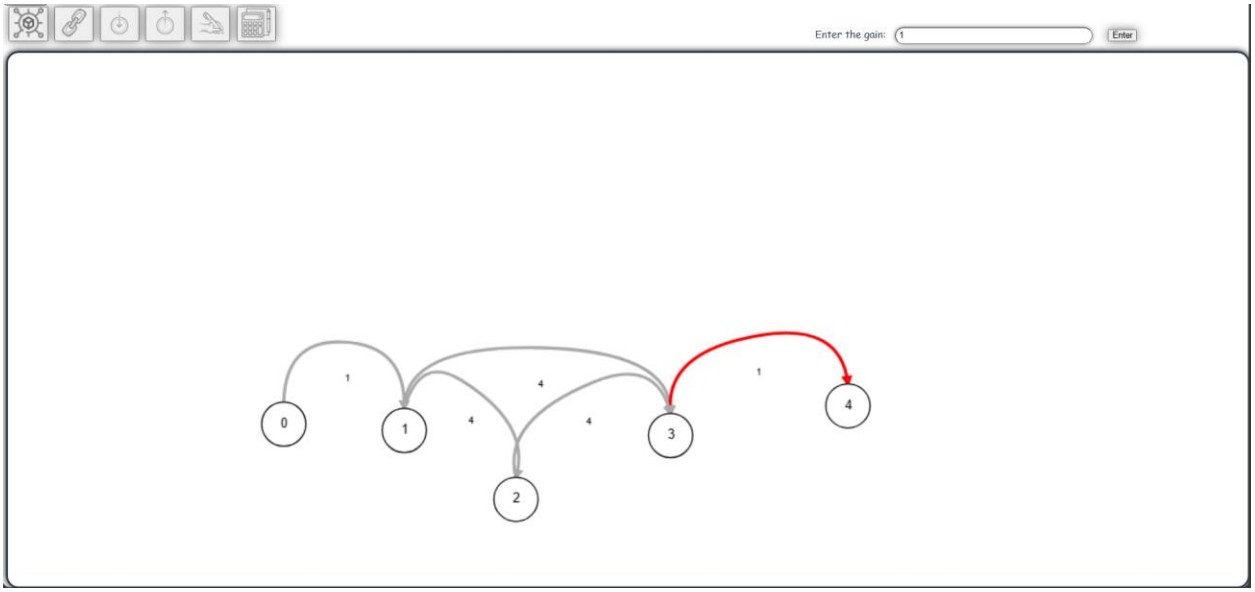
6) Sample Runs

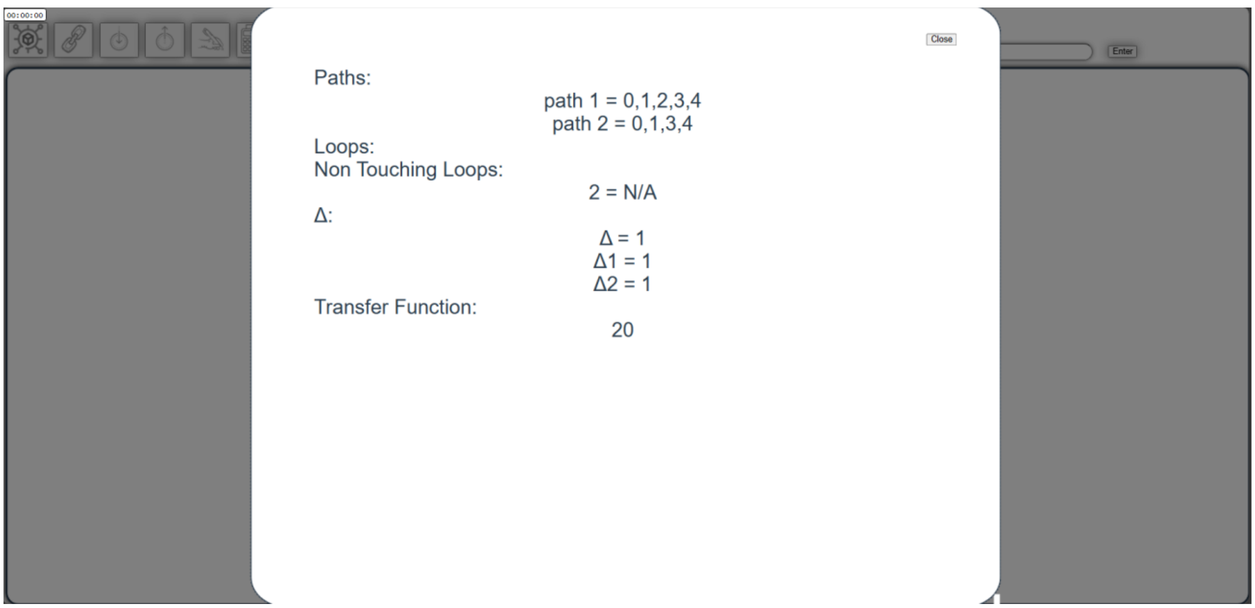


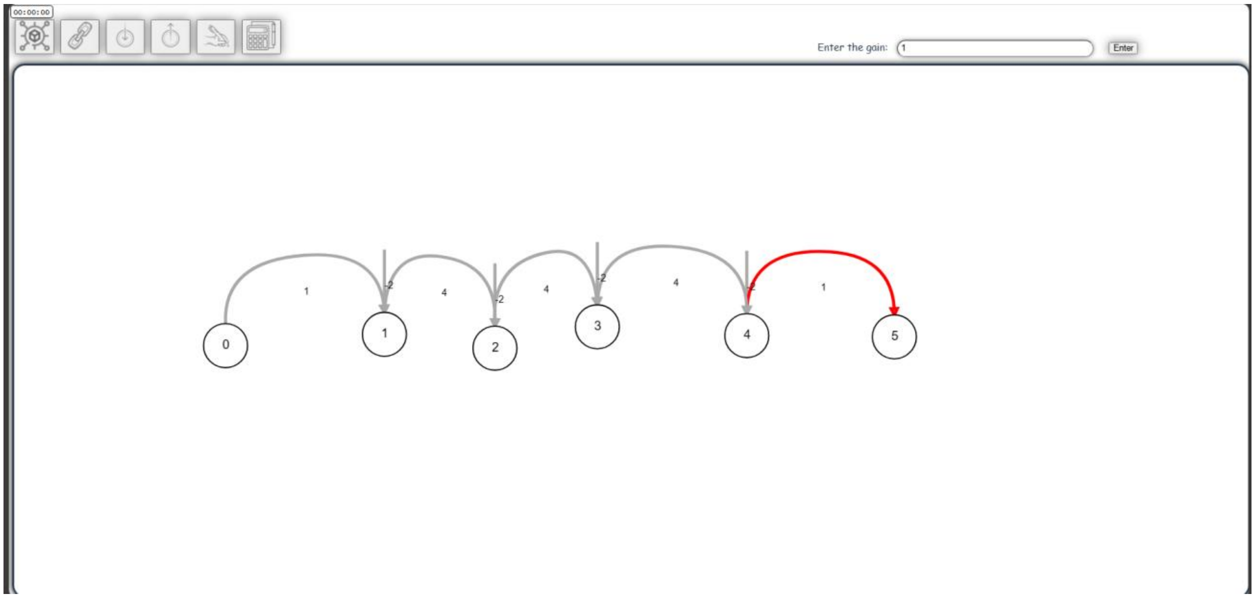

00:00:00
Close
Enter
Paths:
path 1 = 0,1,3,4,6,7
Loops:
loop 1 = 1,3,2
loop 2 = 4,6,5
Non Touching Loops:
2 = 1,3,2,4,6,5
3 = N/A
Δ:
Δ = 529
Δ1 = 1
Transfer Function:
0.40831758034026466

