33rd Annual INCOSE international symposium
hybrid event
Honolulu, HI, USA
July 15 - 20, 2023

Physics-Informed Gas Lifting Oil Well Modelling using Neural Ordinary Differential Equations

Zhe Ban
University of South-Eastern Norway
Kjølnes Ring 56, 3918 Porsgrunn
+47 96950389
zba@usn.com

Carlos Pfeiffer
University of South-Eastern Norway
Kjølnes Ring 56, 3918 Porsgrunn
+47 35575157
carlos.pfeiffer@usn.com

Abstract. Modelling of oil well systems is important for a wide range of petroleum scientific and oil industrial processes. Considering the uncertainty of the measurements and the demand for empirical knowledge, a purely first-principle model and a black-box model based on data are not sufficient for accurately describing an oil well system. Thus, there is a growing body of literature that recognizes the importance of data-driven methods combined with physical knowledge. However, the application of combination methods for dynamic nonlinear systems is still challenging. In this work, we demonstrate the application of a physics-informed neural network to a gas lifting oil well system. The neural ordinary differential equation is the main tool for the modeling and the simulation is examined in Julia programming language. The advantage and drawbacks of the physics-informed data-driven method are analyzed.

Introduction

Accurate estimation of outputs and states in engineering systems plays a vital role in model-based control, optimization and maintenance. Due to model mismatch, random errors and gross errors, most first-principles models cannot precisely describe the system behavior in real life (Knopt 2011). For some complex systems, it might be difficult to describe part of the system due to a lack of physics knowledge. On the other hand, it is difficult for most black-box models to extract interpretable information by learning from observational data (Karniadakis et al. 2021). Therefore, the grey-box model combining physics knowledge and the information from data has been thought of as a crucial solution in the modelling of engineering systems.

There is a growing body of literature that provides a large variety of approaches for different tasks in terms of merging scientific knowledge and machine learning (Bikmukhametov and J¨aschke 2020, Gross et al. 2021, Franklin et al. 2022). In most applications of physics-informed machine learning in oil and gas production systems, the learning algorithms were improved using either physics-informed data or physics-informed models. Thanks to the development of scientific machine learning, it is possible to incorporate more physical knowledge by considering dynamics which are differentiable. In physics-informed machine learning algorithms, physical constraints can be added to the loss function. Prior information can be integrated by designing architecture when the mechanistic model and machine learning were hybridized.

Among the scientific machine learning methods, neural ordinary different equation (Chen et al. 2018) was designed for solving ordinary differential equations (ODEs) by parameterizing the derivative of the hidden states using a continuous depth neural network. A differential equation solver was utilized to calculate the output of the continuous depth network. The adjoint sensitivity method is commonly

adopted to compute gradients, which lowers the memory and computational cost. By substituting ODEs for residual networks, time-dependent dynamics is described continuously. Meanwhile, neural ODEs also allow scalable backpropagation through ODE solvers.

SciML (Christopher Rackauckas, Y. Ma, et al. 2020) was the first differential equation library that generalized packages for various training problems and allowed the application of universal differential equations, including neural ODE, partial differential equations, differential-algebraic equations, stochastic differential equations, delay differential equations and stiff equations. Some packages related to neural ODE in Julia (Bezanson et al. 2017) were chosen in this work. For example, OrdinaryDiffEq.jl in DifferentialEquations.jl (Christopher Rackauckas and Nie 2017) was used for providing the ordinary differential equation solvers. DiffEqFlux.jl (Chris Rackauckas et al. 2019) and Flux.jl (Inneset al. 2018) were utilized for integrating the first-principle model and neural network, as well as optimization of the neural network.

The rest of this paper is organized as follows. In the next section, the gas lifting oil well system and its first-principle model are introduced. Following that, in Section 3, the physics-informed modelling structure, data preparation and the method were explained for improving training results. In Section 4, the performance of the trained model is tested with the dataset collected from the gas lifting oil well simulator. Finally, in Section 5, we drew conclusions and discussed the advantage and drawbacks of the modelling, as well as the future research direction.

Gas Lifting Oil Well Simulator

The gas lifting oil well model used in this work is based on a single oil well system, which is inspired by a modelling work by Sharma (Sharma et al. 2011). Details of the oil well model can be checked in a previous work (Ban et al. 2022). A schematic diagram of a gas lifting oil well is shown in Fig.1.

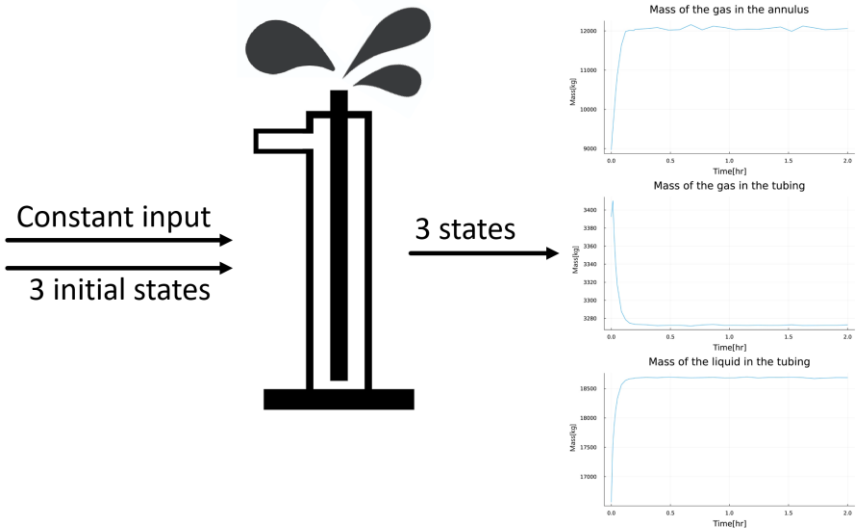

Figure 1. A Single Gas Lifting Oil Well System

During the process of oil and gas production, valves are utilized to control the flow rates, which generate inputs to the oil well system. The input of the gas lifting oil well simulator is the valve opening of the gas lift choke valve, u.

The oil well has some parameters which change during the process. The water cut, WC, is the volume of water produced compared to the volume of total liquids produced from an oil well. The productivity index, PI [$\frac{kg/hr}{bar}$], is a mathematical means of expressing the ability of a reservoir to deliver fluids to the wellbore. GOR is the mass ratio of produced gas to produced liquid (oil and water).

