Modelling of Underwater Vehicles using Physics-Informed Neural Networks with Control

David Felsager
Department of Electrical and Computer Engineering
Aarhus University
Aarhus, Denmark
201904960@post.au.dk

Abdelhakim Amer
Department of Electrical and Computer Engineering
Aarhus University
Aarhus, Denmark
abdelhakim@ece.au.dk

Yury Brodskiy
EIVA a/s
Skanderborg, Denmark
ybr@eiva.com

Andriy Sarabakha
Department of Electrical and Computer Engineering
Aarhus University
Aarhus, Denmark
andriy@ece.au.dk

***Abstract*—Physics-informed neural networks (PINNs) integrate physical laws with data-driven models to improve generalization and sample efficiency. This work introduces the first open-source implementation of the Physics-Informed Neural Network with Control (PINC) framework, designed to model the dynamics of an underwater remotely operated vehicle. Using initial states, control actions, and time inputs, PINC extends PINNs to enable physically consistent transitions beyond the training domain. Various PINC configurations are tested, including differing loss functions, gradient-weighting schemes, and hyperparameters. Validation on a simulated underwater vehicle demonstrates more accurate long-horizon predictions compared to a non-physics-informed baseline.**

***Index Terms*—physics-informed machine learning, dynamics modelling, underwater robotics**

I. Introduction

Underwater robotic systems, such as autonomous underwater vehicles (AUVs) and remotely operated vehicles (ROVs), are critical for tasks such as seabed inspections, pipeline monitoring [1], and deep-sea exploration [2], where human access is limited. With advances in autonomy, underwater robots can tackle complex manipulation tasks, including pipeline repair and maintenance. Such applications require precise control and accurate dynamic models. For example, model predictive control (MPC) provides a framework for achieving complex tasks with high performance, such as inspection [3], while relying on an accurate motion model for prediction. However, modelling underwater robots using first principles is challenging due to non-linearities from friction, hydrodynamic disturbances, and unmodeled higher-order effects [4]. Furthermore, due to the wide variety of underwater vehicle designs, sizes, and degrees of freedom, modeling is often cumbersome and, in some cases, impractical. Data-driven methods on the other-hand offer a unifying framework to estimate the dynamics of these diverse systems. Physics-informed neural networks (PINNs) in particular is a promising approach that integrates data-driven neural networks with physical laws as regularization to produce physically plausible outputs [5].

This work explores the potential of a specialized variant of PINNs, namely the physics-informed neural network with control (PINC) [6], for modelling the dynamics of underwater ROVs. The primary objective is to evaluate whether PINC can effectively model a simplified underwater dynamic system and have the potential to provide an accurate model for control applications. The investigation applies these modelling approaches to an underwater ROV as the physical system of interest. While prior studies on PINC have demonstrated potential benefits in other domains, no existing work (as of this writing) appears to apply PINC and its implicit integration technique, combined with autoregressive predictions, to underwater robotics. This gap presents a unique opportunity to evaluate whether and how PINC might offer advantages – such as improved accuracy or generalization – over a traditional DNN when modelling complex underwater dynamics. To that end, the model's predictive capabilities and robustness are tested by incorporating input noise during training, among other methods. The main contributions of this article are summarized as follows:

- Developed a PINC framework to model simplified ROV dynamics, achieving high long-horizon prediction accuracy with minimal computational overhead.
- Demonstrated the effectiveness of combining physics-based regularization with training optimizations, including residual connections, gradient normalization, and adaptive learning rate scheduling.
- Performed a comprehensive analysis of the hyperparameters' effect on the PINC performance.
- Released the first open-source implementation of the PINC framework[1], including a simulation model of an underwater vehicle for synthetic data generation.

This research was partially supported by the Aarhus University Research Foundation, EIVA a/s and Innovation Fund Denmark under grant 2040-00032B.

[1] https://github.com/eivacom/pinc-xyz-yaw

This paper is organised as follows. Section II provides a comprehensive literature review on PINNs, including their applications and limitations. Section III motivates the importance of accurate and computationally efficient dynamics models for enhancing prediction and control. Section IV outlines the methodological framework for evaluating PINC and baseline models in simulation experiments. Section V presents the simulation results for different PINC configurations and baseline comparisons. Finally, Section VI concludes with a summary of key insights and suggestions for future research directions.

II. Related Works

Recent advances in machine learning for dynamical system identification increasingly leverage PINNs, which embed known physical laws into network training to promote physically consistent predictions. This review focuses on PINNs in the context of non-linear system identification and control, concluding with an in-depth look at PINC. Traditional data-driven methods range from flexible but opaque neural networks to interpretable but potentially limited approaches like Dynamic Mode Decomposition with control (DMDc). Sparse identification of non-linear dynamics with control (SINDyC) [7] constructs a large function library, then applies a sparsity-promoting algorithm to isolate key dynamics for model-based control. Gaussian processes (GPs) [8, 9] can quantify uncertainty but often scale poorly in high dimensions. PINNs bridge this gap by imposing differential equations within the neural network's loss, reducing data needs while preserving interpretability.

Many works extend PINNs for more complex tasks. For instance, [10] applies PINNs to Reinforcement Learning, yielding faster training and robust controllers beyond the training domain. [11] introduces a Robust Adaptive MPC framework leveraging PINNs to integrate physics-based priors with data-driven learning, improving trajectory tracking performance under uncertainties while mitigating the computational burden of traditional robust MPC approaches.

Hybrid architectures like *Deep Lagrangian networks* [12] embed energy-based formulations for better interpretability. Others focus on *Neural ODEs* [13, 14], augmenting nominal physics models with learned residuals. PINNs have also found success modelling AUV dynamics [15], soft robotics [16], and more. PINC extends PINNs to include control inputs, initial states, and time [6]. PINC can model a continuum of initial conditions by treating the network as a continuous time-stepper. Auto-regressive rollouts allow multi-step prediction, and architectures like domain-decoupled PINNs [17] can boost training efficiency. PINC has shown promise for efficient MPC integration and large-scale industrial tasks [18]. Gradient scaling poses a significant challenge when dealing with multiple loss terms, such as data, physics, and rollout losses, which may conflict with each other. Techniques like ConFIG [19] address this issue by aligning gradient directions to mitigate conflicts. Additionally, activation smoothness plays a critical role in PINN performance. As highlighted in [20], the use of non-smooth activation functions can impair performance due to the inaccuracies introduced by automatic differentiation.

