

Development and assessment of a reactor system prognosis model with physics-guided machine learning

Anil Gurgen*, Nam T. Dinh

Department of Nuclear Engineering, North Carolina State University, Campus Box 7909, Raleigh, NC 27695

ARTICLE INFO

Keywords:

Nuclear power plant
Autonomous control
Prognosis
Physics-guided machine learning

ABSTRACT

Autonomous control systems provide recommendations to help operators in decision-making during plant operations ranging from normal operation to accident management. An important step of autonomous control is prognosis. In nuclear engineering domain, prognosis is the process of predicting future conditions of a system or equipment based on present signs and symptoms of a fault. The prognosis model allows predicting future reactor states for possible candidate control strategies so that the outcomes can be evaluated to determine the best control strategy. The prognosis model requires representing direct relationships between the symptoms and the predictions. In nuclear engineering, computational simulations are approximate representations of the operation of the real system. However, prognosis with computational simulations requires high computation power and time due to possible large number of scenarios. Necessary computation resources can be reduced with machine learning (ML) approach for fast predictions by building a surrogate function using the simulation data. A critical issue is, ML models are ignorant of physical knowledge, and these models approximate statistical relationships between the system variables. This ignorance can produce results that are inconsistent with physical laws, even if an optimal result is achieved from a mathematical point of view. Physics-guided machine learning (PGML) is an approach to tackle this issue. This work formulates and illustrates a framework to guide development and assessment of the ML-based prognosis model for autonomous control systems. The development of the prognosis model considers the training of a ML model which consists of optimizing many aspects of the ML approach. The assessment of the prognosis model considers training data limitations and uncertainties of the ML approach. Prognosis models with standalone ML and PGML are developed and assessed on the loss-of-flow scenario of Experimental Breeder Reactor II. The results indicate that PGML based prognosis model has the best performance compared to other prognosis models.

1. Introduction

Autonomous control systems refer to systems that are capable of using computing resources to make decisions and operate in a real-world environment without external control for a period of time[1]. The emergence of autonomous control systems became a reality with Artificial Intelligence (AI). AI can be defined as computer systems that are capable of intelligent behavior, and for autonomous control systems, the expectation of intelligent behavior is to implement a structured decision-making process. While human decision-making is a complex phenomenon, it can briefly be defined as the process of identification and selection of one item from many possible alternatives. In decision-making, there are options to be considered, and each option offers a different trajectory, and the decision-maker chooses the alternative with a trajectory closest to the desired condition. Autonomous control system automates the decision-making process, and the capabilities that are needed for an autonomous control can be listed as (1) diagnose a situation, (2) identify the viable courses of actions, and (3) determine the best, optimal, or at least an acceptable action or a sequence of actions to transition to a safe state[2].

*Corresponding author. Currently affiliated with University of Maryland, Department of Materials Science and Engineering
Email address: agurgen@umd.edu
ORCID(s): 0000-0002-2304-8061 (A. Gurgen)

Autonomous control of nuclear reactors integrates autonomous decision-making in the control procedure of nuclear reactors along with the human operators. Nearly Autonomous Management and Control (NAMAC) system is introduced by Lin *et al.* [3] provides recommendations to operators to maintain safety and performance of the reactor. NAMAC adopts a modular approach at which different steps of decision-making is performed by different modules. The modules are identified as Digital Twin (DT) which is a virtual environment that contains data and related knowledge about the system [4]. An important NAMAC function is prognosis which is the initial step of evaluating decision options in the decision-making framework. The goal of “determining the best, optimal or at least an acceptable action or a sequence of actions to transition to a safe state” in NAMAC is dependent on the successful predictions of the prognosis function.

General definition of prognosis is *a forecast of the future course or outcome of a situation*. In nuclear engineering domain, prognosis is the process of predicting future conditions of a system or equipment based on the present signs and symptoms of a fault. Prognosis in autonomous control is to predict future reactor states for possible candidate control strategies so that the outcomes can be evaluated to determine the best control strategy. On the functional level, prognosis is a time-series forecast function with initial and boundary conditions. The initial condition of the prognosis is the historical values of reactor state variables, and the boundary

condition of the prognosis is the candidate control strategies which defines the progression of the transient. With given initial and boundary conditions, prognosis can forecast the future course of reactor state variables at the time of the initiation of the transient.

In this work, we propose a methodology for the development and assessment of the prognosis model for an autonomous control framework. Section 2 introduces the functional requirements of a prognosis model, and Section 3 illustrates the development and assessment methodology for prognosis. Section 4 gives details of a case study to test the development and assessment methodology, and the results are presented in Section 5.

2. Prognosis Function

Prognosis requires techniques to represent direct relationships between the input parameters and the outcome. In nuclear engineering, computational simulations are approximated representations of the real system operation. System codes are computer codes that numerically solve partial differential equations (PDEs) to simulate complicated physical phenomena in components and system-level interactions between the components of a nuclear power plant. An example of such code is GOTHIC. GOTHIC is an integrated, general-purpose thermal-hydraulics software package for analysis of nuclear power plant containment, confinement buildings, and system components[5]. GOTHIC solves conservation equations for mass, momentum, and energy for thermal fluid problems involving multicomponent and multiphase compressible flow. Reactor state variables follow a distribution governed by non-linear equations of GOTHIC such as

$$S_t + \mathcal{N}[S; \mathcal{P}] = 0 \quad (1)$$

where S_t is the time derivative of the reactor state variables and \mathcal{N} is a non-linear PDE operator that represents the reactor operation, parameterized with \mathcal{P} . The prognosis is a time-series forecast model with initial and boundary conditions. Therefore, from the modeling point of view, the prognosis is not different from system codes. However, system codes take some time to complete the simulation, and a simulation-based prognosis model cannot simulate candidate control strategies in a faster-than-real-time decision-making framework.

The purpose of the development of prognosis is to build a surrogate function that allows a fast-running simulation in an autonomous decision-making framework; therefore, the surrogate function needs to approximate the GOTHIC equations. This can be expressed as

$$f_s(S) = S_t + \mathcal{N}[S; \mathcal{P}] + \epsilon_s \quad (2)$$

where f_s is the data-driven surrogate function, and ϵ_s is the error made by the surrogate function in approximating GOTHIC equations. The surrogate function is a data-driven

model. Data-driven modeling is representing the relationships between physical quantities based on the data itself without explicit knowledge about the system. Machine learning (ML) is a sub-discipline of data-driven modeling, and it can be defined as using statistical learning techniques to extract relationships among physical quantities directly from the data. Even though ML models require much attention and resources for the training, ML models are fast enough to perform multiple predictions in a very short time. Therefore, ML-based surrogate function allows prognosis to simulate multiple candidate control strategies in a faster-than-real-time decision-making framework.

By assuming the GOTHIC solution is the true solution, the predictive capability of the surrogate function is determined by how good the surrogate function is in approximating the GOTHIC solution, which requires quantification of the total error in the approximation. From the validation point of view, the predictive capability of the surrogate function is how to quantify and estimate errors during the development and application of the surrogate function.

Prognosis is a time-series prediction model with initial and boundary conditions, where initial conditions mark the beginning of the transient, and boundary conditions shape the progression of the transient. There are two inputs to the prognosis model. The first input is reactor state variables for a certain length of history, T_h . The initial condition of the prognosis is the reactor state values from the current time to T_h with total of d number of variables.

$$\mathbf{X}^d(t') = [S^d(t), S^d(t-1), \dots, S^d(t-T_h)] \quad (3)$$

The second input to prognosis is control strategies, \bar{I} , which may contain system-level control strategies, component-level control strategies, and temporal-level control strategies that specify the time of execution of control strategies. \bar{I} is the boundary condition of the prognosis that define the progression of the transient.

The output of prognosis is the prediction of quantity of interest (QoI) for a certain length of future, T_p . The length of future is the desired timeline that the prognosis model makes a prediction. With $t' \in [t-T_h, t]$ being history time, $t^* \in [t+1, t+T_p]$ being future time and ω is the model parameters for the surrogate function, the function form of prognosis can be written as

$$S^d(t^*) = f_s(\mathbf{X}^d(t'), \bar{I}(t), \omega) \quad (4)$$

Prognosis uses multi-step prediction scheme to forecast the future. Multi-step prediction is a method based on iterating the predictions until the desired prediction length is achieved. In this scheme, a predictive function is needed to predict the next value for an input sequence, and at the next timestep, the output of the previous timestep is included in the input sequence. The prognosis model uses the f_s to make single-step predictions in a multi-step prediction scheme to complete the time-series prediction. Multi-step prediction of

prognosis is illustrated in Figure 1. In multi-step prediction scheme, f_s needs to be accurate, stable, and consistent so that the prediction error does not increase significantly with each timestep to cause the prediction scheme unstable and inaccurate.