In this study, we focus on state estimation of the gas lifting oil well. These states are the mass of gas in the annulus m_{ga}, in the tubing above the injection point m_{gt} and the mass of liquid in the tubing above injection m_{lt}. According to the mass balance, these states can be calculated based on three ODEs, which are mainly present in Eq. (1):

$$\begin{aligned} \dot{m}_{ga} &= w_{ga} - w_{ginj}, \\ \dot{m}_{gt} &= w_{ginj} + w_{gr} - w_{gp}, \\ \dot{m}_{lt} &= w_{lr} - w_{lp} \end{aligned} \tag{1}$$

where w_{ga} is the flow rate of the gas through the gas lift choke valve which is injected into the annulus. The flow rates of the lift gas from the annulus and reservoir to the tubing are w_{ginj} and w_{gr} respectively. The flow rate of produced gas through the production choke valve is presented as w_{gp}. w_{lr} and w_{lp} are the liquid phase flow from the reservoir into the well and through the production choke valve, respectively. Some of these flow rates are the measurements in the gas and lifting oil well. The rest of the flow rates can be calculated by auxiliary algebra equations which are assembled in the simulator. In this work, we used the simulator to generate datasets. The states, m_{ga}, m_{gt}, m_{lt}, are the features for the physics-informed neural network (PINN) model. The PINN model will be introduced in the methodology section. The initial states are the input and the time series data corresponding to the state dynamics are the observation.

Methodology

In this work, the PINN model plays an important role in the modeling and prediction of the oil well dynamics. This section begins by explaining the design of the PINN model and it will then go on to examine the PINN model using data collected from the aforementioned simulator.

PINN model. The PINN model contains certain layers including a physics model and a multi-layer perceptron. By using Flux. Chian (Inneset al. 2018), a physics model is integrated as the first layer in the PINN model and is written as Eq. (2). x is the state vector to be estimated. The subscript k denotes the time step. The input of the physics model is the current state vector and control input u. The output of the physics model is the derivatives of the state vector, which is then broadcasted to multiple dense layers. The physics model and multiple layers were called in sequence with the given states, followed by an ODE solver differentiating the output of the MLP as the prediction of the states at the next time step.

$$\dot{x} = f(x, u) \tag{3}$$

$$PINN(x_k, t_{k+1}) = \int \frac{d\hat{x}_{k+1}}{dt} = \int NN(f(x_k, t_k), t_{k+1})$$

After calculating the estimation for the whole time series, the outputs of PINN and the samples were used in the loss function for training. Figure 3 illustrates the outline of the PINN model. The upper loop ① indicates the process where the PINN model predicts the time series data based on the previous time step and parameters of multilayer perceptron (MLP) (Noriega 2005). The state of the next time step was calculated in each loop and the iteration stop until data for all time steps is calculated. The loop ② illustrates the epoch during training.

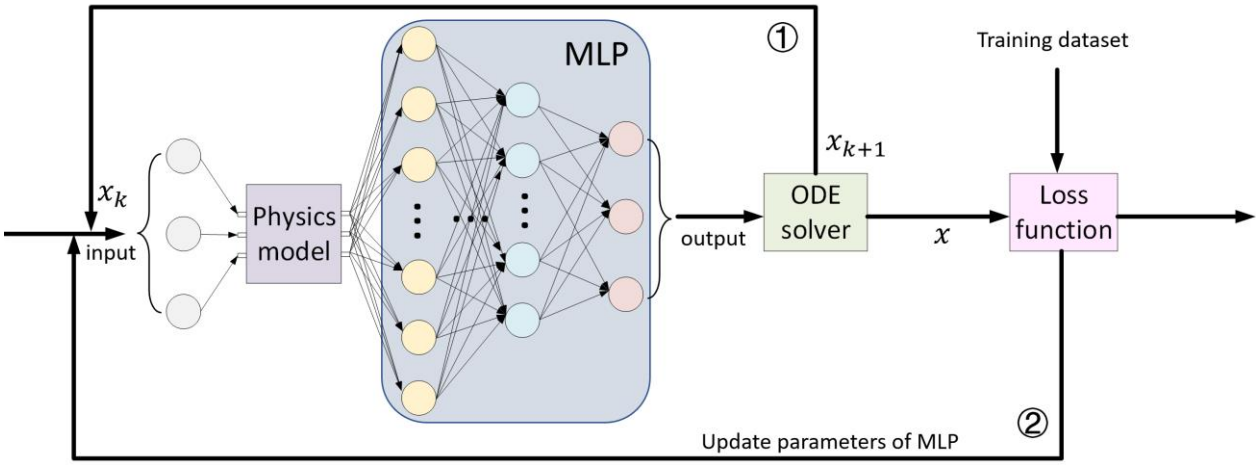

Figure 2. The Outline of the PINN.

A physics model. As a part of the PINN, an ODE model including physical knowledge is expected to generate better performance than the one from a pure neural network. In Fig. 2, the physics model is presented as the purple block to provide input to the MLP. The physics model provides not only the state estimation at the next time step by solving the ODEs, but also the constraints including prior knowledge of the initial states and input, u. The physical model used here is not necessary to be accurate to describe the whole system. The unknown part and uncertainty can be learnt during the training of the PINN model using training data.

The physics model was designed based on part of the physical knowledge of the gas lifting oil well system but lacks some information. For example, we assumed that we did not know the true value of the three parameters, WC, PI, and GOR, in the oil well simulator. Because the neural network proposes training weights randomly during the training process and it is impossible to apply physics constraints for the raining weights of the neural network, the root square parts in the simulator were ignored to avoid calculation error. Compared with the simulator, the physics model is lack of some physics knowledge and does not contain any noise. The different between the output of the physics model and the simulator is presented as residual between the output of the neural network and the training data. A neural network is designed to learn the difference.

Neural Network design. A MLP with dense layers was used in PINN, denoted as Eq. (3). The PINN approximates the solution of the gas lifting oil well system by updating the parameters of the neural network to minimize a loss function using optimizer ADAM and BFGS.

$$NN(x) = W_n \dots \sigma_2(W_2\sigma_1(W_1x + b_1) + b_2) \dots + b_n \tag{3}$$

, where W_i are the weight matrices, b_i are the bias vector and σ_i are activation functions. The input of the neural network includes u and state at current time step.

When a neural network is trained using time series data, the neural network needs to be run through the sequence data, which leads to a deep network. Exploding gradients problem (Geron 2017, Glorot et al. 2021) is one of the most frequently stated challenges during training machine learning models over time steps. The problem was alleviated by modification of sample size, employing feature scaling during data preprocessing and using a nonsaturating activation function for dense layer design in this work, which will be explained in the validation subsection.

