Comparative Evaluation of Extended DMD and Neural ODEs for Trajectory Modeling

In Linearly Immersible and Chaotic Dynamical Systems

Kieran Vlahakis[1], Elizabeth Won[1], Anish Goel[1], Sidd Ojha[1]
[1]California Institute of Technology

ACM 270: Data-Driven Modeling of Dynamical Systems
Spring 2025

Project Objective

- Goal: Explore different methods of modeling non-linear dynamics using tools from this course.
- Method: Compare EDMD vs Neural ODEs on linearly immersible and chaotic systems.
- Systems used:
  - Lotka–Volterra Predator–Prey Model (linearly immersible)
  - Lorenz System (non-linearly immersible)
- Evaluate trade-offs in reconstruction fidelity across both scenarios.

Extended Dynamic Mode Decomposition (EDMD)

Dynamic Mode Decomposition (DMD) operates on time series data:

- $x_1, \ldots, x_{m+1} \in \mathbb{R}^n$
- Form snapshot matrices:

$$\mathbf{X} = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_m \\ | & & | \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} | & & | \\ x_2 & \cdots & x_{m+1} \\ | & & | \end{bmatrix}$$

- DMD seeks $\tilde{A}$ such that $\mathbf{Y} \approx \tilde{A}\mathbf{X}$ via SVD:

$$\tilde{A} = \mathbf{Y} V_r \Sigma_r^{-1} U_r^T$$

From DMD to EDMD

- EDMD lifts data into a higher-dimensional feature space via $\psi : \mathbb{R}^n \to \mathbb{R}^d$
- Define lifted snapshot matrices:

$$\mathbf{X} = \begin{bmatrix} | & & | \\ \psi(x_1) & \cdots & \psi(x_m) \\ | & & | \end{bmatrix} \in \mathbb{R}^{d \times m}$$

$$\mathbf{Y} = \begin{bmatrix} | & & | \\ \psi(x_2) & \cdots & \psi(x_{m+1}) \\ | & & | \end{bmatrix} \in \mathbb{R}^{d \times m}$$

- Solve:

$$A = \arg\min_{\hat{A}} \|\mathbf{Y} - \hat{A}\mathbf{X}\|_F^2 \Rightarrow A = \mathbf{Y}\mathbf{X}^+$$

EDMD and Kernel DMD (KDMD)

- EDMD reduces to DMD when $\psi = \text{identity}$.
- Use SVD on $\mathbf{X}$:

$$\mathbf{X} \approx U_r \Sigma_r V_r^T$$

- Reconstructed operator:

$$A = \mathbf{Y} V_r \Sigma_r^{-1} U_r^T$$

- Complexity: $\mathcal{O}(dmr)$
- **Two regimes:**
  - EDMD: $d \ll m$ (few features, many samples)
  - KDMD: $d \gg m$ (many features, few samples)

Sensitivity Methods for Learning ODEs (1/2)

Optimize-then-Discretize: Continuous-Time Sensitivities

- Treat the model as continuous in time:

$$L(\theta) = \int l(z(t), \theta)\, dt$$

- Differentiate under the integral sign:
  - **Forward sensitivity:** augment $\dot{z} = f(z, \theta)$ with $\frac{\partial z}{\partial \theta}$ equations
  - **Adjoint sensitivity:** solve for $\lambda(t)$ backwards in time:

$$\dot{\lambda} = -\left(\frac{\partial f}{\partial z}\right)^T \lambda, \quad \lambda(T) = \frac{\partial l}{\partial z}(z(T))$$

- Compute $\nabla_\theta L$ by integrating $\frac{\partial f}{\partial \theta}$ against $\lambda(t)$
- Efficient for high-dimensional θ, low memory via checkpointing

Sensitivity Methods for Learning ODEs (2/2)

Discretize-then-Optimize: Discrete Backpropagation

- Apply fixed-step integrator to produce discrete dynamics:

$$z_{n+1} = \Phi_h(z_n; \theta)$$

- Use backpropagation or automatic differentiation over n steps to compute $\nabla_\theta L$
- Can use discrete adjoint formulas or autodiff
- Simpler to implement, but biased if step size h is large or variable

In our project: We use optimize-then-discretize with continuous-time adjoint sensitivity. This enables adaptive solvers and avoids memory costs from full checkpointing.

