Data-Driven Identification of Ordinary Differential Equations

Alexis Guevara
Department of Mathematics and Statistics
California State University Long Beach

August 18, 2025

Importance of Data-Driven Identification

Modern Challenge

- Many systems lack theoretical models but generate abundant data
  - Example: Biological rhythms, engineering vibrations, financial systems

Critical Need

- Extract system dynamics through underlying equations directly from observations
  - Enables prediction, control, and understanding of complex dynamics

Challenges

Key Challenges in System Identification

- *Noise Corruption*: Small measurement errors corrupt derivative calculations
- *Sparsity*: Real dynamics are often sparse (few key terms) but hard to isolate
- *Partial Observations*: Critical state variables may be unmeasured

Research Objectives

Impact of Thesis

Robust Equation Discovery

- Automatically identify the few relevant terms in differential equations from data
- Balance model simplicity (to avoid overfitting) and accuracy

Handling Real-World Constraints

- Reconstruct hidden dynamics when only partial measurements exist
- Maintain performance with noisy or forced systems

Outcome Insights

- Validate methods on synthetic benchmarks and collected data in the real world

Overview

1. Theoretical Foundations
2. Methodology
3. Results
4. Discussion
5. Conclusion

Ordinary Differential Equations

Definition

Ordinary differential equations (ODEs) are equations that contain only one independent variable and one or more of its derivatives with respect to the variable. The general form of ODEs is

$$F(t, x, x', x'', \ldots, x^{(n)}) = 0 \tag{1}$$

where t is the independent variable, x is the dependent variable and $x^{(k)}$, for $k = 1, \ldots, n$, is the k^{th} derivative of x.

Key Properties

- *Order* - Highest derivative in the equation
- *Initial Conditions* - Values of the function and its derivatives at a specific point
- *Linear* - Dependent variable and its derivatives only appear only to the first power

Van der Pol Oscillator

Nonlinear ODE Equation Forms

$$\begin{aligned} \text{Second-Order Equation: } & \frac{d^2x}{dt^2} = \mu(1-x^2)\frac{dx}{dt} - x \\ \text{First-Order System of Equations: } & \frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1-x^2)v - x \end{aligned} \tag{2}$$

with time variable t, position x, velocity v, and scalar parameter μ controlling the nonlinearity and strength of damping.

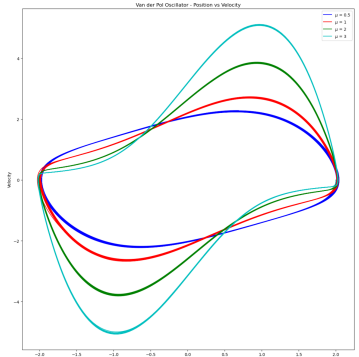
Benchmark Utility

- Simple for analysis, rich enough for real-world relevance
- Nonlinear damping term $\mu(1-x^2)v$ makes equation have no analytic solution

Van der Pol Oscillator

Key Features

- *Limit cycles*: Stable periodic orbits for $\mu > 0$ for all initial conditions
- *Applications*: Electrical circuits, neuronal activity, cardiac rhythms



(a) Van der Pol Oscillator Behavior

Sparse Identification (SINDys)

Sparse Identification of Nonlinear Dynamics (SINDys)

- Data-driven method to identify and model nonlinear dynamic systems
- Introduced by Brunton, Proctor, Kutz in 2016

Core Idea

- Assume governing equations are sparse (few active terms) in a library of candidate functions (e.g., polynomials, trigonometric terms)

Sparse Identification (SINDys)

Purpose

- Identify underlying system dynamics from data
- Extract sparse representations of complex systems

Key Features

- Applicable for linear and nonlinear dynamic systems
- Reduces computational complexity

SINDy Workflow

$$\text{Data} \rightarrow \text{Function Library} \rightarrow \text{Sparse Regression} \rightarrow \text{Identified ODEs}$$

Multiple Linear Regression (MLR)

Definition

Multiple linear regression (MLR) models the relationship between one dependent variable and multiple independent variables

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon \tag{3}$$

where Y is the dependent variable, $X_1, X_2, ..., X_n$ are the independent variables, $\beta_0, \beta_1, ..., \beta_n$ are the coefficients, and ϵ is the error term.

Purpose

- Find best-fit linear equation describing relationship between variables
- Solve via ordinary least squares (OLS): $\min_{\beta} \| Y - X\beta \|_2^2$

Multiple Linear Regression (MLR)

	coef
x	0.3226
sin(x)	-0.5156
cos(x)	-7.2563
v	1.0209
sin(v)	-0.0169
cos(v)	0.0555
x^2	-1.5267
x sin(x)	-2.4244
x cos(x)	0.1896
x v	0.0805
x sin(v)	-0.0470
x cos(v)	-0.0755
sin(x)^2	3.8478
sin(x) cos(x)	-0.0526
sin(x) v	-0.1584
sin(x) sin(v)	0.0848
sin(x) cos(v)	0.1336
cos(x)^2	3.4717
cos(x) v	-0.0278
cos(x) sin(v)	-0.0199
cos(x) cos(v)	0.0095
v^2	0.0253
v sin(v)	0.0058
v cos(v)	-0.0069
sin(v)^2	3.6578
sin(v) cos(v)	0.0108
cos(v)^2	3.6616

OLS Table: Terms and Coefficients

Multiple Linear Regression (MLR)

Advantages

- Results are easy to understand and interpret
- Effectively uses all available data for predictions
- Unbiased estimates for model dynamics

Limitations

- Overfits with many candidate terms
- Fails to enforce sparsity (retains irrelevant terms)
- Outliers in data can significantly affect the model
- High correlation among independent variables can lead to unreliable coefficients

Least Absolute Shrinkage and Selection Operator (LASSO)

Definition

Least Absolute Shrinkage and Selection Operator (Lasso) performs variable selection and regularization to improve the accuracy and interpretability of the model.

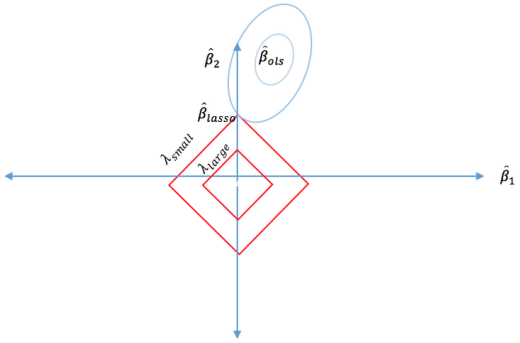
$$\hat{\beta} = \min_{\beta} \left(\sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_1 - ... - \beta_n X_n)^2 + \lambda \sum_{j=1}^{p-1} |\beta_j| \right) \tag{4}$$

where Y_i is the dependent variable, $X_1, X_2, ..., X_n$ are the independent variables, $\beta_0, \beta_1, ..., \beta_n$ are the coefficients, p is the total number of predictors, λ is the penalty parameter, and $\hat{\beta}$ is the optimal set of coefficients that fit the data well while keeping the model simple.

Least Absolute Shrinkage and Selection Operator (LASSO)

Key Innovation

- Adds L_1 penalty
- Sparsity: Drives irrelevant terms to exactly zero



(a) Lasso Visualization

Least Absolute Shrinkage and Selection Operator (LASSO)

	coef
x	0.3226
sin(x)	-0.5156
cos(x)	-7.2563
v	1.0209
sin(v)	-0.0169
cos(v)	0.0555
x^2	-1.5267
x sin(x)	-2.4244
x cos(x)	0.1896
x v	0.0805
x sin(v)	-0.0470
x cos(v)	-0.0755
sin(x)^2	3.8478
sin(x) cos(x)	-0.0526
sin(x) v	-0.1584
sin(x) sin(v)	0.0848
sin(x) cos(v)	0.1336
cos(x)^2	3.4717
cos(x) v	-0.0278
cos(x) sin(v)	-0.0199
cos(x) cos(v)	0.0095
v^2	0.0253
v sin(v)	0.0058
v cos(v)	-0.0069
sin(v)^2	3.6578
sin(v) cos(v)	0.0108
cos(v)^2	3.6616

(a) Modeling with MLR

	coef
x	0.3226
sin(x)	-0.5156
cos(x)	-7.2563
v	1.0209
sin(v)	-0.0169
cos(v)	0.0555

(b) Modeling with LASSO

Sparse Regression Results: Terms and Coefficients

Least Absolute Shrinkage and Selection Operator

Advantages

- Discovers simple equations and creates interpretable models
- Robust to noise (acts as implicit filter)

Limitations

- Can significantly shrink important coefficients too much
- Requires fine tuning of penalty parameter λ

Takens' Embedding Theorem

Reconstructing Hidden Dynamics

Challenge: Real-world data often lacks full state variable measurements

Theorem

- For a d-dimensional system, $m > 2d + 1$ delays of a single observable variable preserve the system's topology
- Guarantees a one-to-one mapping of the true state space

Importance

- Rebuild system phase space using collected data
- Ensure a robust framework to analyze unmeasured state variables

Practical Implementation of Takens' Embedding Theorem

Steps to Embed:

1. Select time-series data of single observable variable
2. Choose delay τ and embedding dimension m
3. Construct library using delays: $\Theta = [x(t), x(t-\tau), \ldots, x(t-(m-1)\tau)]$

Benefits

- Reduce noise by distributing errors across dimensions
- Enhances model identification accuracy

Takens' Theorem Process

$$\text{Time Series } x(t) \rightarrow \text{Delay Embedding } [x(t), x(t-\tau), x(t-2\tau)] \rightarrow \text{Phase Portraits}$$

Identification Process Steps

Step 1 - Data Generation

Step 2 - Estimating Derivatives

Step 3 - Building the Function Library

Step 4 - Hybrid Sparse Regression

Step 5 - Model Validation

Data Generation

Van der Pol System

- Define parameters $\mu > 0$, step size Δt, time interval $t \in [a, b]$ where $a < b$
- Solve numerically using 5th-order Runge-Kutta (RK45):

$$\begin{aligned} \text{Second-Order Equation: } & \frac{d^2x}{dt^2} = \mu(1 - x^2)\frac{dx}{dt} - x \\ \text{First-Order System of Equations: } & \frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1 - x^2)v - x \end{aligned} \tag{5}$$