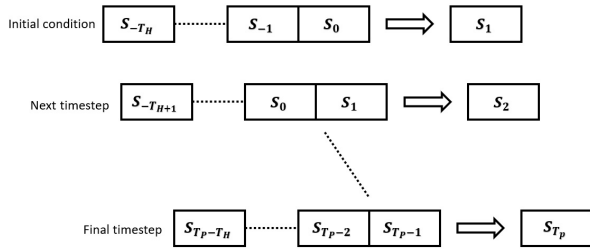


Figure 1: Multi-step prediction of prognosis

Deep neural networks are ML algorithms with multiple layers of nodes, with the outputs of one layer is being the input of the next layer. Multiple hidden layers help neural network to capture complex interactions between the input features and output. Recurrent Neural Network (RNN) is a subclass of neural networks with a recurrent connection, that allows RNN to perform the same task for every element of a sequence; therefore, the output is dependent on the previous computations. Recurrent connection allows the model to make use of sequential information because recurrent connection acts as a memory that captures information about what has been calculated so far. This memory helps RNN to take not only the input data but also the previous information into account to perform time-series prediction. The dynamic pattern in the data is captured by the network through performing backpropagation at each point in time for every element of a sequence with a feedback loop network form[6]. With its time-series prediction capability, RNN is a good candidate as a surrogate function for prognosis.

One of the primary concerns with ML is, ML models are ignorant of physical knowledge, and these models approximate statistical relationships between system variables which can produce results that are inconsistent with physical laws[7]. Therefore, ML models have a chance to fail to generalize in scientific problems. To cope with this limitation, there is a need to explore the bridge between data-driven models and physics-based models for scientific applications. This issue is well known in the scientific community, and it is possible to find different studies with different approaches in literature with the aim of blending scientific knowledge in ML models.

One particular approach is the subject of this work. This approach is representing physics during neural network training with using an additional term to loss function inspired by the physical understanding of the domain. Physics-informed machine learning is introduced in a study[8] that tries to develop data-driven solutions to nonlinear PDEs. In this study, a data-driven model is developed to solve Burger's equation. In addition to the traditional squared loss functions, the authors added another loss function based on

the analytical and numerical solution of Burger's equation. This method is reported to be more accurate and data-efficient learning algorithm. In another study by the same authors[9], Navier-Stokes equations are used in the loss function, and it is reported that Navier-Stokes informed neural network is robust to low resolution and substantial noise in the observation data. Similarly, a study [10] showed that including Navier-Stokes equations into the loss function allows to extract velocity and pressure fields with the images of a concentration of a passive scalar. These studies focus on single physics to inform the neural network, and the equations that govern the problem are also part of the neural network training.

Adding an additional term to loss function is subject to other works. Physics-guided machine learning is introduced in the study [7], and studies [11][12] use a physics-based loss function in addition to traditional loss function to predict transient lake temperature based on the environmental conditions. The method is reported to be showing improved performance over traditional knowledge-based models and standalone ML models, and better consistency with physical knowledge. A significant different in physics-guided machine learning is, partial knowledge about the problem is used in the neural network training. We aim to investigate physics-guided machine learning approach for surrogate function development because the governing PDEs of GOTHIC are known and can be used partially to develop the physics-guide model.

3. Methodology

This section describes the methodology for the development and assessment of the surrogate function of the prognosis model. The methodology is designed to meet the following set of requirements.

- Construct the training database considering the physics of the problem
- Find the optimal parameters and hyperparameter set of the model for the problem with the given training database
- Identify different sources of the uncertainty of the data and model
- Evaluate the predictive capability of the surrogate function and prognosis model

The generic problem of learning is to find a relationship between two variables that are related by an unknown probability distribution[13]. The unknown probability distribution is approximated by an estimate function which is a subset of hypothesis space that contain the set of all possible functions that can return the relationship between the variables. ML is the effort to find the estimate function that has the best predictive capability. The concept of predictive capability of the ML model is defined with the generalization capability. Generalization error refers to a measure of the accuracy of

the prediction that refers to the error between the learner's hypothesis and the unknown probability distribution[14].

It is often not possible to find best estimate function due to practical limitations such as finite model size, finite training data, and finite computing resources. Additionally, the predictive capability of the ML model is dependent on the training data and the model itself. The methodology of surrogate function development and assessment is a practical realization of the minimizing generalization error.

3.1. Step 1: Training data preparation

Training data preparation aims to construct the training database considering the specification of the problem. This step belongs to the preparation for prognosis model development, which optimizes the training data utilization for the surrogate function. This element starts with identifying the prognosis model target considering the issue space of the problem. Issue space is the complete scenario space that an autonomous control framework can identify and make the decision, and the prognosis model needs to predict future conditions in the issue space. Then the time-series data of the issue space is collected. An essential step before training the ML model is selecting the input features. The selected input features need to be sufficient to capture the quantity of interest (QoI) behavior and reflect the physical characteristics of the problem. A causal relationship is required between the input features and the QoI so that the QoI can be uniquely determined from the training data. In this study, the input features are selected with domain expertise considering the physics of time-series data.

Another important step in training data preparation is data scaling of the input features. Data scaling is required for the training database because the reactor state variables have different ranges. Supervised ML infer the relationship between the input features and the output labels. When input features have different magnitudes, the features with wider ranged and higher values have more influence over the training process than others, resulting in errors in the prognosis function parameters. In this study, the input features are linearly scaled between 0 and 1, considering the fixed upper and lower bound values. These values are determined after analyzing the training data.

3.2. Step 2: Model hyperparameter tuning

The hyperparameters of the surrogate function define the model complexity and the training algorithm for learning. This element belongs to the prognosis tuning, which optimizes the hyperparameters of the ML model for a lower generalization error. The optimization of the hyperparameters is a tedious trial-and-error type searching; therefore, even though it does not guarantee to find a global minimum of the generalization error, the deeper search improves the chance of finding the best hyperparameter set. For the given training database, an approximator is needed to the generalization error. We use k-fold cross-validation (CV) to split database into training and validation datasets, so that no information about testing data can leak into the hyperparameter tuning. In k-fold CV, the training data is split into k smaller

datasets, and the ML model is trained using $k - 1$ of the smaller datasets, and the trained model is validated on the remaining dataset. This procedure is repeated k times so that each smaller dataset becomes part of both training and validation. The CV error is the averaged value of errors for k calculations.

The tuning process is a hyperparameter optimization process that finds the hyperparameter set that minimizes the CV error within the search iterations. We adopt Bayesian search which is a direct search to minimize the CV error by selecting hyperparameter sets that potentially have a lower CV error. Bayesian search builds a surrogate probability model of the CV error and finds the hyperparameter set that performs best on the surrogate model. Then the best hyperparameter set is trained with the actual ML model to calculate the actual CV error, and the surrogate model is updated by incorporating the new results. The process is repeated until the desired iteration is reached, the extremum of the objective function is located, or a good result is located. The steps of the Bayesian search are given in Figure 2. g is a random forest regressor that approximates CV error for a given hyperparameter set.

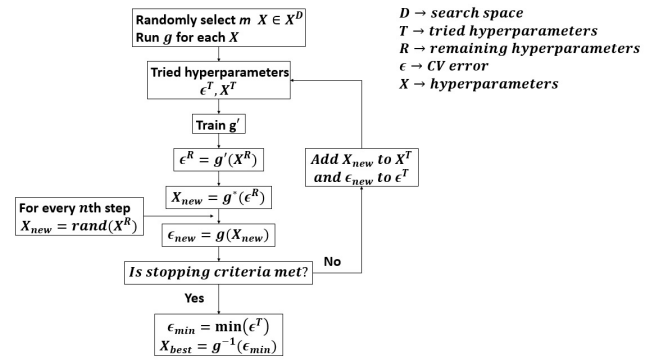


Figure 2: Workflow of Bayesian search for prognosis tuning

Once the iterations are completed, the hyperparameter set that minimizes the CV error within the iterations is selected as the best hyperparameters, and it is assumed that these hyperparameters define the optimal model complexity for given training database.

3.3. Step 3: Model parameter training

This step aims to minimize the training error of the ML model for the given training data without increasing the generalization error. This step belongs to the prognosis training which optimizes the parameters of the ML model. In this step, ML model is trained with the whole training database without splitting the validation database. Learning rate is an important hyperparameter that affects the speed the weights and bias of the network are updated during the training process. During the hyperparameter optimization, the learning rate is separated from the other hyperparameters. Hyperparameter tuning can be seen as exploration activity to spot possible regions of local minimum points in the loss function and to find the optimal hyperparameter set. And

this step can be seen as exploitation activity to use smaller learning rates to reach the local minimum points in the loss function. This step fine-tunes the training with the suggested hyperparameter set and adaptive learning rate. The adaptive learning rate is a method that updates the learning rate during the training based in the performance of the training on the testing database. A step decay[15] is adopted in this element, at which the network is trained with a fixed learning rate, and the learning rate is reduced by half whenever the generalization error stops decreasing. The training algorithm of the model is given in Algorithm 1. In step 2, CV error is used for approximating the generalization error, and in this step, the testing error is used as an approximator to the generalization error.

Algorithm 1 Training algorithm of the model with an adaptive learning rate

```

Initialize model parameters
while  $epoch < epoch_{max}$  do
    Calculate training loss on training database
    Update model parameters
    Calculate testing loss on testing database
    if testing loss is increased then
        Increase counter
        if counter is outside of scheduler window then
            Reduce learning rate by half
        end if
        if "counter is greater than validation patience then
            Stop the training process
        end if
    end if
    Save model parameters
    Report training loss
end while
Export model parameters

```

Once the iterations are completed, the model parameters that minimize the training error without increasing the testing error are selected as the ML model parameters. These parameters give an optimal training error for the optimal hyperparameters.