Scoring Dynamics Modeling

Root Mean Square Error (RMSE):

- Measures difference between predicted $\hat{x}^{(i)}[t]$ and ground truth $x^{(i)}[t]$.
- Averaged over $M = 10$ trajectories and $T = 200$ time steps.
- Initial conditions sampled uniformly from system's state space.
- Equation:

$$\text{RMSE}[t] = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left\|\hat{x}^{(i)}[t] - x^{(i)}[t]\right\|_2^2}$$

Lotka–Volterra Predator–Prey Model

- Nonlinear system describing coupled species (prey x, predator y).
- Prey grows exponentially (αx), predator preys on prey ($-\beta xy$).
- Predator reproduces from prey (δxy), dies naturally ($-\gamma y$).
- Equations:

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y$$

Lotka–Volterra Experimental Parameters

- Ground truth parameters:
  - $\alpha = 1.1$, $\beta = 0.4$, $\delta = 0.1$, $\gamma = 0.4$
- Initial conditions sampled uniformly from $[1, 6]^2$.
- Simulated with time step $\Delta t = 0.025$ for time horizon $T = 25.0$.

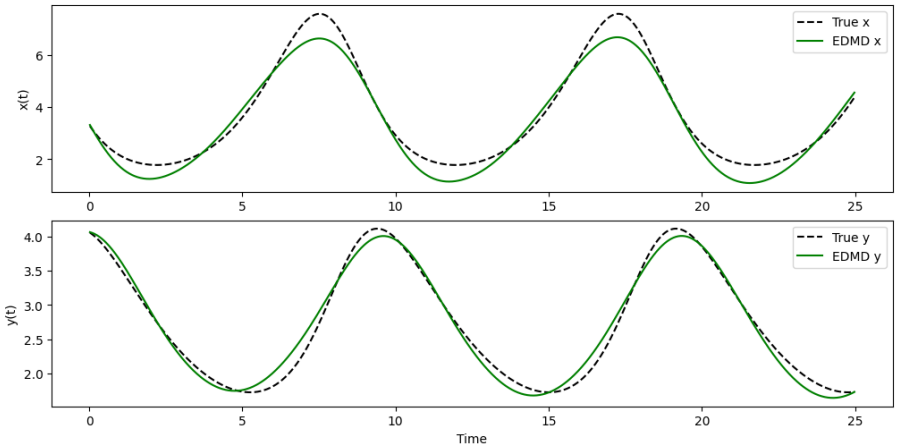
Modeling LV Dynamics: Kernelized EDMD (1/2)

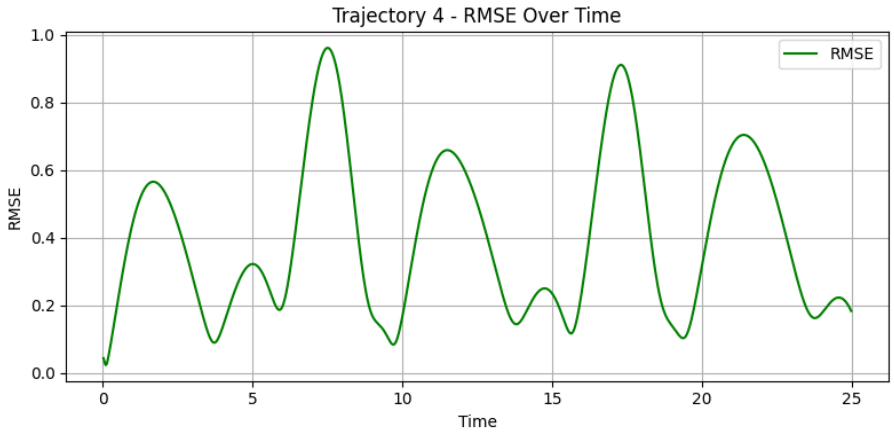
Method Overview:

- Use of kernelized EDMD to capture nonlinear dynamics.
- Kernel trick avoids explicit lifting or basis construction.
- Compared RBF and Polynomial kernels:
  - RBF outperformed Poly, even at various degrees.
  - Chose RBF for final experiments (projects into infinite-dimensional space).
- SVD truncation used (rank 10) to ensure numerical stability.