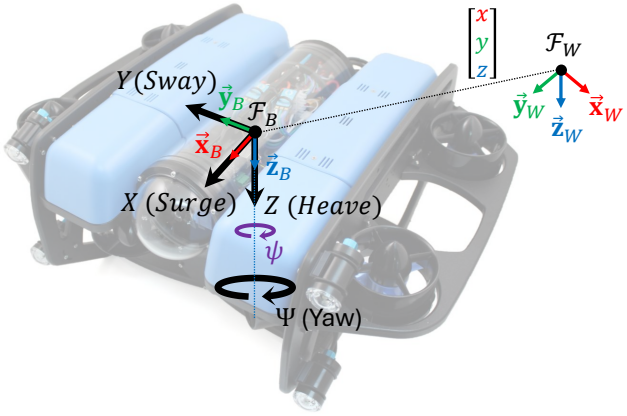


Fig. 1: Coordinate frame of the ROV with respect to the world frame.

III. Problem Formulation

PINNs are effective for solving nonlinear dynamical systems by leveraging both partial knowledge of system dynamics and available data. For a general nonlinear system, where the system's dynamics are represented by a function that is Lipschitz continuous, the solution over time depends on the complete knowledge of this function. However, in practice, this function is often not fully known due to uncertainties and modelling inaccuracies. Instead, nominal dynamics are used, and PINNs approximate the solution using a neural network, which incorporates these nominal dynamics into its training process. In this work, PINNs are applied to the open-loop system identification of the BlueROV2, a versatile underwater ROV commonly used for inspection and manipulation tasks. The BlueROV2 and its coordinate frame are illustrated in Fig. 1, where it is noted that the vehicle has six thrusters, which provide input commands in the form of forces and torques around the ROV's center of gravity, with the thruster dynamics excluded from the model. A simplified 4-degree-of-freedom version of the "Fossen" model [21], with assumptions of small pitch and roll, is adopted for modelling the nominal system dynamics, which can be expressed in state-space form as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} \cos(\psi)u - \sin(\psi)v \\ \sin(\psi)u + \cos(\psi)v \\ w \\ r \\ \frac{1}{m-X_{\dot{u}}}\left(X + (m - Y_{\dot{v}})vr + (X_u + X_{u|u|}|u|)u\right) \\ \frac{1}{m-Y_{\dot{v}}}\left(Y - (m - X_{\dot{u}})ur + (Y_v + Y_{v|v|}|v|)v\right) \\ \frac{1}{m-Z_{\dot{w}}}\left(Z + (Z_w + Z_{w|w|}|w|)w + F_g - V_{sub}\rho_{water}\right) \\ \frac{1}{I_{zz}-N_{\dot{r}}}\left(\Psi - (X_{\dot{u}} - Y_{\dot{v}})uv + (N_r + N_{r|r|}|r|)r\right) \end{bmatrix}, \tag{1}$$

The state vector is defined as $\mathbf{x} = [x, y, z, \psi, u, v, w, r]$, where x, y, and z represent the translational positions in the world-fixed frame $\mathcal{F}_W$, and u, v, and w denote the translational velocities in the body-fixed frame $\mathcal{F}_B$. The transformation

of translational velocities from the body-fixed frame $\mathcal{F}_B$ to the world-fixed frame $\mathcal{F}_W$ is achieved by rotating the body velocities (u, v) around z_b using the yaw angle ψ. Additionally, the rotational velocity is represented by r. The input vector, defined as $\mathbf{u} = [X, Y, Z, \Psi]$, comprises the forces X, Y, and Z, which are the linear components of the combined force applied by the actuators on the AUV body in $\mathcal{F}_B$, and Ψ, which denotes the actuation control moment around z_b. The vehicle's physical parameters include its mass m, the angular moment of inertia about the z-axis I_{zz}, the gravitational constant g, the water density ρ_{water}, and the total submerged volume V_{sub}. The added mass coefficients, are given by $X_{\dot{u}}$, $Y_{\dot{v}}$, $Z_{\dot{w}}$, and $N_{\dot{r}}$. Drag effects are characterized by linear coefficients X_u, Y_v, Z_w, and N_r, and quadratic drag coefficients X_{uc}, Y_{vc}, Z_{wc}, and N_{rc}, which describe the drag along the surge, sway, heave, and yaw directions, respectively.

IV. Proposed Method

A. PINC: Physics-Informed Neural Network with Control

PINNs [22] incorporate known dynamics into a neural network via a *physics loss* enforcing consistency of (1). Standard PINNs take continuous time t and the state $\mathbf{x}(0)$ as inputs. PINC [6] extends PINNs to handle control inputs. We model the solution $\mathbf{x}(t)$ given:

$$\hat{\mathbf{x}}(t) = \mathcal{N}\Big([\,\mathbf{x}(0),\, \mathbf{u}(0),\, t]\Big), \quad \text{for } t \in [\,0, T\,], \tag{2}$$

where $\mathbf{u}(0)$ is the control at the initial step, assumed constant in $[0, T]$.