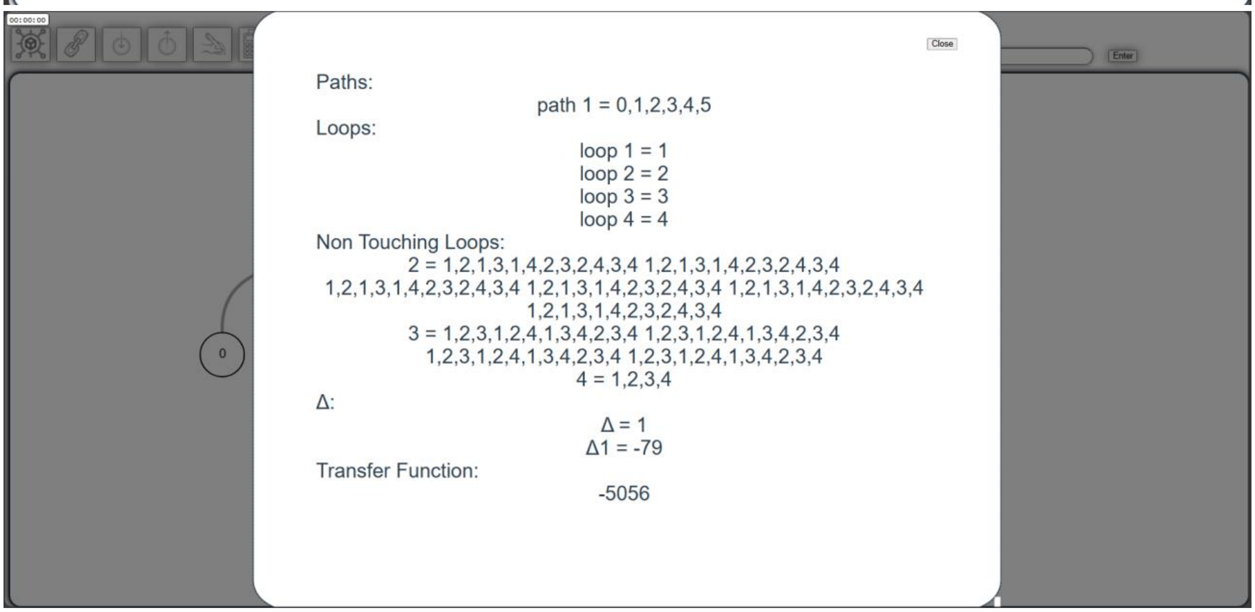

Enter the gain:
1
Enter
1
-2
4
-2
1
4
4
0
1
2
3
4
5

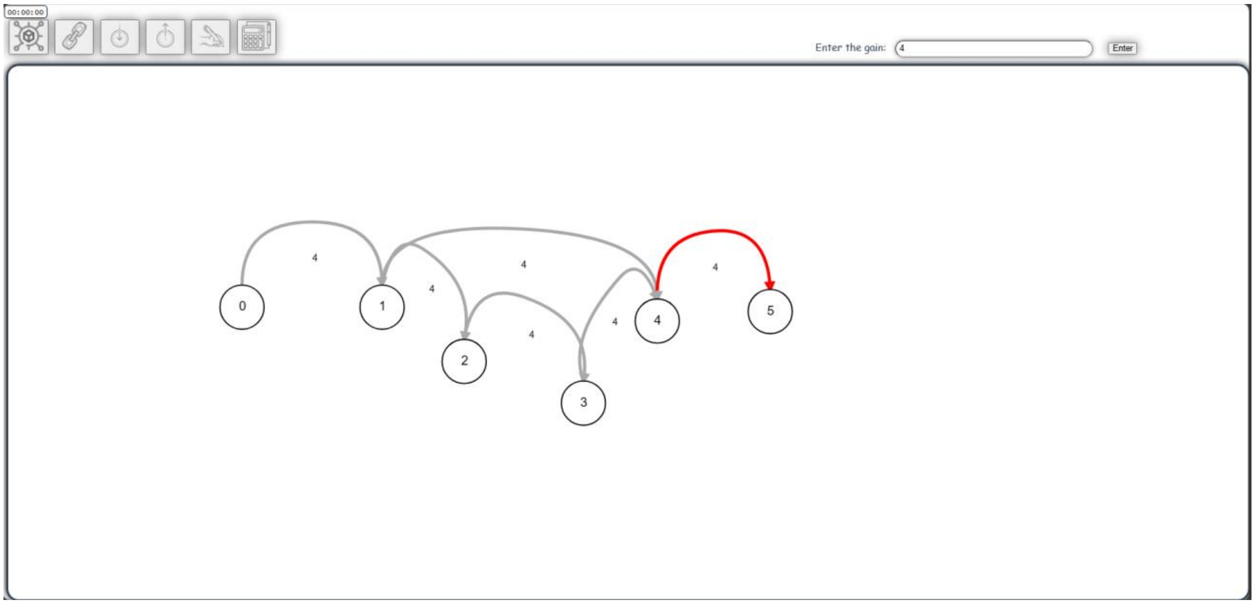

00:00:00
Close
Enter
Paths:
path 1 = 0,1,2,3,4,5
Loops:
loop 1 = 1
loop 2 = 2,3,4
Non Touching Loops:
2 = 1,2,3,4
3 = N/A
Δ:
Δ = 99
Δ1 = 1
