Physically Informed Neural Networks (PINNs)

Calvin Ang, Jichuan Wu, Phoenix Tamaoki, Yuyang Gong

Motivation

- In biological and engineering systems often have a deficiency in data
  - Small amount
  - Noisy
- Usually have some sort of differential equation that can be used to govern the system
  - Navier-Stokes Equation
    - Highly nonlinear
    - Used to describe fluid mechanics which has many applications in engineering
- Inverse Problems
  - Given data and a differential equation what are the parameters?
- Physically Informed Neural Networks (PINNs) are capable of solving these problems

PINNs vs Traditional Models (When to use PINNs)

- Faster (Sometimes)
  - Unlike traditional models for solving differential equations, once trained, PINNs can use their neural networks to solve entire families of differential equations
    - Hemodynamics (Kissas et al 2020)
- Partial Knowledge of Parameters
  - In some cases, you may not know the values of all of your parameters in a differential equation, PINNs can solve both the differential equation and find an estimate of the parameter values by adding the parameters as training variables
    - Fluid dynamics Mao et al (2020)

PINNs vs Traditional Models (When not to use PINNs)

- Complete knowledge of Parameters, Initial Value Conditions, and Boundary Value Conditions
  - Runge-Kutta, Finite Element methods, etc. are very fast and accurate for many of these types of problems
- Time Considerations
  - Due to their nature, PINNs require training a Neural Network, if you only need to solve a small number of related differential equations, it might not be efficient to train rather than use traditional solvers

What are PINNs (Verbal Description)

- Neural networks which take limited amount of data and incorporate physical laws into the solution in order to model physical systems
  - Combination of neural networks and physical based constraints
    - Utilizes neural network framework
    - Incorporates physics principles into loss
    - Boundary and initial conditions

What are PINNs (Mathematical Description)

Suppose the data we have should obey some physical law, namely, for some domain Ω (this could be the time-frame we are interested in, a portion of the plane, etc.), we have that $\forall z \in \Omega \subseteq \mathbb{R}^d$ that

$$\mathcal{F}(\mathbf{u}(\mathbf{z}); \gamma) = \mathbf{f}(\mathbf{z}). \tag{1}$$

where $\mathbf{z} = [x_1, ..., x_{d-t}; t]$ and γ is the set of parameters related to the physical system, usually an Ordinary Differential Equation (ODE) or a Partial Differential equation (PDE).

What are PINNs (Mathematical Description)

These neural networks have a loss function made up of the sum of a data loss and a physical loss. Let us call the physics loss $\mathcal{L}_{\mathcal{F}}$ and the data loss $\mathcal{L}_{\text{data}}$. We define

$$\mathcal{L}_{\text{data}} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left(\mathbf{u}(\mathbf{z}_i) - \hat{\mathbf{u}}(\mathbf{z}_i)\right)^2 \tag{2}$$

where N_d is the number of points we use to calculate the data loss and $\mathbf{z}_i \in \Omega$ $(i = 1, 2, ..., N_{\mathcal{F}})$ are the points in the domain at which we test the data loss

$$\mathcal{L}_{\mathcal{F}} = \frac{1}{N_{\mathcal{F}}} \sum_{i=1}^{N_{\mathcal{F}}} \left(\mathcal{F}(\hat{\mathbf{u}}(\hat{\mathbf{z}}_i); \gamma) - \mathbf{f}(\hat{\mathbf{z}}_i)\right)^2 \tag{3}$$

where $N_{\mathcal{F}}$ is the number of points we use to calculate the physics loss and $\hat{\mathbf{z}}_i \in \Omega$ $(i = 1, 2, ..., N_{\mathcal{F}})$ are the points in the domain at which we test the physics loss.

What are PINNs (Mathematical Description)

We then have that the problem becomes

$$\theta^* = \text{argmin}_\theta(w_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\theta) + w_d\mathcal{L}_{\text{data}}(\theta))$$

where $w_{\mathcal{F}}$ and w_d are experimentally determined weights and θ encompasses the weights within the neural network

Designing PINNs

- Choose Neural Network Architecture
  - Activation Function
    - Usually tanh
  - How Many Layers
    - Usually deep
  - Neural Network
    - Feed-forward
- Choose optimization algorithm
  - Adams or BFGS
- Create loss function
  - Use data error, physics error
- Train Neural Network
  - Train the PINN and the sample data, and use optimization algorithm to minimize the data and physics loss

Properties

- Error Analysis
  - The underlying architecture of the neural network determines how low error can theoretically get
  - Little research existing regarding convergence of PINNs
    - Empirical evidence to show convergence in many cases
    - Some cases where PINNs fail to train at all
- Continuity
  - Given the way PINNs work, unlike classical methods, these are meshless methods
    - Can output a prediction for any value on the domain

Experiment: Logistic Growth

$$\frac{dP}{dt} = rP\left(1 - \frac{P}{K}\right)$$

- $P(t)$ is the population at time t
- r is the growth rate of the population
- K is the carrying capacity of the environment

Designing Loss Functions Example

$$\mathcal{L}_{\text{data}} = \frac{1}{N_s}\sum_{i=1}^{N_s}(p_i - \tilde{p}_i)^2.$$

