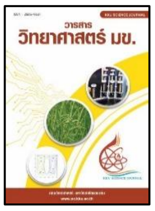
KKU SCIENCE JOURNAL

KKU SCIENCE JOURNAL

Journal Home Page : https://ph01.tci-thaijo.org/index.php/KKUSciJ

Published by the Faculty of Science, Khon Kaen University, Thailand



การพยากรณ์วิถีการเคลื่อนที่ของจรวดขวดน้ำโดยใช้โครงข่ายประสาทเทียมที่ฝังหลักฟิสิกส์ (PINNs) เปรียบเทียบกับโมเดลเรียนรู้เชิงลึก

Trajectory Prediction of a Water Rocket Using Physics-Informed Neural Networks (PINNs) Compared with Deep Learning Models

นิทัศน์ ศรีพงษ์พันธ์[1] และ สิทธิโชค โสมอ่ำ[2*]

Nitat Sripongpun[1] and Sittichoke Som-am[2*]

[1]สาขาวิชาฟิสิกส์ โรงเรียนมหิดลวิทยานุสรณ์ จังหวัดนครปฐม 73170

[2*]สาขาวิชาคณิตศาสตร์และวิทยาการคำนวณ โรงเรียนมหิดลวิทยานุสรณ์ จังหวัดนครปฐม 73170

[1]Department of Physics, Mahidol Wittayanusorn School, Nakhon Pathom, 73170, Thailand

[2*]Department of Mathematic and Computing Science, Mahidol Wittayanusorn School, Nakhon Pathom, 73170, Thailand

บทคัดย่อ

จรวดขวดน้ำเป็นสื่อการทดลองที่ช่วยเสริมสร้างความเข้าใจแนวคิดฟิสิกส์พื้นฐาน เช่น กฎการเคลื่อนที่ของนิวตันและการเคลื่อนที่แบบโปรเจกไทล์ งานวิจัยนี้นำเสนอการเปรียบเทียบประสิทธิภาพระหว่างโมเดลโครงข่ายประสาทเทียมเชิงลึก (Deep Learning: DL) และโครงข่ายประสาทเทียมที่ฝังหลักฟิสิกส์ (Physics-Informed Neural Networks: PINNs) ในการพยากรณ์วิถีการเคลื่อนที่ของจรวดขวดน้ำแบบสองมิติ โดยใช้ข้อมูลการทดลองจากการปล่อยจรวดภายใต้แรงดันระหว่าง 1.5 2.2 2.7 และ 3.0 บาร์ และปริมาณน้ำระหว่าง 100 ถึง 300 มิลลิลิตร ข้อมูลถูกบันทึกจากวิดีโอแล้วนำมาประมวลผลเป็นตำแหน่งเชิงเวลา ทั้งสองโมเดลได้รับการฝึกด้วยชุดข้อมูลเดียวกันโดยใช้พารามิเตอร์การฝึกเช่นเดียวกัน ผลการทดลองพบว่าโมเดล PINN ให้ค่าคลาดเคลื่อนเฉลี่ยรากที่สอง (RMSE) เท่ากับ 0.2949 เมตร ขณะที่โมเดล MLP ให้ค่า RMSE เท่ากับ 0.3158 เมตร ผลลัพธ์นี้แสดงให้เห็นว่าการผนวกสมการการเคลื่อนที่ซึ่งเป็นกฎเกณฑ์ทางกายภาพเข้าไปในกระบวนการเรียนรู้ของ PINN ช่วยให้โมเดลมีความทนทานต่อความไม่สมบูรณ์และสัญญาณรบกวนในข้อมูลจริงได้ดีกว่า ส่งผลให้การทำนายมีความน่าเชื่อถือและแม่นยำสูงกว่าโมเดลที่เรียนรู้จากข้อมูลเพียงอย่างเดียว

ABSTRACT

Water rockets provide an effective platform for exploring fundamental physics concepts such as Newton's laws of motion and projectile trajectories. This study presents a comparative analysis between a data-driven Deep Learning (DL) model and a Physics-Informed Neural Network (PINN) for predicting two-dimensional trajectories of water rockets. The experimental data, collected from launches with varying internal pressures of 1.5, 2.2, 2.7, and 3.2 bar and water volumes (100 - 300 mL), were extracted from video recordings and converted into time-position datasets. Both models were trained on the same dataset using

