PHYSICS-INFORMED STATE-SPACE NEURAL NETWORKS FOR TRANSPORT PHENOMENA

Akshay J. Dave
Nuclear Science and Engineering
Argonne National Laboratory
Lemont, IL 60439
ajd@anl.gov

Richard B. Vilim
Nuclear Science and Engineering
Argonne National Laboratory
Lemont, IL 60439

ABSTRACT

This work introduces Physics-informed State-space neural network Models (PSMs), a novel solution to achieving real-time optimization, flexibility, and fault tolerance in autonomous systems, particularly in transport-dominated systems such as chemical, biomedical, and power plants. Traditional data-driven methods fall short due to a lack of physical constraints like mass conservation; PSMs address this issue by training deep neural networks with sensor data and physics-informing using components' Partial Differential Equations (PDEs), resulting in a physics-constrained, end-to-end differentiable forward dynamics model. Through two in silico experiments — a heated channel and a cooling system loop — we demonstrate that PSMs offer a more accurate approach than a purely data-driven model. In the former experiment, PSMs demonstrated significantly lower average root-mean-square errors across test datasets compared to a purely data-driven neural network, with reductions of 44 %, 48 %, and 94 % in predicting pressure, velocity, and temperature, respectively.

Beyond accuracy, PSMs demonstrate a compelling multitask capability, making them highly versatile. In this work, we showcase two: supervisory control of a nonlinear system through a sequentially updated state-space representation and the proposal of a diagnostic algorithm using residuals from each of the PDEs. The former demonstrates PSMs' ability to handle constant and time-dependent constraints, while the latter illustrates their value in system diagnostics and fault detection.

Keywords State-space model · Physics-informed · Data-driven
Supervisory Control · Diagnostics

1 Introduction

Autonomous systems are envisioned to achieve real-time optimization, be flexible in operation, and be fault-tolerant [1]. These capabilities necessitate a model-based approach. However, purely data-driven methods, which have proliferated in the literature, lack physical constraints like mass conservation, precluding their adoption by industry. In this work, we propose a method that *fuses* physical measurements from sensors with physical knowledge of the conservation laws. The application of this method is specifically developed for systems dominated by transport phenomena.

Transport phenomena, which describe the movement of mass, momentum, and energy, are fundamental to many engineering fields as they offer a predictive understanding of systems' behavior. In chemical engineering, these phenomena facilitate the design and optimization of processes in reactors, separators, and heat exchangers, which are essential for chemical, fuel, and pharmaceutical production [2]. Biomedical engineering applies transport phenomena to comprehend nutrient, oxygen, and waste transportation in biological systems like the circulatory system, driving the development of new medical technologies such as artificial organs [3]. Power plant engineering utilizes transport phenomena in optimizing heat and mass transfer in boilers and heat exchangers for efficient and safe power generation and in emission control [4]. They are also vital in nuclear reactor design, where effective heat transfer from the

reactor core to the coolant is paramount [5]. Consequently, modeling transport phenomena is crucial for the design, optimization, and enhancement of engineering systems and processes.

A set of transport equations models transport phenomena. The solution of transport equations involves formulating functions of space and time for multiple fields (e.g., pressure, velocity, temperature), given a set of boundary and initial conditions. Classically, there have been two major approaches to solving transport equations: analytical and numerical. The former applies to situations where the problem setting or transport phenomena are simple enough to directly use mathematical methods (e.g., separation of variables) – this is generally not the case for systems of interest described above. The latter is used for systems with complex geometries, tightly coupled fields, time-dependent boundary/initial conditions, or parameters that vary with the field itself. Numerical methods include finite difference, finite volume, or finite element [6]. These methods discretize the domain into a grid, and values at fixed locations represent the unknown fields. The transport equations are then approximated using numerical algorithms (e.g., the forward Euler method, the Crank-Nicolson method) and solved iteratively until convergence is achieved.

The numerical methods are the current state-of-the-art approach to modeling transport phenomena for engineering systems. Their main advantage is versatility in addressing a range of complex problems with time-dependent boundary/initial conditions. However, they have several disadvantages. *Discretization error*: numerical methods discretize the domain and approximate the transport equations. This process introduces error, which can be reduced with finer grids – unfortunately increasing computational cost. *Convergence*: because numerical methods involve iterative algorithms, convergence to a solution is not guaranteed and can result in numerical instability. *Static assumptions*: generally, several parameters in a transport equation must be predefined (e.g., the friction factor). These assumptions form a rigidity around the application of numerical methods: they do not provide a straightforward path to adapt to changing system conditions. The first two disadvantages preclude numerical methods as a basis for the control of engineering systems. A majority of model-based control algorithms assume a reduced order or surrogate model for control [7, 8].

In this paper, we introduce Physics-informed State-space Neural Networks for transport phenomena. Physics-Informed Neural Networks (PINNs) are a type of artificial neural network (ANN) used to solve partial differential equations (PDEs) [9]. PINNs combine the expressiveness and flexibility of ANNs with the prior knowledge of physical laws and constraints encoded in PDEs. In a PINN approach, the neural network is trained to approximate the solution of PDEs, given appropriate boundary and initial conditions, by minimizing the difference between the network outputs and the PDE constraints. Thus, PINNs could be a candidate for the solution of a system of PDEs representing transport phenomena. PINNs have several advantages. *Physical priors*: PINNs incorporate prior knowledge about the system through physical constraints. In the context of transport phenomena, prior knowledge is the conservation of mass, momentum, and energy in the spatiotemporal domain. *Mesh-free*: PINNs do not require a mesh or a grid to discretize the spatial domain, as numerical methods do. Instead, PINNs use a neural network to represent the solution in a continuous fashion, without the need for discretization. First, this aspect eliminates the need for mesh generation, which can be time-consuming and challenging. Second, it eliminates discretization error (described in the previous paragraph) arising from the mesh. *Flexible training data*: PINNs can be trained with noisy or scarce data. Thus, sensor measurements, whose outputs have aleatoric sources of noise, may be used to train PINNs. Additionally, the scarcity of data is an essential aspect as the number of sensor locations (i.e., cost) required to utilize PINNs is low.

Specifically, we introduce physics-informed *state-space* neural networks for transport phenomena. A plurality of previous PINN work focuses on solutions of one or more PDEs, with fixed boundary and initial conditions [10],

$$x\left(z,t\right)=\mathcal{F}\left(z,t\right)\ , \tag{1}$$

where $\mathcal{F}$ is a PINN, z is a position vector, t is time, and x is the field variable(s). However, in a control or state estimation setting, a form representing,

$$x\left(z,t,v\right)=\mathcal{F}\left(z,t,v\right)\ , \tag{2}$$

is needed, where v represents a vector of arbitrary control actions. The control actions modify the boundary conditions (BCs) of the problem. In this setting, the PINN must be able to generalize to a much larger input space. Arnold and King [11] attempted a similar task for a system governed by a single PDE but did not train a monolithic PINN for varying v – rather multiple PINNs corresponding to a grid of anticipated control actions.

1.1 Objectives & Contributions

The objective of this work is to develop and demonstrate a method that generates a state-space model by fusing physical measurements from sensors with knowledge of the fundamental physics of the system. We refer to the outcome as a Physics-informed State-space neural network Model (PSM), detailed in Section 2.4. With respect to the existing literature, our work has the following major contributions:

- Integration of PINNs with State-Space Models: Our work introduces a method that combines PINNs with state-space modeling, using a single neural network that can accommodate multiple and continuous control

actions, and output multiple fields over the entire spatial domain of the system. This creates a robust modeling approach that enhances accuracy and integrates physical laws with sensor data. This is particularly significant in systems where the underlying physical law(s) is known yet cannot be captured in a purely data-driven approach due to a lack of instrumentation (or preclusion due to extreme conditions).

- Application to Transport-Dominated Systems: The PSM is specifically developed for engineered systems dominated by transport phenomena, such as those found in chemical, biomedical, and power plants. In these systems, there are several sensors distributed throughout the plant. We simultaneously apply three PDEs (the current state-of-the-art in modeling fluid transport) to constrain the dynamics of the neural network.
- Demonstration through Experiments: We validate the efficacy of PSMs through two in silico experiments – a heated channel and a cooling system loop. These experiments demonstrate the capability in handling complex real-world scenarios in transport-dominated systems, providing evidence of the practical applicability of our approach. Our work demonstrates the robustness in handling both homoscedastic and heteroscedastic noise in sensor measurements, a common challenge in real-world data collection.
- Demonstration of Multitask Capability: The PSM's end-to-end differentiability allows them to be used for model-based control and diagnostics schemes. For the former, we showcase the Reference Governor supervisory control scheme and use it to constrain excessive temperatures during operational transients. For the latter, the ability to compute residuals from the PDEs enables the identification of system degradations or faults.

The paper is organized as follows. In Section 2, the definition and solution of the transport equations, derivation of our proposed method, and its training procedure are presented. Two separate experimental settings were explored. In Section 3, results for modeling a heated channel geometry are presented, and a supervisory control method is demonstrated. In Section 4, results for modeling a cooling system are presented, and a demonstration of a capability to use our proposed method for physics-based diagnostics during a system fault. Finally, Section 5 contains a discussion and conclusions on advantages, disadvantages, and additional capabilities to be explored.

2 Methods

2.1 Modeling fluid transport

There is a large corpus of equations that model fluid transport. This work focuses on one-dimensional single-phase flow at low-speed, i.e., conditions ubiquitous in engineering systems described in the introduction. Under these conditions, the conservation of mass, momentum, and energy are modeled by the following set of PDEs [12]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \left(\rho u\right)}{\partial z} = 0 \,, \tag{3}$$

$$\rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial z} = -\frac{\partial p}{\partial z} + \rho g - \frac{f}{D_h} \frac{\rho u |u|}{2} \,, \tag{4}$$

$$\rho C_p \frac{\partial T}{\partial t} + \rho C_p u \frac{\partial T}{\partial z} = q''' \,. \tag{5}$$

In all equations, z is the spatial dimension, and t is the temporal dimension. This work uses the field variable set (p, u, T), where p is the pressure, u is the one-dimensional velocity, and T is the temperature. Additionally, there are several parameters. These parameters are either constants or modeled via closure relationships. A closure relationship models density, $\rho = \rho(p, T)$, pre-defined via experiments. Specific heat capacity is modeled by closure relationships, $C_p = C_p(p, T)$, pre-defined via experiments. The friction factor, f, can be obtained via closure relationships (Moody chart) or determined via experiments. The gravitational coefficient, g, is a constant that will depend on the orientation of the component(s) with the gravitational field (and its magnitude). The hydraulic diameter, D_h, can be obtained via manufacturer specifications or component measurement. The volumetric heat source, q''', depends on the configuration of the thermal energy source, e.g., heater power magnitude and its location.

2.2 Numerical solution

The approach to obtaining numerical solutions to the transport equations is discussed. A solver, System Analysis Module (SAM) [13], is utilized. The SAM code specializes in the high-fidelity simulation of advanced Nuclear Power Plants. As part of its capabilities, it can solve the transport equations, Eqs. (3) to (5). The Finite Element Method (FEM) is utilized to solve the PDEs. FEM works by approximating a continuous solution over a region of interest as a combination of simple functions (e.g., polynomials) that are defined over a finite set of points. The solution is then

represented as a set of coefficients describing the simple functions' behavior over the entire region. SAM uses the continuous Galerkin FEM and Lagrange shape functions. This work uses first-order elements using piece-wise linear shape functions with the trapezoidal rule for numerical integration. The Jacobian-Free Newton Krylov method is used to solve the system of equations. The second-order backward difference formula is used for time integration. Additional details on the numerical scheme are provided in [13, §7].