A loss function is designed to include the model information and constraints based on mean squared error between the output of the neural network and the dataset. Inspired by the previous neural ODE work (Christopher Rackauckas 2022), the loss function for each sample can be designed as Eq. (4). p is the parameter vector of MLP. The first term, $NN(0) - x_0$, is for satisfying the initial constraint.

The rest part of the loss function is for learning the outputs with the physics model. The residual of the physics model was combined within the mismatch in the training data on the state variables.

$$L(p) = (NN(0) - x_0)^2 + \sum_k (PINN(t_{k+1}) - x_{k+1})^2 \tag{4}$$

ADAM is a widely used method for machine learning in the last decade. In this optimization algorithm, every parameter is provided with an individual learning rate. The gradient of loss functions can be accurately and efficiently computed, even with noisy and sparse data. BFGS (Liu and Nocedal 1989) is a local search optimization algorithm using the second-order derivative of a loss function. In this work, the loss function is minimized iteratively using the ADAM algorithm and BFGS algorithm. To avoid the local minimum and find the optimum solution rapidly, the ADAM algorithm was run first and then the BFGS algorithm was used to precisely search around the minimum area within a small range.

System architecture. The process of examining the PINN model is presented as system architecture here. The outline of the system architecture is shown in Fig. 3. The first step in this process is data collection for training the PINN model. To begin the process, the gas lifting oil well simulator was run to collect data. The collected dataset was noted as D_i. The blue, pink and purple blocks present the training, validation and test dataset respectively. The physics model is designed for providing physical information to followed dense layers. The output of the physics model generates important insight into the trajectory of the dataset in terms of the trend and shape. The residual between the output of the physics model and the training data is learnt by the rest of the PINN model. Meanwhile, the physics model is adopted here to demonstrate the PINN model is able to learn and predict in the case lack of physics information. Following the preparation of the structure of the PINN model, the activation functions were chosen for each dense layers the numbers of the layers and nodes were tuned. The bias of each layer is trainable.

Once the data and the PINN are ready, the training process starts. During the training process, The PINN model generates outputs in every epoch using initial states and proposed parameters. Once the outputs are generated, the loss function is calculated based on the outputs of the neural network and data. Parameters of the PINN are updated using optimization algorithms at every epoch. When the model is trained, the test data set is used for model evaluation.

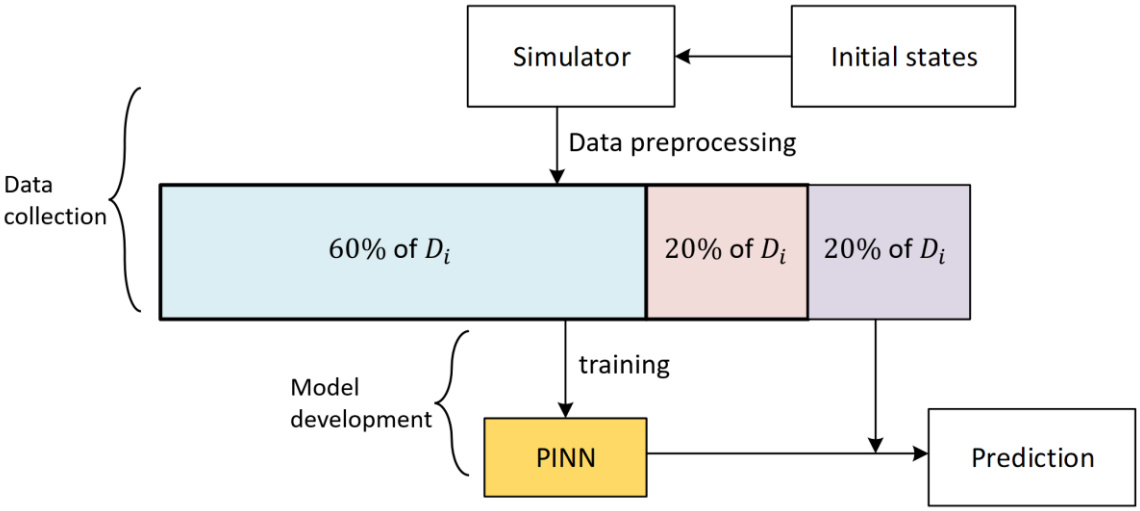

Figure 3. System Architecture

Data collection. In this work, datasets were collected from a gas lifting oil well simulator in different scenarios. In every scenario, the simulator run with random initial states within the prior range and the states were collected as a dataset every time when the simulation finished. Scenarios were run

independently, and the corresponding time series datasets were independent. The collected data contain white noise which follows the normal distribution with zero mean. A 3D tensor was used to store these datasets. Features, timesteps and samples are the axes of the tensor. The dataset for one scenario is noted as one sample and presented as a 2D matrix with observations for every feature at every timestep.

The states in the gas lifting oil well are time-dependent. The trajectories of the observations contain transients and steady states. Trajectories in one of the scenarios are shown as blue lines in Fig. 1. As the transient contains more information about the system performance, the sampling time for the transient is smaller than the sampling time for the steady state. The simulator was run with variant sampling time for choosing the dataset size. The dataset size can be varied from different systems or experiments, but it is expected to be small to lower the computational cost during training. Meanwhile, the observations should also illustrate the state dynamics, namely the shape of the trajectory. Feature scaling was adopted to preprocess data before training. The range of the observation was normalized from rang [3000, 20000] to [0, 1] by as Eq. (5).

$$x_k = (z_k - \min(z))/(\max(z) - \min(z)) \tag{5}$$

, where z is observation and k is related to the number of the time step of the outputs.

One of reasons for feature scaling is that the optimization algorithm adopted in this work, ADAM (Kingma and Ba 2014), uses gradient descent. Values of features influence the step size of the gradient descent. Besides, due to the difference in the feature ranges, the step sizes for the gradient descent of features are updated at different paces. Therefore, feature scaling has a dramatic impact on the speed of gradient descent convergence. Feature scaling improves convergence significantly. Both normalization and standardization methods were tested in this work and unit range normalization was chosen to transform data. Compared with training with the original dataset, the accuracy of the training results was greatly improved with normalized data. Meanwhile, the speed of convergence was faster.

