Data-Driven Black-Box Modeling of Hidden Systems of Ordinary and Partial Differential Equations

Masters Thesis Defense

Cory Suzuki

Department of Mathematics & Statistics
California State University, Long Beach

11 July 2025

Overview

Introduction

- Partial differential equations (PDEs) and systems of Ordinary Differential Equations (ODEs) govern change within physical systems from finance to fluid dynamics
- What if we want to model physical systems and how they change over time for any point in time?
- Goal: Utilize machine learning and statistical techniques to learn about the system and extract a data-driven governing equation to be used as a predictive extrapolation model
- Apply linear regression, regularized LASSO regression, and Proper Orthogonal Decomposition (POD) to extract hidden latent information from video data
- Once the model is obtained, can we generate image extrapolations after training the model for any time step ($t \in \mathbb{R}$) in the future?

Ordinary Differential Equations

Definition of Ordinary Differential Equation

Let $n \in \mathbb{N}$ and $y : Y \rightarrow \mathbb{R}$, where Y is an open subset of $\mathbb{R}^n$. Then an nth-order Ordinary Differential Equation (ODE) follows the functional form:

$$F\left(x, y, \frac{dy}{dx}, \ldots, \frac{d^{k-1}y}{dx^{k-1}}, \frac{d^k y}{dx^k}\right) = 0 \tag{1}$$

where the function $F : \mathbb{R}^{n^k} \times \mathbb{R}^{n^{k-1}} \times \mathbb{R}^n \times \mathbb{R} \times Y \rightarrow \mathbb{R}$ is dependent on x, y, and the derivatives of the independent variable y

- Ordinary Differential Equations (ODEs) can be used to mathematically model phenomena in the natural sciences, physics, and economics
- ODEs can be separated into two classes: linear and nonlinear
- Many complicated ODEs require numerical solutions in the case that closed-form solutions are unobtainable

Partial Differential Equations

Definition of Partial Differential Equation

Let $u : U \to \mathbb{R}$ be an arbitrary, unknown function that is a solution to a partial differential equation. Let $x = (x_1, x_2, ..., x_n)$ be the variables belonging to the open subset U of the Euclidian space $\mathbb{R}^n$. Then the kth-order partial differential equation follows the general form:

$$F(D^k u, D^{k-1} u, ..., Du, u, x) = 0 \tag{2}$$

In this general form, D is the partial differential operator and F is the mapping $F : \mathbb{R}^{n^k} \times \mathbb{R}^{n^{k-1}} \times \mathbb{R}^n \times \mathbb{R} \times U \to \mathbb{R}$.

- Partial Differential Equations (PDEs) are extensions of ODEs to several variables
- PDEs can model change across several variables in a system
- The ODEs and PDEs discussed in this study will be linear
- For computing the solutions to our models, emphasis is placed on using numerical techniques

Heat (Diffusion) Equation

The diffusion equation, sometimes alternatively referred to as the heat equation, takes on the form:

$$u_t = ku_{xx} \tag{3}$$

for $0 < x < L$ and $t \geq 0$, where $k \in \mathbb{R}$ is the heat-diffusive constant.

Wave Equation

The heat equation is given in generality by:

$$u_{tt} = c^2 u_{xx} \tag{4}$$

for $0 < x < L$ and $t \geq 0$ where $c \in \mathbb{R}$ is the wave propagation constant

Fundamental Partial Differential Equations & their Solutions

- Equations 3 and 4 can be solved analytically via Separation of Variables, decomposing the PDEs into ODEs
- These PDEs can be solved given specific boundary and initial conditions
- Particular solutions and the application of separation of variables can be found in [Strauss, 2008]

Particular Solution of Heat Equation via Separation of Variables

Given the initial condition $u(x, 0) = f(x)$ with $f(x)$ being a real-valued function and boundary conditions $u(0, t) = u(L, t) = 0$, the particular solution is:

$$u(x,t) = \sum_{n=1}^{\infty} c_n u_n(x,t) \tag{5}$$

$$c_n = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{n\pi x}{L}\right)dx \tag{6}$$

Systems of ODEs

- Systems of ODEs are useful in capturing linear and nonlinear dynamics in many physical situations
- Consists of n many ODEs to form a system that can be represented in matrix algebra

n First Order System of ODEs

For a system of n first-order linear equations,

$$x_1' = p_{11}(t)x_1 + \ldots + p_{1n}(t)x_n + g_1(t),$$
$$\vdots$$
$$x_n' = p_{n1}(t)x_1 + \ldots + p_{nn}(t)x_n + g_n(t)$$

Systems of ODEs Cont.

