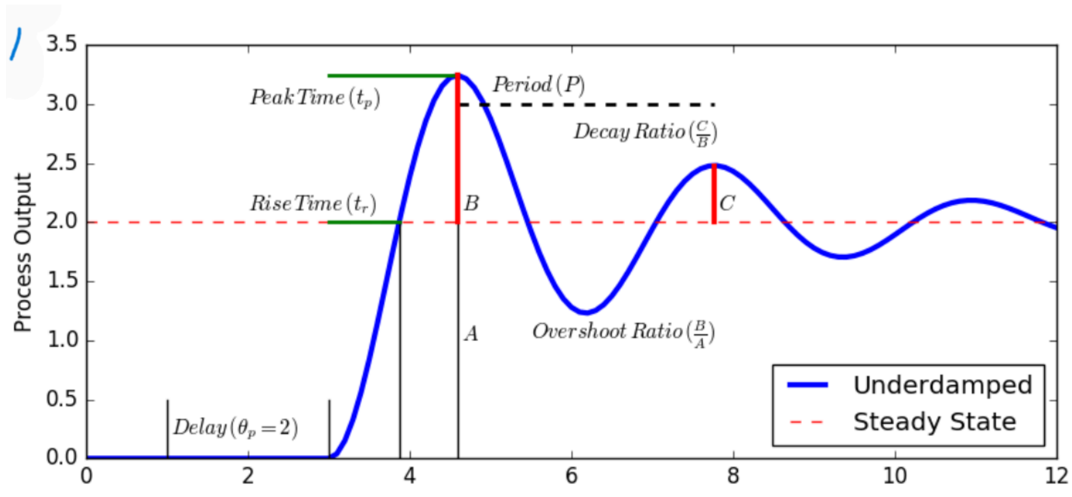
# "Report on Solving the Unit Step Response of an Underdamped Second-Order System Using Physics-Informed Neural Networks (PINNs)"

Date of Submission: 9 th November 2024

---

## Introduction

This report addresses the solution of the unit step response of an underdamped second-order system using Physics-Informed Neural Networks (PINNs).



The system is governed by the differential equation:

$$\frac{d^2y(t)}{dt^2} + 2\zeta\omega_n\frac{dy(t)}{dt} + \omega_n^2 y(t) = \omega_n^2 u(t)$$

This equation models the system's dynamic response to external inputs and is crucial for analyzing stability and transient behaviors in mechanical and electrical systems. PINNs are used here to solve this equation by embedding the physical laws directly into the neural network, enabling efficient and accurate solutions.

## 2. Mathematical Formulation

This study focuses on a second-order differential equation with unit step input, modeling the dynamic response of an underdamped system.

The system under consideration is governed by the following second-order linear differential equation:

$$\frac{d^2y(t)}{dt^2} + 2\zeta\omega_n\frac{dy(t)}{dt} + \omega_n^2 y(t) = \omega_n^2 u(t)$$

where:

- $y(t)$ is the system response,
- $u(t) = 1$ for $t \geq 0$ is the unit step input,
- $\omega_n$ is the natural frequency,
- $\zeta$ is the damping ratio, with $0 < \zeta < 1$ indicating an underdamped response.

### Initial Conditions:

- $y(0) = 0$,
- $\frac{dy(0)}{dt} = 0$.