3.4. Step 4: Evaluating testing error

This element aims to evaluate the generalization error of the ML model. The generalization error of the prognosis model is highly dependent on the training data. For any problem, the constructed training data can have different properties that can affect the generalization error of the prognosis error. The properties if the training data is a result of the finite-sample nature of the training data. These properties are hypothesized as the imperfect training data. Imperfect training data can be due to several reasons, such as training data is not enough, training data is not representative, training data is not balanced, and training data is noisy. These conditions are illustrated in Figure 3.

For a predictive model, the generalization error of the surrogate function needs to decrease with increasing training

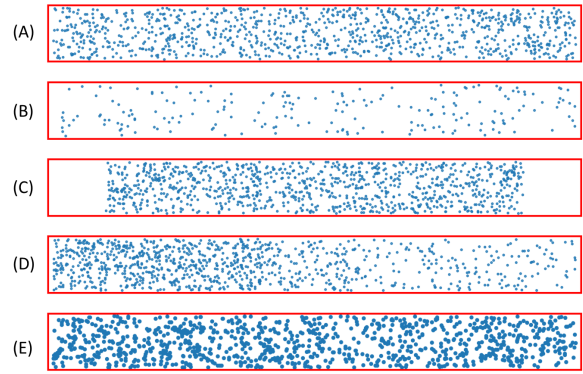


Figure 3: Imperfect data conditions for convergence analysis, A) ideal training data, B) training data is insufficient, C) training data is not representative, D) training data is not balanced, E) training data is noisy

data size. Condition B is proposed to test this property with insufficient data to check if the generalization error decreases with increasing training data. It can also be tested to check if the generalization error of the surrogate function is converged for the given training dataset.

Another condition of the training data is to check of the training data is representative. ML models tend to perform well on tasks of the same nature as the training data. Models with good generalization capability can utilize the patterns learned from the training data to predict similar but new tasks. Training dataset is constructed with the observations, and a key issue with surrogate function is the question of generalizability outside the training domain. This condition is important because even though increasing training data decreases the generalization error, sampling the training data from a definite portion of the probability distribution may not guarantee that the surrogate function can learn all the patterns in the training data. Therefore, the surrogate function can fail to predict scenarios that are not covered by the training data. Condition C tests this condition, and it is required to determine train/test data similarity conditions of the model. In this step, the surrogate function is trained with a training data, and tested against the conditions where similarity decreases.

Next condition of the training data is to check if the training data is balanced, and this step of convergence analysis tests the effect of balance conditions of the training data on the generalization error. This condition is important, because even though the training dataset covers the testing conditions, if the training data is focused on a specific portion of the probability distribution, it is possible that the model learns the patterns on the focused region and cannot capture the actual relationships of the testing conditions. This type of training data can be considered as imbalanced datasets where the observations on the probability distribution is not uniform. In this step, the surrogate function is trained with training datasets with different uniformity, and it is tested to see how the data imbalance affects the generalization error of the surrogate function.

An important measure for condition C and D is to define a dissimilarity measure between the training and testing databases. The prognosis model considers time-series data; therefore, a time dependent distance metric is required to define dissimilarity between points in space. Dynamic time warping (DTW) is a method that aligns the time-series considering the shapes and trends in the time series[16]. The alignment considers some restrictions between the points in time series. Once the alignment is found, the distance is calculated between the aligned points. DTW distance is used as the dissimilarity measure between two time-series in this framework. Training and testing datasets contain multiple time-series, and for a training dataset containing N time-series and a single time-series in the testing dataset, the dissimilarity is measured as the minimum DTW distance between the testing data and the training dataset. This condition can be expressed as

$$\gamma = \arg \min_{i \in N} d(x_i, y) \quad (5)$$

where d is the DTW distance and γ is the dissimilarity measure. For a testing dataset containing M time-series, the dissimilarity is measured as the average DTW distance for each testing episode. This condition is expressed as

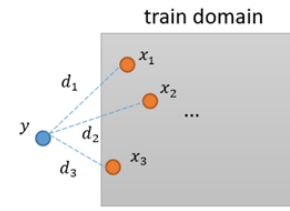
$$\gamma = \frac{1}{M} \sum_{j=1}^M \arg \min_{i \in N} d(x_i, y_j) \quad (6)$$

The dissimilarity for a single testing episode and multiple testing episodes is illustrated in Figure 4. To define the condition of imbalanced training data, the distribution of distances between all episodes in the training data is used. The balance of the training dataset is measured with the skewness of the distribution. For the training database, skewness is defined as the difference between the mean and median of the distribution. If the mean of the distribution is far from the median of the distribution, the database is assumed to be skewed towards the mean of the distribution and not balanced. The skewness is represented with μ .

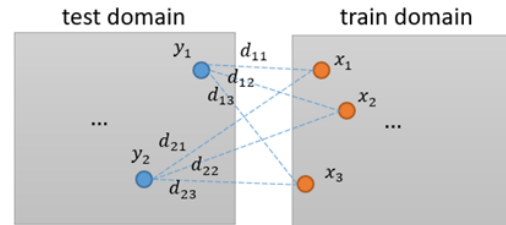
Final condition of the training data is to identify the effect of noise in the training data. The noise in the training data distorts the probability function that the ML model could recover, and this step of convergence analysis tests the effect of noisy data on the generalization error. Surrogate function is trained with training datasets where artificial noise is added and tested against the testing database without noise. With φ being the percentage of the noise, the artificial noise addition is represented by

$$S_{n_i}(t) = S_i(t) + \mathcal{N}(0, \varphi \times S_i(t)) \quad (7)$$

Properties of each condition are given in Table 1. The surrogate function and prognosis model are tested in convergence analysis.



(a) For a single testing episode



(b) For multiple testing episodes

Figure 4: Dissimilarity in time-series data

Table 1

Data conditions for convergence analysis

Data condition	Sample size	Dissimilarity	Skewness	Noise
Condition A	$l = 100\%$	$\gamma \approx 0$	$\mu \approx 0$	$\varphi = 0$
Condition B	$l < 100\%$	$\gamma \approx 0$	$\mu \approx 0$	$\varphi = 0$
Condition C	$l = 100\%$	$\gamma > 0$	$\mu \approx 0$	$\varphi = 0$
Condition D	$l = 100\%$	$\gamma \approx 0$	$\mu > 0$	$\varphi = 0$
Condition E	$l = 100\%$	$\gamma \approx 0$	$\mu \approx 0$	$\varphi > 0$

3.5. Step 5: Quantify model uncertainty

This element aims to propagate and estimate the uncertainties of the ML model in making a prediction. In the optimization process of the ML model, there are parameter and hyperparameter uncertainties, and the proposed approach for uncertainty quantification is to make predictions with multiple prognosis models to estimate mean and variance of the prediction. Hyperparameters of the ML model can be considered as controllable because the user can specify them directly or choose them after a sensitivity study. These uncertainties form the model hyperparameter uncertainty, and they can be quantified by training several models instead of training a single network. Ensemble averaging with K networks with different network structures can be used to quantify uncertainties related to model hyperparameters. Ensemble averaging is a method where multiple diverse models are developed to make a prediction[17]. The idea is to quantify how different hyperparameters and data preparations affect the generalization error by averaging the model predictions.

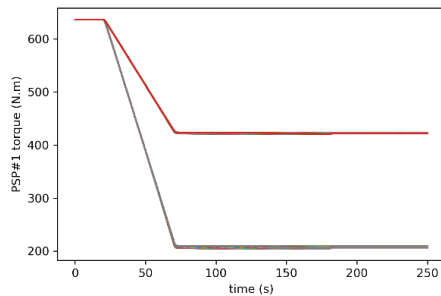
Parameters of the ML model can be considered as uncontrollable because the user does not directly specify their values but relies on a learning algorithm to learn their values numerically. Parameter uncertainties can be quantified by disabling the connections of network parameters during the prediction. Dropout is a method that network nodes are

disabled randomly during training and prediction. Dropout introduces randomness to the prediction, and empirical distribution over the outputs can be used to quantify uncertainties related to model parameters[18]. If the prediction of a single model is repeated several times, the prognosis model gives different predictions so that it is possible to estimate the distribution and uncertainty. The distribution of the predictions results from disabling connections and represents the value of the connection in making prediction and can be attributed to the model parameter uncertainty[19].

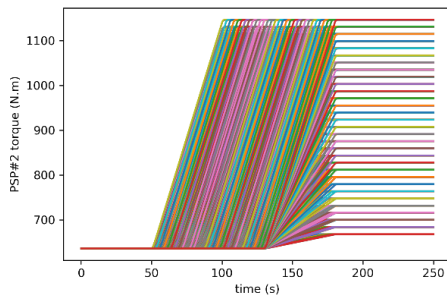
In this step, the variance of the prediction is calculated. By combining the model parameter and hyperparameter uncertainties, it is possible to make multiple predictions, and once the predictions are complete, the mean and the variance are calculated.