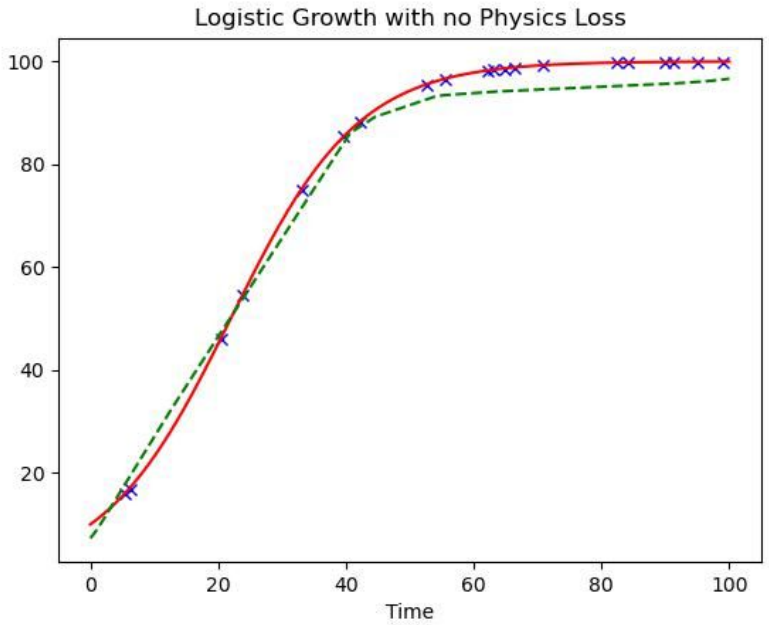
$$\mathcal{L}_{\text{phys}} = \frac{1}{N_p}\sum_{i=1}^{N_p}\left(\frac{d\hat{p}_i}{dt}|_{t=\tau_i} - r\hat{p}_i\left(1 - \frac{\hat{p}_i}{K}\right)\right)^2$$

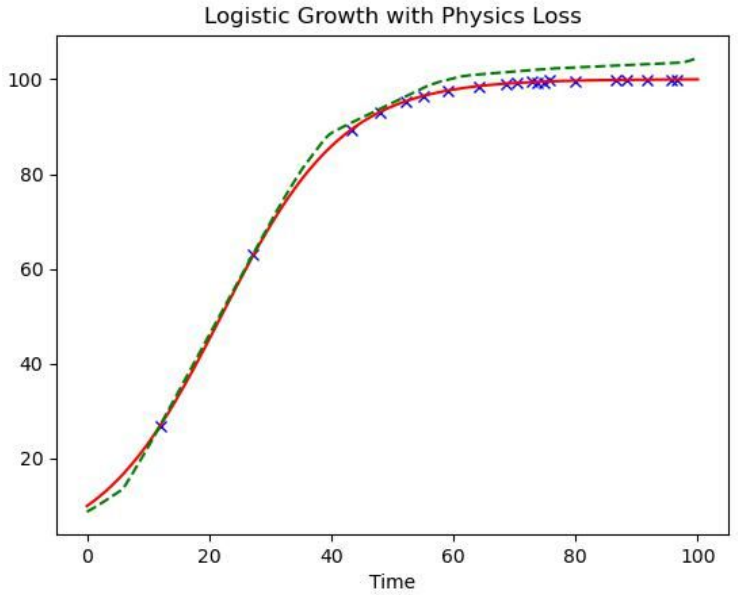
$$\theta^* = \text{argmin}_\theta(w_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\theta) + w_d\mathcal{L}_{\text{data}}(\theta))$$

- p_i are the sample populations at time t_i (the ith sample time)
- $\tilde{p}_i$ are the predicted populations at time t_i
- $\hat{p}_i$ are the predicted populations at the times τ_i (the times we test the physics loss)
- $\frac{d\hat{p}_i}{dt}|_{t=\tau_i}$ is the derivative of the predicted population with respect to time at time τ_i

Experiment 1.1/1.2

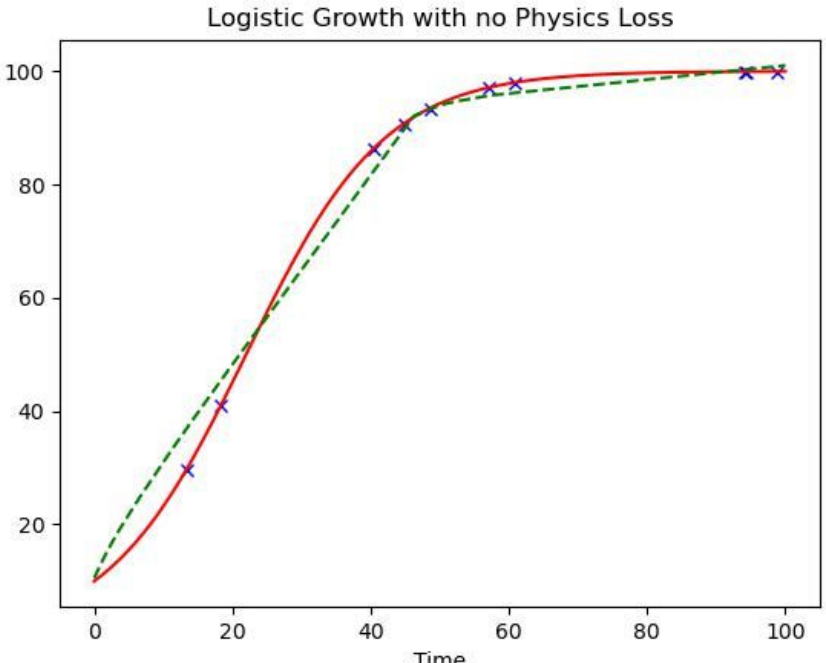
- 700 epochs with training data from whole domain with 20 samples

