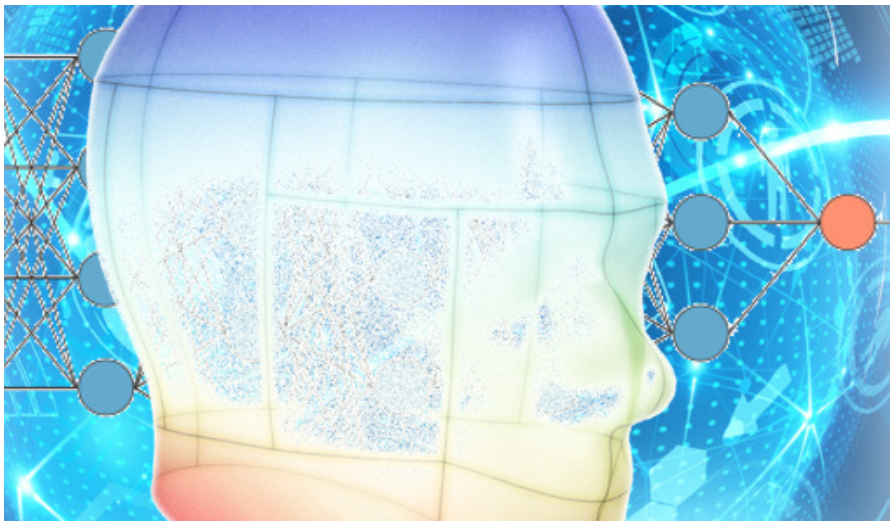
Physics-informed Machine Learning for Control - DNN in a Dynamic Feedback Loop



This Case Study describes an approach to combining physical principles with Machine Learning (ML) for modeling and control of complex systems. Our approach was developed as part of a DARPA-funded research project. It was applied to oil reservoir management. While this Case Study provides an overview, technical details may be found in a separate publication [1].

BACKGROUND

A physical process may be considered as a change in the state of a system with time – driven by physical phenomena that are internal or external to the system. Physicists and engineers attempt to model these processes employing conservation laws supplemented by phenomenological relationships. The resulting models are generally expressed as time-dependent partial differential equations that are often coupled and non-linear. The solution to these equations, with appropriate initial and boundary conditions applied, describe the physical process. This physics-based paradigm has continued to be the primary approach for modeling complex physical systems ranging from chemical micro reactors to planetary climate.

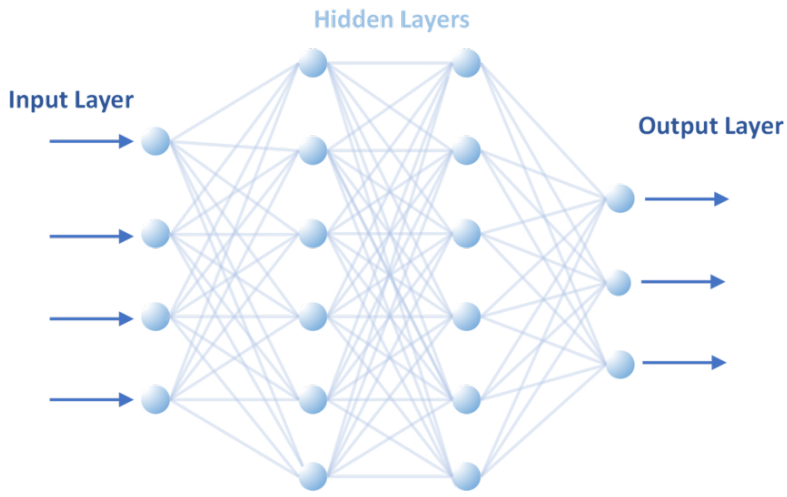
However, simulations with such first-principles, high-fidelity models tend to be computationally intensive, and do not run in real-time. Hence, these models cannot be used in many real-world applications such

as process control or tracking propagation of forest fires.