2.3 PINN solution

PINNs are a type of ANN that can also solve PDEs. The modern PINN concept was initially introduced by Raissi et al. [9] and has since received significant adoption by the scientific machine-learning community – reviewed exhaustively in [10]. PINNs solve parameterized and nonlinear PDEs in the form,

$$\frac{\partial x}{\partial t} + \mathcal{N}(x;\lambda) = 0 \,, \tag{6}$$

where $x = x(z,t)$ is the solution, and $\mathcal{N}$ is a nonlinear operator parameterized by λ. Each transport equation, Eqs. (3) to (5) can be written in this form. A neural network is formulated to approximate the solution,

$$x(z,t) \approx \mathcal{F}(z,t;\theta),\ z \in \Omega,\ t \in \tau \,, \tag{7}$$

where $\mathcal{F}$ is a neural network that is parameterized by θ. The solution is box constrained in an n-dimensional space domain, Ω, and in a one-dimensional time domain, τ. The space domain specification would depend on the geometry of the transport path and whether multiple dimensions are needed. The specification of the time domain depends on the timescale at which the system reaches a steady state following changes in BCs. For instance, if the system settles slowly, a large τ is required, whereas a small τ is needed if the system reaches a steady state quickly.

In a purely data-driven approach, a series of measurements would be collected and stored in sets,

$$\mathcal{X}_m = \{(z_i,t_i)_{i=1}^N\} \,, \tag{8}$$

$$\mathcal{Y} = \{(x_i)_{i=1}^N\} \,, \tag{9}$$

where z_i is the position, t_i is the time, and x_i is the field value for the i^{th} data point. The set $\mathcal{X}_m$ contains positions and times, where subscript m denotes measurement data. The set $\mathcal{Y}$ contains measured field values. The superscript N is the dataset size. In a supervised learning setting, $\mathcal{X}_m$ is the input dataset, and $\mathcal{Y}$ is the output dataset. Formally, $x_i \in \mathbb{R}^m$, where m is the number of fields, $z_i \in \mathbb{R}^n \cap \Omega$ where n is the number of spatial dimensions, and $t_i \in \mathbb{R} \cap \tau$ is a scalar. A corresponding loss function is defined,

$$\mathcal{L}_m = f_{\text{loss}}\left(\mathcal{F}(\mathcal{X}_m), \mathcal{Y}\right), \tag{10}$$

where f_{loss} is a loss function (e.g., the mean squared error), and the $\mathcal{L}_m$ denotes the measurement loss. The first argument of the loss function is the predicted value, and the second argument is the target value. Thus, $\mathcal{L}_m$ quantifies the error (or loss) between predictions from the neural network and the measured field values. The loss is then backpropagated [14] to the neural network's parameters, θ. In the purely data-driven setting, the values of θ are strictly determined by the measured field values.

In a PINN setting, an additional source of information is added. The dynamics of the system being modeled are approximately known. For example, if a liquid is advected in a network of pipes, the transport equations can inform the neural network that, e.g., energy conservation must hold. Thus, the transport PDEs inform the neural network if there are inconsistencies in its solution. How is this achieved? First, the exact solution in Eq. (6) is replaced by that approximated by the neural network in Eq. (7),

$$\frac{\partial \mathcal{F}}{\partial t} + \mathcal{N}\left(\mathcal{F};\lambda\right) = 0 \,. \tag{11}$$

To evaluate the PDEs, e.g., Eqs. (3) to (5), the gradients of $\partial_t u, \partial_z u$, etc. are needed. In the numerical approach, the gradients are obtained by discretizing the space-time domain and approximating them using finite difference. The PINN approach uses a mesh-free method to evaluate the gradients. This is achieved through *forward mode* automatic differentiation, where the derivative of any output of $\mathcal{F}$ is obtainable with respect to any input. The physics-informed loss function is defined as,

$$\mathcal{L}_p = f_{\text{loss}}\left(\frac{\partial \mathcal{F}}{\partial t}(\mathcal{X}_p) + \mathcal{N}\left(\mathcal{F}(\mathcal{X}_p),\lambda\right), \emptyset\right) \,, \tag{12}$$

where $\emptyset$ is a zero set. The target output is always zero to enforce that the neural network satisfies the right-hand side of Eq. (11). It is important to emphasize that the target output is a zero set rather than measured field values. Thus, $\mathcal{L}_p$

can be evaluated at arbitrary inputs sampled from the space-time domain, $(z_p, t_p) \sim \Omega \times \tau$. In this way, the known physics of the system regularizes the neural network to the space-time domain outside of the measurement set, $\mathcal{X}_m$. The additional set of inputs is referred to as collocation points,

$$\mathcal{X}_p = \{(z_p, t_p)_{i=1}^{N}\} \, . \tag{13}$$

Now, during training, the parameters of the neural network, θ, can be updated to minimize both $\mathcal{L}_m$ and $\mathcal{L}_p$. Thus, knowledge gathered from measurements of the system is complemented with a priori knowledge of the physics of the system. The resulting neural network is defined as a PINN. Measurements of a facility require the installation of sensors (e.g., thermocouples and pressure transducers) and associated operation and maintenance costs. The amount of sensors and their locations is itself an optimization task [15, 16]. The PINN approach is particularly powerful when system measurements are sparse or physically precluded [17] (e.g., extreme temperature and pressure environments).

2.4 State-space formulation (PSM)

The objective of this work is to apply aspects of PINN to obtain a nonlinear dynamical model as an alternative to traditional linear state-space models. A discrete-time state-space model is represented by,

$$x_{k+1} = f\left(x_k, v_k\right) \, , \tag{14}$$

$$y_k = g\left(x_k, v_k\right) \, , \tag{15}$$

where x_k, v_k, and y_k are vectors representing states, inputs, and output measurements at time k. Additionally, f is the dynamics model, and g is the measurement model. Linear state-space models are commonly used in control theory because they provide a tractable and simplified approach to analyzing and designing control systems. A linear state-space model utilizes matrices to approximate the f and g functions,

$$x_{k+1} = Ax_k + Bv_k \, , \tag{16}$$

$$y_k = Cx_k + Dv_k \, , \tag{17}$$

where A and B matrices are the state and input matrices, and C and D are the output and control pass-through matrices. The linear state-space representation allows the application of powerful mathematical techniques such as eigenvalue analysis, frequency response analysis, and convex optimization algorithms (e.g., Quadratic Programs for Model Predictive Control). However, many physical systems exhibit nonlinear behavior, including in systems with time-varying parameters or when the system is operating in certain conditions (within steep saddle points). Thus, using linear models can lead to inaccuracies or instability in controlling a nonlinear system.

A nonlinear dynamics model could address the deficiencies of a linear representation. Specifically, a neural network could be used as the nonlinear function representing f and g in Eqs. (14) and (15). The previous section presented PINNs as an approach to solving PDEs, Eq. (11). If the PDEs are a good approximation of the dynamics of a physical system, the PINN will approximate the system's state in the form Eq. (1). However, several issues arise. First, the output of the PINN is a continuous representation of the system state $x = x(z, t)$. A discrete representation could be defined by,

$$x_k = \left[\mathcal{F}\left(z_0, t_k\right), \mathcal{F}\left(z_1, t_k\right), ..., \mathcal{F}\left(z_M, t_k\right)\right]^\intercal \, , \tag{18}$$

where $z_0, ..., z_M$ represent the physical locations at which the system's state is defined, and t_k is the discrete time. Without loss of generality, we can assume that $t_k = k\Delta t$, where k is an integer indexing time, and Δt is a constant time-step size. Therefore, the propagation of the state would be determined by,

$$x_0 = \left[\mathcal{F}\left(z_0, 0\right), \mathcal{F}\left(z_1, 0\right), ..., \mathcal{F}\left(z_M, 0\right)\right]^\intercal \, , \tag{19}$$

$$x_1 = \left[\mathcal{F}\left(z_0, t_1\right), \mathcal{F}\left(z_1, t_1\right), ..., \mathcal{F}\left(z_M, t_k\right)\right]^\intercal \, , \tag{20}$$

$$\vdots$$

$$x_N = \left[\mathcal{F}\left(z_0, t_N\right), \mathcal{F}\left(z_1, t_N\right), ..., \mathcal{F}\left(z_M, t_N\right)\right]^\intercal \, . \tag{21}$$

While useful for systems without any inputs, additional modifications are needed to accommodate systems with varying control inputs, i.e., Eq. (2),

$$x_k = \left[\mathcal{F}\left(z_0, t_k, v_k\right), \mathcal{F}\left(z_1, t_k, v_k\right), ..., \mathcal{F}\left(z_M, t_k, v_k\right)\right]^\intercal \, , \tag{22}$$

where v_k is a vector representing control inputs at step k. The control inputs can manipulate BCs, e.g., the temperature or velocity at a pipe inlet, or modify jump conditions between locations, e.g., pressure jump due to the presence of a pump. A problem with Eq. (22) is the assumption that the system's initial condition, $x(z, t = 0)$, must always be the same. For a dynamical system, an arbitrary final input value, $v_{k\to\infty}$, would result in a new settling point, $x_{k\to\infty}$. If we

want the state-space model to generalize, a form propagating forward from arbitrary x_0 is needed. Therefore, the initial condition itself should be an input for the PINN,

$$x_k = \left[\mathcal{F}\left(z_0, t_k, x_0, v_k\right), \mathcal{F}\left(z_1, t_k, x_0, v_k\right), ..., \mathcal{F}\left(z_M, t_k, x_0, v_k\right)\right]^\intercal \ , \tag{23}$$

where x_0 is the initial condition.

When a dynamical system is subject to an initial condition, the system's behavior in the short term is dominated by the initial state. However, as time passes, $k \gg 1$, the impact of the initial condition on the system's current state becomes progressively smaller. Therefore, a final modification to the formulation is needed,

$$x_k = \left[\mathcal{F}\left(z_0, t_k, x_{0,k}, v_k\right), \mathcal{F}\left(z_1, t_k, x_{0,k}, v_k\right), ..., \mathcal{F}\left(z_M, t_k, x_{0,k}, v_k\right)\right]^\intercal \ , \tag{24}$$

where x_{0_k} is the initial condition of the current time step. Before Eq. (22), it was assumed that $t_k = k\Delta t$. When this assumption is applied to Eq. (24), over a collection of multiple experiments, the value of t_k would arbitrarily correspond to combinations of the inputs, $(x_{0,k}, v_k)$, and outputs, x_k. From the perspective of training the neural net $\mathcal{F}$, there would be no relationship between x_k and t_k. However, t_k is still an important variable to estimate the time derivatives needed for physics informing. The timescale at which t_k does have a relationship between the inputs at outputs is the discrete time step, Δt. The data would be separated into an initial condition at $t_k = 0$, and the next step, $t_k = \Delta t$,

$$x_{0,k} = \left[\mathcal{F}\left(z_0, 0, x_{0,k}, v_k\right), \mathcal{F}\left(z_1, 0, x_{0,k}, v_k\right), ..., \mathcal{F}\left(z_M, 0, x_{0,k}, v_k\right)\right]^\intercal, \text{ and,} \tag{25}$$

$$x_k = \left[\mathcal{F}\left(z_0, \Delta t, x_{0,k}, v_k\right), \mathcal{F}\left(z_1, \Delta t, x_{0,k}, v_k\right), ..., \mathcal{F}\left(z_M, \Delta t, x_{0,k}, v_k\right)\right]^\intercal \ . \tag{26}$$

The relationships between the inputs, $(x_{0,k}, v_k)$, and outputs, x_k, in Eq. (24) is visualized in Fig. 1.