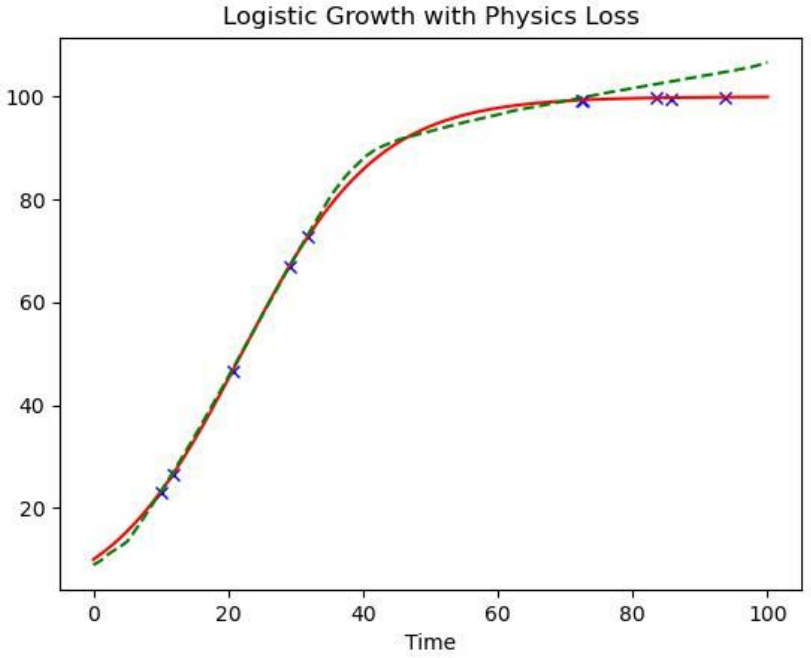




Experiment 1.3/1.4

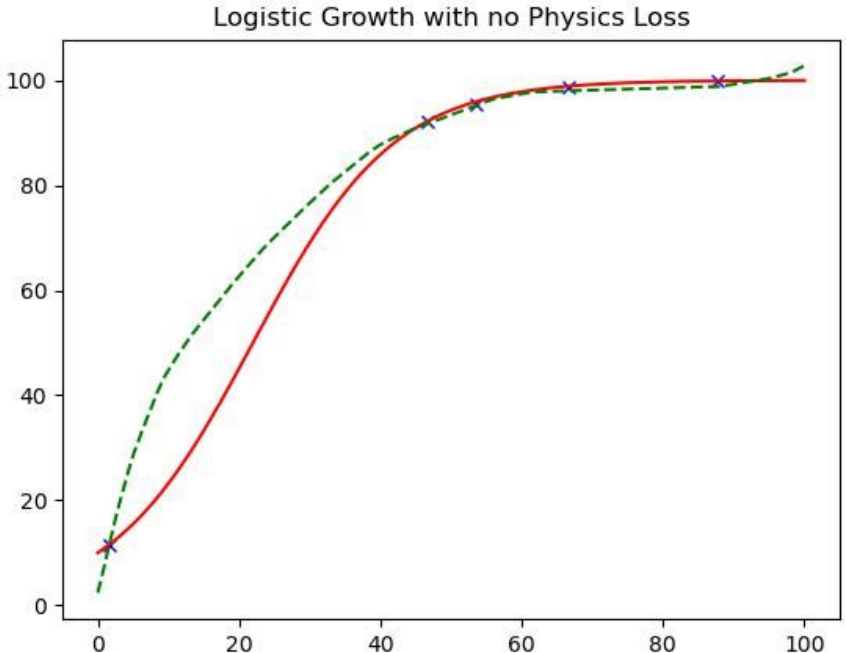
- 700 epochs with training data from whole domain with 10 samples

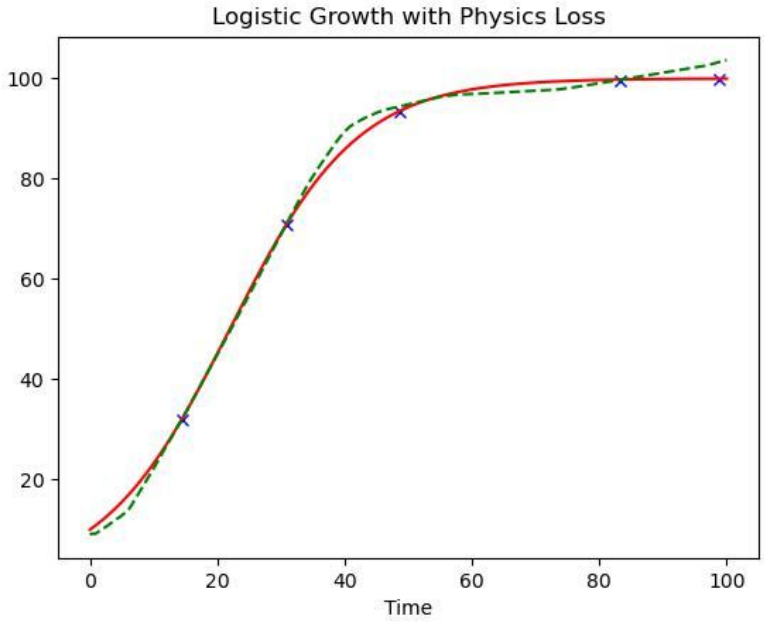




Experiment 1.5/1.6

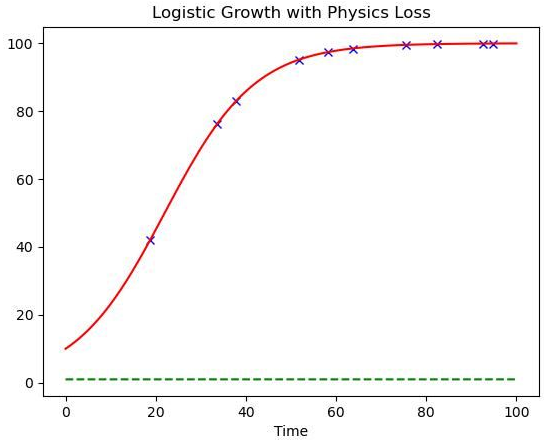
- 700 epochs with training data from whole domain with 5 samples (regularization of physics loss with weight 0.01 rather than the earlier 0.0001)





