Routh-Hurwitz Stability Analyzer: A MATLAB App for Symbolic and Numeric Evaluation of LTI Systems

Harshit Pranjal
Department of Mathematics
Birla Institute of Technology, Mesra, Ranchi
imh1001924@bitmesra.ac.in

Abstract

This paper presents a MATLAB-based application titled *Routh-Hurwitz Stability Analyzer*, designed to analyze the stability of linear time-invariant (LTI) systems using the classical Routh-Hurwitz criterion. The app accepts both symbolic and numeric inputs and automatically generates the corresponding polynomial, constructs the Routh-Hurwitz table, evaluates sign changes, plots system roots, and provides a stability verdict. The application serves as an educational and research verification tool to help reduce human errors in system stability analysis.

1 Introduction

Stability is a fundamental requirement in control systems. A system is considered stable if its output remains bounded for any bounded input. In many real-world systems — such as power electronics, aerospace control, and biomedical applications — ensuring stability is critical for safe operation.

The Routh-Hurwitz criterion offers an algebraic method for determining whether all roots of a system's characteristic polynomial lie in the left half of the complex plane, without explicitly computing the roots. Its application avoids complex root-solving, especially for high-order systems where analytical solutions are impractical.

Despite its robustness, manual construction of the Routh table is prone to human error, especially when encountering special cases like zero rows or symbolic expressions. To address this challenge, this paper introduces a MATLAB-based application that automates the Routh-Hurwitz procedure. It accepts both symbolic and numeric inputs, evaluates stability, generates graphical root plots, and simplifies the process for learners and researchers alike.

The Routh-Hurwitz method determines the stability of a linear time-invariant (LTI) system by examining the sign variations in the first column of a constructed Routh array derived from the characteristic polynomial. The mathematical formulation eliminates the need for explicitly solving high-degree polynomials, which is particularly useful in control system design where analytical root calculation becomes infeasible.

However, manually constructing the Routh table can be tedious and error-prone, especially when handling zero entries or symbolic expressions. These challenges have led to the demand for reliable computational tools to support students and researchers. Motivated by this need, this paper introduces a MATLAB-based application that automates the Routh-Hurwitz process. The application not only supports both symbolic and numeric analysis but also incorporates stability evaluation, graph plotting, and data export functionalities. It aims to facilitate learning and reduce the probability of human error in mathematical verification and research documentation.

2 Theoretical Background

The Routh-Hurwitz criterion determines system stability by analyzing sign changes in the first column of a specially constructed Routh table. For a characteristic polynomial:

$$P(s) = a_0 s^n + a_1 s^{n-1} + \ldots + a_n,$$

the number of roots with positive real parts equals the number of sign changes in the first column of the Routh array.The significance of Routh-Hurwitz-based stability analysis in compartmental disease models, particularly SIR-type models, has been demonstrated in earlier studies [4].

Special handling is required when a zero appears in the first column or when an entire row is zero. The algorithm incorporates auxiliary polynomial methods and small epsilon substitutions to maintain numerical robustness.

The theoretical foundation of the Routh-Hurwitz criterion follows from standard graduate-level control literature, such as the ECE 680 lectures [6].

3 App Design and Features

The *Routh-Hurwitz Stability Analyzer* is built using MATLAB R2025a's App Designer. Key features include:

- Symbolic or numeric input modes
- Polynomial auto-generation (for both numeric and symbolic inputs)
- Routh-Hurwitz table construction
- Stability detection with sign change counter
- Root plotting on the complex plane (numeric input only)
- Downloadable outputs and graph (numeric mode)
- Embedded help system for educational support
- Note: The app is only valid for algebraic polynomials and does not support transcendental polynomials (e.g., those involving exponential or logarithmic terms).