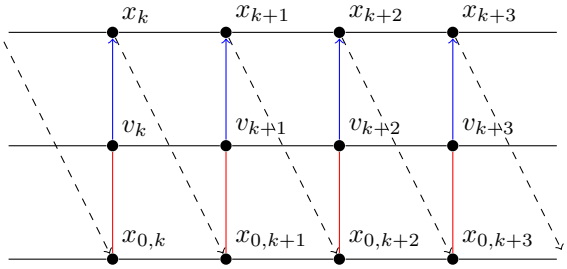


Figure 1: A visualization of the nomenclature and their relationships. The red lines indicate variables needed at each step k, i.e., the initial condition and input, $x_{0,k}$, and v_k. The blue arrows indicate the prediction of the next state, x_k, by the numerical solver or PSM. The dashed arrows indicate the aliasing of x_k as $x_{0,k+1}$.

The resulting neural network is,

$$x\left(z_i, t_i, v_i, x_{0,i}\right) = \mathcal{F}\left(z_i, t_i, v_i, x_{0,i}\right) \ , \tag{27}$$

predicting each component of Eqs. (25) and (26). To train this neural network, the datasets in Eqs. (8) and (9) are modified,

$$\mathcal{X} = \{(z_i, t_i, v_i, x_{0,i})_{i=1}^N\} \ , \tag{28}$$

$$\mathcal{Y} = \{x(z_i, t_i, v_i, x_{0,i})_{i=1}^N\} \ , \tag{29}$$

where v_i is the control input, and $x_{0,i}$ is the initial condition. Formally, $v_i \in \mathbb{R}^p$ where p is the number of control inputs, and $x_{0,i} \in \mathbb{R}^q$ where q is the number of states modeled. Using this formulation, the input space of the neural network increases by $\mathcal{O}(10^{p+q})$. However, this step is essential to allow network usage without relying on a heuristic approach, e.g., a tabular database of multiple models where output is obtained by interpolating between $(v_i, x_{0,i})$. Besides, this would circumvent a desire to show that the network can generalize well. The trained network is referred to as a PSM.

2.5 PSM architecture and training methodology

To train a PSM, an ANN must be constructed. The ANN will represent the operator, $\mathcal{F}$, in Eqs. (10), (12) and (24). In past research, various PINN architectures have been explored, including Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) [10]. For this study, we adopt an MLP architecture. The intuition behind this choice is explained by contrasting an MLP's properties against other architectures. Transport system dynamics are inherently Markovian, meaning that the state x_{k+1} depends solely on the previous state, x_k, rather than a sequence of past states. This characteristic makes a state-less architecture, such as an MLP, more suitable than an RNN for this application. Additionally, CNNs are typically employed for tasks involving image or signal processing,

where the input exhibits a grid-like structure and exhibits strong spatiotemporal correlations, such as multi-dimensional data. However, in this study, the PSMs are designed to model one-dimensional transport, making an MLP architecture adequate for the task.

Each i^{th} layer of the MLP provides an output,

$$y_j^i = \sigma^i \left(w_{jk}^i y_k^{i-1} + b_j^i \right) \ , \tag{30}$$

where w_{jk}^i is the weight matrix, b_j^i is a bias vector, and σ^i is the activation function of the i^{th} layer. The vectors y_j^i and y_k^{i-1} are outputs of the i^{th} and $(i-1)^{\text{th}}$ layers. The sizes of the matrices and vectors (denoted by $j,\ k$) will depend on the layer-wise size of y. In this work, the ANN is built and trained with `PyTorch` [18]. The architecture of the neural network and the training workflow is presented in Fig. 2. The current configuration of the neural network is described next. The activation function between all intermediate layers is the hyperbolic tangent. It was chosen because it is nonlinear, is smooth, and has smooth derivatives. After the input layer, there are three "head" layers of size s_H, followed by an "intermediate" layer of size s_I. The output of the intermediate layer is sent to three separate "tail" layers of size s_T. The layer sizes used are $(s_H, s_I, s_T) = (200, 100, 100)$. There are three tails for each field variable: pressure, velocity, and temperature. The intuition behind this architecture is that the head layers encode the initial condition and control input into a form used by the tail layers. The tail layers then decode the information separately for each field variable.

As discussed previously, two data sets are used to train the PSM. Measurement data from locations $z_i \in \mathcal{X}_m$, Eq. (8), and physics-informing from collocation locations $z_i \in \mathcal{X}_p$, Eq. (13). In the context of transport systems, the measurement locations are those where physical measurements of fluid flow exist, e.g., thermocouples or pressure transducers. On the other hand, collocation locations are arbitrarily chosen across the entire physical domain.

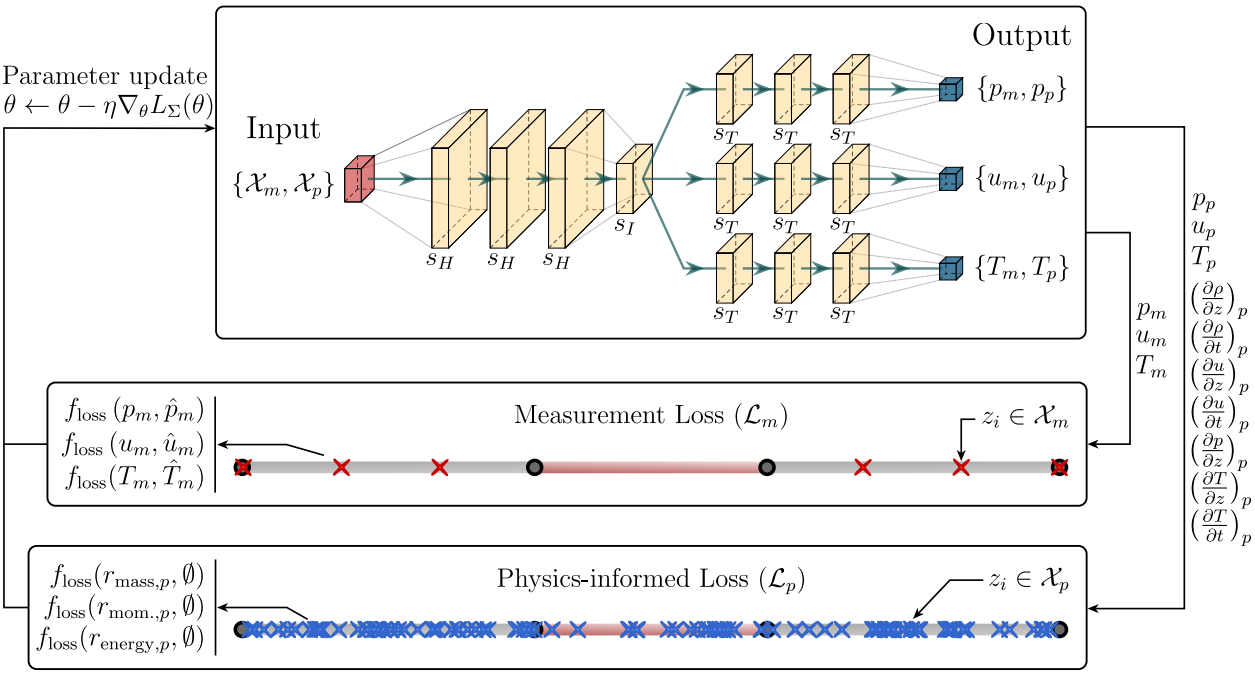


Figure 2: PSM architecture and training workflow. Top: The neural network architecture displaying relative sizes of the MLP layers. Bottom: Measurement and physics-informed losses use output from $\mathcal{F}(\mathcal{X}_m)$ and $\mathcal{F}(\mathcal{X}_p)$, respectively. The combined losses are used to update the parameters of the neural network, θ.

2.5.1 Loss formulation

A discussion is presented to explain the differences in the "Measurement Loss", Eq. (10), and "Physics-informed Loss", Eq. (12), in Fig. 2. In conventional PINN formulation, the objective is to solve for the spatiotemporal distribution of field(s) within the boundaries of a domain under the assumptions of a fixed initial condition and BC(s). In general, it is anticipated that there are no measurements inside the spatial domain. However, in transport systems, the direct measurements of pressure, temperature, and indirect measurement of velocity, are available through sensors (the type of sensor is dictated by the application, accuracy needed, and fluid properties) at multiple locations along the fluid

transport path. Thus, in the PSM formulation, the spatiotemporal distribution of field(s) *between* the measurement locations must be solved. In this setting, the measurement losses enforce initial conditions in Eq. (25) but also act as BCs in Eq. (26). On the other hand, the physics-informed losses are calculated at random locations. The losses from these locations tune the coefficients of the PSM such that the PDE equations defined in Eq. (11) are satisfied. To tune the coefficients of the PSM, it is important to choose an appropriate loss function.

To train the network, the loss function, f_{loss} in Eqs. (10) and (12), must be defined. The loss function quantifies the discrepancy between a target value and a model's prediction. Regression problems in machine learning commonly use the Mean Squared Error (MSE) as the loss function. However, the Log-Cosh loss function, Eq. (31), is utilized in this work. In both equations, y_i is the predicted value, $\hat{y}_i$ is the target, and N is the total number of samples in a batch.

$$f_{\text{loss}}\left(y_i, \hat{y}_i\right) = \frac{1}{N}\sum_{i=1}^{N} \log\left(\cosh\left(y_i - \hat{y}_i\right)\right) \ , \tag{31}$$

The Log-Cosh loss function provides benefits over MSE by being less sensitive to outliers, having bounded gradients for numerical stability, ensuring more reliable convergence in optimization, and performing better with heavy-tailed noise distributions, allowing for quicker, more effective learning and accurate real-world data fitting [19].

Now that the definitions of the measurement loss, the physics-informed loss, and the loss function have been presented, a discussion on the weighting of the losses follows. The total loss forwarded to the optimizer per epoch is defined as,

$$\mathcal{L}_\Sigma = \alpha\mathcal{L}_m + \beta\mathcal{L}_p \ \Big| \ \alpha + \beta = 1 \ , \tag{32}$$

where α and β are coefficients that weigh the importance of the measurement and physics-informed losses, respectively. The correct settings for (α, β) will depend on the degree of certainty in the measurements (assessable from the manufactured accuracy of sensors) and the confidence in the suitability of the PDEs' representation of the system's dynamics. In this work, during training of the PSMs, we set $\alpha = 0.5$ and $\beta = 0.5$, equally weighting both losses. However, further consideration is warranted when applying PSMs to a physical system.

2.5.2 Scaling

Scaling the input and output of a neural network is crucial when processing physical measurements of different units, scales, and ranges, such as pressure, temperature, and velocity. This process facilitates the optimization of the neural network, enabling the model to converge more efficiently and quickly by ensuring that the gradients have a consistent magnitude across all input dimensions [20]. Unscaled inputs with disparate magnitudes can lead to elongated, narrow contours in the loss function landscape, resulting in slow and unstable convergence. In this work, min-max scaling is used,

$$z^* = z/z_{\max} \ , \tag{33}$$
$$t^* = t/t_{\max} \ , \tag{34}$$
$$p^* = (p - p_{\min})/(p_{\max} - p_{\min}) \ , \tag{35}$$
$$u^* = (u - u_{\min})/(u_{\max} - u_{\min}) \ , \tag{36}$$
$$T^* = (T - T_{\min})/(T_{\max} - T_{\min}) \ , \tag{37}$$
$$\rho^* = (\rho - \rho_{\min})/(\rho_{\max} - \rho_{\min}) \ , \tag{38}$$

for transforming position, time, pressure, velocity, temperature, and density, respectively. The value for $z_{\max}$ equals the maximum path length for the liquid. The value for $t_{\max}$ equals the discrete time-step between measurements, Δt in Eq. (26). The minimum and maximum values for the pressure, velocity, temperature, and density are based on the ranges of data that are collected from experiments.

