SciML Bootcamp Assignment 6
Universal Differential Equations for Predator–Prey Dynamics

Arihant Raidani

August 10, 2025

Abstract

We study Universal Differential Equations (UDEs) on the Lotka–Volterra predator–prey system by replacing the interaction terms with neural networks trained from synthetic trajectories. Using Lux (with Zygote adjoints) we fit two small feedforward networks to emulate the unknown interaction functions. Results show that the UDE partially reproduces state trajectories—notably for predators—but struggles to match the sharp, early interaction transients for prey. We discuss causes (scale imbalance, identifiability, and transient stiffness), show diagnostics, and outline practical remedies.

1 Introduction

The classical Lotka–Volterra model captures predator–prey oscillations via linear-in-state growth/decay and bilinear interactions:

$$\frac{dx}{dt} = \alpha x - \beta xy, \tag{1}$$

$$\frac{dy}{dt} = \gamma xy - \delta y, \tag{2}$$

where x and y denote prey and predator populations. In a UDE, unknown or partially known physics is replaced by a trainable function (here a neural network), learned from data while keeping known structure intact. We target the interaction terms, substituting $-\beta xy$ and $+\gamma xy$ with neural networks.

2 Data Generation

Synthetic data were generated by numerically solving Eqs. (1)–(2) with

$$\alpha = 0.4, \quad \beta = 0.01, \quad \gamma = 0.2, \quad \delta = 0.02,$$

initial condition $x(0) = 40$, $y(0) = 9$, time span $t \in [0, 8]$, and sampling $\Delta t = 0.1$ using `Tsit5()`.

3 UDE Formulation

We retain the linear known terms and replace interactions with neural networks $\mathcal{N}_1, \mathcal{N}_2$ taking (x, y) as input:

$$\frac{dx}{dt} = \alpha\, x \; + \; \mathcal{N}_1(x, y), \tag{3}$$

$$\frac{dy}{dt} = -\delta\, y \; + \; \mathcal{N}_2(x, y). \tag{4}$$

Each network is a small MLP in Lux:

$$\texttt{Dense(2,16,ReLU)} \to \texttt{Dense(16,16,ReLU)} \to \texttt{Dense(16,1)}.$$

We train by minimizing mean-squared error between UDE trajectories and the synthetic trajectories at sampled times.

4 Training Setup

Framework. DifferentialEquations.jl + Lux.jl + Zygote, optimized via `Optimization.jl` (ADAM).
Adjoints. Zygote-based adjoint sensitivities (QuadratureAdjoint), CPU.
Hyperparameters. Learning rate 10^{-3}, 5000 iterations; solver tolerances $\texttt{abstol} = \texttt{reltol} = 10^{-6}$.
Runtime. $\sim$3 hours on CPU (due to repeated forward solves and adjoint backprop).

5 Results

5.1 Interaction Terms: True vs Learned

Figure 1 compares the true interaction terms $-\beta xy$ and $+\gamma xy$ with the UDE's learned terms (the networks' outputs). The learned terms are relatively flat and under-represent the sharp early-time peaks that occur when both x and y are simultaneously large.

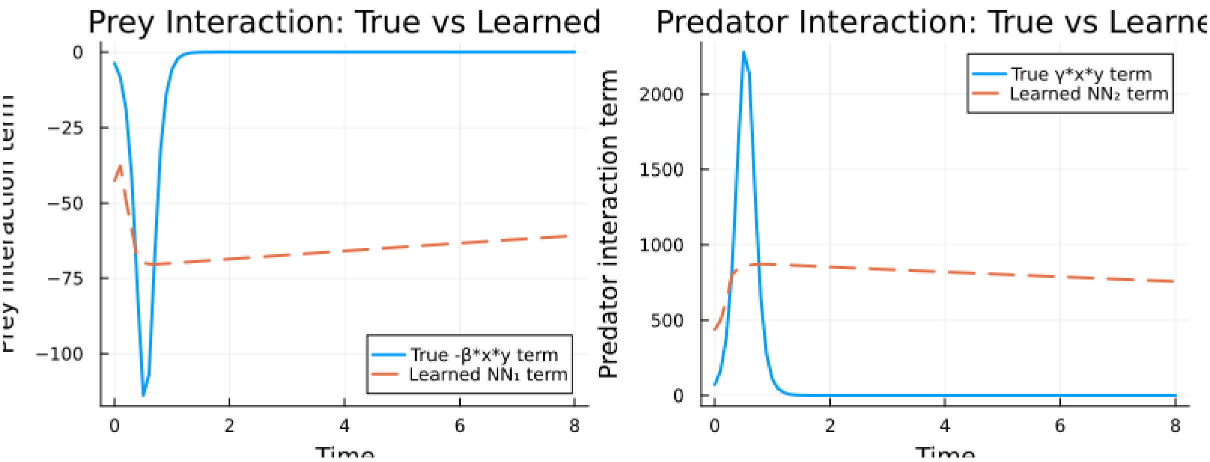

Figure 1: **Interaction terms.** Left: prey equation interaction ($-\beta xy$) vs learned $\mathrm{NN}_1(x, y)$. Right: predator equation interaction (γxy) vs learned $\mathrm{NN}_2(x, y)$. The UDE underfits the sharp early transient.