Matrix Formulation of n First-Order System of ODEs

$\boldsymbol{x}(t)$ is the vector consisting of the elements $x_1(t), \ldots, x_n(t)$, $\boldsymbol{g}(t)$ is the vector consisting of the components $g_1(t), \ldots, g_n(t)$, and $p_{11}(t), \ldots, p_{nn}(t)$ are elements of an $n \times n$ matrix $\boldsymbol{P}(t)$. The resulting equation is $\boldsymbol{x}' = \boldsymbol{P}(t)\boldsymbol{x} + \boldsymbol{g}(t)$. One can solve this system by first finding the homogeneous solution by making $\boldsymbol{g}(t) = \mathbf{0}$. So we can denote the solutions as

$$\boldsymbol{x}^{(1)}(t) = \begin{pmatrix} x_{11}(t) \\ x_{21}(t) \\ \vdots \\ x_{n1}(t) \end{pmatrix}, \ldots, \boldsymbol{x}^{(k)}(t) = \begin{pmatrix} x_{1k}(t) \\ x_{2k}(t) \\ \vdots \\ x_{nk}(t) \end{pmatrix}, \ldots \tag{7}$$

- If closed-form solutions are hard to obtain, numerical methods can be used to find approximated solutions to a set precision
- Finite Difference Schemes used to approximate derivatives and partial derivatives

Finite Difference Example

Let $y(0) = 0$ and $y(5) = 50$ be boundary conditions for the following second-order differential equation:

$$\frac{d^2y}{dx^2} = -g \tag{8}$$

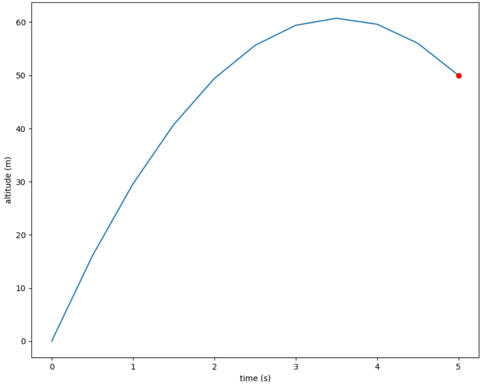
We can use the finite difference scheme

$$\frac{d^2y}{dx^2} = \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} \tag{9}$$

for $i = 1, \ldots, n-1$ to discretize the problem into a system of equations.

Numerical Methods Cont.

$$
\begin{pmatrix}
1 & 0 & & & \\
1 & -2 & 1 & & \\
 & \ddots & \ddots & \ddots & \\
 & & 1 & -2 & 1 \\
 & & & & 1
\end{pmatrix}
\begin{pmatrix}
y_0 \\ y_1 \\ \dots \\ y_{n-1} \\ y_n
\end{pmatrix}
=
\begin{pmatrix}
0 \\ -gh^2 \\ \dots \\ -gh^2 \\ 50
\end{pmatrix}
$$



Multiple Linear Regression

Suppose we have the model $y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon_i$. Then we can rewrite the model in matrix form:

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{10}$$

where $\boldsymbol{y}$ is the response vector $(y_1, y_2, ..., y_n)^T$, $\boldsymbol{\beta}$ is the regression coefficient vector consisting of $(\beta_0, \beta_1, ..., \beta_p)^T$, $\boldsymbol{\epsilon}$ is the random error vector $(\epsilon_1, \epsilon_2, ..., \epsilon_n)^T$

X is the following design matrix:

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

Multiple Linear regression Cont.

- In MLP, we assume each random error in ϵ follows a Normal Distribution with zero mean and finite variance
- Observations are independent
- There is a linear relationship between the dependent and independent variables
- Residuals follow homoscedasticity (constant variance) at every point

Hypothesis Testing for Significance of Regression Coefficients

To test the hypotheses at the $(1-\alpha)\%$ significance level:

$$H_0 : \beta_i = 0$$
$$H_a : \beta_i \neq \beta_j$$

for $i \neq j$, we reject H_0 if the p-value of the test is less than the given significance level. Otherwise, we fail to reject H_0.

Least Absolute Shrinkage and Selection Operator (LASSO) Algorithm

LASSO Algorithm

Under the Multiple Linear Regression model $y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \epsilon_i$, where the random errors $\epsilon_i \sim N(0, \sigma^2)$. Under this regression model with p predictors, the LASSO algorithm can be written as a constrained quadratic programming minimization problem:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

$$\text{subject to: } \sum_{j=1}^{p} |\beta_j| \leq t$$

- p is number of predictors and N is number of observations
- Also known as L_1 regularized regression
- Used for feature selection

LASSO Cont.

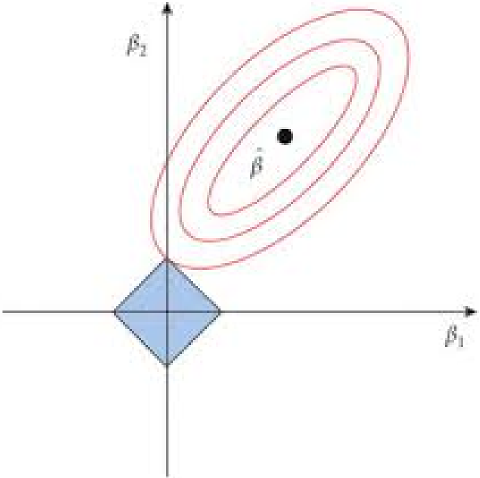


Figure: Visualization of LASSO

Proper Orthogonal Decomposition

- Proper Orthogonal Decomposition (POD) has been applied to analyzing fluid dynamics and turbulence flows
- Chatterjee provides an introduction to POD for matrix decomposition of high-dimensional data to lower-dimensional representations
- The kth order low-rank approximation of A via POD is the most efficient due to no k matrix being close to A in the Frobenius (discretized L_2) norm [Chatterjee, 2000]

Proper Orthogonal Decomposition

Given an $n \times m$ matrix A, POD decomposes the matrix into:

$$A = U \Sigma V^T \tag{11}$$

where U is $n \times n$, Σ is $n \times m$, and V^T is $m \times m$.