After data preprocessing, dataset for scenario i, $D_i = (x_0^i, x_k^i)$ was saved. x_0^i and x_k^i are corresponding to the initial state and the state output, where i is the number of the scenario. As the gas lifting oil well system has three states and the time steps of each simulation is N, each initial state is a 3×1 array and each sample is a $3 \times$ N matrix. These samples were split into 60%, 20% and 20% for training, validation and testing. The loss function was calculated using all training samples. Hyperparameters were tuned using validation dataset by calculating the scoring function. The test samples were then used to evaluate the performance of the PINN.

Validation. A scoring function was designed for comparing the performance of the PINN model with different hyperparameters, θ, including the number of epochs, the number of layers, the learning rate, the number of nodes in each layer and the activation function for each layer. The scoring function is calculated based on negative mean squared error (NMSE) and is denoted as Eq. (6). m is the number of samples.

$$S(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(PINN(x_0^i, t) - x^i)^2 \tag{6}$$

The number of epochs was designed by running the training with a relatively large training set size, 1000 epochs. The scoring function of the training dataset and validation dataset were checked and compared. The epoch which is corresponding to the divergence of the scoring function of training validation is chosen, namely stop training as soon as the scoring function of the validation shows a slower increase than the one of training. By implementing early stopping, the overfitting problem is mitigated and a nice generalization to the data which is outside of the training set is provided.

Activities functions in the MLP need to make sure that the dynamics are continuously differentiable and satisfy the Lipshitz condition(Kim 2021). ReLU and Tanh meet these conditions. In this work, ReLU is used for each layer as an activation function. The primary reason to use ReLU is that the functions do not saturate when the input values are large. Meanwhile, the computation of ReLU is quicker than other activation functions (Geron 2017).

The learning rate of the ADAM algorithm is another hyperparameter to tune. The PINN models with various learning rates were trained several times during a fixed epoch. After comparing the learning curves, a proper learning rate is chosen to provide a fast optimized solution. BFGS is set as default.

The number of layers and nodes were selected after training the PINN model with various layers and nodes. A simple MLP with one single layer is enough to achieve a reasonable result (Geron 2017). Meanwhile, shallow nets and fewer nodes demand less computational cost. During the validation, the number of layers is increased until overfitting occurs. Random number generation within a range is used to set the number of nodes per layer. The value of the scoring function was calculated at the final epoch during each training of the model with various layers and nodes. These hyperparameters were chosen after comparing the value of the scoring function.

Results

The result of scenarios with different initial states was presented to test the performance of the aforementioned method. The range of the initial states were $m_{ga} \epsilon [8000, 9000](kg), m_{gt} \epsilon [2500,4000](kg), m_{lt} \epsilon [15000, 20000](kg)$. These states reach their steady states within 0.2 hr. After comparing the shape of the trajectory with different sampling time, 5 samples were collected in a time span [0, 0.05] with the same interval. Another 5 samples and 20 samples were collected in time span [0.05, 0.2] and [0.2, 2] respectively. There are 200 samples collected. 120 samples were used for training. 40 samples were for evaluation and the rest of 40 samples were for testing.

The aforementioned method was tested using the dataset from the oil well simulator. Firstly, the designed PINN model was trained and a black box model was used to compare their training result. Then, the performance of the PINN model is shown by presenting training loss and validation loss during training. Validation results for training set size, learning rate and the number of layers and nodes were shown. The training results of three random training samples were chosen and compared with the training data. Finally, test results were presented and analyzed. Two random test samples and an additional sample which is out of the training range were utilized to evaluate the model.

Comparation between PINN and a black box model. The designed PINN model and a pure neural network as a black box model were trained with the same data set. The pure neural network has the same structure as the MLP used in the PINN model. After 200 epochs of training, a random training result was presented to compare the training performance between a pure neural network and a PINN in Fig. 4. To distinguish the training result and the training data, the training data are plotted as line charts and training data are plotted as scatters. The orange and green scatter plot shows the output of the PINN model and the black box model respectively.

Both PINN and the pure neural network can learn the data for steady states and smooth transients, though PINN shows a notable advantage in learning from training data in terms of learning speed and accuracy. The training loss of PINN decreased faster than the one of the pure neural network. PINN shows better performance on learning details, such as for the transient part of the mass of the gas in the tubing at time 0.1 hr to 0.3 hr shown in the middle plot in Fig. 4. The data is zoomed in and presented as blue scatters to show the detail of the transient part. The PINN learnt the shape of two continuous oscillations, while the pure neural network simplified the oscillation by fitting with a smooth curve. It is worthwhile to note that increasing the number of layers and nodes (the layer size was increased from 5 to 9 and the node size was increased from [5, 30] to [10, 70]) in the black box model does not significantly contribute to the fitting result. This problem is addressed in the

previous literature (Turan et al. 2021). Multiple shooting is a potential solution to improve performance.

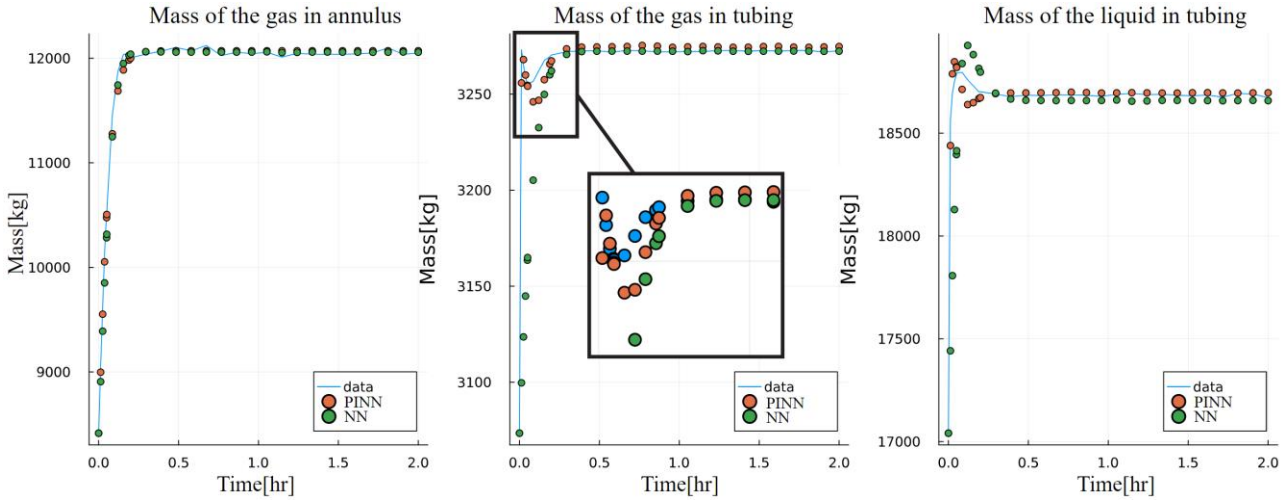

Figure 4. Comparation of Training Results of a PINN and a Pure Neural Network

Training loss and validation loss. The following results will focus on the performance of the PINN model. Training data corresponding to 120 different initial states was used for training. The training began with ADAM optimizer until 50 epochs and switched to BFGS for the rest of the training. During the parameters of the PINN model updated, the value of the loss function was recorded in each epoch. Training data and the training result of PINN were used in the loss function for calculating the training loss. Meanwhile, validation data and the prediction of PINN were utilized for validation loss to check if the model is overtrained and overfitted. The training loss and validation loss during 200 epochs are shown in Fig. 5.