5.2 State Trajectories

Figure 2 shows prey and predator trajectories from the true model and the trained UDE. Predator dynamics are captured reasonably well after the initial transient. Prey shows a notable mismatch (including negative values), indicating that interaction learning was insufficient to stabilize early dynamics in the prey channel.

5.3 Training Diagnostics

Figure 3 plots the training loss versus iteration. The large spike at the beginning reflects the initial mismatch on the transient; loss then decreases and plateaus.

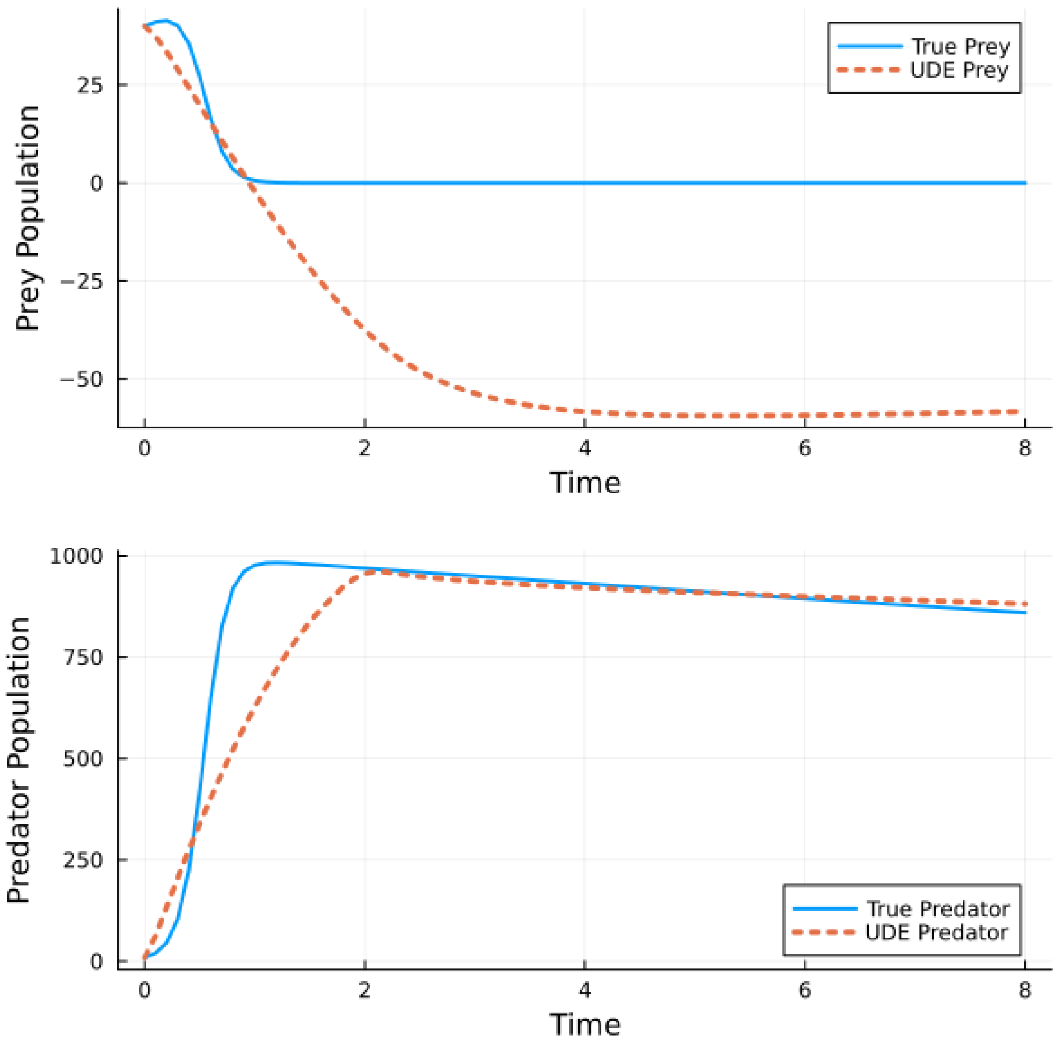

Figure 2: **Trajectories.** Top: prey $x(t)$; Bottom: predator $y(t)$. The UDE tracks predators better than prey; early prey transient is not well captured, leading to drift and negativity.

