Stability-Constrained Inverse Modeling for Parameter Identification in Dynamical Systems via Neural Surrogates

Menacho Ordoñez, Daniel

UNIVERSITY OF MICHIGAN

Abstract

Dynamical systems are governed by differential equations that require accurate **parameter identification** to achieve desired stable behavior. Traditional forward modeling approaches predict system outputs from given parameters but lack mechanisms to ensure that the identified parameters lead to stable long-term dynamics. In this work, we propose a **stability-constrained inverse modeling** framework that integrates **neural surrogate models** with **Lyapunov-based stability guarantees** to delimit admissible parameter spaces, ensuring system stability. Our proposed approach relies on Deep Operator Network as flexible surrogate dynamic models and incorporates a **physics-informed learnable Lyapunov function** to characterize regions of attraction. The preliminaries results show that the inverse model has the tendency to select positive and physically consistent parameter values, ensuring the stability of the system.

Introduction

Stability analysis constitutes a cornerstone of research in control systems [1]. Ensuring stability guarantees that the system maintains consistent qualitative behavior in its trajectories relative to an equilibrium state [2]. The traditional method for stability analysis relies on the manual design of a Lyapunov function. To automate this process, computational approaches based on sums-of-squares optimization have gained widespread attention [3]. Recent advances in Physics-Informed Neural Networks [4] and Neural Ordinary Differential Equations [5] have begun to incorporate stability representations through Lyapunov candidate functions. The main drawback of these forward approaches lies in their focus on achieving stability for fixed inputs, without analyzing the true range of parameter variations that lead to stable behavior. To address these limitations, we propose a stability-constrained inverse modeling framework that integrates neural surrogate models with Lyapunov-based stability constraints.

Proposed Method

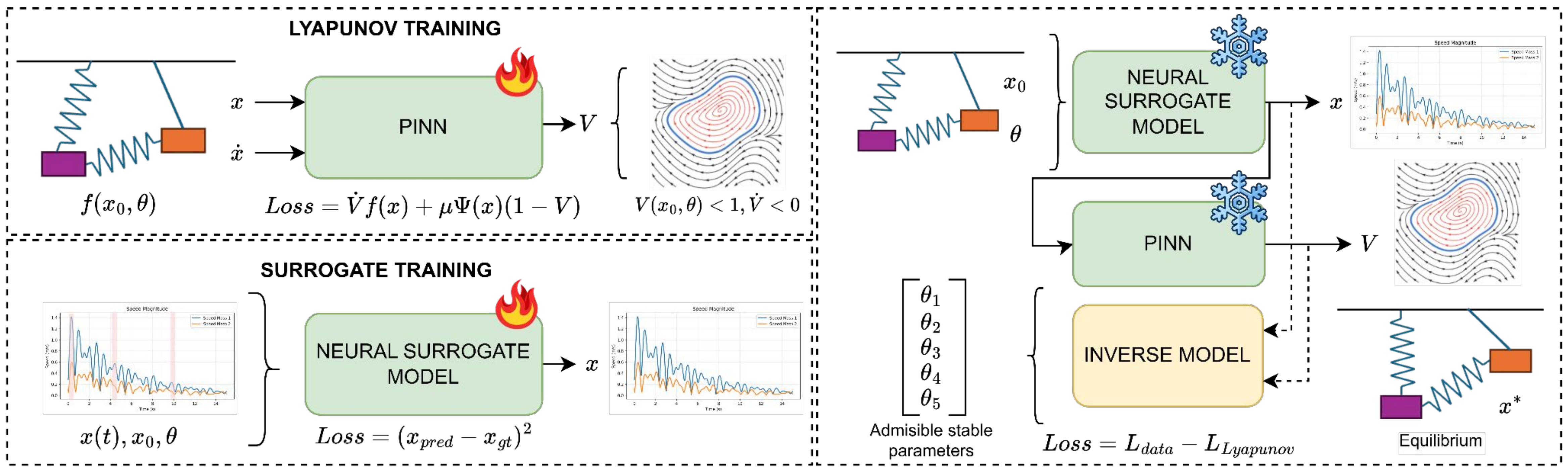
LYAPUNOV TRAINING
x
$\dot{x}$
PINN
V
$f(x_0, \theta)$
$Loss = \dot{V} f(x) + \mu\Psi(x)(1-V)$
$V(x_0, \theta) < 1, \dot{V} < 0$
SURROGATE TRAINING
$x(t), x_0, \theta$
NEURAL SURROGATE MODEL
$Loss = (x_{pred} - x_{gt})^2$
x_0
θ
NEURAL SURROGATE MODEL
PINN
INVERSE MODEL
θ_1 θ_2 θ_3 θ_4 θ_5
Admisible stable parameters
$Loss = L_{data} - L_{Lyapunov}$
x^*
Equilibrium

