Solving the Two and Three Body Problems with Deep Learning Models: Machine Learning in Physics Course Exam

Julian Roddeman (s1080491)
Radboud University
(Dated: July 27, 2024)

This study addresses the two-body and three-body problems using deep learning models, with a focus on Physics-Informed Neural Networks (PINNs). This papers covers data generation for multiple orbital scenarios, implement and compare different neural network architectures (MLP, LSTM, and PINN), and evaluate their performance in predicting trajectories. The approach demonstrates the potential of machine learning techniques in predicting orbits and velocities, particularly highlighting the advantages of physics-informed methods. The codebase for this project is available on GitHub at `https://github.com/JulianRodd/Orbit_Prediction`. Appendix E provides details about the repository structure and the README file in the root directory constains instructions for running the code.

I. INTRODUCTION

The two-body and three-body problems are challenges in physics. This study explores the application of deep learning models to solve these problems. We begin by simulating two-body and three-body problems with varying initial conditions and increased acceleration and additional force scenarios. We then implement and compare different neural network architectures, including Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and Physics-Informed Neural Networks (PINNs). Our evaluation focuses on model performance in predicting orbital trajectories for different time horizons, specifically 10, 100, and 500 steps. We analyze the advantages of incorporating physical constraints in neural networks through PINNs, aiming to improve prediction accuracy and physical consistency. By comparing the performance of these models across the various scenarios, we aim to provide insights into the strengths and limitations of different deep learning approaches in solving gravitational problems.

II. LITERATURE REVIEW

Neural networks have shown promise in orbital mechanics, with Liao et al. [1] successfully applying them to find and classify periodic orbits in the three-body problem. The introduction of Physics-Informed Neural Networks (PINNs) by Raissi et al. [2] has further advanced the field, offering a framework to incorporate physical laws directly into neural network architectures. In the context of gravitational problems, Martin and Schaub [3] demonstrated PINNs' efficacy in modeling gravity fields of celestial bodies.

Deep learning approaches have also been applied to related challenges in astrodynamics. Cheng et al. [4] utilized neural networks for real-time optimal control in asteroid landings, while Gao and Liao [5] proposed an efficient gravity field modeling method based on Gaussian process regression. These advancements highlight the potential of machine learning in addressing complex gravitational dynamics.

However, challenges persist in ensuring physical constraints are properly enforced, avoiding overfitting, and generalizing to new scenarios. The physics-informed approach partially addresses these issues, but further research is needed to optimize network architectures, incorporate more knowledge, and extend these methods to more complex N-body scenarios [6]. This literature review sets the stage for our study, which aims to contribute to this field by exploring the application of different neural networks architectures to two-body and three-body gravitational problems.

III. METHODOLOGY

A. Data Generation

The data generation process was designed to create diverse orbital scenarios for two-body and three-body gravitational problems, while including measurement uncertainties to reflect real-world challenges. We implemented three main simulation scenarios: the standard two-body problem, a two-body problem with increased acceleration and an additional force, and a three-body problem. For the two-body problem, we simulated the motion of a light rotating object (spaceship) around a very heavy stationary object. The initial radius was set to 10,000 with the initial velocity calculated to achieve a stable circular orbit. In the increased acceleration + force scenario, we introduced an additional sinusoidal force on the spaceship, simulating a spaceship system malfunction. The three-body problem involved simulating the motion of a light object (spaceship) under the gravitational influence of two heavy bodies. We generated multiple initial conditions by randomizing the initial position by ± 5%. All simulations were based on Newton's law of gravitation:

$$F = -G\frac{Mm}{r^2}\hat{r}$$

where G is the gravitational constant, M and m are the masses of the objects, r is the distance between them, and $\hat{r}$ is the unit vector in the direction of r. We used the Euler method for numerical integration to update the position and velocity of the objects:

$$v(t + \Delta t) = v(t) + a(t)\Delta t$$

$$r(t + \Delta t) = r(t) + v(t)\Delta t$$

To reflect real-world measurement challenges, we introduced a 1% relative uncertainty to the angle, radius, and velocity measurements of the orbiting object. For each scenario, we simulated a total of 10 trajectories (spaceships) with different initial conditions, each for 10,000 time steps. We implemented some checks to ensure the generated orbits are valid and realistic. For the two-body problem, we verified that the trajectory forms at least one complete orbit, spans a significant range in both x and y directions, and doesn't simply fly away directly. The data was split into training, validation, and test sets. Nine spaceships were used for training and validation, with their trajectories divided into sequences and split using 5-fold cross-validation. One spaceship was reserved for the test set, providing a truly unseen trajectory for final model evaluation.

B. Model Architectures

To address the challenge of predicting orbital dynamics, we implemented and compared three neural network architectures: a simple regression Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and Physics-Informed Neural Network (PINN). All of these where constructed with Pytorch. Table I provides an overview of the key characteristics of each model.