Modeling LV Dynamics: Kernelized EDMD (2/2)

Visual Results:

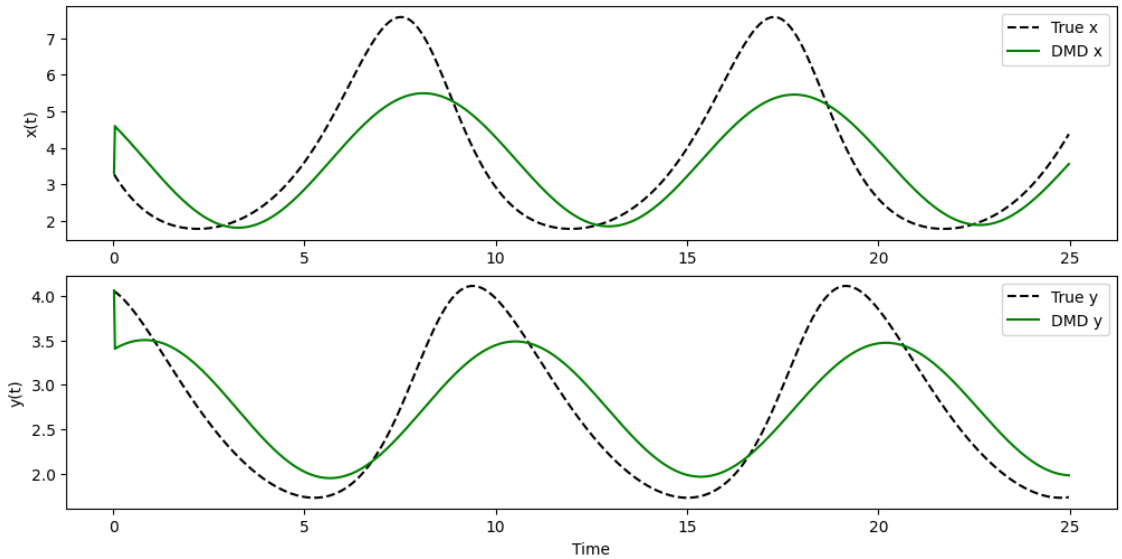




- EDMD tracks the LV trajectory closely.
- Outperforms classic DMD, which misses initial conditions.
- Low RMSE supports fidelity of EDMD (right).

Classic DMD Comparison

Limitations of Standard DMD:



- Fails to capture true dynamics.
- Poor initial condition fit.
- Shows consistent underperformance across all trajectories.

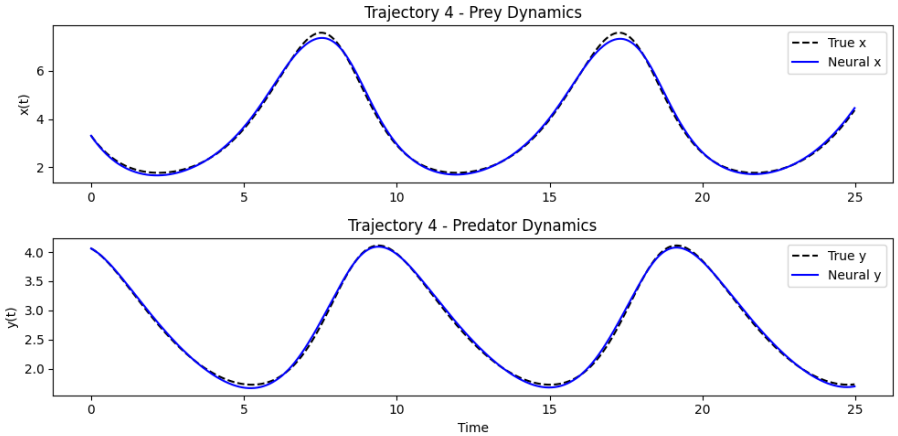
Modeling LV Dynamics: Neural ODE (1/2)

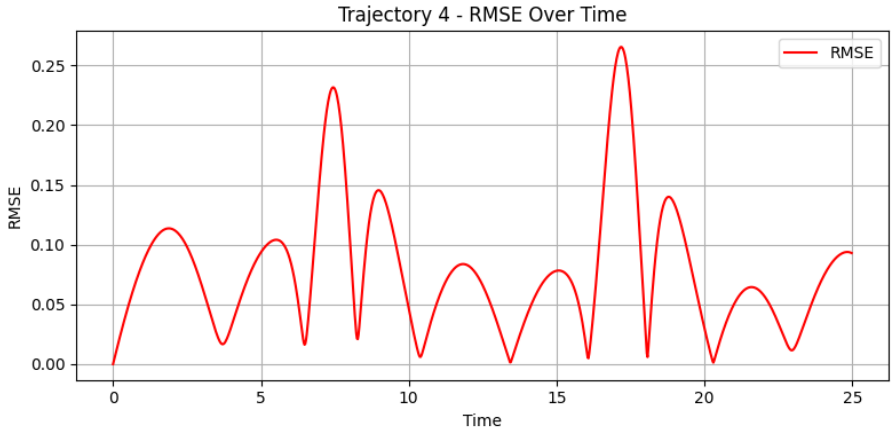
Model Structure:

- Learn $f_\theta(z) \approx \frac{dz}{dt}$ via neural net.
- Input: $z \in \mathbb{R}^2$ (populations), Output: $\frac{dz}{dt}$
- Architecture: 1 hidden layer, 64 units, tanh activation.
- Trained using same data and setup as EDMD.

