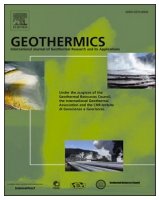
Geothermics 116 (2024) 102824

Contents lists available at ScienceDirect

Geothermics

journal homepage: www.elsevier.com/locate/geothermics



Physics-Guided Deep Learning for Prediction of Energy Production from Geothermal Reservoirs

Zhen Qin [a], Anyue Jiang [a], Dave Faulder [b], Trenton T. Cladouhos [b], Behnam Jafarpour [a,*]

[a] Viterbi School of Engineering, University of Southern California, USA
[b] Cyrq Energy Inc., USA

ARTICLE INFO

Keywords:
Physics-Guided Neural Networks
Recurrent Neural Network
Deep Learning
Machine Learning
Geothermal Reservoir

ABSTRACT

Predictive models are traditionally used for the development and management of geothermal reservoirs. While field operation optimization based on physics-based simulations offers dependable strategies, simulation models require detailed descriptions of reservoir conditions and properties and entail extensive computational efforts. As efficient alternatives to traditional physics-based simulation, data-driven predictive models such as deep learning-based models can provide fast predictions to facilitate complex iterative tasks that otherwise entail high computation time. However, purely data-driven models that are trained using limited data often provide physically inconsistent predictions and fail to generalize beyond the training data. This has important consequences in optimization applications where, during optimization, the well control strategies are likely to fall beyond the training data. These limitations undermine the suitability and strength of data-driven models in scientific and engineering applications, where the amount of data is typically limited but physical laws are well-established and frequently used. To address the above challenges, we propose a novel physics-guided machine learning model by incorporating the general structure of the physics-based equations into deep learning models. A typical approach for incorporating physics is adding physics-based constraints in the loss function to regularize the trainable parameters. However, this approach does not exploit or adapt the architecture of the neural network. In this work, the architecture of the proposed recurrent neural networks (RNN) is designed to represent the differential equations of the subsurface flow system. We present the physics-guided RNN models in detail and demonstrate their connection to the underlying differential equations describing the fluid flow physics. We investigate the prediction performance of the proposed models by first applying them to controlled example to evaluate their extrapolation power, before using them with simulated and field datasets.

1. Introduction

Simulation models that can provide accurate long-term production predictions are traditionally used for production optimization of geothermal reservoirs. This task is commonly referred to as model-based optimization or model predictive control. While field operation optimization based on physics-based simulation offers reliable operation strategies, the simulation model requires reliable and detailed descriptions of reservoir conditions and properties and entails extensive computational efforts. As efficient alternatives, data-driven predictive models facilitate the implementation of complex iterative workflows that are computationally demanding to use with full the simulation model. In recent years, deep learning models have become increasingly popular in the subsurface flow domain for various tasks such as time-series production prediction (Tian and Horne, 2017; Gudmundsdottir and Horne, 2020; Jiang et al., 2021; Shi et al., 2021) and inverse modeling (Laloy et al., 2017; Razak and Jafarpour, 2020; Jiang & Jafarpour, 2021a, 2021b). Deep learning models are powerful in extracting complex statistical patterns from training data and using them to generate fast predictions based on the learned input-output relationships. However, purely data-driven deep learning models have their limitations. One major drawback of data-driven models is their limited ability to extrapolate beyond the training data (Karniadakis et al., 2021; Willard et al., 2021).

Extrapolation involves using a model that is trained on an original training dataset to predict values outside the training data range. Time-series forecasting is a type of extrapolation where the implicit variable "time" is out of the training range, especially when the time-series output

* Corresponding author.
E-mail address: jafarpou@usc.edu (B. Jafarpour).

https://doi.org/10.1016/j.geothermics.2023.102824
Received 1 May 2023; Received in revised form 7 August 2023; Accepted 28 August 2023
Available online 11 October 2023

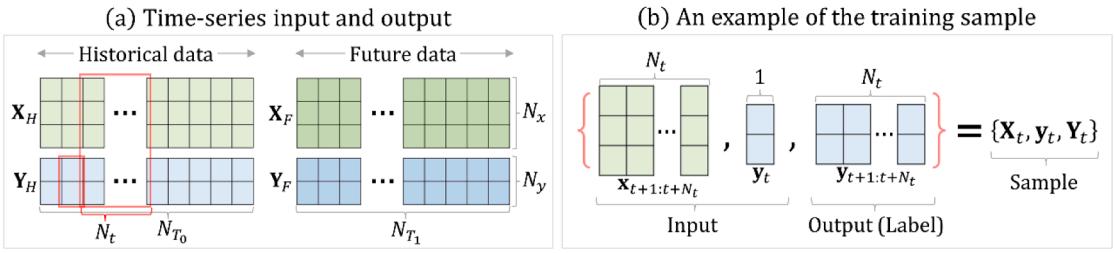


Fig. 1. Illustration of (a) time-series input and output and (b) an example of the training sample. The training sample is obtained by slicing the historical input $\mathbf{X}_H$ and output $\mathbf{Y}_H$, and consists of control variables $\mathbf{X}_t$ and initial states $\mathbf{y}_t$ as the input to the deep learning model, and the future states $\mathbf{Y}_t$ as the label. The tensors within the red block in (a) refer to a single training sample shown in (b)

Table 1
Summary of hyperparameters for different experimental cases.

	Training Set (Time Step)	Validation Set (Time Step)	Testing Set (Time Step)	Learning Rate
Test Cases	1400	100	2500	0.0035
Simulated Temperature	150, 250, 350, 450, 550	50	455, 555, 655, 755, 855	0.005
Simulated BHP	250, 350, 450, 550	50	455, 555, 655, 755	0.005
Field Case	1050	50	1000	0.0005

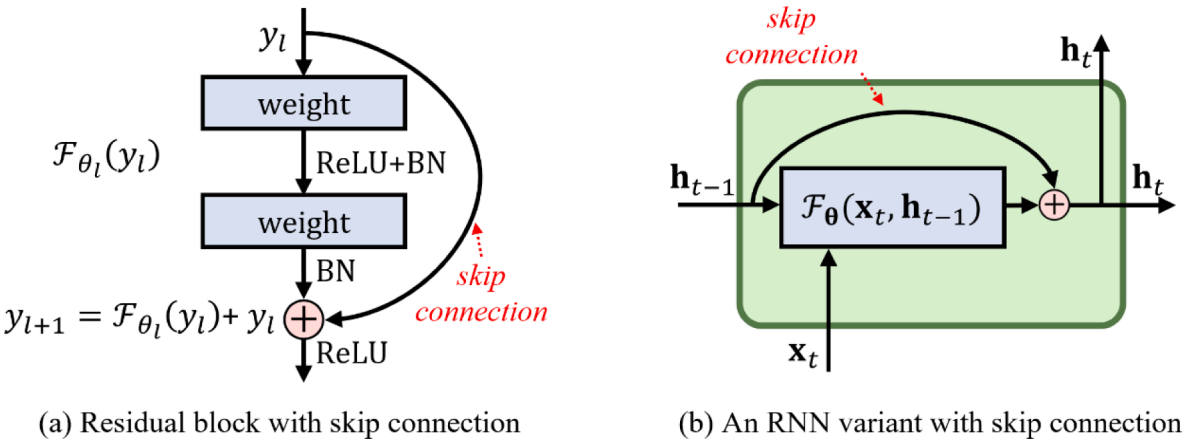
has non-stationary behavior. Non-stationary time series can have statistical properties, such as mean and variance, that change over time. In geothermal reservoirs, predicting a declining temperature (or enthalpy) is an extrapolation task. Therefore, predicting future production temperature trends based on historical data constitutes an extrapolation problem.

Extrapolation can also commonly occur when the input variable to the reservoir model is high-dimensional, making it difficult to cover the entire range of input variability in the training data (Balestriero et al., 2021). In optimization tasks, mass flow rate or bottom-hole pressure (BHP) can be high-dimensional, where each element represents the control variables of different wells over different time steps (Qin et al., 2023). Another example if high-dimensional problems is inverse modeling or uncertainty quantification tasks, where uncertain reservoir properties (e.g., permeability map) are inherently high-dimensional, and pose a challenging extrapolation problem. Generating more training samples could alleviate the extrapolation issue related to high-dimensional learning. However, unlike the vast amount of accessible data in computer science applications, data in scientific and engineering applications is limited due to the cost of sampling in the field and time-consuming simulation that needs to be run to generate labeled (simulated) data.

In geothermal reservoir engineering, conventional predictive models that are based on dynamic production history include numerical reservoir simulation and lumped-parameter models (Ciriaco et al., 2020). While numerical simulation models provide reliable predictions, their high runtime makes them prohibitive for complex iterative workflows, such as optimization, inverse modeling, and uncertainty quantification. Lumped parameter models leverage the physics of the subsurface system but are built based on a coarsely discretized grid system and/or simplified governing equations (Axelsson, 1989; Alkan and Satman, 1990; Sarak et al., 2005; Tureyen and Akyapı, 2011). These models typically contain a smaller number of homogeneous grid blocks and provide faster predictions compared to high-fidelity simulation models. Similar works in the petroleum domain include the capacitance-resistance model (CRM) (Yousef et al., 2006; Holanda et al., 2018) and methods that resemble streamline models, such as the Flow-Network model (Lerlertpakdee et al., 2014) and interwell numerical simulation model (INSIM) (Zhao et al., 2015, 2016). Predictive models with simplifications are typically referred to as physics-based or reduced-physics proxy (or surrogate) models.

Proxy models are designed to provide fast predictions at the cost of marginal reduction in accuracy/fidelity. They are often built to replace high-fidelity reservoir simulation in computationally complex workflows. Proxy models can be classified into data-driven (statistical), reduced-physics, and reduced-order approaches (Zubarev, 2009; Asher et al., 2015). Linear regression method and its extensions, as a common type of data-driven proxy model, have been applied to a wide range of applications for geothermal energy, such as well placement (Chen et al., 2015; Schulte et al., 2020), field operation optimization, and resource assessment (Quinao and Zarrouk, 2018). In recent years, deep learning models, as a more powerful data-driven predictive approach, have been increasingly applied to the prediction of production trends in subsurface flow systems. Recurrent neural networks (RNNs), as variants of neural networks (NN), are commonly used for predicting sequences of dynamical time-series data. RNNs have been applied to the geothermal domain for production performance prediction (Tian and Horne, 2017; Jiang et al., 2021; Shi et al., 2021) and tracer concentration prediction (Gudmundsdottir and Horne, 2020). The long short-term memory (LSTM) model (Hochreiter and Schmidhuber, 1997), as a popular variant of RNN, has also been used for the oil production prediction in the waterflooding problem (Kim and Durlofsky, 2021; Kim et al., 2022).

Purely data-driven models, including deep learning models, typically suffer from the limitations of extensive data requirements and poor generalizability beyond training data. These limitations undermine their suitability and strength in scientific and engineering applications, where well-established physical laws are frequently used. Anactive research area to improve the predictive power of data-driven models is the integration of physical principles and domain knowledge into neural networks (Karniadakis et al., 2021; Willard et al., 2021). A typical approach to incorporating the physics in a soft manner is embedding physical equations into the loss function of NNs during training to regularize the trainable parameters. Representative examples include the deep Galerkin method (DGM) (Sirignano and Spiliopoulos, 2018) and the physics-informed neural network (PINN) (Raissi et al., 2019). In the DGM and PINN, the solution $f(t, x)$ of the partial differential equation (PDE) is approximated as a deep neural network (DNN) model. The learning of the system is treated as an optimization problem where the objective function includes the mismatch losses and the residual errors of the PDE, boundary, and initial conditions. The PINN provides a flexible implementation for different physical systems and has been applied in the subsurface domain as the proxy model (Zhu et al., 2019; Wang et al., 2020). The emerging field of physics-informed machine learning (PIML) has shown promise in enhancing traditional numerical solvers for various scientific problems (Markidis, 2021; Cuomo et al., 2022). However, when it comes to applying PINN to the subsurface domain, there are certain challenges that need to be addressed. Specifically, the applicability of the PIML framework to high-dimensional data and more heterogeneous problems is still a concern (Muther et al., 2023). Furthermore, this approach requires known physical equations and does not explore the interpretability of the architecture of the neural network.



(a) Residual block with skip connection (b) An RNN variant with skip connection

Fig. 2. Illustration of the skip connection in (a) ResNet and (b) the proposed RNN model. (a) is the vanilla ResNet, where the skip connection (also called identity mapping) is first introduced. (b) is a general form of RNN-type variants, which is combined with skip connection.

Another fit-for-purpose approach is to design specialized NN architectures that implicitly embed physical knowledge by specifying node connections. By capturing physical dependencies among variables, the resulting architectures can efficiently learn PDEs or Ordinary Differential Equations (ODEs). The Residual Neural Network (ResNet) (He et al., 2016) is an example that builds the connection between deep learning models and numerical schemes of ODEs (E, 2017; Lu et al., 2018). The ResNet can be seen as the forward Euler discretization of ODEs (E, 2017). Lu et al. (2018) extend this work by linking other ResNet-type models to different numerical schemes of ODEs, such as Runge-Kutta schemes. The ResNet is also presented in a recurrent way to make predictions for multiple time steps (Qin et al., 2019). Another approach links convolutional neural networks (CNNs) with the finite difference stencil for the discretization of spatial derivatives (Long et al., 2018, 2019; Ruthotto and Haber, 2020). In their works, CNNs are built within ResNet architectures and designed to approximate PDEs. The above approaches can be approximately interpreted as ODEs and PDEs with little or no modification to the original architectures. Recent studies have modified architectures of RNNs based on domain knowledge with different scales of modification. For example, Sun et al. (2020) embed physical equations into an RNN unit for seismic inverse problems. Daw et al. (2020) modify the LSTM model to preserve the physical constraint for lake temperature modeling (i.e., monotonically decreasing behavior in temperature). Jiang et al. (2023) introduce a multiscale RNN model for extrapolating geothermal energy production time-series data. They train a two-part neural network model to capture long-term and short-term trends separately and use an output layer that combines the two parts to forecast energy production.

In this work, we propose two physics-guided RNN models to make fast and accurate long-term predictions using only historical data. The proposed models take well control variables (i.e., mass flow rate) as input and predict the production performance of wells (i.e., temperature and BHP), rather than the entire response of the reservoir. The input-output model reduces the dimension of features to learn and therefore requires fewer training samples. The proposed models are designed as variants of the RNN and gated recurrent unit (GRU) (Cho et al., 2014) by incorporating the numerical temporal discretization of the unknown physical equation. The proposed physics-guided RNNs are applied to handle two different types of extrapolations. In the first type, the output has a decreasing trend along the timeline and is non-stationary. In the second type of extrapolation, the input variable in the test set is outside the range seen by the data-driven model in the training set. In this paper, we demonstrate that by introducing modifications to the architectures of RNN and GRU to reflect the general form of the governing equations of the system, physics-guided RNNs can address both types of extrapolation problems. The integration of physics enables physics-guided RNNs to efficiently learn the dynamics from historical data and make more accurate long-term predictions over the lifetime of the geothermal reservoir project. Compared with physics-guided RNNs, the traditional RNN and GRU models cannot extrapolate the out-of-distribution inputs or outputs due to the nature of statistical learning and lack of causality.

In the remainder of this paper, we present the physics-guided RNNs in detail and demonstrate their connections to the physical equations. We investigate the extrapolation performance by first applying the models to a test case where the first type of extrapolation is considered. To investigate the second type of extrapolation, we also apply the physics-guided RNNs to a simulated dataset generated by a field-scale geothermal reservoir models. For the simulated temperature data, we compare the prediction results with a physics-based proxy model from our previous work (Qin et al., 2022). Lastly, we apply the proposed physics-guided RNNs to real field data to investigate their suitability for practical applications.

2. Methodology

2.1. Problem statement

We start by defining the problem statement, including the description of the input and output variables, as well as the relevant notations used throughout the paper. For a geothermal reservoir with N_{prod} production wells and N_{inj} injection wells, the task is to predict the production temperature or BHP (output variables) based on the mass flow rate of production and injection wells (input variables). The data-driven predictive model serves as an input-output model and is formulated as $y = f(x)$, where $f(\cdot)$ is the predictive model, and x and y are the input and output variables, respectively. Specifically, the input and output variables over N_T time steps are represented as multivariate time-series data, i.e., $\mathbf{X} \in \mathbb{R}^{N_x \times N_T}$ and $\mathbf{Y} \in \mathbb{R}^{N_y \times N_T}$, where $\mathbf{X}$ and $\mathbf{Y}$ are the normalized (0-1 min-max normalization) values of the mass flow rate and temperature (or BHP), respectively. The notations N_x and N_y denote the number of input and output variables, respectively. In our examples, we have $N_x = N_{prod} + N_{inj}$ and $N_y = N_{prod}$ for temperature (or $N_y = N_{prod} + N_{inj}$ for BHP).