Perturbations

- Additive Noise: Gaussian Distribution
- Forcing Terms: Linear, Periodic

Derivative Calculation

Numerical Derivatives: Forward Euler Method

$$\begin{aligned} \frac{dx}{dt} &= \frac{\Delta x}{\Delta t} = \frac{x_{i+1} - x_i}{t_{i+1} - t_i} \\ \frac{dv}{dt} &= \frac{\Delta v}{\Delta t} = \frac{v_{i+1} - v_i}{t_{i+1} - t_i} \end{aligned} \tag{6}$$

Iterative Method:

$$\begin{aligned} x_{i+1} &= x_i + \frac{dx}{dt} \cdot \Delta t \\ v_{i+1} &= v_i + \frac{dv}{dt} \cdot \Delta t \end{aligned} \tag{7}$$

Key Features

- *Pros*: Computationally efficient
- *Cons*: Amplifies high-frequency noise

Function Library Construction

Function Library terms

- Functions representing system dynamics
  - Polynomial (e.g. x,x^2,x^3)
  - Trigonometric (e.g $\sin(x), \cos(x)$)
- Combinations of function types (ex. $x \cdot sin(x), x \cdot v^2$)

Delay Embedding (Takens'):

- For partial observations: $[x(t), x(t-\tau), \ldots, x(t-(m-1)\tau)]$
- Delay τ and embedding dimension m

Matrix Diagram

Columns = Candidate Terms

Rows = Data time points

Sparse Regression (LASSO + MLR)

Hybrid Sparse Regression

1. LASSO
   - Identify active terms (sparsity)
   - Adjust penalty parameter λ to control model complexity
2. OLS Refinement
   - Re-fit selected terms via original least squares
   - Removes LASSO coefficient bias

Validation Metrics

Qualitative Validation

- State space and phase portrait comparison
- Dynamical system behavior preservation

Objective

- Compare predicted and actual values of derivatives at time points
- Ensure stability and accuracy in system dynamics of predicted models and sparse equations

Real-World Data

PTB Diagnostic ECG Dataset

Clinical Relevance

- Dataset collected for cardiology research
- Signals correspond to ECG shapes of heartbeats to differentiate normal cases and those affected by different diseases (e.g. arrythmias)

Source and Composition

- Database: Physionet PTB Diagnostic Database
- Number of Samples: 14552
- Signals: Single-lead sampled at 125 Hz

Significance

- Identify disease dynamics
- Nonlinearity and stability of differing heartbeat cases

Adaptations for Real-World Data

ECG Adaptations

1. Preprocessing
   - Removed records with trailing zeros
   - Salitzky-Golay filter
   - Z-score normalization

2. Parameters
   - Takens' Embedding: Manual adjustment of delay τ and embedding dimension m
   - LASSO Regression: $\lambda = 0.01$

Results

Van der Pol Oscillator

- Baseline Form
- Appended Forcing Terms

Electrocardiogram (ECG) Signals

Baseline Van der Pol (Noiseless)

First-Order System of Equations:

$$\frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1 - x^2)v - x$$

Initial Conditions:

$$\begin{aligned} \mu &= 2 \\ x(0) &= 2 \\ v(0) &= 0 \\ [t_0, T] &= [0, 50] \\ \Delta t &= 0.01 \end{aligned}$$

Baseline Van der Pol (Noiseless)

Equation Term Recovery

```
# === Lasso-selected coefficients ===
('v_t_0', 0.996138)
('x_t_0*x_t_0*x_t_0', -0.000843)
('x_t_0*x_t_0*v_t_0', -0.001630)
('x_t_0*v_t_0*v_t_0', -0.005732)
('v_t_0*v_t_0*v_t_0', 0.001824)

# === OLS-refitted coefficients ===
('v_t_0', 1.010024)
('x_t_0*x_t_0*v_t_0', -0.010083)
```

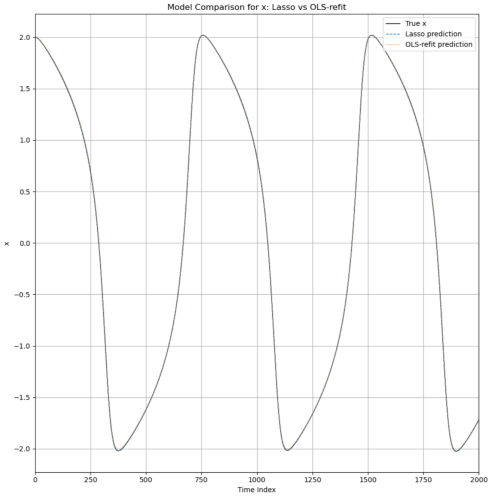
(a) Modeling $\frac{dx}{dt}$

```
# === Lasso-selected coefficients ===
('x_t_0', -0.944451)
('v_t_0', 1.990389)
('x_t_0*v_t_0', -0.000171)
('x_t_0*x_t_0*x_t_0', -0.007284)
('x_t_0*x_t_0*v_t_0', -1.958206)
('x_t_0*v_t_0*v_t_0', -0.041506)
('v_t_0*v_t_0*v_t_0', 0.002795)

# === OLS-refitted coefficients ===
('x_t_0', -1.002721)
('v_t_0', 1.979598)
('x_t_0*x_t_0*v_t_0', -2.003927)
```

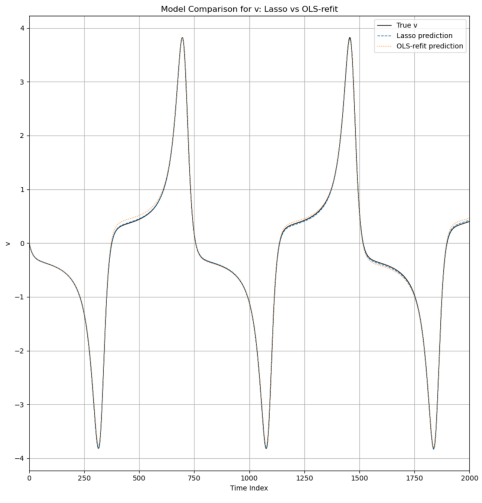
(b) Modeling $\frac{dv}{dt}$

Function Library: Terms and Coefficients

Baseline Van der Pol (Noiseless)

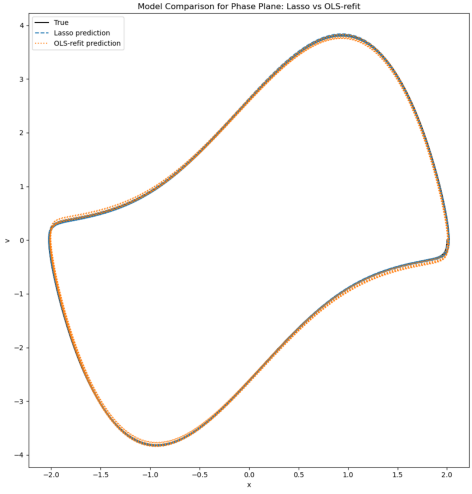


(a) Position vs Time



(b) Velocity vs Time

Baseline Van der Pol (Noiseless)



(a) Position vs Velocity

Baseline Van der Pol (Noisy)

First-Order System of Equations:

$$\frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1 - x^2)v - x$$

Initial Conditions:

$$\begin{aligned} \mu &= 2 \\ x(0) &= 2 \\ v(0) &= 0 \\ [t_0, T] &= [0, 50] \\ \Delta t &= 0.01 \end{aligned}$$

Noise Condition: Additive Gaussian noise $N(0, \sigma^2)$ where $\sigma = 0.1$

Baseline Van der Pol (Noisy)

Equation Term Recovery

```
# === Lasso-selected coefficients ===
('v_t_0', 0.990944)
('x_t_0*x_t_0', 0.000496)
('v_t_0*v_t_0', -0.000598)
('x_t_0*x_t_0*x_t_0', -0.001125)
('x_t_0*x_t_0*v_t_0', -0.000165)
('x_t_0*v_t_0*v_t_0', -0.006251)
('v_t_0*v_t_0*v_t_0', 0.002340)

# === OLS-refitted coefficients ===
('v_t_0', 1.009636)
('x_t_0*x_t_0*x_t_0', -0.001717)
('x_t_0*x_t_0*v_t_0', -0.009733)
```

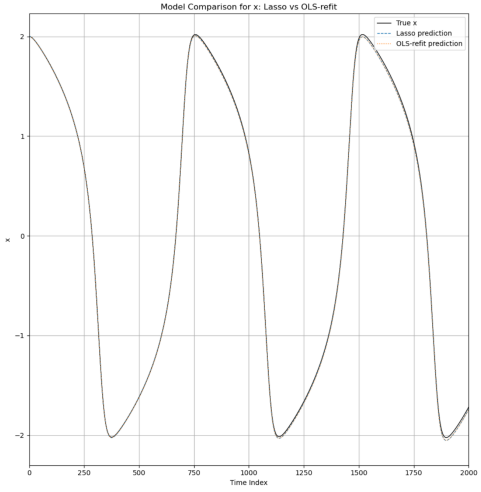
(a) Modeling $\frac{dx}{dt}$

```
# === Lasso-selected coefficients ===
('x_t_0', -0.941703)
('v_t_0', 1.990888)
('v_t_0*v_t_0', 0.000624)
('x_t_0*x_t_0*x_t_0', -0.008070)
('x_t_0*x_t_0*v_t_0', -1.958282)
('x_t_0*v_t_0*v_t_0', -0.042837)
('v_t_0*v_t_0*v_t_0', 0.002926)

# === OLS-refitted coefficients ===
('x_t_0', -1.003459)
('v_t_0', 1.979916)
('x_t_0*x_t_0*v_t_0', -2.005411)
```

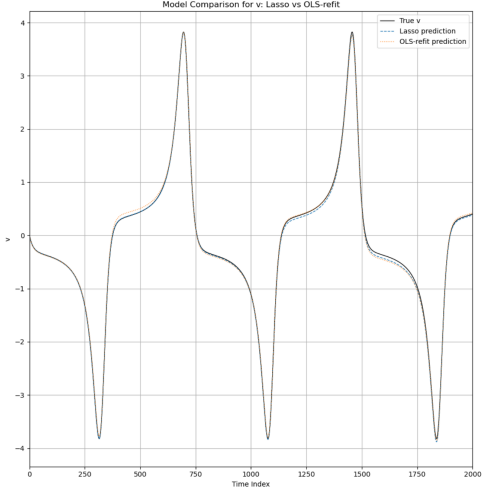
(b) Modeling $\frac{dv}{dt}$

Function Library: Terms and Coefficients

Baseline Van der Pol (Noisy)

