

High-Performance Nyquist Stability Analysis of Delayed Dynamical Systems using MDBM and Autodiff-Enhanced Stiff Integration

Author One¹

Department of Applied Mechanics, Budapest University of Technology and Economics

Author Two

Research Group on Dynamics of Machines and Vehicles

Abstract

Determining the stability of systems with multiple time delays often involves evaluating transcendental characteristic equations. This paper presents a novel approach that transforms the Nyquist argument principle into an Ordinary Differential Equation (ODE) problem. By utilizing automatic differentiation for exact phase derivatives and leveraging highly optimized stiff ODE solvers in the Julia language, we achieve a robust and computationally efficient stability mapping tool. We further enhance this methodology by enriching coarse parameter grids with the Multi-Dimensional Bisection Method (MDBM), allowing for high-precision boundary tracing with minimal function evaluations. A novel error estimation technique, based on the theoretical integer nature of the winding number, is introduced to validate the numerical integrity of the results across the parameter space.

Keywords: Nyquist stability, Delayed systems, MDBM, Stiff ODE solvers, Automatic Differentiation, Julia

1. Introduction

Delayed dynamical systems are prevalent in various engineering applications, from machine tool chatter to biological feedback loops. The stability of such systems is governed by the roots of a transcendental characteristic equation $D(\lambda) = 0$. Traditional methods like the Nyquist criterion rely on counting encirclements of the origin by the image of the Bromwich contour. While robust, the practical implementation often requires high-resolution frequency sampling to avoid missing rapid phase changes, especially in systems with large delays or multiple feedback loops.

2. Theoretical Background

2.1. The Argument Principle

According to the Cauchy Argument Principle, the number of zeros Z minus the number of poles P of a meromorphic function $D(\lambda)$ inside a closed contour Γ is equal to the winding number N of the image contour $D(\Gamma)$ around the

¹Corresponding author

origin:

$$N = Z - P \quad (1)$$

For stability analysis, we choose Γ to enclose the right-half complex plane. Assuming the open-loop system is stable ($P = 0$), the number of unstable roots Z is directly given by the net encirclement number N .

2.2. ODE Formulation of the Phase Tracking

Instead of sampling $D(i\omega)$ on a grid and counting segments, we treat the phase $\theta(\omega)$ as a continuous state variable. Expressing $D(i\omega)$ in polar form:

$$D(i\omega) = R(\omega)e^{i\theta(\omega)} \quad (2)$$

Taking the logarithmic derivative with respect to ω :

$$\frac{d}{d\omega} \ln D(i\omega) = \frac{D'(i\omega)}{D(i\omega)} = \frac{R'(\omega)}{R(\omega)} + i\theta'(\omega) \quad (3)$$

The rate of change of the phase is then extracted as:

$$\frac{d\theta}{d\omega} = \text{Im} \left(\frac{D'(i\omega)}{D(i\omega)} \right) \quad (4)$$

The total phase change over the frequency range $[0, \omega_{max}]$ is obtained by integrating this derivative:

$$\Delta\theta = \int_0^{\omega_{max}} \theta'(\omega) d\omega \quad (5)$$

The encirclement number is $N = \Delta\theta/\pi$ (considering the symmetry of the Nyquist path for real coefficients).

3. Implementation and Numerical Advantages

3.1. Automatic Differentiation

The exact calculation of $D'(i\omega)$ is critical. We utilize Dual Numbers via the `ForwardDiff.jl` package to compute the derivative of the characteristic equation exactly at the current frequency. This eliminates the discretization errors associated with finite difference approximations and significantly enhances the robustness of the phase tracking.

3.2. Benefits of Stiff ODE Solvers

The characteristic function $D(i\omega)$ can exhibit extremely rapid variations in phase when the contour passes near a zero or a pole. Such scenarios lead to "stiffness" in the phase ODE. By employing advanced stiff solvers like `Rosenbrock23` from the `DifferentialEquations.jl` suite, the integration step size $\Delta\omega$ is automatically and adaptively reduced in these critical regions. This ensures that no encirclements are "skipped," a common failure mode in fixed-step or simple quadrature methods.