The time-series data $\{\mathbf{X},\ \mathbf{Y}\}$ from a geothermal reservoir are used to train and test the recurrent neural network (RNN) as a deep learning-based predictive model. The model architecture accommodates physics constraints, as discussed in Section 2.2. The data sequence consists of N_T total time steps with a fixed time interval. The first N_{T_0} time steps of data are treated as historical data and used to generate training and validation sets. The remaining N_{T_1} time steps of data points are treated as future data and are used as testing set to investigate the long-term prediction performance of the model. The historical data $\{\mathbf{X}_H,\ \mathbf{Y}_H\}$ is

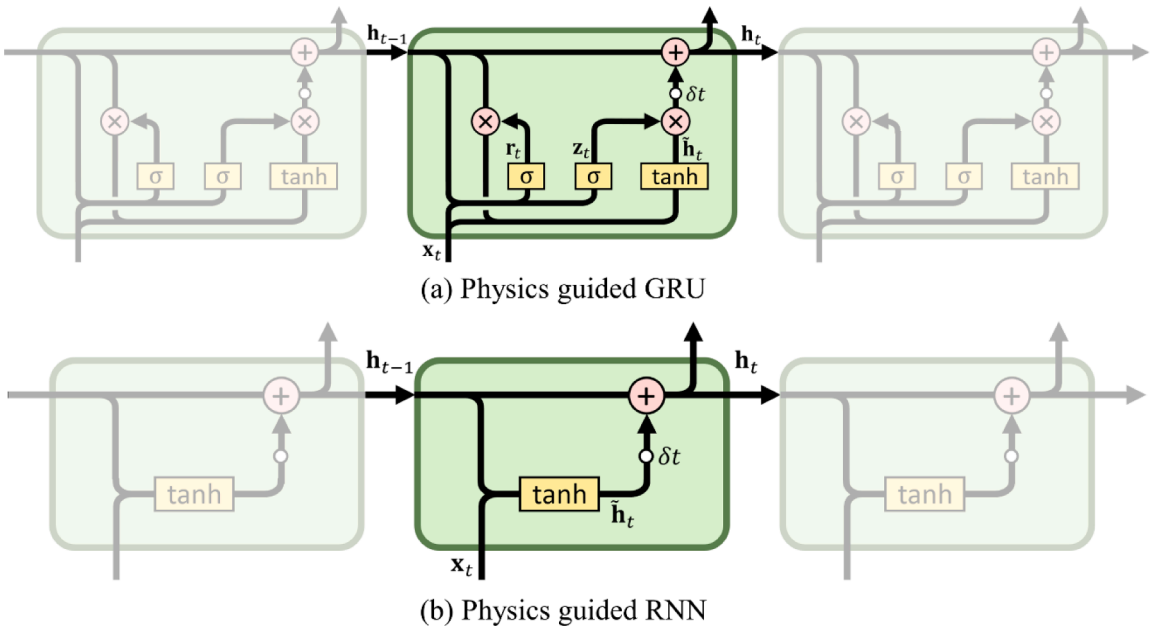


Fig. 3. Illustration of the unfolded representation of two physics-guided deep learning models: (a) physics guided GRU (*phy*-GRU) and (b) physics guided RNN (*phy*-RNN).

split into a set of samples for training and validating. As illustrated in Fig. 1, each sample consists of the following: input control variables $\mathbf{X}_t = \mathbf{x}_{t+1:t+N_t} \in \mathbb{R}^{N_x \times N_t}$, input initial states $\mathbf{y}_t \in \mathbb{R}^{N_y \times 1}$, and output labels $\mathbf{Y}_t = \mathbf{y}_{t+1:t+N_t} \in \mathbb{R}^{N_y \times N_t}$, where the subscript t denotes an arbitrary time step, and N_t is the prediction window for the RNN-type predictive model during training. During testing, the entire sequence of future input control variables $\mathbf{X}_F$, together with an initial state $\mathbf{y}_{T_0}$, are provided as the input to the trained model. The predictive model makes predictions over the time horizon with a length of N_{T_1} without taking any newer ground-truth label as initial state. The predicted output $\hat{\mathbf{y}}$ from the previous step is provided to the current step as the new initial state $\mathbf{y}_t$ until the end of the prediction horizon. For the experiments in this paper, the hyperparameters for training the predictive models, such as the learning rate and the number of time steps for the training, validation, and testing sets, are presented in Table 1.

2.2. Physics-guided deep learning model

We present the proposed physics-guided deep learning model in this section. Specifically, the architectures of the proposed deep learning models are designed as variants of RNNs to accommodate the physical constraints. We describe the details of the RNN models, including the standard RNN and GRU models. The connection between deep learning models and physical equations is then explained, followed by a description of the physics-guided RNN.

2.2.1. Gated recurrent unit

RNN is a class of neural networks designed to predict sequential data with temporal/dynamic behavior. RNNs have recurrent connections between the internal hidden states to process temporal sequences with arbitrary lengths. As common variants of RNN models, LSTM and GRU are designed to enhance the dependence on past states through gate mechanisms and by regulating the information flow. Therefore, GRU and LSTM typically outperform the regular RNN due to their complex architecture and the resulting higher learning capacity. Furthermore, GRU has shown comparable performance with LSTM (Chung et al., 2014), and better performance for small datasets (Yang et al., 2020). In this work, we only describe the standard RNN and GRU models that are used in developing our physics-guided RNNs.

For each time step t, a GRU unit receives the current input $\mathbf{x}_t \in \mathbb{R}^{N_x \times 1}$ and the pervious hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^{N_h \times 1}$ to generate the updated hidden state $\mathbf{h}_t$ through linear transformations, element-wise nonlinearities (through activation functions), and gate mechanisms. The superscript N_h denotes the dimension of the hidden state. The internal gate mechanism of a GRU unit consists of two gates, i.e., reset gate $\mathbf{r}_t \in \mathbb{R}^{N_h \times 1}$ and update gate $\mathbf{z}_t \in \mathbb{R}^{N_h \times 1}$. The reset gate chooses the proportion of the past information it needs to remember in the candidate hidden state $\tilde{\mathbf{h}}_t \in \mathbb{R}^{N_h \times 1}$ and is defined as

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r), \tag{1}$$

where $\sigma(\cdot)$ is the element-wise sigmoid activation function that scales the vector $\mathbf{r}_t$ to be between (0, 1). The time-invariant weight and bias terms have the following dimensions, $\mathbf{W}_r \in \mathbb{R}^{N_h \times N_x}$, $\mathbf{U}_r \in \mathbb{R}^{N_h \times N_h}$, and $\mathbf{b}_r \in \mathbb{R}^{N_h \times 1}$. The update gate determines how the current state $\mathbf{h}_t$ is selected in the range spanned by the previous hidden state $\mathbf{h}_{t-1}$ and the candidate hidden state $\tilde{\mathbf{h}}_t$. Mathematically, the update gate can be expressed as

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z), \tag{2}$$

where $\mathbf{W}_z \in \mathbb{R}^{N_h \times N_x}$, $\mathbf{U}_z \in \mathbb{R}^{N_h \times N_h}$, and $\mathbf{b}_z \in \mathbb{R}^{N_h \times 1}$. The candidate hidden state $\tilde{\mathbf{h}}_t$ for updating the new state is determined by integrating the reset gate $\mathbf{r}_t$ with past information $\mathbf{h}_{t-1}$, as well as the current input $\mathbf{x}_t$:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}(\boldsymbol{r}_t \circ \boldsymbol{h}_{t-1}) + \boldsymbol{b}), \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{N_h \times N_x}$, $\mathbf{U} \in \mathbb{R}^{N_h \times N_h}$, and $\mathbf{b} \in \mathbb{R}^{N_h \times 1}$; the symbol $\circ$ is the element-wise product operator; *tanh* is the element-wise hyperbolic tangent activation function that scales the vector $\tilde{\mathbf{h}}_t$ to be between (-1, 1). The standard form of RNN can be represented as $\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b})$, which resembles Eq. (3), except for the reset gate $\mathbf{r}_t$. Hence, the candidate hidden state itself can be viewed as a variant of RNN with a gate mechanism to selectively pass the information. The vector $\tilde{\mathbf{h}}_t$ partly contributes to updating the new hidden state $\mathbf{h}_t$, which is defined as

$$\boldsymbol{h}_t = \boldsymbol{z}_t \circ \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \circ \tilde{\boldsymbol{h}}_t. \tag{4}$$

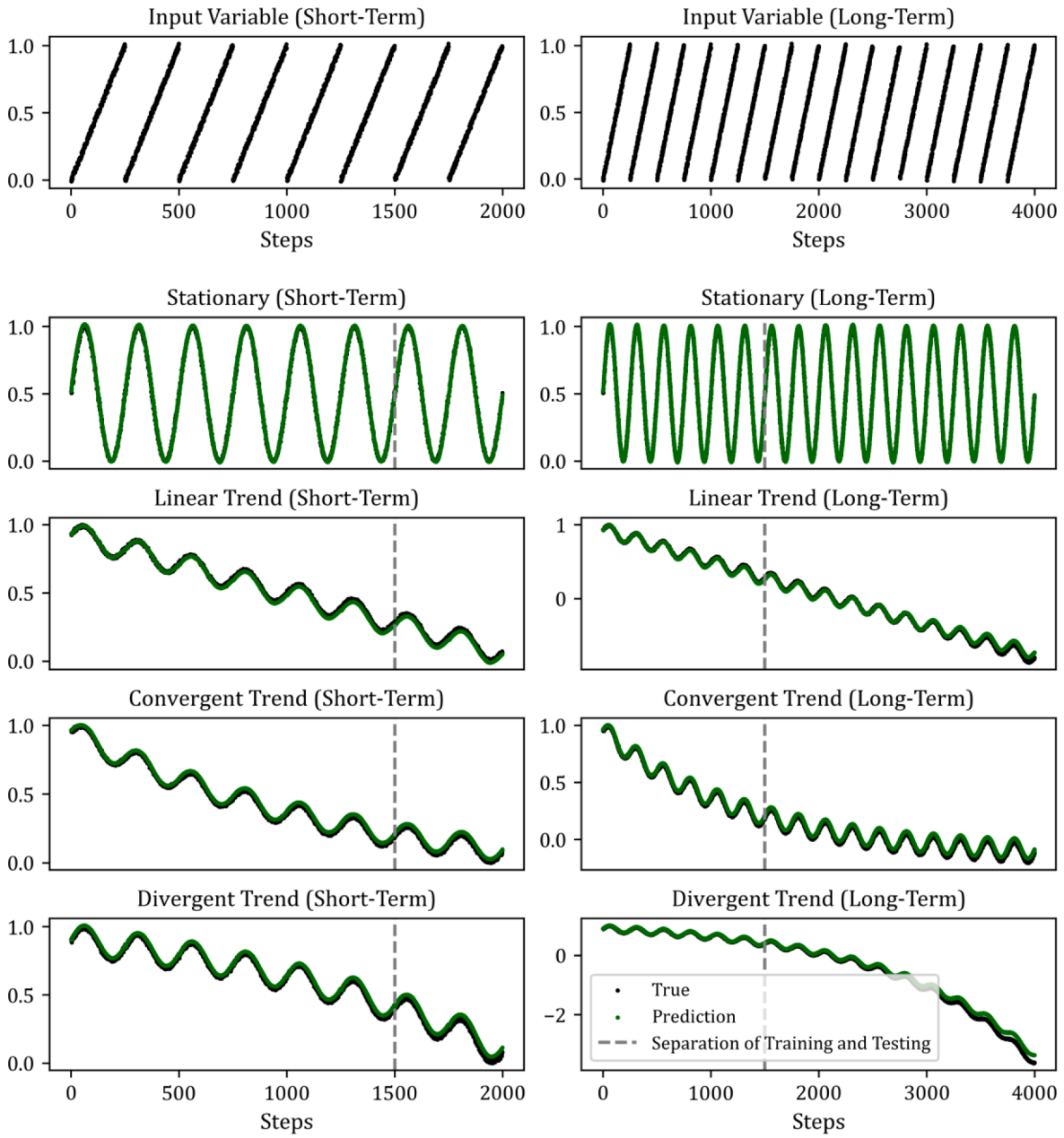


Fig. 4. Visualization of the toy example with different trends. The first row represents the input sequences over 2000 and 4000 steps, respectively. The output sequences (from 2nd to 4th rows) cover the first 2000 steps on the left column, and 4000 steps on the right column. The prediction (green dots) is generated by *phy*-GRU model.

The update gate $\mathbf{z}_t$ contains the weights used to linearly combine $\mathbf{h}_{t-1}$ and $\widetilde{\mathbf{h}}_t$. When $\mathbf{z}_t$ approaches 1, the new hidden state $\mathbf{h}_t$ is close to the previous state $\mathbf{h}_{t-1}$, and vice versa. Therefore, one can view the GRU as an RNN with two gates to 1) selectively pass the information (i.e., reset gate) and 2) introduce an additional update (weighted average) for the current state (i.e., update gate).

2.2.2. Skip connection and differential equation

Residual Network (ResNet) was originally proposed for image recognition in the computer science domain (He et al., 2016). The residual block of ResNet (Fig. 2**(a)**) has a skip connection to keep features in different layers in the same scale and to avoid gradient vanishing (Lu et al., 2018). The skip connection, also known as identity mapping, preserves the input features and transfers them to the next operation. Each residual block is represented as $y_{l+1} = \mathscr{F}_{\theta_l}(y_l) + y_l$, where y_l is the input feature to the l-th layer and $\mathscr{F}_{\theta_l}(\cdot)$ is the neural network operation with the weights θ_l. The input feature y_l is sent to the $(l+1)$-th layer through the skip connection. Each residual block can be viewed as a forward Euler discretization step for the ODEs (E, 2017). Given an initial state y_0, the initial value problem (IVP) over the prediction horizon $t \in (0,\ T]$ can be defined as

$$\begin{aligned} \frac{dy}{dt} &= f(t,\ y),\ for\ t \in (0,\ T] \\ y(0) &= y_0. \end{aligned} \tag{5}$$

The forward Euler discretization of the differential equation with a step size Δt_n is then derived as

$$y_n = y_{n-1} + \Delta t_n \cdot f(t_n,\ y_n), \tag{6}$$

where t_n and y_n denote the time and solution of the n-th step, respec-

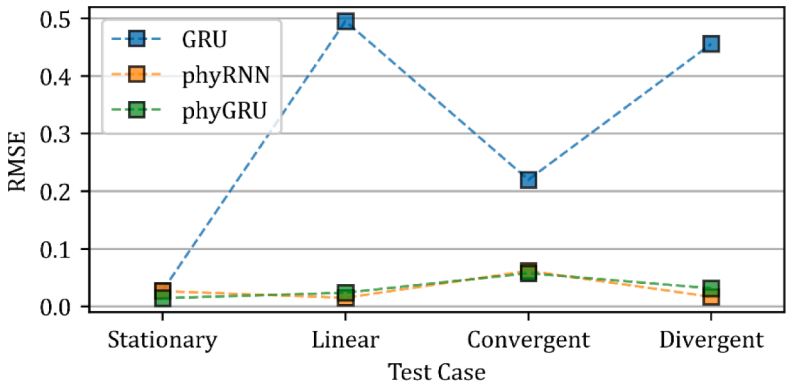


Fig. 5. The testing RMSEs for three different data-driven predictive models for long-term predictions in the toy examples. The RMSE values are calculated over the last 1500 time steps, during which the output and predictions are re-normalized (re-scaled).