Modeling LV Dynamics: Neural ODE (2/2)

Visual Results:

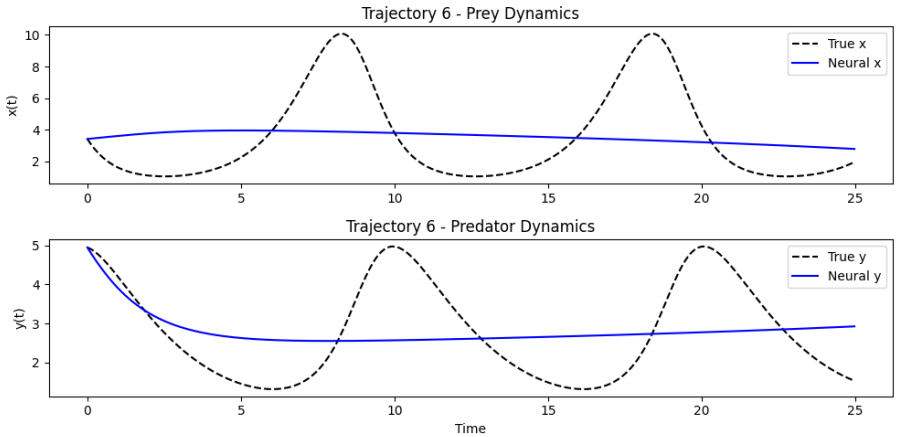


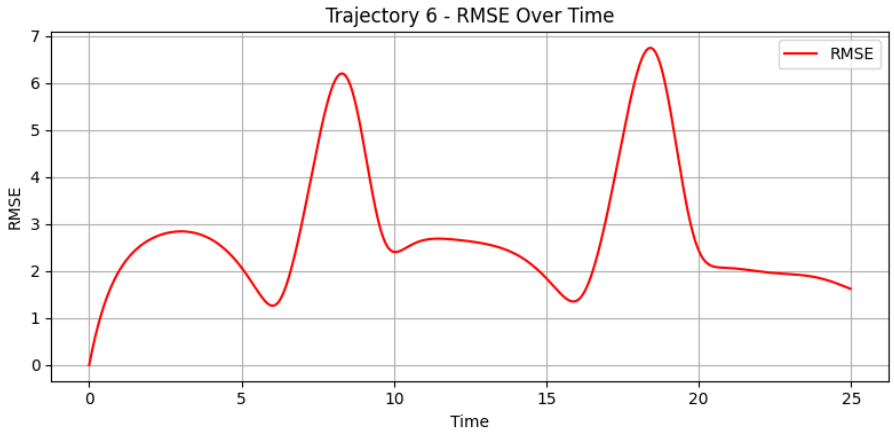


- Closely fits trajectory; low RMSE.
- Similar RMSE shape as EDMD suggests shared difficult regions.
- Performance varies per trajectory.

Neural ODE: Failure Case

Example of Poor Generalization:

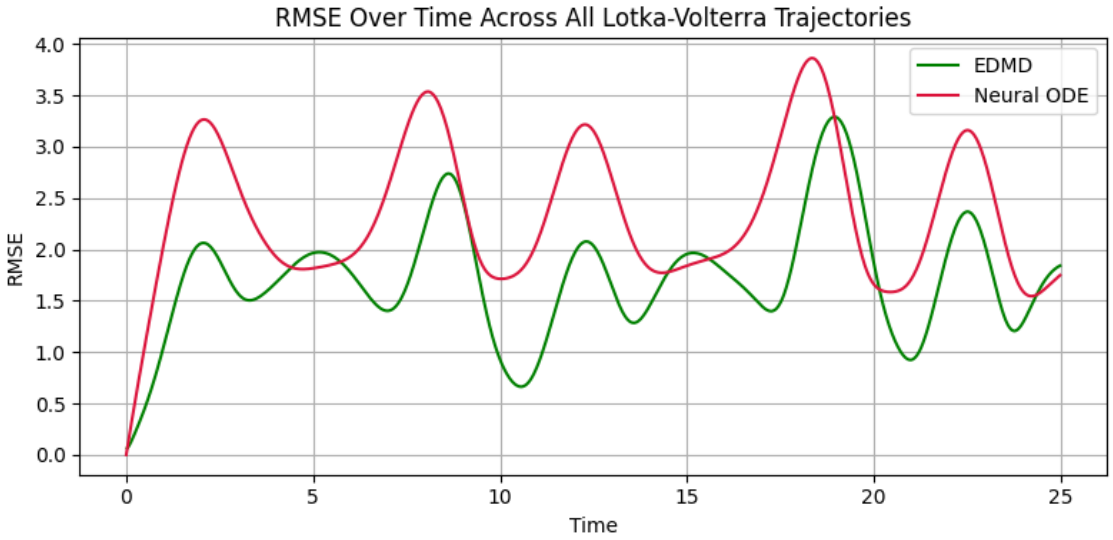




- Failed to capture oscillations.
- Smoothed through periodic structure.
- Possibly trapped in poor local optimum.

Lotka–Volterra: Method Comparison

Aggregate Performance:



- Average RMSE is comparable.
- EDMD is more stable across trajectories.
- Neural ODEs are more variable (hit or miss).

Lorenz System

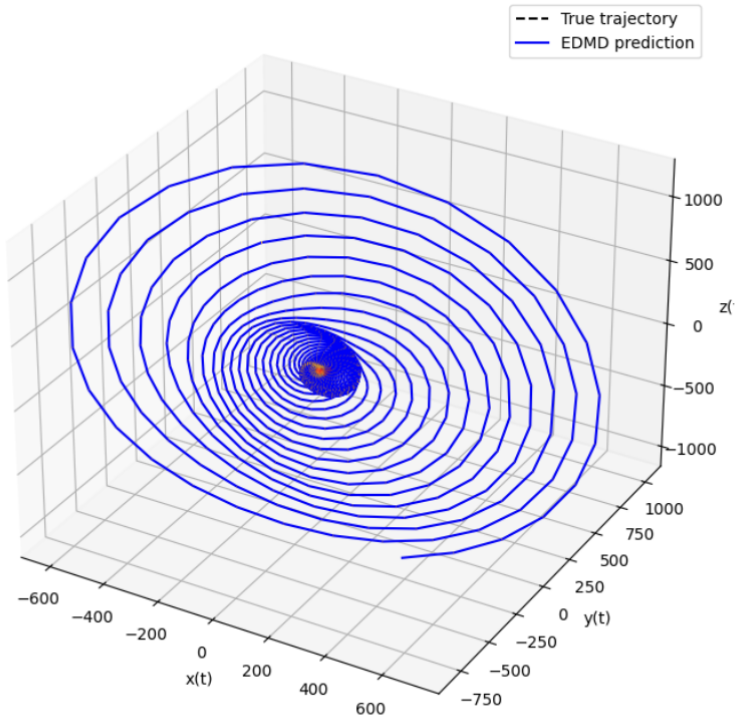
- 3D system modeling convective atmospheric flow.
- Equations:

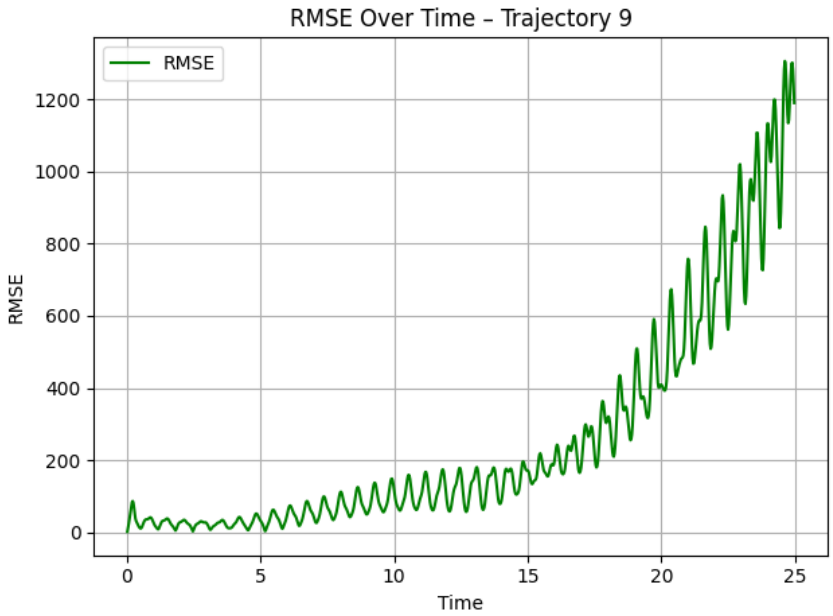
$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dz}{dt} = xy - \beta z$$

- Parameters used: $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$ (classical chaotic setting).
- Initial conditions sampled from $[-5, 5]^3$, forward evolved for $T = 25.0$, $\Delta t = 0.025$.