TABLE I: Overview of Model Architectures

Feature	**MLP**	**LSTM**	**PINN**
Input Size	12	12	12
Hidden Layers	3	2	4
Hidden Size	256	128	256
Output Size	4	4	4
Activation	LeakyReLU	ReLU	Tanh
Dropout Rate	0.2	0.2	0.1
Normalization	Batch	Layer	Batch
Residual Conn.	Yes	Yes	Yes

The MLP serves as our baseline model. The LSTM architecture utalizes two bidirectional layers, this setting makes it learn additional information from both direction. The PINN architecture incorporates physical constraints into the learning process. It uses a custom loss function that combines mean squared error with physics-based constraints:

$$L_{\text{total}} = L_{\text{MSE}} + 0.1 \cdot (L_{\text{energy}} + L_{\text{momentum}}) \quad (1)$$

In this equation, L_{MSE} represents the standard mean squared error between predicted and true values. L_{energy} enforces energy conservation and is defined as:

$$L_{\text{energy}} = \frac{1}{N}\sum_{i=1}^{N}(E_{\text{pred},i} - E_{\text{true},i})^2 \quad (2)$$

where $E = \frac{1}{2}v^2 - \frac{GM_1}{r}$. Here, v is velocity, G is the gravitational constant, M_1 is the mass of the central body, and r is the distance from the central body. The L_{momentum} term ensures momentum conservation:

$$L_{\text{momentum}} = \frac{1}{N}\sum_{i=1}^{N}(p_{\text{pred},i} - p_{\text{true},i})^2 \quad (3)$$

with $p = mv$ representing momentum.

By incorporating these physics-based constraints, the PINN architecture aims to learn solutions that not only fit the data but also respect the underlying physical principles of orbital mechanics. This hopefully improves model predictions, particularly for complex scenarios like the three-body problem and the cases with increased acceleration.

All models use residual connections and dropout to facilitate gradient flow and prevent overfitting. They were trained using the Adam optimizer with a learning rate of 0.001.

All models were trained with a batch size of 32 for a maximum of 20 epochs, with early stopping implemented using a patience of 5 epochs. To ensure robust prediction and evaluation, we employed 5-fold cross-validation. Training was conducted for each model type (MLP, LSTM, PINN) on three dataset types: the two-body problem, the two-body problem with increased acceleration, and the three-body problem. The input data consisted of three time steps of position and velocity information, with the model tasked with predicting the position and velocity at the next time step. For training the input data was normalized using Min-Max scaling. For model validation we also use scaled Min-Max values.

C. Inference and Evaluation

Our evaluation process includes the assessment of the ensembled model performance. For each of the nine different model types (3 architectures × 3 scenarios), we trained five models using 5-fold cross-validation.The models were then evaluated on the unseen test set, with predictions made for position and velocity at 10, 100, and 500 steps into the future. The mean squared error (MSE) was calculated for each prediction horizon and aggregated across all models to provide estimates of performance. The standard deviation of MSE values was also computed to assess how to certain the models were in their predictions. To test if the models also perform well on the data they were trained on/tuned on, we evaluated them on nine train and validation spaceships too.

IV. RESULTS

A. Data Generation and Simulation Results

Figure 1 in appendix A presents an overview of the trajectories and velocities for all spaceships across the different scenarios (two-body, two-body with force and increased acceleration and three-body).

In the standard two-body problem (Figures 1a and 1b), we observe mostly circular orbits. There is a little amount of variation between trajectories, which aligns with our expectations given the introduced measurement. The velocity profiles show consistent periodic patterns.

The introduction of an additional force and some increased acceleration (Figures 1c and 1d) maintains the circular nature of the orbits but with some key differences. The trajectories show more variation in the y-axis, Interestingly not in the x-axis. Notably, the velocities show higher overall amplitudes while also having smaller wavelengths in comparison to the regular two-body scenario. This indicates that the additional force increases the speed of the spaceships without significantly.

The three-body problem (Figures 1e and 1f) introduces significant complexity and chaos. The trajectories show extreme variability, with some spaceships maintaining bounded orbits while others are catapulted into space entirely. This chaotic behavior is also in the velocities shown, which display a wide range of amplitudes and frequencies. We observe a mix of very high and very low amplitude oscillations, often with extremely short wavelengths. These extreme variations likely correspond to close encounters with the heavy bodies, resulting in rapid accelerations and decelerations.

B. Model Performance Comparison

Table II reveals some patterns in the performance of MLP, LSTM, and PINN models across various scenarios. In the standard two-body problem, PINN demonstrates best performance, consistently achieving the lowest position MSE with remarkably low standard deviations, particularly for short-term predictions. Interestingly, for velocity predictions in this scenario, all models perform comparably, with a slight edge to LSTM on the test set for 500 steps.

The two-body problem with acceleration presents a different picture, where MLP unexpectedly outperforms both LSTM and PINN, especially in position predictions.

The three-body problem shows the most differences in model performance. While PINN and LSTM show reasonable performance, MLP exhibits extreme instability for 100-step predictions, with position MSE values soaring to the order of 10^{27}. This instability highlights the challenge of predicting chaotic systems with simpler architectures. LSTM demonstrates robust performance in this complex scenario, particularly for longer prediction horizons, indicating its strength in capturing long-term dependencies in chaotic systems.