4. Case study formulation

This section illustrates the development of prognosis models with a case study on a transient scenario. The transient scenario is considered as the accident scenario of primary flow anomaly of Experimental Breeder Reactor (EBR-II)[20], and GOTHIC EBR-II[21] model is used to generate transient data. The model has been validated with experimental results[21]. In this scenario, primary sodium pump (PSP) #1 ramps down as an indicator to transient,



(a) PSP#1 torque



(b) PSP#2 torque

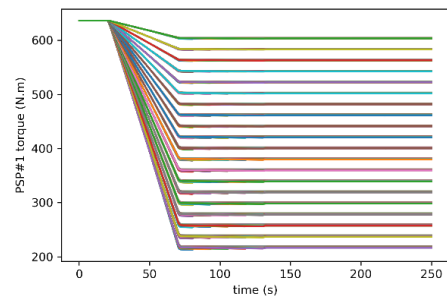
Figure 5: PSP torque plots in the SR database

and PSP#2 ramps up to cope with the transient. The databases are generated by sampling PSP#2 end torque and ramping time by uniform distribution. PSP#1 torque is pre-defined before sampling the PSP#2 parameters. There are

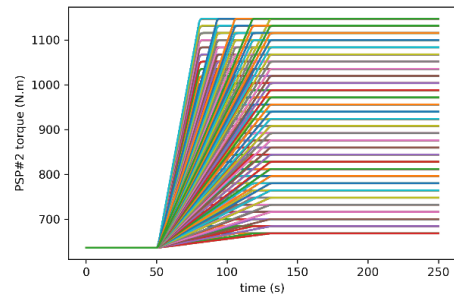
two scenarios: a steady ramp (SR) case and a moving ramp (MR) case. SR case is used for both training and testing the prognosis models, and MR case is used for testing only to assess the scalability of prognosis models.

SR database is generated by ramping down the PSP1 torque to 33.4% and 66.7%, and ramping up the PSP#2 torque to 31 different values with 31 different ramping initiation times. In the SR database, PSP#2 always ramps up in 50 seconds, and the time to initiate the ramping changes for different episodes. PSP#1 and PSP#2 torque plots for all episodes in the SR database are given in Figure 5

MR database is generated by ramping down the PSP#1 torque to 20 different values and ramping up the PSP#2 torque to 31 different values with 5 different ramping times. In the MR database, PSP#2 always starts ramping up after 50 seconds from the transient, and ramping time changes for different episodes. PSP#1 and PSP#2 torque plots for all episodes in the MR database are given in Figure 6.



(a) PSP#1 torque



(b) PSP#2 torque

Figure 6: PSP torque plots in the MR database

4.1. Input features

The database is constructed from the issue space by selecting the input features and QoI. The QoI is the maximum fuel centerline temperature (T_{FCL}), and the input features are selected based on how GOTHIC calculates the maximum T_{FCL} . The prognosis model is expected to approximate the GOTHIC equations; therefore, the selected input features represent the GOTHIC equations and assumptions to calculate T_{FCL} . All the input features of the database are given in Table 2

Table 2

Input features of prognosis model

PSP#1 mass flow rate
PSP#2 mass flow rate
Core channel inlet mass flow rate
Core channel outlet mass flow rate
Z-pipe mass flow rate
High pressure lower plenum temperature
Low pressure lower plenum temperature
Upper plenum temperature
Core cell temperatures
Core cell pressures
Core cell mass flow rates
Active core rod centerline temperature
Active core clad wall temperature
Total power deposited in core channel
Total core power generation
Intermediate heat exchanger cooling power
PSP#1 speed
PSP#2 speed

4.2. ML model for surrogate function

The ML model selected for the case study is RNN considering the time-series prediction capability. Long-short Term Memory (LSTM) is a special type of RNN that is introduced to overcome vanishing gradient problems in RNNs[22]. LSTMs are explicitly designed to learn long-term dependency problems, and they contain gates that decide which information should be thrown away or kept, which helps to keep old valuable information in the memory. LSTM can learn and remember dynamic patterns in the data with complex interactions between its gates, even for length sequence data analysis[23][24]. Therefore, RNN with LSTM blocks are used in this work. The hyperparameters of the LSTM are optimized with the Bayesian search given in Figure 2, and the parameters of the LSTM are learnt with the algorithm given in Algorithm 1 with using ADAM[25] optimizer with PyTorch[26] library.

The learning of the ML model is obtained by updating parameters of the model based on feedback from the training data. Loss function is an indicator of how well the ML model is working as intended. Accuracy of the ML model shows the ability of how close the predictions of the ML model to the actual values of the variables and Mean Square Error (MSE) is considered. Low accuracy doesn't always indicate a good predictive capability due to the possibility of overfitting. Overfitting is an issue in which the model fits well to the training dataset, but it shows bad predictive capability against a new dataset. In order to avoid overfitting, regularization is used in the training of ML. Regularization parameter(λ_{reg}) tries to estimate the overfitting and penalizes a more complex model so that the model focuses on approximating the probability distribution of the data rather than capturing every point in the dataset. This brings another evaluation metric, simplicity, which can be defined as the state of how robust the model is, where a robust model is defined as a model that produces little or no changes

in its output for small changes in its inputs. Accuracy and simplicity are metrics in loss function of a standalone ML model.

In order to improve the standalone ML-based surrogate function, PGML is considered in this work. PGML is an approach that explicit knowledge is accounted for during the training of ML models. It is a hybrid approach that a physics-guide model guides the training of the ML model by penalizing the training if it is inconsistent with the physics-guide model. This penalization brings a new evaluation metric to the loss function, which is consistency. Consistency can be defined as how compatible our learning method is with known scientific principles and known relevant and validated knowledge. Consistency metric is added to the loss function in addition to accuracy and simplicity; therefore, the loss function of PGML punishes non-accurate learning, learning a more complex model, and physically inconsistent learning. The architecture of the PGML is given in Figure 7 with conservation of mass, momentum and energy-based physical consistency.

\mathcal{L}_{con} is the physical consistency term of the loss function and λ_{con} is the balance parameter that defines the degree of penalization of physical consistency. λ_{con} is a parameter that needs to be tuned during the ML training and using a large value for λ_{con} enforces the ML model to conform to the physics-guide model and share the limitations of the physics-guide model. Using a small value for λ_{con} results with loss of consistency information, and the ML model converges to a pure ML model with no physics guidance.

The physics-guide model can be described as a physics-based model that represents the knowledge about the system, and the relationships among the input and output of the ML model can be partially described in the form of explicit equations. A general physical consistency loss function has the form given in Equation 8.

$$\mathcal{L}_{con} = \sum ReLU(|\hat{y} - y'| - \tau) \quad (8)$$

In this equation, \hat{y} is the predicted value of the physical parameter by ML model, and y' is the predicted value of the physical parameter by the physics-guide model. τ is the threshold parameter that is introduced because the physics-guide model is subjected to the physical understanding of the problem; therefore, it has biases and uncertainties. It is highly possible that the data contains physical processes that are not included in the physics-guide model and the ML model can learn the processes directly from the data. Therefore, threshold value allows for relaxing the penalization of physically inconsistent learning by considering the limitations of the physics-guide model. τ is an important parameter in the training of PGML and using a high value of threshold decreases the effect of physics guidance, and using a small value of threshold adds significant additional error to the loss function, which affects the convergence of ML model.

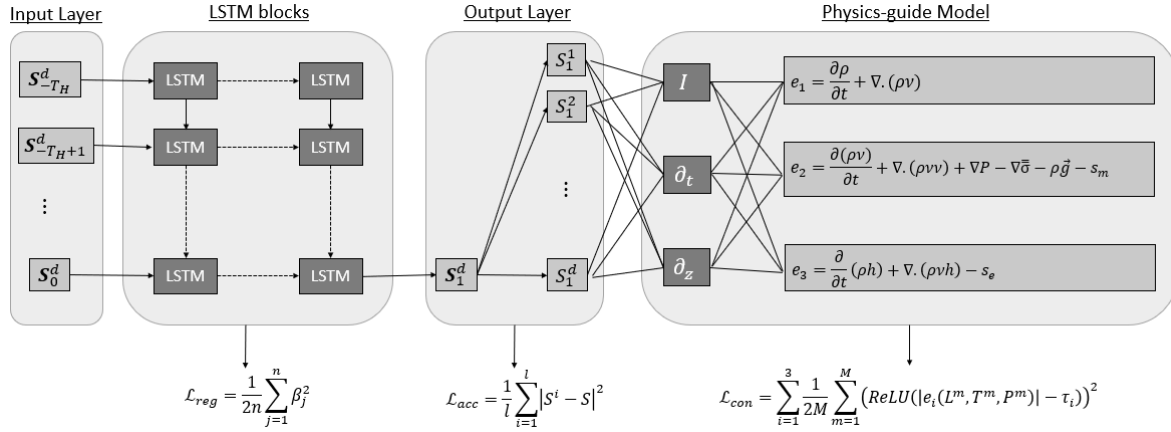


Figure 7: Architecture of prognosis model

4.3. Physics-guide model

GOTHIC is a thermal-hydraulics code; therefore, it solves mass, momentum, and energy conservation equations. The physics-guide model is a combination of mass, momentum, and energy conservation at the core region because T_{FCL} is highly dependent on the core channel. Figure 7 contains the differential forms of the conservation equations in the physics-guide model, however they are not applicable with this form. In order to calculate the physical inconsistencies, conservation equations are approximated with finite volume approach. Mass, momentum and energy conservation residuals of the physics-guide model are given in Equations 9-11 respectively. It must be noted here that the residuals can be calculated with the input features given in Table 2 where values at $t + 1$ are the ML model prediction and values at t are the inputs to the ML model.