Over the past few decades, techniques have been developed to generate low-order models from high-fidelity models that enable simulations to be run in real-time or faster. These dynamic fast models attempt to accurately model the key physical state variables of interest while sacrificing accuracy on other state variables of less importance. Such fast low-order models may be developed mathematically e.g., proper orthogonal decomposition (POD), or developed using simplified physics that also judiciously reduce the number of states through aggregation. However, these fast models often cannot account for disturbances and process drifts. Additionally, the effect of unmodeled physics (left unmodeled intentionally or unintentionally) may be significant, including model parameters that may change often with time.

With the proliferation of a wide range of sensors and the consequent availability of vast amount of data, this physical modeling paradigm is being challenged by the data-driven machine learning (ML) paradigm. For example, deep neural networks (DNN) are used in lieu of physics-based models because they can use these data to make comparably accurate predictions.

In its simplest definition, DNN's are computational models consisting of a large number of processing layers designed to accurately map the input data to the output data as shown in the figure below. Each layer consists of several neurons, with each neuron connected to other neurons in the preceding and succeeding layers. The DNNs are trained to incorporate complex underlying relationships between the inputs and outputs in a manner that mimics operation of the human brain. In mathematical terms, training a neural network for a specific problem involves solving an optimization problem where the objective (cost function) is minimized to find the optimal set of parameters, which happen to be the weights to the inputs to the neurons in each layer of the network.



The choice of the number of hidden layers and the number of artificial neurons in each layer are design parameters that are problem dependent and are presently determined mostly by trial and error. DNNs employ multiple hidden layers (typically more than three) to capture all the relationships between the input and output layers. The output of an artificial neuron, **y**, in each layer is the value of a transfer function, called activation function, whose argument is the sum of the neuron's **n** inputs where each input, $\mathbf{x_i}$, has a weight $\mathbf{a_i}$, and an additional bias term, $\mathbf{b_i}$, as shown below:

$$y = f(z);\ z = \sum_i^n a_i x_i + b\,.$$

Although DNNs have shown promising results in a wide range of applications, they require large amounts of data for training and are prone to overfitting, making them ineffective in unfamiliar or challenging situations outside the training dataset. An ML model will likely fail if it is applied outside the training space, especially if the system has

changed with time. By training space, we are referring to the range of values of inputs and parameters of the system that was spanned when the data was acquired for training the ML model. The reason for failure is that the ML model's non-linear approximations of the actual physics may be very inaccurate on extrapolation. Even within the training space, a ML model may not produce accurate predictions when the training dataset is not sufficiently rich, i.e., the dataset despite being large in size has not adequately resolved the parameter or input space.

An alternative approach is to use both physics-based models and data-driven ML techniques in a complementary way that leverages the strengths of both approaches. A successful form of this approach, often referred to as physics-informed (or physics-infused) machine learning (PIML), would produce accurate simulation results by accommodating some unmodeled physics (e.g., system property variations in time), while showing robustness outside the training space.