PINN, while performing well in short-term predictions for the three-body problem, shows increasing MSE for longer horizons.

For a visual representation of the models' performance on trajectories, refer to Appendices B, C, and D, which contain trajectory plots comparing predicted and actual paths for the first spaceship in each scenario.

V. DISCUSSION

Our investigation into applying deep learning models to gravitational problems has shown some insights into the strengths and limitations of different architectures for predicting orbital dynamics. In the two-body scenario, PINN demonstrated best consistency, particularly for short-term predictions, highlighting the advantage of incorporating physical constraints in simpler orbital dynamics. Surprisingly, for the two-body problem with increased acceleration, MLP outperformed both LSTM and PINN, achieving a mean squared error (MSE) of 4.21e+7 for position in 500-step predictions on the test set. This unexpected result suggests that MLPs are particularly effective for scenarios with additional forces beyond standard gravitation, and that their flexible architecture may be more suited for capturing these added complexities. The three-body problem revealed the most dramatic differences in model performance. PINN showed strength in short-term predictions, while LSTM excelled in longer horizons, achieving an MSE of 2.40e+10 for position and 0.13 for velocity in 500-step predictions on the test set. This highlights LSTM's capacity to model complex temporal dependencies in chaotic systems. However, all models showed degrading performance as the prediction horizon extended from 10 to 500 steps, especially in the three-body case. Our input sequence length of three time steps proved adequate for short-term predictions but insufficient for longer horizons, especially in the three-body problem. This indicates that increasing the input sequence length could enhance long-term predictions for all models, particularly for LSTM. Adding more LSTM layers might also improve performance. The models in this study were tasked with predicting both trajectory and velocity simultaneously. Splitting these tasks and training separate models for each could potentially improve performance by allowing models to specialize in predicting either trajectory or velocity. Future work should explore adaptive input lengths and more sophisticated physical constraints in PINN models. Additionally, investigating model performance on real astronomical data would provide valuable insights.