Discrete-time steps are denoted by index n, and continuous time by t. The nth initial state in a trajectory is $\mathbf{x}_n(0)$ with corresponding control actions $\mathbf{u}_n(0)$. Each point in a trajectory serves as an initial condition, making the one-step-ahead prediction $\hat{\mathbf{x}}_n(T)$ approximate the next state $\mathbf{x}_{n+1}(0)$:

$$\mathbf{x}_n(T) = \mathbf{x}_{n+1}(0). \tag{3}$$

Thus, the one-step-ahead predicted state is:

$$\hat{\mathbf{x}}_n(T) = \mathcal{N}\Big(\underbrace{[\mathbf{x}_n(0) \quad \mathbf{u}_n(0) \quad T]}_{\mathbf{z}_n(0)}\Big) \approx \mathbf{x}_{n+1}(0). \tag{4}$$

Long-horizon predictions (rollouts) over N steps are defined as:

$$\mathbf{x}_0(NT) = \mathbf{x}_N(0). \tag{5}$$

These are achieved by autoregressively applying the dynamics model $\mathcal{N}$:

$$\hat{\mathbf{x}}_0(NT) = \underbrace{\mathcal{N}\Big(\mathcal{N}\big(\mathbf{x}_0(0), \mathbf{u}_0(0), T\big) \ldots, \mathbf{u}_{N-1}(0), T\Big)}_{\text{N times}}. \tag{6}$$

During training, multiple trajectories are batched together indexed by m (batch size N_B), each containing N_D points indexed by n. For multi-step predictions and physics loss computations, the index k and number of points N_P are respectively used for the number of prediction steps and number of physics collocation points.

B. Neural Network Architecture

The residual deep neural network architecture is used to learn ROV dynamics in a physics-informed manner.

1) Residual Formulation for ODE Integration: Solving an ODE over a time interval $I = [0, T]$ can be written as

$$\mathbf{x}(T) = \mathbf{x}(0) + \int_0^T f(\mathbf{x}(\tau), \mathbf{u}(\tau))d\tau. \tag{7}$$

When the control is assumed constant over I (zero-order hold), the integral can be approximated with a neural network that learns to "integrate" the dynamics from $\mathbf{x}(0)$ to $\mathbf{x}(T)$:

$$\mathbf{x}(T) \approx \mathbf{x}(0) + \mathcal{N}\Big(\underbrace{[\mathbf{x}(0) \quad \mathbf{u}(0) \quad T]}_{\mathbf{z}(0)}\Big). \tag{8}$$

2) Layers, Neurons, and Activations: The architecture includes fully connected N_L hidden layers, each containing N_H neurons. Both adaptive `tanh` and `softplus` activation functions are tested with an adaptable parameter, denoted β, that is unique for each layer.

3) State Re-Parameterization for Yaw: Since yaw angle ψ wraps around at $\pm\pi$, it is replaced with two states: $\cos(\psi)$ and $\sin(\psi)$. This ensures the continuity of the states and avoids angle discontinuities.

4) Rotational Structural Information: To reflect the natural geometry of planar motion, the network's predicted increments in x and y are rotated from the body frame $\mathcal{F}_B$ to the world-fixed $\mathcal{F}_W$. Specifically, if the raw network outputs are $\Delta\hat{x}_b$ and $\Delta\hat{y}_b$, they are transformed as follows:

$$\begin{bmatrix}\Delta\hat{x}_n \\ \Delta\hat{y}_n\end{bmatrix} = \begin{bmatrix}\cos(\hat{\psi}) & -\sin(\hat{\psi}) \\ \sin(\hat{\psi}) & \cos(\hat{\psi})\end{bmatrix} \begin{bmatrix}\Delta\hat{x}_b \\ \Delta\hat{y}_b\end{bmatrix}. \tag{9}$$

Because the model learns increments in the body frame, it can capture the simpler local dynamics. Those increments are then rotated back into $\mathcal{F}_W$.

5) Layer Normalization: Layer normalization [23] is another regularization technique employed in this work, which normalizes activations within each layer using learnable parameters, mitigating internal covariate shifts.

C. Loss Functions

PINC integrates multiple loss terms to accurately learn ROV dynamics by combining data-driven predictions with physics-based constraints.

1) One-step-ahead Prediction Loss: The one-step-ahead prediction loss ($\mathcal{L}_D$) measures the mean squared error between the predicted state $\hat{\mathbf{x}}_{n,m}(T)$ and the ground-truth next state $\mathbf{x}_{n+1,m}(0)$ for each consecutive pair in all trajectories:

$$\mathcal{L}_D = \frac{1}{N_B(N_D-1)} \sum_{m=0}^{N_B-1} \sum_{n=0}^{N_D-2} ||\mathbf{x}_{n+1,m}(0) - \hat{\mathbf{x}}_{n,m}(T)||_2^2, \tag{10}$$

This loss encourages the model to match the known data at discrete intervals T.

2) Physics Loss: The physics loss ($\mathcal{L}_P$) regularizes the model's predictions to respect the underlying physics by penalizing deviations from the governing differential equations. Fig. 2 illustrates the processing of the data and physics losses within the PINC framework. Each point in each trajectory has some corresponding collocation points N_P, indexed by k. Thus, the total number of collocation points evaluated in a batch is $N_D \times N_B \times N_P$. The kth collocation point in the nth trajectory, at its mth point is denoted $T^{coll}_{n,m,k}$, which is sampled by LHS on the interval $T^{coll} \in [0, T]$. The physics residual at each collocation point is defined as:

$$F(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}) = \dot{\mathbf{x}} - f(\mathbf{x}, \mathbf{u}). \tag{11}$$

When $F(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}) = 0$, the learned dynamics perfectly match the true dynamics. Specifically the physics residual $F(\cdot) = F\left(\hat{\mathbf{x}}_{n,m}(T^{\text{coll}}_{n,m,k}), \dot{\hat{\mathbf{x}}}_{n,m}(T^{\text{coll}}_{n,m,k}), \mathbf{u}_{n,m}(0)\right)$ is used in the physics loss. The physics loss is then computed as the mean squared error of these residuals across all collocation points, trajectories, and time steps:

$$\mathcal{L}_P = \frac{1}{N_B N_D N_P} \sum_{k=0}^{N_P-1} \sum_{m=0}^{N_B-1} \sum_{n=0}^{N_D-1} \|F(\cdot)\|_2^2 . \tag{12}$$