Figure 1. The pipeline consists of two stages: (1) a pre-training stage, where the Lyapunov-based PINN learns stability characteristics and the Neural Surrogate Model captures the system dynamics, and (2) an inverse modeling stage that uses both pretrained components to compute the loss and identify admissible stable parameters.

Dataset

The dataset contains **50,000 trajectories** sampled every 0.1 seconds over a 15-second horizon, computed using SciPy's **Runge–Kutta ODE solver**.

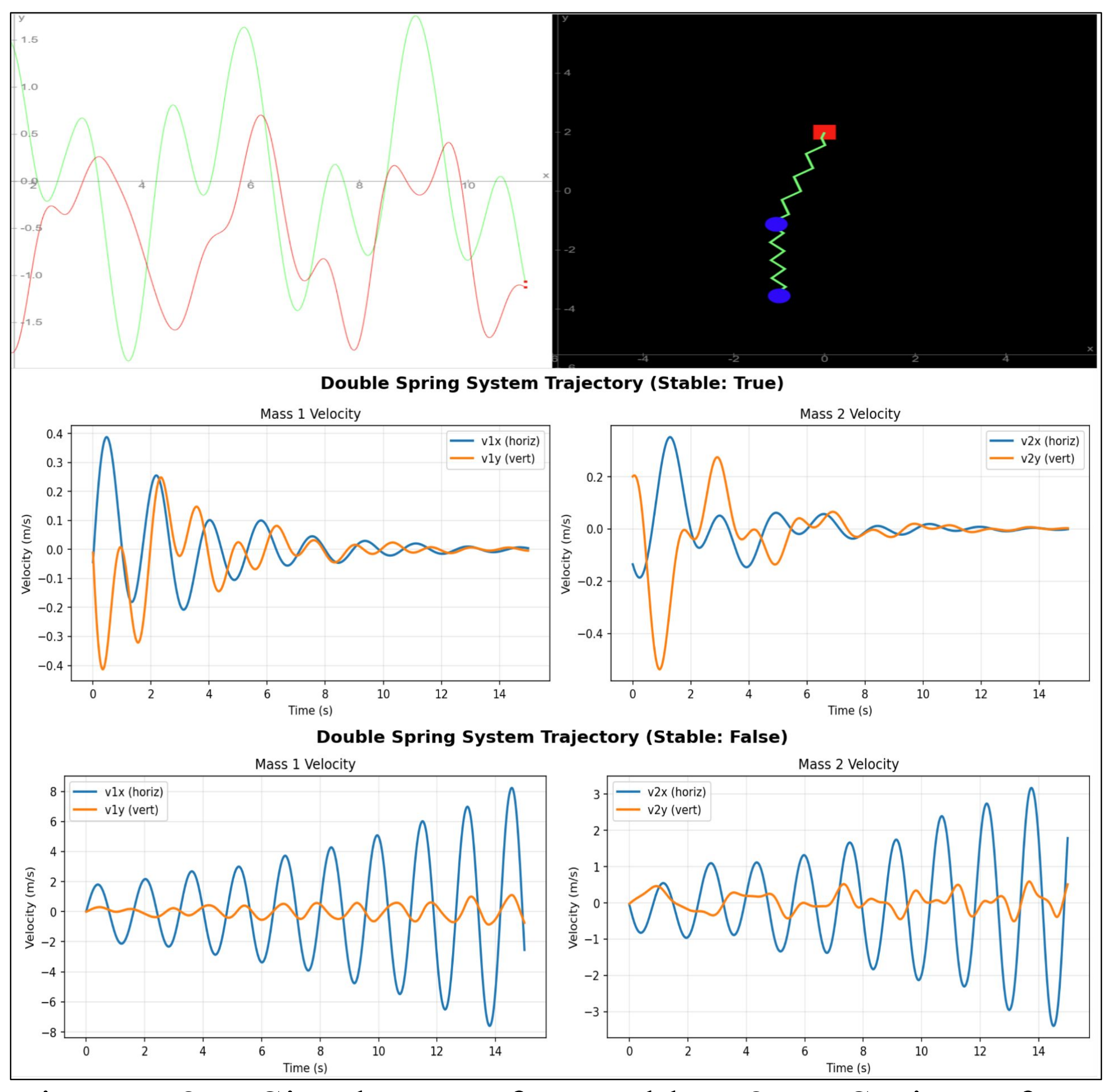

Figure 2. Simulator of Double 2D Spring from MyPhysicLab and Stable/Unstable sample from our dataset.