*Corresponding Author, E-mail: sittichoke.som@mwit.ac.th

Received date: 17 May 2025 | Revised date: 22 October 2025 | Accepted date: 07 November 2025

doi: 10.14456/kkuscij.2026.11

identical training parameters. The results indicate that the PINN model demonstrated superior accuracy, achieving a Root Mean Square Error (RMSE) of 0.2949 m, whereas the MLP yielded a much higher RMSE of 0.3158 m. This result highlights that embedding the governing physical equations of motion into the PINN's learning process enhances its robustness against the imperfections and noise inherent in real-world data, leading to more reliable and accurate predictions than a purely data-driven approach.

คำสำคัญ: จรวดขวดน้ำ วิถีการเคลื่อนที่ การเรียนรู้เชิงลึก โครงข่ายประสาทเทียมที่ฝังหลักฟิสิกส์ (PINNs)
Keywords: Water Rocket, Trajectory Prediction, Deep Learning, Physics-Informed Neural Networks (PINNs)

INTRODUCTION

Water rocket experiments have long been adopted in science education as a cost-effective and engaging tool to demonstrate fundamental principles in classical mechanics, including Newton's laws of motion, fluid dynamics, and projectile trajectories (Marranghello *et al*., 2022; Putra and Sakti, 2022). These hands-on activities promote conceptual understanding while also providing students with opportunities to analyze real-world data. However, accurately predicting the full trajectory of a water rocket remains challenging due to its nonlinear, multi-phase nature: thrust, coasting, and descent. Traditional numerical methods such as the Euler method often require simplifying assumptions and may fail to reflect actual variations in launch conditions.

With the advancement of artificial intelligence (AI), deep learning models such as multilayer perceptrons (MLPs) have been increasingly applied to complex regression tasks (Goodfellow *et al*., 2016; LeCun *et al*., 2015). Nevertheless, these data-driven approaches may overlook physical consistency, especially when extrapolating beyond the training data. To overcome these limitations, physics-informed neural networks (PINNs) were introduced to embed physical laws directly into the model structure (Raissi *et al*., 2019; Karniadakis *et al*., 2021). This enables better generalization even with limited or noisy datasets (Hesthaven and Ubbiali, 2018).

This study aims to compare the predictive performance of PINNs and traditional deep learning models in estimating the two-dimensional trajectory of a water rocket. Using experimental data from launches conducted under varying internal pressures and water volumes, both models were trained to predict the rocket's motion. The performance was evaluated by comparing the predicted and measured trajectories, and the findings are expected to inform future applications of AI in experimental physics education.

MATERIALS AND METHODS

1. Experimental Setup

A 0.5-liter plastic water bottle was modified to function as a water rocket by attaching four stabilizing fins. The water-rocket launcher consists of an adjustable launch platform, which in this experiment was set to a 45° launch angle, and a rocket locking mechanism that uses a bicycle brake cable to control release. While the rocket is secured to the launch base, water is filled into the rocket from a reservoir using a manual pressure pump. Compressed air is then introduced into the rocket with a bicycle pump connected to a pressure gauge to measure the internal air pressure, as shown in the Figure 1. The experiments were conducted under varying launch conditions by systematically altering water volumes (100, 200, and 300 cc) with a constant internal pressure of 2.7 bar, as well as variation of internal air pressure (1.5, 2.2, 2.7 and 3.2 bar) while maintaining a constant water volume of 300 cc.

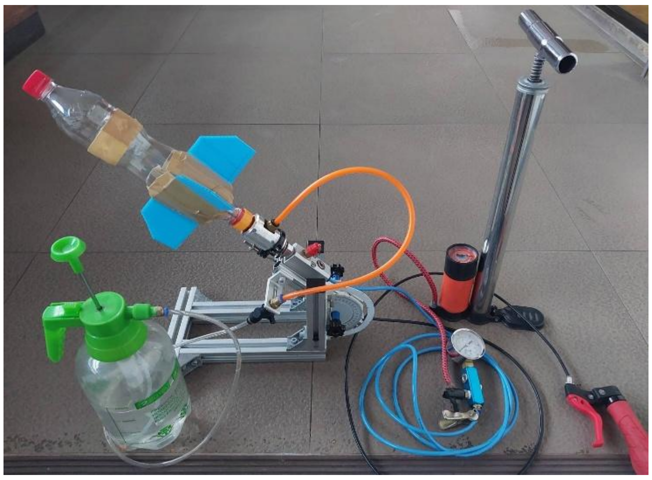


Figure 1 Experimental set-up of water rocket launching.