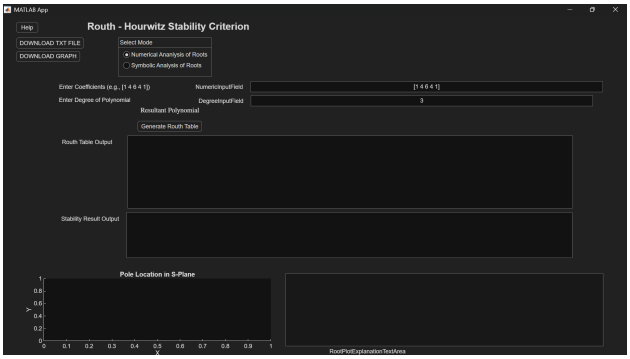

Figure 1: Workflow of the Routh-Hurwitz Stability Analyzer app

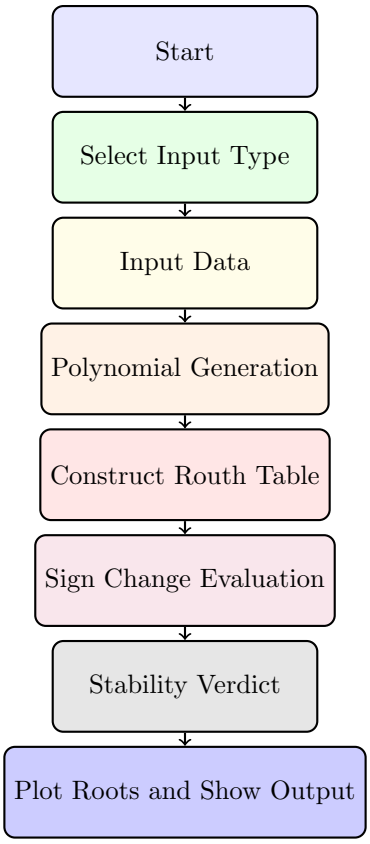

Figure 2: Simplified vertical flowchart of the Routh-Hurwitz Stability Analyzer app

4 Methodology

The methodology consists of the numerical construction of the Routh-Hurwitz table from the user's input, automated handling of special cases, and derivation of the system's stability conclusion. The core computational logic is encapsulated in the custom MATLAB function `routhAnalyzer`. This function executes the following steps:

1. Accept the input coefficients or degree, depending on mode.
2. Construct the first two rows of the Routh table from the input.
3. Fill subsequent rows using the standard Routh-Hurwitz determinant formula.
4. Implement zero handling logic:
   - Replace near-zero entries with epsilon ($\varepsilon = 10^{-6}$) to avoid division by zero.
   - When an entire row becomes zero, generate the auxiliary polynomial using the row above and differentiate to replace it.
5. Count sign changes in the first column.
6. Generate the verdict: Stable (no sign changes) or Unstable (sign changes present).
7. Plot the roots on the complex plane for graphical validation.

The MATLAB implementation is summarized below:

Listing 1: routhAnalyzer.m – Core Logic

```
function [routhTable, \stabilityStatement, signChanges] = \routhAnalyzer(coeffs)
% Computes \Routh–Hurwitz table and determines system stability
... % Paste your MATLAB code here, trimmed if needed
end
```

The core computation for constructing the Routh table and analyzing system stability is implemented in the MATLAB script `routhAnalyzer.m`, which is responsible for all backend numerical and symbolic evaluations. The graphical user interface (GUI), user input handling, mode selection, result display, and export functionalities are integrated through MATLAB App Designer and bundled as `RouthStability.mlapp`.

For code access or academic reference, readers may contact the author. Both the logic script and GUI app file are available upon request or may be shared via the institutional repository (if hosted).

This methodology ensures accurate, symbolic-compatible, and robust Routh table construction suitable for educational and research-level control system design.

5 Case Studies

5.1 Numeric Example

Input: `[1 4 6 4 1]`

```
--- Routh-Hurwitz Stability Report ---

Polynomial:
   4*s + 6*s^2 + 4*s^3 + s^4 + 1

Routh Table:
       1.0000    6.0000    1.0000
       4.0000    4.0000         0
       5.0000    1.0000         0
       3.2000         0         0
       1.0000         0         0



Stability Verdict:
   The system is stable.
   Out of 4 roots, 4 have negative real parts.
```

Figure 3: *
(a) Routh Table Output

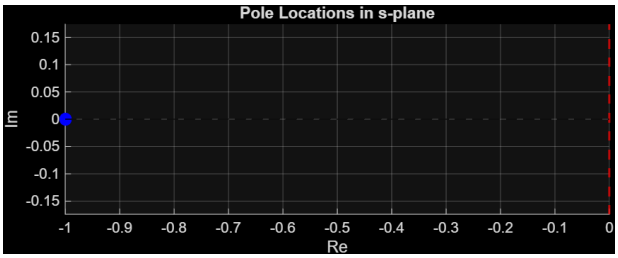

Figure 4: *
(b) Root Plot

Figure 5: Numeric Input Analysis: Table and Root Plot

5.2 Numeric Example 2

Input: `[1 4 6 5 7 1]`

```
--- Routh-Hurwitz Stability Report ---

Polynomial:
   7*s + 5*s^2 + 6*s^3 + 4*s^4 + s^5 + 1

Routh Table:
       1.0000    6.0000    7.0000
       4.0000    5.0000    1.0000
       4.7500    6.7500         0
      -0.6842    1.0000         0
      13.6923         0         0
       1.0000         0         0


Stability Verdict:
   The system is unstable. Sign changes: 2
   Out of 5 roots, 3 have negative real parts.
```