Transfer Function:
0.6464646464646465

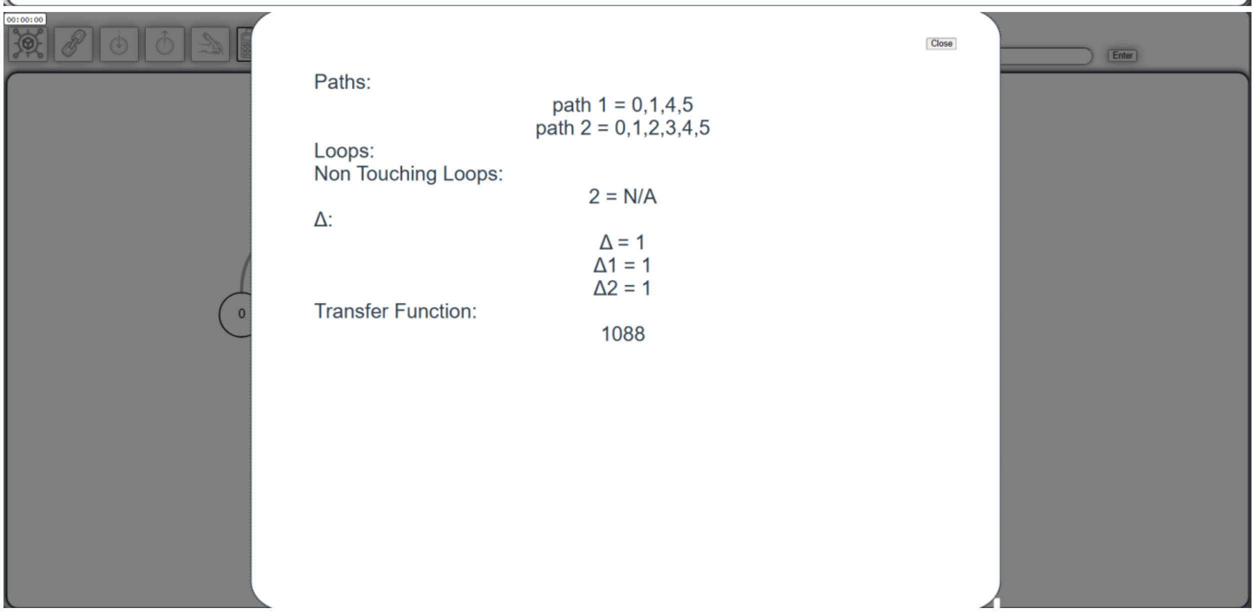

Enter the gain:
1
Enter
1
4
4
4
1
0
1
2
3
4


00:00:00
Close
Enter
Paths:
path 1 = 0,1,2,3,4
path 2 = 0,1,3,4
Loops:
Non Touching Loops:
2 = N/A
Δ:
Δ = 1
Δ1 = 1
Δ2 = 1
Transfer Function:
20