In order to determine the τ for each conservation equation, Equations 9-11 are tested in SR database. The maximum of residuals is defined as the threshold for the conservation equation in the physical consistency loss function.

$$R_1 = V \frac{\rho_z^{t+1} - \rho_z^t}{\Delta t} + (\rho v A)_z^{t+1} - (\rho v A)_z^{t+1} \quad (9)$$

$$R_2 = V \frac{(\rho v)_z^{t+1} - (\rho v)_z^t}{\Delta t} + (\rho v A)_{z+1}^{t+1} - (\rho v A)_z^{t+1} + A(P_{z+1}^{t+1} - P_z^{t+1}) + V \rho_z^{t+1} g + \left(2\mu_{z+1}^{t+1} \frac{v_z^{t+1} - v_{z+1}^{t+1}}{\Delta z} A \right)_{z+1}^{t+1} - \left(2\mu_z^{t+1} \frac{v_z^{t+1} - v_{z+1}^{t+1}}{\Delta z} A \right)_z^{t+1} + A \frac{f_{fric} \Delta z}{2D_h} \rho_z^{t+1} v_z^{t+1} v_{z+1}^{t+1} \quad (10)$$

$$R_3 = V \frac{(\rho h)_z^{t+1} - (\rho h)_z^t}{\Delta t} + (\rho v A h)_{z+1}^{t+1} - (\rho v A h)_z^{t+1} - Q^{t+1} \quad (11)$$

4.4. Prognosis models

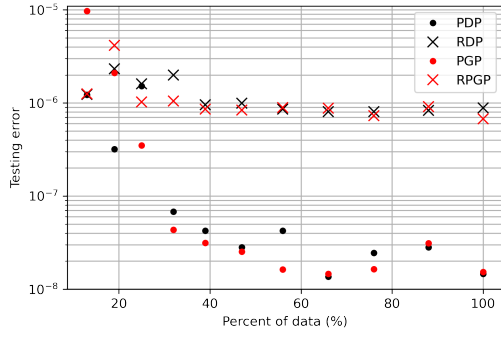
Based on the contributors to the loss function, four different prognosis models are developed with PGML based

surrogate function. Pure data-driven prognosis (PDP) only considers accuracy term in the loss function, regularized data-driven prognosis (RDP) considers accuracy and simplicity terms in the loss function, physics-guided data-driven prognosis (PGP) considers accuracy and consistency terms in the loss function, and regularized, physics-guided data-driven prognosis (RPGP) considers accuracy, simplicity and consistency terms in the loss function.

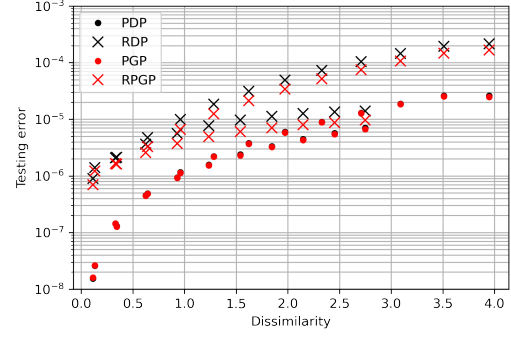
The training data is constructed from the SR database, and in the training database, PSP#1 torque ramps down to 66.7%, and PSP#2 torque ramps up to 16 end values between [105%, 180%], and PSP#2 torque ramping starts at 16 times between [50s, 130s]. There are 256 episodes used for training, and remaining episodes in the SR database are used for testing purposes. The sampling frequency of the episodes in the training database is 1second, and each episode is run for 200seconds. The time-series data contains the values of all input features for the entire length of the simulation, but for the training step, the training database is divided into mini sequences that label the next values of an input sequence. The ML model is trained with the mini-sequences, that provide a single-step prediction during the training.

5. Analysis of case study

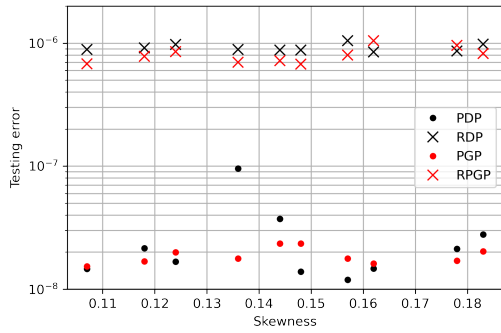
The convergence analysis is performed on the testing database. The testing database is constructed considering the boundary conditions of the training database. The testing database is also taken from the SR database, and the PSP#1 torque ramps down to 66.7%, and PSP#2 torque ramps up to 7 end values between [105%, 180%], and PSP#2 torque ramping starts at 7 times between [50s, 130s]. Testing database PSP#2 torque end value and ramping time are different from the training database, but they are selected homogeneously from the range of the training database. There are 49 episodes in the testing database. There are two types of errors are presented, testing error and prognosis error.



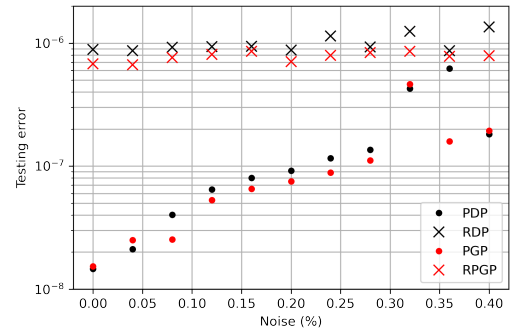
(a) Training data size



(b) Training data representativeness



(c) Training data balance



(d) Training data noise

Figure 8: Convergence of testing errors

The testing error is the error of predicting each time-step of the data for all of the reactor state variables. It has the same properties of the training error; therefore, the testing error indicates how well the ML model is trained for given settings. The convergence analysis on testing error is an indicator that the developed ML model is a good proxy for best hypothetical estimate function. In convergence analysis, the ML-based surrogate functions are trained on training databases with properties perturbed to satisfy conditions given in Table 1. The plots of the testing error for each prognosis model are given in Figure 8.

The testing error decreases with increasing sample size and increases with increasing dissimilarity and noise. The skewness of the training data is found to be having minimal effect on the testing error. The convergence analysis of testing error suggest that PDP and PGP have enough training data so that the optimal model is developed, and the prediction should be accurate for the testing conditions. Restricting complexity of the surrogate function prevented to learn RDP and RPGP dynamic patterns in the time-series data, and these models performed worse than PDP and PGP.

The prognosis error is the error of predicting T_{FCL} in the whole transient with multi-step prediction scheme. It is fundamentally different than training error; however, it is required to quantify how well the prognosis model is

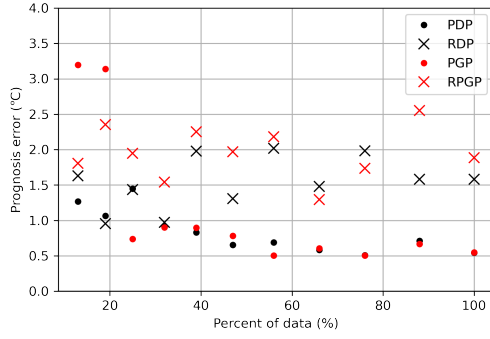
developed in an autonomous control framework. The plots of convergence of prognosis error are given in Figure 9.

The prognosis error shows similar trends of testing error. An important difference is that the prognosis error is not significantly affected by the presence of noise in the training data, indicating that the multi-step prediction scheme is more robust to data noise than single-step prediction. PDP and PGP prognosis errors are converged with increasing training data that can qualify the expected prognosis error on testing conditions that are similar to the training data. PGML-based surrogate function performs better than standalone ML-based surrogate function in prognosis model development.

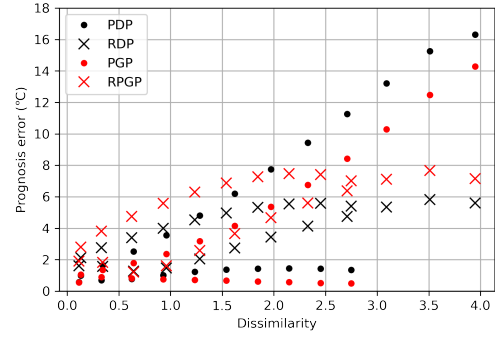
To expand the analysis of the prognosis models, they are tested against potential requirements of an autonomous control framework. These requirements are identified as adaptability, uncertainty, and consistency that all of them are explained in their respective sub-sections.