Pseudocode

```
Algorithm 1: Stability-Constrained Inverse Design
Input: Baseline $x_0$, $\theta$, desired output $y^*$, learning rates
       $\eta_1, \eta_2, \eta_3$, penalty $\alpha$, tolerance $\varepsilon_{\text{total}}$, number of
       restarts $N_{\text{restarts}}$, number of epochs $N_{epochs}$
Output: Admissible set $\mathcal{A} = \{(x_0, \theta)\}$
Train Deep Operator Network:
for $i = 1$ to $N_{epochs}$ do
    Sample batch $(x_i, \theta_i, y_i)$ from simulator
    Compute $\hat{y}_i \leftarrow M_{\text{fwd}_\phi}(x_i, \theta_i)$
    Compute data loss $\mathcal{L}_{\text{data}} \leftarrow \|\hat{y}_i - y_i\|^2$
    Update $\phi \leftarrow \phi - \eta_1 \nabla_\phi \mathcal{L}_{\text{data}}$

Train Neural Lyapunov Function:
for $i = 1$ to $N_{epochs}$ do
    Sample $x$ in state space
    Compute value: $V_\psi(x)$
    Compute gradient $\nabla_x V_\psi(x)$
    Compute Lie derivative: $\dot{V}_\psi(x) \leftarrow \nabla_x V_\psi(x) \cdot f(x, \theta)$
    Compute Lyapunov loss: $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{origin}}$
    Zubov Residual: $\mathcal{L}_{\text{PDE}} \leftarrow \left(\dot{V}_\psi(x) + \mu h(x)(1 - V_\psi)\right)^2$
    Boundary Condition: $\mathcal{L}_{\text{origin}} \leftarrow V_\psi(0)^2 + \nabla_x V_\psi(0)^2$
    Update parameters: $\psi \leftarrow \psi - \eta_2 \nabla_\psi$

Inverse Optimization (Admissible Set Discovery):
Initialize admissible set $\mathcal{A} \leftarrow \emptyset$
for $k = 1$ to $N_{restarts}$ do
    Draw random $x_0' \leftarrow x_0 + \text{Uniform}(-\delta_x, \delta_x)$
    Draw random $\theta' \leftarrow \theta + \text{Uniform}(-\delta_\theta, \delta_\theta)$
    while not converged and $\mathcal{L}_{\text{total}} > \varepsilon_{total}$ do
        Compute predicted output: $\hat{y} \leftarrow M_{\text{fwd}_\phi}(x_0', \theta')$
        Compute data loss: $\mathcal{L}_{\text{data}}(x_0', \theta') \leftarrow \|\hat{y} - y^*\|^2$
        Compute Lyapunov penalty:
            $\mathcal{L}_{\text{Lyapunov}}(x_0', \theta') \leftarrow \max(0, -V_\psi(x_0', \theta')) +$
            $\max(0, \nabla_x V_\psi(x_0', \theta') \cdot M_{\text{fwd}_\phi}(x_0', \theta') +$
            $\alpha V_\psi(x_0', \theta'))$
        Compute total loss: $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{Lyapunov}}$
        Update:
            $x_0' \leftarrow x_0' - \eta_3 \nabla_{x_0'} \mathcal{L}_{\text{total}}$
            $\theta' \leftarrow \theta' - \eta_3 \nabla_{\theta'} \mathcal{L}_{\text{total}}$
    if $\mathcal{L}_{\text{total}} < \varepsilon_{total}$ then
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{(x_0', \theta')\}$
return $\mathcal{A}$
```

Result

RQ1. Does the Lyapunov-based PINN correctly represent the system's stability in accordance with Lyapunov's stability conditions?
RQ2. Does the inverse model return parameter values that enforce system stability?
RQ3. How can we interpret the role of the Lyapunov constraint in guiding the search for stable parameters?