2.5.3 Training procedure

The PSM training procedure is listed in Alg. 1. During each epoch, the weights are updated from a fixed measurement dataset. Whereas the collocation points are updated at each epoch. A step learning rate scheduler was used, with a rate decay of 0.5 every 50 epochs. All models were trained for 500 epochs with a batch size of 2048. These settings attempt to stabilize the training of the neural network and to decrease the components of $\mathcal{L}_\Sigma$ by 3-5 orders of magnitude.

Algorithm 1 Training a PSM

Input: Measurement data $(\mathcal{X}_m, \mathcal{Y})$, Governing equations G, Loss coefficients (α, β), Number of epochs N, Batch size B, Learning rate scheduler $\mathcal{S}(\cdot)$
Initialize: Network $\mathcal{F}$ with random weights θ
for $n = 1, 2, \ldots, N$ **do**
  Update learning rate $\eta \leftarrow \mathcal{S}(n)$
  for each mini-batch $(\mathcal{X}_{m,i}, \mathcal{Y}_i) \subset (\mathcal{X}_m, \mathcal{Y})$ with size B **do**
    Compute predictions $\hat{\mathcal{Y}}_i = \mathcal{F}(\mathcal{X}_{m,i})$
    Compute measurement loss $\mathcal{L}_m(\hat{\mathcal{Y}}_i, \mathcal{Y}_i)$ using Eq. (10)
    Randomly sample B collocation points $\mathcal{X}_{p,i}$, and extract $(x_{0,i}, v_i)$ from $\mathcal{X}_{m,i}$
    Compute the residual values $\mathcal{R}_i = G(\mathcal{F}, \mathcal{X}_{p,i}, x_{0,i}, v_i)$
    Compute physics-informed loss $\mathcal{L}_p(\mathcal{R}_i, \emptyset)$ using Eq. (12)
    Compute total loss $\mathcal{L}_\Sigma = \alpha\mathcal{L}_m + \beta\mathcal{L}_p$
    Update network weights using optimizer: $\theta \leftarrow \theta - \eta\nabla_\theta L_\Sigma(\theta)$
  end for
end for
return Trained PSM model $\mathcal{F}$

3 Results: applying PSMs for transport in a Heated Channel

Two experiments are designed to investigate PSMs' performance for transport phenomena. The first experiment is a system of three conduits in series (referred to as pipes). The first and last pipes will be adiabatic and $1.0\,\mathrm{m}$ long. The middle pipe will be heated with a constant volumetric heat source of $q''' = 50.0\,\mathrm{MW/m^3}$, and $0.8\,\mathrm{m}$ long. All pipes have a total cross sectional area of $0.449\,\mathrm{m^2}$ and a hydraulic diameter of $2.972\,\mathrm{mm}$. No solids are modeled – only the internal flow area. Each pipe is discretized into ten elements. The inlet of the first pipe will have varying velocity and temperature BCs,

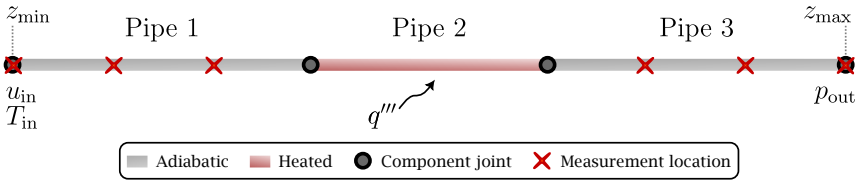
$$u(z = z_{\min}, t) = u_{\text{in}}(t)\ , \tag{39}$$

$$T(z = z_{\min}, t) = T_{\text{in}}(t)\ . \tag{40}$$

The trajectory of the BCs is defined by linear ramps that vary in ramp rate, maximum/minimum amplitude, and rest time between manipulations. These parameters are varied to encourage regularization of the PSM. Additionally, the outlet of the third pipe will be a constant pressure BC,

$$p(z = z_{\max}, t) = p_{\text{out}}. \tag{41}$$

In this work, the pressure BCs are presented as absolute pressure; however, the pressure field is presented as gage pressure. The experiment configuration is visualized in Fig. 3. A visualization of the evolution of the field variables is presented in Fig. 4. The PSM will be tasked with predicting these spatiotemporal distributions by learning from a few measurement locations and the approximate PDEs of the system. Hereafter, this experiment will be referred to as the "heated channel" configuration.



p_{out}	$135.6\,\mathrm{kPa}$
$u_{\text{in,min}}$	$0.549\,\mathrm{m/s}$
$u_{\text{in,max}}$	$0.749\,\mathrm{m/s}$
$T_{\text{in,min}}$	$531.5\,^\circ\mathrm{C}$
$T_{\text{in,max}}$	$611.5\,^\circ\mathrm{C}$

Figure 3: Configuration of the pipes in series. Left: Layout of the pipes and identifications of the locations of imposed BCs and heat source term. Right: A list of the fixed outlet pressure BC, and ranges for the inlet BCs.

To close the transport equations, Eqs. (3) to (5), the thermophysical properties of the fluid must be defined. This work utilizes a molten salt, $\mathrm{LiF{-}BeF_2}$, known colloquially as 'flibe.' The reason is that flibe is the leading candidate for the primary working fluid of next-generation molten salt reactor designs currently under development [21]. The density and specific heat capacity of flibe are defined by [22],

$$\rho = 2413 - 0.488T\ [\mathrm{kg/m^3}]\ , \tag{42}$$

$$C_p = 2414\ [\mathrm{J/(kg\,K)}]\ , \tag{43}$$

where T is the fluid temperature in Kelvin. These correlations are adopted to calculate the physics-informed loss in Eq. (12).

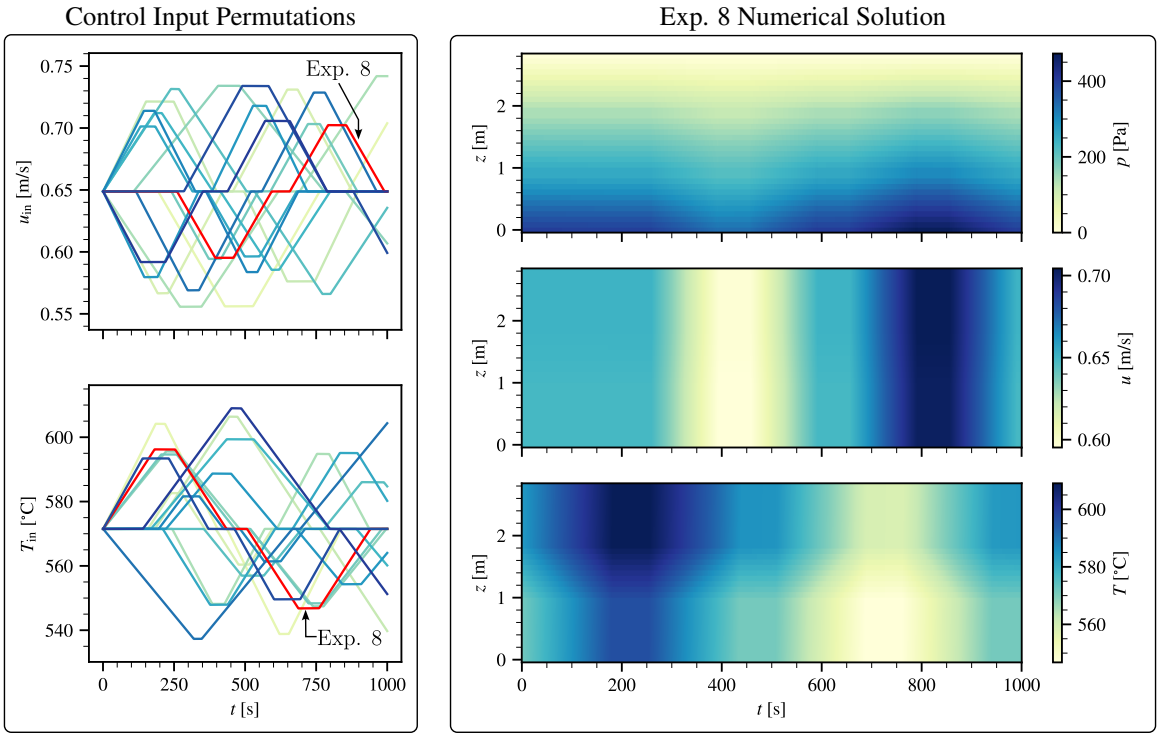


Figure 4: An overview of the training dataset for the heated channel configuration. Left: Velocity and temperature BCs at the inlet that are manipulated in time. The experiment chosen to display the solution is annotated ("Exp. 8"). Right: Numerical solution of all field variables over the entire spatiotemporal domain.

3.1 Evaluation with test data

The results for the heated channel are discussed in this section. A sample of the control input permutations and numerical solution of the field variables were presented in Fig. 4. In this setting, the PSM must accurately propagate the system's state, x_k, from a given initial condition, $x_{0,k=0}$, and input trajectory, $v_{k=0\dots N}$. This problem is a classical advection problem where the inlet boundary conditions are transported through the pipe incur: pressure losses due to friction from the pipe, modifications in mass flow rate (velocity) due to boundary conditions, and changes in temperature due to energy deposition.

In Fig. 5, a rollout of a single input, selected from the *test* dataset, i.e., data that is not used to train the models, is presented. Subplots A and B show the input trajectory of the velocity and temperature at the inlet, u_{in} and T_{in}. Subplots C-K show pressure, velocity, and temperature at fixed positions vs. time. Subplots L-T show the fields at fixed times vs. position. In subplots C-T, results from the numerical solver ("Num. Sol."), the PSM model, and an Artificial Neural Network (ANN) model are displayed. The PSM model is trained with Eq. (32), which contains both measurement and physics-informed loss, Eqs. (10) and (12), respectively. Whereas the ANN model uses the same neural network architecture as the PSM but is trained only with the measurement loss, Eq. (10). Thus, the ANN results represent the classical, purely data-driven approach to modeling. In contrast, the PSM results combine knowledge from data with a priori knowledge of the system's dynamics in the form of transport PDEs.

The ANN appears to perform on par with the PSM. Comparing the pressure and velocity spatiotemporal predictions, indeed, the ANN does perform as well as the PSM. However, comparing the temperature distributions, there is a significant error introduced by the ANN. The errors occur at $1.0 \leq z \leq 1.8\,\text{m}$, i.e., the heated pipe's location. This observation is more explicitly visualized in Fig. 6, which compares spatiotemporal error in predicting temperature averaged over all test datasets. The PSM significantly outperforms the ANN in the prediction of the temperature field.

Why does the ANN fail to accurately predict the temperature field only? This question can be answered by understanding the underlying physics and the sensor measurement locations.