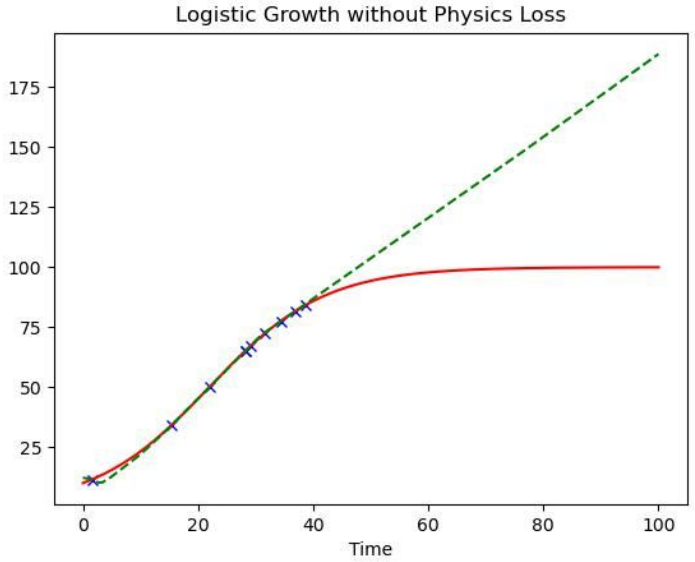
Experiment 1.7

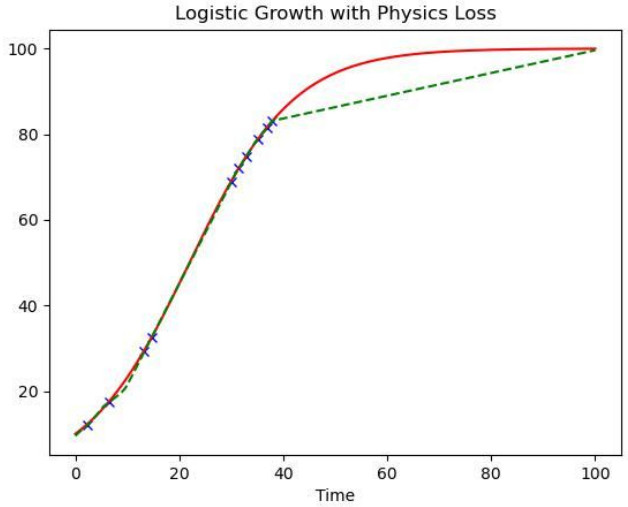
- In this experiment we used the tanh activation function and as can be seen here we have that the model is not capable of learning. We believe this might be because the model gets stuck at a local optimum. We also think that this might be because of the phenomenon of vanishing gradients problem. This occurs when the gradient becomes extremely small in magnitude and thus the update does almost nothing.