3) Initial Condition Loss: The initial condition loss $\mathcal{L}_{IC}$ ensures internal consistency of the network's outputs at $t = 0$, treating each data point as a new initial condition. This is formulated as follows:

$$\mathcal{L}_{IC} = \frac{1}{N_B N_D} \sum_{m=0}^{N_B-1} \sum_{n=0}^{N_D-1} ||\mathbf{x}_{n,m}(0) - \hat{\mathbf{x}}_{n,m}(0)||_2^2. \tag{13}$$

4) Rollout Loss: Inspired by [15], an N-step-ahead (rollout) loss is used to penalize the accumulation of errors when predicting multiple steps forward. In a trajectory with N_D, points, only the first $N_R = N_D - N_{pred}$ points can be rolled out. For each trajectory, rollouts are initiated from the first N_R points, and the predicted sequence is compared to the ground truth trajectory using the rollout loss function:

$$\mathcal{L}_R = \frac{1}{N_B N_R N_P} \sum_{k=1}^{N_P} \sum_{m=0}^{N_B-1} \sum_{n=0}^{N_R-1} ||\mathbf{x}_{n+k,m}(0) - \hat{\mathbf{x}}_{n,m}(kT)||_2^2. \tag{14}$$

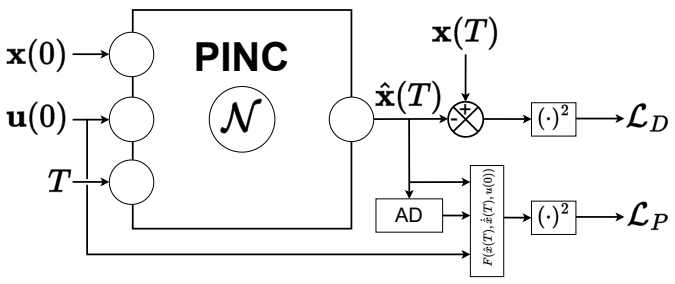


Fig. 2: PINC model output used to calculate the MSE data loss by comparing it to the ground truth state and used with AD and the underlying physics function to find the MSE physics loss.

5) Physics Rollout Loss: When both the rollout loss and physics loss are included in the optimization, a physics rollout loss is also computed and incorporated to regularize the intermediate states in the multistep predictions, ensuring adherence to the governing differential equations.

D. Gradient Weighting

Multitask learning is addressed using the three methods following methods for combining gradients. The first method, simple summation, directly adds the gradients from each loss term. The second approach, ConFIG [19], dynamically adjusts and reorients gradients so that the combined gradient does not conflict with any individual gradient. It also scales the magnitude based on cosine similarity, making it larger (or smaller) depending on how aligned or misaligned the losses are. The final method, gradient normalization, normalizes each gradient $\mathbf{g}_n$ to match the norm of a "reference" gradient (usually the data-loss gradient $\mathbf{g}_0$), and is then weighed by user-set coefficients w_n. Finally, the sum is re-normalized to match the reference gradient's norm:

$$\bar{\mathbf{g}}_n = \mathbf{g}_n \frac{\|\mathbf{g}_0\|_2}{\|\mathbf{g}_n\|_2}, \quad \mathbf{g} = \sum_{n=0}^{N-1} w_n \bar{\mathbf{g}}_n, \quad \bar{\mathbf{g}} = \mathbf{g} \frac{\|\mathbf{g}_0\|_2}{\|\mathbf{g}\|_2}. \tag{15}$$

In addition, the norm of the combined gradient is clipped at $||\bar{\mathbf{g}}||_2 \leq c_{|g|,max} = 5.0$ to avoid huge steps due to high variance or numeric issues.

E. Model Evaluation

The learned models are evaluated in two ways:

1) Accuracy of Predictions:

- *One-step-ahead* prediction error on the development set ($\mathcal{L}_{data,dev}$) and test set ($\mathcal{L}_{data,test,interp}$ for interpolation and $\mathcal{L}_{data,test,extrap}$ for extrapolation).
- *Rollout* prediction error ($\mathcal{L}_{roll,dev}$) over 10 steps, capturing cumulative error in sequential predictions.

2) Long-Horizon Prediction Validity: The valid prediction time (VPT) is the largest time interval over which the model's predicted position error remains below a threshold, i.e.

$$\|\mathbf{e}_{\text{pos}}(t)\|_2 = \sqrt{\Delta_x^2 + \Delta_y^2 + \Delta_z^2} \leq 0.05\,\text{m}, \tag{16}$$

where $\Delta_x = x(t) - \hat{x}(t)$, $\Delta_y = y(t) - \hat{y}(t)$ and $\Delta_z = z(t) - \hat{z}(t)$. For all $t <$ VPT, the predictions are considered valid up to that horizon. Once the error exceeds $0.05\,\text{m}$, the prediction is deemed unsafe or unusable for control.

To evaluate the VPT, each trajectory is rolled out from its first initial condition in full, using $N_{steps} = 65$, so it matches the ground-truth trajectory length. The VPT is reported in seconds but can readily be converted to discrete steps by dividing by the sampling interval T. This metric is computed on the development, interpolation, and extrapolation test sets.

The interpolation and extrapolation test sets are created using time interval sizes T different from those used for training and validation while maintaining the same input types and initial condition intervals as the validation set. To assess interpolation, the time interval size is decreased, testing the

model's robustness and its ability to learn the differential equation solution over the entire interval $t \in [0, T]$, aided by the physics loss. For extrapolation, the interval size T. is increased. Test sets are generated with the following time interval sizes:

$$\begin{cases} T_{test,interp} &= T - 0.25T = 0.08 - 0.02 = 0.06 \\ T_{test,extrap} &= T + 0.25T = 0.08 + 0.02 = 0.1. \end{cases} \tag{17}$$

V. Simulation Results

A fully translational model with yaw rotation was selected because it incorporates quadratic damping, rotational non-linearities, and gravitational and buoyancy forces. The quadratic non-linearities arise from damping effects. At the same time, the rotational non-linearity in yaw, arising from the planar velocities u and v being rotated from the body frame $\mathcal{F}_B$ to the world frame $\mathcal{F}_W$, in the position dynamics, introduces a limited form of rotational dynamics, allowing to test rotational effects.