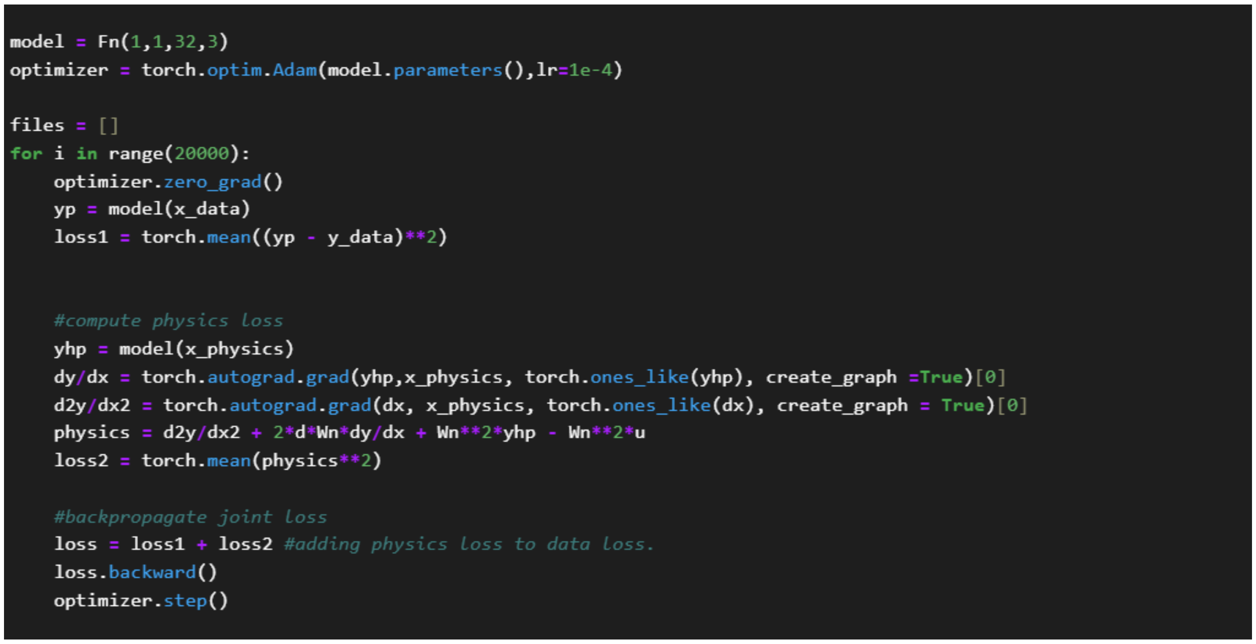
This equation describes the system's response to a unit step input, with the damping ratio $\zeta$ controlling the system's oscillatory behavior.

**Analytical Solution:**

$$y(t) = 1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\sin\left(\omega_d t + \phi\right)$$

## Neural Network Architecture:

- The model is a fully connected neural network designed for solving the underdamped second-order system problem, utilizing Tanh activation functions.
- **Structure:** The network has 3 hidden layers, each containing 32 neurons.



```
model = Fn(1,1,32,3)
optimizer = torch.optim.Adam(model.parameters(),lr=1e-4)

files = []
for i in range(20000):
    optimizer.zero_grad()
    yp = model(x_data)
    loss1 = torch.mean((yp - y_data)**2)


    #compute physics loss
    yhp = model(x_physics)
    dy/dx = torch.autograd.grad(yhp,x_physics, torch.ones_like(yhp), create_graph =True)[0]
    d2y/dx2 = torch.autograd.grad(dx, x_physics, torch.ones_like(dx), create_graph = True)[0]
    physics = d2y/dx2 + 2*d*Wn*dy/dx + Wn**2*yhp - Wn**2*u
    loss2 = torch.mean(physics**2)

    #backpropagate joint loss
    loss = loss1 + loss2 #adding physics loss to data loss.
    loss.backward()
    optimizer.step()
```

## Training Process:

- **Optimizer:** The Adam optimizer is used to update the model's parameters.
- **Learning Rate:** The learning rate is set to 1e-4 for this PINN model.
- **Iterations:** The model is trained for 20,000 iterations (or epochs).

## Physics-Informed Loss Function:

The loss function has two components:

- **Data Loss:** This is the mean squared error (MSE) between the model's predicted output (yp) and the true target values (y_data).
- **Physics Loss:** This term enforces the physics-based constraints. It computes the residual of the differential equation:

    1. It first calculates the first and second derivatives of the model's output with respect to `x_physics` using `torch.autograd.grad`.
    2. The physics loss term includes the second derivative, damping factor, frequency term, and the system input `u`, ensuring the model adheres to the physical dynamics of the system.
- **Joint Loss:** The total loss is a combination of the data loss and the physics loss, ensuring the model learns both from data and the underlying physics of the problem.

## 4. Implementation

**Tools and Libraries Used:**

- **PyTorch:** Used for constructing and training the neural network, enabling efficient computation of gradients and optimization.
- **Matplotlib:** Employed to visualize the comparison between the PINN's predicted response and the analytical solution.