00:00:00
Enter the gain:
1
Enter
1
4
4
4
1
-2
-2
-2
-2
0
1
2
3
4
5


00:00:00
Close
Enter
Paths:
path 1 = 0,1,2,3,4,5
Loops:
loop 1 = 1
loop 2 = 2
loop 3 = 3
loop 4 = 4
Non Touching Loops:
2 = 1,2,1,3,1,4,2,3,2,4,3,4 1,2,1,3,1,4,2,3,2,4,3,4
1,2,1,3,1,4,2,3,2,4,3,4 1,2,1,3,1,4,2,3,2,4,3,4 1,2,1,3,1,4,2,3,2,4,3,4
1,2,1,3,1,4,2,3,2,4,3,4
3 = 1,2,3,1,2,4,1,3,4,2,3,4 1,2,3,1,2,4,1,3,4,2,3,4
1,2,3,1,2,4,1,3,4,2,3,4 1,2,3,1,2,4,1,3,4,2,3,4
4 = 1,2,3,4
Δ:
Δ = 1
Δ1 = -79
Transfer Function:
-5056
0


00:00:00
Enter the gain:
4
Enter
4
4
4
4
4
4
0
1
2
3
4
5


00:00:00
Close
Enter
Paths:
path 1 = 0,1,4,5
path 2 = 0,1,2,3,4,5
Loops:
Non Touching Loops:
2 = N/A
Δ:
Δ = 1
Δ1 = 1
Δ2 = 1
Transfer Function:
1088
0

7) Simple User Guide:

- you have to download frontend code and backend code.

- Verify the presence of Node.js on your system.

- Utilize the command "npm i serve" to install the requisite modules.

- Launch the front end using the command "npm run serve."

- after this you click on the link in the terminal and this window will appear:

- you can choose add nodes using this button:

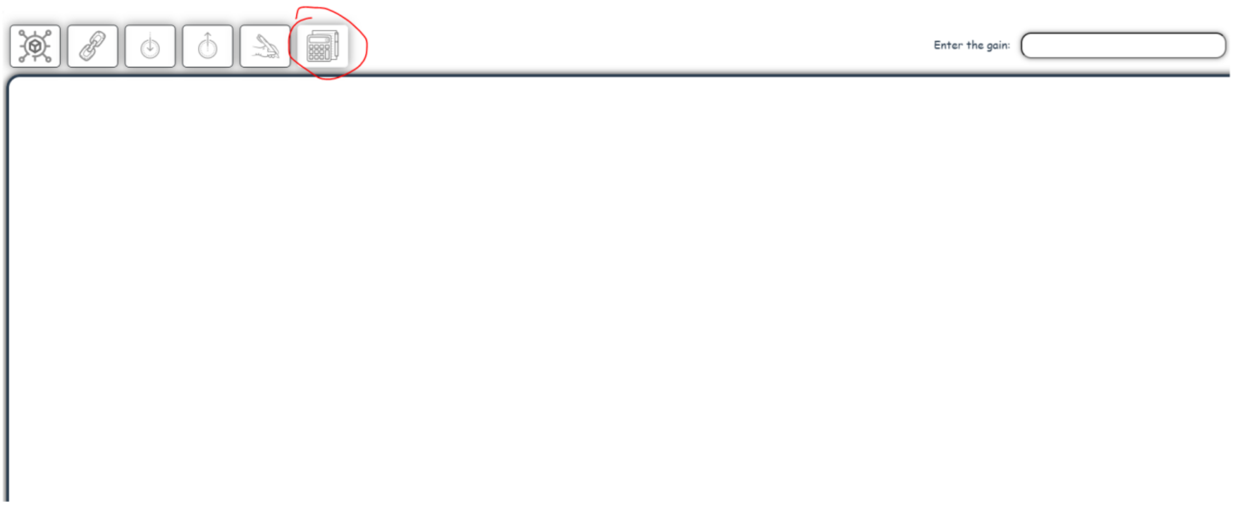
- You can link node using this button:

- You can determine the input node using this button:

- You can determine the output node using this button:

- You can enter the gain by clicking on this button and finally enter the gain in this text box and click on the link then enter:

- Finally, you can click on this button to show all information like transfer function, paths, loop, and self loops.