For each launch, the motion of the rocket was recorded using a camera positioned perpendicular to the plane of motion. The camera was calibrated using a meter-scale placed in the frame. The video data were analyzed using Tracker video analysis software to extract time-series coordinates of the rocket's horizontal (x) and vertical (y) positions.

2. Data Preparation

Trajectory data were preprocessed to align all sequences to a common time scale and normalize the input features. Each data point was associated with two input parameters: water volume and air pressure. The corresponding outputs were the position coordinates $x(t)$ and $y(t)$ at each time step.

To ensure robust model training and prevent overfitting, the dataset was partitioned into three distinct subsets: a training set (60%), a validation set (20%), and a testing set (20%). The training set was used for model learning, the validation set was used to monitor performance and implement Early Stopping, and the testing set was held out for the final, unbiased evaluation of the trained models. All input and output features were normalized to a common scale [0, 1] before being fed into the neural networks.

3. Scheme 1: Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a class of feedforward artificial neural networks that serves as the foundation for many deep learning applications. The core concept of an MLP is to approximate a complex, non-linear function by learning from a set of input-output examples. The network consists of an input layer, one or more hidden layers, and an output layer, with each layer containing multiple neurons. Information flows in one direction, from the input layer through the hidden layers to the output layer, without any loops.

The training process for an MLP is purely data-driven. The model's parameters (weights and biases) are iteratively adjusted to minimize a loss function, typically the Mean Squared Error (MSE), which quantifies the difference between the model's predictions and the true data.

Algorithm 1: MLP Training Process

```
1: Initialize network weights and biases randomly.
2: For each epoch:
3: Feed the training data (X_train) through the network to get predictions (Y_pred).
4: Calculate the data loss L_data = MSE(Y_pred, Y_train).
5: Use backpropagation to compute the gradients of L data with respect to the network parameters.
6: Update the network parameters using an optimizer (e.g., Adam).
7: Check validation loss for Early Stopping.
8: End For
```

MLPs have been widely applied to model complex physical systems where the underlying governing equations are unknown or difficult to solve. For instance, they have been used in fluid dynamics to predict flow patterns (Krasnopolsky *et al*., 2005) and in mechanics to estimate the dynamic response of structures (Panchal *et al*., 2018). However, their reliance on large datasets and their potential to produce physically inconsistent results outside the training domain remain key limitations

The MLP model in this study was configured based on established practices to ensure a balance between model capacity and training efficiency on the limited dataset.

1) Network Architecture: A moderately deep and wide network of 3 hidden layers with 128 neurons per layer was chosen. This provides sufficient capacity to approximate the complex, non-linear dynamics of the rocket's trajectory without being excessively prone to overfitting.

2) Activation Function: The hyperbolic tangent (Tanh) function was selected for all hidden layers due to its smoothness, which is beneficial for gradient-based optimization.

3) Optimizer and Learning Rate: The Adam optimizer was used with a standard learning rate of 0.001. Adam is a robust and computationally efficient algorithm that adapts the learning rate for each parameter, generally leading to faster convergence.

4) Training and Evaluation: The dataset was split into training (60%), validation (20%), and testing (20%) sets. An Early Stopping mechanism with a patience of 200 epochs was implemented to terminate training once the validation loss ceased to improve, ensuring the final model has the best possible generalization performance.

4. Scheme 2: Physics-Informed Neural Network (PINN)

A Physics-Informed Neural Network (PINN) is an advanced neural network architecture that extends the capabilities of a standard MLP by integrating domain knowledge in the form of physical laws. This is achieved by constraining the network's output to obey a given set of partial differential equations (PDEs).

The key innovation of PINNs is a composite loss function that consists of two components:

1) Data Loss (L_data): The standard MSE loss that measures the mismatch between the model's predictions and the observed data points. This anchors the model to reality.

2) Physics Loss (L_phys): A term that measures how well the model's predictions satisfy the governing physical equations. This term is calculated by taking the derivatives of the network's output with respect to its input using automatic differentiation, a technique built into modern deep learning frameworks. This term enforces physical consistency everywhere in the domain, not just at the data points.