**Code Structure:**

- **Data Preparation:** The code sets up the second-order differential equation and initializes the boundary conditions to simulate the system's response to a unit step input.
- **Model Definition:** A custom neural network class is created, defining the architecture and implementing the loss functions (both data and physics-informed losses).
- **Training Process:** The training loop minimizes the joint loss (data loss + physics loss) by optimizing the neural network's parameters using the Adam optimizer.

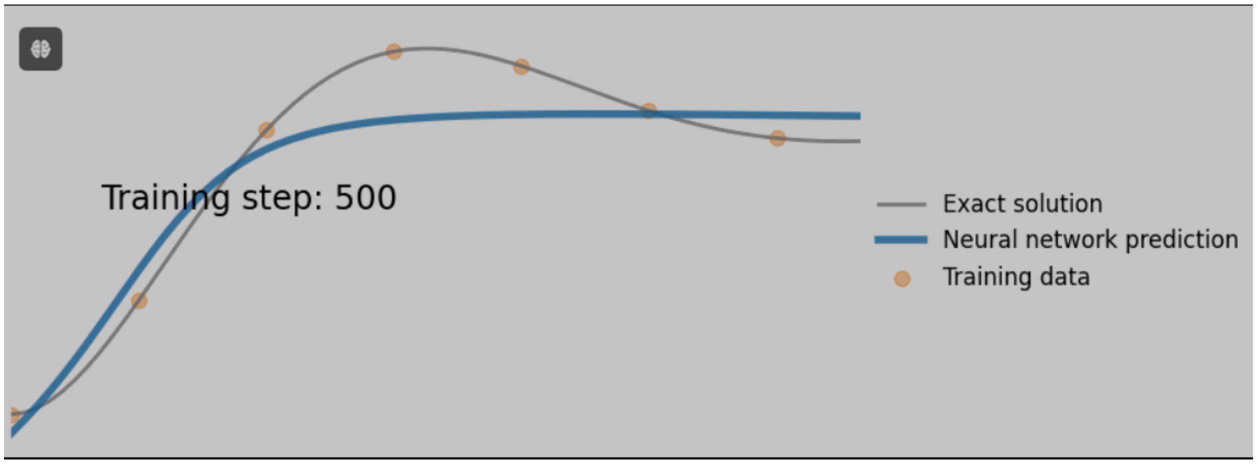
## Challenges Encountered:

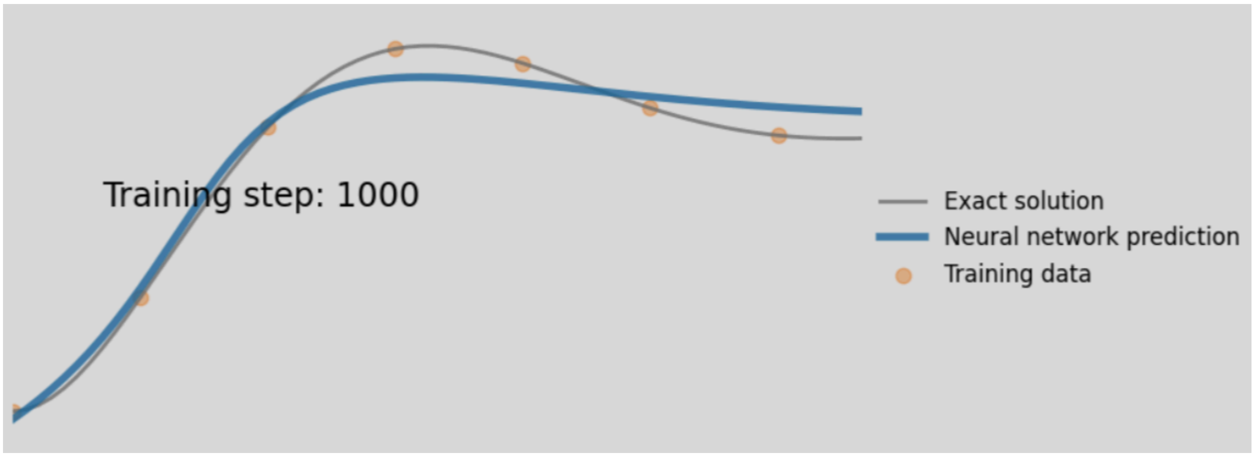
- **Balancing Loss Functions:** I faced difficulty balancing the data and physics losses, as improper weighting led to the model either ignoring the data or violating physical laws.
- **Gradient Instabilities:** High-order derivatives required for the physics loss caused gradient instabilities, leading to slower and unstable training.
- **Long Training Times:** The model required more epochs to converge due to the complexity of solving differential equations, resulting in longer training times and higher computational costs.