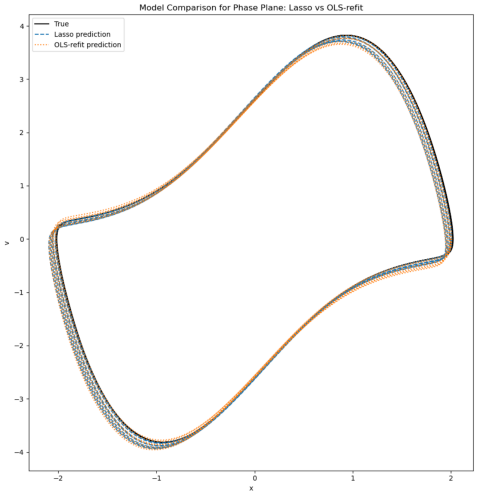


(a) Position vs Time



(b) Velocity vs Time

Baseline Van der Pol (Noisy)



(a) Position vs Velocity

Linearly-Forced Van der Pol (Noiseless)

First-Order System of Equations:

$$\frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1 - x^2)v - x + C \cdot t$$

Initial Conditions:

$$\begin{aligned} \mu &= 2 \\ x(0) &= 2 \\ v(0) &= 0 \\ [t_0, T] &= [0, 50] \\ \Delta t &= 0.01 \\ C &= 0.1 \end{aligned}$$

Linearly-Forced Van der Pol (Noiseless)

Term Selection

```
# === Lasso-selected coefficients ===
('v_t_0', 0.998322)
('v_t_0*v_t_0', -0.000352)
('x_t_0*x_t_0*x_t_0', -0.000873)
('x_t_0*x_t_0*v_t_0', -0.002621)
('x_t_0*v_t_0*v_t_0', -0.004894)
('v_t_0*v_t_0*v_t_0', 0.001500)

# === OLS-refitted coefficients ===
('v_t_0', 1.010084)
('x_t_0*x_t_0*x_t_0', -0.001459)
('x_t_0*x_t_0*v_t_0', -0.010071)
```

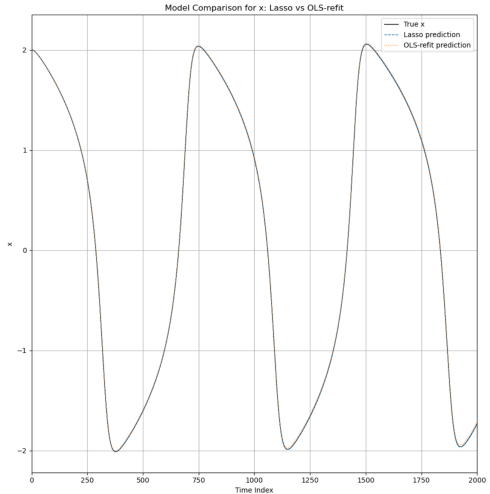
(a) Modeling $\frac{dx}{dt}$

```
# === Lasso-selected coefficients ===
('x_t_0', -0.943541)
('v_t_0', 1.899287)
('x_t_0*x_t_0', -0.013183)
('x_t_0*v_t_0', 0.027733)
('v_t_0*v_t_0', -0.014591)
('x_t_0*x_t_0*x_t_0', 0.000375)
('x_t_0*x_t_0*v_t_0', -1.904191)
('x_t_0*v_t_0*v_t_0', -0.079765)
('v_t_0*v_t_0*v_t_0', 0.016416)

# === OLS-refitted coefficients ===
('x_t_0', -0.958778)
('v_t_0', 2.013089)
('x_t_0*x_t_0*v_t_0', -1.967232)
('x_t_0*v_t_0*v_t_0', -0.032162)
```

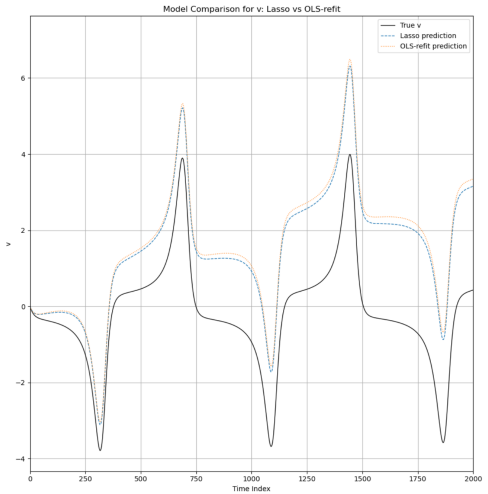
(b) Modeling $\frac{dv}{dt}$

Function Library: Terms and Coefficients

Linearly-Forced Van der Pol (Noiseless)

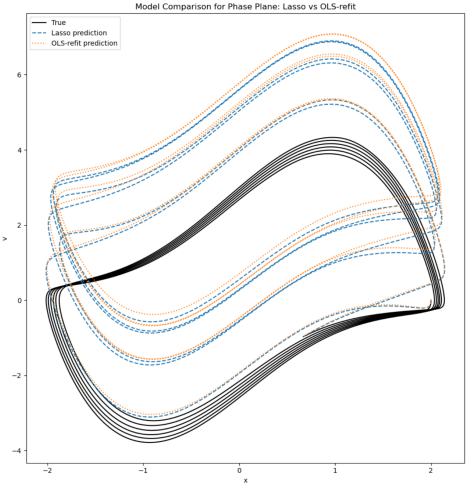


(a) Position vs Time



(b) Velocity vs Time

Linearly-Forced Van der Pol (Noiseless)



(a) Position vs Velocity

Linearly-Forced Van der Pol (Noiseless)

Term Selection Progression for Modeling $\frac{dx}{dt}$

Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

Delay $\tau = 1$

```
# === Lasso-selected coefficients ===
('v_t_0', 0.972336)
('v_t_1', 0.024212)
('v_t_1*v_t_1', -0.000436)
('x_t_0*v_t_0*v_t_1', -0.000045)
('x_t_0*x_t_1*x_t_1', -0.000420)
('x_t_0*v_t_1*v_t_1', -0.001603)
('v_t_0*v_t_0*v_t_0', 0.001740)
('v_t_0*v_t_0*x_t_1', -0.001410)
('v_t_0*x_t_1*x_t_1', -0.001761)
('v_t_0*x_t_1*v_t_1', -0.000747)
('x_t_1*x_t_1*x_t_1', -0.000377)
('x_t_1*v_t_1*v_t_1', -0.001608)

# === OLS-refitted coefficients ===
('v_t_0', 0.750341)
('v_t_1', 0.254714)
('x_t_0*x_t_1*x_t_1', -6.284833)
('x_t_0*v_t_1*v_t_1', -0.000619)
('v_t_0*x_t_1*x_t_1', -0.065123)
('x_t_1*x_t_1*x_t_1', 6.208139)
```

(a) $m = 1$

```
# === Lasso-selected coefficients ===
('v_t_0', 0.964676)
('v_t_1', 0.015679)
('v_t_2', 0.015349)
('v_t_2*v_t_2', -0.000476)
('x_t_0*x_t_1*x_t_2', -0.000069)
('x_t_0*v_t_1*x_t_2', -0.000581)
('x_t_0*x_t_2*x_t_2', -0.000165)
('x_t_0*v_t_2*v_t_2', -0.001205)
('v_t_0*v_t_0*v_t_0', 0.001537)
('v_t_0*v_t_0*x_t_1', -0.000235)
('v_t_0*x_t_1*v_t_1', -0.000653)
('v_t_0*v_t_1*v_t_2', -0.001864)
('v_t_0*x_t_1*v_t_2', -0.000505)
('v_t_0*x_t_2*v_t_2', -0.000433)
('x_t_1*x_t_1*x_t_2', -0.000196)
('x_t_1*v_t_1*v_t_2', -0.000746)
('x_t_1*x_t_2*v_t_2', -0.000193)
('x_t_1*v_t_2*v_t_2', -0.001181)
('v_t_1*v_t_1*v_t_2', 0.000179)
('x_t_2*x_t_2*v_t_2', -0.000150)
('v_t_2*v_t_2*v_t_2', -0.000444)

# === OLS-refitted coefficients ===
('v_t_0', -0.479156)
('v_t_1', 2.435532)
('v_t_2', -0.956721)
('x_t_0*x_t_2*v_t_2', -47.970464)
('v_t_0*v_t_0*x_t_1', -0.008709)
('v_t_0*v_t_1*v_t_2', -0.303320)
('x_t_1*x_t_1*x_t_2', 297.702522)
('x_t_1*x_t_2*v_t_2', -529.774179)
('x_t_1*v_t_2*v_t_2', 280.051126)
('v_t_2*v_t_2*v_t_2', -0.024807)
```

(b) $m = 2$

```
# === Lasso-selected coefficients ===
('v_t_0', 0.952747)
('v_t_1', 0.014196)
('v_t_2', 0.013877)
('v_t_3', 0.013682)
('v_t_2*v_t_3', -0.000128)
('x_t_0*v_t_1*v_t_3', -0.000106)
('x_t_0*v_t_2*v_t_3', -0.000127)
('x_t_0*v_t_3*v_t_3', -0.000152)
('v_t_0*v_t_0*v_t_0', 0.001535)
('v_t_0*v_t_1*v_t_2', -0.000570)
('v_t_0*v_t_1*v_t_3', -0.000648)
('v_t_0*v_t_2*v_t_3', -0.000997)
('v_t_0*v_t_3*v_t_3', -0.000015)
('x_t_1*v_t_1*v_t_2', -0.000039)
('x_t_1*v_t_1*v_t_3', -0.000030)
('x_t_1*v_t_2*v_t_3', -0.000253)
('x_t_1*v_t_3*v_t_3', -0.000026)
('x_t_2*v_t_2*v_t_3', -0.000205)
('x_t_2*v_t_3*v_t_3', -0.000031)
('v_t_1*v_t_1*v_t_2', -0.000064)
('v_t_1*v_t_1*v_t_3', -0.001145)
('v_t_1*v_t_2*v_t_3', 0.000421)
('x_t_2*v_t_3*v_t_3', -0.000194)
('v_t_2*v_t_3*v_t_3', -0.000953)
('v_t_3*v_t_3*v_t_3', -0.000054)

# === OLS-refitted coefficients ===
('v_t_0', -0.280112)
('v_t_1', 1.649347)
('v_t_2', -0.303571)
('x_t_0*v_t_3*v_t_3', -49.145392)
('v_t_0*v_t_1*v_t_2', 0.009440)
('v_t_0*v_t_3*v_t_3', -0.297445)
('v_t_1*v_t_2*v_t_3', -0.044954)
('x_t_1*v_t_3*v_t_3', 187.647340)
('x_t_2*v_t_3*v_t_3', -79.3194151)
('v_t_1*v_t_3*v_t_3', -225.415244)
('v_t_2*v_t_3*v_t_3', 137.031044)
('v_t_3*v_t_3*v_t_3', 29.164783)
```