POD Example

Figure: POD Approximation of Matrix A Reconstruction

Overview for Models A & B

- Sparse linear regression will be used to extract the data-driven model
- LASSO regularized regression is used for feature selection to validate the data-driven model for a physics-driven model
- Once the PDEs have been discovered, numerical methods such as the Forward Euler algorithm will be used to solve for the reconstructed images from the governing equation
- Extrapolations are generated to reconstruct images and predict future extrapolations for future nonnegative time steps

Functional Form of Black-Box PDEs

Both models A and B follow the form:

$$u_t = F(u, u_x, u_{xx}, u_{xy}, u_y, u_{yy}) \tag{12}$$

where the left-hand side of equation 12 contains the time derivative and the right-hand side contains the spatial derivatives.

Data Preprocessing & Workflow

- Created 50 slides to synthetically mimic video frame splicing with equidistant object movement
- Converted images to grayscale (white $= 1$, black $= 0$)
- Computed partial derivatives of each frame (representing u) u_x, u_{xx}, u_y, u_{yy} and formed them into a data matrix
- Performed linear regression on the data matrix to extract data-driven PDE
- Performed LASSO on the data matrix to extract physics-driven PDE
- Once PDEs are obtained, perform an iterative Euler method to solve PDE u_t and reconstruct approximated images $\tilde{u}$

Data-Driven Regression Results

	coef	std err	t	P>|t|	[0.025	0.975]
x1	-0.0014	1.51e-05	-94.544	0.000	-0.001	-0.001
x2	-13.3337	0.001	-1.18e+04	0.000	-13.336	-13.331
x3	1.297e+05	1.3e+09	9.97e-05	1.000	-2.55e+09	2.55e+09
x4	95.2591	0.074	1288.315	0.000	95.114	95.404
x5	-1.297e+05	1.3e+09	-9.97e-05	1.000	-2.55e+09	2.55e+09

Omnibus:	3642833.020	Durbin-Watson:	0.000
Prob(Omnibus):	0.000	Jarque-Bera (JB):	393422392.159
Skew:	-0.039	Prob(JB):	0.00
Kurtosis:	29.481	Cond. No.	1.49e+14

Figure: Data-Driven Regression

LASSO A

```
LASSO Coefficients: [-0.00925312 -0.         -0.          0.         -0.         ]
```

Figure: LASSO Feature Selection for Model A

Physics-Driven Regression and LASSO Results

	coef	std err	t	P>|t|	[0.025	0.975]
dx	-13.3348	0.001	-1.08e+04	0.000	-13.337	-13.332