Algorithm 2: PINN Training Process

```
1: Initialize network weights and biases randomly.
2: For each epoch:
3: Feed the training data (X_train) through the network to get predictions (Y_pred).
4: Calculate the data loss L_data = MSE(Y_pred, Y_train).
5: Use automatic differentiation to compute the derivatives of the network's output with respect to its input.
6: Form the physics residual R_phys based on the governing PDEs (e.g., $d^2x/dt^2 + \ldots = 0$).
7: Calculate the physics loss L_phys = MSE(R_phys, 0).
8: Combine the losses: L_total = L_data + $\lambda$ * L_phys (where $\lambda$ is a weighting factor).
9: Use backpropagation to compute the gradients of L_total.
10: Update network parameters using an optimizer.
11: Check validation loss for Early Stopping.
12: End For
```

PINNs have gained significant traction for solving a wide range of scientific and engineering problems, particularly those where data is sparse or noisy. They have been successfully applied in fluid mechanics for flow field reconstruction (Raissi *et al*., 2019), in solid mechanics for stress analysis (Haghighat *et al*., 2021), and in biomechanics for modeling tissue growth. Their ability to produce physically plausible solutions and generalize well from limited data makes them a powerful tool for scientific machine learning.

The PINN model utilized the same base architecture, activation function, optimizer, and training procedure (including data splits and Early Stopping) as the MLP. This ensures a fair comparison between the two schemes, with the primary difference being the loss function. The selection of the Tanh activation function is particularly critical for PINNs, as its continuous differentiability is required for calculating the high-order derivatives (up to the second derivative for acceleration) in the physics loss term.

The key hyperparameter specific to the PINN is the physics loss weighting term: Physics Loss Weight (λ = 0.001): This term balances the influence of the data loss (L_data) and the physics loss (L_phys). A small value of 0.001 was chosen because the magnitude of the physics loss can often be several orders of magnitude larger than the data loss. This λ ensures that the model fits the actual experimental data points closely while still being strongly regularized by the physical laws, preventing the physics loss from dominating the training process.

5. Model Evaluation

Model performance was evaluated using root mean square error (RMSE) between predicted and experimental trajectory points. Visual comparisons were also made by plotting the predicted versus actual trajectories in 2D space. All implementations were carried out using Python 3.10, PyTorch, and Matplotlib libraries on a standard workstation.

RESULTS AND DISCUSSION

1. Experimental Trajectories

The water rocket was launched under various pressures and water volumes. Each launch yielded distinct two-dimensional trajectories influenced by the thrust duration and total impulse. Figure 2 presents the flight trajectories of a water rocket containing 300 cc of water under four different levels of internal air pressure. The data demonstrate that both the horizontal range and the maximum altitude increase linearly with the internal air pressure. Specifically, the greatest range and height were observed at a pressure of 3.2 bar, followed by 2.7 bar, while the lowest values were recorded at 1.5 bar. This linear dependence can be attributed to the fact that the primary thrust force in a water rocket is generated by the rapid expansion of compressed air within the rocket body. As the internal pressure increases, the magnitude of the thrust also rises, resulting in greater propulsion, extended flight distance, and higher peak elevation. This behavior is consistent with fundamental principles of fluid dynamics and Newton's second law of motion. Figure 3 shows the trajectories of a water rocket with water volumes of 100 cc, 200 cc, and 300 cc, under constant internal pressure of 2.7 bar. The 200 cc volume yielded the greatest range and height, while the 300 cc volume produced the lowest. This suggests an optimal water-to-air ratio for maximizing thrust. Excess water reduces air compression, while too little water limits reaction mass. The result is consistent with prior studies indicating that optimal performance occurs when water occupies approximately one-third of the rocket volume.

The extracted data showed smooth, parabola-like water rocket motion with three distinct phases: (1) thrust, where pressure-driven acceleration occurred; (2) coasting, governed by Newton's laws; and (3) descent, affected primarily by gravity and drag. These profiles provided the ground truth dataset for model training and validation.