(c) $m = 3$

Function Library: Terms and Coefficients

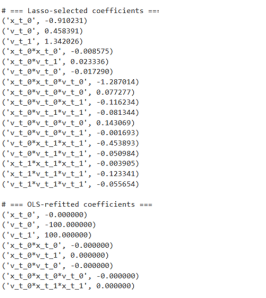
Linearly-Forced Van der Pol (Noiseless)

Term Selection Progression for Modeling $\frac{dv}{dt}$

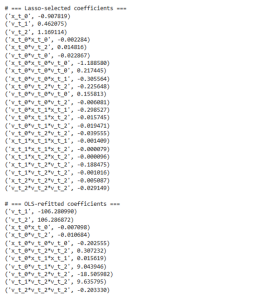
Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

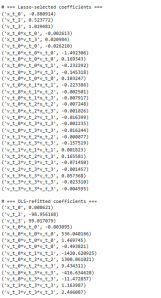
Delay $\tau = 1$



(a) $m = 1$



(b) $m = 2$



(c) $m = 3$

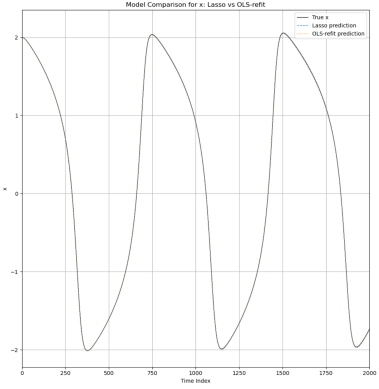
Function Library: Terms and Coefficients

Linearly-Forced Van der Pol (Noiseless)

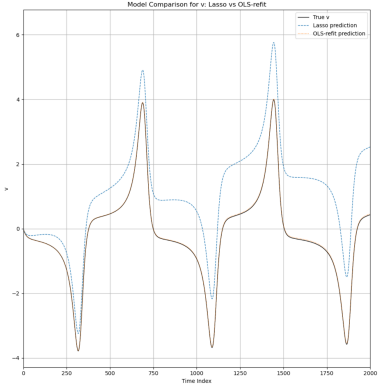
Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

Delay $\tau = 1$



(a) Position vs Time



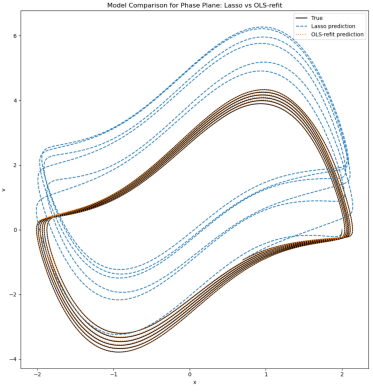
(b) Velocity vs Time

Linearly-Forced Van der Pol (Noiseless)

Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

Delay $\tau = 1$



(a) Position vs Velocity

Linearly-Forced Van der Pol (Noisy)

First-Order System of Equations:

$$\frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1 - x^2)v - x + C \cdot t$$

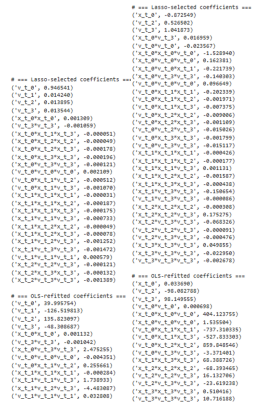
Initial Conditions:

$$\begin{aligned} \mu &= 2 \\ x(0) &= 2 \\ v(0) &= 0 \\ [t_0, T] &= [0, 50] \\ \Delta t &= 0.01 \\ C &= 0.1 \end{aligned}$$

Noise Condition: Additive Gaussian noise $N(0, \sigma^2)$ where $\sigma = 0.1$

Linearly-Forced Van der Pol (Noisy)

Term Selection



(a) Modeling $\frac{dx}{dt}$ (b) Modeling $\frac{dv}{dt}$

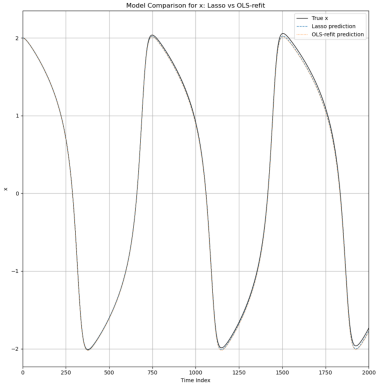
Function Library: Terms and Coefficients

Linearly-Forced Van der Pol (Noisy)

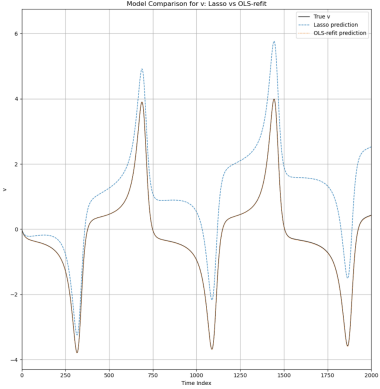
Takens' Embedding Theorem

Embedding Dimensions $m = 3$

Delay $\tau = 1$



(a) Position vs Time



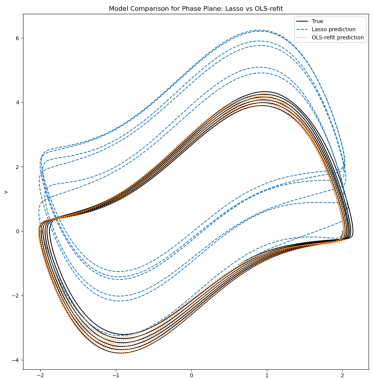
(b) Velocity vs Time

Linearly-Forced Van der Pol (Noisy)

Takens' Embedding Theorem

Embedding Dimensions $m = 3$

Delay $\tau = 1$



(a) Position vs Velocity

Periodically-Forced Van der Pol (Noiseless)

First-Order System of Equations:

$$\frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1 - x^2)v - x + D \cdot \sin(t)$$

Initial Conditions:

$$\begin{aligned} \mu &= 2 \\ x(0) &= 2 \\ v(0) &= 0 \\ [t_0, T] &= [0, 50] \\ \Delta t &= 0.01 \\ D &= 1 \end{aligned}$$

Term Selection

```
# === Lasso-selected coefficients ===
('v_t_0', 0.999099)
('x_t_0*x_t_0*x_t_0', -0.000980)
('x_t_0*x_t_0*v_t_0', -0.002994)
('x_t_0*v_t_0*v_t_0', -0.004231)
('v_t_0*v_t_0*v_t_0', 0.001098)

# === OLS-refitted coefficients ===
('v_t_0', 1.010768)
('x_t_0*x_t_0*v_t_0', -0.009790)
```

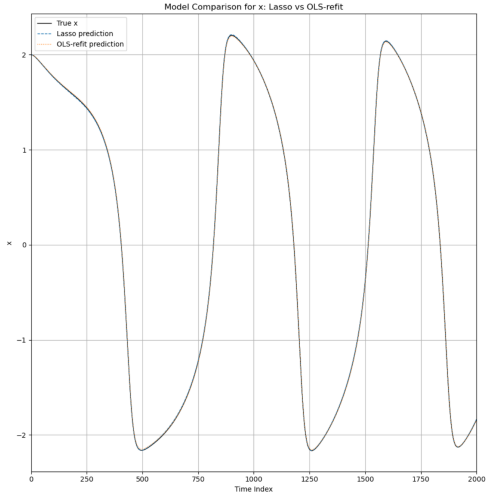
(a) Modeling $\frac{dx}{dt}$

```
# === Lasso-selected coefficients ===
('x_t_0', -1.374948)
('v_t_0', 1.868589)
('x_t_0*x_t_0', -0.080357)
('x_t_0*v_t_0', 0.067343)
('v_t_0*v_t_0', -0.034393)
('x_t_0*x_t_0*x_t_0', 0.074964)
('x_t_0*x_t_0*v_t_0', -1.735205)
('x_t_0*v_t_0*v_t_0', -0.157906)
('v_t_0*v_t_0*v_t_0', 0.030243)

# === OLS-refitted coefficients ===
('x_t_0', -1.115247)
('v_t_0', 2.224734)
('x_t_0*x_t_0*v_t_0', -1.848681)
('x_t_0*v_t_0*v_t_0', -0.095224)
```

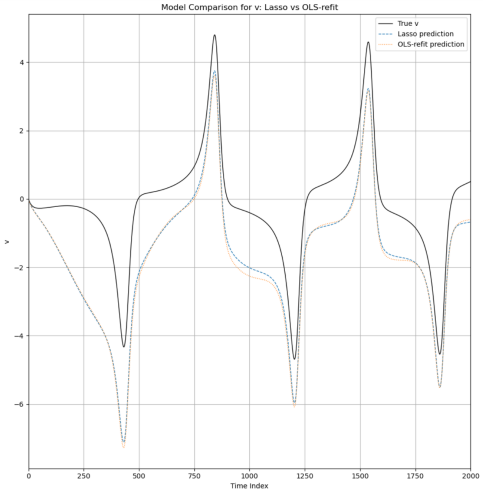
(b) Modeling $\frac{dv}{dt}$

Function Library: Terms and Coefficients

Periodically-Forced Van der Pol (Noiseless)