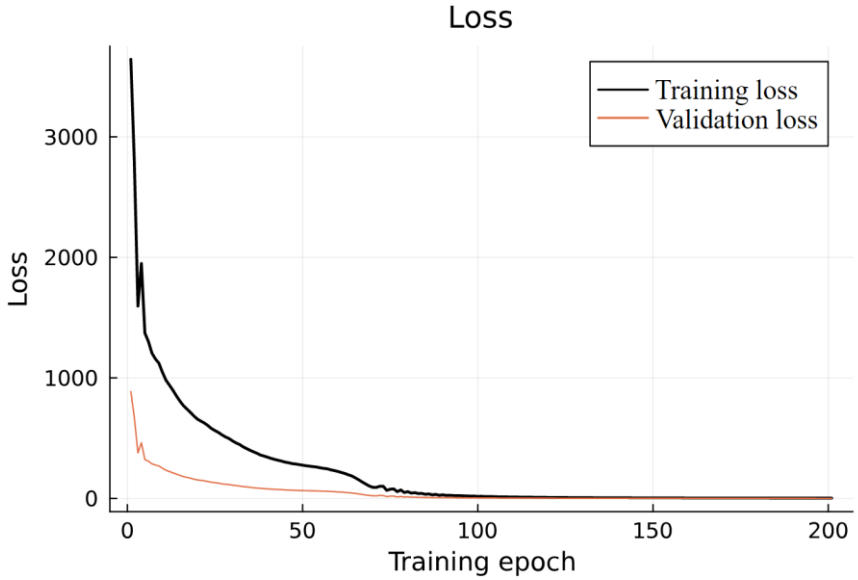

Figure 5. Training Loss and Validation Loss During Training Process

It took around 3 hours to train with 200 epochs at a workstation with Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz, 64-bit operating system, x64-based processor and 32.0 GB installed RAM. Both training loss and validation loss reached plateaus around 80 epochs. The training achieved the performance with a training loss less than 1 after about 80 epochs and continued decreased after that. Both losses reduced fast during using the ADAM optimizer and slowed down after changed to BFGS. The training loss is bigger than the validation loss because the size of the training data is three times than the size of the validation data. The training loss decreases from 3644.0083 to

2.4291277 within 200 epochs and the validation loss generally decreased from 887.6681 to 0.52164114.

Validation. Figure 6 illustrates the change in scoring function during 1000 training epochs using training data and validation data. The black and orange lines show the score of training and validation respectively. NMSE for both starts from -120 and increases to -0.27 and -0.29 at 1000 epochs. To show more details of these plots, the trajectories are zoomed in and are only presented in the small window within the range, [-0.5, 0], of the y axis. These trajectories increased at a similar pace before approximately 350 epochs. The score of training increased faster than the one for validation. The divergence of the two trajectories is likely because of the overtraining of the model. Therefore, the training set size is chosen as 350.

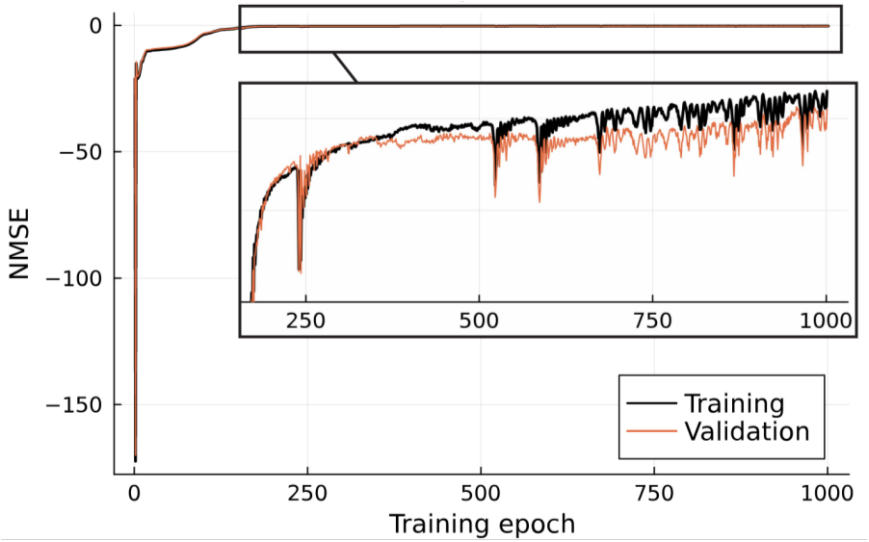

Figure 6. Learning Curves for Choosing Training Set Size

With 350 epochs, the learning rate between 0.0001 and 0.05 was validated and presented in Fig. 7. The scoring function of validation data was calculated and recorded during training. The large learning rate is related to the large optimal step and vice versa. The scoring function of the model with the learning rate 0.0001 increases more slowly than the scoring functions corresponding to other learning rates, namely it takes more than 350 epochs to converge to the optimum. Learning rate 0.05 to 0.005 provide the best NMSE among all validation results. However, their scoring functions increase rapidly at the beginning of the first 100 epochs, namely the step is too big and the optimizer hardly precisely searches around the optimum. The learning rate 0.001 was selected.