The advection of pressure and velocity depends on the frictional losses and the inlet mass flow rate. Because the friction factor and pipe geometry are constant, we expect a constant pressure gradient with respect to position. This is displayed

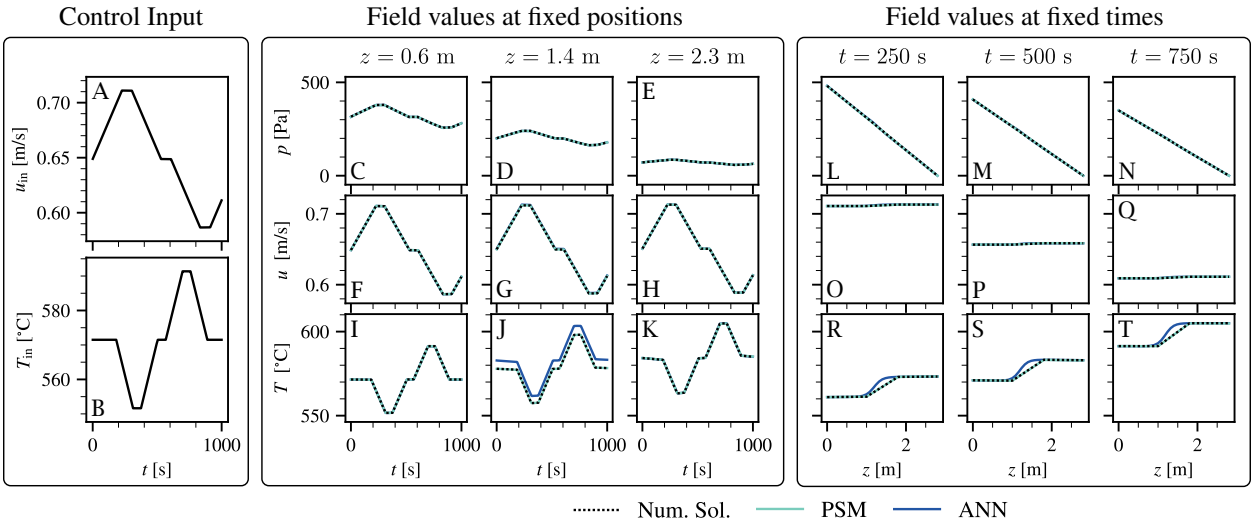


Figure 5: Contrasting the performance of PSM and ANN models in predicting the evolution of pressure, velocity, and temperature for the heated channel. Subplots A-B: control input setting from the test dataset. Subplots C-K: field temporal evolution at fixed positions. Subplots L-T: field spatial distribution at fixed times.

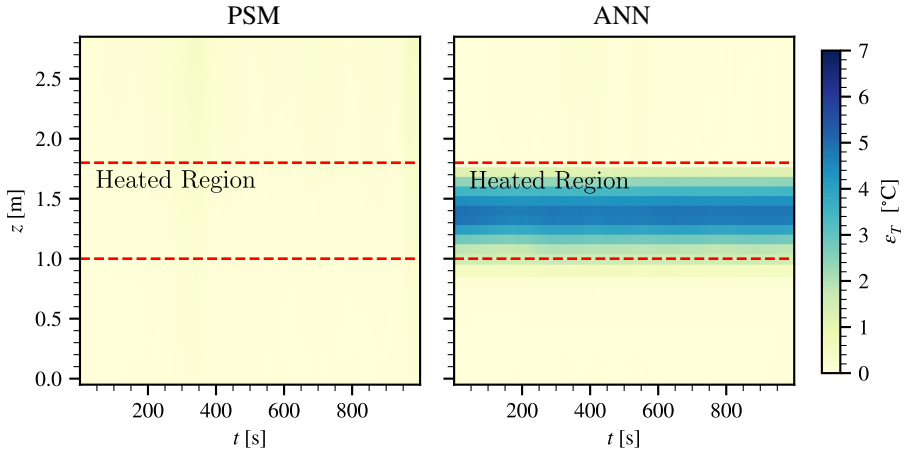


Figure 6: Comparison of root-mean-square error (RMSE) in the prediction of temperature for PSM (left) and ANN (right) models, with a unified colorbar scale applied. The RMSE values are averaged over all test datasets. The heated region is present between the dashed red lines.

in Subplots L-N in Fig. 5. The measurement of the pressure and velocity fields occurs in the first and last unheated pipes. Therefore, the ANN has to simply interpolate between the measurement locations of pressure before and after the heated section to correctly estimate the pressure field and be able to modify the gradient according to the change in inlet velocity mass flow rate. The latter is provided as an input (v_k) to the ANN. The results show that the ANN model can accomplish these tasks well for the pressure field.

For the velocity field, we expect the spatial distribution to be nearly constant, except after the heated region. There is a change in the density because of the heating (Eq. (42)), which causes the velocity to change in order to conserve mass (Eq. (3)). This causes a small error in ANN prediction, noticeable upon close inspection of Subplots O-Q in Fig. 5. Therefore, the performance for velocity is still acceptable.

However, for the temperature field, there is a large discrepancy between ANN and PSM performance within the heated region. The temperature field is affected by energy deposition (q''' in Eq. (5)), and inlet BCs. When $q''' = 0$, there is no gradient in temperature. When $q''' = c$, a constant, there is a linear gradient in temperature. The PSM correctly predicted both situations, as shown in Subplots I-K and R-T in Fig. 5. However, the ANN has no knowledge of q''', and thus it interpolates between the sensor readings. Therefore, the ANN model can only correctly predict temperature

distributions in regions where $q''' = 0$, i.e., there is no temperature gradient and any modification is provided directly by the inlet BC.

3.2 Training with noisy measurements

The data from sensor measurements in a physical plant are noisy. The noise stems from inherent randomness during fluid transport & signal measurement itself. It is important to quantify the performance of PSMs in such environments. The introduction of homoscedastic and heteroscedastic noise is studied. The former is introduced by,

$$x_{\text{noisy}} = x + \sigma \cdot \epsilon \,, \tag{44}$$

where x_{noisy} is the noisy signal, x is the noise-free signal, σ is the noise standard deviation, and ϵ is a random noise vector sampled from a standard normal distribution. This type of noise model scenarios where the variance is constant across all magnitudes. Heteroscedastic noise is introduced by,

$$x_{\text{noisy}} = x + \sqrt{|x| \cdot \xi} \cdot \epsilon \,, \tag{45}$$

where ξ is the variance factor. The variance factor is a parameter that controls the relationship between the signal's magnitude and the added noise's variance. Thus, this type of noise model scenarios where the noise variance is not constant but depends on the underlying signal's magnitude. The noisy signals are visualized in Fig. 7. During training, the signals are manipulated randomly for each batch. Noise is added to the sensor measurements and the initial conditions (x_k and $x_{0,k}$, respectively, in Eq. (26)). It is assumed that the system's actuation is not noisy (i.e., the control signals, v_k are noise-free).

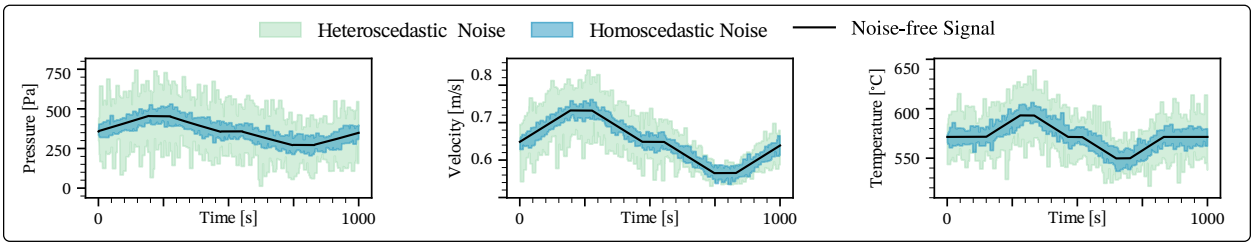


Figure 7: Example of the manipulation of sensor measurements for all fields. During training, the signals are manipulated randomly for each batch.

The results from introducing noise are discussed next. The training procedure remains the same, except that artificial noise is added during each batch. In Table 1, the results are tabulated for both types of noise, and varying magnitudes of noise factors (σ in Eq. (44), ξ in Eq. (45)). First, inspecting the mean RMSE values, the PSM has a significant advantage in predicting velocity and temperature fields. There is a lower advantage for the pressure fields. This might be because the nominal RMSE values for pressure are already very low when contrasted with the pressure range (ΔP) across all experiments. Inspecting the maximum RMSE values, the PSM retains a significant advantage in predicting velocity and temperature fields, with some anomalies noted for $\sigma = 0.05$ and $\xi = 0.05$. The PSM's advantage for the pressure field is relatively moderate, likely because it has a simple spatiotemporal relationship in the current experimental setting. To summarize, the results show that even when noise is added to the signals, PSMs have a significant advantage over a purely data-driven model.

3.3 PSM Capability: Model-based Control

In the previous section, the PSM architecture was shown to accurately approximate the spatiotemporal distribution of all the fields. Because the PSM is an end-to-end differentiable model, we can calculate the gradients for inputs and outputs using auto differentiation. In this section, the capability of a PSM to enable supervisory control of a nonlinear system is demonstrated. The Reference Governor (RG) supervisory control scheme is adopted to demonstrate this capability. The RG is a model-based scheme that enforces pointwise-in-time constraints on a system by modifying the reference input setpoints [23].

The RG algorithm is summarized next. At each time-step $k \in \mathbb{Z}_+$, the RG receives a reference input, $r_k \in \mathbb{R}^m$, where m is the number of inputs. Depending on the system's current state, expected evolution, and imposed constraints, an admissible input $v_k \in \mathbb{R}^m$,

$$v_k = v_{k-1} + \kappa_k \left(r_k - v_{k-1} \right) \,, \tag{46}$$

	$\varepsilon_{P,\text{ANN}}[\text{Pa}]$	$\varepsilon_{u,\text{ANN}}[\text{mm/s}]$	$\varepsilon_{T,\text{ANN}}[^\circ\text{C}]$	$\frac{\varepsilon_{P,\text{PSM}}}{\varepsilon_{P,\text{ANN}}}[\%]$	$\frac{\varepsilon_{u,\text{PSM}}}{\varepsilon_{u,\text{ANN}}}[\%]$	$\frac{\varepsilon_{T,\text{PSM}}}{\varepsilon_{T,\text{ANN}}}[\%]$
Mean						
Noise-free	0.51	0.17	1.01	58	52	6
Homos. $\sigma = 0.01$	1.25	0.25	1.15	12	25	3
Homos. $\sigma = 0.05$	0.46	0.22	0.98	143	35	17
Homos. $\sigma = 0.10$	0.83	0.23	1.01	91	51	28
Heteros. $\xi = 0.01$	0.71	0.25	1.15	94	41	24
Heteros. $\xi = 0.05$	0.94	0.27	1.10	94	68	27
Heteros. $\xi = 0.10$	1.07	0.34	0.97	89	63	33
Maximum						
Noise-free	4.82	1.25	5.81	113	52	31
Homos. $\sigma = 0.01$	8.97	1.24	6.08	10	43	13
Homos. $\sigma = 0.05$	6.17	1.24	7.00	53	104	14
Homos. $\sigma = 0.10$	6.49	2.38	6.87	78	35	27
Heteros. $\xi = 0.01$	4.80	1.34	7.70	81	51	19
Heteros. $\xi = 0.05$	5.47	1.42	6.54	95	120	35
Heteros. $\xi = 0.10$	6.31	1.97	5.46	76	69	38
Range across all experiments: $\Delta P = 531\,\text{Pa}$, $\Delta u = 200.0\,\text{mm/s}$, $\Delta T = 92.6\,^\circ\text{C}$						

Table 1: Comparison of the mean and maximum RMSE for field variables using the ANN across all test datasets. The relative RMSE error for PSMs is displayed in the last three columns: the values are color-coded to highlight significant performance improvement when below 75 % and a degradation in performance if above 100 %. The range of field values over all experiments is displayed at the bottom.

is obtained. The admissible input is then sent to lower-level controllers of the system (e.g., a PID controller). In the Scalar RG (SRG) formulation, $\kappa_k \in [0, 1]$ is a *scalar* that governs admissible changes to the inputs, such that a complete rejection, $v_k = v_{k-1}$, acceptance, $v_k = r_k$, or an intermediate change, $v_{k-1} \leq v_k \leq r_k$, is possible.