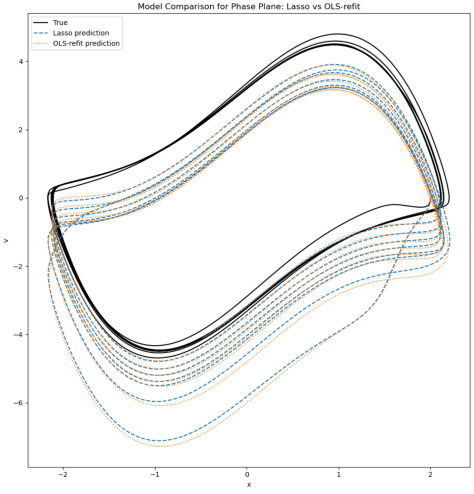


(a) Position vs Time



(b) Velocity vs Time

Periodically-Forced Van der Pol (Noiseless)



(a) Position vs Velocity

Periodically-Forced Van der Pol (Noiseless)

Term Selection Progression for Modeling $\frac{dx}{dt}$

Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

Delay $\tau = 1$

```
# === Lasso-selected coefficients ===
('v_t_0', 0.984489)
('v_t_1', 0.012367)
('x_t_0*x_t_0*x_t_0', -0.000202)
('x_t_0*x_t_0*x_t_1', -0.000168)
('x_t_0*v_t_0*x_t_1', -0.000187)
('x_t_0*x_t_1*x_t_1', -0.000422)
('x_t_0*v_t_1*v_t_1', -0.001182)
('v_t_0*v_t_0*v_t_0', 0.001332)
('v_t_0*v_t_0*x_t_1', -0.001523)
('v_t_0*x_t_1*x_t_1', -0.001975)
('v_t_0*x_t_1*v_t_1', -0.000624)
('x_t_1*x_t_1*x_t_1', -0.000128)
('x_t_1*v_t_1*v_t_1', -0.001455)

# === OLS-refitted coefficients ===
('v_t_0', 0.502889)
('v_t_1', 0.498079)
('x_t_0*v_t_1*v_t_1', -0.118952)
('v_t_0*v_t_0*v_t_0', -0.001322)
('v_t_0*x_t_1*x_t_1', -0.000215)
('x_t_1*v_t_1*v_t_1', 0.119386)
```

(a) $m = 1$

```
# === Lasso-selected coefficients ===
('v_t_0', 0.969223)
('v_t_1', 0.012978)
('v_t_2', 0.012474)
('x_t_0*x_t_1*x_t_1', -0.000076)
('x_t_0*x_t_1*x_t_2', -0.000155)
('x_t_0*x_t_2*x_t_2', -0.000209)
('x_t_0*v_t_2*v_t_2', -0.000805)
('v_t_0*v_t_0*v_t_0', 0.001520)
('v_t_0*x_t_1*x_t_1', -0.000178)
('v_t_0*x_t_1*v_t_1', -0.000165)
('v_t_0*x_t_1*x_t_2', -0.001323)
('v_t_0*x_t_1*v_t_2', -0.000997)
('v_t_0*x_t_2*x_t_2', -0.000023)
('x_t_1*x_t_1*x_t_1', -0.000282)
('x_t_1*x_t_1*x_t_2', -0.000150)
('x_t_1*x_t_1*v_t_2', -0.001466)
('x_t_1*v_t_2*v_t_2', -0.001862)

# === OLS-refitted coefficients ===
('v_t_0', 0.178654)
('v_t_1', 1.112345)
('v_t_2', -0.291232)
('x_t_0*x_t_1*x_t_1', -48.084373)
('x_t_0*v_t_2*v_t_2', -0.000135)
('v_t_0*x_t_1*x_t_1', -0.311041)
('x_t_1*x_t_1*x_t_1', 65.068082)
('x_t_1*x_t_1*x_t_2', -16.976217)
```

(b) $m = 2$

```
# === Lasso-selected coefficients ===
('v_t_0', 0.934437)
('v_t_1', 0.019398)
('v_t_2', 0.018574)
('v_t_3', 0.017878)
('v_t_3*v_t_3', -0.000018)
('x_t_0*v_t_0*v_t_0', 0.002152)
('x_t_1*x_t_1*x_t_2', -0.000140)
('x_t_1*x_t_1*x_t_3', -0.000199)
('x_t_1*x_t_2*x_t_2', -0.000248)
('x_t_1*x_t_2*x_t_3', -0.000081)
('x_t_1*v_t_2*v_t_3', -0.002804)
('x_t_1*x_t_3*x_t_3', -0.000022)
('x_t_1*v_t_3*v_t_3', -0.003454)
('x_t_1*x_t_3*v_t_3', -0.000015)
('x_t_2*v_t_3*v_t_3', -0.000097)
('v_t_2*v_t_3*v_t_3', -0.000102)
('v_t_3*v_t_3*v_t_3', -0.000152)

# === OLS-refitted coefficients ===
('v_t_0', -1.775066)
('v_t_1', 8.538273)
('v_t_2', -9.275502)
('v_t_3', 3.521672)
('v_t_3*v_t_3', 0.000014)
('x_t_1*x_t_1*x_t_2', -0.000083)
('x_t_1*v_t_2*v_t_3', -0.003446)
('x_t_1*v_t_3*v_t_3', 0.004182)
```

(c) $m = 3$

Function Library: Terms and Coefficients

Periodically-Forced Van der Pol (Noiseless)

Term Selection Progression for Modeling $\frac{dv}{dt}$

Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

Delay $\tau = 1$

```
# === Lasso-selected coefficients ===
('x_t_0', -0.296882)
('v_t_1', 1.543340)
('x_t_0*x_t_0', -0.023408)
('x_t_0*v_t_1', 0.027068)
('v_t_0*v_t_0', -0.012781)
('x_t_0*x_t_0*x_t_0', -0.058268)
('x_t_0*x_t_0*v_t_0', -4.166178)
('x_t_0*x_t_0*x_t_1', -0.006509)
('x_t_0*x_t_0*v_t_1', 3.727467)
('x_t_0*v_t_0*v_t_0', -0.292190)
('x_t_0*v_t_0*x_t_1', -3.063864)
('x_t_0*v_t_0*v_t_1', 0.528848)
('x_t_0*x_t_1*v_t_1', 2.767281)
('x_t_0*v_t_1*v_t_1', -0.308114)
('v_t_0*v_t_0*v_t_0', 0.057184)
('v_t_0*v_t_0*x_t_1', -0.273925)
('v_t_0*v_t_0*v_t_1', 0.017418)
('v_t_0*x_t_1*x_t_1', -2.868996)
('v_t_0*x_t_1*v_t_1', 0.423225)
('v_t_0*v_t_1*v_t_1', 0.049469)
('x_t_1*x_t_1*v_t_1', 2.537067)
('x_t_1*v_t_1*v_t_1', -0.452307)
('v_t_1*v_t_1*v_t_1', -0.065232)

# === OLS-refitted coefficients ===
('v_t_1', 1.424795)
('x_t_0*x_t_0', -0.003663)
('x_t_0*v_t_1', 0.015475)
('v_t_0*v_t_0', -0.005006)
('x_t_0*x_t_0*x_t_1', -0.059538)
('x_t_0*x_t_0*v_t_1', 15.614449)
('v_t_0*v_t_0*v_t_0', -430.960220)
('v_t_0*v_t_0*v_t_1', 1321.377281)
('v_t_0*x_t_1*x_t_1', -16.296095)
('v_t_0*v_t_1*v_t_1', -1350.743739)
('v_t_1*v_t_1*v_t_1', 460.355006)
```

(a) $m = 1$

```
# === Lasso-selected coefficients ===
('x_t_0', -0.000470)
('v_t_2', 1.276849)
('x_t_0*x_t_0', -0.001711)
('v_t_0*v_t_0', 0.001864)
('v_t_0*x_t_1', 0.007422)
('v_t_0*x_t_2', 0.001993)
('v_t_2*v_t_2', -0.006354)
('x_t_0*x_t_0*x_t_0', -0.045104)
('x_t_0*x_t_0*v_t_0', -3.505705)
('x_t_0*x_t_0*x_t_1', -0.010419)
('x_t_0*x_t_0*x_t_2', -0.006520)
('x_t_0*x_t_0*v_t_2', 2.192522)
('x_t_0*v_t_0*v_t_0', -0.019073)
('x_t_0*v_t_0*x_t_1', -1.976000)
('x_t_0*v_t_0*v_t_1', 0.049165)
('x_t_0*v_t_0*x_t_2', -0.067378)
('x_t_0*x_t_1*v_t_2', 2.100109)
('x_t_0*v_t_1*v_t_1', -0.131603)
('x_t_0*v_t_1*v_t_2', -0.094668)
('x_t_0*x_t_2*v_t_2', 0.329644)
('x_t_0*v_t_2*v_t_2', -0.164650)
('v_t_0*v_t_0*v_t_0', -0.076314)
('v_t_0*v_t_0*x_t_1', 0.136192)
('v_t_0*v_t_0*v_t_1', -0.014227)
('v_t_0*v_t_0*x_t_2', 0.045209)
('v_t_0*x_t_1*x_t_1', -1.520995)
('v_t_0*x_t_1*v_t_1', 0.281339)
('v_t_0*x_t_1*x_t_2', -0.436655)
('v_t_0*x_t_1*v_t_2', 0.090771)
('v_t_0*v_t_1*v_t_1', 0.029349)
('v_t_0*v_t_1*x_t_2', 0.026312)
('v_t_0*v_t_1*v_t_2', 0.023112)
('v_t_0*x_t_2*x_t_2', -0.414435)
('v_t_0*v_t_2*v_t_2', 0.053070)
('x_t_1*x_t_1*v_t_2', 1.744750)
('x_t_1*v_t_1*v_t_1', -0.250478)
('x_t_1*v_t_1*v_t_2', -0.062990)
('x_t_1*x_t_2*x_t_2', -0.007578)
('x_t_1*x_t_2*v_t_2', 0.481062)
('x_t_1*v_t_2*v_t_2', -0.229332)
('v_t_1*v_t_1*v_t_1', -0.027903)
('v_t_1*v_t_1*v_t_2', 0.004749)
('v_t_1*v_t_2*v_t_2', 0.028434)
('x_t_2*x_t_2*x_t_2', -0.012197)
('x_t_2*x_t_2*v_t_2', 0.307795)
('x_t_2*v_t_2*v_t_2', -0.108750)
('v_t_2*v_t_2*v_t_2', 0.042797)

# === OLS-refitted coefficients ===
('v_t_2', 1.264202)
('x_t_0*x_t_0', 0.008720)
('v_t_0*v_t_0', 1.261035)
('v_t_0*x_t_1', 195.509704)
('v_t_0*x_t_2', -195.521694)
('v_t_2*v_t_2', 0.697301)
('x_t_0*x_t_0*x_t_1', -843.545189)
('x_t_0*x_t_0*x_t_2', 1186.031247)
('x_t_0*v_t_0*v_t_0', -112.533456)
('x_t_0*x_t_1*v_t_2', 6.721867)
('v_t_0*v_t_0*v_t_0', -27642.706867)
('v_t_0*v_t_0*x_t_1', 213.857960)
('v_t_0*v_t_0*v_t_1', 94337.617570)
('v_t_0*v_t_0*x_t_2', -101.165556)
('v_t_0*v_t_1*v_t_1', -56412.330715)
('v_t_0*v_t_1*v_t_2', -121284.843127)
('v_t_0*x_t_2*x_t_2', -5.446265)
('v_t_0*v_t_2*v_t_2', 71979.009676)
('x_t_1*x_t_2*x_t_2', -342.529534)
('v_t_1*v_t_1*v_t_1', -43775.129933)
('v_t_1*v_t_1*v_t_2', 232974.317230)
('v_t_1*v_t_2*v_t_2', -194911.933625)
('v_t_2*v_t_2*v_t_2', 44735.925389)
```