Figure: Physics-Driven Regression

Modified Image-iterative Euler Method A

Algorithm 2 Modified Image-iterative Euler Method

$N \leftarrow$ steps

$u \leftarrow u_0$

images $\leftarrow u$

for i in $1, ..., N$ **do**

$u_x \leftarrow \frac{u(x)^{(i+1)} - u(x)^{(i)}}{2h}$

$u_{xx} \leftarrow \frac{u(x)^{(i+1)} - 2u(x)^{(i)} + u(x)^{(i-1)}}{h^2}$

$u_{yy} \leftarrow \frac{u(y)^{(i+1)} - 2u(y)^{(i)} + u(y)^{(i-1)}}{h^2}$

$u_t \leftarrow -0.0014u - 13.3337u_x + 95.2591u_{xx}$

$u \leftarrow u + (dt * u_t)$

$u \leftarrow gaussian\_filter(u, \sigma = 10)$

images $\leftarrow u$

end for

Data-Driven Results

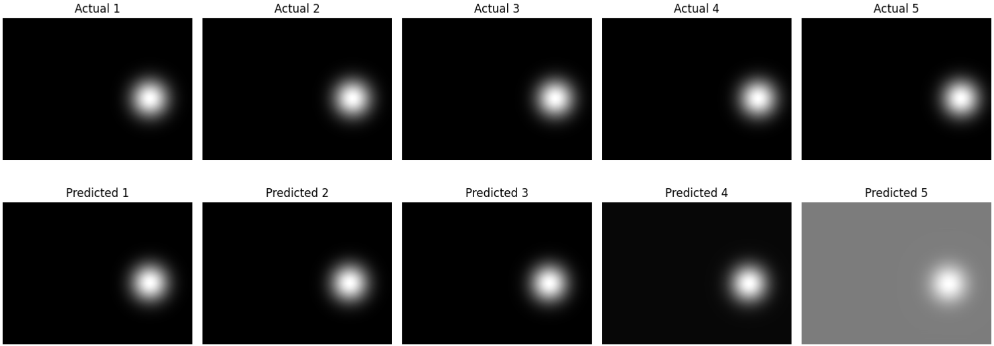


Figure: Data-Driven Extrapolations Before Denoising

Data-Driven Results Continued

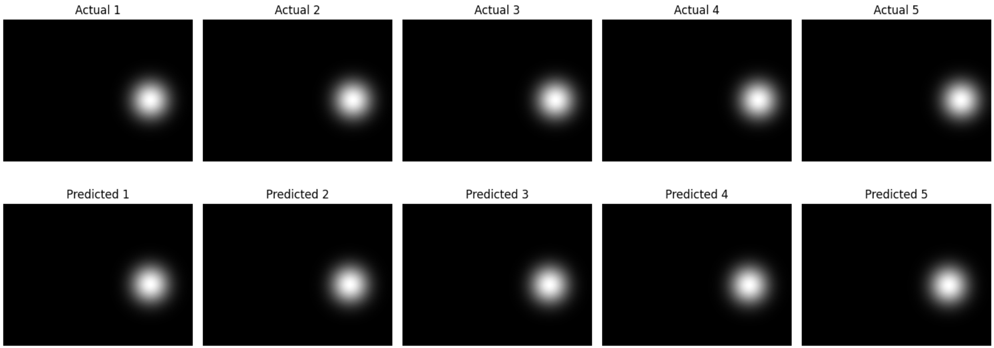


Figure: Data-Driven Extrapolations After Denoising

Physics-Driven Results

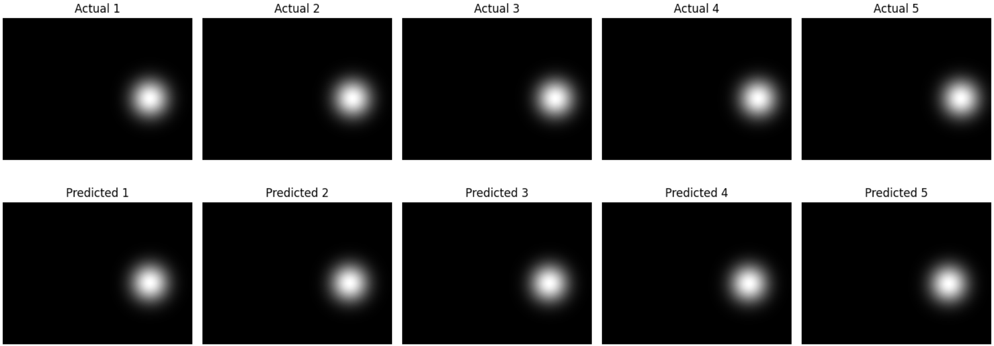


Figure: Physics-Driven Extrapolations Before Denoising

Physics-Driven Results Continued

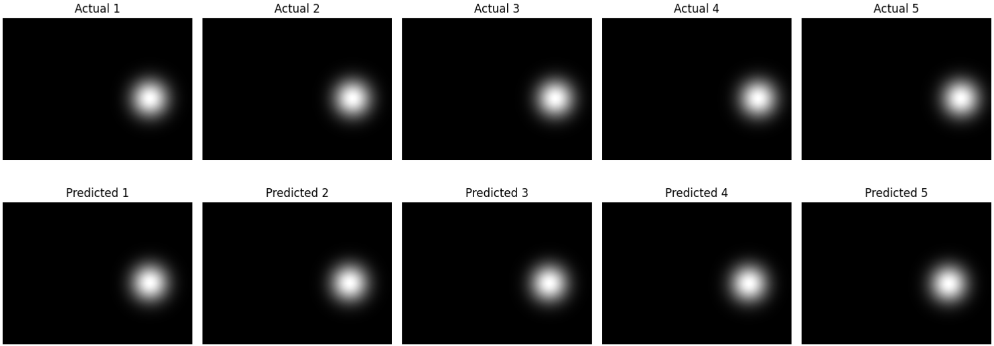


Figure: Physics-Driven Extrapolations After Denoising

Table of MSEs for Model A

Model Type	**Image 1**	**Image 2**	**Image 3**	**Image 4**	**Image 5**
Data-Driven	0.0	0.0024	0.0099	0.0221	0.0425
Physics-Driven	0.0	0.0024	0.0099	0.0221	0.0372

Table: Table of MSEs for Model A

Data Preprocessing & Workflow

- Created 50 slides to synthetically mimic video frame splicing with equidistant object movement
- Converted images to grayscale (white $= 1$, black $= 0$)
- Computed partial derivatives of each frame (representing u) u_x, u_{xx}, u_y, u_{xy}, u_{yy} and formed them into a data matrix
- We do not compute u_{yx} here due to the application of Clairaut's Theorem [Tao, 2006].