Next, how the SRG algorithm determines κ_k is summarized. The RG uses a linear state-space representation, Eqs. (16) and (17), and then imposes constraints on the output variables, $y_k \in Y$, where a set of linear inequalities defines Y. Then, a construct referred to as the Maximal Output Admissible Set (O_∞) is defined – it is the set of all x_k, and constant input $\tilde{v}$, such that,

$$O_\infty = (x_k, \tilde{v}) : y_{t+k} \in Y, v_{t+k} = \tilde{v}, \forall k \in \mathbb{Z}^+ \,. \tag{47}$$

The $k \to \infty$ assumption of O_∞ is generally relaxed to a suitably large finite horizon, T. Thus, at each time-step k, the SRG algorithm determines the admissible $\tilde{v}$, and therefore κ_k, such that the system remains in O_∞. The O_∞ is constructed by a roll-out of Eqs. (16) and (17), detailed further in [23, §2]. For a fixed set of state-space matrices and constraints, the matrices needed to construct O_∞ can be precomputed and stored for each RG evaluation.

A drawback of the SRG formulation is that κ_k is a scalar. If $v_k - r_k$ is multi-dimensional and $\kappa_k < 1$, input movement in all dimensions is bounded. In this work, we utilize the Command Governor (CG), a variant of the RG which selects v_k by solving the quadratic program,

$$\begin{aligned} v_k = &\underset{v_k}{\text{argmin}} \, ||v_k - r_k||_Q^2 \,, \\ &\text{s.t.} \;\; (x_k, v_k = \tilde{v}) \in O_\infty \,, \end{aligned} \tag{48}$$

where Q is a positive definite matrix signifying the relative importance of each v_k component. The quadratic program is solved by the `CVXPY` library [24]. Using the CG, constraints can be enforced by manipulating v_k arbitrarily in $\mathbb{R}^m$, constrained by O_∞.

To utilize the CG, a linear representation of the system is needed. This work assumes full-state feedback, $C = \mathbb{I}$, and no control pass-through, $D = \mathbf{0}$. Because the PSM is end-to-end differentiable, we can approximate the nonlinear model, Eq. (14), as a linear representation using Jacobian linearization,

$$x_{k+1} \approx x_{00} + A\delta x_k + B\delta v_k \,, \tag{49}$$

$$\delta x_k = x_k - x_{00} \,, \tag{50}$$

$$\delta v_k = v_k - v_{00} \,, \tag{51}$$

where x_{00} and v_{00} are state and input values at the linearization point. The matrices A, and B are approximated by the Jacobians of $\mathcal{F}$, Eq. (27), at the linearization point,

$$A = \left[\frac{\partial \mathcal{F}_{x_{k+1}}}{\partial x_{k,0}}, \frac{\partial \mathcal{F}_{x_{k+1}}}{\partial x_{k,1}}, ..., \frac{\partial \mathcal{F}_{x_{k+1}}}{\partial x_{k,N}} \right]^{\intercal}_{(x_{00}, v_{00})}, \tag{52}$$

$$B = \left[\frac{\partial \mathcal{F}_{x_{k+1}}}{\partial v_{k,0}}, \frac{\partial \mathcal{F}_{x_{k+1}}}{\partial v_{k,1}}, ..., \frac{\partial \mathcal{F}_{x_{k+1}}}{\partial v_{k,M}} \right]^{\intercal}_{(x_{00}, v_{00})}, \tag{53}$$

where N and M are the number of states and inputs. In summary, the RG was introduced as the framework to enforce constraints, followed by a specific version, the CG, which can manipulate multiple inputs to enforce constraints on a system. Then, the approach to utilize the PSM to approximate the linear state-space matrices required by the CG was introduced. The algorithm to utilize this procedure for supervisory control is listed in Alg. 2.

Algorithm 2 Sequential Neural Command Governor (NCG) using PSMs

Input: PSM model $\mathcal{F}$, Requested set point trajectory r, Time-dependent constraints on states Y_k, Input matrix Q, Matrices A and B update interval γ
Initialize: Initial condition of system x_0, Iteration counter $i \leftarrow 0$
for $k = 1$ to length(r) **do**
    if $k \mod \gamma = 0$ **then**
        Obtain matrices A and B by Jacobian Linearization of $\mathcal{F}$ at state x_k, v_k
        Update the O_∞ matrices with the current linear state-space model
    end if
    if Constraint update **then**
        Update the O_∞ matrices with the current Y_k
    end if
    Formulate the quadratic program in Eq. (48) with inputs r_k, x_k
    Solve the quadratic program to obtain the admissible set point v_k
    Return Admissible set point v_k to controlled system
    Update state of system $x_k \leftarrow x_{k+1}$
end for

Next, results demonstrating Alg. 2 are presented. In Fig. 8, an input permutation from the test dataset is chosen randomly to manipulate the inlet BCs. The variation of the temperature BC, T_{in}, causes a significant temperature rise after the heated channel, observable by comparing subplots I-K. It is desirable to constrain excessive increases in temperature to avoid damage to equipment or mitigate corrosion. Thus, a temperature constraint is assigned at a location after the heated channel, $z = 2.3\,\text{m}$, shown in subplot K. The NCG algorithm admits input changes to enforce these constraints, shown in subplots A and B. The results demonstrate that the approach is successful in meeting the enforced constraints. There are a few key takeaways. First, the NCG algorithm independently manipulates each input value, minimizing the distance from the requested input trajectories, i.e., Eq. (48). Second, a time-dependent constraint is demonstrated (annotated in subplot K). Accommodating time-dependent constraints enables flexibility during operation – it is envisioned that, as part of a comprehensive autonomous operation framework, an external algorithm could update the constraints enforced on a system depending on its health condition.

It is important to note that in this demonstration, the PSM is only used to update the A and B matrices – the numerical solver is used as the environment with which the NCG algorithm interacts. The accuracy of Jacobean linearization is generally high near the linearization point, (x_{00}, v_{00}), but can decrease as the system moves away from this point. It is most effective for systems that exhibit mild nonlinearity around the operating point. Therefore, when implemented in practice, the matrix update interval, γ in Alg. 2, must be chosen carefully. Lastly, a model-based control scheme may have additional stability assumptions. For the RG, it is assumed that the system being controlled is asymptotically stable, i.e., the matrix A is Schur [23]. This can be verified through eigenvalue analysis of the linearized matrix.

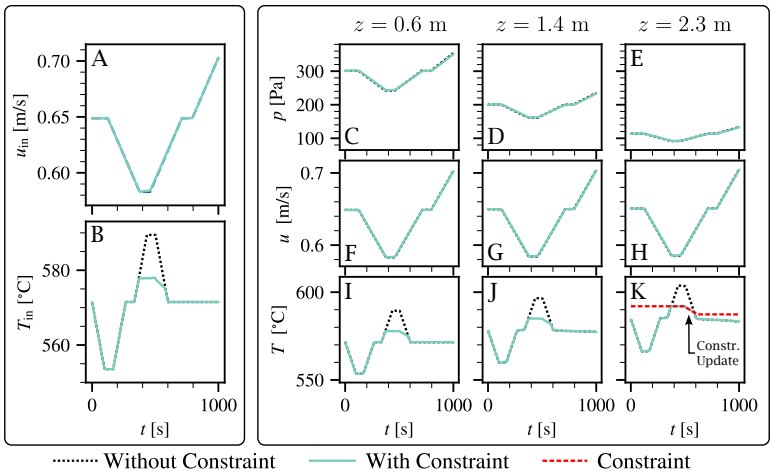


Figure 8: Demonstration of the NCG algorithm. The PSM instantiates the models needed to utilize the supervisory control algorithm. The numerical solver is used as the environment with which the NCG algorithm interacts. Subplots A-B: Control input permutations. The input value without constraints is the reference input trajectories for the NCG algorithm. Subplots C-K: Field values at fixed locations. Subplot K: A constraint is assigned at this location and manipulated as a function of time.

4 Results: applying PSMs for transport in a Cooling System

In addition to the heated channel configuration, a "cooling system" configuration was adopted. A series of six pipes are arranged in a loop. All pipes have a total cross sectional area of $0.449\,\mathrm{m}^2$ and a hydraulic diameter of $2.972\,\mathrm{mm}$. No solids are modeled – only the internal flow area. All pipes are $1.0\,\mathrm{m}$ in length, except for the pipes immediately after the heated and cooled sections, which are $2.0\,\mathrm{m}$ long. The pipes are discretized into 10 elements per meter. A pump provides the head necessary to drive the fluid flow. In addition to a volumetric heat source, a volumetric heat sink is also present. This configuration is commonly found in advanced cooling systems, such as those used in high-performance electronic devices, nuclear reactors, and aerospace applications. The loop configuration is visualized in Fig. 9. There are two independent control inputs,

$$q'''_{\mathrm{in}} = q'''_{\mathrm{in}}(t)\,, \tag{54}$$

$$q'''_{\mathrm{out}} = -q'''_{\mathrm{in}}(t)\,, \tag{55}$$

$$\Delta p_{\mathrm{pump}} = \Delta p_{\mathrm{pump}}(t)\,, \tag{56}$$

where q'''_{in} is the volumetric heat generation rate, q'''_{out} is the volumetric cooling rate, and Δp_{pump} is the pressure head provided by the pump. A control strategy is adopted that synchronizes the heating and cooling rates to increase/decrease the system's heat generation/cooling capability while maintaining constant temperatures throughout the loop. Both independent control inputs $(q'''_{\mathrm{in}}(t), \Delta p_{\mathrm{pump}}(t))$ are a function of time and will be defined by linear ramps that vary in ramp rate, maximum/minimum amplitude, and rest time between manipulations. Additionally, there is one boundary condition,

$$p(z = z_{\mathrm{set}}, t) = p_{\mathrm{loop}}\,, \tag{57}$$

which is a static pressure boundary that defines the relative value of the pressure in the loop.

Unlike the heated channel configuration, the loop configuration is more challenging from a dynamics modeling perspective. First, there is no time-dependent manipulation of the field variables via boundary conditions. The manipulation of the pump head and heat source indirectly impact the field variables. Their evolution will have a spatiotemporal effect dictated by the transport equations, Eqs. (3) to (5). A visualization of the evolution of field variables is presented in Fig. 10. The pump head trajectory strongly impacts the velocity of the fluid. Due to the control strategy, the temperature of the fluid remains approximately constant with respect to time but has a spatial variation.

4.1 Evaluation with test data

The performance of the PSM and ANN models for the cooling system is discussed next. A sample of the control input permutations and the numerical solution of the field variables are presented in Fig. 11. Through visual inspection, it is apparent that the ANN struggles with accurately predicting the qualitative evolution of all the field variables. Because

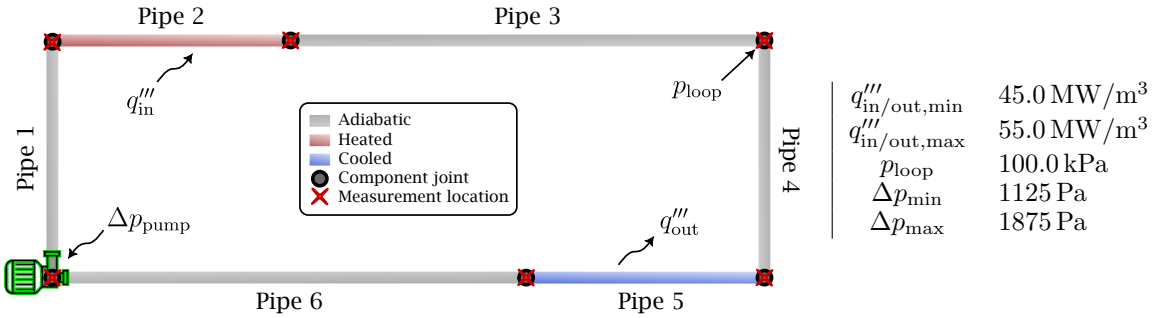


Figure 9: Configuration of pipes in a loop. Left: The imposed pump BC location and heat source/sink terms. Right: A list of the ranges for the pump pressure jump, heat source/sink terms, and loop pressure setting.