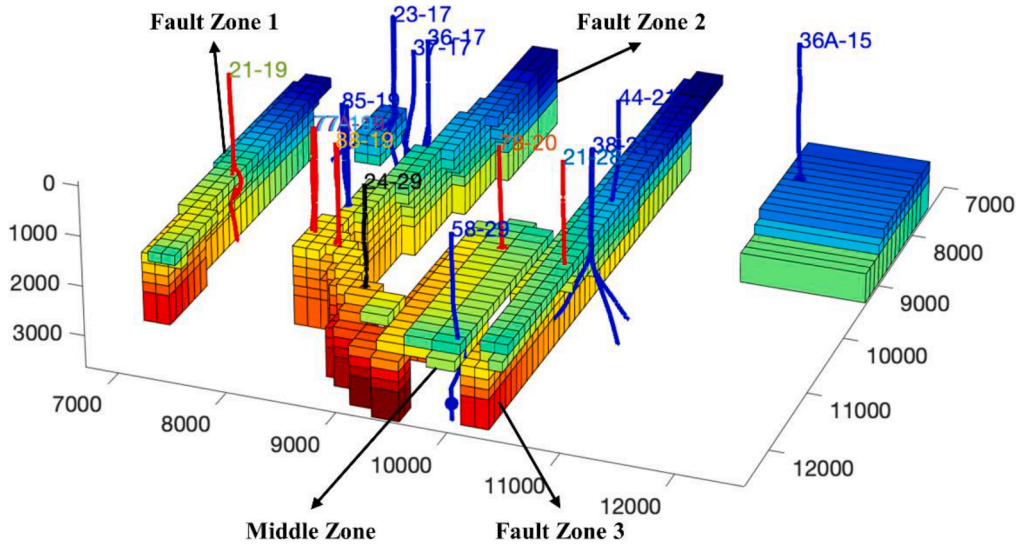


Fig. 6. Temperature distribution for the main fault zones in the physics-based simulation model of the field-scale geothermal reservoir. The main fault zones are named Fault Zone 1, Fault Zone 2, Middle Zone, and Fault Zone 3, from West to East, respectively.

tively. Here, we use the subscript n to represent each time step in the physical problem statement (to avoid confusion with the time notation t used in statistical problems, e.g., GRU model description). Hereafter, we use the subscript n for the physical problem and t for the statistical problem description. The term $\Delta t_n \cdot f(t_n,\ y_n)$ can be approximated by the neural network $\mathscr{F}_{\theta_l}(\cdot)$, with the layer index l representing the time step n. The structure of $\mathscr{F}_{\theta_l}(\cdot)$ is not specified and can be any fit-for-purpose NN, such as fully connected NN and convolutional filters. A recursive way of using the ResNet can be applied to solve y_n for any arbitrary step. The weights of each residual block in the ResNet can either be shared or remain independent of other blocks (Qin et al., 2019).

In this work, the problem is slightly different from the above initial value problem due to the additional input variables to the dynamical system. In our previous work (Qin et al., 2022), we simplify the PDE-based dynamical system for temperature prediction to take the form

$$\frac{\partial y}{\partial t} = f(t,\ x,\ y), \tag{7}$$

where x is the well control (i.e., mass flow rate). The function $f(\cdot)$ consists of different sources of temperature decline, such as conduction, sink/source term, and convection. For the step n, the explicitly discretized form of Eq. (7) is expressed as

$$y_n = y_{n-1} + \Delta t_n \cdot f(t_n,\ x_n,\ y_{n-1}). \tag{8}$$

To approximate the numerical discretization, we modify the architecture of the residual block by adding additional input variables. The resulting architecture of the predictive model as a variant of RNN is presented in Fig. 2**(b)**. The term $\Delta t_n\, f(t_n,\ x_n,\ y_{n-1})$ is approximated by the function $\mathscr{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{h}_{t-1})$ with the trainable weights $\boldsymbol{\theta}$, leading to the mathematical representation

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} + \mathscr{F}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}), \tag{9}$$

where we replace the notation y_n with $\mathbf{h}_t$ for consistency with the representation of RNN in the previous section. The weight-sharing property of RNN implies that the weights $\boldsymbol{\theta}$ for any time step n are fixed. Hence, an implicit assumption is that the step size Δt_n in Eq. (8) is also fixed.

2.2.3. *Physics-guided RNNs*

We now introduce two physics-guided deep-learning models with different architectures for the function $\mathscr{F}_{\theta}(x_t, h_{t-1})$. The overall architecture takes the form of RNN-type deep learning models and merges the skip connection with RNNs. The first physics-guided deep learning model (Fig. 3a) is designed based on the architecture of GRU and is named physics-guided GRU (*phy*-GRU). By removing the update gate $\mathbf{z}_t$ in Eq. (4), we can convert the GRU to the RNN variant with a skip connection. The difference between *phy*-GRU and standard GRU is in the

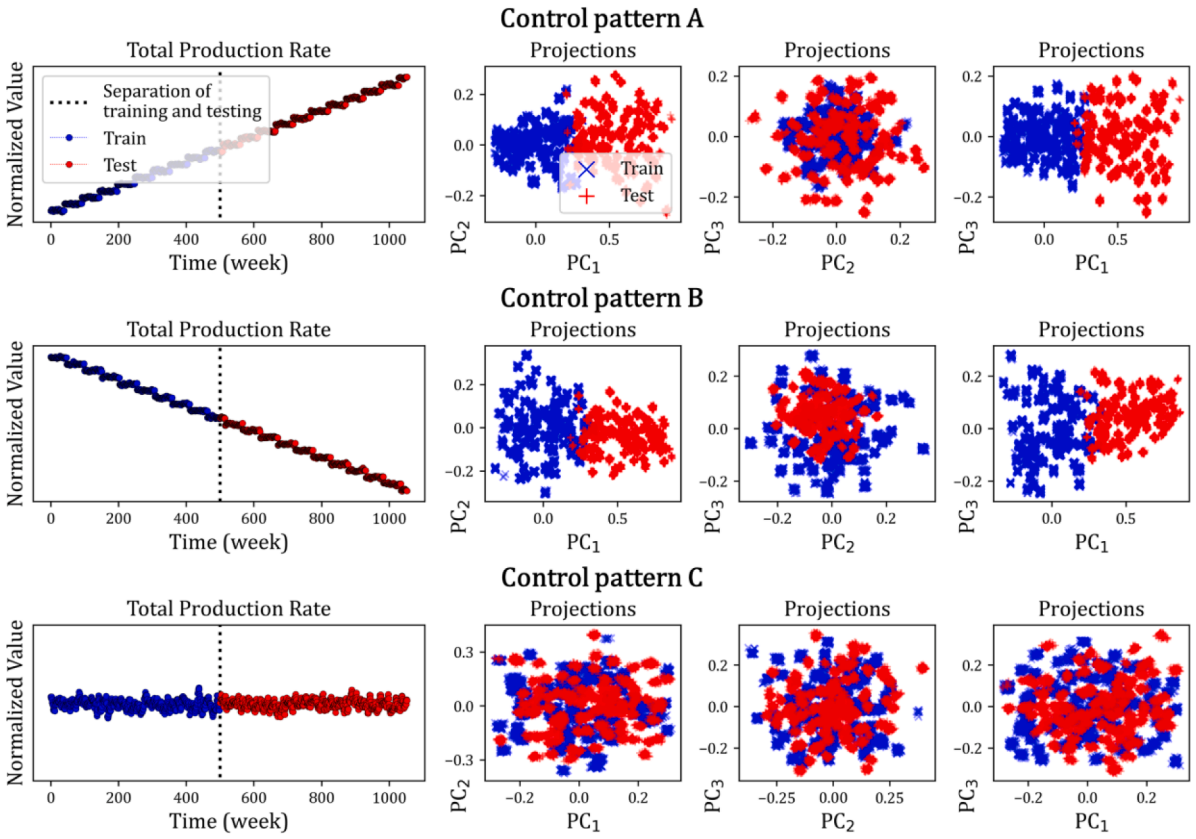


Fig. 7. Input control patterns and their corresponding projection on their three principal components (PC). The notations PC_i denotes the i^{th} PC. The first column shows the total mass rate for all production wells. The projections onto the first three PCs are derived from the input controls of all ten instances for six production wells.

last step by converting Eq. (4) to

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} + \delta t \cdot \boldsymbol{z}_t \circ \widetilde{\boldsymbol{h}}_t, \tag{10}$$

where the general form of neural network $\mathscr{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{h}_{t-1})$ is represented as $\delta t \cdot \mathbf{z}_t \circ \widetilde{\mathbf{h}}_t$. The notation $\delta t \in \mathbb{R}^1$ is a scalar trainable parameter denoting the time interval. The mathematical form of $\mathbf{z}_t$ and $\widetilde{\mathbf{h}}_t$ remain the same as in GRU [i.e., Eqs. (2) and (3)]. After removing the multiplier $\mathbf{z}_t$ in front of $\mathbf{h}_{t-1}$, Eq. (10) serves as a residual block with external input vector $\mathbf{x}_t$. For conciseness, the multiplier of $\widetilde{\mathbf{h}}_t$ is changed from $1-\mathbf{z}_t$ (Eq. (4)) to $\mathbf{z}_t$ (Eq. (10)). The new $\mathbf{z}_t$ is not an update gate anymore as it is not used to calculate the weighted average of $\mathbf{h}_{t-1}$ and $\widetilde{\mathbf{h}}_t$.

The second physics-guided deep learning model (Fig. 3b) is designed to introduce the skip connection into RNN and is named physics-guided RNN (*phy*-RNN). In this section, we will explore the connection between *phy*-RNN and a simplified physical equation known as the lumped parameter model in the literature. The lumped parameter model consists of multiple tanks and a matrix of connections that denotes the communication among tanks. Each tank represents a certain area of the geothermal reservoir of which the average geological properties are assigned to the tanks (porosity, permeability, etc.). For generalized lumped parameter models, the relationship between the input control and output drawdown is expressed as (Sigurdardottir et al., 2015; Li et al., 2017)

$$\boldsymbol{K}\frac{\partial \boldsymbol{d}}{\partial t} = \boldsymbol{S}\,\boldsymbol{d} + d_\infty \cdot \boldsymbol{\sigma}_\infty + \boldsymbol{m}, \tag{11}$$

where $\mathbf{K} \in \mathbb{R}^{T\times T}$ and $\mathbf{S} \in \mathbb{R}^{T\times T}$ represent the storage capacity and conductance matrix for the tanks with dimension T, respectively; $\mathbf{d} \in \mathbb{R}^{T\times 1}$ and $\mathbf{m} \in \mathbb{R}^{T\times 1}$ denote the output drawdown and input control vectors, respectively; $d_\infty \in \mathbb{R}^1$ stands for the drawdown for infinite recharge source (treated as a fixed parameter); $\boldsymbol{\sigma}_\infty \in \mathbb{R}^{T\times 1}$ is the conductance vector between the tanks and an infinite recharge source. Eq. (11) can be discretized using an explicit method as follows:

$$\boldsymbol{d}_t = \boldsymbol{d}_{t-1} + \Delta t \cdot (\boldsymbol{W}_d \boldsymbol{d}_{t-1} + \boldsymbol{W}_m \boldsymbol{m}_t + \boldsymbol{b}), \tag{12}$$

where $\mathbf{W}_{\mathrm{d}} = \mathbf{K}^{-1}\mathbf{S}$, $\mathbf{b} = \mathbf{K}^{-1}\mathrm{d}_\infty\boldsymbol{\sigma}_\infty$, and $\mathbf{W}_m = \mathbf{K}^{-1}$. The diagonal matrix $\mathbf{K}$ is positive definite and invertible. The second term on the right-hand side (RHS) of Eq. (12) resembles the standard RNN equation

$$\boldsymbol{h}_t = \tanh(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b}), \tag{13}$$

where $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ represent terms $\mathbf{d}_{t-1}$ and $\mathbf{m}_t$, respectively. The first term $\mathbf{d}_{t-1}$ on the RHS of Eq. (12) acts as the identity mapping (or skip connection) in the ResNet. Mathematically, *phy*-RNN can be represented as

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} + \delta t \cdot \tanh(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b}), \tag{14}$$

Both *phy*-GRU and *phy*-RNN are designed to merge the concept of skip connections with regular RNNs to accommodate the physical constraints introduced by the finite difference scheme. However, the corresponding architectures of $\mathscr{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{h}_{t-1})$, which are used to approximate the physical mapping $\Delta t \cdot f(\cdot)$, are different. The *phy*-GRU model adopts the main structure of the standard GRU, whereas the *phy*-RNN follows the equations in lumped parameter modeling and resembles the standard RNN. The differences in the $\mathscr{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{h}_{t-1})$ architecture lead to different behaviors in the two models. For example, the output from *phy*-RNN is more sensitive to the input control $\mathbf{x}_t$ since it has a shallower and simpler NN, whereas the *phy*-GRU typically provides smoother outputs due to the multiplier $\mathbf{z}_t$. In the next section, we further investigate the performance of these models through numerical experiments

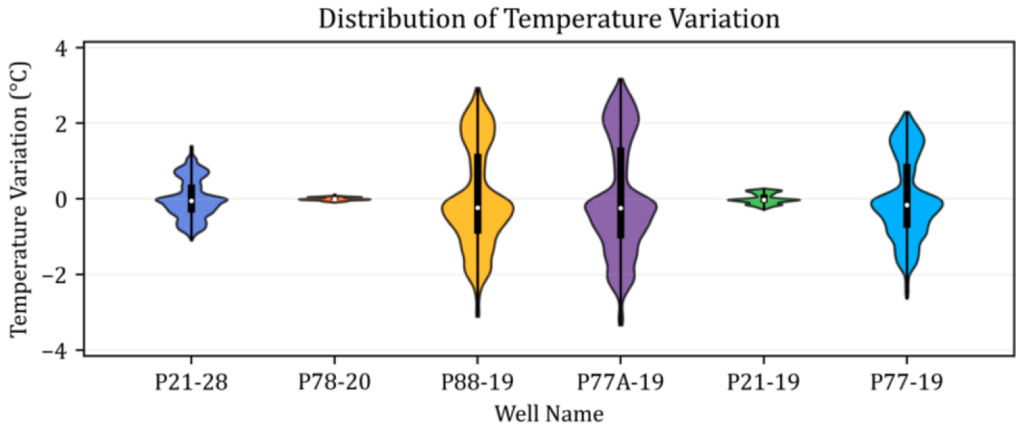


Fig. 8. Distribution of production temperature variation over 20 years of simulation under different control scenarios. The black bar represents the range between 25 and 75 percentiles of the distributions. The white point represents the mean of the distribution.

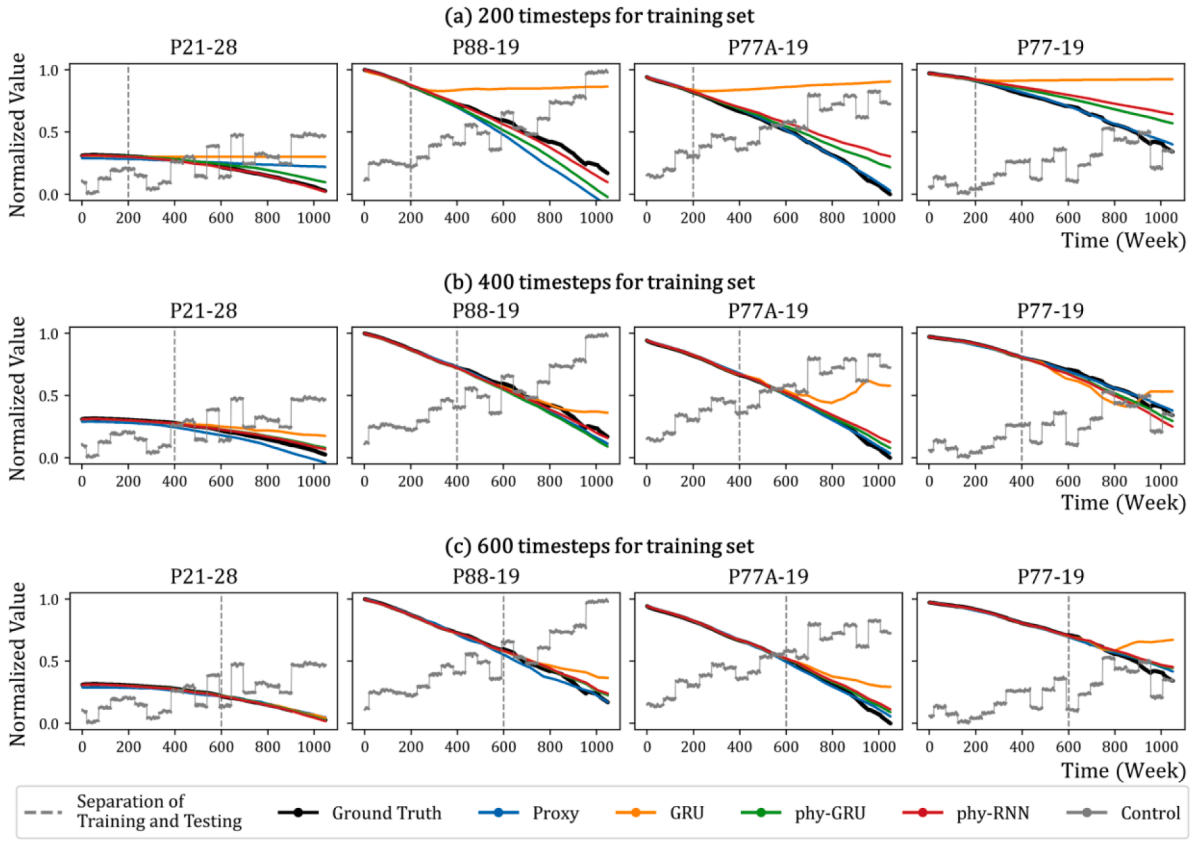


Fig. 9. Prediction results of production temperature for Control Pattern A using four different predictive models. Each model is trained with a varying number of time steps (samples), as follows: (a) Trained using 200 time steps; (b) Trained using 400 time steps; and (c) Trained using 600 time steps of samples. The ranges of output (production temperature) and input control (indicated by the gray line) are from 0 to 1. The data points after the vertical dashed lines represent the test sets.

and a field example.

