Signal Flow Graphs & Routh Stability Criterion

Control Programming Assignment

Team members

Name	ID
Ahmed Ashraf	21010040
Ahmed Osama	21010037
Hager Ashraf	21011505
Nayra Ibrahim	21011504
Saifullah Mosaad	21010651
Youssef Taman	21011621

Source Code

https://github.com/AhmedIAshraf/Control-Systems-Signal-Flow-Routh-.git

Problem 1: Signal Flow Graph

- ### Problem Statement
  - #### Given
    - Total number of nodes and branches gain
  - #### Required
    1. Graphical Interface.
    2. Draw the signal flow graph showing nodes, branches, gains,....
    3. Get all forward paths, individual loops, and all combinations of non-touching loops.
    4. Calculate the value of Δ, Δ1, Δ2,......Δm where m is the number of forward paths.
    5. Calculate overall system transfer function.

- ### Main Features
  1. Getting the nodes and branches gains from the user.
  2. Displaying Signal flow graph, forward paths, individual loops, and all combinations of non-touching loops.
  3. Providing the values for Δ, Δ1, Δ2,......Δm where m is the number of forward paths.
  4. Providing the value of the overall transfer function of the system.

- ## Extra Features

  No extra features.

- ## Data Structures

  1. Adjacency List to represent the graph, each node provided with 2 ArrayList for adjacent nodes and their branches gains.
  2. ArrayList of ArrayLists to hold the forward paths, individual loops, non-touching loops for each loop, and the loops of each forward path.
  3. ArrayList to hold the paths gains, loops gains, and delta for each path.

- ## Main Modules

  1. ### User Interface Module:
     a. Input area: Allows users to input the nodes and branches gains.

     b. Display area: Shows the signal flow graph, forward paths, individual loops, all combinations of non-touching loops, values of Δ, and the overall T.F of the system.
  2. ### Data Processing Module:
     a. Input parsing: Parses the user input, extracting the nodes and adjacent nodes and branches gains for each node.

     b. Get the forward paths and their gains, individual loops and their gains, all combinations of non-touching loops.

     c. Calculate the overall transfer function of the system.

- ## Algorithms Used

  1. DFS to get the forward paths and individual loops.

- Explanation of Application

1. Node class:-

Representing the graph by ArrayList of Nodes, each node has id, ArrayList of Nodes for adjacent nodes and ArrayList of double for gain of each branch.

2. GraphRep class:-

- repGraph(): to represent the graph as ArrayList<Nodes>.

- getForwardPaths(): call GetForwardPaths() in signalFlow and return forward_paths (ArrayList<ArrayList<Integer>>) each element is forward Path of nodes' id.

- getForwardPathsGains(): return path_gain(ArrayList of Double) each Element is gain for the path at the same index of forward_paths.

- getIndividualLoops(): call GetLoops() in signalFlow and then call GetNonDoublicateLoops() in signalFlow to remove duplicate loops and return loops(ArrayList<ArrayList<Integer>>)each element is a loop of nodes' id.
  (note: first and last elements of loop itself are same)

- getIndividualLoopsGains(): return loop_gain(ArrayList<Double>) each Element is gain for the loop at the same index of loops.

- getNonTouchingLoops(): call GetAllCombinationsOfNonTouchingLoops() In signalFlow and then call GetNonDoublicateLoops() in signalFlow to remove duplicate combinations of non_touching_loops and return non_touching_loops(ArrayList<ArrayList<Integer>>) each element is Possible combination of non-touching loops indexes.

- getLoopsForEachPath(): for each forward path, it calls getLoopsForPath() in signalFlow to get loops for that path. This function returns path_delta_gain (ArrayList<ArrayList<Integer>>) each element is an ArrayList of loop indexes of path that has the same index in forward_paths.

- getDelta(): return the delta of the system.

- getPathDelta(): return path_delta(ArrayList<Double>) each element is the delta value for that path that has the same index in forward_paths.

- getOverallTF(): return the overall TF of the system.

3. SignalFlow class:-

- GetForwardPaths():-
    - It receives graph arrayList as input and returns ForwardPaths (ArrayList<ArrayList<Integer>>).
    - Call GetForwardPath() which will get all forward paths and their gains recursively using the DFS algorithm, we use one arrayList by removing node by node and also branch's gain.

- GetLoops():-
    - It receives graph arrayList as input and returns Loops (ArrayList<ArrayList<Integer>>).
    - Call GetLoop() which will get all loops and their gains recursively using the DFS algorithm, when we detect visited node, we get node indexes of that loop and its gain.
    - It has a problem, there are duplicate loops.

- GetNonDoublicateLoops():-
    - This function removes duplicate loops and their gains by comparing each loop with the loops coming after it.
    - This function is also used to remove duplicate combinations of non-touching-loops.