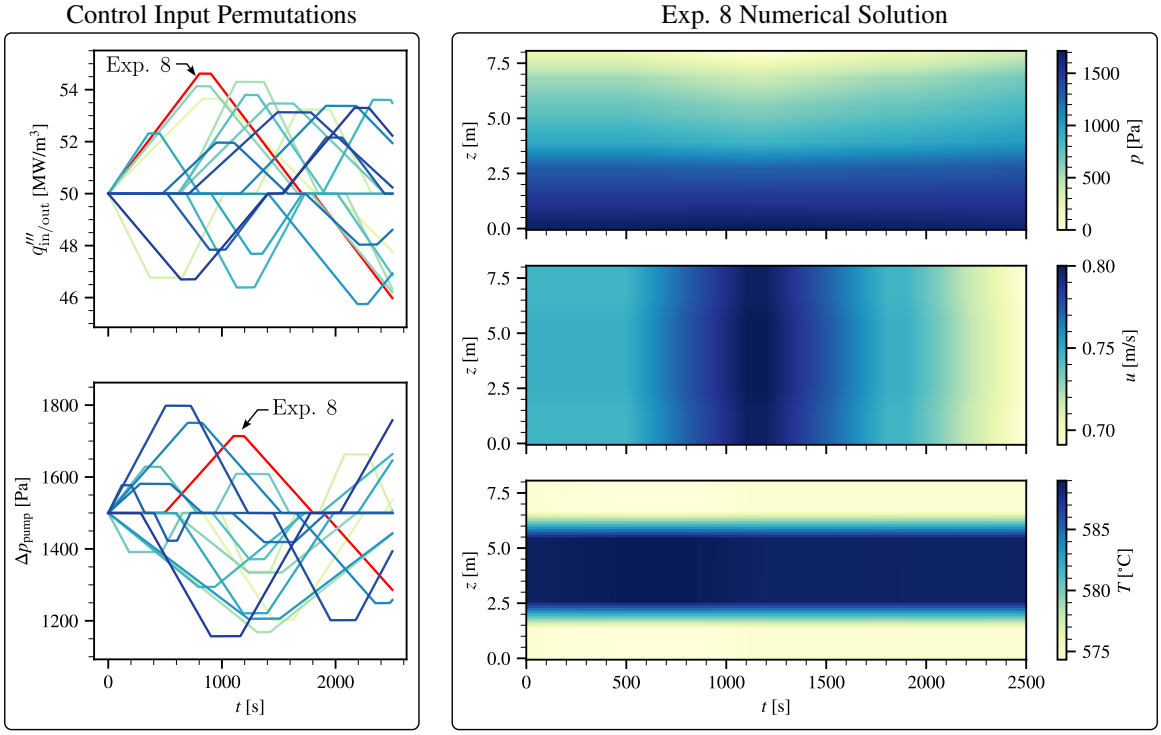


Figure 10: An overview of the training dataset for the loop configuration. Left: Pump pressure jump and heat source inputs manipulated in time. The experiment chosen to display the solution is annotated ("Exp. 8"). Right: Numerical solution of all field variables over the entire spatiotemporal domain.

the ANN relies on a purely data-driven approach, the neural network correctly fits the field values about the spatial locations corresponding with sensors. Yet, it fails to correctly interpolate the field values between the sensor locations – because the problem is more complicated than the advection-dominated heated channel. Whereas, the PSM has good quantitative and qualitative spatiotemporal prediction of all field variables.

The contrast between the ANN and PSM models is illustrated further in Fig. 12. The figure presents error in predicting all field variables averaged over all test datasets. The PSM model offers a significant performance advantage. As the prediction horizon for the ANN increases, a significant error is introduced in all fields. However, even for small time horizons, $t \leq 100\,\text{s}$, there are significant errors in the prediction of temperature and pressure. In summary, the results in this section show that in a complex experimental setting, a purely data-driven approach, i.e., the ANN model, overfits to sensor measurements and results in unphysical predictions over the entire spatiotemporal domain. In contrast, the PSM can generalize well to test data and achieve low error.

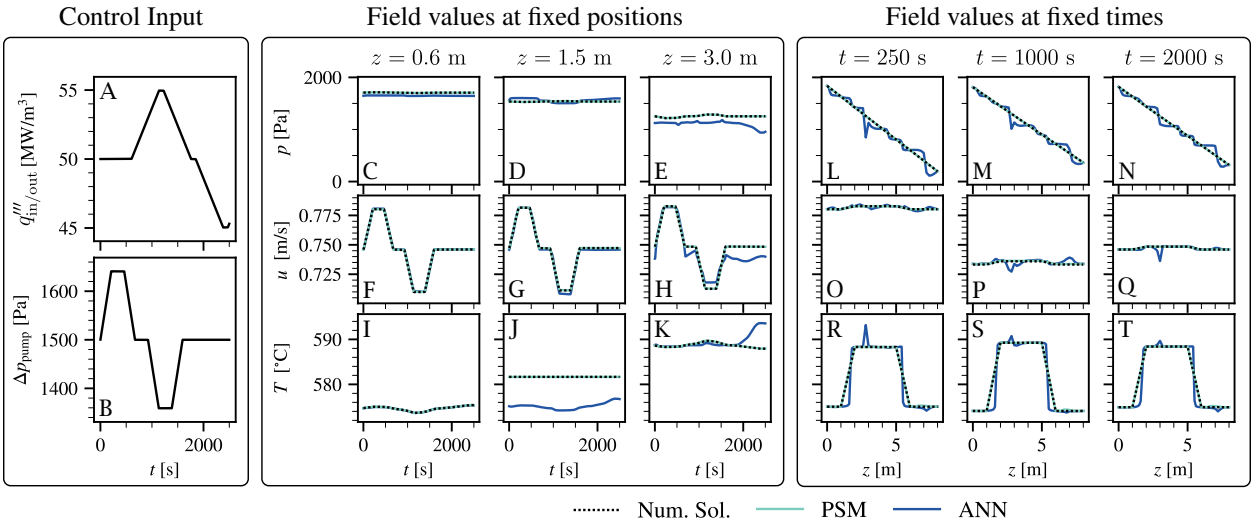


Figure 11: Contrasting the performance of PSM and ANN models in predicting the cooling system's evolution of pressure, velocity, and temperature. Subplots A-B: control input setting from the test dataset. Subplots C-K: field temporal evolution at fixed positions. Subplots L-T: field spatial distribution at fixed times.

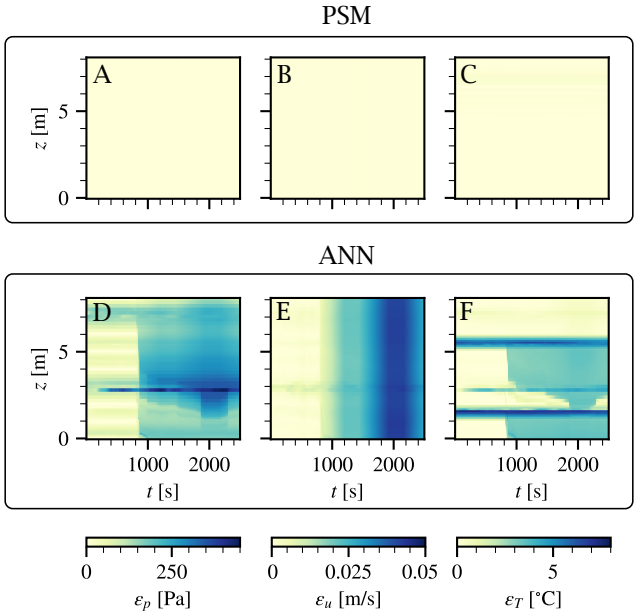


Figure 12: Comparison of the RMSE in predicting the spatiotemporal evolution of all field variables, averaged over all test datasets. Each field has its own unified colorbar scale, presented at the bottom. Subplots A-C display error for the PSM model, and subplots D-F display error for the ANN model.

4.2 PSM Capability: Diagnostics

Diagnostics in engineering systems are critical to maintaining operational efficiency and safety. It involves the detection, identification, and characterization of faults or degradations that might occur. There are both model-free [25] and model-based approaches [26] to diagnostics. Model-based methods offer the advantage of leveraging the system's underlying physics and systematically interpreting deviations from expected behavior [27]. Generally, in a model-based approach, we can identify the problem that has occurred based on a residual signature,

$$r = \mathcal{F}_{\text{nom}} - \mathcal{F}_{\text{m}} \,, \tag{58}$$

where r is the computed residual value, and $\mathcal{F}_{\text{nom}}$ and $\mathcal{F}_{\text{m}}$ are the system model's nominal vs. measured values, respectively. A non-zero valued r indicates that the system's current performance is deviating from nominal. For a

system with multiple fields and components, r would be multi-dimensional, generating a unique signature for each fault or degradation. Suppose the model, $\mathcal{F}$, incorporates the fundamental physics of the system. In that case, this approach is powerful because it enables a physics-based interpretation of the degradation and enables us to discriminate between the faults that have occurred.

A key benefit of utilizing PSMs is their ability to incorporate the governing physics of the system through PDEs. In a fully trained PSM, residuals from the mass, momentum, and energy PDEs, Eqs. (3) to (5), stabilize during the system's nominal fault-free operation. If a component degrades during operation, these residuals will change and lead to a unique r in Eq. (58). An algorithm can then use these changes in the residual signature to identify and classify the nature of the degradation.

To validate this approach, a test case was designed to simulate a degradation in the cooling system's performance. In this case, an abrupt partial blockage of the system occurs in Pipe 3 (immediately before the heat sink). The blockage was simulated by increasing the friction factor, f in Eq. (4), of that pipe by an order of magnitude. In Alg. 3, a procedure is proposed to detect degradation and use a classification algorithm to diagnose the incident. The algorithm requires continuous collection and analysis of new datasets, $(\mathcal{X}_{\text{new}}, \mathcal{Y}_{\text{new}})$, and analyzing them. It operates using two PSMs: a 'nominal model', which represents the fault-free system, and a 'twin model' whose parameters are updated once degradation is detected. The update is triggered by a parameter, ζ, in Line 6.

After the twin model is updated, residuals from both PSMs are analyzed using a classification model to infer the type of degradation that has occurred. The development of a classification model will be addressed in future work.