Problem	Model	Set	10 Steps		100 Steps		500 Steps	
			Pos M (SD)	Vel M (SD)	Pos M (SD)	Vel M (SD)	Pos M (SD)	Vel M (SD)
Two-Body	MLP	Tr	4.99e+7 (2.22e+6)	0.41 (2.63e-5)	5.26e+7 (3.43e+6)	0.41 (4.69e-4)	5.27e+7 (3.44e+6)	0.41 (4.88e-4)
		Va	4.98e+7 (2.24e+6)	0.42 (3.09e-5)	5.26e+7 (3.41e+6)	0.41 (4.70e-4)	5.27e+7 (3.44e+6)	0.41 (4.91e-4)
		Te	4.98e+7 (2.24e+6)	0.40 (3.06e-5)	5.26e+7 (3.41e+6)	0.40 (4.59e-4)	5.27e+7 (3.44e+6)	0.40 (4.79e-4)
	LSTM	Tr	7.47e+7 (2.25e+7)	0.41 (9.06e-4)	7.66e+7 (2.62e+7)	0.41 (2.04e-3)	7.73e+7 (2.63e+7)	0.41 (2.11e-3)
		Va	7.42e+7 (2.31e+7)	0.42 (8.86e-4)	7.49e+7 (2.85e+7)	0.41 (2.02e-3)	7.56e+7 (2.87e+7)	0.41 (2.13e-3)
		Te	7.41e+7 (2.32e+7)	0.40 (8.65e-4)	7.49e+7 (2.85e+7)	0.40 (1.97e-3)	7.56e+7 (2.87e+7)	**0.39** (2.07e-3)
	PINN	Tr	**4.88e+7** (**7.52e+5**)	0.41 (6.49e-5)	**4.85e+7** (2.46e+6)	0.41 (8.23e-4)	**4.82e+7** (2.88e+6)	0.41 (9.81e-4)
		Va	**4.92e+7** (**6.12e+5**)	0.42 (6.14e-5)	**4.85e+7** (2.46e+6)	0.42 (8.14e-4)	**4.82e+7** (2.87e+6)	0.41 (9.84e-4)
		Te	**4.88e+7** (**7.52e+5**)	0.40 (6.46e-5)	**4.85e+7** (2.46e+6)	0.40 (7.98e-4)	**4.82e+7** (2.87e+6)	0.40 (9.60e-4)
Two-Body Accel.	MLP	Tr	**4.40e+7** (2.40e+6)	0.41 (2.09e-4)	**4.47e+7** (8.41e+6)	0.41 (1.04e-3)	**4.21e+7** (7.91e+6)	0.41 (1.56e-3)
		Va	**4.38e+7** (2.40e+6)	**0.36** (1.95e-4)	**4.47e+7** (8.45e+6)	**0.36** (9.63e-4)	**4.21e+7** (7.91e+6)	**0.36** (1.45e-3)
		Te	**4.39e+7** (2.44e+6)	0.45 (2.16e-4)	**4.47e+7** (8.42e+6)	0.45 (1.07e-3)	**4.21e+7** (7.89e+6)	0.45 (1.63e-3)
	LSTM	Tr	5.31e+7 (9.20e+6)	0.41 (3.57e-4)	5.41e+7 (1.20e+7)	0.41 (1.12e-3)	5.42e+7 (1.19e+7)	0.41 (1.13e-3)
		Va	5.28e+7 (9.51e+6)	**0.36** (3.34e-4)	5.41e+7 (1.19e+7)	**0.36** (1.05e-3)	5.42e+7 (1.19e+7)	**0.36** (1.05e-3)
		Te	5.31e+7 (9.13e+6)	0.45 (3.65e-4)	5.40e+7 (1.20e+7)	0.45 (1.17e-3)	5.41e+7 (1.19e+7)	0.45 (1.17e-3)
	PINN	Tr	7.49e+7 (2.92e+7)	0.41 (1.39e-3)	7.11e+7 (3.11e+7)	0.41 (1.04e-3)	7.47e+7 (2.73e+7)	0.41 (9.67e-4)
		Va	7.43e+7 (2.97e+7)	**0.36** (1.29e-3)	7.11e+7 (3.11e+7)	**0.36** (9.67e-4)	7.47e+7 (2.73e+7)	**0.36** (8.97e-4)
		Te	7.52e+7 (2.88e+7)	0.45 (1.45e-3)	7.12e+7 (3.12e+7)	0.45 (1.07e-3)	7.49e+7 (2.73e+7)	0.45 (9.92e-4)
Three-Body	MLP	Tr	9.91e+8 (1.01e+9)	0.28 (3.94e-3)	**7.16e+27** (**1.43e+28**)	**8.72e+15** (**1.74e+16**)	9.55e+9 (1.60e+10)	0.28 (4.86e-3)
		Va	1.13e+9 (1.18e+9)	**0.12** (3.47e-3)	**7.43e+27** (**1.49e+28**)	**9.06e+15** (**1.81e+16**)	9.47e+9 (1.60e+10)	**0.12** (4.41e-3)
		Te	3.07e+9 (4.52e+9)	0.18 (2.68e-3)	**4.51e+27** (**9.02e+27**)	**5.49e+15** (**1.10e+16**)	1.08e+11 (1.65e+11)	0.17 (6.58e-3)
	LSTM	Tr	1.07e+10 (7.68e+9)	0.27 (9.74e-3)	2.94e+10 (1.97e+10)	0.26 (9.69e-3)	2.94e+10 (1.98e+10)	0.26 (9.71e-3)
		Va	1.47e+10 (1.16e+10)	**0.11** (5.90e-3)	2.51e+10 (1.71e+10)	**0.11** (6.37e-3)	2.51e+10 (1.71e+10)	**0.11** (6.38e-3)
		Te	**1.22e+10** (**1.06e+10**)	0.17 (5.69e-3)	**2.48e+10** (**1.33e+10**)	0.16 (7.13e-3)	**2.48e+10** (**1.34e+10**)	0.16 (7.11e-3)
	PINN	Tr	**2.53e+9** (**3.84e+9**)	0.28 (6.15e-3)	3.76e+10 (7.15e+10)	0.27 (8.13e-3)	3.62e+10 (6.84e+10)	0.27 (8.20e-3)
		Va	**4.51e+9** (**7.87e+9**)	**0.12** (5.13e-3)	3.76e+10 (7.13e+10)	**0.12** (6.23e-3)	3.62e+10 (6.84e+10)	**0.12** (6.07e-3)
		Te	**3.27e+9** (**5.10e+9**)	0.18 (6.64e-3)	3.74e+10 (7.12e+10)	0.17 (6.65e-3)	3.62e+10 (6.84e+10)	0.17 (7.18e-3)

TABLE II: Mean Squared Error (MSE) for different models across problem types, showing mean (M) and standard deviation (SD) (model uncertainty) over 5 folds for Position (Pos) and Velocity (Vel) at 10, 100, and 500 prediction steps on train (Tr), validation (Va), and test (Te) sets. Significant and interesting results are highlighted in bold.

VI. CONCLUSION

In conclusion, while our deep learning models show significant promise in solving gravitational problems, there remains substantial room for improvement, particularly in long-term predictions and that of chaotic systems.

[1] S. Liao, X. Li, and Y. Yang, Three-body problem—from newton to supercomputer plus machine learning, New Astronomy **96**, 101850 (2022).

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis, 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 (2019).

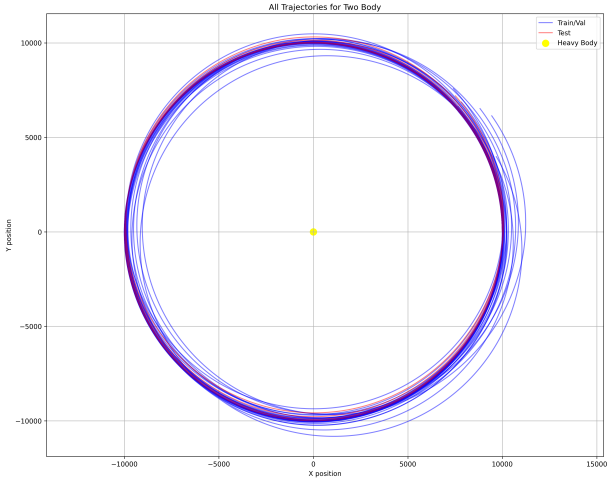
[3] J. Martin and H. Schaub, Physics-informed neural networks for gravity field modeling of the earth and moon, Celestial Mechanics and Dynamical Astronomy **134**, 13 (2022).