3. Experiments

3.1. Test cases

We first investigate the ability of the models in capturing long-term trends using a toy example. A difficulty faced in data-driven modeling is extrapolation, especially for non-stationary data. An example is the production temperature decline under continuous injection of cold water, even when the input controls (injection and production rates) remain fixed. In this case, long-term prediction of temperature is bound to fall out of the range spanned by historical data. To investigate the performance of the proposed models in this extrapolation task, we generate synthetic datasets with different types of intrinsic trends in the output data (Fig. 4). The trends in the output sequence are independent of the input control variable. As shown in the first row of Fig. 4, the input control variables $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{N_T}\}$ are stationary and periodic, with their values remaining in the same range over 4000 time steps. The output data $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_{N_T}\}$ are generated by adding periodic (sine) functions $\sin(2\pi\mathbf{x})$ to different types of trends, named as "Stationary", "Linear trend", "Convergent trend", and "Divergent trend", shown in the

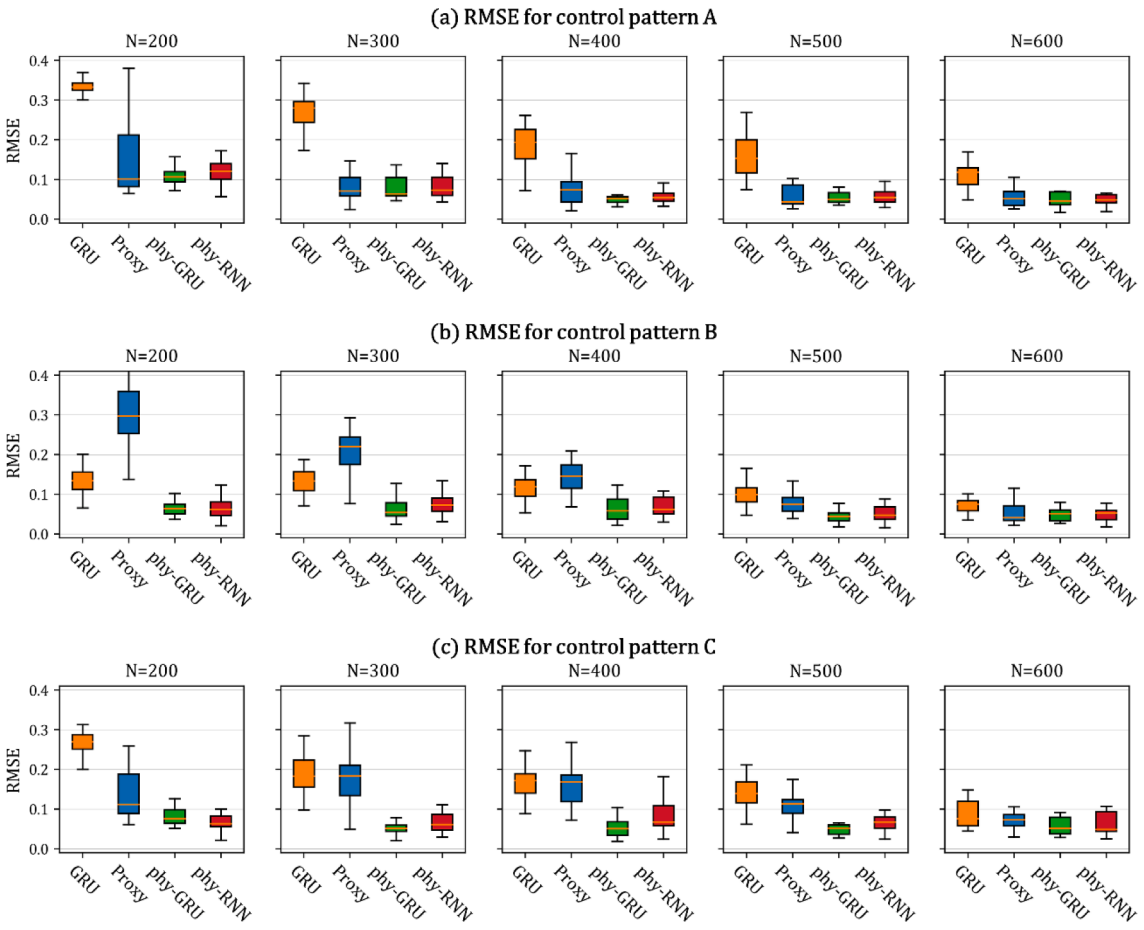


Fig. 10. The RMSE distributions of the test dataset of production temperature across ten instances in all four models. Five columns represent the distribution for the models trained using different number of time steps, ranging from 200 to 600 as indicated in the subtitles. In each subfigure, the box plot corresponds to the first quartile (25^{th}) and third quartile (75^{th}) of the RMSE distribution over ten instances.

second to the last row of Fig. 4. In this example, $N_x = 1$ and $N_y = 1$. Therefore, the input $\mathbf{X}$ and output $\mathbf{Y}$ are 1-by-4000 vectors. The prediction is an interpolation task for the "Stationary" case, but an extrapolation task for the cases of "Linear", "Convergent", and "Divergent" trends.

We assume that only the first 2000 steps of input and output sequences are available at the beginning. Therefore, the min-max normalization is performed only based on the first 2000 steps. The last 2000 steps of data points are treated as future data that are unseen by the predictive models. The proposed model is first developed using the first 1500 data points and generates short-term predictions for the next 500 steps. To investigate the extrapolation property of the models, we make long-term predictions over the last 2000 steps using the trained model and compare the predictions with future data. The predictive model used here is a stacked RNN with two layers, consisting of the regular GRU layer and the physics-guided RNNs (*phy*-RNN or *phy*-GRU) (See Appendix A). The first GRU layer is designed to handle the nonlinear mapping introduced by the sine function, while the second physics-guided RNN layer is designed to handle the intrinsic trend. For each forward run on one training sample, the GRU layer is fed with the input sequence $\mathbf{x}_t$ and initial hidden state $\mathbf{z}_t \in \mathbb{R}^{N_z \times 1}$, where the initial hidden state is a zero vector by default. The output from the GRU layer is the matrix of latent variables $\mathbf{Z}_t \in \mathbb{R}^{N_z \times N_t}$, where $N_t = 102$ and $N_z = 3$. The *phy*-RNNs take $\mathbf{Z}_t$ as input sequence and $\mathbf{y}_t \in \mathbb{R}^{N_y \times 1}$ as its initial hidden state and predict $\widehat{\mathbf{Y}}_t \in \mathbb{R}^{N_y \times N_t}$ over N_t steps. The loss function of the predictive model for a batch of N_s samples is defined as

$$\mathscr{L}(\boldsymbol{\theta}) = \sum_{t}^{N_s} \| \widehat{\boldsymbol{Y}}_t - \boldsymbol{Y}_t \|_2^2, \tag{15}$$

where $\mathbf{Y}_t \in \mathbb{R}^{N_y \times N_T}$ is the ground truth and $\boldsymbol{\theta}$ contains the trainable parameters of the predictive model. The loss function is optimized by using the Adam optimizer (Kingma and Ba, 2014). Further information about the model complexity and hyperparameters is provided in Appendix A.

Fig. 4 shows the results generated by the proposed *phy*-GRU on four different cases. The proposed predictive model accurately predicts local nonlinear variants caused by the changes in the input control. More importantly, *phy*-GRU can capture different trends over the next 2500 steps by only learning from the first 1500 steps. This example suggests that the proposed approach can better handle extrapolation tasks by providing long-term predictions of non-stationarity trends. We further compare the performance of *phy*-RNN and *phy*-GRU with purely data-driven model. Specifically, a stacked RNN model with two GRU layers is applied to the toy example. The only difference between the three different models is the second layer. Fig. 5 compares the root-mean-square errors (RMSE) of three different models over four cases. Both *phy*-RNN and *phy*-GRU can predict the long-term trends for different cases and capture the nonlinear local perturbations. On the other hand, the GRU model works well in the stationary case but fails in all the other three non-stationary cases, especially for the "Linear" and "Divergent" cases.

Overall, the experiment suggests that the proposed *phy*-RNNs are capable of handling time-series data with different trends and making long-term predictions in the presence of intrinsic non-stationarity. The

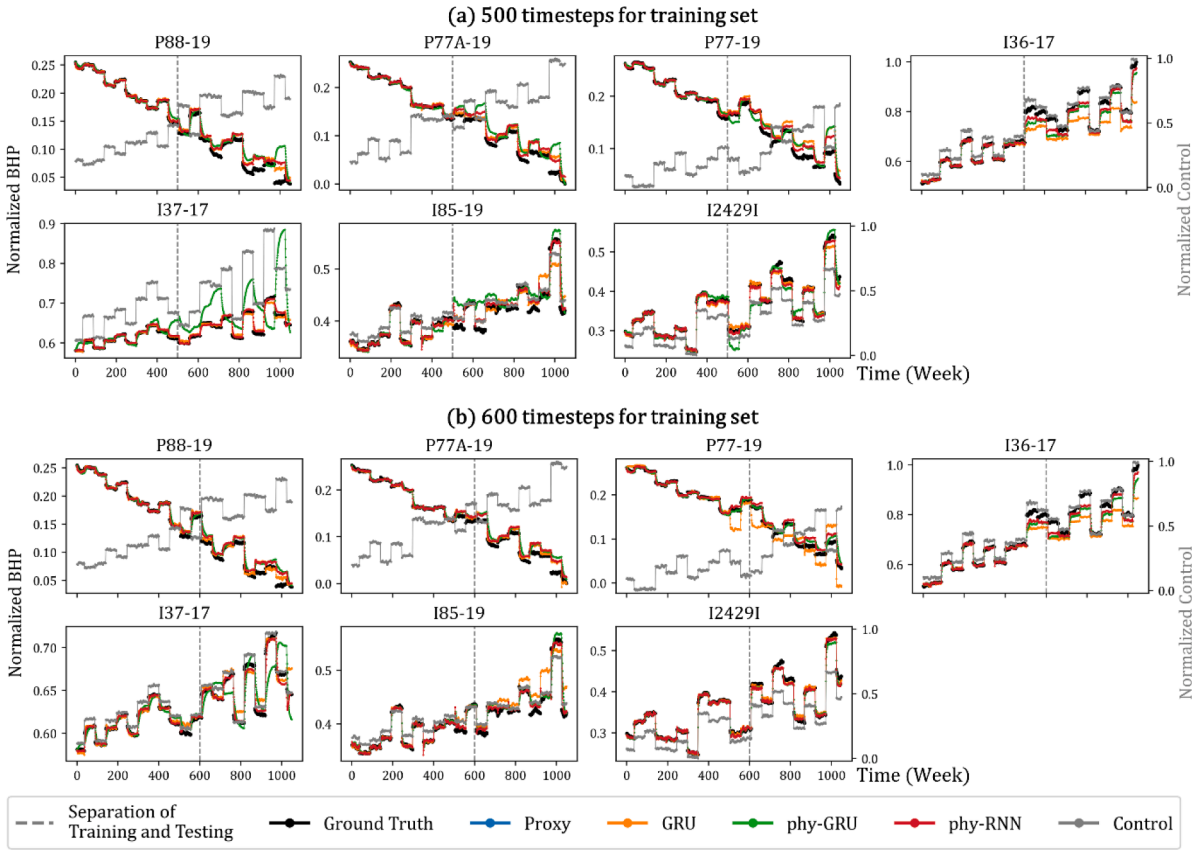


Fig. 11. BHP prediction results for Control Pattern A using four different predictive models that are trained using 500 and 600 time steps of data. The range of input control (indicated by the gray line) is shown on the right y-axis and spans from 0 to 1.

phy-RNNs are better able to handle non-stationarity that can result in the outputs falling outside the training data distribution.

3.2. Simulated field cases

We apply the proposed physics-guided RNNs to a simulated dataset obtained from a simulation model of a field-scale geothermal reservoir. A simulation model of a low-enthalpy single-phase geothermal reservoir is applied to generate the simulated dataset (Fig. 6). The geothermal reservoir features high-permeability zones, named "Fault Zones", of which the temperature distribution is shown in Fig. 6. There are six production wells shown as red columns, which are named 21-19, 77-19, 77A-19, 88-19, 78-20, and 21-28 from West to East. Eight injection wells named 23-17, 36-17, 37-17, 85-19, 24-29, 38-21, 44-21, and 36A-15 exist and are shown in the blue column. Well 24-29 is converted from an observation well to an injection well. The simulation model spans over around twenty years with a fixed time interval of seven days, amounting to 1055 time steps in total, i.e., $N_T = 1055$. The input control variables to the simulation model are the mass flow rate of both production and injection wells. The output observations collected from the simulation model are the brine temperature of six production wells and the BHP of both production and injection wells.

We investigate the second type of extrapolation, where the input variables in the training dataset $\mathbf{X}_{train} = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$ and testing dataset $\mathbf{X}_{test} = \{\mathbf{x}_{M+1}, \cdots, \mathbf{x}_N\}$ are sampled from two distributions with different ranges. The investigation of the second extrapolation problem is necessary since the control optimization implementation with data-driven models typically leads to an extrapolation situation where the control vectors are high-dimensional and typically fall beyond the range of training samples (Yu and Jafarpour, 2022). We define the difference between the sampling distribution in the training and testing datasets as the "inconsistency". As shown in Fig. 7, we generate three simulated datasets with different levels of inconsistency for the input control, named Control Patterns A, B, and C. The total production rates of the first two patterns have ramp-up and ramp-down trends, respectively. In contrast, the third control pattern has a consistent range. The sampling of the controls must meet the following constraints: 1) the total production mass rate equals the total injection mass rate, 2) the mass rate control of each well is bounded by upper and lower limits, and 3) the total production/injection rate follows the control pattern shown in Fig. 7. For each control pattern, we generate ten instances by randomly allocating the total rate to each well while honoring the above constraints.