(b) $m = 2$

```
# === Lasso-selected coefficients ===
('v_t_3', 1.205371)
('v_t_0*v_t_0', 0.009302)
('v_t_0*x_t_2', 0.004876)
('v_t_3*v_t_3', -0.012307)
('x_t_0*x_t_0*x_t_0', -0.041514)
('x_t_0*x_t_0*v_t_0', -2.986948)
('x_t_0*x_t_0*x_t_1', -0.016046)
('x_t_0*x_t_0*x_t_2', -0.002347)
('x_t_0*x_t_0*v_t_2', 0.137323)
('x_t_0*x_t_0*x_t_3', -0.009487)
('x_t_0*x_t_0*v_t_3', 1.497516)
('x_t_0*v_t_0*x_t_1', -1.449709)
('x_t_0*v_t_0*x_t_2', -0.000152)
('x_t_0*v_t_0*x_t_3', -0.000482)
('x_t_0*x_t_1*v_t_2', 0.119587)
('x_t_0*x_t_1*v_t_3', 1.444000)
('x_t_0*v_t_1*v_t_2', -0.078083)
('x_t_0*v_t_1*v_t_3', -0.051130)
('x_t_0*x_t_2*v_t_3', 0.179548)
('x_t_0*v_t_2*v_t_2', -0.003486)
('x_t_0*v_t_2*v_t_3', -0.074483)
('x_t_0*x_t_3*x_t_3', -0.000042)
('x_t_0*x_t_3*v_t_3', 0.092311)
('x_t_0*v_t_3*v_t_3', -0.117417)
('v_t_0*v_t_0*v_t_0', -0.063260)
('v_t_0*v_t_0*x_t_1', 0.032951)
('v_t_0*v_t_0*v_t_1', -0.002958)
('v_t_0*v_t_0*x_t_2', 0.008230)
('v_t_0*v_t_0*x_t_3', 0.002417)
('v_t_0*x_t_1*x_t_1', -0.927333)
('v_t_0*x_t_1*v_t_1', 0.218225)
('v_t_0*x_t_1*x_t_2', -0.335353)
('v_t_0*x_t_1*v_t_2', 0.074890)
('v_t_0*x_t_1*x_t_3', -0.107592)
('v_t_0*v_t_1*v_t_1', 0.001888)
('v_t_0*v_t_1*x_t_2', 0.072442)
('v_t_0*v_t_1*x_t_3', 0.009006)
('v_t_0*v_t_1*v_t_3', 0.007259)
('v_t_0*x_t_2*x_t_2', -0.169704)
('v_t_0*x_t_2*x_t_3', -0.001410)
('v_t_0*v_t_2*v_t_2', 0.006473)
('v_t_0*v_t_2*v_t_3', 0.019305)
('v_t_0*x_t_3*x_t_3', -0.121507)
('v_t_0*v_t_3*v_t_3', 0.028715)
('x_t_1*x_t_1*v_t_3', 1.349591)
('x_t_1*v_t_1*v_t_1', -0.016081)
('x_t_1*v_t_1*v_t_2', -0.078247)
('x_t_1*v_t_1*v_t_3', -0.046624)
('x_t_1*x_t_2*v_t_3', 0.205194)
('x_t_1*v_t_2*v_t_2', -0.045142)
('x_t_1*v_t_2*v_t_3', -0.066416)
('x_t_1*x_t_3*v_t_3', 0.149806)
('x_t_1*v_t_3*v_t_3', -0.126742)
('v_t_1*v_t_1*v_t_1', -0.049911)
('v_t_1*v_t_1*v_t_3', 0.004362)
('v_t_1*v_t_2*v_t_3', 0.018692)
('v_t_1*v_t_3*v_t_3', 0.024453)
('x_t_2*x_t_2*v_t_3', 0.222979)
('x_t_2*v_t_3*v_t_3', -0.052391)
('v_t_2*v_t_2*v_t_2', -0.002398)
('v_t_2*v_t_3*v_t_3', 0.023449)
('x_t_3*v_t_3*v_t_3', -0.065381)
('v_t_3*v_t_3*v_t_3', 0.043177)

# === OLS-refitted coefficients ===
('v_t_3', 1.165258)
('v_t_0*v_t_0', 0.027046)
('v_t_0*x_t_2', 0.001851)
('v_t_3*v_t_3', -0.027379)
('x_t_0*x_t_0*x_t_0', 1957.962852)
('x_t_0*x_t_0*x_t_1', -3286.636945)
('x_t_0*x_t_0*x_t_2', 1217.421165)
('x_t_0*x_t_0*v_t_2', -157.926169)
('x_t_0*x_t_0*v_t_3', 109.232418)
('x_t_0*v_t_0*x_t_3', 884.309371)
('x_t_0*x_t_3*x_t_3', 111.219586)
('v_t_0*v_t_0*v_t_0', 48926.673458)
('v_t_0*v_t_0*x_t_1', -278.595635)
('v_t_0*v_t_0*v_t_1', -211978.106735)
('v_t_0*v_t_0*x_t_2', 465.254532)
('v_t_0*v_t_0*x_t_3', -181.588731)
('v_t_0*x_t_1*x_t_1', -1064.772238)
('v_t_0*x_t_1*x_t_2', 1962.908577)
('v_t_0*x_t_1*x_t_3', -1732.442459)
('v_t_0*v_t_1*v_t_1', 288504.374635)
('v_t_0*v_t_1*v_t_3', 213483.552157)
('v_t_0*v_t_2*v_t_2', 110521.605298)
('v_t_0*v_t_2*v_t_3', -565206.908146)
('v_t_0*v_t_3*v_t_3', 229824.747615)
('v_t_1*v_t_1*v_t_1', -141339.241144)
('v_t_1*v_t_1*v_t_3', -364482.191046)
('v_t_1*v_t_2*v_t_3', 906262.265521)
('v_t_1*v_t_3*v_t_3', -331630.745215)
('v_t_2*v_t_2*v_t_2', -167693.446816)
('v_t_2*v_t_3*v_t_3', -59212.192454)
('v_t_3*v_t_3*v_t_3', 44018.767606)
```

(c) $m = 3$

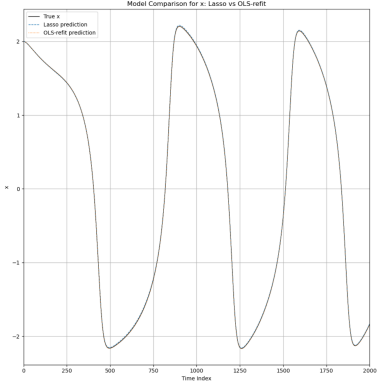
Function Library: Terms and Coefficients

Periodically-Forced Van der Pol (Noiseless)

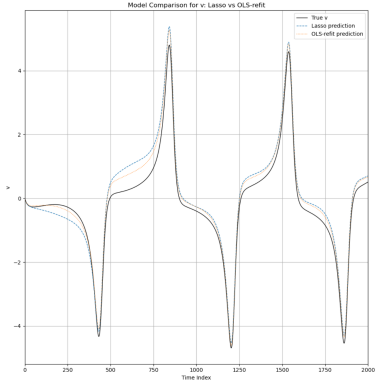
Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

Delay $\tau = 1$



(a) Position vs Time



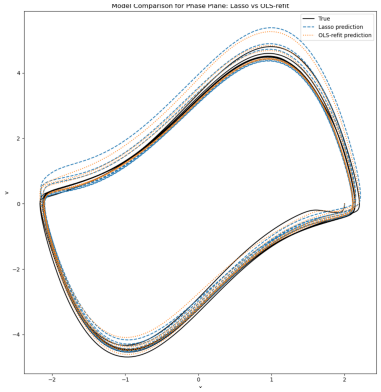
(b) Velocity vs Time

Periodically-Forced Van der Pol (Noiseless)

Takens' Embedding Theorem

Embedding Dimensions $m = \{0, 1, 2, 3\}$

Delay $\tau = 1$



(a) Position vs Velocity

Periodically-Forced Van der Pol (Noisy)

First-Order System of Equations:

$$\frac{dx}{dt} = v \ , \ \frac{dv}{dt} = \mu(1 - x^2)v - x + D \cdot \sin(t)$$

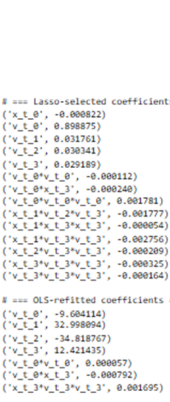
Initial Conditions:

$$\begin{aligned} \mu &= 2 \\ x(0) &= 2 \\ v(0) &= 0 \\ [t_0, T] &= [0, 50] \\ \Delta t &= 0.01 \\ D &= 1 \end{aligned}$$