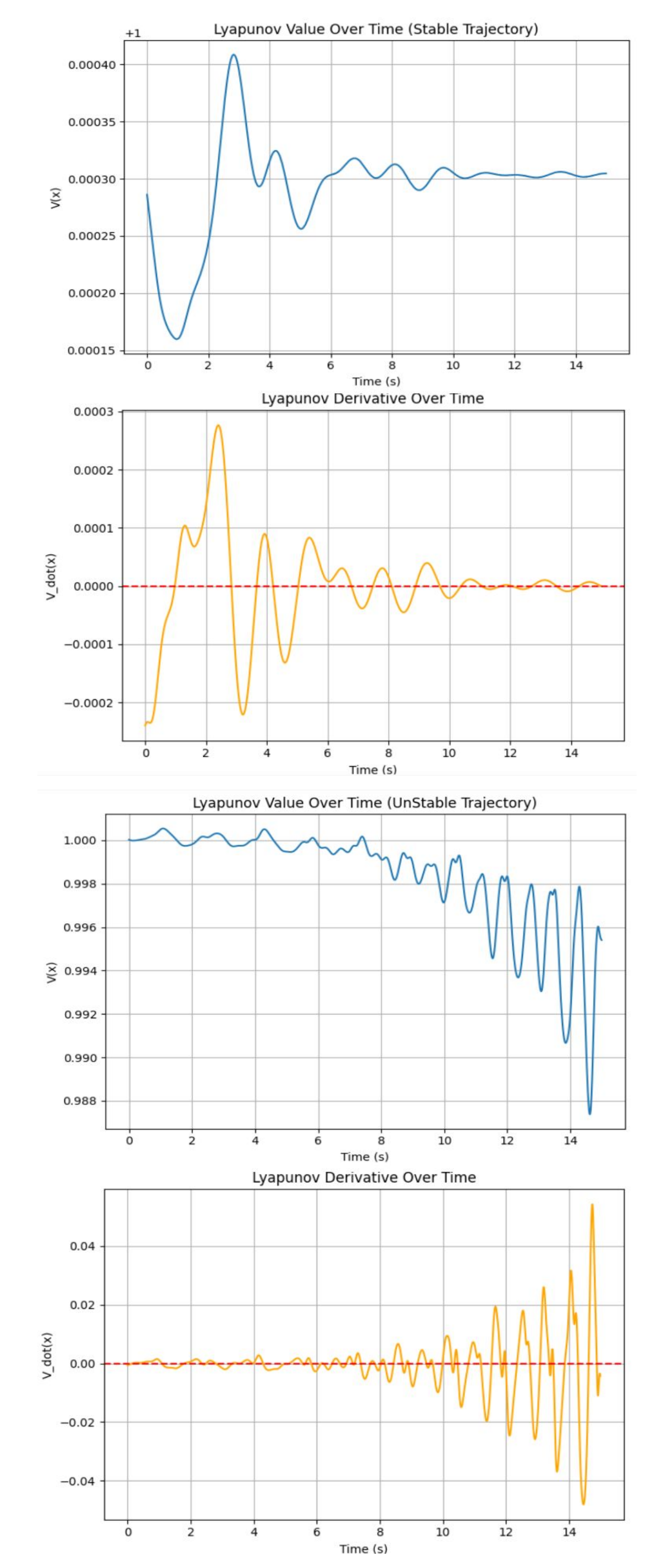

Figure 3. Lyapunpov value and derivative behaviour over the time for unstable and stable sample.