5.1. Adaptability analysis

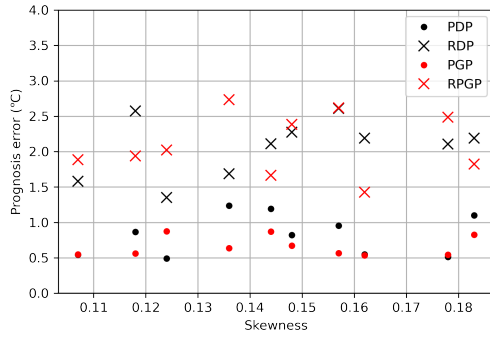
The requirement for adaptability can be summarized as the ability of prognosis model to make accurate predictions in different initial and boundary conditions. For adaptability analysis, three cases are defined, and prognosis errors are calculated for each case. These cases can be summarized as interpolation case, extrapolation case, and different ramping case. Interpolation case has the same boundary conditions



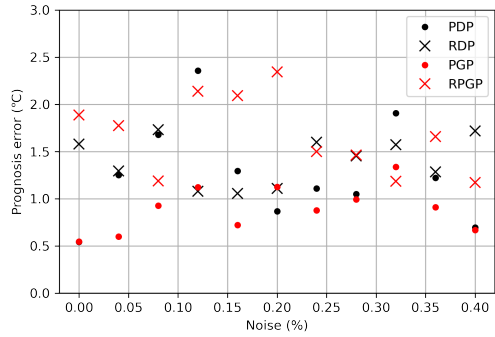
(a) Training data size



(b) Training data representativeness



(c) Training data balance



(d) Training data noise

Figure 9: Convergence of prognosis errors

of the training database and testing database at which the convergence analysis is performed. There are 651 episodes in the interpolation case.

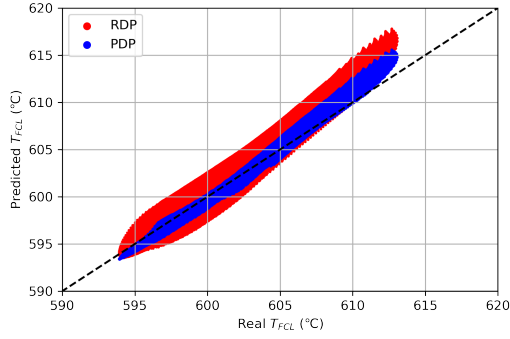
Extrapolation case has a similar ramping behavior as the testing database, but the boundary conditions are different. In extrapolation case, the testing database is constructed from the SR database and PSP#1 torque ramps down to 33.4%. There are 958 episodes in the extrapolation case.

Different ramping case has different ramping behavior that the testing database, and the boundary conditions are also different. In this case, the testing database is constructed from MR database, and PSP#1 torque ramps down to 20 end values between [34%, 95%]. There are 2789 episodes in the different ramping case, and this case is proposed to evaluate the scalability of prognosis models in conditions that both ramping behavior and boundary conditions are different.

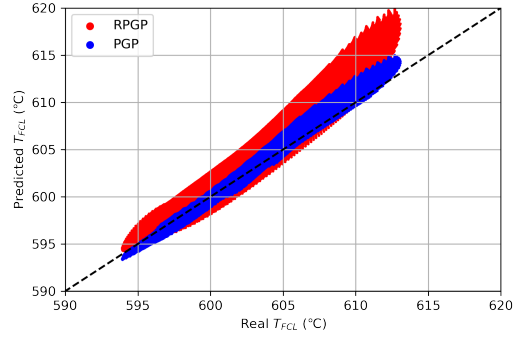
T_{FCL} prediction of all prognosis models for all cases are given in Figure 10. In the convergence analysis, it is observed that PDP and PGP converge for the given training and testing conditions, implying that these models are well developed, and they are reliable for testing conditions. The interpolation case is highly similar to the testing database; therefore, PDP and PGP have successfully captured the transients in the interpolation case. In the view of adaptability, PDP and PGP are adaptable to scenarios similar to the training conditions. The extrapolation case boundary conditions are different

from the training database, and all prognosis models have higher prognosis errors in this case. This is compatible with the observations of the convergence analysis that prognosis error increases with increasing dissimilarity. RDP and RPGP are regularized, and they are more resistant to extrapolation conditions. The different ramping case has a different ramping behavior with boundary conditions close to interpolation or extrapolation case. However, despite the different ramping behavior, prognosis models capture the transient of the different ramping behavior. In the view of adaptability, all prognosis models are adaptable to moving ramp condition, and PDP and PGP perform better with the tending to overestimate the transient.

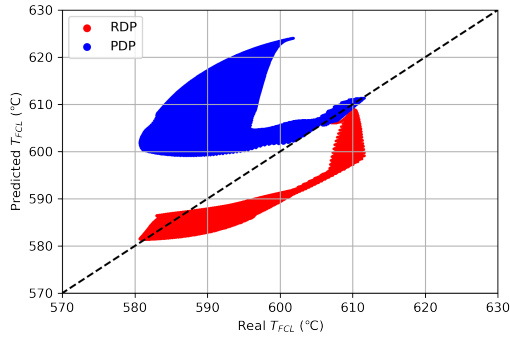
T_{FCL} prediction errors of prognosis models for all episodes in all cases are summarized in Table 3. Overall, PGP has lower prognosis error than PDP for all cases. This implies that including physics-guide model in training improves the learning of standalone ML-based prognosis. However, including the physics-guide model to the training of RDP had the opposite effect, by increasing the prognosis error for all cases. RPGP has more constraint during the training than other prognosis models; therefore, it is possible that the training data may not be enough to apply simplicity and consistency together to the loss function.



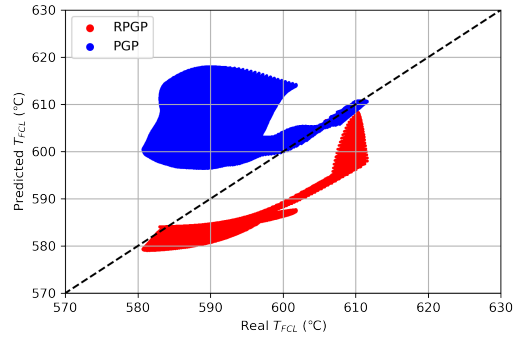
(a) PDP and RDP in interpolation case



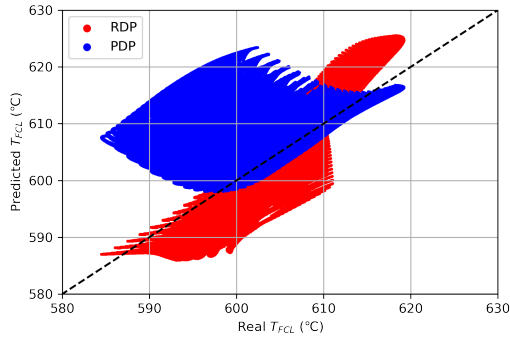
(b) PGP and RPGP in interpolation case



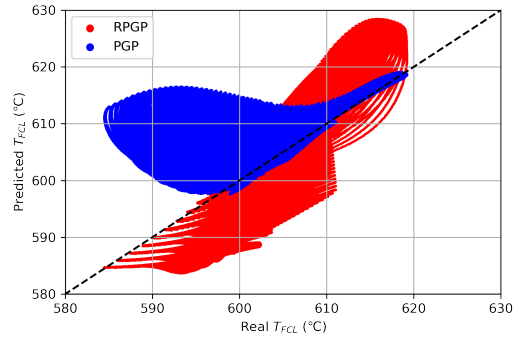
(c) PDP and RDP in extrapolation case



(d) PGP and RPGP in extrapolation case



(e) PDP and RDP in different ramping case



(f) PGP and RPGP in different ramping case

Figure 10: T_{FCL} predictions of prognosis models

Table 3

T_{FCL} prediction errors of prognosis models for all episodes in all cases

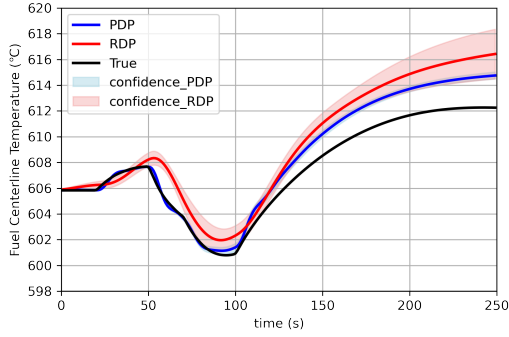
	PDP	RDP	PGP	RPGP
Interpolation	0.569°C	1.607°C	0.567°C	1.898°C
Extrapolation	16.311°C	5.604°C	14.276°C	7.161°C
Different ramping	5.949°C	4.148°C	4.553°C	5.572°C

5.2. Uncertainty Analysis

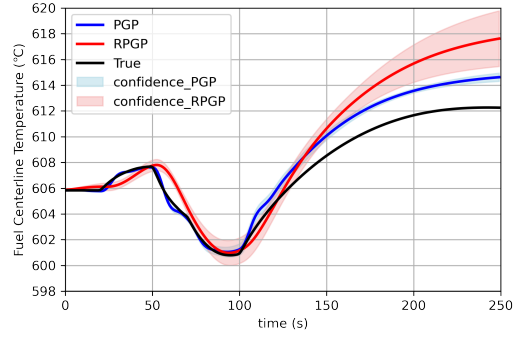
The requirements for uncertainty can be summarized as the prognosis model's ability to predict the transient and

quantify the uncertainty of the prediction. The uncertainties of prognosis models' predictions also quantified in interpolation, extrapolation, and different ramping cases, with the method described in step 5 of the methodology.