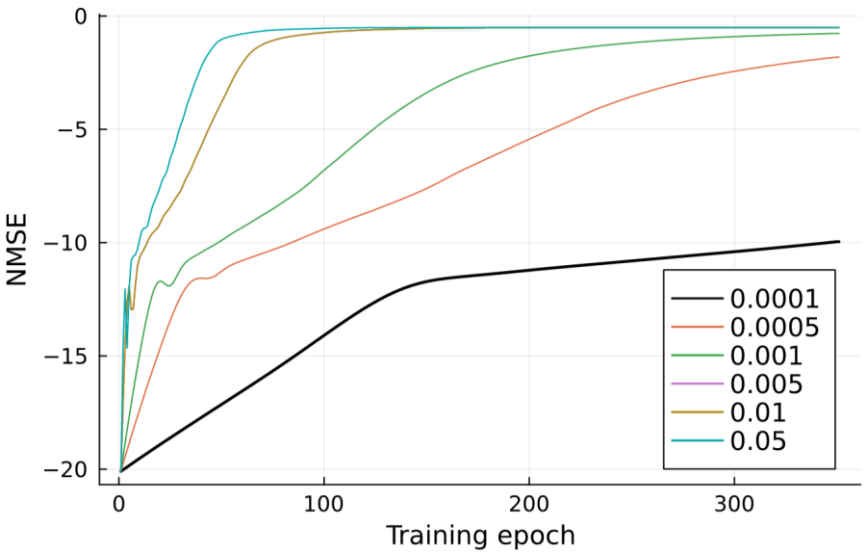

Figure 7. Learning Curves for Choosing Learning Rate

The number of layers was set from 3 and increased to 7 when the model started to overfit the training dataset within 350 epochs. With the same number of layers, 5 models were trained after randomly setting node size per layer. The range of the node size was set within [10, 70]. NMSE was calculated and recorded to present the performance of each model. The model with the structure shown in Table 1 has the biggest NMSE. The model with corresponding hyperparameters and trained parameters are saved for testing. Incorporating the aforementioned parameter tuning, the PINN model has three input states as the input layer and three vectors as the output layer. The model contains 6 layers including 4 hidden layers, an input layer and an output layer. ReLU was used as activities functions for each hidden layer.

Table 1: The Number of The Hidden Layers and the Number of the Nodes in Each Hidden Layer

Hidden layers	Nodes in layer 2	Nodes in layer 3	Nodes in layer 4	Nodes in layer 5
4	16	23	36	16

Training result. Three random sets of training results were randomly chosen and shown in Fig. 8.

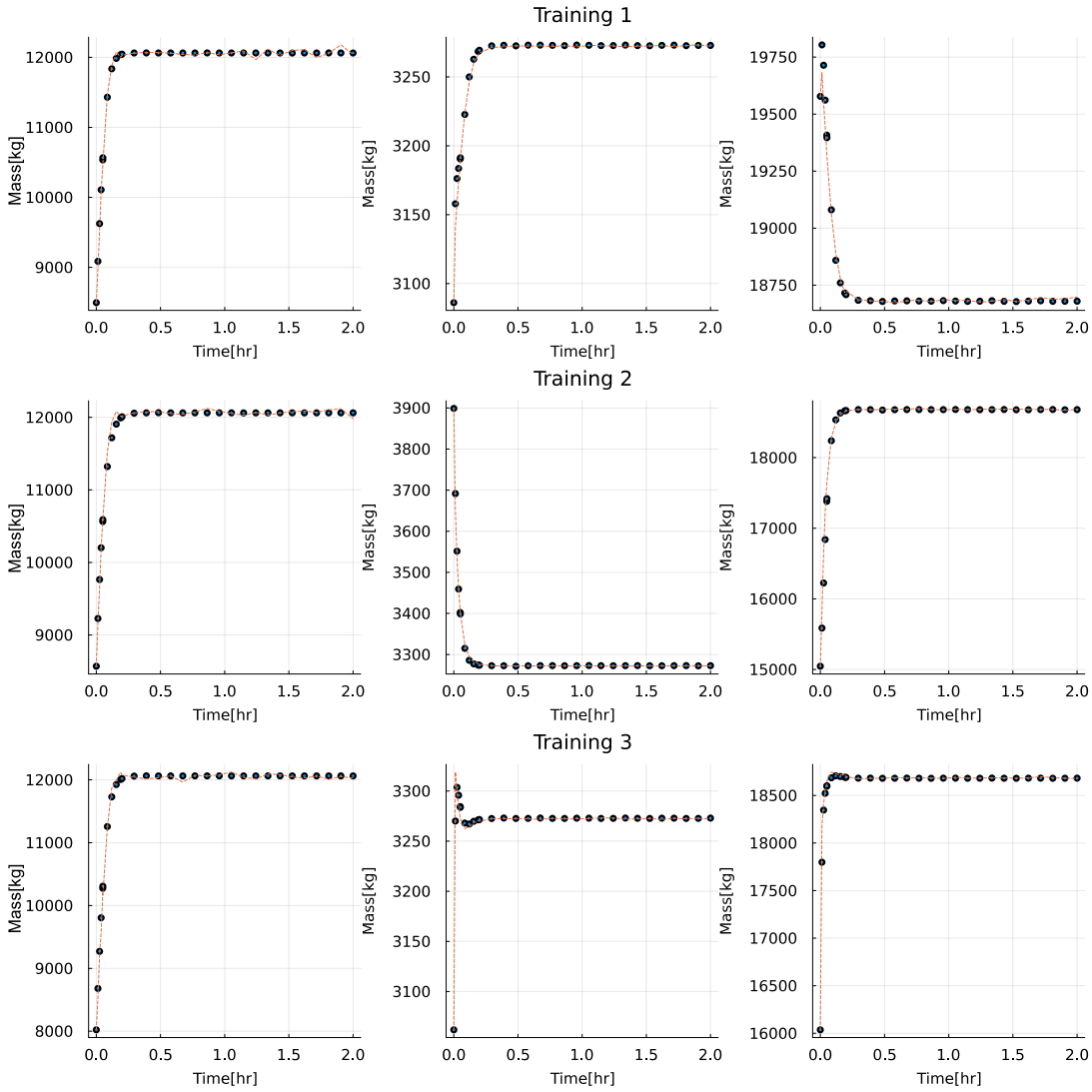

Figure8. Training Result from Three Training Data Set. The Scatter Plots Show Outputs of the PINN Model and the Dash Lines are the Training Data.

The left column plots illustrate the training result of the mass of gas in the annulus. The middle and right column plots show the training result of the mass of gas in the annulus and the mass of liquid in the tubing above injection correspondingly. The training data were generated from the simulator

with different initial states. According to the prior knowledge, the ranges of these initial states are [8000, 9000](kg) for the mass of gas in the annulus, [2500, 4000] (kg) for the mass of gas in the tubing above the injection point and [15000, 20000] (kg) for the mass of liquid in the tubing above injection.