Experiment 1.8/1.9

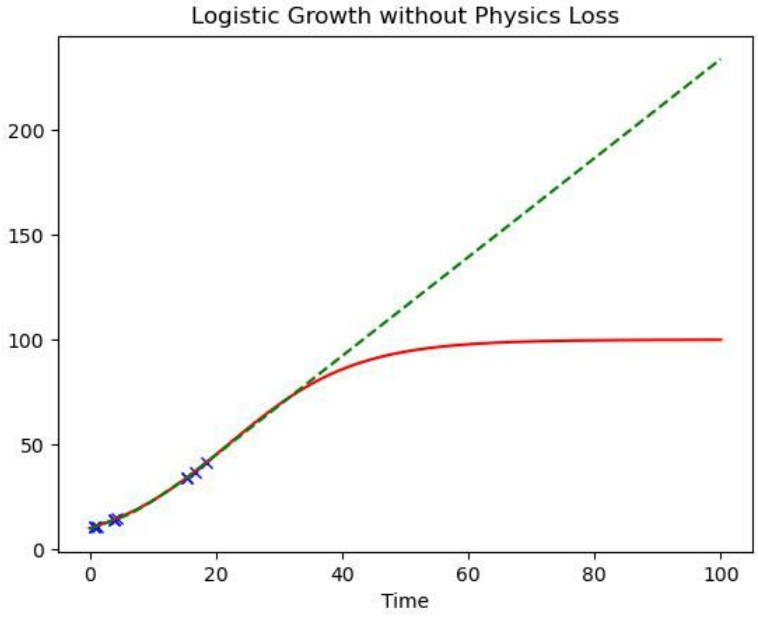
- 20000 epochs, data range from 0-40

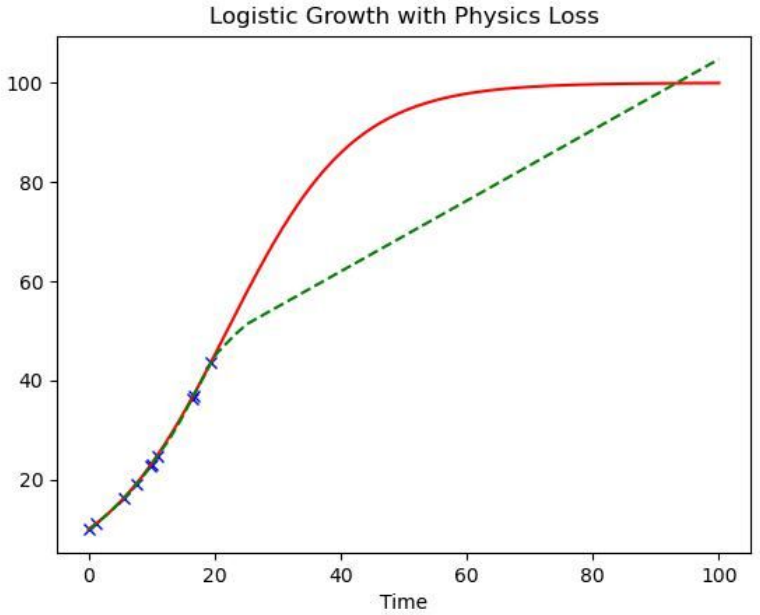




Experiment 1.10/1.11

- 20000 epochs, data range from 0-20



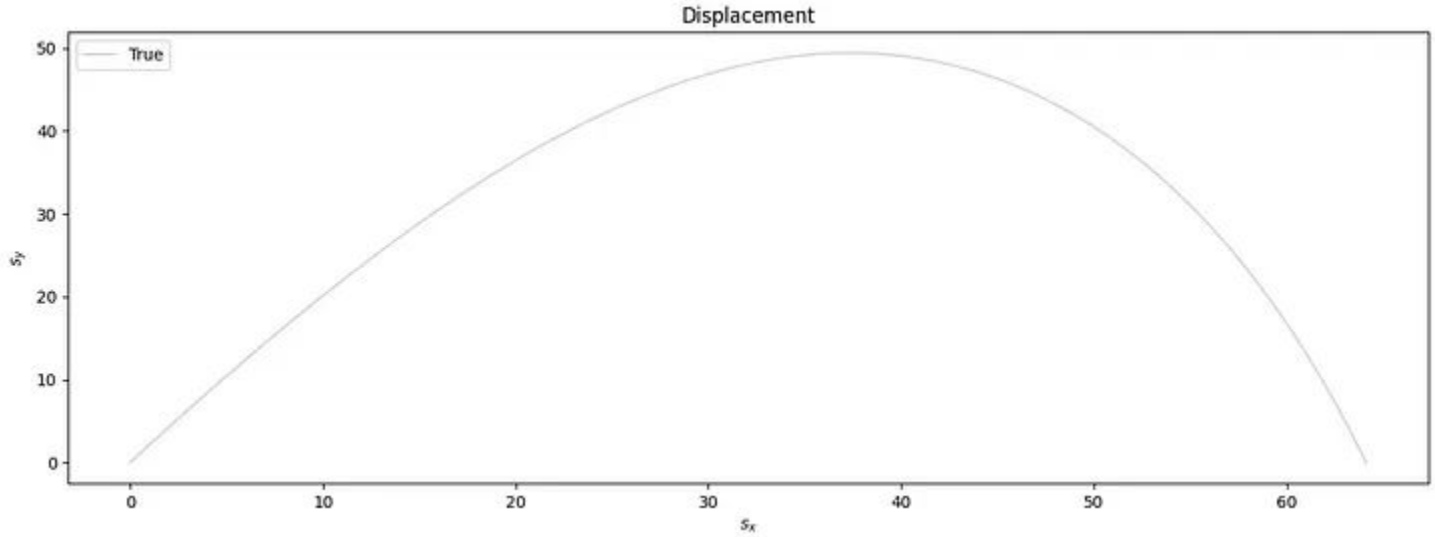


Experiments: Projectile Motion

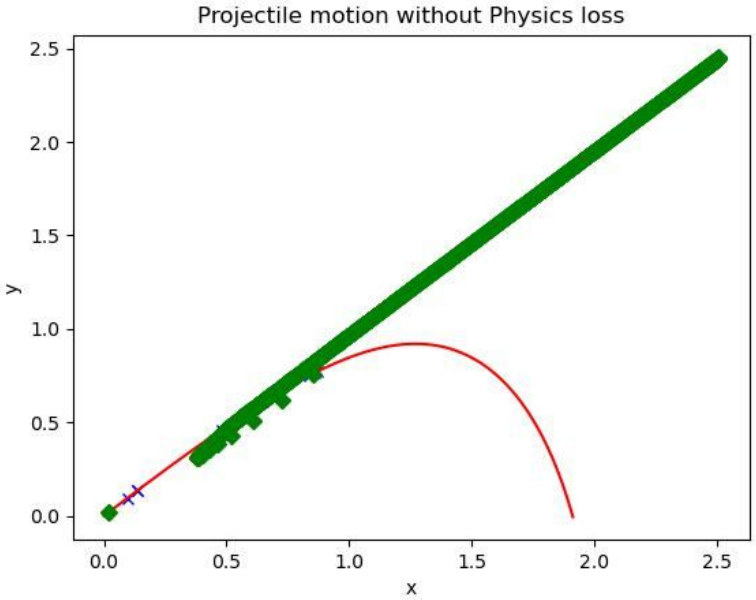
$$\frac{d^2\vec{s}}{dt^2} = -\mu \left\| \frac{d\vec{s}}{dt} \right\|_2 \frac{d\vec{s}}{dt} - \vec{g}$$