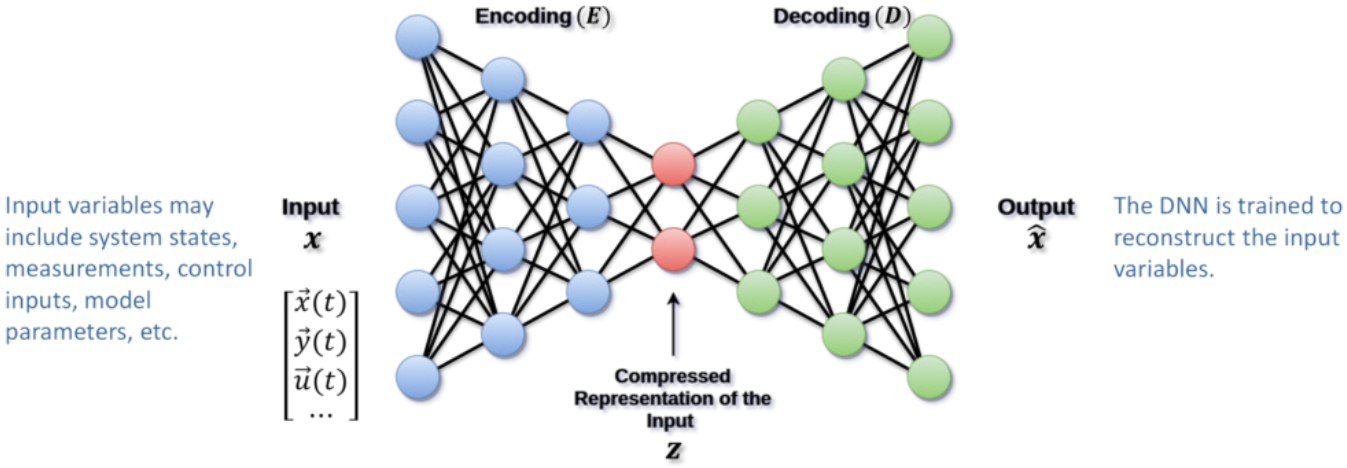
Many PIML techniques have been developed that incorporate physics-based constraints and empirical prior knowledge (e.g., spatial symmetry), or a mix of both in designing and training the DNN [2][4]. As described below, SC's approach is a variation on this technique that uses simulation results generated from fast physics-based models to train the DNN. We incorporate physics-based constraints into the DNN design, and uses the DNN to control the system in a dynamic feedback loop.

PIML IMPLEMENTED WITH DNN IN A DYNAMIC FEEDBACK LOOP FOR CONTROL

In our approach, prior knowledge in the form of physics-based models has been incorporated in the ML model both in the construction of the DNN as well as in the latter's training.

We have used an autoencoder architecture for the DNN as shown in the figure below. The autoencoder employs a narrowing of the intermediate (hidden) layers between the input and output layers of the DNN. This bottleneck in the network forces a compression of the information in the original input vector (using the encoder part of the network), **x**, followed by a subsequent reconstruction (using the decoder part of the network) to obtain the DNN output, $\hat{\mathbf{x}}$ This compressed representation of the inputs is denoted by **Z**, and may be considered as a latent variable that is associated with **x** and can reduce the dimensionality of the input data [5].

We note that the input, **x**, of the autoencoder DNN does not necessarily correspond to the inputs of the model or to the physical system. It may include any combination of measurements, model inputs (e.g., voltage or heat flux),



model outputs (e.g., velocity or temperature), and model parameters (e.g., mass, heat capacity), even though they are related to each other. In ML, descriptors such as velocity, mass, etc., are referred to input data labels and the data is characterized as a labeled dataset.

An ML model is said to undergo supervised learning when the model is trained using a labelled dataset to determine the mapping function to map the inputs to the outputs. In contrast, unsupervised learning occurs when the model itself finds the hidden patterns from the supplied dataset using techniques such as clustering, rule-based association, and dimensionality reduction. Since autoencoders do not need explicit labels for training they may be trained using an unsupervised learning technique.

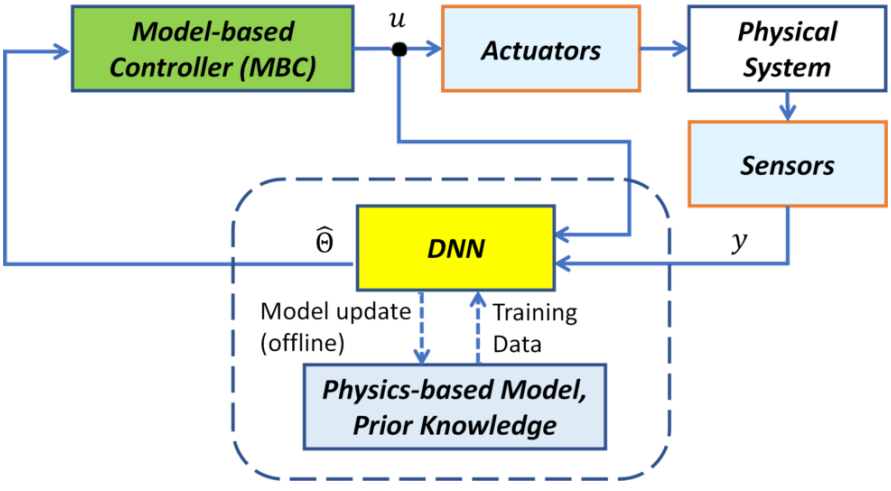
In training the DNN, we determine the values of these weights and biases such that a specific loss function is minimized. The process involves constructing the "loss function", which is the objective or cost function that is minimized in the optimization problem. A typical loss function for autoencoder consists of a reconstruction loss that encourages the model to be sensitive to the inputs and a regularizer that discourages memorization or overfitting.