[4] L. Cheng, Z. Wang, Y. Song, and F. Jiang, Real-time optimal control for irregular asteroid landings using deep neural networks, Acta Astronautica **170**, 66 (2020).

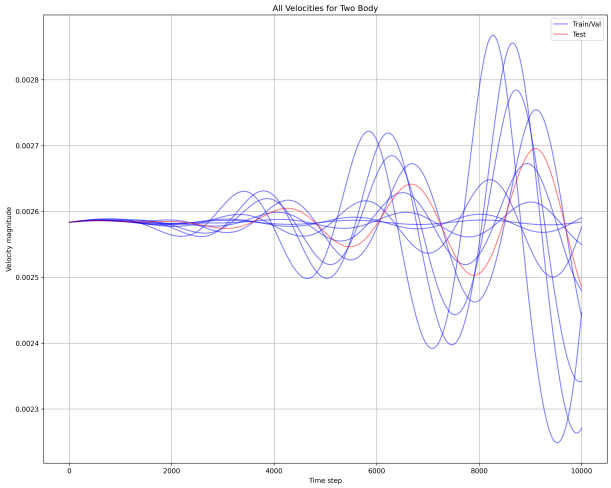
[5] A. Gao and W. Liao, Efficient gravity field modeling method for small bodies based on gaussian process regression, Acta Astronautica **157**, 73 (2019).

[6] C. Meng, S. Seo, D. Cao, S. Griesemer, and Y. Liu, When physics meets machine learning: A survey of physics-informed machine learning, arXiv preprint arXiv:2203.16797 (2022).

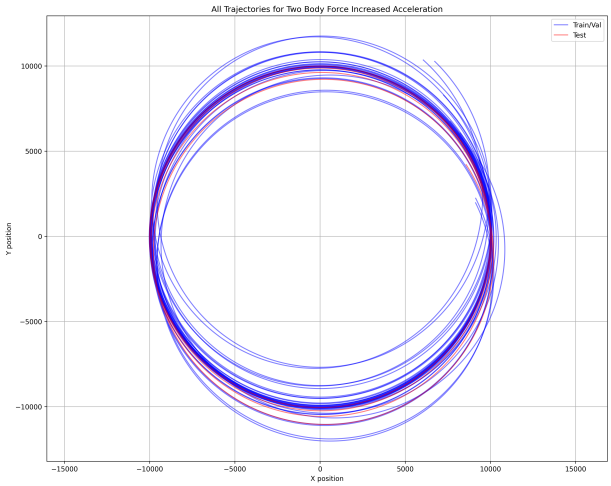
Appendix A: Generated Trajectories and Velocities for Different Orbital Scenarios



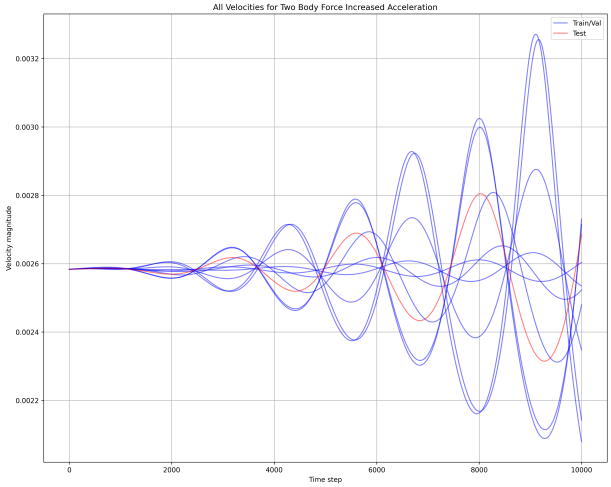
(a) Trajectories of all 10 spaceships in the two-body problem



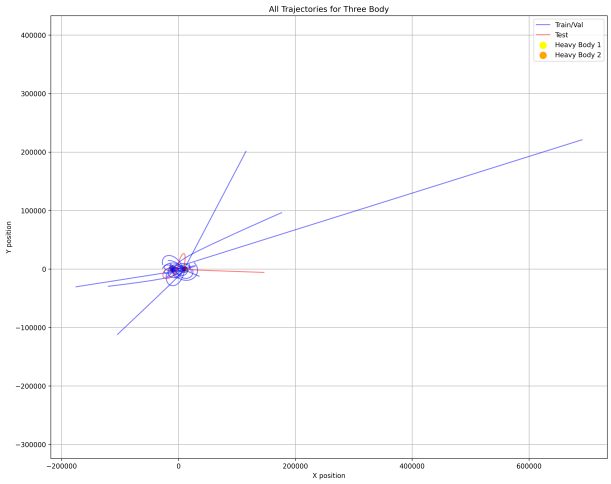
(b) Absolute velocity of all the 10 spaceships over time in the two-body problem



