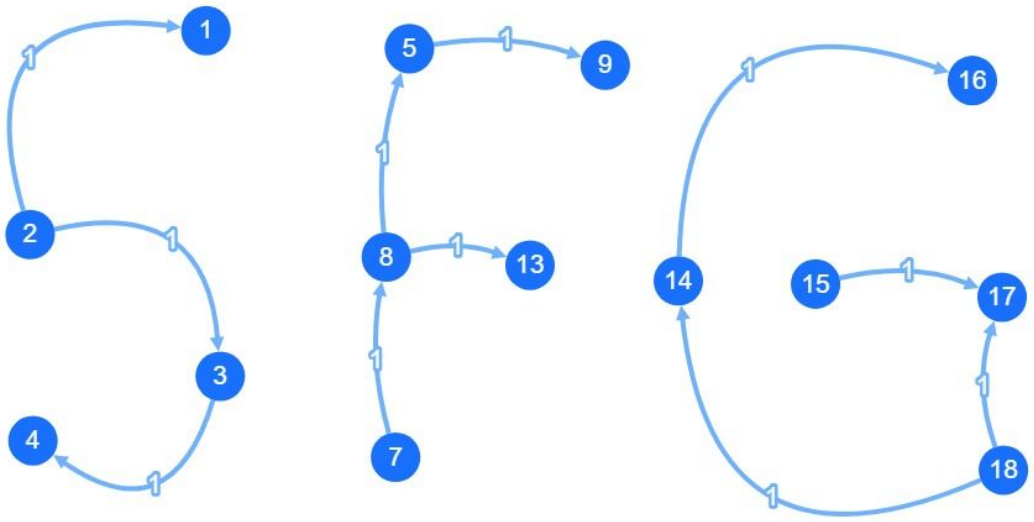
CS241 Linear Control Systems

# Signal Flow Graph Solver

## Report




## Team Members


| | |
|---|---|
| Mahmoud Tarek | (62) |
| Mostafa Labib | (67) |

# Table of Contents

## Problem Statement

A signal-flow graph (SFG), often called a Mason graph, is a specialized flow graph, a directed graph in which nodes represent system variables, and branches (edges, arcs, or arrows) represent functional connections between pairs of nodes. SFG's are most commonly used to represent signal flow in a physical system and its controller(s), forming a cyber-physical system. The problem is to compute the overall gain with mason's rule.

## Overview

We developed the project as a web app. The tools we used include:

- **Programming Language**
  - **Javascript** : our programming language of choice due to its flexibility and robustness. Its vast collection of libraries and frameworks was of great use in addition to its web capabilities.
- **Open Source Libraries**
  - **graphlib.js** : a javascript library that provides data structures for undirected and directed multi-graphs along with algorithms that can be used with them. We used it for the backend part.
  - **cytoscape.js** : a javascript Graph theory / network library for visualisation and analysis. we used it for the frontend part.
- **Git** : for versioning.

The source code is on github:
https://github.com/MahmoudTarek97/Signal-Flow-Graph-Solver

## Main Features

- We support user-friendly drag & drop mode.
- We don't need to know number of nodes or edges previously.
- The user can take an image of the graph using snapshot feature.
- The user can save the graph as JSON file then loading it again.

### Data structure

We used graph data structure for the backend from graphlib.js . It contains mainly the next features :

- Set Node
- Set Edge
- Get Node
- Get Nodes
- Get Edge
- Get Edges

We used another graph data structure for the visualization of the graph (i.e. frontend) from cytoscape.js .

## Main modules

We divided our project into modules. Each module is a javascript file containing a collection functions relevant to each other or belonging to the same process as the following:

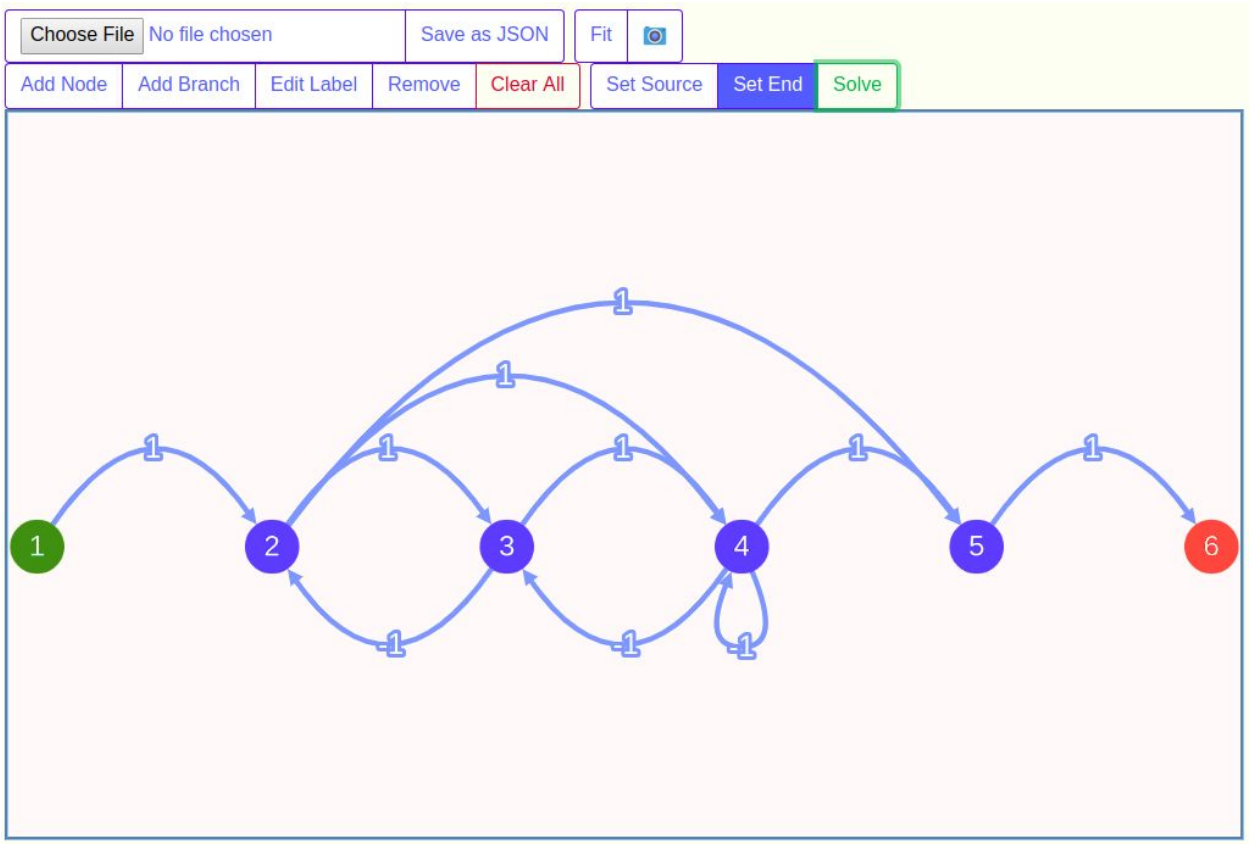
- gui.js file that contains all the gui components and functions.
- guiUtilities that contains some helpers functions used in gui.
- SignalFlowGraph.js file that has all used algorithms required to fulfil the requirements.

## Algorithms Used

We mainly used the DFS that everything else depends on it.
we used DFS to get forward paths and loops.
To get all non touching loops we first get all combinations and extract the non touched ones from it.