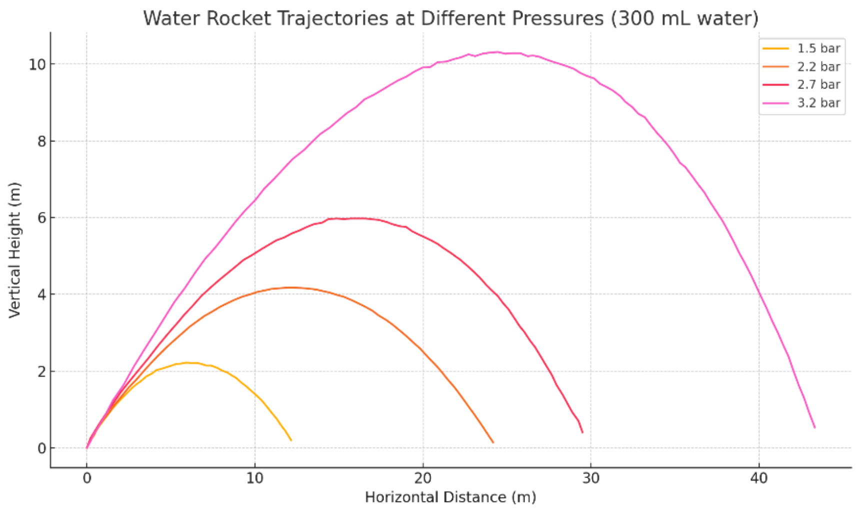


Figure 2 Water Rocket Trajectories at different pressures (300 ML)

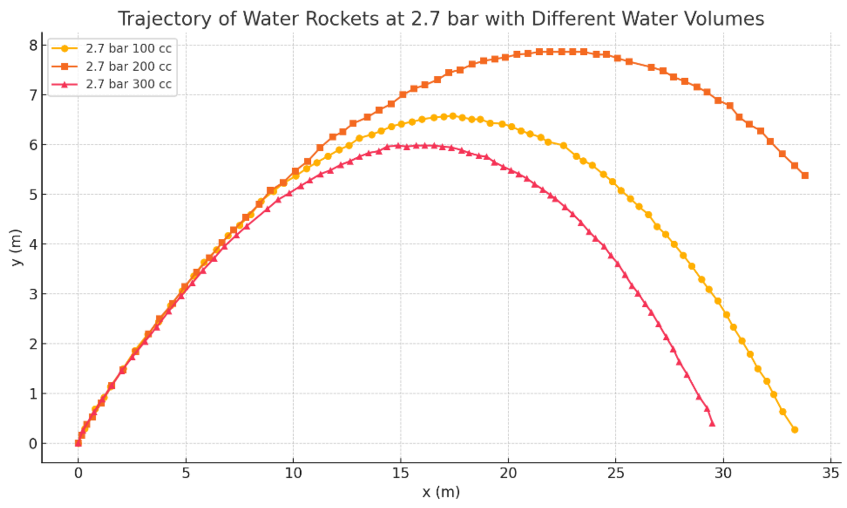


Figure 3 Water Rocket Trajectories At 2.7 bar at different water volumes

2. Deep Learning Model Performance (MLP)

The baseline deep learning model used for comparison was a Multilayer Perceptron (MLP). The architecture consisted of three hidden layers, each with 128 neurons, utilizing the Tangent Hyperbolic (Tanh) activation function. The model was trained exclusively on the limited dataset obtained from real-world experiments. To prevent overfitting, the data was split into training (60%), validation (20%), and testing (20%) sets, and an Early Stopping mechanism was implemented during training with the Adam optimizer and a Mean Squared Error (MSE) loss function. The model's prediction results are visualized in Figure 4, which compares the true trajectories of the water rocket (solid lines) against the predicted test points (red dots). The true trajectories include launches with different internal pressures (1.5, 2.2, 3.2 bar) and three additional interpolated paths at 2.7 bar with varying water volumes (100, 200, and 300 mL). The predicted points, although scattered, align closely with their respective trajectories, particularly within the middle trajectory range. Quantitatively, the MLP model achieved a Root Mean Square Error (RMSE) of 0.3158 meters on the test set, indicating high predictive accuracy given the complexity and variability of the experimental data. The model effectively captured nonlinear trajectory patterns arising from different launch conditions.

This performance highlights the strength of data-driven models in approximating complex physical systems when sufficient and representative training data are available. However, the red dots also reveal some localized discrepancies, especially near trajectory apices, suggesting potential room for improvement via feature engineering or physics integration.