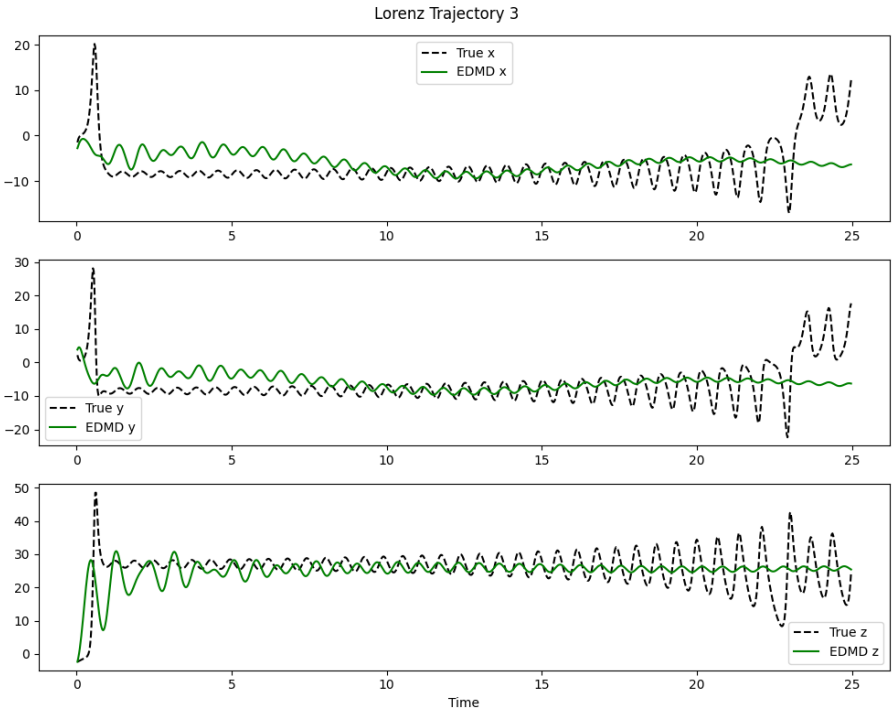
Modeling Lorenz: EDMD (1/2)

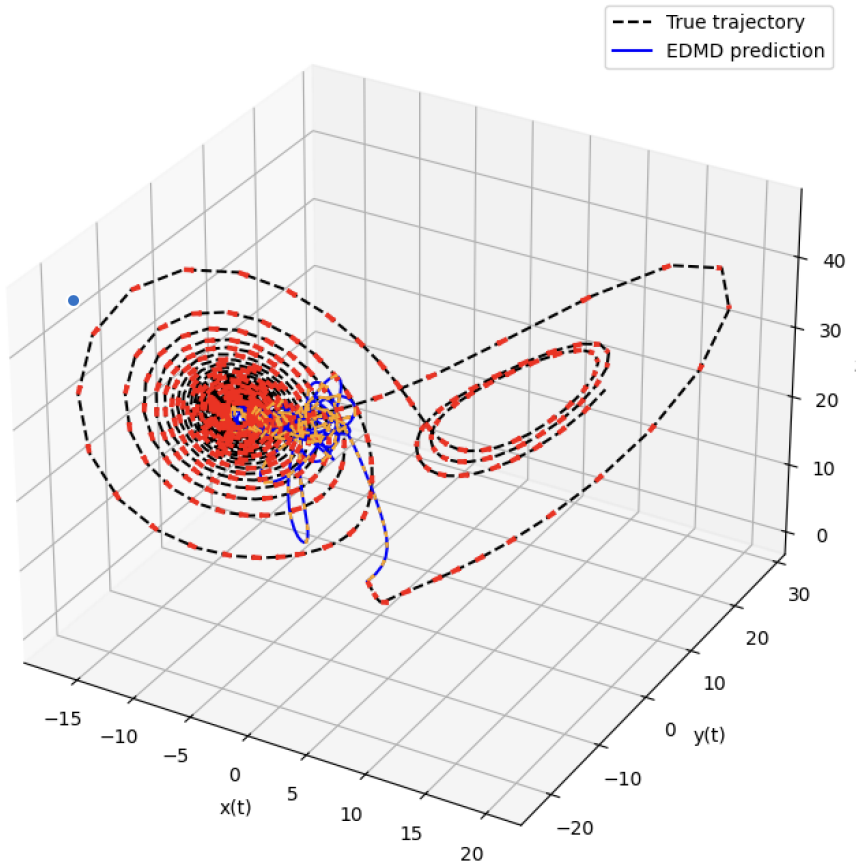
- Same setup as LV system.
- One trajectory diverged badly under RBF kernel.
- Polynomial kernel (degree 2) avoided this instability.
- Expressiveness of RBF may harm in chaotic regimes.