Next, we investigate the performance of the proposed *phy*-RNN and *phy*-GRU models for long-term prediction by applying them to simulated datasets with different control patterns. Specifically, we apply the proposed models to predict production temperature and BHP. These two types of outputs have distinct features in terms of extrapolation. Under continuous injection, the production temperature exhibits an intrinsic trend, i.e., temperature decline, which is the first type of extrapolation studied in this work. The second type of extrapolation, caused by the inconsistency in the control range, exists in both temperature and BHP data. On the other hand, the BHP can be maintained within a consistent range when the injected cold water provides sufficient pressure support. Therefore, BHP prediction typically involves only the second type of extrapolation, while temperature prediction includes both types. In this experiment, the proposed predictive models are single-layer RNNs utilizing either *phy*-RNN or *phy*-GRU as the recurrent unit (See Appendix A). The decision to employ single-layer RNNs, as opposed to two-layer RNNs in the previous example is based on the following reasons. Compared to the test case in Section 3.1, the simulated dataset in this experiment exhibits less nonlinearity, thereby facilitating more

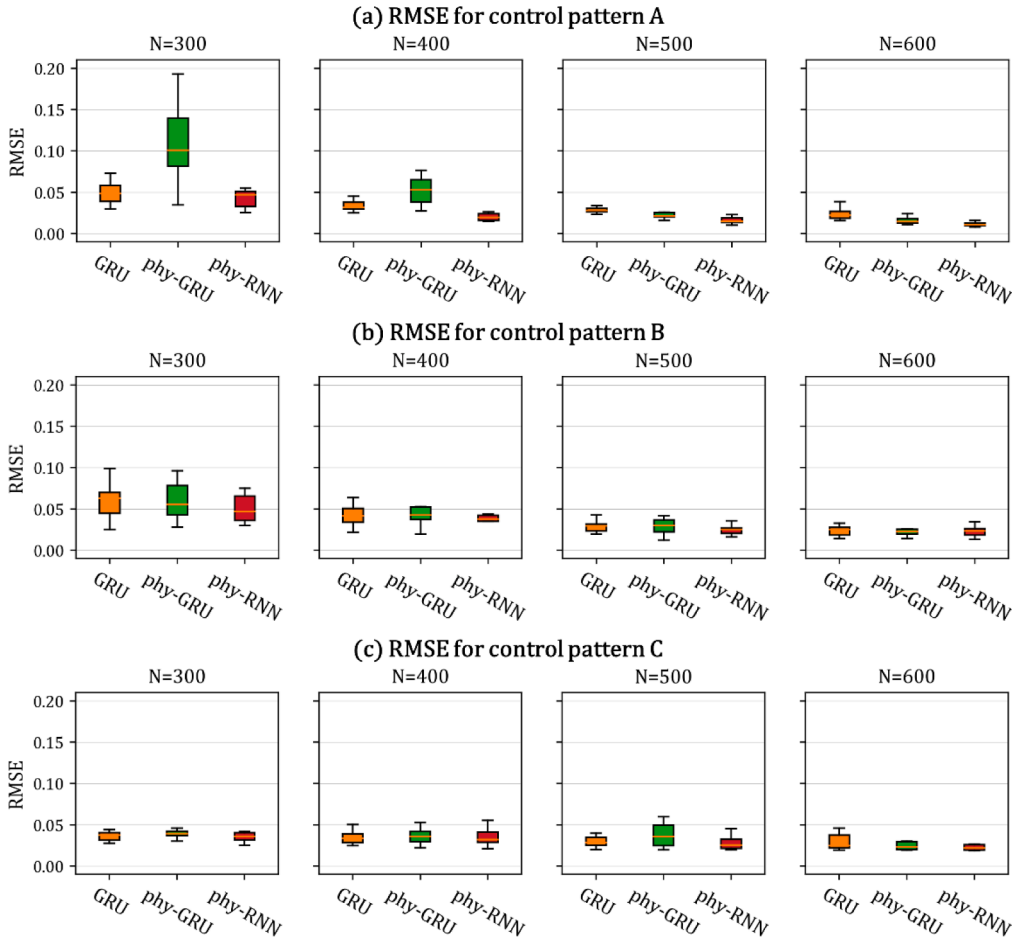


Fig. 12. RMSE distributions for the BHP test dataset corresponding to three models and across ten instances. Four columns represent the number of time steps used for training the models (300, 400, 500, and 600, respectively).

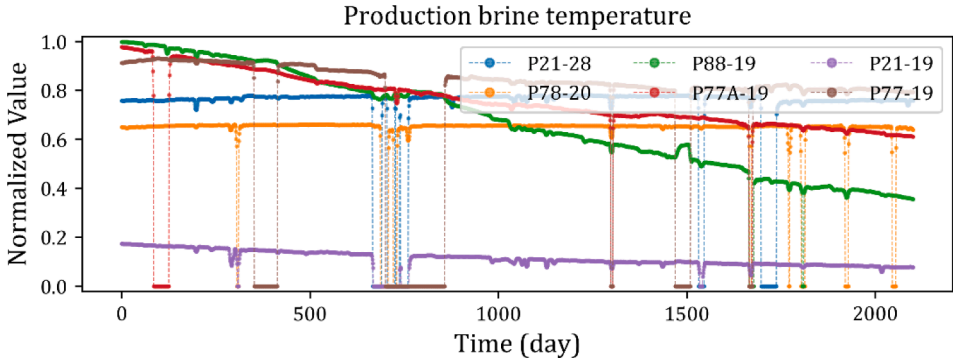


Fig. 13. Normalized temperature of the produced brine in the field. The zeros in the data indicate missing values.

accurate predictions by single-layer *phy*-RNNs. Moreover, the single-layer architecture has fewer trainable weights and more modest training data requirements.

3.2.1. *Production temperature prediction*

Fig. 8 shows the temperature variation for 1053 time steps over three different control patterns and ten instances. The variation for each time step is defined as the difference between the temperature and the mean value of temperature over different patterns and instances. It is shown that, under different control patterns, the temperature profiles of Wells 21-28, 88-19, 77A-19, and 77-19 show clear variation due to their proximity to injection wells. The brine temperatures of Wells 78-20 and 21-19 are stable over the whole simulation time (around 20 years). We exclude Wells 78-20 and 21-19 in our temperature prediction task. The output $\mathbf{Y} \in \mathbb{R}^{4 \times 1053}$ is the production temperature of Wells 21-28, 88-19, 77A-19, and 77-19 over 1053 time steps. The input $\mathbf{X} \in \mathbb{R}^{11 \times 1053}$ is the mass flow rate of four production wells and seven injection wells located in Fault Zones 2 and 3. In addition, we set different values for the length of the training set N_{T_0} to investigate its effect on the prediction performance. That is, we use the first N_{T_0} (= 200, 300, 400, 500, and 600) time steps of the simulated dataset $\{\mathbf{X}, \mathbf{Y}\}$ to generate the training set. Other than the two physics-guided deep learning models, we include the regular GRU model as well as the physics-based proxy model (denoted as Proxy in this paper) proposed by Qin et al. (2022) for

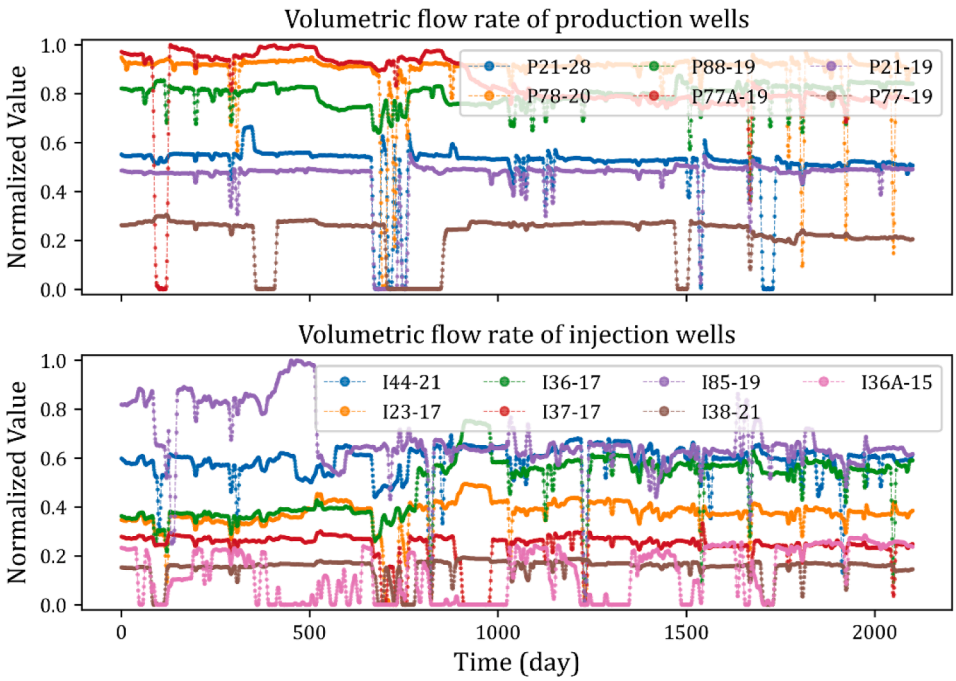


Fig. 14. Normalized (a) production and (b) injection volumetric flow rate in the geothermal field. The zeros in the data indicate shut-in periods or missing values.

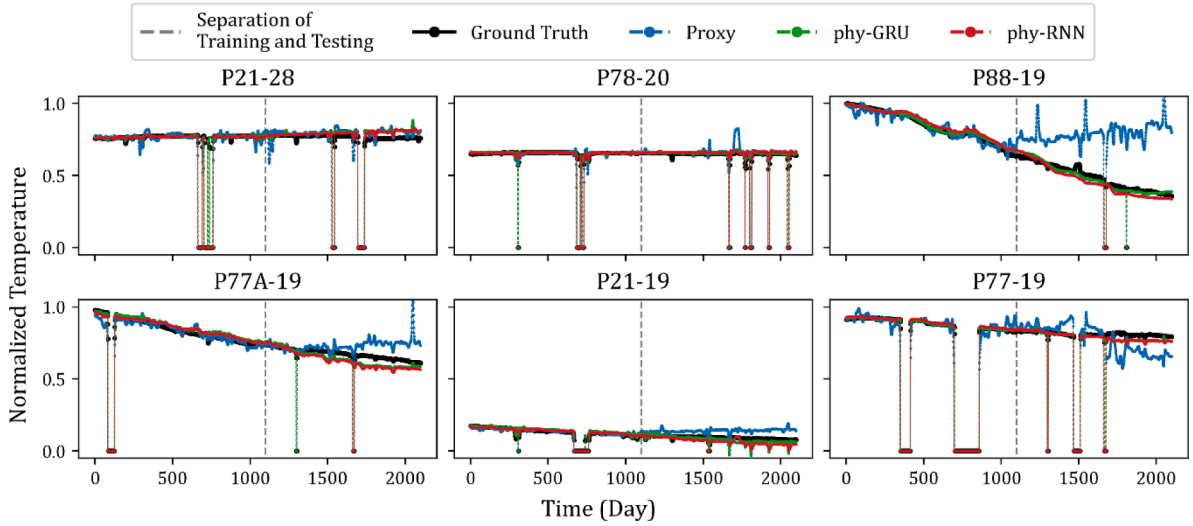


Fig. 15. Prediction results for brine temperature in six production wells using three different models: the proxy model (Proxy), *phy*-GRU, and *phy*-RNN. All three models are trained using the first 1100 time steps, with testing performed after the 1100th time step.

comparison.

Fig. 9 shows the prediction results for the case with "Control Pattern A" using four different predictive models that are trained on training samples with different sizes. The results for the cases with Control Patterns B and C are shown in Appendix B. For $N_{T_0} = 200$ (Fig. 9**(a)**), the predictions from the four models do not consistently follow the correct trends over all the production wells. Specifically, the predictions from *phy*-RNN and Proxy models appear as straight lines for Well 21-28 (Proxy) as well as for Wells 77-19 and 77A-19 (*phy*-RNNs). Instead, the ground truth of temperature shows a "Divergent" curve due to the increasing injection rate in this case. This indicates that the models need additional training samples to learn the input-output relationship properly, rather than being overfitted to the historical trend with a simple linear trend. In contrast, the predictions from the regular GRU consistently reach a plateau over the future steps and exhibit significant discrepancies. The average RMSE over ten different instances of models for the four wells is around 0.33 (as shown in Fig. 10(a)), significantly higher than that of the *phy*-RNNs and Proxy models (i.e., 0.1). However, it is worth noting that the RMSE of the Proxy and $N_{T_0} = 200$ in Fig. 10(a) has a significant variance and ranges from 0.07 to 0.38. This is partly attributed to the fact that temperature prediction by the Proxy relies on the BHP predictions from a different deep learning model (the *phy*-RNN model). This design results in more trainable parameters for the Proxy, which is easily overfitted. Furthermore, any errors arising from the BHP prediction will propagate to the temperature prediction and potentially accumulate. However, the design of this joint prediction is necessary for the Proxy model since it is rule-based and requires the pressure as an input for temperature prediction.

When N_{T_0} is increased to 400 (Fig. 9**(b)**), the long-term predictions from *phy*-RNNs and Proxy can capture the temperature trend over more

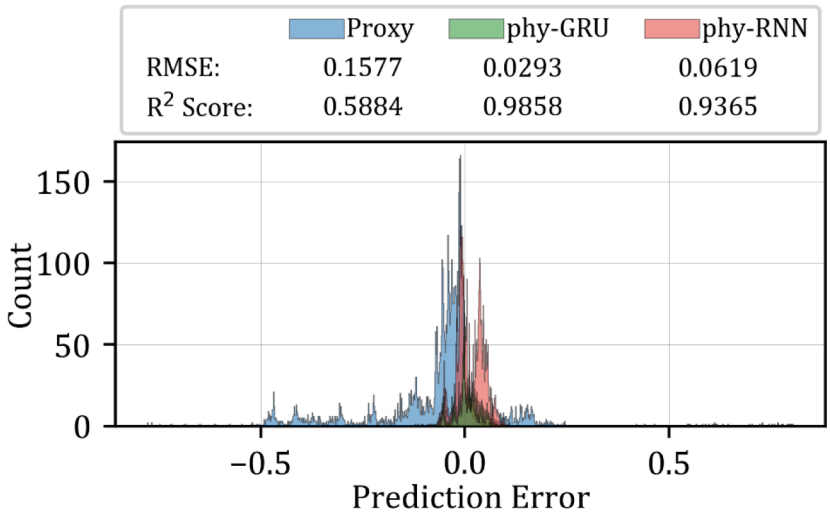


Fig. 16. The distribution of prediction errors ($\mathbf{y} - \hat{\mathbf{y}}$) for three predictive models in the field example. The legend displays the RMSEs and R^2 scores between $\mathbf{y}$ (measured) and $\hat{\mathbf{y}}$ (predicted) for the three models. The y-axis represents the frequency of occurrences of error value, with a total of 6000 instances across 1000 time steps and 6 wells.

than 600 time steps, amounting to around 12 years. In contrast, the predictions from the GRU model continue to deviate from the ground truth as the prediction horizon extends. As shown in Fig. 10, the average RMSE values for *phy*-RNNs converge to around 0.06 when $N_{T_0} = 400$ for three control patterns. The Proxy model provides comparable results only when N_{T_0} is increased to 400 or above and provides unstable predictions with fewer training samples, as reflected by the high RMSE variance.

Additionally, as shown in Fig. 10(b), the regular GRU model achieves lower prediction errors compared to the other two control patterns. This finding highlights the behavior of the regular GRU model that performs better in capturing the "Convergent" trend compared to the "Divergent" and "Linear" trends, as reflected by the results in Section 3.1. In the simulated example, the temperature decline in Control Pattern B exhibits a “Convergent” trend as the injection rate of cold water progressively decreases over time. Therefore, the prediction from the regular GRU for Control Pattern B exhibits lower RMSE than that for the other two control patterns.

In summary, the proposed *phy*-RNNs outperform the other two models over all the cases with different patterns and sizes of training set N_{T_0}. The GRU model typically fails to extrapolate the temperature decline and works slightly better with Control Pattern B, where the temperature data shows a convergent trend. The Proxy model shows comparable performance only when there are sufficient training samples (**i.e.**, N_{T_0}=400 or 500).

3.2.2. *BHP prediction*

In this example, we apply the two *phy*-RNNs to predict the BHP data from the same dataset. In the case of temperature prediction, we only focused on the temperature of four production wells. In the case of BHP prediction, however, both injection and production wells are of interest. To reduce the model complexity and the size of training samples, we only focus on the wells from Fault Zone 2, which has the most complex well connectivity. The input $\mathbf{X} \in \mathbb{R}^{7 \times 1053}$ contains the controls of three production wells (i.e., 88-19, 77A-19, and 77-19) and four injection wells (36-17, 37-17, 85-19, and 24-29). The output $\mathbf{Y} \in \mathbb{R}^{7 \times 1053}$ is the BHP from the same wells. In addition to the two *phy*-RNNs, we apply the regular GRU model to this example for comparison. In this example, we start with a larger training window for training (N_{T_0} = 300, 400, 500, and 600) since the complexities of the three models are larger due to the higher dimension in the output feature. Fig. 11 shows the prediction results from the case “Control Pattern A” for the three different models. The normalized control for each well increases from zero to one over time. For the cases with N_{T_0}=500 and 600, *phy*-RNN provides accurate predictions even though the control range of the testing set becomes inconsistent. The predictions from *phy*-GRU are accurate for most wells but lose fidelity for Well 37-17. Fig. 12 presents a summary of the RMSE losses for three predictive models across three different control patterns. Among the three models, the *phy*-RNN model exhibits lower RMSE mean and variance, outperforming the other two models across most cases. In contrast, *phy*-GRU and regular GRU provide comparable results only when N_{T_0} is increased to 500 or 600 for three control patterns. The *phy*-GRU shares similar architecture with *phy*-RNN but contains more components (e.g., $\mathbf{z}_t$ and $\mathbf{r}_t$) which might involve more samples to train. Additionally, the *phy*-GRU tends to generate smoother predictions due to these additional components. This default behavior of the *phy*-GRU model may conflict with the non-smooth nature of BHP. On the other hand, the *phy*-RNN model is designed based on the water level calculation from the lumped parameter model. This design choice enables the *phy*-RNN model to benefit from a concise input-output relationship, allowing it to capture the essential dynamics with fewer training samples.