The results demonstrate that the PINN model is able to learn the oil well dynamics using dataset with uncertainty. There are small distinct biases in the transient parts between the learning result and the training data. These biases decreased in the steady-state part.

Test result. Figure 9 provides some test results of the trained PINN model. Two prediction results, prediction 1 and 2, were tested using two test samples. The initial states of these test data are within the training range. Meanwhile, a test data, data 3, was tested. The initial states of data 3 are [10000, 2006, 16000], which is outside of the training range. The data set and prediction result corresponding to the same initial states are presented with the same color. Grey shadows present the training dataset ranges. The transients of each prediction were zoomed in to clearly show the comparison between test dataset and the prediction.

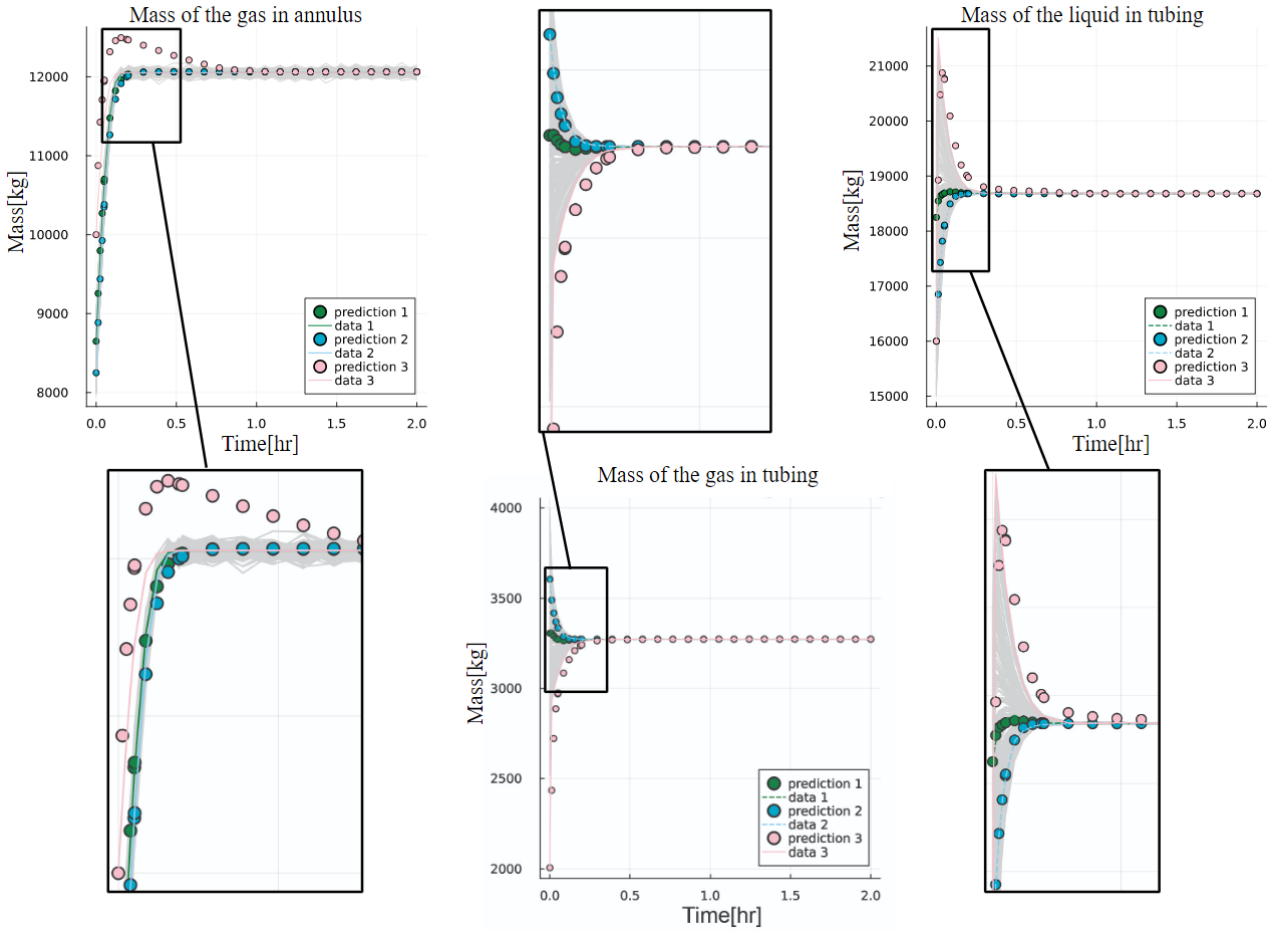

Figure 9. Test Result of Trained Neural Network

The PINN model is able to make accurate predictions within the training ranges. The prediction result maintains a relatively negligible bias. In contrast, the prediction for the initial states outside of training ranges, as expected, cannot provide reliable information, especially the prediction of the transient part.

Discussion

A physics model was used in this work for helping train the neural network model. However, the design of the physics model is subject to certain limitations due to the connection between the physics

model and the neural network in PINN. The parameters of the neural network in PINN do not have physical meanings. During training, the neural network might generate some outputs which are outside of the prior range. When the physics model is differentiated using these outputs, errors might occur. Besides, the output of the neural network might trigger negative values in a square root and some other problems. Therefore, some part of the physical model is not possible to be included in the reduced model. One of the solutions to include all parts of the physics model can be integrating a condition module, which rejects the proposed parameters of the neural network when these parameters lead to a computational crash. However, the computational time will increase due to the rejection. A further study could improve the solution and assess the tradeoff between the prediction accuracy and the computational cost.

Compared with some traditional neural networks, neural ODE shows advantages in time-related training problems using an ODE solver. The bias between the prediction and the real result is within an acceptable range with the NMSE increasing to -1.4662051. Data preprocessing was employed to avoid the uneven bias of the transient part and the steady state part for both training results and test results.

In the validation process, models with different hyperparameters were trained and compared. This manual method roughly provides a solution for hyperparameter selection. The model with better performance can be found after trying more combinations of the hyperparameters. In the validation part, this study was limited by the absence of a broad range of hyperparameters due to practical constraints. Grid search using cross-validation can be an alternative for automatically seeking optimum hyperparameters if the programming environment allows so.

In the case where the initial states are unknown, the prediction of the PINN model diverges dramatically. To solve this problem, the physics model can be used to estimate the initial states as the first step. Filtering technique and Markov chain Monte Carlo can be used for the estimation with considering uncertainty.