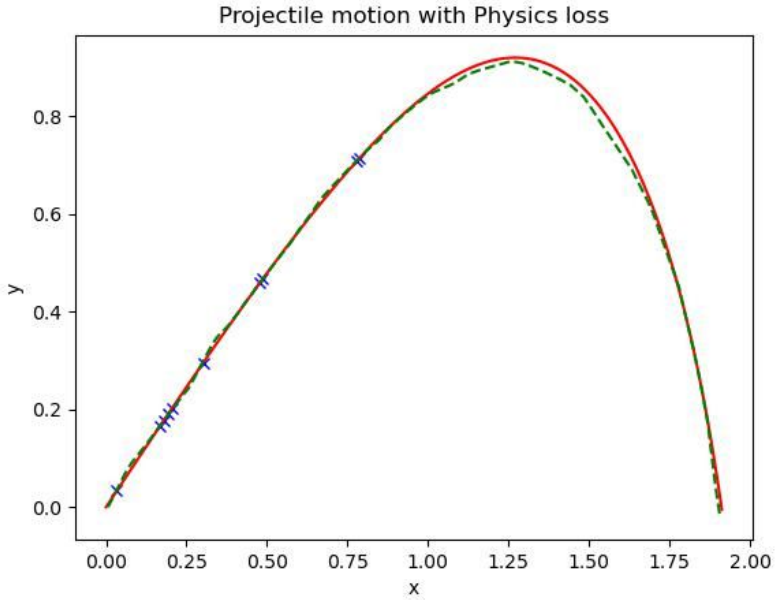
- $\vec{s}(t)$ is the vector composed of x and y components of position as a function of time
- μ is the coefficient of drag
- $\vec{g}$ is the vector of acceleration due to gravity