# Sample Runs

## Run #1:



## Output

overall gain = 1.25

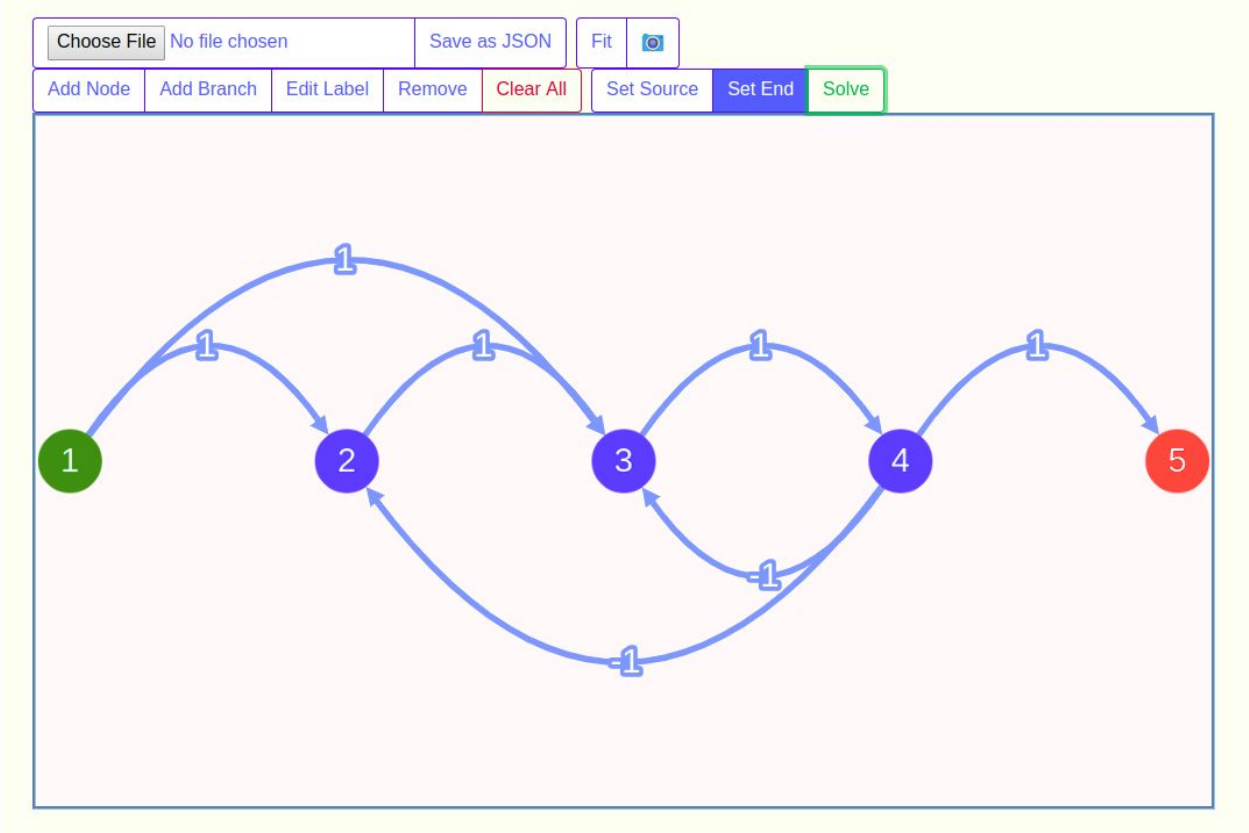
▼ Forward Paths
M1 : {1-2-3-4-5-6}
M2 : {1-2-4-5-6}
M3 : {1-2-5-6}

▼ Individual Loops
L1 : {2-3}
L2 : {2-4-3}
L3 : {3-4}
L4 : {4}

▼ Non Touching Loops
2-non touching :
{2-3} {4}

▼ Deltas
Δ = 4
Δ1 = 1
Δ2 = 1
Δ3 = 3

## Run #2:


Choose File
No file chosen
Save as JSON
Fit
Add Node
Add Branch
Edit Label
Remove
Clear All
Set Source
Set End
Solve
1
2
3
4
5
1
1
1
1
1
-1
-1