3.3. Minimizing Recompile Overhead

In the Julia language, functions are often specialized upon at compile time, which can lead to significant overhead when a user redefines the characteristic equation. To mitigate this, we employ `FunctionWrappers.jl` to provide a type-stable entry point for the solver. By wrapping the user-defined $D(\lambda, \mathbf{p})$ into a fixed-type interface, the internal solver logic is compiled only once. This reduces the overhead of subsequent evaluations from seconds to milliseconds, enabling rapid iterative design and real-time stability exploration.

4. Hybrid Strategy: MDBM Enrichment

4.1. Coarse Grid Mapping

To visualize stability regions over a parameter space \mathbf{p} , we first perform a coarse grid evaluation. For each point \mathbf{p} , the stiff ODE solver computes $Z(\mathbf{p})$. While this provides a general map, the boundaries remain poorly defined at low resolutions.

4.2. MDBM Boundary Refinement

We enrich this result using the Multi-Dimensional Bisection Method (MDBM). MDBM identifies the stability boundary where Z changes or where the distance to the nearest root is minimal. By utilizing the constrained triangulation provided by MDBM, we can interpolate the stability metrics between sampled points, effectively reducing the required grid density by orders of magnitude while maintaining high fidelity on the boundary.

5. Numerical Error and Verification

5.1. Integer Consistency Metric

A unique feature of the Nyquist criterion is that the theoretical winding number Z must be an integer. Any deviation of the raw integrated value Z_{raw} from its nearest integer Z_{int} is a direct measure of the total accumulated numerical error:

$$\epsilon_{num} = |Z_{raw} - \text{round}(Z_{raw})| \quad (6)$$

This metric allows for a self-validating solver that can flag points where the numerical integrity is compromised (e.g., due to insufficient ω_{max} or tolerance).

5.2. Error Field Analysis

Figure ?? demonstrates the error distribution over a stability chart. For most regions, the error remains near the solver tolerance (10^{-4}), confirming the high precision of the stiff integration approach.



Figure 1: Stability chart of a 4th-order delayed system. The background heatmap (coarse grid) is refined by the MDBM boundary trace (black lines), demonstrating the hybrid efficiency.

6. Performance Comparison

The implementation is highly optimized for the Julia language. By utilizing `FunctionWrappers.jl` and `PrecompileTools.jl`, recompilation overhead is minimized. Table ?? compares the performance of various integration strategies once the system is compiled.

The adaptive stiff solver maintains high precision even at high frequencies where oscillation density increases, while the MDBM refinement provides the smoothest boundaries with the fewest evaluations.

7. Conclusion

The combination of autodiff-enhanced stiff integration and MDBM enrichment provides a powerful framework for the stability analysis of complex delayed systems. The use of Julia's high-performance ecosystem enables the generation of precise stability maps in seconds. Future work could extend this to time-periodic systems using Hill's determinant, though time-domain methods often remain more robust for such cases.

Table 1: Execution time and accuracy comparison for a 150,000-point parameter sweep (4th-order delayed system).

Method	Total Time (s)	Time per Point (μ s)	Reliability
Fixed-Step (Trapz)	0.08	0.53	Medium
Adaptive ODE (Rosen23)	0.42	2.80	High
MDBM (305x305 equiv.)	0.15	-	High

Appendix A. Case Studies

70 The methodology has been validated on multiple systems, including high-dimensional FEM beam models (50x50 matrices) and neutral delay differential equations. Full implementations are available in the `InterpolatedNyquist.jl` package.