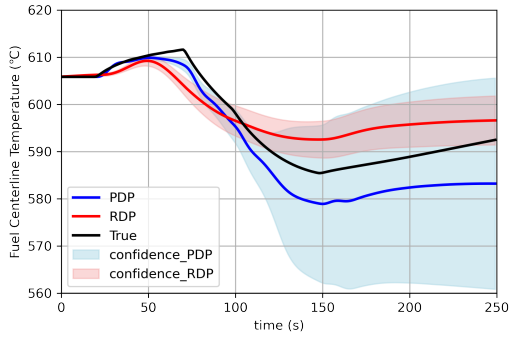
T_{FCL} predictions of prognosis models with uncertainty are plotted in Figure 11 for the worst episodes in each case. The worst episode is defined as the episode at which PDP has the largest prognosis error in each of the case database. For the interpolation case, all the developed PDP and PGP have made similar predictions; therefore, the standard deviation of the predictions are very low. RDP and RPGP have lower model complexity, and training of these models is



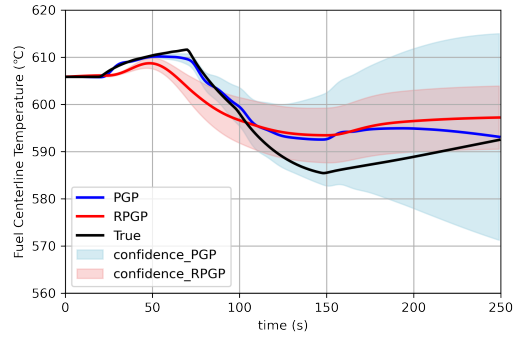
(a) PDP and RDP in interpolation case



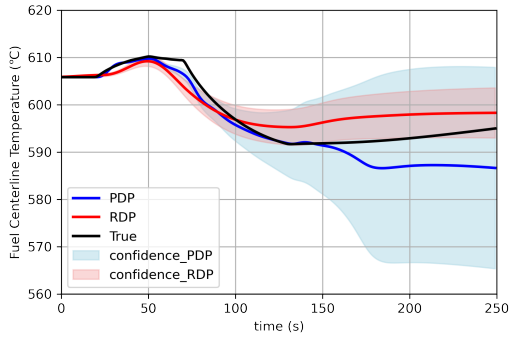
(b) PGP and RPGP in interpolation case



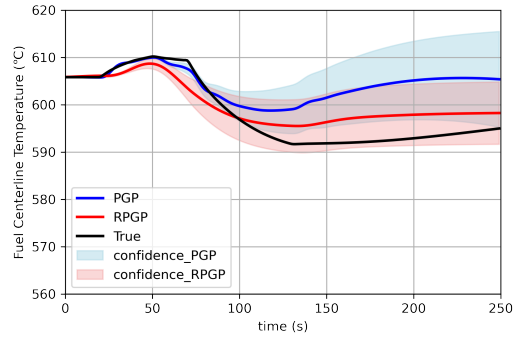
(c) PDP and RDP in extrapolation case



(d) PGP and RPGP in extrapolation case



(e) PDP and RDP in different ramping case



(f) PGP and RPGP in different ramping case

Figure 11: T_{FCL} predictions of prognosis models in worst episodes

more restricted because of the Algorithm 1. Therefore, RDP and RPGP are more sensitive to initial model parameters and hyperparameters and have higher prognosis error and uncertainty in the interpolation case. Prediction uncertainty increases with increasing extrapolation, especially for PDP and PGP. RDP and RPGP have lower model complexities due to regularization during raining therefore, the model variances of these models are lower. Therefore, the prognosis error and uncertainty of RDP and RPGP are better than PDP and PGP in extrapolation case. For different ramping case, the performance of PGP is close to RDP and RPGP, and

PGP can better capture the boundary condition changes in the episodes compared to PDP.

Mean and standard deviation of T_{FCL} prediction errors of prognosis models for all episodes in all cases are given in Table 4. Including the physics-guide model to PDP reduced the prediction uncertainty for all cases; therefore, PGP performs better than PDP for all cases. However, similar to the observation made in adaptability analysis, including the physics-guide model to RDP increased the mean error and uncertainty for extrapolation and different ramping cases. This observation further supports that the training data may not be enough to apply simplicity and consistency together to

Table 4

Mean and standard deviation T_{FCL} prediction errors of prognosis models for all episodes in all cases

	PDP		RDP		PGP		RPGP	
	$\hat{S}(^{\circ}C)$	$\sigma_s(^{\circ}C)$	$\hat{S}(^{\circ}C)$	$\sigma_s(^{\circ}C)$	$\hat{S}(^{\circ}C)$	$\sigma_s(^{\circ}C)$	$\hat{S}(^{\circ}C)$	$\sigma_s(^{\circ}C)$
Interpolation	0.804	0.134	1.569	0.575	0.753	0.125	1.438	0.382
Extrapolation	18.129	15.244	7.503	3.579	15.812	9.646	8.605	5.340
Different ramping	6.024	3.991	4.522	1.454	4.718	1.991	4.442	2.000

the loss function. The behavior of mean prediction error for each prognosis model for all databases shows similarity to the deterministic prognosis errors calculated in convergence analysis. This implies that the observations of convergence analysis are valid for the ensemble model.

5.3. Consistency Analysis

Prognosis models adopt a multi-step prediction scheme, and one of the requirements of multi-step prediction scheme is to conserve physics at each prediction step. In this section, consistency analysis is performed for prognosis models where consistency is defined as the energy conservation at the core channel. Only energy conservation is considered in consistency analysis because energy conservation directly affects the T_{FCL} prediction; therefore, the energy conservation error better represents the consistency of prognosis models. For consistency analysis, inconsistency is defined as the energy conservation error at the core channel. The physics-guide model is developed to guide the training of the ML algorithm, but it can also be used to quantify the inconsistency of the prediction of prognosis models.

To calculate the inconsistencies of prognosis models, the same approach in uncertainty analysis is adopted. The developed surrogate functions of prognosis models to quantify prediction uncertainty in ensemble averaging are also used to calculate mean and standard deviation of inconsistencies. With this approach, it is possible to plot prognosis errors and inconsistencies on the same graph with their error bars. The inconsistencies of prognosis models for interpolation, extrapolation and different ramping cases are given in Figure 12. PDP and PGP have low prognosis errors, and PGP also slightly have lower inconsistency than PDP. However, although RPGP has a lower prognosis error than RDP, there is no improvement for the inconsistency. For the extrapolation case, despite the prognosis error of PGP is higher than RDP and RPGP, the inconsistency of PGP is the lowest. This observation suggests that low physical inconsistency does not imply low prognosis error. However, it is observed that having low inconsistency has a tendency to lower the prognosis error.

6. Conclusions

This study is motivated by the need for fast-running simulation tools for autonomous decision-making systems, focusing on prognosis. Prognosis is the process of predicting future conditions of a system or equipment based on present signs and symptoms of a fault. Prognosis is an important step in autonomous decision-making by allowing to predict

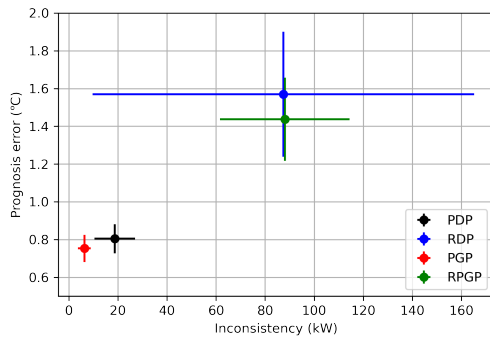
future reactor states for possible candidate control strategies so that the outcomes can be evaluated to determine the best control strategy. The prognosis model is a time-series forecast model with initial and boundary conditions. From the modeling point of view, the prognosis model is not different from system codes. However, system codes take some time to complete the simulations, and for fast-running simulation, the prognosis model needs to be a machine learning (ML) model developed with the simulation results of the system code. This study explored Recurrent Neural Network to develop surrogate function of system code for prognosis model that allows fast-running simulation which cannot be achieved with system codes.