Remark 1: All quantities are provided in SI units, i.e., time is in seconds [s], and position is in meters [m].

Remark 2: For ease of comparing, losses $\mathcal{L}_D$, $\mathcal{L}_R$, and $\mathcal{L}_P$ are reported on a $\log_{10}(\cdot)$ scale.

Remark 3: The loss terms in the figures are labeled as follows: $\mathcal{L}_{data,dev}$ (L_1), $\mathcal{L}_{roll,dev}$ (L_2), $\mathcal{L}_{phy,dev}$ (L_3), $\mathcal{L}_{data,test,interp}$ (L_4), and $\mathcal{L}_{data,test,extrap}$ (L_5). The metrics VPT_{dev}, $\text{VPT}_{test,interp}$, and $\text{VPT}_{test,extrap}$ are labeled as VPT_1, VPT_2, and VPT_3, respectively.

A. Data Generation

A dataset was generated and partitioned into training, development (validation), and test sets, each with distinct initial conditions and input types. Numerical integration of the dynamics (1) was performed using trajectories starting from randomly sampled initial states $\boldsymbol{x}_0$. Each state variable was independently drawn from uniform intervals: position $(x, y, z) \in [-x_{\max}, x_{\max}], [-y_{\max}, y_{\max}], [-z_{\max}, z_{\max}]$; heading angle $\psi \in [-\pi, \pi]$; linear velocities (u, v, w) within specified bounds with $w \geq 0$ to simulate diving behavior; and yaw rate r within a designated range.

Control input sequences $\mathbf{u}(t)$ for the training set were generated using ramp-based patterns with random signs, offsets, and ramp-up/ramp-down profiles. Sine waves with fixed amplitude, random frequencies, and phases were used for the development and test sets. Each input channel was scaled to ensure realistic input magnitudes, as detailed in the respective experiments.

Trajectories were integrated over a total time of $T_{tot} = 5.2$ seconds with a sampling period of $T = 0.08$ seconds, resulting in $N_{steps} = 66$ points per trajectory. To maintain continuity in yaw, ψ was represented by its sine and cosine values, i.e., $(\cos(\psi), \sin(\psi))$ instead of ψ directly. For physics loss computation, additional collocation points $\tau \in [0, T]$ were sampled within each sampling interval using Latin Hypercube Sampling (LHS). The number and placement of collocation points (N_P) were adjustable to enforce the governing physics throughout each interval.

B. Experiment Protocol

A series of experiments is conducted to examine how different design factors influence the performance of PINC. By systematically varying these factors, the aim is to identify PINC configurations that yield the best trade-offs between accuracy, robustness, and computational efficiency. We vary:

1) **Neural Network Size:** Varying the number of layers and neurons to balance expressiveness and complexity.
2) **Residual Connection Effect:** Assessing the impact of an integral-form residual connection on model accuracy.
3) **Activation Function Type:** Comparing adaptive activation functions to identify optimal non-linearities.
4) **Batch Size Variations:** Investigating batch size as a regularizer and its effect on convergence speed.
5) **Physics Loss:** Evaluating whether embedding known physical dynamics in the loss improves model generalization.
6) **Rollout Loss:** Studying how penalizing multi-step prediction affects long-horizon accuracy.
7) **Initial Condition Loss:** Ensuring consistency at $T = 0$ to improve predictions at future time steps.
8) **Gradient Weighting:** Testing summation, ConFIG, and normalization-based schemes for combining gradients.
9) **Physics Collocation:** Adjusting the number and placement of collocation points to enforce system dynamics.
10) **Noisy Inputs:** Injecting Gaussian noise to evaluate the model's resilience to sensor and measurement errors.
11) **Learning Rate Scheduling:** Exploring if a decaying learning rate improves loss convergence.

C. Data and Training Parameters

Unless otherwise stated, the following hyperparameters are used:

- Models are trained for $N_{\text{epoch}} = 1200$ epochs (or more if needed for convergence).
- A batch size of $N_{\text{batch}} = 3$ is used (unless varied in experiments on batch size).
- The AdamW optimizer is applied with an initial learning rate of $\text{lr}_0 = 8 \times 10^{-3}$ and no scheduling, except when explicitly stated, where the scheduler uses a 'minimum value of $\text{lr}_{\min} = 10^{-4}$ with a "patience" of 100.
- Layer normalization is employed on every second layer.
- Gradient weighting is applied using a normalization scheme with weights: $\{w_{\text{data}} = 1.0, w_{\text{roll}} = 1.0, w_{\text{phy}} = 0.5, w_{\text{phy,roll}} = 0.5, w_{\text{ic}} = 0.5\}$.
- The output planar position change is rotated by the predicted yaw (unless the structural information is ablated).

The models are trained using a dataset of $N_{\text{traj,data}} = 400$ trajectories with ramp-based inputs. The development set initially contains $N_{\text{traj,dev}} = 80$ trajectories, but for evaluation, is increased to $N_{\text{traj,dev}} = 1000$ to match the size of the interpolation and extrapolation test sets. Both the development and test sets use sinusoidal inputs. The number of steps is kept at $N_{\text{steps}} = 66$, and the total time in the test sets is varied to ensure consistent trajectory lengths.

For all experiments except the last two, the training data are generated under the following initial condition ranges: $x_{\max} = y_{\max} = z_{\max} = 1.0$, $\psi_{\max} = \pi$, $u_{\max} = 1.0$, $v_{\max} = 0.0$, $w_{\max} = 0.1$, $r_{\max} = 0.0$. The inputs for training are ramp-based rather than step-based. Before any sign, scaling, or offset is applied, each ramp starts at 0, increases to 1 (peaking at $0.5T$), and then decreases back to 0. The offsets are sampled from a zero mean normal distribution with $\sigma^2 = 0.25$, the ramp's sign from a Bernoulli random variable with $p = 0.5$. All experiments are conducted using a single seed for randomness in data generation and network initialization. The seed is fixed at 0 for reproducibility.