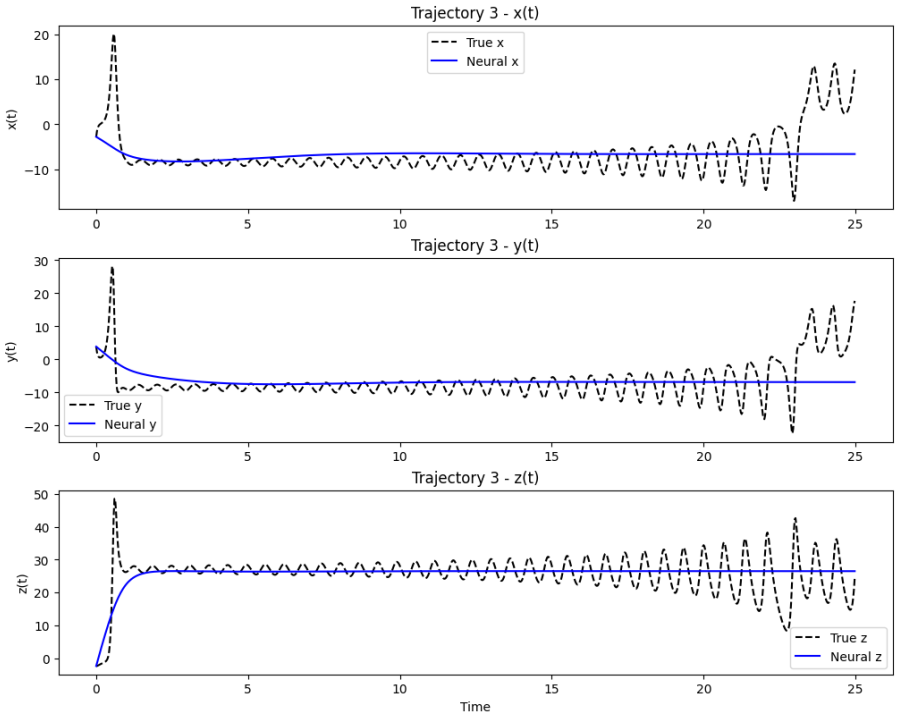
Modeling Lorenz: EDMD (2/2)