Among the cases with $N_{T_0} = 200$ over different control patterns, it is observed in Fig. 12 that the three predictive models demonstrate significantly lower variances in RMSE for Control Pattern C compared to Control Patterns A and B. In Control Pattern C, the BHP displays a stationarity behavior over time, as depicted in Fig. B-4, mainly because the control ranges used for both the training and testing sets remain consistent with each other. As a result, the predictive models can capture the underlying patterns effectively, leading to lower RMSE variances (Fig. 12(c)). Conversely, Control Patterns A and B exhibit non-stationary BHP behavior. This non-stationarity is attributed to the out-of-distribution controls in the testing set, relative to the control range during training. Consequently, the predictive models trained using 300 time steps of samples struggle to accurately capture the trends caused by control patterns.

3.3. *Field case*

We apply the *phy*-RNNs to real field data from a single-phase geothermal reservoir to predict the brine temperature at the production wells. The studied reservoir field is the same reservoir simulated in Section 3.2. The raw data are sampled hourly and contain a significant amount of redundant information, for example, noises and missing steps. The raw data is first downsampled to a lower frequency of daily

sampling, which helps the predictive model learn long-term trends. We also smooth both the input and output sequences to remove the noises and redundant fluctuations caused by missing data or short shut-in periods. Other than the missing data and noises, there are occasional gaps within the data caused by longer shut-in periods. We use the labeling scheme proposed by (Jiang et al., 2022) to avoid the prediction of those gaps. The input sequence $\mathbf{X} \in \mathbb{R}^{13 \times 2100}$ is the volumetric flow rate of seven injection wells and six production wells (Fig. 13). The output sequence $\mathbf{Y} \in \mathbb{R}^{6 \times 2100}$ is the brine temperature of six production wells (Fig. 14). The first 1100 time steps are used to generate training samples, and the next 1000 time steps are for testing. Other than two physics-guided RNNs, we still apply the proxy model in this example for comparison.

Fig. 15 shows the prediction results of the field brine temperature using the three different models. One major difficulty in predicting the field data is caused by measurement noises and irregular shut-ins. The predictions from the two *phy*-RNN models accurately capture the long-term trends and disregard the noise in the controls. This highlights the robustness of *phy*-RNNs to poor data quality. In contrast, the proxy model is more disturbed by noises and fails to predict the trends. Furthermore, in Fig. 16, the prediction error of *phy*-GRU is lower than that of the *phy*-RNN model, despite both models achieving R^2 scores close to 1. This observation suggests that the *phy*-GRU model is more suitable for temperature prediction than the *phy*-RNN model. The smoother predictions generated by the *phy*-GRU's default behavior may contribute to its lower prediction error compared to the *phy*-RNN model.

4. Discussion

4.1. Two types of extrapolations

We work on an extrapolation task and divide it into two types to better define the problem. The first type happens when the time-series output exhibits non-stationarity, such as the temperature decline. The challenge in this case is how to learn the intrinsic dynamics from training samples and make reliable long-term predictions. The second type of extrapolation happens when the input vector is out of the range spanned by the input in the training set, a situation that can occur when the input variable is high-dimensional (Balestriero et al., 2021). The mapping from a high-dimensional input space to the output space requires an extremely large dataset to cover the input space, which is impractical for engineering problems. However, a high-dimensional input space commonly exists in practical problems, such as field optimization and inverse modeling. In the field optimization problem, the input control might contain more than one hundred elements for multiple wells and multiple time steps. Furthermore, the control optimization using data-driven models typically leads to control solutions that fall beyond the training samples. Therefore, the control optimization needs a reliable predictive model that can address this type of extrapolation. In inverse modeling, the dimension of the geological variable (e.g., permeability) could be more than hundreds of thousands due to the high-dimensional grid system of a simulation model. The purely data-driven model typically fails to extrapolate beyond the training set, despite its powerful learning capability and efficiency. On the other hand, a reliable physics-based simulation model entails computationally expensive run time, which makes it prohibitive for complex iterative workflow (e.g., control optimization, inverse modeling, and uncertainty quantification).

We propose fit-for-purpose predictive models that integrate prior knowledge of physics into the architecture of deep learning models, which are named physics-guided RNNs (*phy*-RNNs). We investigate their prediction performance on test cases, field-scale simulated datasets, and field data. The results on the toy example show that the proposed deep learning models work well with the first type of extrapolation and learn the long-term trend from training samples, whereas the regular GRU can only handle the stationary case. In the simulated examples, we investigate the prediction performance of *phy*-RNNs on the production temperature and BHP, which are two distinct time-series data. Similar to the time series in test cases, production temperatures are non-stationary and refer to the first type of extrapolation. In contrast, the BHP data typically remains stationary when there is mass balance between production and injection. In addition, we set the control range of testing set to be out of the control range in the training set and therefore introduce the second type of extrapolation to both temperature and BHP predictions. Under different control patterns, the *phy*-RNNs provide reliable long-term predictions with sufficient training samples ($N_{T_0} = 300$ for temperature and $N_{T_0} = 500$ for BHP). We also applied two different models to the second experiment for comparison: the purely data-driven model (GRU) and the physics-based proxy model. The results show that the GRU model fares poorly for the extrapolation task, whereas the physics-based proxy model shows comparable performance only with enough training samples. Compared with *phy*-RNNs, the proxy model also requires the prediction of BHP data, which introduces more complexity to the model and therefore requires more samples for training.

4.2. Design of the architecture

In this work, we introduce a purely data-driven model (regular GRU) and the physics-based proxy model for comparison. The proxy model works as a rule-based method because the physics is embedded in the architecture. Furthermore, the physical relationships need to be simplified such that they can be represented as neural networks. Compared with the proxy model, the proposed *phy*-RNNs only keep the discretization scheme of the simplified governing equations and approximate the physical mapping $f(\cdot)$ by the neural networks $\mathscr{F}_\theta(\cdot)$. This powerful learning capability allows the neural networks to approximate the nonlinear physical mapping given sufficient training samples. As shown in the field example, the rule-based proxy model fails to provide reliable predictions when the input-output relationship is affected by poor data quality.

On the other hand, the number of trainable parameters in the regular GRU is comparable to that in the *phy*-GRU, whereas the GRU fails in extrapolations. The proposed *phy*-GRU takes the regular GRU as the prototype and makes minimum modifications to it. The only difference between these two models is the last step, which is converted from a weighted average in GRU to a skip connection in *phy*-GRU. The issue with the regular GRU lies in its last step, where the relationship between each two successive hidden states is equivalent to a backward Euler scheme of an exponential function (Appendix C). This equivalence with an exponential function limits the performance of the regular GRU model. On the contrary, the use of the skip connection allows the model to efficiently learn the trends in the time-series data.

Compared to *phy*-GRU, *phy*-RNN has a simpler architecture and generates output $\mathbf{y}$ that has a more direct relationship with the input $\mathbf{x}$. Furthermore, the *phy*-RNN can be viewed as a constrained *phy*-GRU, with the constraints of $\mathbf{r}_t = 1$ and $\mathbf{z}_t = 1$ enforced by the architecture of *phy*-RNN. Despite *phy*-GRU being a more general form than *phy*-RNN, the results for the simulated BHP data show that *phy*-RNN outperforms *phy*-GRU under different sizes of training samples. It suggests that the constraints on $\mathbf{r}_t$ and $\mathbf{z}_t$ introduced by the architecture of *phy*-RNN are more effective than those imposed by the training samples. Furthermore, the *phy*-RNN is designed based on the mass balance equation in the lumped parameter model that predicts the water level of wells in a low-temperature geothermal reservoir. This fit-for-purpose architecture allows *phy*-RNN to efficiently learn the input-output relationship from a relatively small training dataset.

5. Conclusion

This work proposes physics-guided RNN models (*phy*-RNN and *phy*-GRU) to handle long-term predictions and extrapolation tasks in

energy production from geothermal reservoirs. For comparison, we also apply two different models to the second experiment: a purely data-driven model (GRU) and a physics-based proxy model. The GRU model fares poorly in the extrapolation task and only works well in the Stationary case, as reflected in Fig. 5. As discussed in Section 3.2, the predictions of temperature and BHP time-series in geothermal reservoirs involve two types of extrapolations, which can create difficulty for the regular GRU model in capturing the long-term trends. This is because data-driven models, such as GRU, only rely on statistical input-output relationship and lack causality. The integration of physics or domain knowledge into the architecture can solve the limitations encountered by data-driven models and significantly improves the prediction performance of deep learning models. The physics-based proxy model incorporates the underlying governing equations with certain simplifications to provide better extrapolation capability. However, the strict design of the architecture reduces the learning capacity of the proxy model and limits its generalizability (e.g., to problems where the underlying physics my be uncertain).

Compared with the regular GRU and physics-based proxy, the proposed *phy*-RNNs exhibit extrapolation capability and generalizability as shown in our results. Through the integration of physics, *phy*-RNNs improves the extrapolation power of RNN and cangenerate more reliable long-term predictions for both temperature and BHP. The *phy*-RNNs can be applied to different types of trends with non-stationarity and under different ranges of input controls. Compared with the proxy model that is only limited to temperature and the GRU model that is only applicable to stationary cases, *phy*-RNNs can be applied to predict both temperature and BHP with various types of trends.

Although the physics-guided RNN retains most of the architecture of RNN and GRU models, more complex neural network architecture can be designed to approximate the underlying nonlinear mapping. Recent research has combined the skip connection with convolutional kernels to approximate the discretization of PDEs. This application might help the design of physics-guided deep learning models for predicting the entire geothermal reservoirs and facilitate more complex problems such as inverse modeling and uncertainty quantification.

CRediT authorship contribution statement

Zhen Qin: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Anyue Jiang:** Software, Validation, Formal analysis. **Dave Faulder:** Software. **Trenton T. Cladouhos:** Data curation. **Behnam Jafarpour:** Conceptualization, Validation, Formal analysis, Resources, Writing – review & editing, Supervision.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Behnam Jafarpour reports financial support was provided by US Department of Energy. Behnam Jafarpour reports financial support was provided by Energi Simulation Foundation.

Data availability

The authors do not have permission to share data.

Acknowledgement

This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Geothermal Technologies Office award number DE-EE0008765. The authors acknowledge the support provided by Energi Simulation through an Industrial Research Chair on Subsurface Energy Data Science at USC. The simulated data in this work were generated using the TETRAD software.

Appendix A. Details of predictive models and hyperparameters

A.1. Regular GRU and RNN model

A.2. Implementation details of deep learning models

All three deep learning models used in the test cases consist of two layers, with the first layer being GRU. The second layer of these three models are GRU, *phy*-RNN and *phy*-GRU, respectively. The illustration of the two-layer *phy*-RNNs is shown in Fig. A-3. The single-layer *phy*-RNNs (Fig. A-4) are applied to the simulated and field examples. The physics-based Proxy model always consists of two layers, which are the *phy*-RNN for BHP prediction and Proxy for temperature prediction.

The following tables (Tables A-1 to A-4) provide additional information of predictive models used in different experiments, including test cases, simulated cases, and field case. The dash symbol " – " in the column "Component" means that the RNN-type models are single-layer. The learning rate is selected based on the training and validation losses. The deep learning models, including Proxy, RNNs and *phy*-RNNs, are trained on an Intel Core i7-9700K CPU @ 3.60 GHz with eight cores, with an installed memory (RAM) of 64.0 GB. The training speeds of each model for different experimental

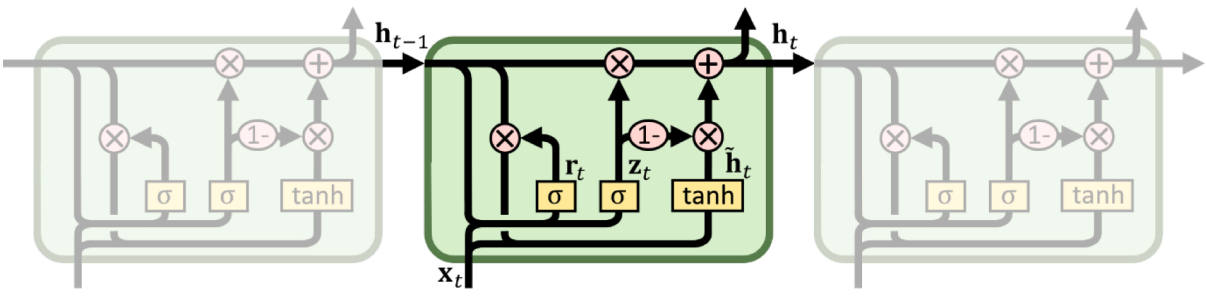


Fig. A-1. An illustration of GRU.

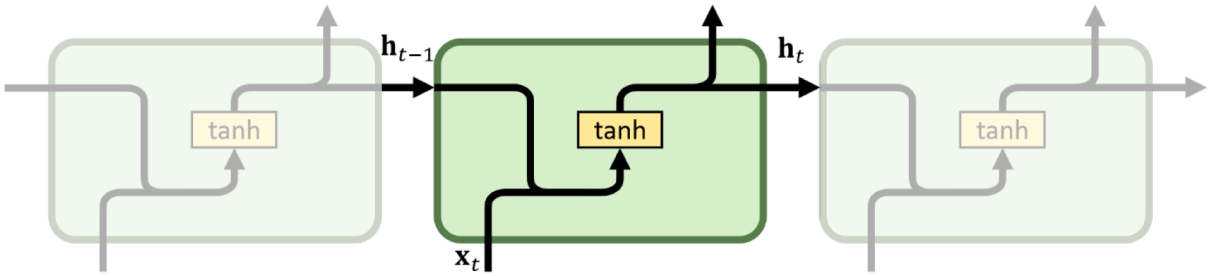


Fig. A-2. An illustration of RNN.

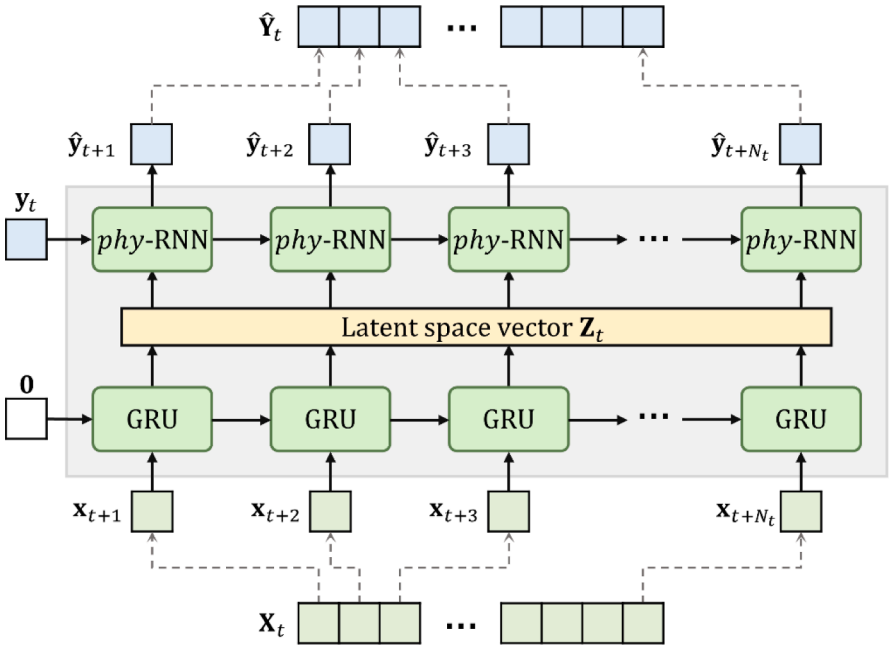


Fig. A-3. Illustration of the two-layer RNN model applied to the test cases. The "*phy*-RNN" denotes the proposed physics-guided RNN or physics-guided GRU. The input **0** to the GRU is the initial hidden state, which is a vector of zeros by default.