Algorithm 3 Degradation Detection and Classification

1: **Input:** Original dataset $(\mathcal{X}, \mathcal{Y})$, New dataset $(\mathcal{X}_{\text{new}}, \mathcal{Y}_{\text{new}})$, Nominal PSM model $\mathcal{F}_{\text{nom}}$, Threshold ζ, Classification model $\mathcal{C}$
2: **Initialize:** Twin PSM model $\mathcal{F}_{\text{m}}$, Degradation flag *latch* $\leftarrow$ False
3: **for all** sample (x_k, y_k) in $(\mathcal{X}_{\text{new}}, \mathcal{Y}_{\text{new}})$ **do**
4: Predict $\hat{y}_k = \mathcal{F}_{\text{nom}}(x_k)$
5: Compute $E = \frac{1}{n}\sum_{k=1}^{n}(\hat{y}_k - y_k)^2$
6: **if** $E > \zeta$ **and** *latch* is **False then**
7: Set *latch* to **True**
8: **end if**
9: **if** *latch* is **True then**
10: Train $\mathcal{F}_{\text{m}}$ on (x_k, y_k) without physics-informing
11: **end if**
12: **end for**
13: Compute nominal residuals as $r_{\text{nom}} = \text{PDEs}(\mathcal{F}_{\text{nom}}(x))$ for x in $\mathcal{X}$
14: Compute measured residuals as $r_{\text{m}} = \text{PDEs}(\mathcal{F}_{\text{m}}(x))$ for x in $\mathcal{X}$
15: Calculate residual $r = r_{\text{nom}} - r_{\text{m}}$, in Eq. (58)
16: Predict degradation type d using $\mathcal{C}$ on r: $d = \mathcal{C}(r)$ **return** d

The application of Alg. 3 to the data generated from the degradation test case is discussed next. In Fig. 13, the scaled residual values are contrasted for nominal and degraded models. The residuals are presented as a function of position. Within the cooling system, Pipe 3 is located at $4.0\,\text{m} \leq z \leq 5.0\,\text{m}$. There are a few important observations to consider. First, the nominal model has a relatively constant spatial distribution for the momentum residual, $r_{\text{mom.}}$. Whereas, for the degraded model, the spatial distribution shows a qualitative shift in the residual around the location of Pipe 3. Second, the mass and energy residuals do not show a significant shift in the spatial distribution of residuals. From these observations, a classification algorithm can be trained to identify the degradation that has occurred. In summary, in this section, an algorithm was proposed and utilized on a test case to show that PSMs can provide meaningful data to a diagnostics algorithm to identify system degradation or fault.

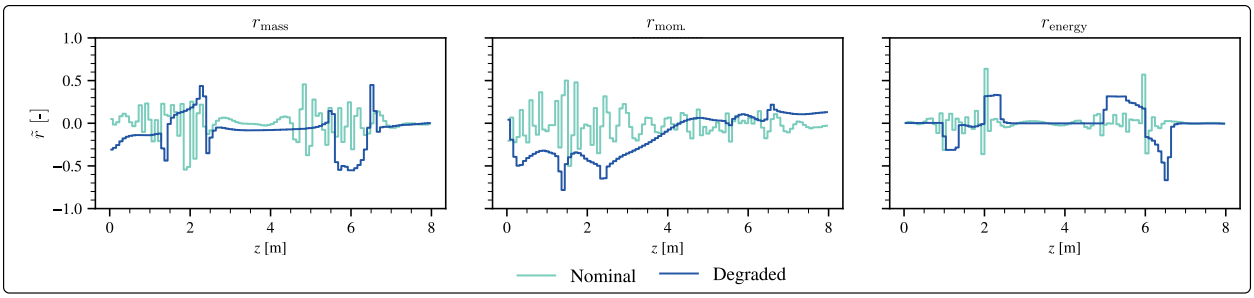


Figure 13: A comparison of the spatial distribution of nominal and degraded residuals, r and r_{deg}, respectively, in Alg. 3. The residual values have been scaled by their maximum and minimum value.

5 Discussion and Concluding Remarks

Autonomous systems are envisioned to achieve real-time optimization, be flexible in operation, and be fault tolerant. These capabilities necessitate a model-based approach. However, purely data-driven methods lack physical constraints like mass conservation. Consequently, model-free approaches, such as PID control, dominate the operation of modern transport-dominated systems such as chemical, biomedical, and power plants. To address this challenge, we propose the adoption of PSMs.

A PSM is a deep neural network trained by fusing sensor data with physics-informing using the components' PDEs. The result is a physics-constrained end-to-end differentiable forward dynamics model. Two separate experiments were designed, in silico, to demonstrate PSM capabilities for transport-dominated systems. The first experiment was a heated channel, and the second was a cooling system loop. The datasets used to train the PSM models are available online[1]. Evaluation on test data indicates that the PSM approach is quantitatively and qualitatively more accurate than a purely data-driven ANN. Additionally, both homo- and heteroscedastic noise were introduced during training to demonstrate the applicability of PSMs to a physical plant.

Owing to their design as end-to-end differentiable state-space models, PSMs demonstrated a multitask capability:

1. The first demonstration used a PSM model to construct a supervisory controller (Section 3.3). To achieve this, the proposed NCG algorithm, Alg. 2, used a linear state-space representation sequentially updated through Jacobian Linearization of the PSM. The capability was utilized to demonstrate both constant and time-dependent constraint enforcement. Thus, PSMs can be utilized for control schemes that require a nonlinear or linear state-space representation.

2. The second demonstration proposed a diagnostics algorithm that utilized the PSM as a basis (Section 4.2). Conservation laws, in the form of PDEs, can be evaluated using the PSM. The proposed algorithm, Alg. 3, uses residuals from each of the PDEs, which are available as a function of location. The demonstration simulated a partial blockage of a pipe in the cooling system. Two separate PSMs were used to obtain PDE residuals, one trained on the nominal dataset and one that had been transfer-learned on the degraded dataset. It was demonstrated that the differences in the residuals of the momentum equation can then be used as inputs to a classification algorithm.

3. An additional use-case of PSMs, not covered in this work, is as a basis for Digital Twins (DTs) [28]. DTs require that the "digital" model used to represent the physical asset is updated frequently through feedback from measurements to match drifts from anticipated dynamics e.g., due to wear and tear. Because the PSM is end-to-end differentiable, this feature can be accommodated through online learning of PDE coefficients.

There are potential drawbacks of the PSM that warrant further research. First, this work did not rigorously study the number and placement of sensors (measurement locations). The optimal choice would be highly dependent on the characteristics of the system. If a PSM is applied to an existing facility, adding additional sensors may not be possible. Second, the current work intentionally focused on relatively small systems with few components. For large systems, e.g., large power plants with multiple loops, a single PSM would not be sufficient to model the entire facility. Additional methods are needed to train and coordinate multiple PSMs representing separate parts of a plant. A potential approach may be to adopt a graphical architecture where PSMs represent transport between nodes.

[1]Dataset will be released on journal acceptance.

Nomenclature

Acronyms

ANN	Artificial Neural Network
BC	Boundary Condition
CG	Command Governor
MSE	Mean Squared Error
NCG	Neural Command Governor
PDE	Partial Differential Equation
PID	Proportional-Integral-Derivative controller
PINN	Physics-Informed Neural Network
PSM	Physics-informed State-space neural network Model
RG	Reference Governor
RMSE	Root-Mean-Square Error

Greek Symbols

α, β	Weighting coefficients for total loss, Eq. (32)
$\mathcal{L}_\Sigma$	Total loss, Eq. (32)
Ω	Space domain, Eq. (7)
ρ	Fluid density, Eqs. (3) to (5)
σ	Standard deviation of homoscedastic noise, Eq. (44)
τ	Time domain, Eq. (7)
θ	Set of parameters in a neural network, Eq. (7)
ξ	Variance factor of heteroscedastic noise, Eq. (45)

Roman Symbols

$\mathcal{F}$	Neural-network model (ANN, or PSM)
$\mathcal{L}_m$	Measurement loss, Eq. (10)
$\mathcal{L}_p$	Physics-informed loss, Eq. (12)
$\mathcal{X}_m$	Measurement input dataset, Eq. (8)
$\mathcal{X}_p$	Physics-informed input dataset, Eq. (13)
A, B	Linear state-space matrices, Eqs. (16), (52) and (53)
f	Friction factor, Eq. (4)
f_{loss}	Loss function used in network training, Eqs. (10), (12) and (31)
O_∞	Maximal output admissible set, Eq. (47)
p	Fluid pressure, Eq. (4)
q'''	Volumetric heat source, Eq. (5)
r	Residuals during diagnostics, Eq. (58)
r_k	RG reference input at step k, Eq. (46)
T	Fluid temperature, Eq. (5)
t	Time
u	Fluid speed, Eq. (4)
v	Control input
x	State
x_{00}, v_{00}	State and input values at the linearization point, Eqs. (50) and (51)
z	Position

CRediT authorship contribution statement

Akshay J. Dave: Conceptualization, Methodology, Writing - Original Draft, Funding acquisition. **Richard B. Vilim:** Supervision, Methodology, Writing - Review & Editing.

Acknowledgements

Argonne National Laboratory supports this work under the Laboratory Directed Research & Development Program, Project 2022-0077.

References

[1] Thomas Gamer, Mario Hoernicke, Benjamin Kloepper, Reinhard Bauer, and Alf J Isaksson. The autonomous industrial plant–future of process engineering, operations and maintenance. *Journal of Process Control*, 88: 101–110, 2020.

[2] Laurence A Belfiore. *Transport phenomena for chemical reactor design*. John Wiley & Sons, 2003.

[3] Ronald L Fournier. *Basic transport phenomena in biomedical engineering*. CRC press, 2011.

[4] R Byron Bird, Warren E Stewart, and Edwin N Lightfoot. *Transport Phenomena*, volume 1. John Wiley & Sons, 2006.

[5] Neil E Todreas and Mujid S Kazimi. *Nuclear systems volume I: Thermal hydraulic fundamentals*. CRC press, 2021.

[6] Claudio Mattiussi. An analysis of finite volume, finite element, and finite difference methods using some concepts from algebraic topology. *Journal of Computational Physics*, 133(2):289–309, 1997.

[7] J Richalet. Industrial applications of model based predictive control. *Automatica*, 29(5):1251–1274, 1993.

[8] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[9] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

[10] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92(3):88, July 2022. ISSN 1573-7691. doi:10.1007/s10915-022-01939-z.

[11] Florian Arnold and Rudibert King. State–space modeling for control based on physics-informed neural networks. *Engineering Applications of Artificial Intelligence*, 101:104195, 2021.

[12] Rui Hu. An advanced one-dimensional finite element model for incompressible thermally expandable flow. *Nuclear Technology*, 190(3):313–322, 2015.

[13] Rui Hu. SAM Theory Manual. Technical report, Argonne National Laboratory (ANL), Argonne, IL (United States), March 2017. URL `http://www.osti.gov/servlets/purl/1353375/`.

[14] Raúl Rojas. The backpropagation algorithm. *Neural networks: a systematic introduction*, pages 149–182, 1996.

[15] Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2008.

[16] Salman Khalid, Hyunho Hwang, and Heung Soo Kim. Real-world data-driven machine-learning-based optimal sensor selection approach for equipment fault detection in a thermal power plant. *Mathematics*, 9(21):2814, 2021.

[17] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[19] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, pages 1–26, 2020.

[20] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

[21] Thomas E Rehm. Advanced nuclear energy: the safest and most renewable clean energy. *Current Opinion in Chemical Engineering*, 39:100878, 2023.

[22] Rebecca Rose Romatoski and Lin-Wen Hu. Fluoride salt coolant properties for nuclear reactor applications: A review. *Annals of Nuclear Energy*, 109:635–647, 2017.

[23] Emanuele Garone, Stefano Di Cairano, and Ilya Kolmanovsky. Reference and command governors for systems with constraints. *Automatica*, 75:306–328, 2017. ISSN 00051098. doi:10.1016/j.automatica.2016.08.013.

[24] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[25] Janos Gertler. *Fault detection and diagnosis in engineering systems*. CRC press, 2017.

[26] Rolf Isermann. Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in control*, 29 (1):71–85, 2005.

[27] Tat Nghia Nguyen, Roberto Ponciroli, Timothy Kibler, Marc Anderson, Molly J. Strasser, and Richard B. Vilim. A physics-based parametric regression approach for feedwater pump system diagnosis. *Annals of Nuclear Energy*, 166, 2022. doi:10.1016/J.ANUCENE.2021.108692.

[28] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020.