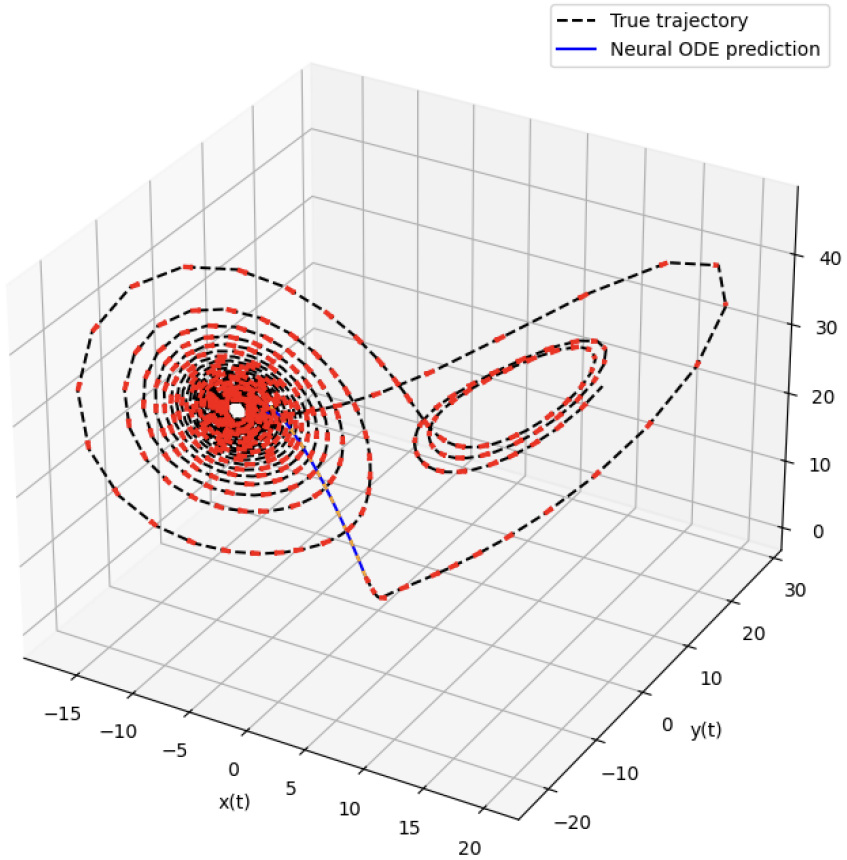




- Captures general attractor location.
- Fails to reproduce fine-grained motion.
- Reconstruction meanders around attractor.

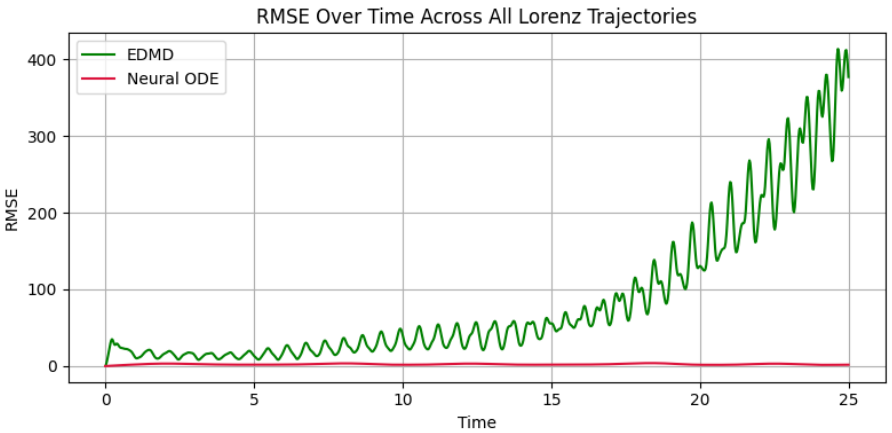
Modeling Lorenz: Neural ODE (1/2)

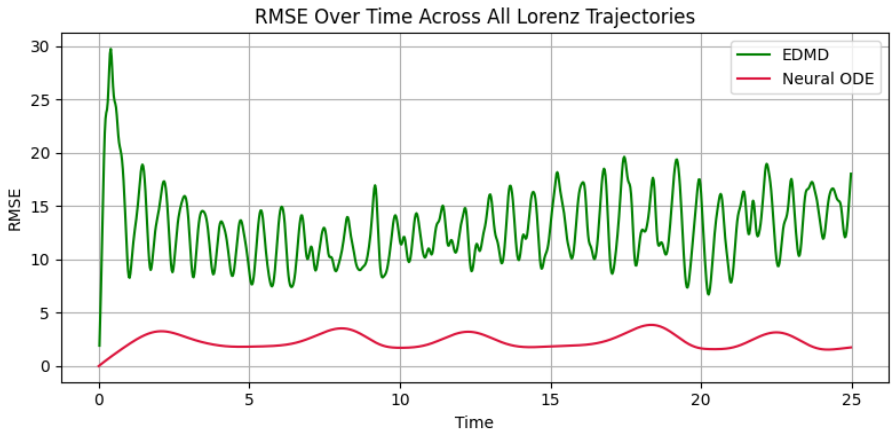




- Neural ODE jumps to center of attractor.
- Avoids transient behavior (oscillations).
- Model finds local minimum that misses short-term dynamics.

Lorenz: Method Comparison





- Including all trajectories: EDMD appears worse.
- If we exclude the outlier: Neural ODE slightly outperforms.
- EDMD meanders; Neural ODE oversimplifies.

Discussion and Conclusion

- **LV System:** EDMD stable, Neural ODE variable but potentially more accurate.
- **Lorenz:** Both struggled — EDMD diverged/meandered, Neural ODE oversimplified.
- Tradeoff: EDMD offers robustness, Neural ODE offers expressiveness.
- Future: Explore hybrid models combining strengths of both.