MLP was used for the neural network part of the PINN model. The differential equations in the physics model and an ODE solver are vital for building the connection inside the time series data. Other neural network structures, such as convolutional neural networks, can be tried to substitute MLP for more complex time series prediction. Further research could also be conducted to compare the proposed PINN model with other time series prediction approaches, such as the LSTM model using recurrent neural networks.

Conclusion

This research study set out to propose a physics-informed data-driven method for modelling a gas lifting oil well. The method can be generalized for systems which partly lack physics information. Compared with a black box model, the PINN model is observed with faster training time with 350 epochs and better performance with less than 0.522 of training loss. The results of this study present the training, validation, and test result, which show that the trained PINN model is able to predict the outputs of states with initial states which is within the training range for both steady-state and transient. The findings of this study provide insights for combining physics models and data. The scope of this study was limited in terms of the prediction of the states with dynamic input. The potential application of the PINN model is an auxiliary model based on data and prior physical knowledge of the oil production process, which might help provide information for model-based control.

Acknowledgment

This work was supported by The Research Council of Norway and Equinor ASA through Research Council project “308817 - Digital wells for optimal production and drainage” (DigiWell).

References

Alshaher, H., 2021. Studying the effects of feature scaling in machine learning (Doctoral dissertation, North Carolina Agricultural and Technical State University).

Al-Qutami, T.A., Ibrahim, R., Ismail, I., Ishak, M.A., 2017, Development of soft sensor to estimate multiphase flow rates using neural networks and early stopping. *International Journal on Smart Sensing and Intelligent Systems* 10, 199.

AL-Qutami, T.A., Ibrahim, R., Ismail, I., Ishak, M.A., 2018, Virtual multiphase flow metering using diverse neural network ensemble and adaptive simulated annealing. *Expert Systems with Applications* 93, 72–85.

Andrianov, N., 2018, A machine learning approach for virtual flow metering and forecasting. IFAC-PapersOnLine 51, 191–196.

Ban, Z., Ghaderi, A., Janatian, N., Pfeiffer, C., 2022. Parameter Estimation for a Gas Lifting Oil Well Model Using Bayes' Rule and the MetropolisHastings Algorithm. *Modeling Identification And Control*, 43(2), pp.39-53.

Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B., 2017, Julia: A fresh approach to numerical computing. *SIAM review* 59, 65–98.

Bikmukhametov, T., J¨aschke, J., 2020, First principles and machine learning virtual flow metering: a literature review. *Journal of Petroleum Science and Engineering* 184, 106487.

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

Franklin, T.S., Souza, L.S., Fontes, R.M., Martins, M.A., 2022, A physics-informed neural networks (pinn) oriented approach to flow metering in oil wells: an esp lifted oil well system as a case study. *Digital Chemical Engineering* 5, 100056.

Geron, A., 2017, Hands-On Machine Learning with Scikit-Learn & Tensorflow O'Reilly Media, Inc. *O'Reilly Media*, Inc, 1005, p.564.

Glorot, X. and Bengio, Y., 2010, March. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference Proceedings.

Gross, M.R., Hyman, J.D., Srinivasan, S., O'Malley, D., Karra, S., Mudunuru, M.K., Sweeney, M., Frash, L., Carey, B., Guthrie, B., et al., 2021, A physics-informed machine learning workflow to forecast production in a fractured marcellus shale reservoir, in: SPE/AAPG/SEG Unconventional Resources Technology Conference, OnePetro.

Innes, M., Saba, E., Fischer, K., Gandhi, D., Rudilosso, M.C., Joy, N.M., Karmali, T., Pal, A., Shah, V., 2018, Fashionable modelling with flux. arXiv preprint arXiv:1811.01457.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S. and Yang, L., 2021. Physics-informed machine learning. *Nature Reviews Physics*, 3(6), pp.422-440.

Kim, H., Papamakarios, G. and Mnih, A., 2021, July. The lipschitz constant of self-attention. In International Conference on Machine Learning (pp. 5562-5571). PMLR.

Kingma, D.P., Ba, J., 2014, Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Knopf, F.C., 2011. *Modeling, analysis and optimization of process and energy systems*. John Wiley & Sons.

Liu, D.C., Nocedal, J., 1989, On the limited memory bfgs method for large scale optimization. *Mathematical programming* 45, 503–528.

Noriega, L., 2005. Multilayer perceptron tutorial. School of Computing. Staffordshire University, 4, p.5.

Rackauckas, C., 2022, Parallel computing and scientific machine learning (sciml): Methods and applications. < https://github.com/SciML/SciMLBook, doi:10.5281/zenodo.6917234.>

Rackauckas, C., Innes, M., Ma, Y., Bettencourt, J., White, L., Dixit, V., 2019, Diffeqflux. jl-a julia library for neural differential equations. arXiv preprint arXiv:1902.02376 .

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., Edelman, A., 2020, Universal differential equations for scientific machine learning. arXiv preprint arXiv:2001.04385 .

Rackauckas, C., Nie, Q., 2017, Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of open research software* 5.

Sharma, R., Fjalestad, K. and Glemmestad, B., 2011. Modeling and control of gas lifted oil field with five oil wells. In 52nd International Conference of Scandinavian Simulation Society, SIMS (pp. 29-30).

Sun, J., Ma, X., Kazi, M., 2018, Comparison of decline curve analysis dca with recursive neural networks rnn for production forecast of multiple wells, in: SPE Western Regional Meeting, OnePetro.

Turan, E.M. and Jäschke, J., 2021. Multiple shooting for training neural differential equations on time series. *IEEE Control Systems Letters*, 6, pp.1897-1902.

Biography

Zhe Ban received the B.S. degree in automation in 2018 and the M.S. degree in advanced control engineering in 2019. She is currently pursuing the PhD degree in information technology at University of South-Eastern Norway, Porsgrunn, Telemark, Norway.

From 2019 to 2020, she was a teaching assistant at The University of Manchester. Her research interest includes data-driven physics-informed modelling, the application of Bayesian analysis to nonlinear dynamic systems and data reconciliation.

Carlos F. Pfeiffer. Received a PhD in Chemical Engineering with specialty in Process Control from the University of Texas in Austin, in 1999. He worked at the Computer Science Department at ITESM, Monterrey, Mexico, from 2001 to 2010. From 2011 he has been working at the Electrical Engineering and Cybernetics Department at USN, Norway, where he is currently a professor.

His present research interest include Modeling of Dynamics Systems, Advanced Process Control and Optimization, and Energy Systems.