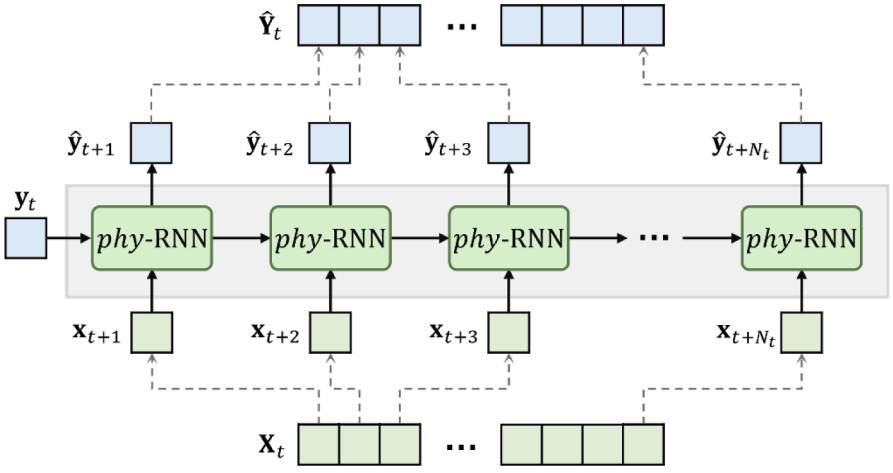


Fig. A-4. Illustration of the single-layer *phy*-RNNs applied to simulated and field case. The "*phy*-RNN" denotes the proposed physics-guided RNN or physics-guided GRU.

cases are presented in the following tables. The training speed is measured in milliseconds (ms) per sample (step), with a batch size of 1. The computational cost of training in this work are trivial for all different models due to the simplicity of their architectures and the small number of parameters involved. Additionally, the memory usage of CPU is negligible for all the deep learning models in this work due to their lightweight architectures.

A.3. Equations of regular RNNs and physics-guided RNNs

Table A-1
Detailed description of GRU model in different experiments.

	Component	Number of Parameters	Learning Rate	Training Speed (ms/step)
Test Cases	GRU (First Layer)	54	0.0035	9
	GRU (Second Layer)	18		
Simulated Temperature	-	204	0.005	7
Simulated BHP	-	336	0.005	9

Table A-2
Detailed description of ***phy***-RNN model in different experiments.

	Component	Number of Parameters	Learning Rate	Training Speed (ms/step)
Test Cases	GRU (First Layer)	54	0.0035	5
	phy-RNN (Second Layer)	7		
Simulated Temperature	-	69	0.005	3
Simulated BHP	-	113	0.005	5
Field Case	-	127	0.0005	6

Table A-3
Detailed description of ***phy***-GRU model in different experiments.

	Component	Number of Parameters	Learning Rate	Training Speed (ms/step)
Test Cases	GRU (First Layer)	54	0.0035	9
	phy-GRU (Second Layer)	19		
Simulated Temperature	-	205	0.005	7
Simulated BHP	-	337	0.005	9
Field Case	-	379	0.0005	11

Table A-4
Detailed description of proxy model in different experiments.

	Component	Number of Parameters	Learning Rate	Training Speed (ms/step)
Simulated Temperature	*phy*-RNN (First Layer)	265	0.0035	15
	Proxy Unit (Second Layer)	78		
Field Case	*phy*-RNN (First Layer)	365	0.0005	18
	Proxy Unit (Second Layer)	140		

Table A-5
Detailed description of equations in regular RNNs and *phy*-RNNs.

	GRU	***phy***-GRU
Equations	$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r)$ $\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z)$ $\widetilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b})$ $\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \circ \widetilde{\mathbf{h}}_t$	$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r)$ $\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z)$ $\widetilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b})$ $\mathbf{h}_t = \mathbf{h}_{t-1} + \delta t \cdot \mathbf{z}_t \circ \widetilde{\mathbf{h}}_t$
	RNN	***phy***-RNN
Equations	$\mathbf{h}_t = \tanh(\mathbf{W}_h\mathbf{h}_{t-1} + \mathbf{W}_x\mathbf{x}_t + \mathbf{b})$	$\mathbf{h}_t = \mathbf{h}_{t-1} + \delta t \cdot \tanh(\mathbf{W}_h\mathbf{h}_{t-1} + \mathbf{W}_x\mathbf{x}_t + \mathbf{b})$

Appendix B. Results of production temperatures

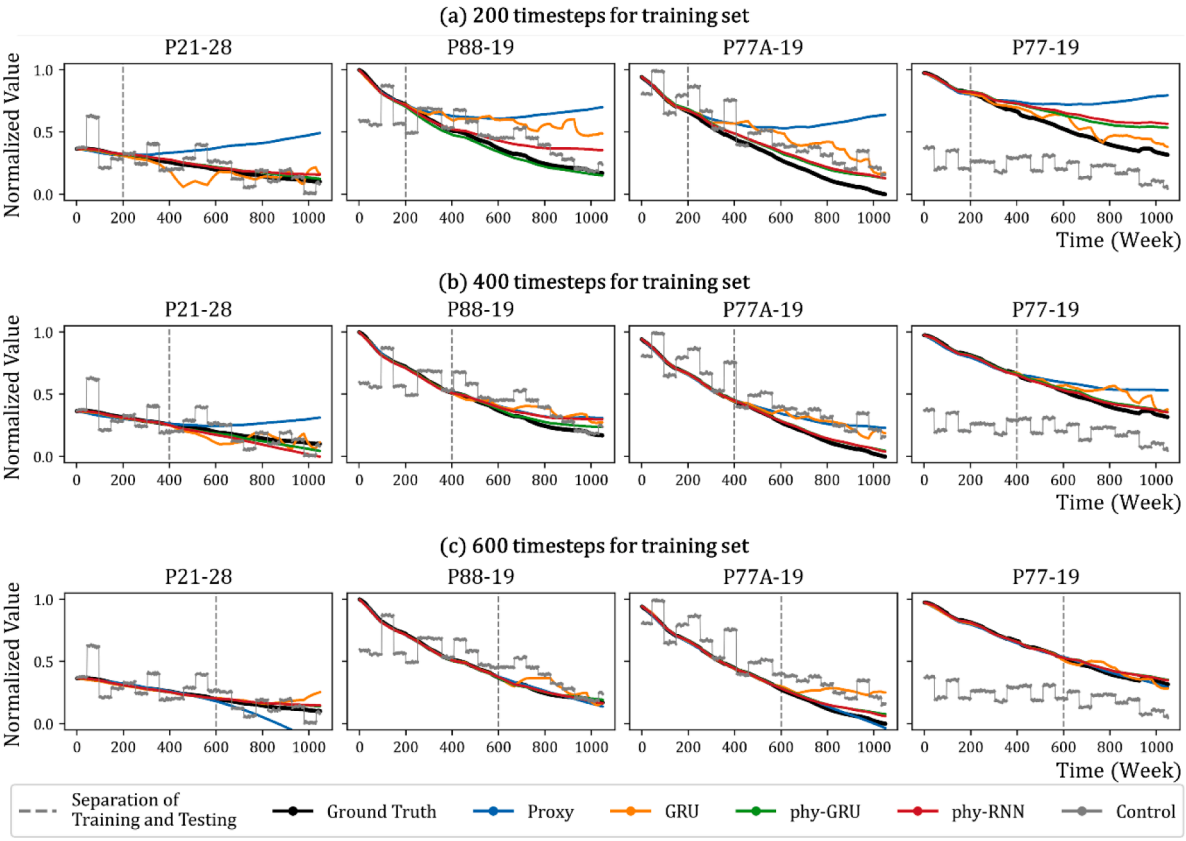


Fig. B-1. Prediction results of production temperature for the control pattern B.

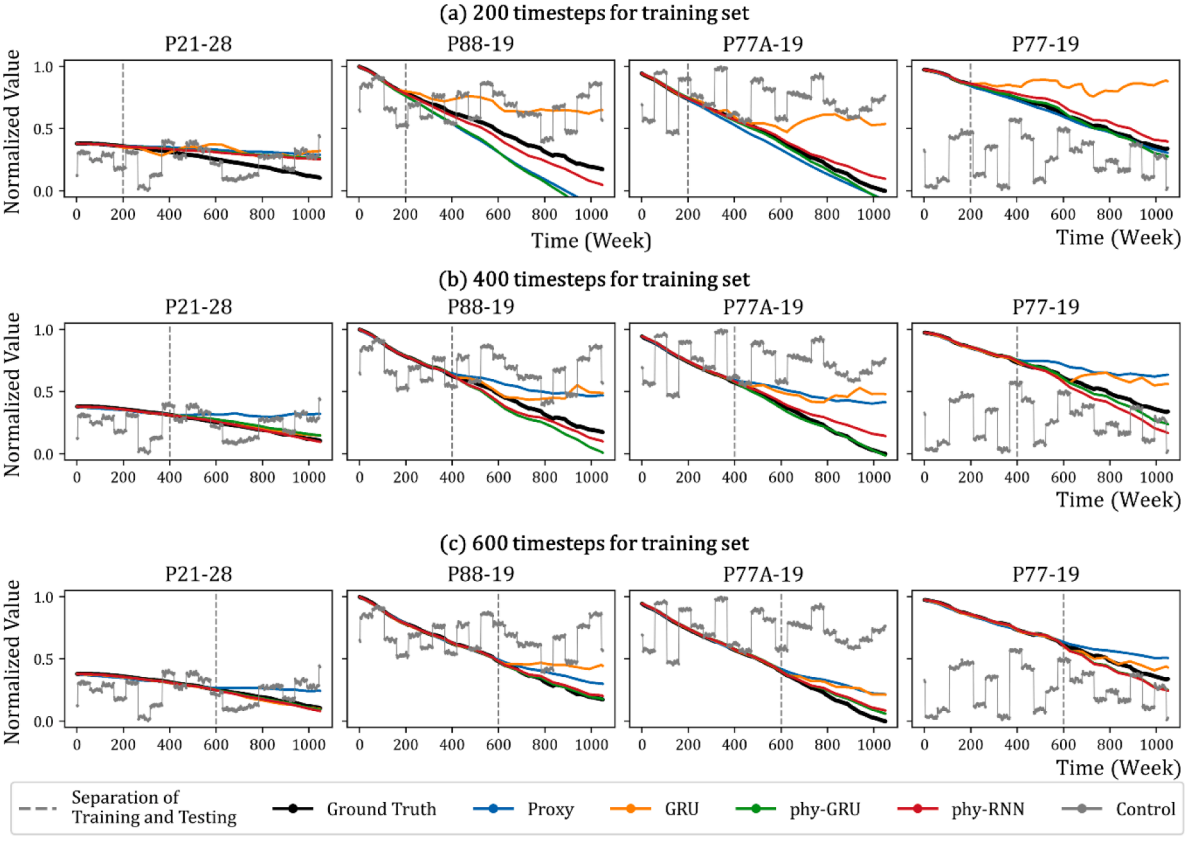


Fig. B-2. Prediction results of production temperature for the control pattern C.

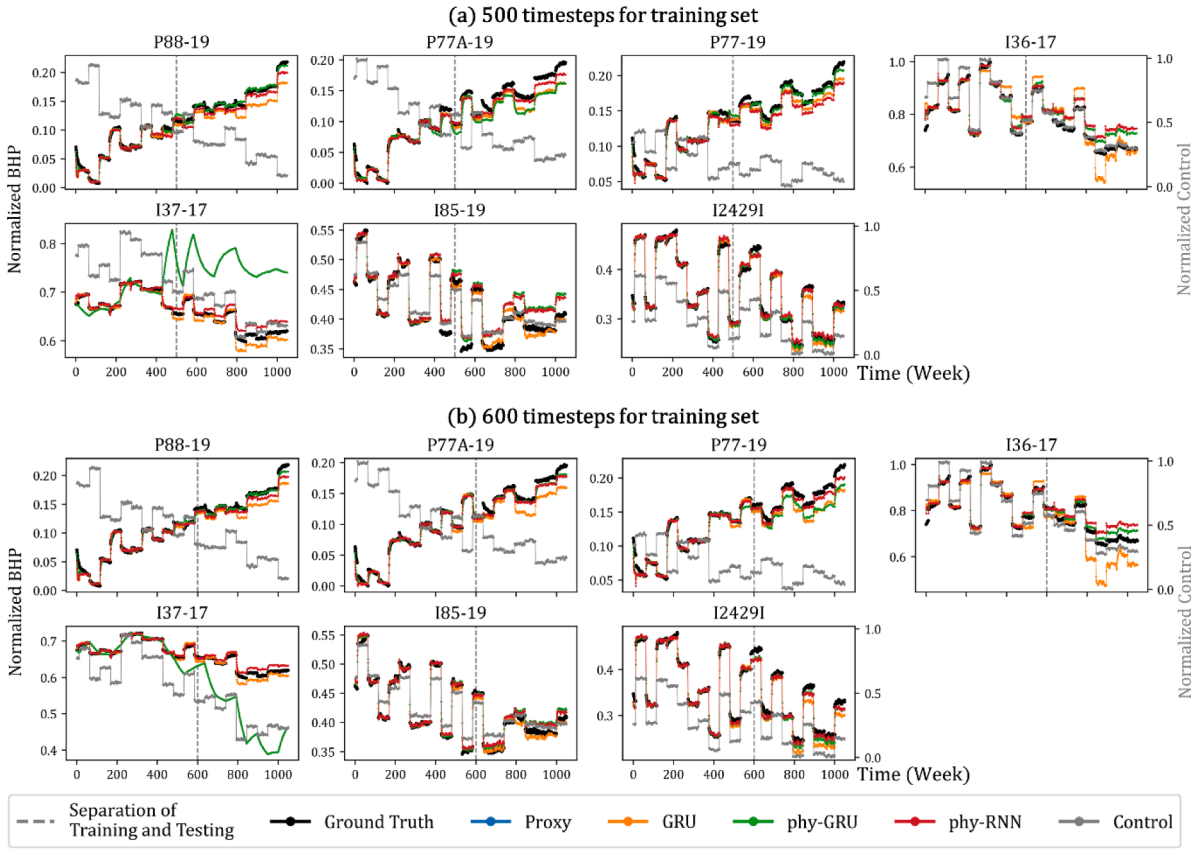


Fig. B-3. Prediction results of BHP for the control pattern B.

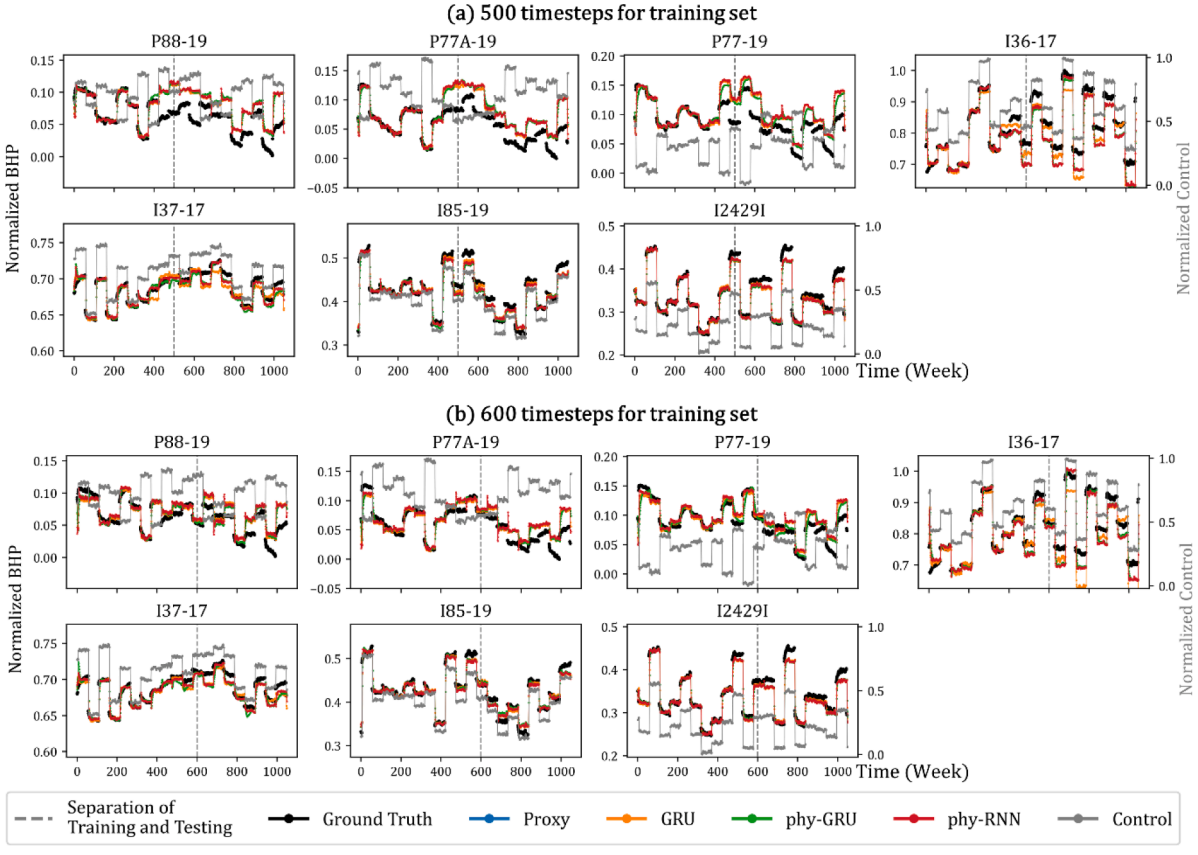


Fig. B-4. Prediction results of BHP for the control pattern C.