In this study, ML-based prognosis models are formulated, developed, and demonstrated for a fast-running simulation tool. The prognosis models can predict fuel centerline temperature with associated prediction uncertainty based on initial and boundary conditions. A development and assessment framework is proposed to develop prognosis models, and four different surrogate functions are developed as a case study for potential prognosis models considering standalone ML and physics-guided machine learning (PGML) approaches. ML model performance is highly dependent on the training data itself and developed prognosis models are tested in imperfect training data conditions. Imperfect training data is hypothesized to be due to training data is not enough, training data is not representative, training data is not balanced, and training data is noisy. The prognosis models are tested on the imperfect training conditions, and prognosis models without model complexity regularization performed better implying restricting the model complexity reduces the prognosis model's ability to learn dynamic patterns in the time-series data

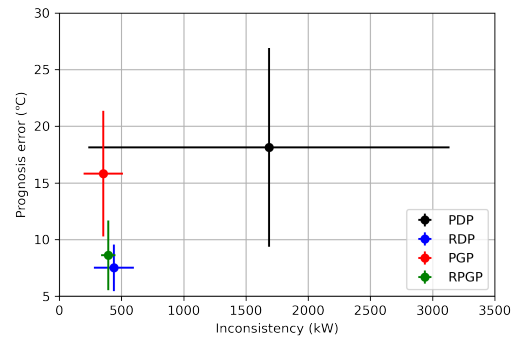
PGML is a novel method to improve the standalone ML approach. PGML requires a physics-guide model that represents the knowledge about the problem, and the relationship among the input and output of the ML model can be partially described in the form of explicit equations. In this study, a physics-guide model is developed for physics-guided prognosis models considering the conservation of mass, momentum, and energy at the core channel. PGML approach improves the accuracy, reduces the uncertainty and inconsistency of the prognosis models. The results obtained in this work demonstrate the potential advancements of the hybrid approach for prognosis that a physics-based model guiding the training of the ML model.

Acknowledgements

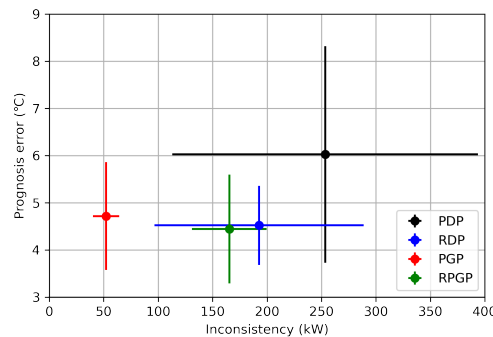
This work is fully supported by the U.S. Department of Energy, Advanced Research Projects Agency-Energy (ARPA-E), Modeling-Enhanced Innovations Trailblazing Nuclear Energy Reinvigoration (MEITNER) program (DE-AR0000976) under the project entitled "Development of a Nearly Autonomous Management and Control System for Advanced Reactors". Thanks to Zachry Nuclear Engineering, Inc. for providing GOTHIC license, and Pascal



(a) Interpolation case



(b) Extrapolation case



(c) Different ramping case

Figure 12: Prognosis errors and inconsistencies with error bands

Rouxelin of North Carolina State University for generating and organizing the databases. Special thanks to Linyu Lin of Idaho National Laboratory for perspectives on NAMAC and for inspiring discussion.

References

- [1] M. Rodríguez, J. Bermejo-Alonso, C. Hernandez Corbato, R. Sanz, Ontology-driven description and engineering of autonomous systems: application to process systems engineering, in: I. D. L. Bogle, M. Fairweather (Eds.), 22nd European Symposium on Computer Aided Process Engineering, Vol. 30 of Computer Aided Chemical Engineering, Elsevier, 2012, pp. 717–721. doi:<https://doi.org/10.1016/B978-0-444-59520-1.50002-6>.
- [2] S. Cetiner, P. Ramuhalli, Transformational challenge reactor autonomous control system framework and key enabling technologies (2019). doi:[10.2172/1530084](https://doi.org/10.2172/1530084).
- [3] L. Lin, P. Athe, P. Rouxelin, M. Avramova, A. Gupta, R. Youngblood, J. Lane, N. Dinh, Development and assessment of a nearly autonomous management and control system for advanced reactors, *Annals of Nuclear Energy* 150 (2021) 107861. doi:<https://doi.org/10.1016/j.anucene.2020.107861>.
- [4] L. Lin, P. Athe, P. Rouxelin, M. Avramova, A. Gupta, R. Youngblood, J. Lane, N. Dinh, Digital-twin-based improvements to diagnosis, prognosis, strategy assessment, and discrepancy checking in a nearly autonomous management and control system, *Annals of Nuclear Energy* 166 (2022) 108715. doi:<https://doi.org/10.1016/j.anucene.2021.108715>.
- [5] EPRI, Gothic thermal hydraulic analysis package qualification report.
- [6] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent neural network regularization (2014). URL <https://arxiv.org/abs/1409.2329>
- [7] A. Karpatne, G. Atluri, J. H. Faghmous, M. S. Steinbach, A. Banerjee, A. R. Ganguly, S. Shekhar, N. F. Samatova, V. Kumar, Theory-guided data science: A new paradigm for scientific discovery, *CoRR abs/1612.08544* (2016). arXiv:1612.08544. URL <http://arxiv.org/abs/1612.08544>
- [8] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, *CoRR abs/1711.10561* (2017). arXiv:1711.10561. URL <http://arxiv.org/abs/1711.10561>
- [9] Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics* 378 (2019) 686–707. doi:<https://doi.org/10.1016/j.jcp.2018.10.045>.
- [10] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, *Science* 367 (6481) (2020) 1026–1030. arXiv:<https://www.science.org/doi/pdf/10.1126/science.aaw4741>, doi:[10.1126/science.aaw4741](https://doi.org/10.1126/science.aaw4741).
- [11] X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. S. Steinbach, V. Kumar, Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles, *CoRR abs/2001.11086* (2020). arXiv:2001.11086. URL <https://arxiv.org/abs/2001.11086>
- [12] A. Karpatne, W. Watkins, J. S. Read, V. Kumar, Physics-guided neural networks (PGNN): an application in lake temperature modeling, *CoRR abs/1710.11431* (2017). arXiv:1710.11431. URL <http://arxiv.org/abs/1710.11431>
- [13] V. N. Vapnik, An overview of statistical learning theory, *IEEE transactions on neural networks* 10 5 (1999) 988–99.

- [14] P. Niyogi, F. Girosi, On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions, *Neural Computation* 8 (4) (1996) 819–842. doi:10.1162/neco.1996.8.4.819.
- [15] R. Ge, S. M. Kakade, R. Kidambi, P. Netrapalli, The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Vol. 32, Curran Associates, Inc., 2019.
URL <https://proceedings.neurips.cc/paper/2019/file/2f4059ce1227f021edc5d9c6f0f17dc1-Paper.pdf>
- [16] E. J. Keogh, M. J. Pazzani, Scaling up dynamic time warping for datamining applications, in: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00*, Association for Computing Machinery, New York, NY, USA, 2000, p. 285–289. doi:10.1145/347090.347153.
URL <https://doi.org/10.1145/347090.347153>
- [17] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, 1st Edition, Chapman Hall/CRC, 2012.
- [18] L. Zhu, N. Laptev, Deep and confident prediction for time series at uber, in: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2017. doi:10.1109/icdmw.2017.19.
URL <https://doi.org/10.1109%2Ficdmw.2017.19>
- [19] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: M. F. Balcan, K. Q. Weinberger (Eds.), *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48 of *Proceedings of Machine Learning Research*, PMLR, New York, New York, USA, 2016, pp. 1050–1059.
URL <https://proceedings.mlr.press/v48/gal16.html>
- [20] H. Planchon, R. Singer, D. Mohr, E. Feldman, L. Chang, P. Betten, The experimental breeder reactor ii inherent shutdown and heat removal tests — results and analysis, *Nuclear Engineering and Design* 91 (3) (1986) 287–296. doi:https://doi.org/10.1016/0029-5493(86)90082-8.
URL <https://www.sciencedirect.com/science/article/pii/S0029549386900828>
- [21] J. W. Lane, J. M. Link, J. M. King, T. L. George, S. W. Claybrook, Benchmark of gothic to ebr-ii shrt-17 and shrt-45r tests, *Nuclear Technology* 206 (7) (2020) 1019–1035. arXiv:https://doi.org/10.1080/00295450.2019.1698896, doi:10.1080/00295450.2019.1698896.
URL <https://doi.org/10.1080/00295450.2019.1698896>
- [22] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural Comput.* 9 (8) (1997) 1735–1780. doi:10.1162/neco.1997.9.8.1735.
URL <https://doi.org/10.1162/neco.1997.9.8.1735>
- [23] H. Sak, A. W. Senior, F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, in: *INTERSPEECH*, 2014, pp. 338–342.
- [24] X. Ma, Z. Tao, Y. Wang, H. Yu, Y. Wang, Long short-term memory neural network for traffic speed prediction using remote microwave sensor data, *Transportation Research Part C: Emerging Technologies* 54 (2015) 187–197. doi:https://doi.org/10.1016/j.trc.2015.03.014.
URL <https://www.sciencedirect.com/science/article/pii/S0968090X15000935>
- [25] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL <http://arxiv.org/abs/1412.6980>
- [26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, *CoRR* abs/1912.01703 (2019). arXiv:1912.01703.
URL <http://arxiv.org/abs/1912.01703>

Highlights

Development and assessment of a reactor system prognosis model with physics-guided machine learning

Anil Gurgen, Nam T. Dinh

- Nearly Autonomous Management and Control System
- Time-series forecast with reactor system prognosis model
- Surrogate function development using physics-guided machine learning