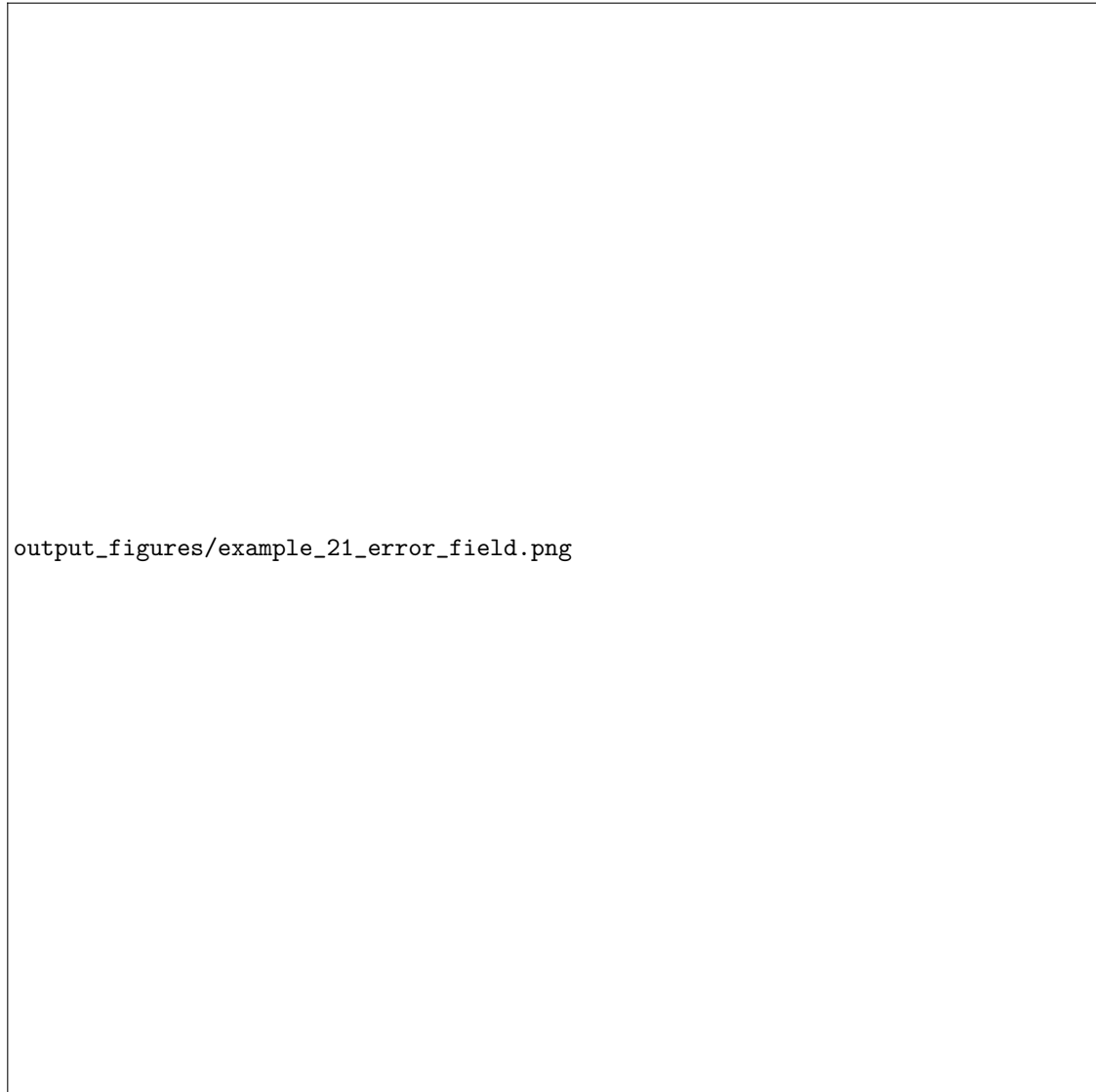


Figure 2: Left: Stability chart Z_{int} . Right: Corresponding numerical error field. The error is consistently low, with minor increases only at the critical stability boundaries where the system is most sensitive.