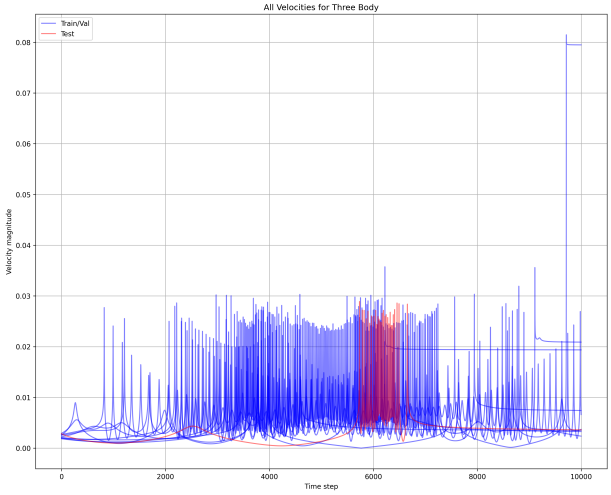
(c) Trajectories of all 10 spaceships in the two-body problem with increased acceleration



(d) Absolute velocity of all the 10 spaceships over time in the two-body problem with increased acceleration



(e) Trajectories of all 10 spaceships in the three-body problem

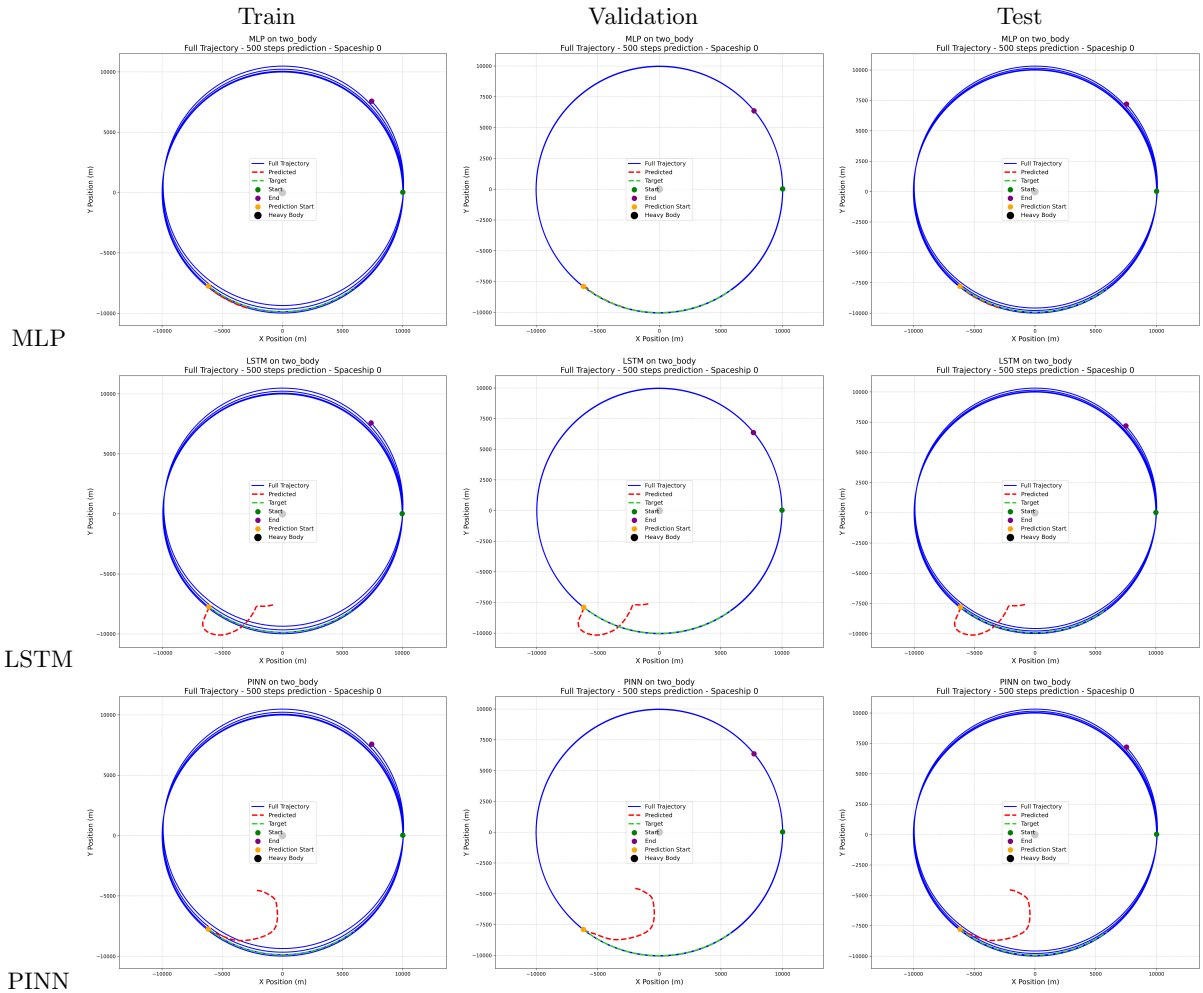


(f) Absolute velocity of all the 10 spaceships over time in the three-body problem