Noise Condition: Additive Gaussian noise $N(0, \sigma^2)$ where $\sigma = 0.1$

Periodically-Forced Van der Pol (Noisy)

Term Selection



(a) Modeling $\frac{dx}{dt}$ (b) Modeling $\frac{dv}{dt}$

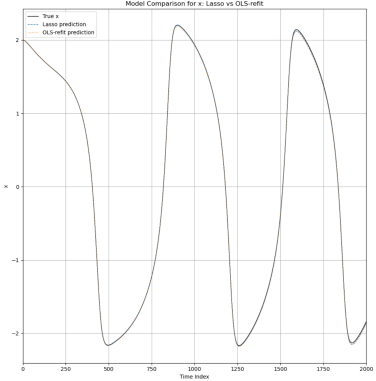
Function Library: Terms and Coefficients

Periodically-Forced Van der Pol (Noisy)

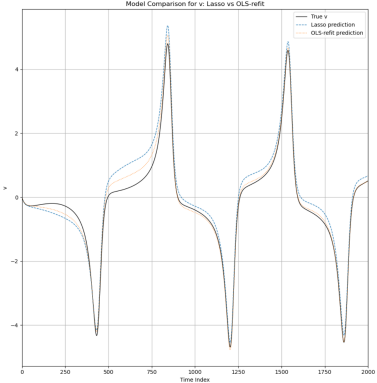
Takens' Embedding Theorem

Embedding Dimensions $m = 3$

Delay $\tau = 1$



(a) Position vs Time



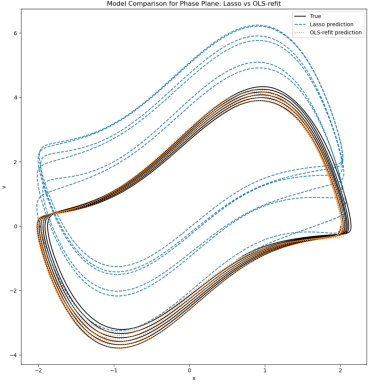
(b) Velocity vs Time

Periodically-Forced Van der Pol (Noisy)

Takens' Embedding Theorem

Embedding Dimensions $m = 3$

Delay $\tau = 1$



(a) Position vs Velocity

Baseline Van der Pol (Noiseless)

Second-Order Equation:

$$\frac{d^2x}{dt^2} = \mu(1 - x^2)\frac{dx}{dt} - x$$

Initial Conditions:

$$\begin{aligned} \mu &= 2 \\ x(0) &= 2 \\ [t_0, T] &= [0, 50] \\ \Delta t &= 0.01 \end{aligned}$$

Baseline Van der Pol (Noiseless)

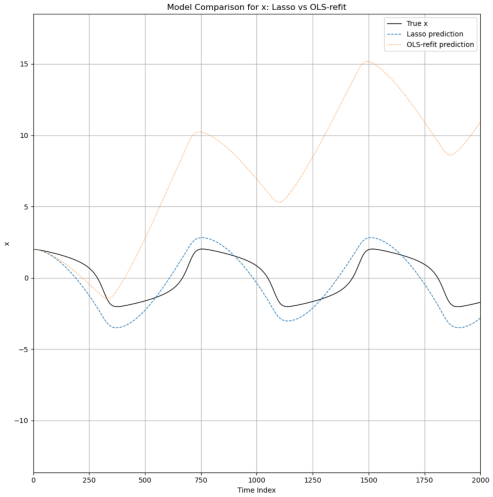
Equation Recovery

```
# === Lasso-selected coefficients ===
('x_t_0', -25.568020)
('x_t_1', 24.705498)
('x_t_1*x_t_1', 0.079007)
('x_t_0*x_t_0*x_t_0', 35.346413)
('x_t_0*x_t_0*x_t_1', -11.103980)
('x_t_0*x_t_1*x_t_1', -12.101070)
('x_t_1*x_t_1*x_t_1', -12.185770)

# === OLS-refitted coefficients ===
('x_t_0', -0.957568)
('x_t_0*x_t_0*x_t_0', 32.848179)
('x_t_1*x_t_1*x_t_1', -32.863311)
```

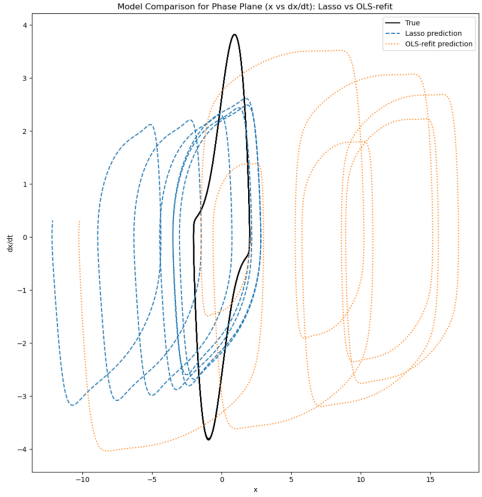
(a) Modeling $\frac{d^2x}{dt^2}$

Function Library: Terms and Coefficients

Baseline Van der Pol (Noiseless)



(a) Position vs Time



(b) Position vs Velocity

Baseline Van der Pol (Noiseless)

Equation Recovery

Takens' Embedding Theorem

Embedding Dimensions $m = 5$

Delay $\tau = 1$

```
# === Lasso-selected coefficients ===
('x_t_0', -72.767607)
('x_t_1', 9.970130)
('x_t_2', 19.853485)
('x_t_3', 20.691529)
('x_t_4', 21.496002)
('x_t_0*x_t_0', 0.004457)
('x_t_0*x_t_0*x_t_0', 23.798777)
('x_t_0*x_t_0*x_t_1', -1.408638)
('x_t_0*x_t_0*x_t_2', -2.164900)
('x_t_0*x_t_0*x_t_3', -2.018881)
('x_t_0*x_t_0*x_t_4', -1.877614)
('x_t_0*x_t_1*x_t_1', 2.835992)
('x_t_0*x_t_1*x_t_2', -1.257132)
('x_t_0*x_t_1*x_t_3', -2.016590)
('x_t_0*x_t_1*x_t_4', -1.873902)
('x_t_0*x_t_2*x_t_2', 0.820240)
('x_t_0*x_t_2*x_t_3', -1.105790)
('x_t_0*x_t_2*x_t_4', -1.869041)
('x_t_0*x_t_3*x_t_3', -0.139636)
('x_t_0*x_t_3*x_t_4', -1.863052)
('x_t_0*x_t_4*x_t_4', -1.855995)
('x_t_1*x_t_1*x_t_1', 8.839786)
('x_t_1*x_t_1*x_t_2', -1.253382)
('x_t_1*x_t_1*x_t_3', -2.011235)
('x_t_1*x_t_1*x_t_4', -1.867106)
('x_t_1*x_t_2*x_t_2', 0.814796)
('x_t_1*x_t_2*x_t_3', -1.097973)
('x_t_1*x_t_2*x_t_4', -1.859535)
('x_t_1*x_t_3*x_t_3', -0.138398)
('x_t_1*x_t_3*x_t_4', -1.850799)
('x_t_1*x_t_4*x_t_4', -1.840960)
('x_t_2*x_t_2*x_t_2', 4.637018)
('x_t_2*x_t_2*x_t_3', -1.089003)
('x_t_2*x_t_2*x_t_4', -1.849054)
('x_t_2*x_t_3*x_t_3', -0.137094)
('x_t_2*x_t_3*x_t_4', -1.837538)
('x_t_2*x_t_4*x_t_4', -1.824884)
('x_t_3*x_t_3*x_t_3', 0.781716)
('x_t_3*x_t_3*x_t_4', -0.916029)
('x_t_3*x_t_4*x_t_4', -1.807688)
('x_t_4*x_t_4*x_t_4', -1.789301)
```

```
# === OLS-refitted coefficients ===
('x_t_0', 16913.403980)
('x_t_1', -40068.780254)
('x_t_2', 27876.049043)
('x_t_3', -3166.169412)
('x_t_4', -1554.463270)
('x_t_0*x_t_0*x_t_0', -1722532.529595)
('x_t_0*x_t_0*x_t_1', 5197831.944658)
('x_t_0*x_t_0*x_t_2', 2242355.781644)
('x_t_0*x_t_0*x_t_4', 137369.559002)
('x_t_0*x_t_1*x_t_1', -4994702.459790)
('x_t_0*x_t_1*x_t_2', -10681651.307364)
('x_t_0*x_t_2*x_t_2', 5941975.420477)
('x_t_0*x_t_4*x_t_4', -253111.885689)
('x_t_1*x_t_1*x_t_1', 5157639.192886)
('x_t_2*x_t_2*x_t_3', -3882154.801162)
('x_t_2*x_t_2*x_t_4', 336141.127342)
('x_t_2*x_t_3*x_t_3', 3648576.273707)
('x_t_3*x_t_3*x_t_3', -1038376.157844)
('x_t_3*x_t_3*x_t_4', -150736.377093)
('x_t_4*x_t_4*x_t_4', 61376.216620)
```

(a) Modeling $\frac{d^2x}{dt^2}$

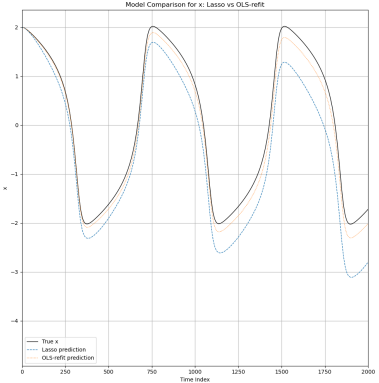
Function Library: Terms and Coefficients

Baseline Van der Pol (Noiseless)