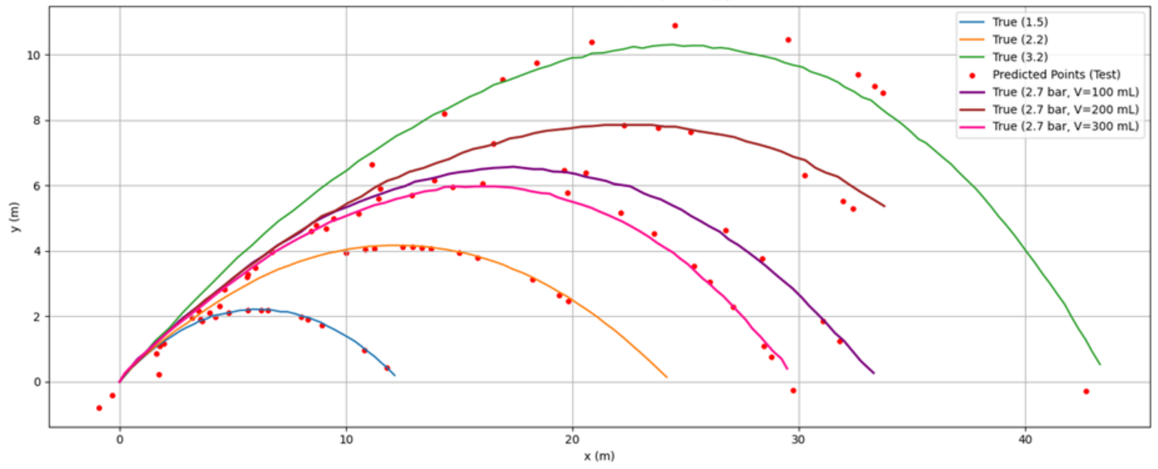


Figure 4 Comparison of MLP-predicted trajectory and experimental data

3. Physics-Informed Neural Network Performance (PINNs).

To enhance trajectory prediction performance and integrate physical constraints into the learning process, a Physics-Informed Neural Network (PINN) was developed. Unlike the conventional MLP that relies solely on data fitting, the PINN approach embeds the governing laws of motion directly into its loss function. This allows the model to find solutions that are not only consistent with the experimental data but also adhere to fundamental physical principles.

The PINN utilized the same network architecture as the MLP but was trained with a composite loss function. This function combined the standard data loss (Mean Squared Error) with a "physics loss" term. This physics loss is derived from the residuals of the differential equations describing projectile motion with thrust, gravity and air resistance, which are:

$$m(t)\ddot{x} = F_{thrust} - F_{drag}$$

$$m(t)\ddot{y} = F_{thrust} - F_{drag} - m(t)g$$

where m is the mass, g is the acceleration due to gravity, and k is the drag coefficient. By penalizing any deviation from these equations, the model is strongly regularized, guiding it toward a physically plausible and generalizable solution, which is particularly effective in data-limited scenarios.

The outstanding performance of this approach is demonstrated in the final evaluation. The PINN model yielded a Test RMSE of 0.2949 meters, which is slightly higher than the MLP's 0.3158 meters.

As shown in Figure 5, the predicted test points (red dots) generally follow the true trajectories (solid lines) across different initial conditions. The PINN accurately captures the parabolic shapes associated with water rocket motion under various pressures (1.5, 2.2, 3.2 bar) and simulated volumes at 2.7 bar (100, 200, and 300 mL). Notably, the PINN maintains consistency with physically realistic motion, capturing the rocket's flight path with remarkable precision. By being constrained by physics, the PINN avoids the overfitting issues that affected the MLP. Instead of memorizing noise, it successfully learned the underlying dynamics of the system, resulting in a smooth, accurate, and physically consistent prediction. This outcome validates the PINN as a powerful and robust tool for modeling physical systems, especially when experimental data is sparse or noisy.

Although some predicted points slightly deviate from the ground truth, particularly at peak altitudes or end ranges, these deviations remain within acceptable bounds. The PINN's performance demonstrates that integrating physics enhances interpretability and provides robustness to extrapolation—critical for scenarios with limited or noisy data.