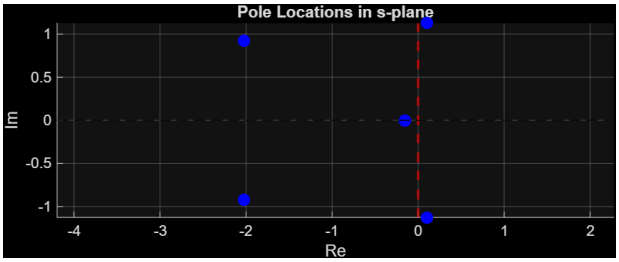

Figure 6: *
(a) Routh Table Output

Figure 7: *
(b) Root Plot

Figure 8: Numeric Input Analysis for $[1, 4, 6, 5, 7, 1]$: Table and Root Plot

5.3 Symbolic Examples

Input: Degree = 3 and 4

```
--- Routh-Hurwitz Stability Report ---

Polynomial: (could not parse input)

Routh Table:
   [                 a3, a1]
   [                 a2, a0]
   [-(a0*a3 - a1*a2)/a2,  0]
   [                 a0,  0]



Stability Verdict:
   Condition 1: a3 > 0
   Condition 2: a2 > 0
   Condition 3: -(a0*a3 - a1*a2)/a2 > 0
   Condition 4: a0 > 0
```

```
--- Routh-Hurwitz Stability Report ---

Polynomial: (could not parse input)

Routh Table:
   [                                       a4, a2, a0]
   [                                       a3, a1,  0]
   [                          -(a1*a4 - a2*a3)/a3, a0,  0]
   [(a4*a1^2 - a2*a1*a3 + a0*a3^2)/(a1*a4 - a2*a3),  0,  0]
   [                                       a0,  0,  0]


Stability Verdict:
   Condition 1: a4 > 0
   Condition 2: a3 > 0
   Condition 3: -(a1*a4 - a2*a3)/a3 > 0
   Condition 4: (a0*a3^2 + a1^2*a4 - a1*a2*a3)/(a1*a4 - a2*a3) > 0
   Condition 5: a0 > 0
```

Figure 9: *
(a) Symbolic Routh Table (Degree = 3)

Figure 10: *
(b) Symbolic Routh Table (Degree = 4)

Figure 11: Symbolic Evaluation for Degrees 3 and 4

6 Conclusion and Future Scope

The presented MATLAB app provides a robust, accessible way to perform Routh-Hurwitz stability analysis. It reduces human error, improves accessibility for new researchers, and enhances learning through symbolic and numeric flexibility.

In future versions, the app can be extended to handle delay differential equations and systems with fractional-order dynamics. Additionally, integration with eigenvalue analysis modules and machine learning tools may enhance its capability in automatic error correction and research verification workflows. Such improvements would broaden its utility from an educational aid to a comprehensive research tool. The proposed system can be extended toward fractional-order dynamic modeling frameworks as seen in recent research on plant disease systems [5].

Currently, the application supports stability analysis for algebraic polynomials only. It does not handle transcendental characteristic equations (such as those involving exponential or logarithmic terms), which commonly arise in time-delay or infinite-dimensional systems. The author is actively exploring methods to incorporate such systems into future versions of the app, including extensions toward quasi-polynomials and Laplace domain representations.

Final Remarks. This tool bridges the gap between theoretical Routh-Hurwitz analysis and practical system validation by offering a user-friendly, error-resilient computational interface. It makes research in this field more engaging, visual, and largely error-free — catering even to young researchers who have just stepped into control system theory.

Author's Note

The author has independently developed this application and prepared the accompanying analysis, aiming to support fellow researchers and learners working in the domain of system stability.

References

References

[1] K. Ogata, *Modern Control Engineering*, 5th ed., Prentice Hall, 2010.

[2] MATLAB and App Designer, Version R2025a, The MathWorks, Inc., Natick, Massachusetts, United States.

[3] P. K. Shaw, "Impact of Vaccination and Treatment in a Fractional SIRV Measles Model: A Comparative Numerical Study," Manuscript.

[4] D. Ezekiel, "Stability Analysis of an SIR Infectious Disease Model," *International Journal of Nonlinear Science*, vol. 12, no. 2, pp. 107–113, 2011.

[5] P. K. Shaw, S. Kumar, S. Momani, and S. Hadid, "Dynamical Analysis of Fractional Plant Disease Model with Curative and Preventive Treatments," *Chaos, Solitons & Fractals*, vol. 156, Article ID 111830, 2022.

[6] ECE 680 Lecture Notes, "Modern Automatic Control: Routh's Stability Criterion," Purdue University, 2007.