Appendix C. Default behavior of GRU

In this section, we explore the GRU model for its exponential-like decaying behavior. Given a simple exponential function $y = \exp(-a{\cdot}t)$, we can concert it to an ODE as follow:

$$\frac{dy}{dt} = -a{\cdot}y, \tag{C-1}$$

Then, we solve it numerically by using backward Euler method for time $t \in [0, +\infty)$ and constant coefficient $a \in (0, +\infty)$. The resulting numerical solution for the exponential function is shown as follow:

$$y_n = \frac{y_{n-1}}{(1 + a{\cdot}\Delta t_n)}, \tag{C-2}$$

where n denotes the n-th time step. The Equation (C-2) can also be rewritten as follow:

$$y_n = z_n{\cdot}y_{n-1}, \tag{C-3}$$

where

$$z_n = \frac{1}{1 + a{\cdot}\Delta t_n}, \quad z_n \in [0, 1). \tag{C-4}$$

As shown in Equation (C-3), there is a linear relationship between the current state y_n and previous state y_{n-1}, with a multiplier of z_n. It is noted that the last step of GRU, shown in Equation (4), has the same relationship between the previous state and current state. The only difference between Equation (4) and Equation (C-3) is the second term $(1 - \mathbf{z}_t) \circ \tilde{\mathbf{h}}_t$ in the Equation (4). However, this term only affects the convergence value of the GRU, not its decaying behavior. The multiplier z_n in the Equation (C-3) and the variable $\mathbf{z}_t$ in the GRU are not identical in the sense that $\mathbf{z}_t$ is a function of $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$, despite the same range for both two coefficients. However, the variable $\mathbf{z}_t$ of the GRU amounts to a time-variant coefficient a or a changing time step for the exponential function $\exp(-a{\cdot}t)$.

A simple example is shown below that connects the predictions from regular GRU to an exponential function (Fig. C-1). The GRU model is trained on a straight line with a certain slope over four thousand time steps. The output from the GRU plateaus after around 1500 time steps. We then remove the second term $(1 - z_t) \circ \tilde{h}_t$ in Equation (4) and only keep the first term $z_t \circ h_{t-1}$, which is shown as red curve in Fig. C-1. By converting the multiplier z_t to $a{\cdot}\Delta t_n$ using Equation (C-4), we can represent the first term $h_t = z_t \circ h_{t-1}$ by using the exponential function $y_n = \exp(-a{\cdot}\Delta t_n)$, which is shown as green dash curve in Fig. C-1. It can be observed that the curve $h_t = z_t \circ h_{t-1}$ is equivalent to $y_n = \exp(-a{\cdot}\Delta t_n)$ once we find the corresponding values for Δt_n.

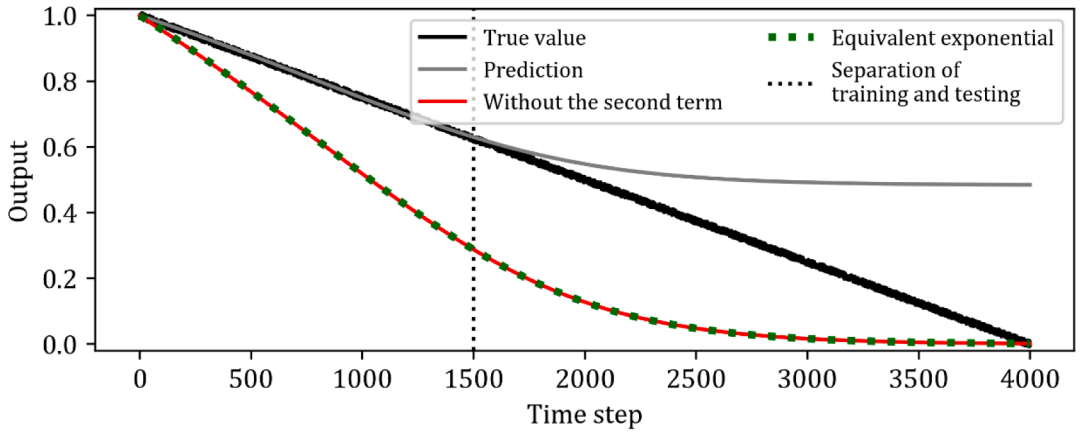


Fig. C-1. Illustration of default behavior of the regular GRU model. The first composition $z_t \circ h_{t-1}$ of the last step within the GRU model is compared with an exponential function, of which the parameter is a function of first composition z_t.

References

Alkan, H., Satman, A., 1990. A new lumped parameter model for geothermal reservoirs in the presence of carbon dioxide. Geothermics 19, 469–479. https://doi.org/10.1016/0375-6505(90)90059-K.

Asher, M.J., Croke, B.F.W., Jakeman, A.J., Peeters, L.J.M., 2015. A review of surrogate models and their application to groundwater modeling. Water Resour. Res. 51, 5957–5973. https://doi.org/10.1002/2015WR016967.

Axelsson, G., 1989. Simulation of Pressure Response Data from Geothermal Reservoirs By Lumped Parameter Models. Stanford University, pp. 257–263.

Balestriero, R., Pesenti, J., LeCun, Y., 2021. Learning in high dimension always amounts to extrapolation. doi:10.48550/ARXIV.2110.09485.

Chen, M., Tompson, A.F.B., Mellors, R.J., Abdalla, O., 2015. An efficient optimization of well placement and control for a geothermal prospect under geological uncertainty. Appl. Energy 137, 352–363. https://doi.org/10.1016/j.apenergy.2014.10.036.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. doi:10.48550/ARXIV.1406.1078.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555 [cs].

Ciriaco, A.E., Zarrouk, S.J., Zakeri, G., 2020. Geothermal resource and reserve assessment methodology: overview, analysis and future directions. Renew. Sustain. Energy Rev. 119, 109515 https://doi.org/10.1016/j.rser.2019.109515.

Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F., 2022. Scientific machine learning through physics–informed neural networks: where we are and what's next. J. Sci. Comput. 92 (88) https://doi.org/10.1007/s10915-022-01939-z.

Daw, A., Thomas, R.Q., Carey, C.C., Read, J.S., Appling, A.P., Karpatne, A., 2020. Physics-guided architecture (PGA) of neural networks for quantifying uncertainty in lake temperature modeling. In: Proceedings of the 2020 SIAM International Conference on Data Mining (SDM), Proceedings. Society for Industrial and Applied Mathematics, pp. 532–540. https://doi.org/10.1137/1.9781611976236.60.

Gudmundsdottir, H., Horne, R.N., 2020. Prediction modeling for geothermal reservoirs using deep learning. PROCEEDINGS, 45th Workshop on Geothermal Reservoir Engineering Stanford University, Stanford, California, February 10-12, 2020, SGP-TR-216.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. pp. 770–778.

E, W., 2017. A proposal on machine learning via dynamical systems. Commun. Math. Stat. 1, 1–11.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Holanda, R.W.de, Gildin, E., Jensen, J.L., Lake, L.W., Kabir, C.S., 2018. A state-of-the-art literature review on capacitance resistance models for reservoir characterization and performance forecasting. Energies 11, 3368. https://doi.org/10.3390/en11123368.

Jiang, A., Jafarpour, B., 2021a. Inverting subsurface flow data for geologic scenarios selection with convolutional neural networks. Adv. Water Resour. 149, 103840 https://doi.org/10.1016/j.advwatres.2020.103840.

Jiang, A., Jafarpour, B., 2021b. Deep convolutional autoencoders for robust flow model calibration under uncertainty in geologic continuity. Water Resour. Res. 57 https://doi.org/10.1029/2021WR029754.

Jiang, A., Qin, Z., Cladouhos, T.T., Faulder, D., Jafarpour, B., 2021. Recurrent neural networks for prediction of geothermal reservoir performance. PROCEEDINGS, 46th Workshop on Geothermal Reservoir Engineering Stanford University, Stanford, California, February 15-17, 2021, SGP-TR-218.

Jiang, A., Qin, Z., Faulder, D., Cladouhos, T.T., Jafarpour, B., 2022. Recurrent neural networks for short-term and long-term prediction of geothermal reservoirs. Geothermics 104, 102439. https://doi.org/10.1016/j.geothermics.2022.102439.

Jiang, A., Qin, Z., Faulder, D., Cladouhos, T.T., Jafarpour, B., 2023. A multiscale recurrent neural network model for predicting energy production from geothermal reservoirs. Geothermics 110, 102643. https://doi.org/10.1016/j.geothermics.2022.102643.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nat. Rev. Phys. 3, 422–440. https://doi.org/10.1038/s42254-021-00314-5.

Kim, J., Choe, J., Lee, K., 2022. Sequential field development plan through robust optimization coupling with CNN and LSTM-based proxy models. J. Petrol. Sci. Eng. 209, 109887 https://doi.org/10.1016/j.petrol.2021.109887.

Kim, Y.D., Durlofsky, L.J., 2021. A recurrent neural network–based proxy model for well-control optimization with nonlinear output constraints. SPE J. 26, 1837–1857. https://doi.org/10.2118/203980-PA.

Kingma, D.P., Ba, J., 2014. Adam: a method for stochastic optimization. doi:10.48550/ARXIV.1412.6980.

Laloy, E., Hérault, R., Lee, J., Jacques, D., Linde, N., 2017. Inversion using a new low-dimensional representation of complex binary geological media based on a deep neural network. Adv. Water Resour. 110, 387–405. https://doi.org/10.1016/j.advwatres.2017.09.029.

Lerlertpakdee, P., Jafarpour, B., Gildin, E., 2014. Efficient production optimization with flow-network models. SPE J. 19, 1083–1095. https://doi.org/10.2118/170241-PA.

Li, Y., Júlíusson, E., Pálsson, H., Stefánsson, H., Valfells, Á., 2017. Machine learning for creation of generalized lumped parameter tank models of low temperature geothermal reservoir systems. Geothermics 70, 62–84. https://doi.org/10.1016/j.geothermics.2017.05.009.

Long, Z., Lu, Y., Dong, B., 2019. PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network. J. Comput. Phys. 399, 108925 https://doi.org/10.1016/j.jcp.2019.108925.

Long, Z., Lu, Y., Ma, X., Dong, B., 2018. PDE-Net: learning PDEs from data. In: Proceedings of the 35th International Conference on Machine Learning. PMLR, pp. 3208–3216.

Lu, Y., Zhong, A., Li, Q., Dong, B., 2018. Beyond finite layer neural networks: bridging deep architectures and numerical differential equations. In: Proceedings of the 35th International Conference on Machine Learning. PMLR, pp. 3276–3285.

Markidis, S., 2021. The old and the new: can physics-informed deep-learning replace traditional linear solvers? Front. Big Data 4, 669097. https://doi.org/10.3389/fdata.2021.669097.

Muther, T., Dahaghi, A.K., Syed, F.I., Van Pham, V., 2023. Physical laws meet machine intelligence: current developments and future directions. Artif. Intell. Rev. 56, 6947–7013. https://doi.org/10.1007/s10462-022-10329-8.

Qin, T., Wu, K., Xiu, D., 2019. Data driven governing equations approximation using deep neural networks. J. Comput. Phys. 395, 620–635. https://doi.org/10.1016/j.jcp.2019.06.042.

Qin, Z., Jiang, A., Faulder, D., Cladouhos, T.T., Faulder, D., Jafarpour, B., 2022. Physics-guided deep learning for prediction of geothermal reservoir performance. In: 47th Workshop on Geothermal Reservoir Engineering Stanford University. Stanford, California, pp. 1–10.

Qin, Z., Jiang, A., Faulder, D., Cladouhos, T.T., Jafarpour, B., 2023. Efficient optimization of energy recovery from geothermal reservoirs with recurrent neural network predictive models. Water Resour. Res. 59 https://doi.org/10.1029/2022WR032653 e2022WR032653.

Quinao, J.J.D., Zarrouk, S.J., 2018. Geothermal resource assessment using experimental design and response surface methods: the Ngatamariki geothermal field, New Zealand. Renew. Energy 116, 324–334. https://doi.org/10.1016/j.renene.2017.09.084.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045.

Razak, S.M., Jafarpour, B., 2020. Convolutional neural networks (CNN) for feature-based model calibration under uncertain geologic scenarios. Comput. Geosci. 24, 1625–1649. https://doi.org/10.1007/s10596-020-09971-4.

Ruthotto, L., Haber, E., 2020. Deep neural networks motivated by partial differential equations. J. Math. Imaging Vis. 62, 352–364. https://doi.org/10.1007/s10851-019-00903-1.

Sarak, H., Onur, M., Satman, A., 2005. Lumped-parameter models for low-temperature geothermal fields and their application. Geothermics 34, 728–755. https://doi.org/10.1016/j.geothermics.2005.09.001.

Schulte, D.O., Arnold, D., Geiger, S., Demyanov, V., Sass, I., 2020. Multi-objective optimization under uncertainty of geothermal reservoirs using experimental design-based proxy models. Geothermics 86, 101792. https://doi.org/10.1016/j.geothermics.2019.101792.

Shi, Y., Song, X., Song, G., 2021. Productivity prediction of a multilateral-well geothermal system based on a long short-term memory and multi-layer perceptron combinational neural network. Appl. Energy 282, 116046. https://doi.org/10.1016/j.apenergy.2020.116046.

Sigurdardottir, S.R., Valfells, A., Palsson, H., Stefansson, H., 2015. Mixed integer optimization model for utilizing a geothermal reservoir. Geothermics 55, 171–181. https://doi.org/10.1016/j.geothermics.2015.01.006.

Sirignano, J., Spiliopoulos, K., 2018. DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. 375, 1339–1364. https://doi.org/10.1016/j.jcp.2018.08.029.

Sun, J., Niu, Z., Innanen, K.A., Li, J., Trad, D.O., 2020. A theory-guided deep-learning formulation and optimization of seismic waveform inversion. GEOPHYSICS 85, R87–R99. https://doi.org/10.1190/geo2019-0138.1.

Tian, C., Horne, R.N., 2017. Recurrent neural networks for permanent downhole gauge data analysis. Day 1 Mon. SPE, San Antonio, Texas, USA. https://doi.org/10.2118/187181-MS. October 09, 2017D011S008R007.

Tureyen, O.I., Akyapı, E., 2011. A generalized non-isothermal tank model for liquid dominated geothermal reservoirs. Geothermics 40, 50–57. https://doi.org/10.1016/j.geothermics.2010.10.004.

Wang, N., Zhang, D., Chang, H., Li, H., 2020. Deep learning of subsurface flow via theory-guided neural network. J. Hydrol. (Amst) 584, 124700. https://doi.org/10.1016/j.jhydrol.2020.124700.

Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V., 2021. Integrating scientific knowledge with machine learning for engineering and environmental systems. arXiv:2003.04919 [physics, stat].

Yang, S., Yu, X., Zhou, Y., 2020. LSTM and GRU neural network performance comparison study: taking yelp review dataset as an example. In: 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), pp. 98–101. https://doi.org/10.1109/IWECAI50956.2020.00027.

Yousef, A.A., Gentil, P., Jensen, J.L., Lake, L.W., 2006. A capacitance model to infer interwell connectivity from production- and injection-rate fluctuations. SPE Reserv. Eval. Eng. 9, 630–646. https://doi.org/10.2118/95322-PA.

Yu, J., Jafarpour, B., 2022. Active learning for well control optimization with surrogate models. SPE J. 1–21 https://doi.org/10.2118/209191-PA.

Zhao, H., Kang, Z., Zhang, X., Sun, H., Cao, L., Reynolds, A.C., 2016. A physics-based data-driven numerical model for reservoir history matching and prediction with a field application. SPE J. 21, 2175–2194. https://doi.org/10.2118/173213-PA.

Zhao, H., Kang, Z., Zhang, X., Sun, H., Cao, L., Reynolds, A.C., 2015. INSIM: a data-driven model for history matching and prediction for water flooding monitoring and management with a field application. OnePetro.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., Perdikaris, P., 2019. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. J. Comput. Phys. 394, 56–81. https://doi.org/10.1016/j.jcp.2019.05.024.

Zubarev, D.I., 2009. Pros and cons of applying proxy-models as a substitute for full reservoir simulations. All Days. SPE, New Orleans, Louisiana, 124815. https://doi.org/10.2118/124815-MS. SPE–-MS.