Experiments: Projectile Motion



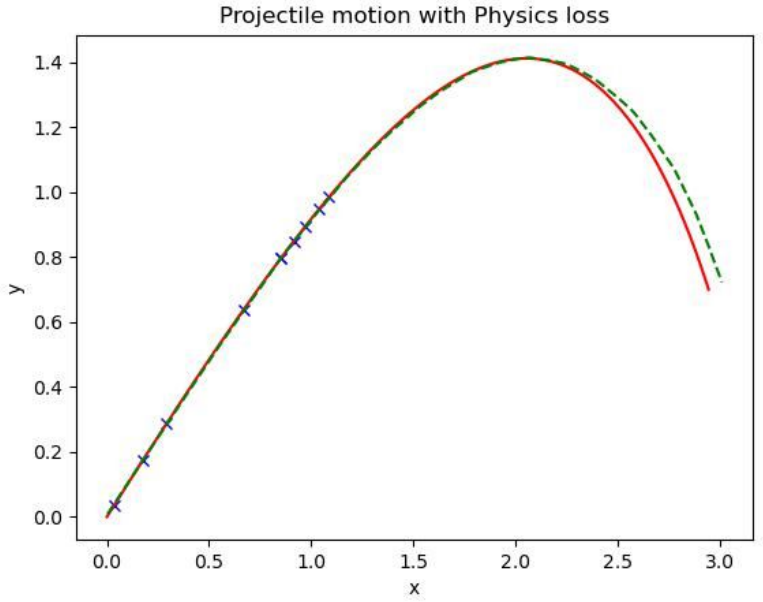
Experiment 2.1 / 2,2



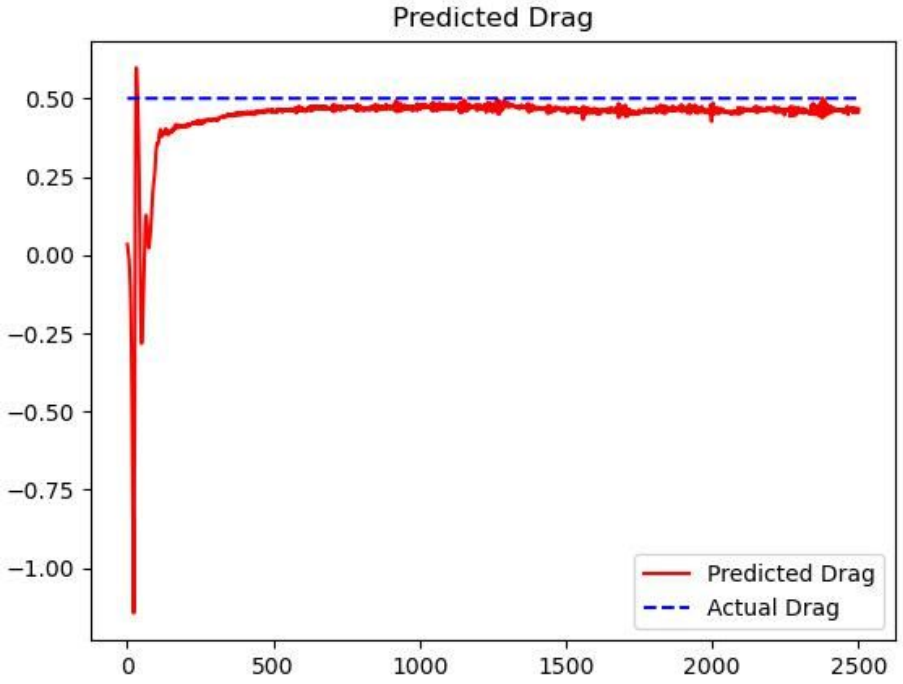


Experiment 2.3

- Drag coefficient learned



Experiment 2.4



Moving on…

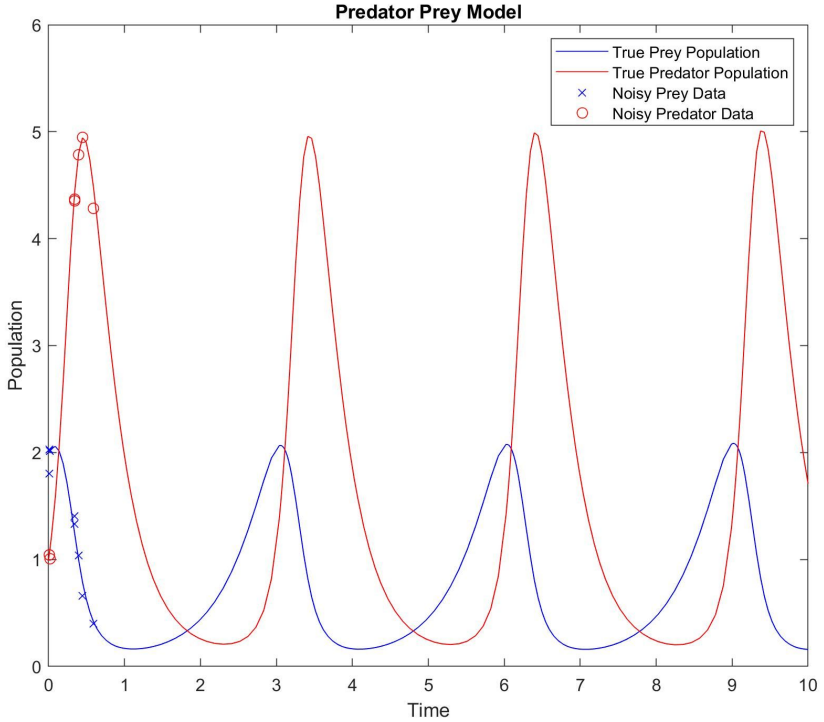
Next, we generalize to a more complex nonlinear ODE, the Lotka-Volterra Equation which models the population dynamics of an ecosystem consisting of a prey species and a predator species.