Theorem (Clairaut's Theorem)

Let $f : X, Y \to Z$ be a function on the open region $\mathbb{R} \subset \mathbb{R}^2$. If f has continuous second-order partial derivatives that exist at every point in $\mathbb{R}$, then $f_{xy} = f_{yx}$.

- Performed linear regression on the data matrix to extract data-driven PDE
- Performed LASSO on the data matrix to extract physics-driven PDE
- Once PDEs are obtained, perform an iterative Euler method to solve PDE u_t and reconstruct approximated images $\tilde{u}$

Data-Driven Regression Results

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.0321	8.47e-06	3787.074	0.000	0.032	0.032
x2	-10.9031	0.001	-1.12e+04	0.000	-10.905	-10.901
x3	4.728e+04	8.98e+08	5.26e-05	1.000	-1.76e+09	1.76e+09
x4	-3.3925	0.124	-27.409	0.000	-3.635	-3.150
x5	138.6948	0.085	1633.341	0.000	138.528	138.861
x6	-4.728e+04	8.98e+08	-5.26e-05	1.000	-1.76e+09	1.76e+09

Figure: Data-Driven Regression

LASSO B Results

```
LASSO Coefficients: [ 0.0173299 -0.         0.        -0.         0.         0.        ]
```

Figure: LASSO Feature Selection for Model B

Physics-Driven Regression Results

	coef	std err	t	P>|t|	[0.025	0.975]
u	0.0321	8.47e-06	3787.074	0.000	0.032	0.032
dx	-10.9031	0.001	-1.12e+04	0.000	-10.905	-10.901
dxy	-3.3925	0.124	-27.409	0.000	-3.635	-3.150
dxx	138.6948	0.085	1633.341	0.000	138.528	138.861
dyy	0.0031	0.001	3.165	0.002	0.001	0.005

Figure: Physics-Driven Regression

Modified Image-iterative Euler Method B

Algorithm 3 Modified Image-iterative Euler Method

$N \leftarrow$ steps
$u \leftarrow u_0$
images $\leftarrow u$
for i in $1, ..., N$ **do**
  for j in $1, ..., N$ **do**
    $u_x \leftarrow \frac{u(x)^{(i+1)} - u(x)^{(i)}}{2h}$
    $u_{xx} \leftarrow \frac{u(x)^{(i+1)} - 2u(x)^{(i)} + u(x)^{(i-1)}}{h^2}$
    $u_{yy} \leftarrow \frac{u(y)^{(i+1)} - 2u(y)^{(i)} + u(y)^{(i-1)}}{h^2}$
    $u_{xy} = \frac{u(x,y)^{(i+1,j+1)} - u(x,y)^{(i+1,j-1)} - u(x,y)^{(i-1,j+1)} + u(x,y)^{(i-1,j-1)}}{4h}$
    $u_t \leftarrow 0.0321u - 10.9031u_x - 3.3925u_{xy} + 138.6948u_{xx} + 0.0031u_{yy}$
    $u \leftarrow u + (dt * u_t)$
    $u \leftarrow gaussian\_filter(u, \sigma = 55)$
    images $\leftarrow u$
  end for
end for