# Output

overall gain = 0.6666666666666666

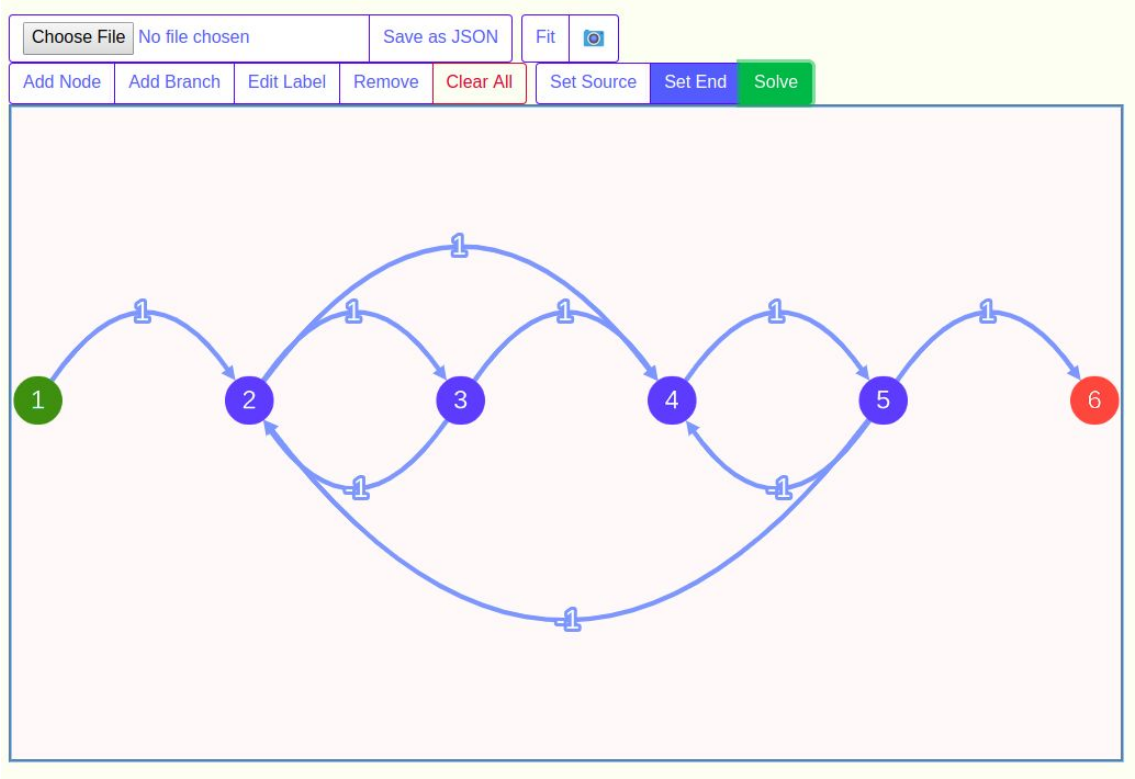
▼ Forward Paths
M1 : {1-2-3-4-5}
M2 : {1-3-4-5}

▼ Individual Loops
L1 : {2-3-4}
L2 : {3-4}

▼ Non Touching Loops

▼ Deltas
Δ = 3
Δ1 = 1
Δ2 = 1

## Run #3:


Choose File
No file chosen
Save as JSON
Fit
Add Node
Add Branch
Edit Label
Remove
Clear All
Set Source
Set End
Solve
1
2
3
4
5
6
1
1
1
1
1
1
-1
-1
-1

# Output

overall gain = 0.3333333333333333

▼ Forward Paths

M1 : {1-2-3-4-5-6}

M2 : {1-2-4-5-6}

▼ Individual Loops

L1 : {2-3-4-5}

L2 : {2-3}

L3 : {2-4-5}

L4 : {4-5}

▼ Non Touching Loops

2-non touching :

{2-3} {4-5}

▼ Deltas

Δ = 6

Δ1 = 1

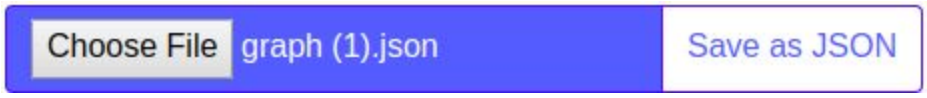
Δ2 = 1

# User guide

The program allows the user to create the graph very easily with the buttons that we provide in a toolbar


- The user can add a node or remove it.
- Then add the branches between nodes.
- The user can clear all the graph and reconstruct it.
- The user is able to rename the nodes names and branches gains.
- Then user should select the source and the end nodes.
- Then he can solve the graph using mason's rule.

The program allows the user to save his graph and load it whenever he wants




- To save the graph he just should clcik the save button.
- To load it he should choose the json file.

The user has the ability to take a picture of the graph as


- To take the snapshot the user can fit the graph first to the center of the screen and then take the snapshot.