Part2

1) Problem Statement

Given the characteristic equation of a system, implement the Routh-Hurwitz stability criterion to determine if the system is stable or not. If the system is unstable, list the number and values of poles in the right-half plane (RHP) of the s-plane.

2) Main Features of the Program and Additional Options

- Parses the characteristic equation to extract coefficients.
- Constructs the Routh array based on the extracted coefficients.
- Determines system stability using the Routh-Hurwitz criterion.
- Finds the roots of the characteristic equation and identifies poles in the RHP.
- Displays the Routh array, system stability status, and roots of the equation.

3) Data Structure

- Map: Used to store coefficients for each power of 's' in the characteristic equation.
- Double Array: Represents the Routh array, initialized with null values for ease of calculation.

4) Main Modules

- constructRouthArray: Constructs the Routh array based on the coefficients of the characteristic equation.
- countSignChanges: Counts the number of sign changes in the first column of the Routh array.
- findRoots: Finds the roots of the characteristic equation using the LaguerreSolver.

5) Algorithms Used

- Routh-Hurwitz Criterion: Determines system stability based on the number of sign changes in the first column of the Routh array.
- Laguerre's Method: Finds the roots of the characteristic equation using the LaguerreSolver.

6) Sample Runs

```
Enter the characteristic equation: s^4+2s^3+3s^2+4s+5
Routh Array:
[1.0, 3.0, 5.0]
[2.0, 4.0, null]
[1.0, 5.0, null]
[-6.0, 0.0, null]
[5.0, 0.0, null]
The system is unstable with 2 poles in RHS
Roots:
Root: 0.29 - 1.42i
Root: 0.29 + 1.42i
Root: -1.29 + 0.86i
Root: -1.29 - 0.86i
```

```
Enter the characteristic equation: s^3+2s^2+1s+2
Routh Array:
[1.0, 1.0]
[2.0, 2.0]
[1.0E-9, null]
[2.0, null]
The system is stable
Roots:
Root: 0.00 - 1.00i
Root: 0.00 + 1.00i
Root: -2.00
```

```
Enter the characteristic equation: s^5+2s^4+2s^3+4s^2+11s+10
Routh Array:
[1.0, 2.0, 11.0]
[2.0, 4.0, 10.0]
[1.0E-9, 6.0, null]
[-1.1999999995999998E10, 10.0, null]
[6.0, 0.0, null]
[10.0, 0.0, null]
The system is unstable with 2 poles in RHS
Roots:
Root: -1.31
Root: -1.24 - 1.04i
Root: -1.24 + 1.04i
Root: 0.90 - 1.46i
Root: 0.90 + 1.46i
```

```
Enter the characteristic equation: s^5+2s^4+24s^3+48s^2-25s-50
Routh Array:
[1.0, 24.0, -25.0]
[2.0, 48.0, -50.0]
[8.0, 96.0, 0.0]
[24.0, -50.0, null]
[112.66666666666667, 0.0, null]
[-50.0, 0.0, null]
The system is unstable with 1 poles in RHS
Roots:
Root: -1.00
Root: 1.00
Root: -2.00
Root: -0.00 - 5.00i
Root: -0.00 + 5.00i
```

7) Simple User Guide

1)Input the characteristic equation of the system as shown in sample runs enter all coefficients even if they are 1 and don't use spaces.

2)The program will display the Routh array, system stability status, and roots of the characteristic equation.

3)If the system is unstable, the program will also list the number and values of poles in the right-half plane.