The loss function for our application was defined as shown in the equation below. Here, the first two terms are the reconstruction loss of the input data and its time derivatives respectively. The third term, the physics-based regularizer, has its origin in the physics-based model and constrains the space of admissible solutions for the weights by imposing symmetry, invariance, or conservation principles originating from the physical laws that govern the system as described below. These terms may be scaled with the parameters λ_1 and λ_2 to control the trade-off between the three goals.

$$L = \|x - \hat{x}\|^2 + \lambda_1 \|\dot{x} - \dot{\hat{x}}\|^2 + \lambda_2 \|\dot{z} - \phi(z)\hat{\theta}\|^2$$

Reconstruction loss in x | Reconstruction loss in $\dot{x}$ | Constraint from reduced-order physics model, $[\dot{z}] = \phi(z)\Theta$

The DNN design reflects our intent to use the PIML model for model-based control (MBC) as shown in the figure below.



The controller uses the same fast, physics-based model that is used in the DNN. The trained DNN used in such a dynamic feedback loop (DFL) employing effective control algorithms can achieve robustness to system variations and uncertainty, reduction of response to noise and disturbances, and optimality of the dynamic response.

The accuracy of such models for dynamic systems may often degrade over time because of physical (and chemical) changes that are reflected in the changing values of the model parameters. It would be desirable to be able to dynamically update these model parameters in the MBC. As we described earlier, our autoencoder architecture has new estimates of the parameters as part of the

DNN output. These updated parameter estimates, $\hat{\Theta}$, are used by the controller.

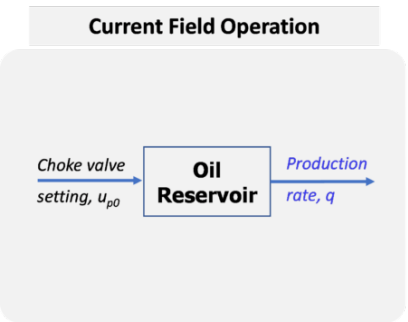
When the estimated parameters fall outside a user-specified range, the DNN would need to be retrained offline. Both model simulation results and measurements are used for the retaining. Use of measurements in training data attempts to compensate for model inaccuracy both as a result of approximations in the lower-order fast model as well as for some unmodeled physics. In the next section, we describe how this approach was used in oil reservoir management.

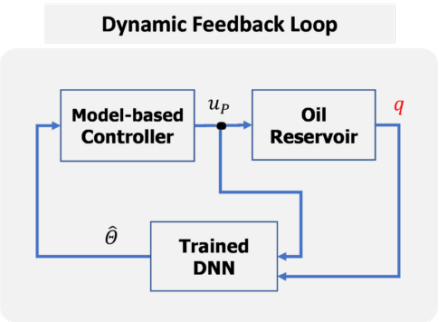
APPLICATION TO OIL RESERVOIR MANAGEMENT

This PIML approach was developed in the DARPA program, Physics of Artificial Intelligence (PAI), in which SC partnered with NeoTek Energy. A predictive analytical tool for oil reservoir management was developed based on the PIML approach. The tool was used in the dynamic feedback loop (DFL) described earlier to optimize operational parameters such as choke valve settings and injection rates. The optimization goal was to reach specific target metrics of production, e.g., maximizing oil production while minimizing the associated (but unwanted) by-products such as natural gas. The effectiveness of the DFL was demonstrated in the field tests performed in the Yates field.