Data-Driven Results

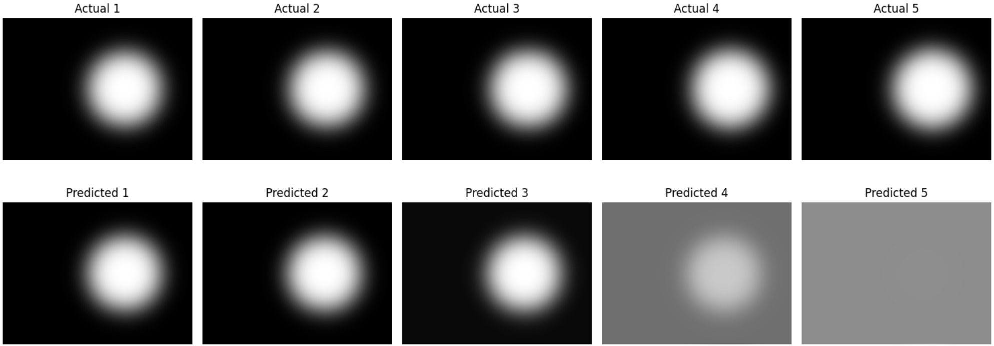


Figure: Data-Driven Extrapolations Before Denoising

Data-Driven Results Continued

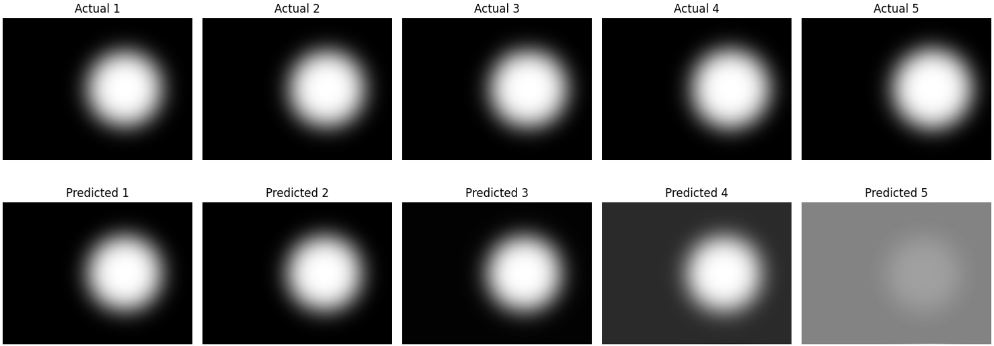


Figure: Data-Driven Extrapolations After Denoising

Physics-Driven Results

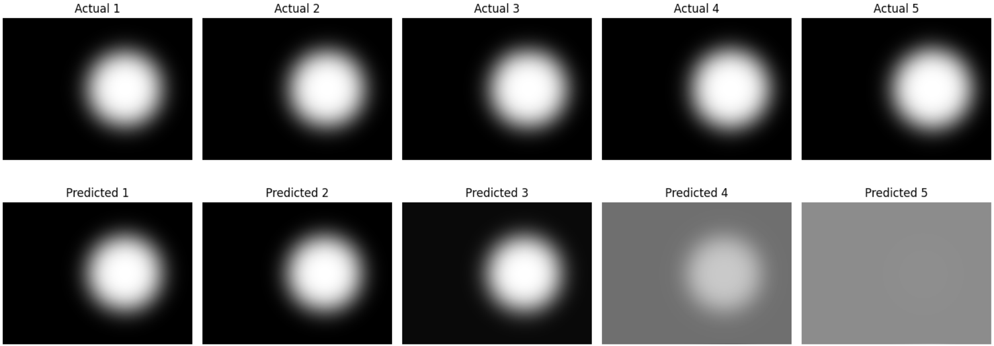


Figure: Physics-Driven Extrapolations Before Denoising

Physics-Driven Results Continued

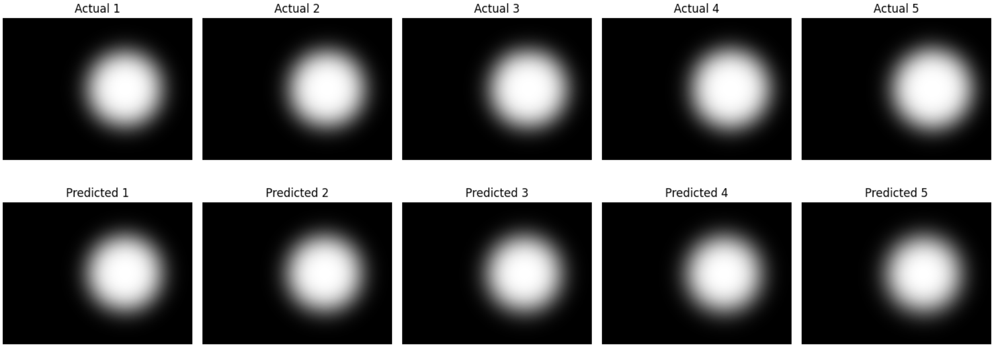


Figure: Physics-Driven Extrapolations After Denoising

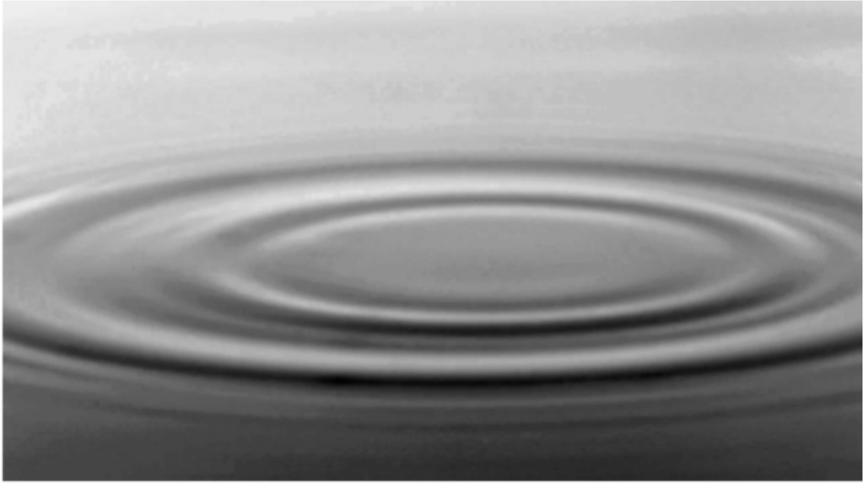
Table of MSEs of Model B

Model Type	**Image 1**	**Image 2**	**Image 3**	**Image 4**	**Image 5**
Data-Driven	0.0	0.0158	0.0501	0.1161	0.1974
Physics-Driven	0.0	0.0158	0.0501	0.1161	0.1974

Table: Table of MSEs for Model B

Data Preprocessing

- Splice video into fifty equidistant time step snapshot frames
- Transform images into grayscale prior to performing POD