- GetAllCombinationsOfNonTouchingLoops():-
    - This function gets all combinations of non-touching-loops.
    - For each loop, it calls GetNonTouchingLoops() to get all combinations of non-touching-loops that have that loop by first get non-touching-loops to that loop then recursively add one of them to that loop, so we can get 3,4,... combinations.
    - It has a problem, there are duplicate combinations.

- getLoopsForPath():-
    - It receives the forward path and loops to get remaining loops after removing that path and returning ArrayList<Integer> that has indexes of those loops.

- Sample Runs

🗀 Signal Flow Graph

User Guide

Introduction

Okay, we are coming with something amazing for you,The Signal Flow Graph Application user guide. This guide will help you effectively use our application to get the overall transfer function of the system by representing it as a graph of nodes and branches . Below are instructions on how to use our application easily.

Getting Started

- Accessing the Application
    - Run the Backend Application on http://localhost:8080.
    - Install npm module ➜ npm install.
    - Install axios module ➜ npm install axios.
    - Install gojs module ➜ npm install gojs.
    - Run SignalFlow Vue App ➜ npm run serve.

    The application interface includes a button to select the number of nodes, a button to generate the nodes and a button for displaying the analysis.

- Using the Application
    1. Select the number of nodes of the graph.
    2. Generate the nodes.
    3. Draw the graph by connecting the nodes with the branches and adjusting the gains.
    4. Generate the analysis.

- Interpreting the Results
    - Displaying table for forward paths, their gains and their deltas.
    - Displaying table for individual loops and their gains.
    - Displaying table for 2,3,..... Non-touching-loops.
    - Output the delta.
    - Output the Overall TF of the system.

Problem 2: Routh Stability Criterion

- ### Problem Statement
  - Given
    - Characteristic equation of the system. Assume that all the coefficients of S_0 to Sn are given.
    - Input example: s^5+s^4+10s^3+72s^2+152s+240 or s ^ 3 + s ^ 2 + s + 1
  - Required
    1. Using Routh criteria, state if the system is stable or not.
    2. If the system is not stable, list the number and values of poles in the RHS of the s-plane.

- ### Main Features
  1. Getting the characteristic equation from the user.
  2. Displaying Routh table for the given system.
  3. Check whether the system is stable or not.
  4. Providing RHS roots and their number.

- ### Extra Features
  1. Checking input validity.

- ### Data Structures
  1. 2D - Array for Routh Table.
  2. List for equation coefficients.

- Main Modules
    1. User Interface Module:
        a. Input form: Allows users to input the characteristic equation.
        b. Display area: Shows the Routh table, stability analysis result, and the number of right-hand side (RHS) roots (if the system is unstable).
    2. Data Processing Module:
        a. Input parsing: Parses the user input, extracting the coefficients of the characteristic equation.
        b. Routh table generation: Computes the Routh table using the provided coefficients.
        c. Stability analysis: Determines the stability of the system based on the Routh table.
        d. Counting RHS roots: Calculates the number of roots of the characteristic equation on the right-hand side of the complex plane if the system is unstable.

- Algorithms Used
    1. Laguerre Solver for finding the equation roots.

Sample Runs

🗀 Routh Hurwitz

  🗀 Normal Samples

  🗀 Zero in the first column (replace zero with .000001)

  🗀 Entire row is zero

  🗀 Invalid input

User Guide

Introduction

Welcome to the Routh-Hurwitz Stability Analysis Web Application user guide. This guide will help you effectively use our application to analyze the stability of linear systems using the Routh-Hurwitz criteria. Below are instructions on getting started, using the application, troubleshooting, and additional resources.

Getting Started

- Accessing the Application
  - Run BackendApplication.java on 8080 local host
  - Install npm module ➜ npm install
  - Run Rowth Hurwitz Vue App ➜ npm run serve

  The application interface includes input fields for entering the coefficients of the characteristic equation, a button to generate the Routh table, and areas to display the results.

- Using the Application

  1. Inputting the Characteristic Equation
  2. Enter the characteristic equation into the provided input field, ensuring correct order and format.
  3. Generating the Routh Table: Click the "Generate Routh Table" button after entering coefficients to generate the Routh table.

- Interpreting the Results
  - The Routh table will be displayed, along with the stability analysis result indicating system stability (stable, marginally stable, or unstable).

- Understanding the Number of RHS Roots
    - The count of roots on the right-hand side of the complex plane will also be displayed, indicating system stability.
- Tips and Best Practices

    Input Guidelines

    - Enter coefficients accurately and in the correct order to generate the Routh table correctly.
    - Avoid entering invalid characters or symbols in the input fields.
    - Interpreting Results.

- Troubleshooting

    Error Messages

    - If encountering an error message indicating invalid input, double-check entered coefficients for correct format and order.

- Algorithm Explanation
    - For deeper understanding of the Routh-Hurwitz stability criteria algorithm, refer to [Routh-Hurwitz stability criterion - wikipedia].