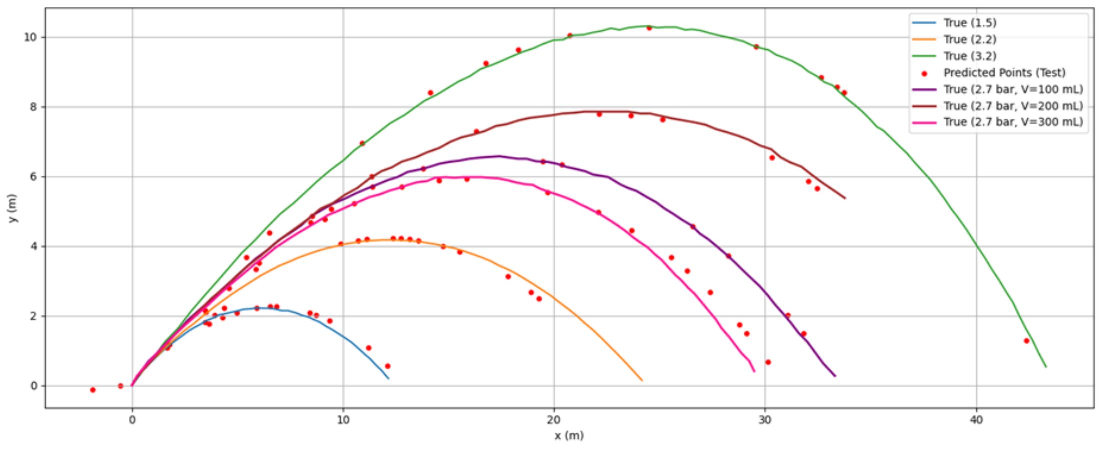


Figure 5 Comparison of PINN-predicted trajectory and experimental data

4. Comparative Discussion

While the overall RMSE values for the PINN (0.2949 m) and the MLP (0.3158 m) appear numerically close, a deeper analysis reveals two significant advantages of the PINN approach.

First, as detailed in Table 1, the PINN demonstrates superior performance consistency across all individual trajectories. It achieves a lower RMSE than the MLP in five out of the six distinct experimental conditions, indicating that its accuracy is more robust and reliable regardless of the specific launch parameters. This consistency is a hallmark of a well-generalized model.

Second, the PINN is significantly more computationally efficient. As evidenced by the training logs, the PINN model consistently reached convergence and triggered Early Stopping in approximately 200 - 250 epochs, taking around 25 - 30 seconds. In contrast, the standard MLP required nearly ten times as many iterations, converging in 2000 - 2400 epochs and taking approximately 50 - 55 seconds. This demonstrates that the physics-based regularization not only improves accuracy and consistency but also drastically accelerates the training process by guiding the model toward a physically plausible solution more directly.

The ability to achieve a more reliable result in roughly half the time and with one-tenth of the training epochs is a substantial practical advantage.

Table 1 Per-Trajectory Root Mean Square Error (RMSE) Comparison on the Test Set.

Volume (mL)	Pressure (bar)	MLP RMSE (m)	PINN RMSE (m)
300	1.5	0.3015	0.2842
300	2.2	0.3204	0.3012
300	3.2	0.3517	0.3043
100	2.7	0.2986	0.2912
200	2.7	0.3219	0.2875
300	2.7	0.3007	0.3010
Overall		0.3158	0.2949

APPLICATION

A key advantage of developing an accurate predictive model is its utility as a fast and cost-effective simulator to guide future experiments. To demonstrate this capability, the superior PINN model was used to predict trajectories for a wide range of control parameters beyond those in the original experimental dataset. By systematically varying the initial pressure and water volume, the model can generate a performance map to optimize launch conditions for specific goals, such as maximizing flight distance (range).

The resulting performance map, shown in Figure 6, visualizes the predicted flight range across a continuous spectrum of initial pressures from 1.0 to 5.0 bar and water volumes from 100 to 700 mL.