Workflow

- It is hard to extract a formal black-box PDE/System of ODEs in this case due to nonlinear motion
- *Goal*: Use nonlinear regression and LASSO techniques as a data-driven approach to extract the system of solutions that would satisfy such a model
- Perform POD and extract the orthogonal component modes and time coefficients
- Approximate POD modes as sine and cosine functions of t via sparse harmonic regression and LASSO
- Formulate a linear combination of the modes and functions and generate reconstructions of original snapshots
- Further utilize this solution to extrapolate predictions of future images valid for any real-valued time step

POD Modes

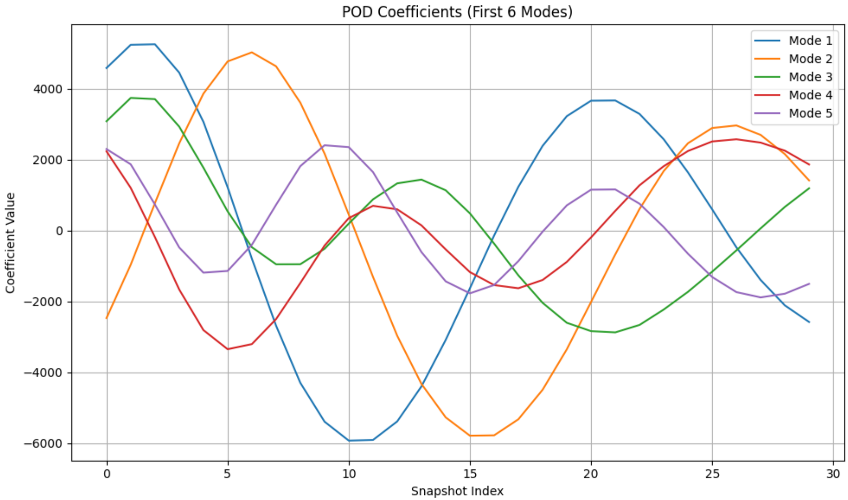


Figure: Graph of the Six POD Modes

Example of Harmonic Regression of POD Mode 5

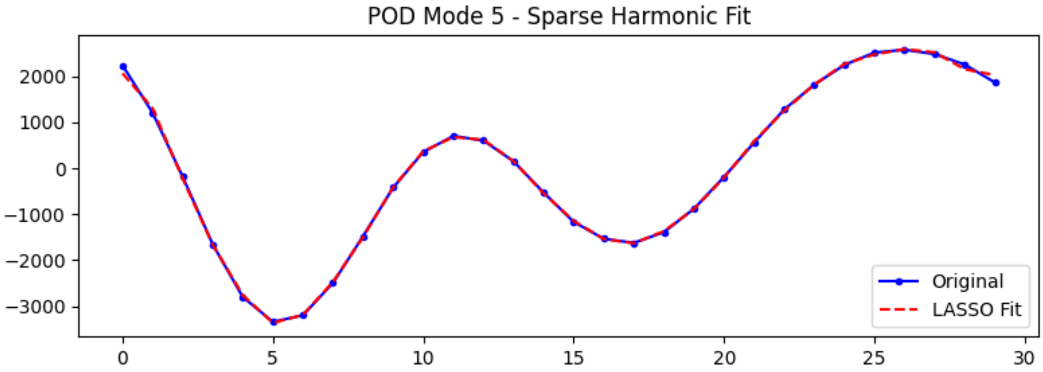


Figure: Sparse harmonic regression has been shown to be effective in capturing the nonlinear nature of the fifth POD mode

Approximated Functional Solutions from Harmonic Regression and LASSO Workflow

Approximation Example of First POD Mode

$$f_1(t) \approx -0.0483\sin(2\pi t) + 0.1467\sin(3\pi t) + 0.0222\sin(4\pi t) + 0.0122\sin(5\pi t)$$

$$+0.0314\cos(\pi t) + 0.0930\cos(2\pi t) - 0.0104\cos(4\pi t)$$

- This workflow yielded six functional approximations such as the one presented in the block above
- We can now combine POD mode functions with their coefficients and use this to generate reconstructions and extrapolations for future t

Functional Solution to the Waterdrop Model

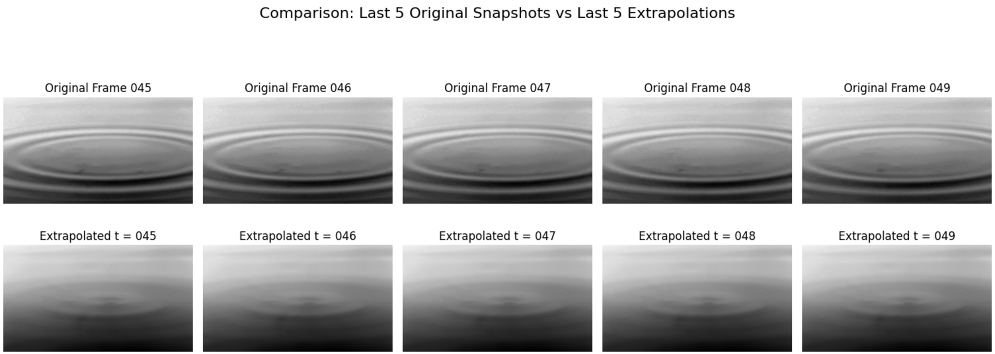
Functional Solution

After obtaining the function forms of the POD mode coefficients, we can now obtain the theoretical solution, denoted as $T(t)$, for the system of ODEs. This solution follows the form