FIG. 1: Trajectories and velocities of spaceships in different orbital scenarios

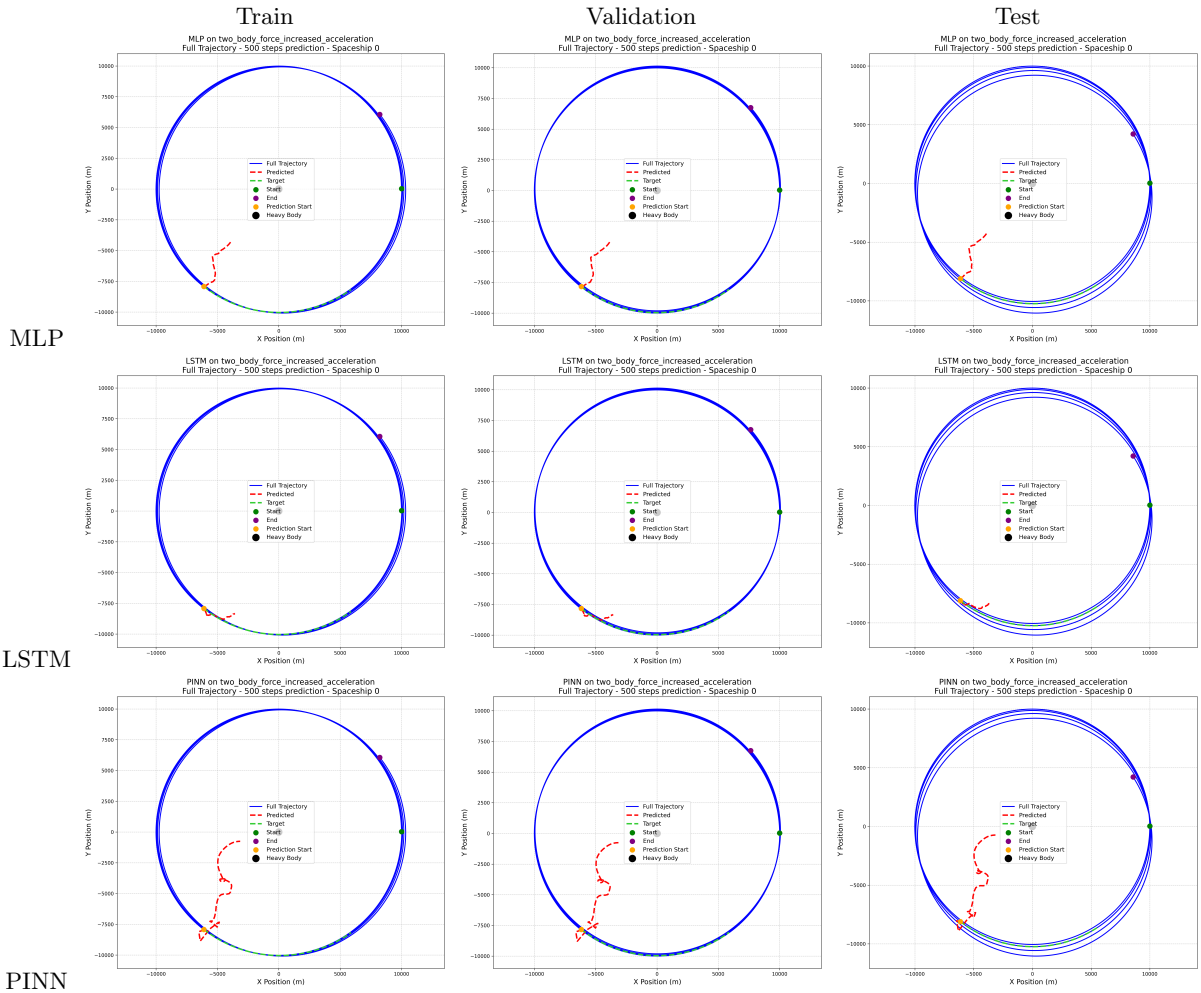
Appendix B: Two-Body Problem Trajectory Predictions

FIG. 2: Two-Body Problem: Model Predictions vs. Actual Trajectories for first spaceship