The development and test sets are generated with $x_{\max} = y_{\max} = z_{\max} = u_{\max} = v_{\max} = w_{\max} = r_{\max} = 0.0$ and $\psi_{\max} = \pi$, so that the models are evaluated only on their ability to handle different inputs rather than different initial conditions. The inputs are sinusoids with an amplitude of $A = 3$, a random frequency sampled uniformly in the interval $[0.01, 0.2]$, and a random phase uniformly sampled in the interval $[0, 2\pi]$.

Remark 4: The inputs in all datasets are scaled as follows: Y is scaled down by 0.1, M_z is scaled down by 0.05, and Z is scaled up by 5 and then taken as an absolute, ensuring only downward motion is commanded since the ROV's static buoyancy force naturally surfaces the ROV.

D. Neural Network Architecture Experiments

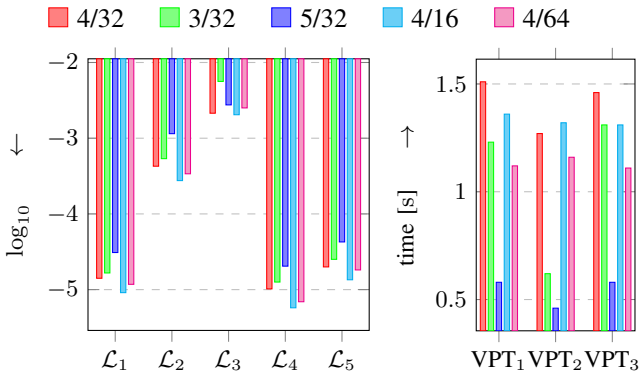
1) Neural Network Size: The number of hidden layers N_L and the number of neurons N_H in each layer is varied in this experiment, which performance metrics are shown in Fig. 3a. It is found that all losses are the lowest and VPTs the highest for the configuration with $N_L = 4$ and $N_H = 32$. This configuration will, therefore, be used in all subsequent experiments.

2) Residual Connection: The residual connection is ablated in the experiment shown in Fig. 3b, comparing the model with and without the input state residual connection. Without the residual connection that makes the model follow the integration form of an ODE solution, its performance is degraded, having a $\mathcal{L}_{data,dev}$ that is more than 4 orders of magnitude higher.

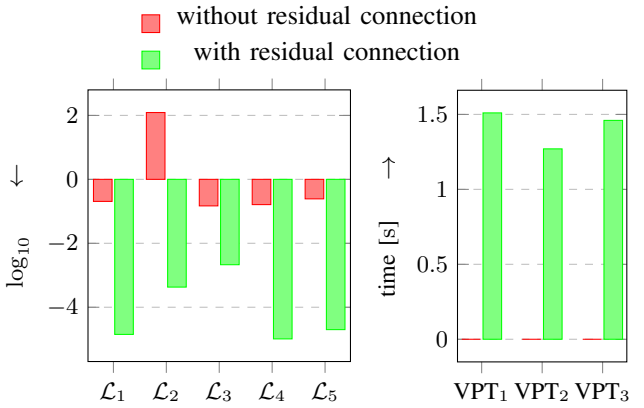
3) Activation Function: The use of an adaptive `softplus` activation function is compared with an adaptive `tanh` as the model's non-linearity. The hypothesis is that `tanh` might yield smaller errors due to its bounded nature but could suffer from vanishing gradients, while `softplus` is unbounded and generally avoids those problems. The results shown in Fig. 3c indicate that while the losses are mostly lower with `tanh`, the VPTs are significantly better when using `softplus`. A likely explanation is that `softplus` remains smooth and avoids saturation.

E. Effect of adding Rollout Loss on Learning

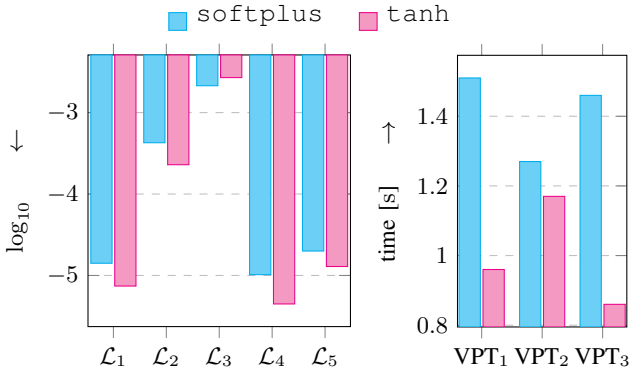
The impact of adding the rollout loss $\mathcal{L}_P$ (Fig. 4) to the overall learning process is explored by comparing the performance of the PINC with and without the rollout loss. While the introduction of the physics loss $\mathcal{L}_P$ alone leads to improvements in several performance metrics, the inclusion of the rollout loss does not consistently enhance the learning. In fact, in some instances, the rollout loss negatively impacts performance, suggesting that the additional regularization from the rollout transformation may not always contribute positively to the overall model training. These results highlight the complex interaction between structural information and physics-based regularization, with the rollout loss sometimes leading



(a) Neural network size.



(b) Residual connection ablation experiment.



(c) Adaptive activation function comparison.

() Fig. 3: Overview of architecture experimental results: (a) Neural Network Size: $N_L = 4$ and $N_H = 32$ yield the lowest losses and highest VPTs. (b) Residual Connection: Ablating the residual connection causes a significant performance drop, with $\mathcal{L}_{data,dev}$ over four orders of magnitude higher. (c) Activation Function: `softplus` achieves better VPTs than `tanh`.

to suboptimal learning outcomes.