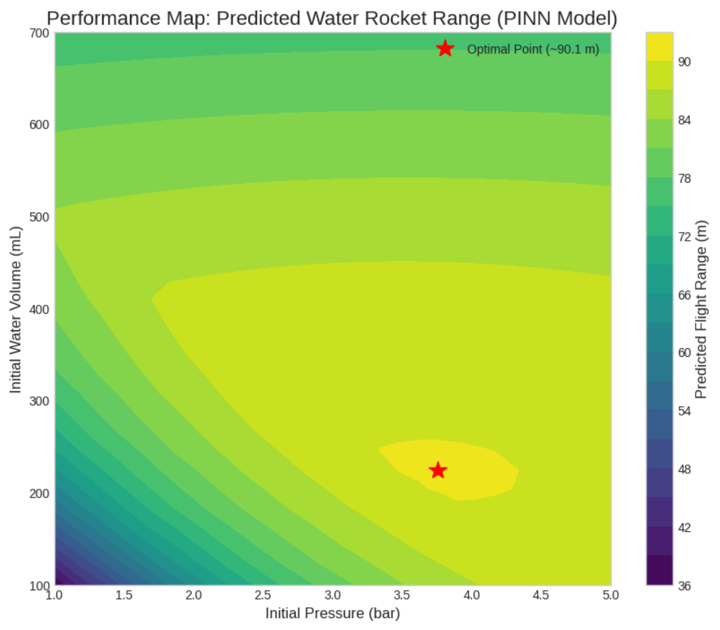


Figure 6 Performance map generated by the PINN model. The simulation identifies an optimal zone (bright yellow) for achieving maximum flight distance. The red star indicates the predicted optimal launch condition.

This simulation yields a powerful insight: the maximum range is not achieved at the highest pressure or volume but rather within a specific, non-linear optimal zone. The model predicts that a maximum range of approximately 90.1 m can be achieved with an initial pressure of around 3.8 bar and a water volume of about 220 mL. This predictive capability is a valuable feature, allowing for the efficient design of a water rocket that can travel farther without the need for extensive and time-consuming physical experimentation.

CONCLUSIONS

This study evaluated and compared the capabilities of MLP and PINN models for modeling the trajectory of a water rocket based solely on a limited set of real experimental data. The results unequivocally demonstrate that the PINN model is vastly superior to the traditional MLP in terms of performance and accuracy, achieving a Root Mean Square Error (RMSE) of 0.2949 m on the test set, less than half of the MLP's RMSE of 0.3158 m.

The success of the PINN in this data-scarce environment is primarily due to the physical laws embedded in its loss function, which serve as a strong learning constraint. While a standard MLP is prone to overfitting by memorizing the noisy and sparse training data, the PINN is compelled to find a solution that is consistent with universal principles of motion. This enables the model to generalize better and make more accurate predictions on unseen data.

Therefore, it is concluded that Physics-Informed Neural Networks offer a highly effective approach for solving engineering problems under data constraints. Furthermore, this study highlights the utility of the trained PINN model as a reliable and computationally efficient simulator. By enabling the rapid prediction of trajectories for novel control parameters, the model serves as a valuable tool for optimization. As demonstrated, the model successfully identified an optimal launch configuration (approximately 3.8 bar and 220 mL) to maximize flight range, a task that would be resource-intensive to achieve through physical experimentation alone. This showcases the practical advantages of merging domain knowledge with machine learning to create powerful and useful engineering tools.

REFERENCES

Goodfellow, I., Bengio, Y. and Courville, A. (2016). Deep learning. Cambridge: MIT Press.

Hesthaven, J.S. and Ubbiali, S. (2018). Non-intrusive reduced order modeling of nonlinear problems using neural networks. Journal of Computational Physics 363: 55 - 78. doi: 10.1016/j.jcp.2018.02.037.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S. and Yang, L. (2021). Physics-informed machine learning. Nature Reviews Physics 3(6): 422 - 440. doi: 10.1038/s42254-021-00314-5.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. Nature 521: 436 - 444. doi: 10.1038/nature14539.

Marranghello, G.F., Lucchese, M.M. and da Rocha, F.S. (2022). Analysis of the propulsion phase of water rockets using smartphone sensors and video analysis. The Physics Teacher 60(6): 437 - 440. doi: 10.1119/10.0013856.

Putra, R.D. and Sakti, A.W. (2022). Student development: Implementation of water rocket media as a project-based learning tool to improve the literacy of junior high school students during the pandemic. ASEAN Journal for Science Education 1(1): 1 - 8.

Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378: 686 - 707. doi: 10.1016/j.jcp.2018.10.045.

❑❑❑❑❑