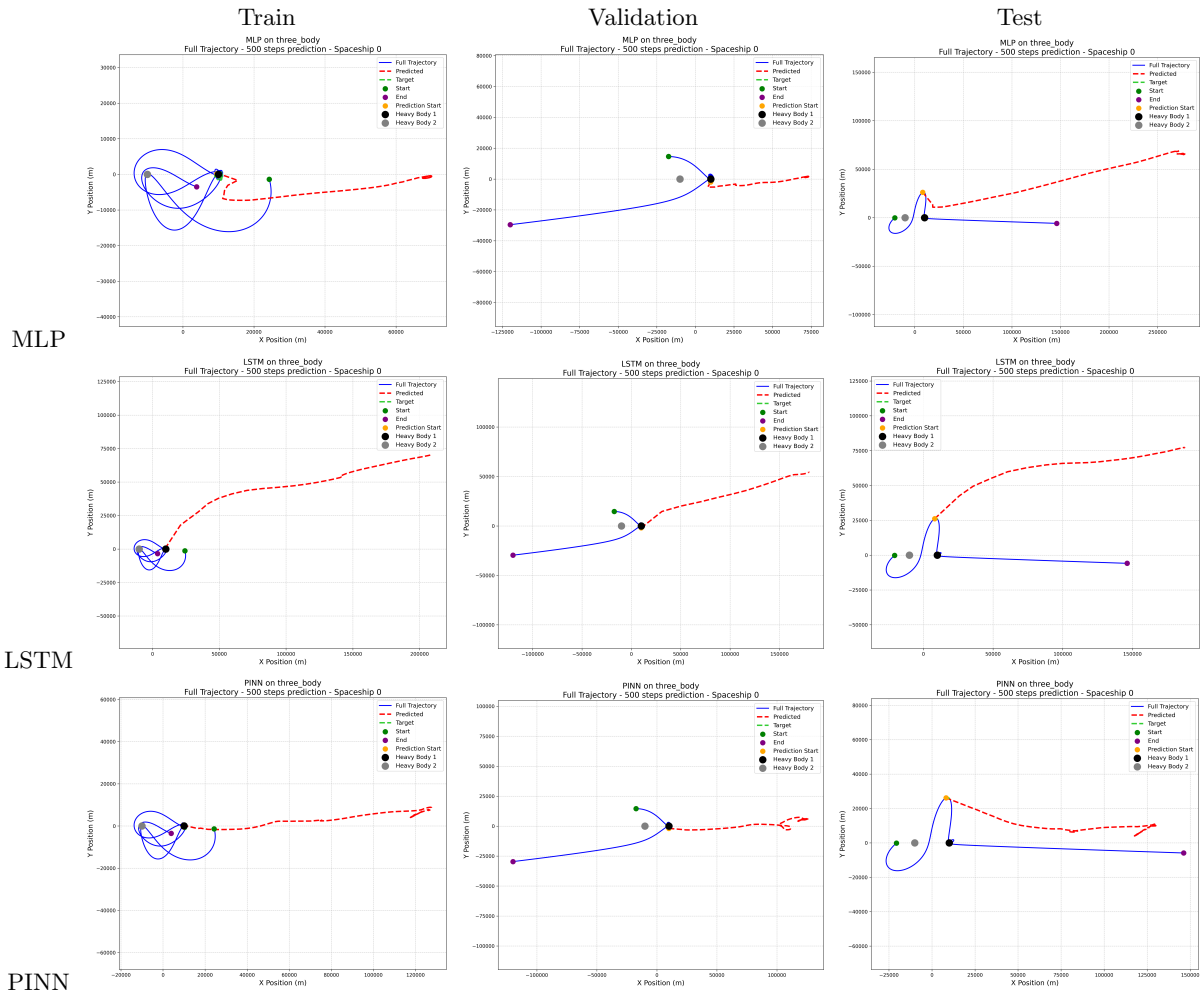
Appendix C: Two-Body Problem with Increased Acceleration Trajectory Predictions

FIG. 3: Two-Body Problem with Increased Acceleration: Model Predictions vs. Actual Trajectories for first spaceship



Appendix D: Three-Body Problem Trajectory Predictions

FIG. 4: Three-Body Problem: Model Predictions vs. Actual Trajectories for first spaceship



Appendix E: Repository

- **checkpoints/**: Contains saved model states for different architectures and datasets
  - Subfolders for LSTM, MLP, and PINN models
  - Further subdivided by dataset types (three_body, two_body, two_body_force_increased_acceleration)
  - Each subfolder contains model states (saved with Pytorch) for different folds (e.g., lstm_fold0.pth, lstm_fold1.pth, etc.)
- **code/**: Main source code directory
  - `data_generation/`: Scripts for generating simulation data
  - `inference/`: Code for running model inference and evaluation
  - `training/`: Implementation of different model architectures and training procedures
    - Separate subfolders for LSTM, MLP, and PINN models
    - Each subfolder contains architecture, constants, and training scripts
- **datasets/**: Contains the generated datasets for different scenarios
  - Subfolders for three_body, two_body, and two_body_force_increased_acceleration
  - Each scenario has train, validation, and test data splits
  - Training and validation data further divided into multiple folds for cross-validation
- **images/**: Visualizations of trajectories and velocities for different scenarios
  - Separate subfolders for each scenario (three_body, two_body, two_body_force_increased_acceleration)
  - Contains plots of individual trajectories, velocities, and combined
- **inference_results/**: Stores the output from model inference and evaluation
  - CSV files with aggregated results for each model and dataset combination
  - `consolidated_results.csv` summarizing all experiments, used in table II
  - Subfolders containing detailed predictions and visualizations for each model, dataset, and prediction step combination
  - Subfolder `sequence_results` contains the actual predictions for each sequence length / model / dataset combination