Oil and gas reservoirs may be considered as complex dynamical physical systems interacting with their external environment through a set of wells that serve as inputs and outputs to the system, i.e., through the wellbores that connect to the oil reservoir. The system is non-linear, and its dynamic behavior is time varying. Examples of inputs are fluid injections (flow rates of injected species) and wellhead choke valve settings (pressure and/or flow rate control), while outputs are production rates of oil, water, and gas.

A high-fidelity physical model of this complex system involves a large number of time-varying parameters, whose values





have significant uncertainties. Consequently, predictions of such a model may have limited accuracy making it difficult to use these predictions for optimal reservoir management. Hence, for optimizing production, the oil industry has generally relied on heuristics, expert knowledge, and experience.

As described in the previous section, the DFL control system consists of an autoencoder-type DNN that extracts physics-based parameters from the real-time measurements and updates the relevant parameters that are fed to the model-based controller. As shown in the figure below, in existing field operation, a set of control inputs (uP0) is selected, based on expertise and best estimates from existing data. Typically, these inputs are left unchanged for a specified period before repeating the process. However, in our approach, the optimal set of controls (uP) are continuously adjusted based on the DNN's prediction of key parameters ($\hat{\Theta}$) from the production data (q) as shown in the figure below.

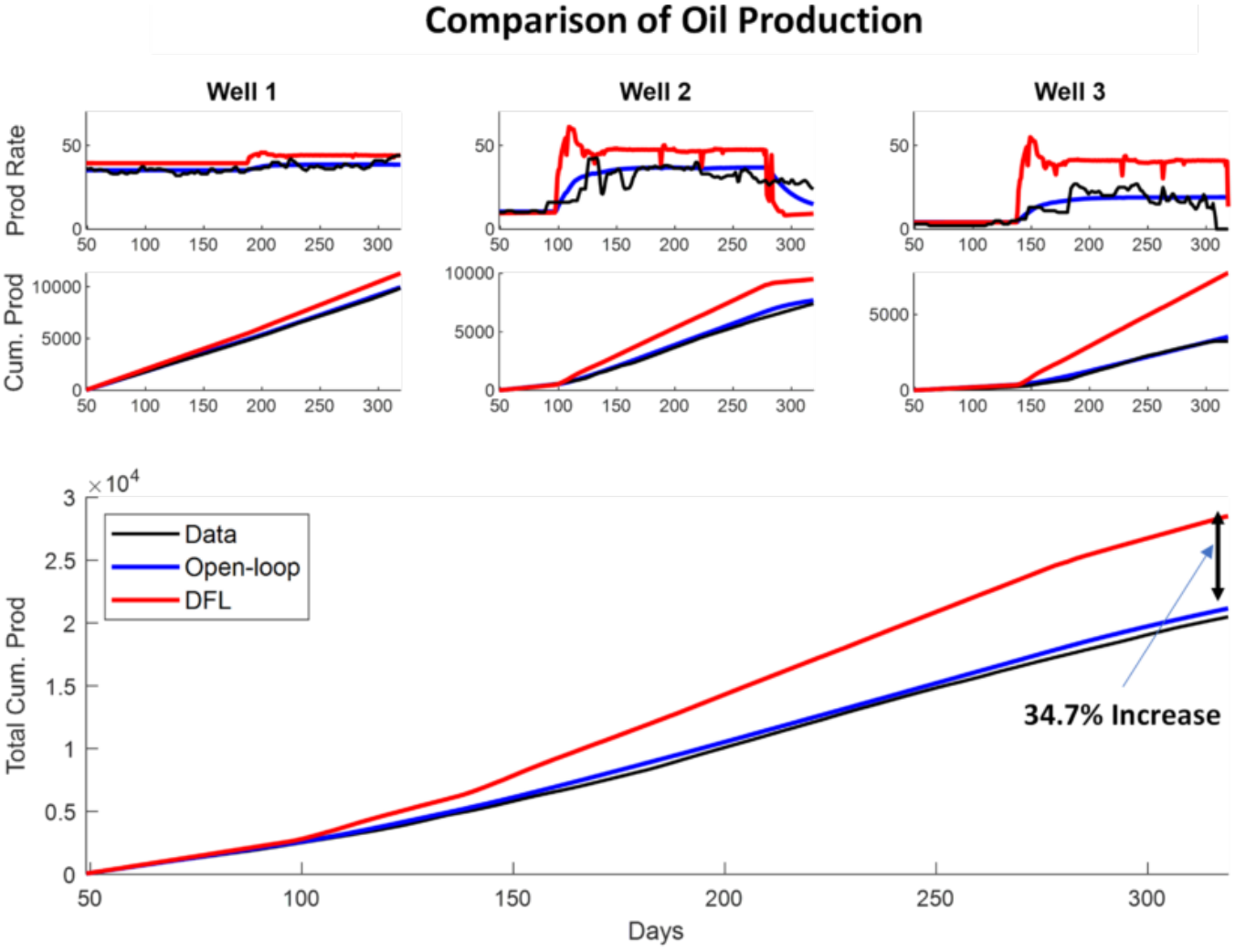
We developed a fast, low-order, physics-based model of a reservoir with multiple wells that combined the capacitance-resistance model from the literature [6] that incorporates inter-well connectivity, and our original coning model to simulate flow at individual wells [1]. This multi-well reservoir model was calibrated and validated with the historical production data of the three pilot wells in the Yates field. Results from model simulation