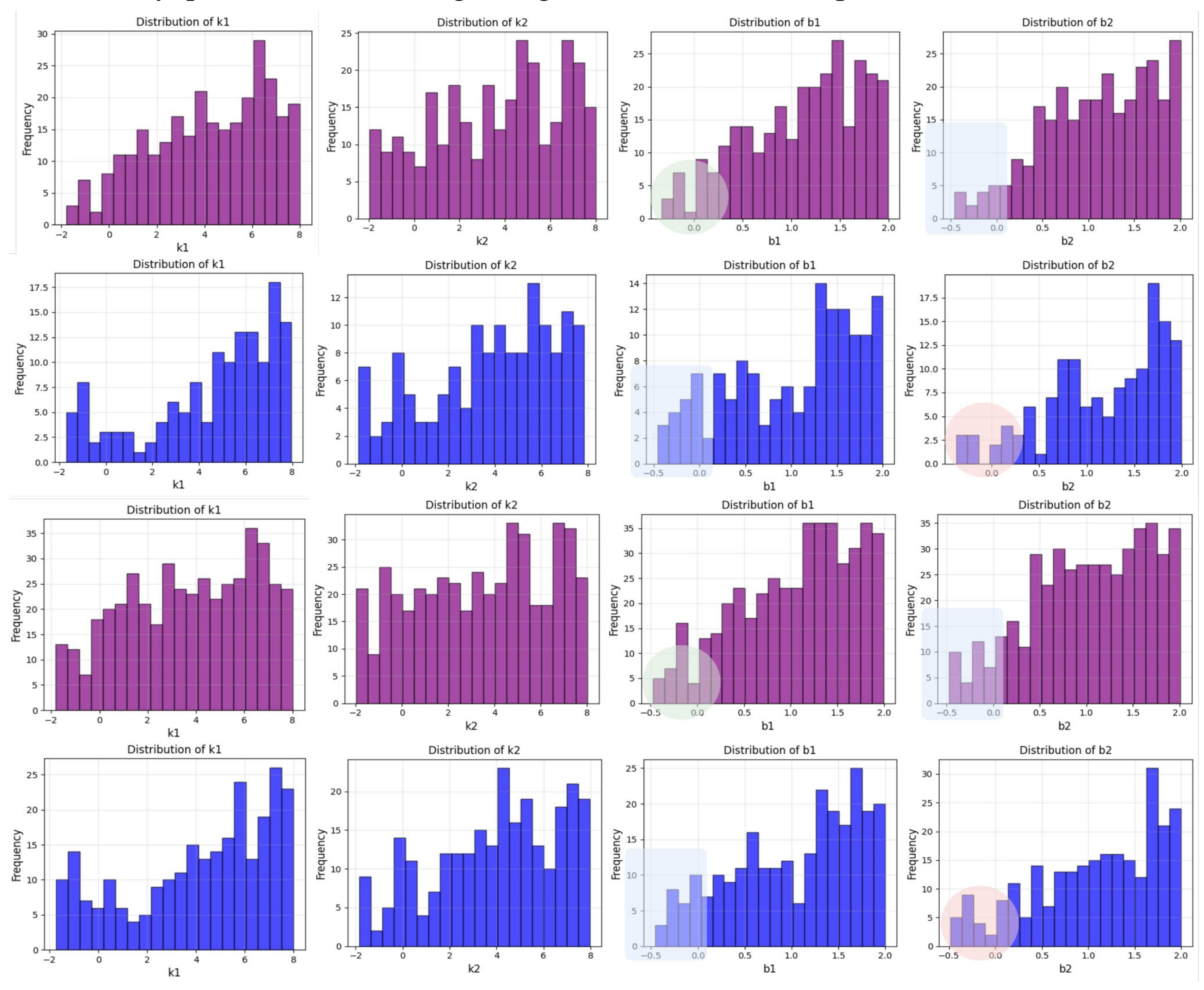

Figure 4. Parameter distribution for reaching stable trajectories. The purple and blue experiments use the initial conditions $[0.5, 1.0, -0.5, -4.0]$ and $[0.25, 0.5, -0.2, -1.0]$, respectively. The first two rows correspond to the total optimization loss in the inverse model, while the remaining rows use only the data loss.

Reference

[1] Jun Liu, et al. TOOL LyZNet: A Lightweight Python Tool for Learning and Verifying Neural Lyapunov Functions and Regions of Attraction. In Proceedings of the 27th ACM International Conference on Hybrid Systems. 2024. doi:10.1145/3641513.3650134

[2] Anthony N Michel, et al. Stability of dynamical systems. Springer. 2008.

[3] Morgan Jones and Matthew M. Peet. Converse Lyapunov Functions and Converging Inner Approximations to Maximal Regions of Attraction of Nonlinear Systems. In 2021 60th IEEE Conference on Decision and Control. 2021. doi:10.1109/CDC45484.2021.9682949

[4] Haoyu Chu, et al. Structure-preserving physics-informed neural networks with energy or Lyapunov structure. In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. 2024. doi: doi:10.24963/ijcai.2024/428

[5] Ivan Jimenez et al. LyaNet: A Lyapunov Framework for Training Neural ODEs. International Conference on Machine Learning. 2022. https://api.semanticscholar.org/CorpusID:246634091