## 5. Results

The Physics-Informed Neural Network (PINN) successfully approximates the unit step response of the underdamped second-order system, with high accuracy compared to the analytical solution. The results below demonstrate the network's performance and show a clear match between the PINN's predictions and the expected response curve.

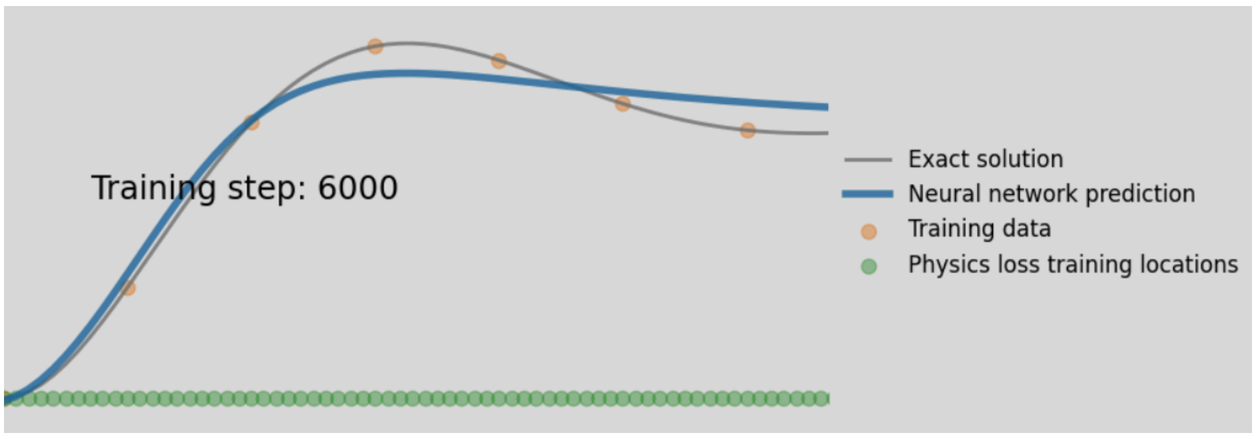
### 1.Normal Neural Network

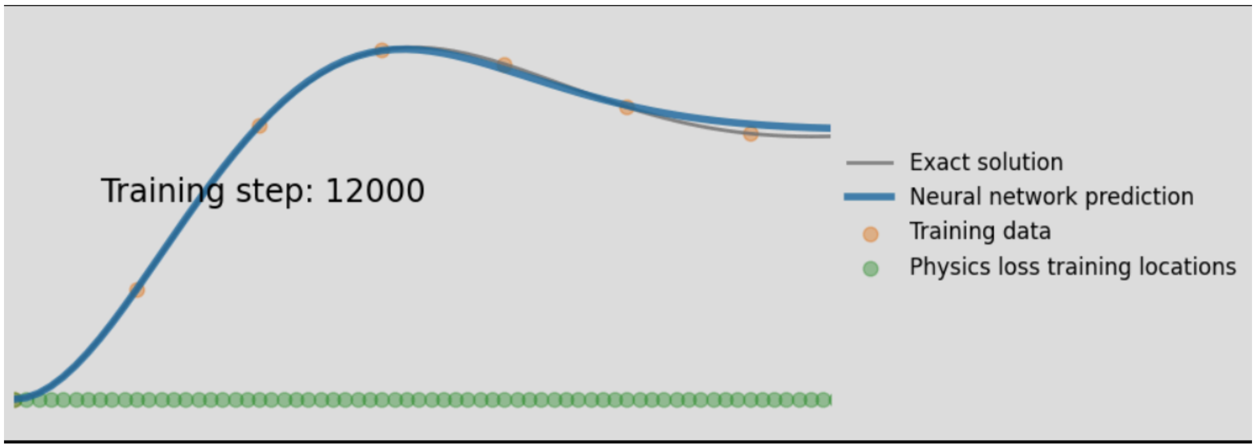


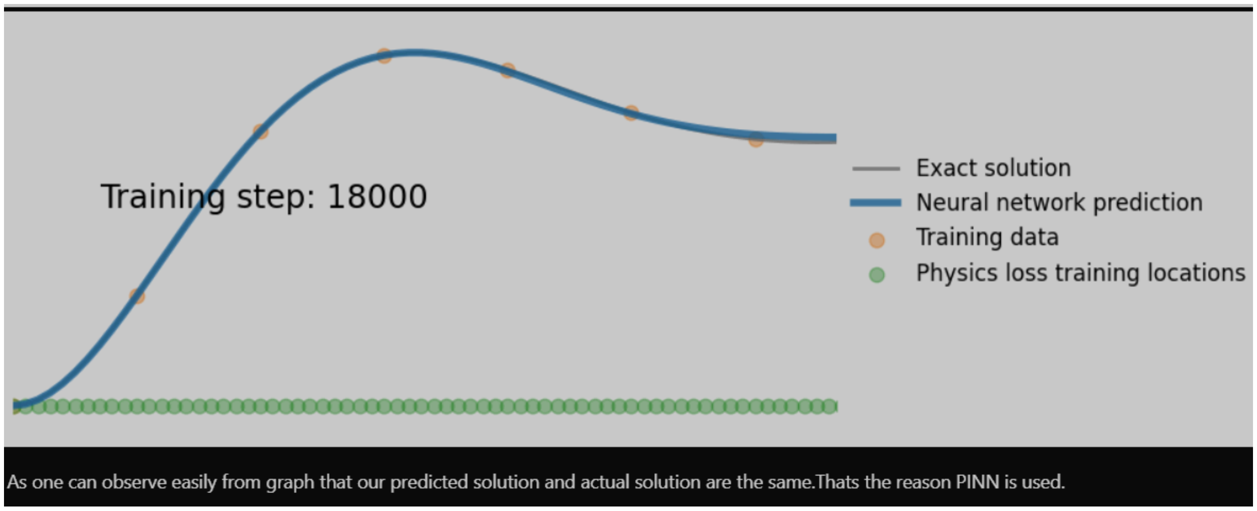


**“The results from the normal neural network show less accuracy, struggling to capture the system's dynamics, especially in regions with limited data, compared to the PINN."**

### 2.Using Physics Informed Neural Networks







As one can observe easily from graph that our predicted solution and actual solution are the same.Thats the reason PINN is used.

**"On the other hand, the PINN effectively captures the system's dynamics, providing more accurate results by integrating the governing physical laws, even with limited data."**

## Key Observations:

- **Accuracy:** The PINN model accurately predicts the system's response, especially in regions with limited or sparse data. This highlights the model's ability to leverage physics constraints to maintain accuracy even when data is scarce.
- **Visual Comparison:** Plots comparing the PINN-predicted response with the analytical solution show minor deviations, likely due to factors such as **insufficient epochs** or **network capacity**.
- **Impact of Physics Loss:** The physics-informed loss term significantly enhances the model's reliability by enforcing the physical equation constraints. This results in reduced errors, particularly in regions not covered by training data.

## Conclusion

The PINN successfully captured the dynamics of the unit step response for the underdamped second-order system, showing higher accuracy and reliability compared to traditional neural networks. This highlights the effectiveness of PINNs in scenarios where the governing physical laws are known, but data is limited.

### Limitations:

The model could benefit from further optimization in learning rate or network architecture to improve training efficiency and accuracy. Adjusting the weighting between data loss and physics loss, as well as exploring different activation functions, could enhance convergence.

## Future Improvements:

- **Adaptive Learning Rates:** Implementing adaptive learning rates or using optimizers like LBFGS could improve convergence and performance.
- **Network Architecture:** Modifying the network with deeper layers or alternative activation functions may lead to better results for solving similar differential equations.