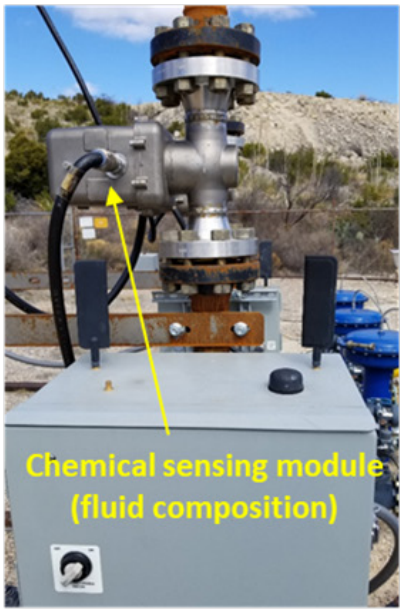
were then used to train the DNNs to estimate the changes in the key parameters of oil reservoir using data from NeoTek's real-time sensors installed on each monitored well (GORA Analyzer).

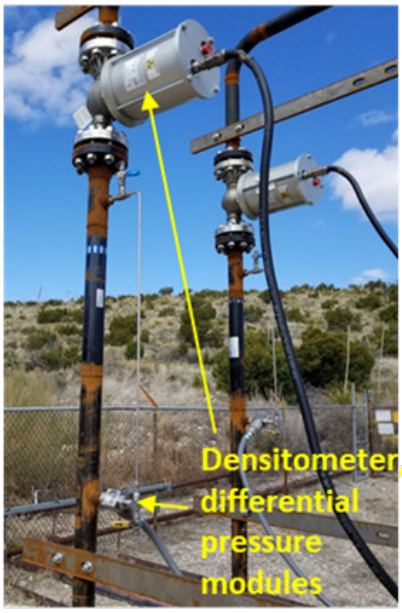
The controller was developed to maximize the cumulative amount of oil produced over time, penalized by the total amount of gas and/or water that also came out of the wells over the same time period. The controller generates a set of actuator inputs (choke valve settings) for a specified time horizon based on the parameters estimated by the trained DNN, in a manner similar to model predictive control.

We compared the oil production between the open-loop control representing the current field operation and the DFL scheme in simulation using the three-well reservoir model. The results are shown in the figure below where the blue lines represent the open-loop system, and the red lines represent the system with the DFL. The simulation results predict that the use of DFL would result in an increase in the cumulative annual oil production of 34.7%.