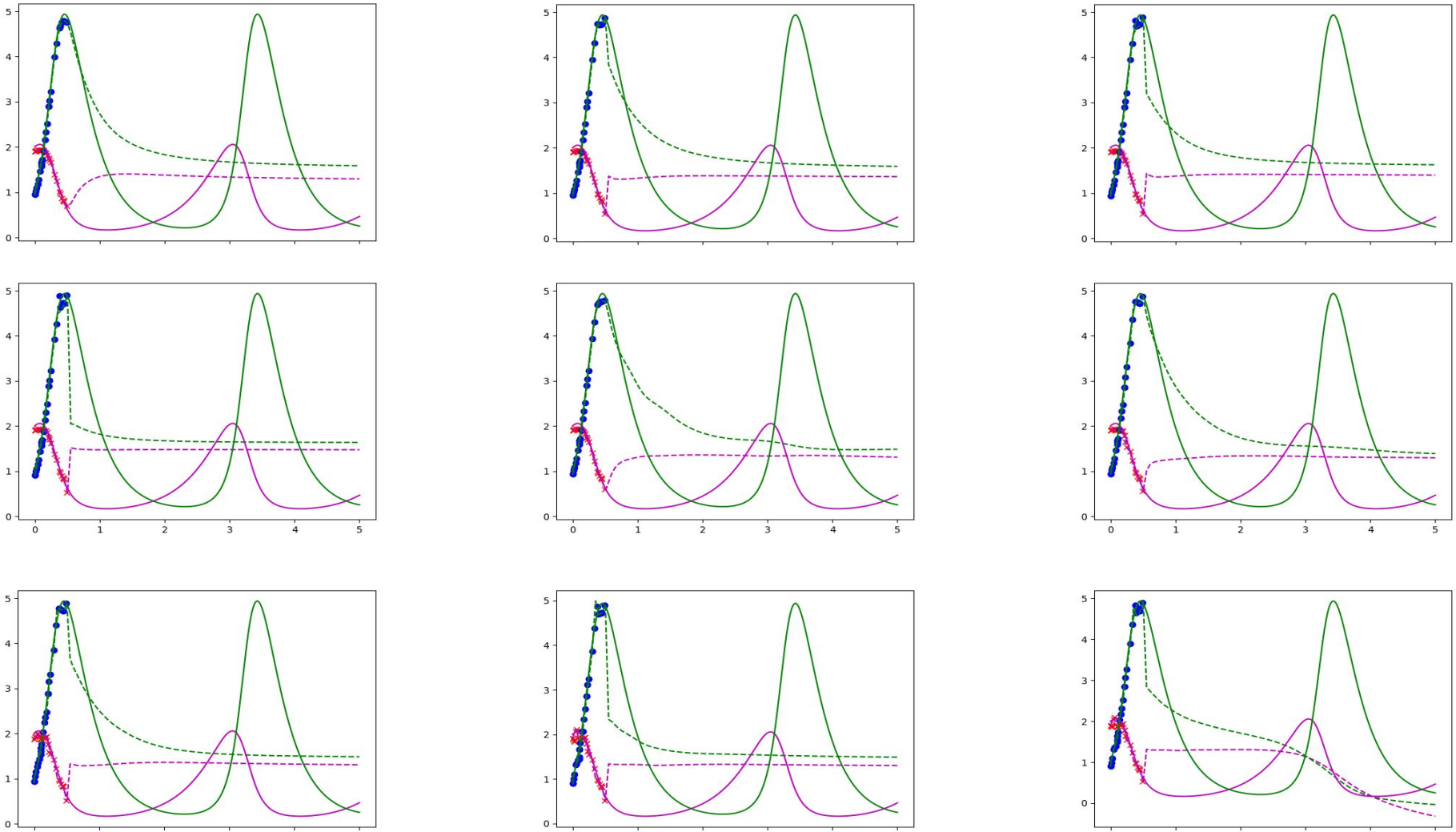
Lotka–Volterra equations

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

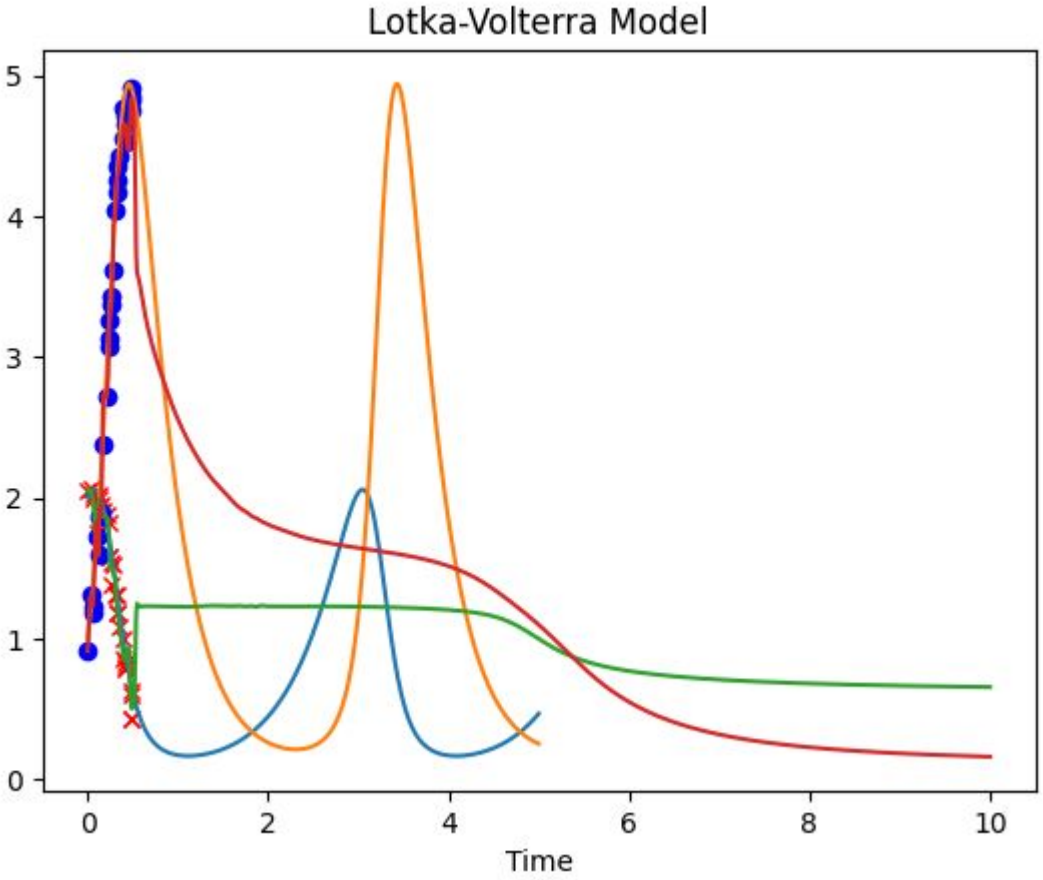
- $x(t)$ is the prey population at time t
- $y(t)$ is the predator population at time t
- α prey birth rate
- β prey death rate
- δ predator birth rate
- γ prey death rate

Sample Data


Predator Prey Model
True Prey Population
True Predator Population
Noisy Prey Data
Noisy Predator Data
Population
Time
0
1
2
3
4
5
6
7
8
9
10



Final Model



Future Research Directions

- Error Analysis
  - As discussed earlier more research needs to be done regarding when PINNs converge and when they may not
- Architecture
  - Specialized optimization algorithms
  - Selection of learning rates

Works Cited

Cuomo, S. et al. (no date) Scientific machine learning through physics-informed neural ... - arxiv.org. Available at: https://arxiv.org/pdf/2201.05624.pdf.

Henderson, I. (2022) Physics informed Neural Networks (pinns): An intuitive guide, Medium. Available at: https://towardsdatascience.com/physics-informed-neural-networks-pinns-an-intuitive-guide-fff138069563.

Kissas G, Yang Y, Hwuang E, et al (2020) Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering 358:112,623. https://doi.org/10.1016/j.cma.2019.112623, URL https://www.sciencedirect.com/science/article/pii/ S0045782519305055

Markidis, S. (2021) The old and the new: Can physics-informed deep-learning replace traditional linear solvers?, Frontiers. Available at: https://www.frontiersin.org/articles/10.3389/fdata.2021.669097/full.

Mao Z, Jagtap AD, Karniadakis GE (2020) Physics-informed neural networks for high-speed flows. Computer Methods in Applied Mechanics and Engineering 360:112,789. https://doi.org/10.1016/j.cma.2019.112789, URL https://www.sciencedirect.com/science/article/pii/S0045782519306814

Mishra, S. and Molinaro, R. (2021) Estimates on the generalization error of physics informed Neural Networks (pinns) for approximating pdes, arXiv.org. Available at: https://arxiv.org/abs/2006.16144.