F. Input Noise Robustness Experiment

Real-world ROV data often contain sensor or measurement noise. The model's robustness to noisy inputs is assessed by injecting Gaussian noise, $\boldsymbol{\epsilon}$, with a standard deviation of $\sigma = 0.05$ into the neural network's inputs during training. In each epoch, a noise vector for each state in each trajectory is sampled and injected:

$$\mathbf{x}_{\text{noisy}} = \mathbf{x} + \boldsymbol{\epsilon}. \tag{18}$$

In Fig. 5, the no-physics and physics+rollout models degrade when noise is introduced. Nonetheless, the physics+rollout model remains more robust, implying that physics regularization can partially compensate for noisy input signals when training the model. The models are evaluated without noise.

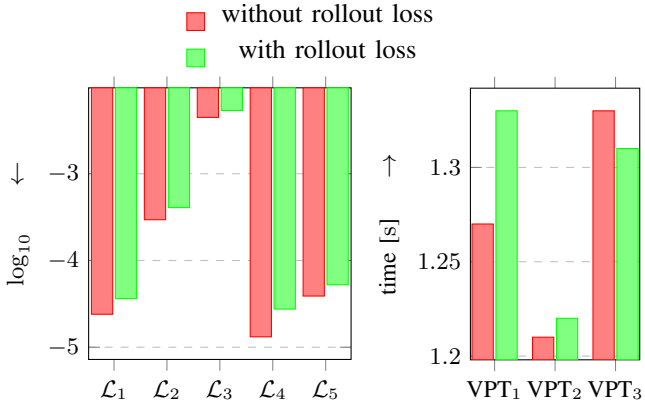


Fig. 4: Effect of adding rollout loss vs. physics loss on VPT: Incorporating rollout loss with physics loss does not yield any additional performance gains compared to using physics loss alone, as shown in the Loss and VPT metrics.

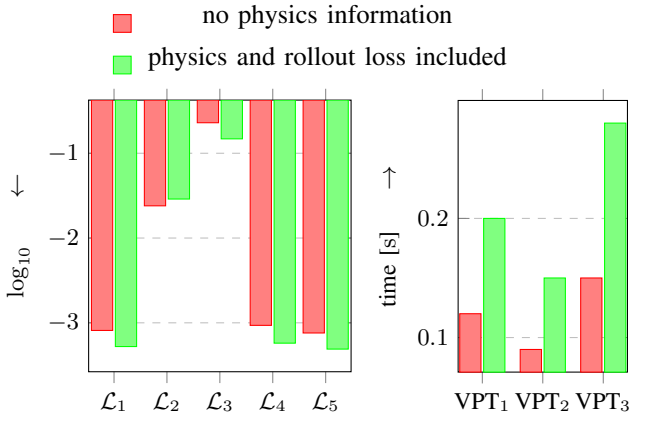


Fig. 5: Effect of noise on learning: Noise was introduced into the ROV simulation model to evaluate the learning performance. Incorporating physics information significantly reduces underfitting and enhances robustness to noise, enabling more reliable learning.

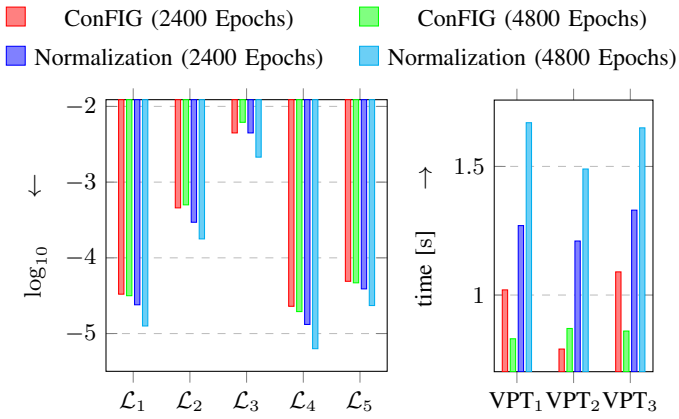


Fig. 6: Comparison of PINC models trained with ConFIG and gradient normalization techniques for 2400 and 4800 epochs, using the best network architecture configuration determined from prior ablation studies. Results show that our proposed normalization method consistently outperforms the state-of-the-art ConFIG method in terms of VPT and loss reduction.

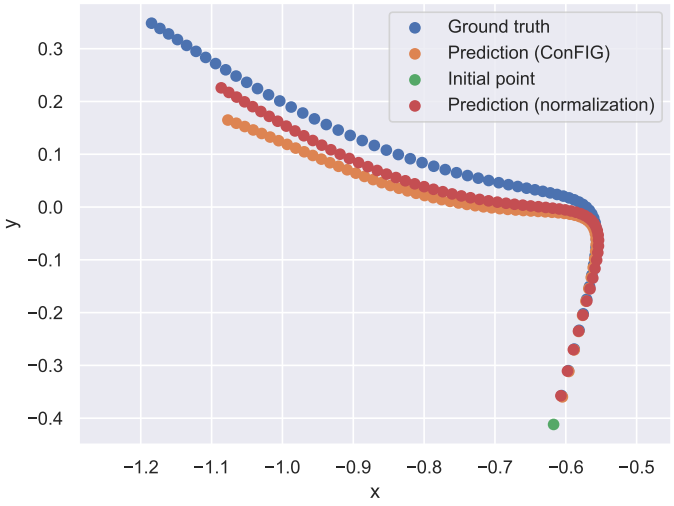


Fig. 7: Many-step-ahead prediction position trajectories using the best model configuration with ConFIG and gradient normalization.

G. Best Model Configuration Experiments

Based on our findings, we combined the optimal architectural and training choices, utilizing only data and physics losses. Additionally, we evaluated the impact of adding extra collocation points, which had previously improved Valid Prediction Times (VPTs). These models were trained on a more complex dataset using a learning rate scheduler and a batch size of $N_B = 10$ for 2400 and 4800 epochs. The performance metrics are presented in Fig. 6.