6 Discussion

Effectiveness. The UDE succeeds at partially reproducing the system's behavior: the predator trajectory is approximated fairly well after the transient. However, the learned interaction functions do not recover the sharp early peaks, and the prey trajectory deviates substantially.

Why the underfit?

- **Scale imbalance.** Predator values reach $\mathcal{O}(10^3)$, whereas prey is $\mathcal{O}(10^1)$; unweighted MSE lets predator dominate, so the optimizer prioritizes fitting $y(t)$.
- **Transient sensitivity.** The product xy is largest at early times, creating a narrow, stiff peak that's hard to match without explicit regularization or time-weighted loss.
- **Identifiability.** With α and δ fixed, the networks must explain all coupling via $\mathcal{N}_{1,2}$. Multiple functions can fit trajectories similarly post-transient, so the early-time structure is not uniquely enforced.

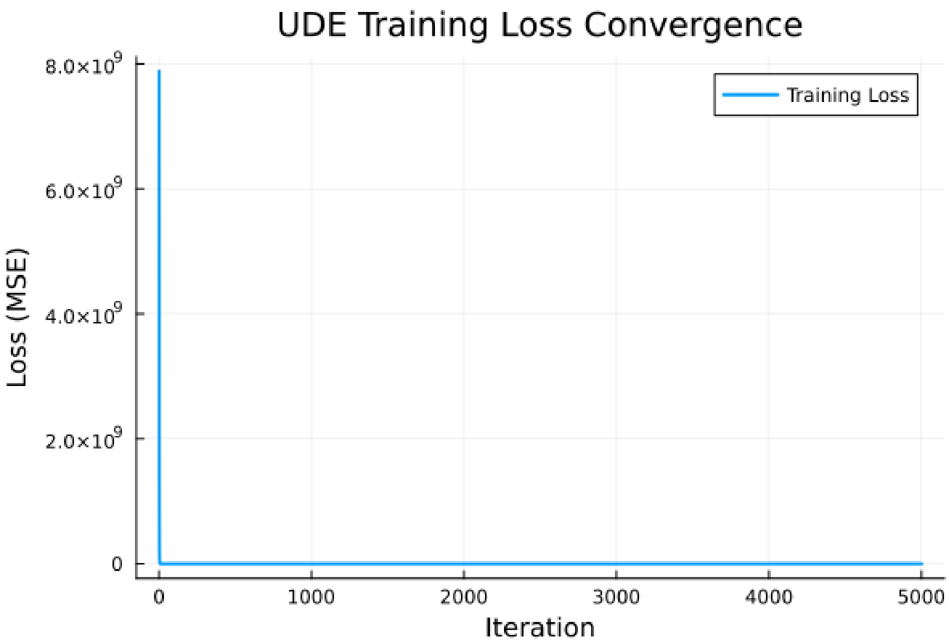

Figure 3: **Training loss.** MSE over iterations. Loss rapidly decreases from a large initial value and then plateaus.

Simple remedies (for future work).

- **Loss balancing/standardization:** standardize x, y or weight prey higher to avoid predator dominance.
- **Physics-informed constraints:** enforce $\mathcal{N}_1(x, y) \approx -\beta(x, y)\, xy$ and $\mathcal{N}_2(x, y) \approx +\gamma(x, y)\, xy$ with nonnegativity on β, γ (e.g., softplus) to prevent sign errors and negative prey.
- **Adjoint/solver settings:** looser tolerances or `InterpolatingAdjoint` to iterate faster; curriculum on time grids (fit transient first).
- **Model capacity & regularization:** mildly deeper/wider nets with L2 or spectral norm penalties to capture peaks without overfitting noise.

7 Conclusions

Embedding neural networks into mechanistic ODEs allows data-driven refinement of interaction laws. On this problem, the UDE captured broad predator dynamics but underfit sharp prey transients, highlighting practical issues (scaling, transients, identifiability). With straightforward adjustments (loss balancing, physics-consistent parameterization, and modest regularization), we expect the UDE to better recover the true interactions.

Reproducibility (Key Settings)

- Packages: `DifferentialEquations.jl`, `Lux.jl`, `Zygote.jl`, `Optimization.jl`, `ComponentArrays.jl`, `Plots.jl`.
- Data: Lotka–Volterra with $(\alpha, \beta, \gamma, \delta) = (0.4, 0.01, 0.2, 0.02)$; $u_0 = (40, 9)$; $t \in [0, 8]$; $\Delta t = 0.1$.
- UDE nets: two MLPs: $(2 \to 16 \to 16 \to 1)$ with ReLU.

- Training: ADAM (lr 10^{-3}), 5000 iterations; `Tsit5()` solver; `abstol`=`reltol`=10^{-6}; Zygote adjoints (QuadratureAdjoint); CPU; runtime ∼3 h.
- Figures: `interaction_terms.png`, `trajectories_comparison.png`, `training_loss_curve.png`.