$$T(t) = \Sigma_{i=1}^{6} f_i(t)\,\mathrm{mode}_i \tag{13}$$

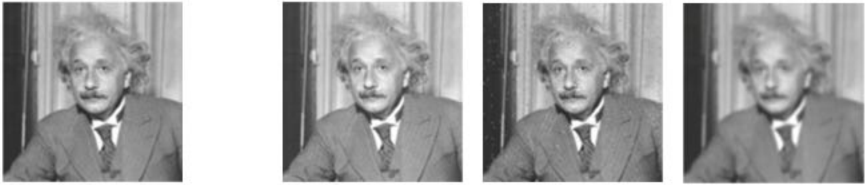
where $f_i(t)$ are the approximated functions and the modes are obtained from POD.

Extrapolations



Structural Similarity (SSIM) Index Example

- Due to the uncertainty of noise accumulation, we also introduce the SSIM metric to ensure image quality
- Closer to 0 means less similar, closer to 1 means more similar to original image



Original SSIM=1

PSNR=26.547 SSIM=0.988

PSNR=26.547 SSIM=0.840

PSNR=26.547 SSIM=0.694

Table of MSEs and SSIMs for Waterdrop Model

Image 1	**Image 2**	**Image 3**	**Image 4**	**Image 5**
29.549	27.217	28.064	29.878	27.261

Table: Table of MSEs for Waterdrop Model

Image 1	**Image 2**	**Image 3**	**Image 4**	**Image 5**
0.9812	0.9813	0.9822	0.9815	0.9824

Table: Table of SSIM Scores for Waterdrop Model

References I

[1] G. Berkooz and et al. The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. *Annual Review of Fluid Mechanics*, 25:539–575.

[2] William Boyce and Richard DiPrima. *Elementary Differential Equations and Boundary Value Problems*. John Wiley & Sons, 2012.

[3] Steven L. Brunton and Nathan J. Kutz. *Data-Driven Science & Engineering: Machine Learning, Dynamical Systems, and Control*. Brunton & Kutz, Washington, 2017.

[4] Richard L. Burden and et al. *Numerical Analysis*. Cengage Learning, Massachusetts, 2016.

[5] Anindya Chatterjee. An Introduction to the Proper Orthogonal Decomposition. *Current Science*, 78(7):808–817, 2000.

References II

[6] Cormen and et al. *Introduction to Algorithms.* The MIT Pres, 2009.

[7] Michael Kutner and et al. *Applied Linear Regression Models.* McGraw-Hill, 2004.

[8] Anne Kvaerno. *Partial Differential Equations and Finite Difference Methods.* 2020.

[9] Seungjoon Lee and et al. Course-scale PDEs from Fine-Scale Observations via Machine Learning. *arXiv*, pages 1–13, 2021.

[10] Lele Luan and et al. Uncovering Closed-Form Governing Equations of Nonlinear Dynamics from Videos. *arXiv*, pages 1–25.

References III

[11] Henry Stark and John W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice-Hall Inc., 1986.

[12] Walter A. Strauss. *Partial Differential Equations: An Introduction*. John Wiley & Sons, 2008.

[13] Terence Tao. *Analysis II*. Hindustan Book Agency, 2006.

[14] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

References IV

[15] Natsuki Tsutsumi and et al. Data-Driven ODE Modeling of the High-Frequency Complex Dynamics Via a Low-Frequency Dynamics Model. *arXiv*, 2022.

[16] Dennis Wackerly and et al. *Mathematical Statistics with Applications*. Thomson Higher Education, 2008.

Asahina Mafuyu Says...

If you would like the full source code and paper for this thesis, email Cory at: cory.suzuki-SA@csulb.edu or check out his Github: `https://github.com/CorySuzuki1729/Masters-Thesis`

The End

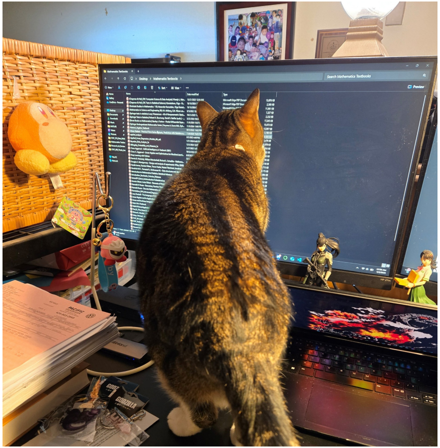


Figure: My cat Emma, not a statistician but a major contributor to my studies