Thus, the best PINC configuration employs the data and physics losses, learning rate scheduling, gradient normalization, a batch size of $N_B = 10$, and a single collocation point in the prediction interval T. Fig. 7 illustrates the performance differences between ConFIG and gradient normalization models.

VI. Conclusions and Future Work

This work applied PINC to model an underwater vehicle's dynamics, mapping the current state, input, and time to the predicted next state at that given time instant. Empirically, combining one-step-ahead data loss with physics-based regularization provided the best long-horizon accuracy at minimal computational overhead. A key insight was that residual connections, treating the network output as an integral increment, are essential. Adaptive learning rate scheduling, gradient normalization and weighting, and batch sizes in the range of 10–20 all enhanced predictive performance.

Despite these promising results, several refinements remain. A single fixed collocation point consistently offered strong results, suggesting alternative architectures such as RNNs could further reduce overhead. Benchmark datasets for ROV system identification, with realistic inputs, would also provide consistent performance comparisons. Extending PINC to handle full rotational dynamics – optimally representing roll, pitch, and yaw – could improve accuracy for realistic manoeuvres. Finally, investigating the minimal set of losses, real-world testing, integration with MPC, and online adaptation would make PINC an even more versatile framework for ROV applications.

References

[1] A. Amer, O. Álvarez-Tuñón, H. İ. Uğurlu, J. L. F. Sejersen, Y. Brodskiy, and E. Kayacan, "Unav-sim: A visually realistic underwater robotics simulator and synthetic data-generation framework," in *2023 21st International Conference on Advanced Robotics (ICAR)*, 2023, pp. 570–576.

[2] C. Kunz, C. Murphy, R. Camilli, H. Singh, J. Bailey, R. Eustice, M. Jakuba, K.-i. Nakamura, C. Roman, T. Sato *et al.*, "Deep sea underwater robotic exploration in the ice-covered arctic ocean with auvs," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3654–3660.

[3] A. Amer, M. Mehndiratta, J. le Fevre Sejersen, H. X. Pham, and E. Kayacan, "Visual tracking nonlinear model predictive control method for autonomous wind turbine inspection," in *2023 21st International Conference on Advanced Robotics (ICAR)*. IEEE, 2023, pp. 431–438.

[4] S. Lakshminarayanan, D. Duecker, A. Sarabakha, A. Ganguly, L. Takayama, and S. Haddadin, "Estimation of External Force Acting on Underwater Robots," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024, pp. 3125–3131.

[5] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[6] E. A. Antonelo, E. Camponogara, L. O. Seman, E. R. de Souza, J. P. Jordanou, and J. F. Hubner, "Physics-Informed Neural Nets for Control of Dynamical Systems," *Neurocomputing*, vol. 579, Apr. 2024, arXiv:2104.02556 [cs].

[7] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering*. Cambridge University Press, 2021.

[8] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, 3rd ed., ser. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 2008.

[9] A. Amer, M. Mehndiratta, Y. Brodskiy, and E. Kayacan, "Empowering autonomous underwater vehicles using learning-based model predictive control with dynamic forgetting gaussian processes," *IEEE Transactions on Control Systems Technology*, 2025.

[10] R. Faria, B. Capron, A. Secchi, and M. De Souza, "A data-driven tracking control framework using physics-informed neural networks and deep reinforcement learning for dynamical systems," *Engineering Applications of Artificial Intelligence*, vol. 127, Jan. 2024.

[11] S. Sanyal and K. Roy, "Ramp-net: A robust adaptive mpc for quadrotors via physics-informed neural network," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1019–1025.

[12] M. Lutter and J. Peters, "Combining physics and deep learning to learn continuous-time dynamics models," *The International Journal of Robotics Research*, vol. 42, no. 3, pp. 83–107, Mar. 2023.

[13] J. Ma, Y. Li, J. Tu, Y. Zhang, J. Ai, and Y. Dong, "Development and Implementation of Physics-Informed Neural ODE for Dynamics Modeling of a Fixed-Wing Aircraft Under Icing/Fault," *Guidance, Navigation and Control*, vol. 04, no. 01, Feb. 2024.

[14] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[15] Y. Zhao, Z. Hu, W. Du, L. Geng, and Y. Yang, "Research on Modeling Method of Autonomous Underwater Vehicle Based on a Physics-Informed Neural Network," *Journal of Marine Science and Engineering*, vol. 12, no. 5, p. 801, May 2024.

[16] J. Gao, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann, "Sim-to-Real of Soft Robots with Learned Residual Physics," *IEEE Robotics and Automation Letters*, pp. 1–8, 2024, conference Name: IEEE Robotics and Automation Letters.

[17] H. Krauss, T.-L. Habich, M. Bartholdt, T. Seel, and M. Schappler, "Domain-decoupled Physics-informed Neural Networks with Closed-form Gradients for Fast Model Learning of Dynamical Systems," Aug. 2024.

[18] J. E. Kittelsen, E. A. Antonelo, E. Camponogara, and L. S. Imsland, "Physics-Informed Neural Networks with Skip Connections for Modeling and Control of Gas-Lifted Oil Wells," *Applied Soft Computing*, vol. 158, p. 111603, Jun. 2024.

[19] Q. Liu, M. Chu, and N. Thuerey, "ConFIG: Towards Conflict-free Training of Physics Informed Neural Networks," Aug. 2024.

[20] J. Nicodemus, J. Kneifl, J. Fehr, and B. Unger, "Physics-informed Neural Networks-based Model Predictive Control for Multi-link Manipulators," *IFAC-PapersOnLine*, vol. 55, no. 20, pp. 331–336, Jan. 2022.

[21] T. I. Fossen, "Handbook of marine craft hydrodynamics and motion control," in *Handbook of marine craft hydrodynamics and motion control*, 2nd ed. Hoboken, NJ: Wiley, 2021.

[22] 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*, vol. 378, pp. 686–707, Feb. 2019.

[23] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," Jul. 2016.