The DFL was field-tested in the Yate's field. For the field test, three GORA sensor systems were deployed on the three pilot wells in the Yate's field. Action-reaction experiments ('bump tests') were designed and conducted to identify the physical model parameters for the pilot wells to identify the nominal model parameters for use in the DNN training.



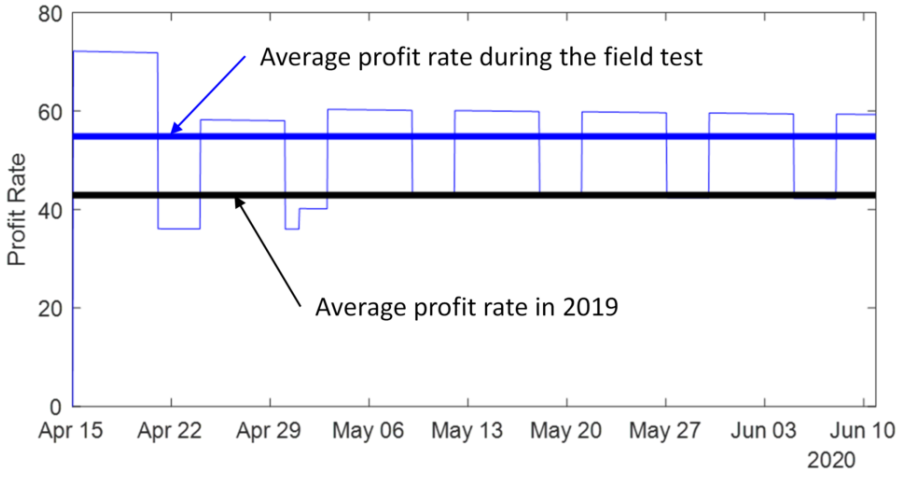




In the field test, the controller periodically generated a set of inputs (choke valve settings) based on the outputs of the DNNs, which in turn was supplied with real-time measurements from the GORA Analyzers. The time horizon was about a week. The operator in the field subsequently implemented these new settings. The average profit rate with the recommended choke settings was compared with the previous year's data as shown in the figure below. With the recommended settings applied, the average profit rate was increased by 27.9% over the same period of the previous year. The technical approach, including the formulation of the physics model and the DNN training, is explained in greater detail in paper that is available at this link [1].

BENEFITS

Physics-infused Machine Learning (PIML) attempts to embed physics and prior knowledge into machine learning that helps overcome the challenges of sparse data and facilitate the development of predictive models for complex processes that are causal and physically meaningful. Such an approach can be useful in many applications including model-based control, virtual sensing, design of next-generation equipment, troubleshooting, and process optimization. If you would like more information regarding use of PIML-based model development for potential application to your system, please contact us at 408-617-4525.



REFERENCES

[1] De Bruyker, R. L. Kosut, R. Valdez, S. Haymes, L. Schoeling, M. Petro, D. Weiner, A. Joseph, J. K. Lee, A. Emami-Naeini, J. L. Ebert, and S. Ghosal, "Improving Recovery in the Yates Field Using Dynamic Feedback Loop based on Physics-Informed Artificial Intelligence," Proceedings of SPE Improved Oil Recovery Conference, Society of Petroleum Engineers, August 2020.

[2] L. Brunton, J. L. Proctor, J. N. Kutz, "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems," Proc. Natl. Acad. Sci. U.S.A., 113, 3932–3937, 2016.

[3] Raissi, P. Perdikaris, G. E. Karniadakis, "Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems," arXiv:1801.01236, 2018.

[4] Mezic, "Spectral Properties of Dynamical Systems, Model Reduction and Decompositions," Nonlinear Dyn. 41, 309–325 (2005). Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, MIT Press, vol. 1, 2016.

[5] Sayarpour, E. Zuluaga, C. S. Kabir, L. W. Lake, "The use of capacitance–resistance models for rapid estimation of waterflood performance and optimization," J. Pet. Sci. Eng., 2009.