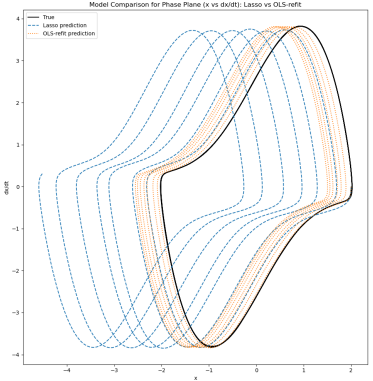
Takens' Embedding Theorem

Embedding Dimensions $m = 5$

Delay $\tau = 1$



(a) Position vs Time



(b) Position vs Velocity

ECG Application

Initial Conditions:

$$[t_0, T] = [0, 125]$$
$$\Delta t = 0.01$$

Takens' Embedding Theorem

Embedding Dimensions $m = \{1, 4, 7\}$

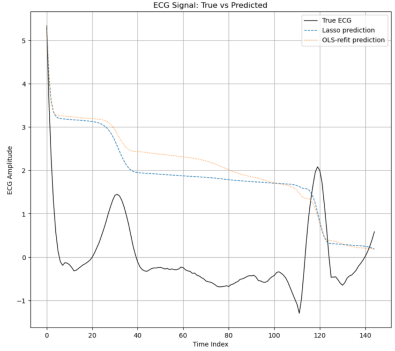
Delay $\tau = \{1, 5\}$

ECG Application

Takens' Embedding Theorem

Embedding Dimensions $m = 1$

Delay $\tau = 1$



(a) ECG Amplitude vs Time

```
==== Modeling dECG/dt ====

# === Lasso-selected coefficients ===
('ECG(t-0)', -3.478432)
('ECG(t-0) × ECG(t-0)', -4.439278)

# === OLS-refitted coefficients ===
('ECG(t-0) × ECG(t-0)', -5.308340)
```

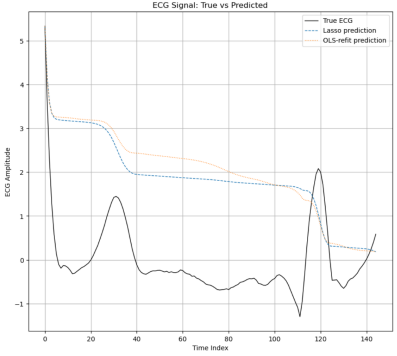
(b) Function Library

ECG Application

Takens' Embedding Theorem

Embedding Dimensions $m = 1$

Delay $\tau = 5$



(a) ECG Amplitude vs Time

```
==== Modeling dECG/dt ====

# === Lasso-selected coefficients ===
('ECG(t-0)', -3.478432)
('ECG(t-0) × ECG(t-0)', -4.439278)

# === OLS-refitted coefficients ===
('ECG(t-0) × ECG(t-0)', -5.308340)
```

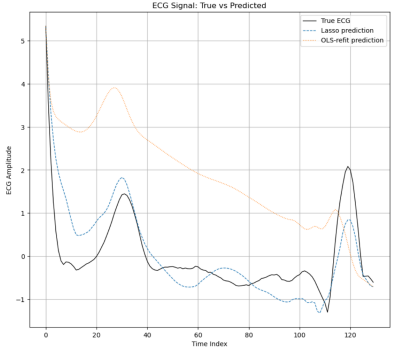
(b) Function Library

ECG Application

Takens' Embedding Theorem

Embedding Dimensions $m = 4$

Delay $\tau = 5$



(a) ECG Amplitude vs Time

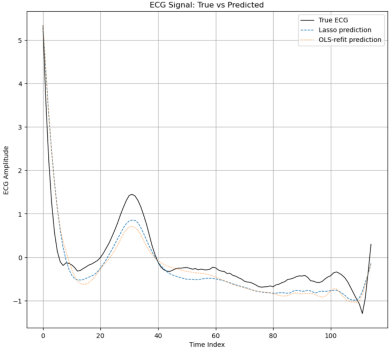
```
==== Modeling dECG/dt ====

# === Lasso-selected coefficients ===
('ECG(t-0)', -40.719787)
('ECG(t-5)', 24.972056)
('ECG(t-10)', -5.062534)
('ECG(t-15)', -33.506476)
('ECG(t-0) × ECG(t-0)', -1.249113)
('ECG(t-0) × ECG(t-5)', 4.448063)
('ECG(t-0) × ECG(t-15)', -103.921448)
('ECG(t-5) × ECG(t-5)', -1.709816)
('ECG(t-5) × ECG(t-10)', 22.587585)
('ECG(t-5) × ECG(t-15)', 22.825517)
('ECG(t-10) × ECG(t-10)', -2.241461)
('ECG(t-10) × ECG(t-15)', 8.753226)
('ECG(t-15) × ECG(t-15)', 13.610117)

# === OLS-refitted coefficients ===
('ECG(t-0)', -1.517287)
('ECG(t-10)', 4.218464)
('ECG(t-0) × ECG(t-0)', -4.564975)
('ECG(t-5) × ECG(t-10)', 8.925940)
('ECG(t-10) × ECG(t-15)', 2.874816)
('ECG(t-15) × ECG(t-15)', 2.135717)
```

(b) Function Library

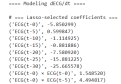
Takens' Embedding Theorem

Embedding Dimensions $m = 7$

Delay $\tau = 5$



(a) ECG Amplitude vs Time



```
==== Modeling dECG/dt ====

# === Lasso-selected coefficients ===
('ECG(t-0)', -5.056098)
('ECG(t-5)', 8.509843)
('ECG(t-10)', -1.114915)
('ECG(t-15)', -0.681886)
('ECG(t-20)', -7.508920)
('ECG(t-25)', -15.881225)
('ECG(t-30)', -25.665578)
('ECG(t-0) × ECG(t-0)', 5.508520)
('ECG(t-0) × ECG(t-5)', 4.496817)
('ECG(t-0) × ECG(t-10)', -1.857445)
('ECG(t-0) × ECG(t-15)', 15.877273)
('ECG(t-0) × ECG(t-25)', -3.704506)
('ECG(t-0) × ECG(t-30)', -7.404020)
('ECG(t-5) × ECG(t-5)', 13.572024)
('ECG(t-5) × ECG(t-10)', -7.135512)
('ECG(t-5) × ECG(t-15)', 8.570155)
('ECG(t-5) × ECG(t-20)', -11.291540)
('ECG(t-5) × ECG(t-25)', -20.718802)
('ECG(t-10) × ECG(t-10)', 5.520211)
('ECG(t-10) × ECG(t-15)', 2.811698)
('ECG(t-10) × ECG(t-20)', 8.027904)
('ECG(t-10) × ECG(t-30)', 0.7476521)
('ECG(t-15) × ECG(t-15)', 2.122483)
('ECG(t-15) × ECG(t-20)', 3.227254)
('ECG(t-15) × ECG(t-25)', 4.159228)
('ECG(t-15) × ECG(t-30)', -32.439137)
('ECG(t-20) × ECG(t-20)', 8.554202)
('ECG(t-20) × ECG(t-25)', 10.809954)
('ECG(t-20) × ECG(t-30)', -5.936427)
('ECG(t-25) × ECG(t-25)', 2.569669)
('ECG(t-25) × ECG(t-30)', -1.127430)
('ECG(t-30) × ECG(t-30)', 3.098081)

# === OLS-refitted coefficients ===
('ECG(t-0)', -12.787168)
('ECG(t-5)', -1.423611)
('ECG(t-20)', 10.109510)
('ECG(t-25)', -36.298010)
('ECG(t-0) × ECG(t-0)', 2.271769)
('ECG(t-0) × ECG(t-15)', 7.066506)
('ECG(t-0) × ECG(t-25)', -22.594130)
('ECG(t-5) × ECG(t-5)', 11.200680)
('ECG(t-5) × ECG(t-15)', -9.180666)
('ECG(t-5) × ECG(t-20)', 12.029439)
('ECG(t-5) × ECG(t-25)', -46.799861)
('ECG(t-10) × ECG(t-30)', 0.532217)
('ECG(t-10) × ECG(t-15)', 3.527048)
('ECG(t-15) × ECG(t-25)', 1.540111)
('ECG(t-15) × ECG(t-30)', -3.856256)
('ECG(t-20) × ECG(t-20)', 1.203194)
```

(b) Function Library

Discussion

Key Observations from Results

Baseline Van der Pol

- 1st-Order system of equations over 2nd-Order equation
- Hybrid sparse regression (LASSO + OLS-refit) method successfully recovered key terms in noiseless and noisy conditions
- Significant reduction of terms through hybrid sparse regression method

Forced Systems

- Time-delay embedding ($\tau = 1, m = 3$) captured forcing dynamics with sufficient accuracy in both noiseless and noisy environments for hybrid sparse regression
- LASSO regression method fails in capturing forcing dynamics regardless of noise effects
- Higher value in embedding dimensions m created more complex sparse regression equations

Discussion

Key Observations from Results

ECG Application

- Time-delay embedding ($\tau = 5, m = 7$) almost reconstructed the ECG signals
- LASSO and hybrid sparse regression techniques produced similar predictions and behavior

Clinical Implications

Disease Detection

- Reconstructed state spaces revealed instability behavior

Limitations

- Delay terms (e.g, $x(t - 5\tau)$) lack physiological interpretability for clinicians

Limitations

Boundaries of the Framework

Derivative Estimation

- Forward Euler method introduces local truncation error to data

Function Library Design

- Polynomials struggled with discontinuities from real-world data

Embedding Complexity

- Increased values of time-delay embedding (τ, m) provide more terms and present obscurity in interpreting the resulting sparse equations
- High multicollinearity between terms resulted in extremely high corresponding coefficients

Future Directions

Next Steps for Robustness and Impact

Algorithmic

- Replace Forward Euler method with total variation regularization for noisy data
- Consider other sparse regression methods (e.g., stepwise) to reduce multicollinearity between correlated terms
- Construct adaptive λ selection for LASSO to consider optimal number of terms

Applications

- Extend hybrid sparse regression technique to different datasets and practices (e.g., physics, engineering)
- Determine relevance of applying Takens' embedding to real-world scenarios

Key Contributions

What We Achieved

Hybrid Regression Framework

- LASSO for sparsity + MLR for accuracy

Noise-Robust Identification

- Maintained sufficient term accuracy at $\sigma = 0.1$

Real-World Validation

- ECG embedding ($\tau = 5, m = 7$) captured dynamical features

Impacts

Importance of Research

- Generalized framework for observing forced systems in real-world applications
- Potential strength of hybrid sparse regression technique for discovering nonlinear differential equations
- Integrate Takens' embedding theorem to sparse regression techniques

The End