Deep-CRM: A New Deep Learning Approach For Capacitance Resistive Models

Abderrahmane Yewgat, Daniel Busby, Max Chevalier, Corentin J Lapeyre, Olivier Teste

ECMOR XVII
17TH EUROPEAN CONFERENCE ON THE MATHEMATICS OF OIL RECOVERY

EAGE

Deep-CRM: A New Deep Learning Approach For Capacitance Resistive Models

Abderrahmane YEWGAT
Total R&D
Toulouse informatics research institute (IRIT)
Université Toulouse Jean Jaurès
abderrahmane.yewgat@total.com

Daniel BUSBY
Total R&D
daniel.busby@total.com

Max CHEVALIER
Toulouse informatics research institute (IRIT)
Université Toulouse III - Paul Sabatier (IRIT)
Max.chevalier@irit.fr

Corentin LAPEYRE
Research Scientist
Coordinator of the Helios workgroup
CERFACS - COOP Team
lapeyre@cerfacs.fr

Olivier TESTE
Toulouse informatics research institute (IRIT)
Université Toulouse II – Jean Jaurès
Olivier.teste@irit.fr

1- Introduction

Classical reservoir simulators are built upon different underlying models: geological models integrating all the knowledge of the subsurface properties, fluid flow models integrating reservoir fluid physical properties, wells, and surface installations. The construction of such models however is very time and resources consuming. In the case of mature fields, where historical production data are available, data driven models can represent a suitable alternative or can be complementary to classical reservoir modelling as they require much less computation time and allocated resources.
Among such models are Capacitance Resistive Models (CRMs), based on set of coupled ordinary differential equations representing material balance. These aim to predict flow rate in a reservoir using only dynamic data of production rates, water injections and Bottom Hole Pressure (BHP). In addition, CRMs can explain the underlying connectivity between several injectors and producers that could be a valuable information for dynamic synthesis and for better understanding of fluid flows in the reservoir. Most of the current work on CRMs optimizes a nonlinear multivariate regression of the CRM's parameters. Such optimization needs a closed form solution of the CRM ODEs. which is only possible under conditions: constant or linear variation in injection or in BHP. Once we have optimized the CRM's parameters, we can use the closed form solution to perform forecasting.
The aim of this work is to propose a complete approach to optimize the CRM's parameters and forecast future production. This approach is not based on any assumptions on injections or on producers' BHP. To this end, we introduce a new approach based on a deep learning strategy: Physics-Informed Neural Networks (PINNs) for CRMs.
This paper is organized as follows: first we introduce the related work on CRMs. Second, we detail the theory of CRMs and PINNs. Our approach, called Deep-CRMs, is presented in the third section. We focus on the mathematical description of Deep-CRMs and show experiments in order to compare our approach to the nonlinear multivariate regression of the closed form solution. These experiments are based on two datasets: the first is a synthetic dataset generated using ECLIPSE® and SISMAGE®, and the second is a real field dataset provided by one of our affiliates.

2- Related Work

In this section we introduce the related work that we have identified and which we consider close to our objective which is optimizing CRMs' parameters and forecasting the production rate. CRMs are material balance models capable of optimizing waterflood injection and explaining the underlying connectivity between injectors and producers. In his PhD dissertation [3] and in Youssef et al. [2], Youssef gave a widespread introduction to CRMs from theorical backgrounds to real field applications. Wanderley de Holanda et al [1] presented a scientific literature review on CRMs. In Sayarpour et al [4], authors firstly solve the CRMs ODE analytically, in a closed form, using superposition in time and space, and then prefer the use of data driven methods like nonlinear multivariate regression to optimize the physical parameters.
Previous CRMs applications rely exclusively on the availability of a closed form solution of CRMs' ODE, by assuming that injections and productions respect some conditions. Our proposal aims to present a complete and fully based Deep Leaning approach capable of optimizing the CRMs' parameters and forecasting rate production, without considering any assumptions on the injections and BHP. To do this we detail in the next section the proposed approach based on physics-informed neural networks.

3- Our proposal: Deep-CRMs

Deep learning using Artificial Neural Networks (ANNs) presents a new paradigm for learning by mimicking the function of the human brain. Thanks to the universal approximation theorem [5], ANNs can approximate any continuous function, which makes them a powerful tool to model complex phenomenon with strong nonlinearities, for which classical methods fail.

Reservoir modelling deals with complex phenomena, resulting from different factors (geology, fluid flow…). Some of these phenomena satisfy given partial differential equations, making it crucial to incorporate theses PDEs in any modelling of the reservoir to enhance interpretability. Indeed, while ANNs are powerful function approximators that can achieve high accuracy, in the context of reservoir modelling this cannot come at the cost of interpretability. The physical constraints, in the form of PDEs, must be accounted for, not only to enhance prediction, but also to give ANNs more physical meaning than simple mathematical approximations. In this context a new approach called Physics Informed Neural Networks was recently introduced [6,7] using the power of ANNs to model and solve complex PDEs. Promising results were proven for several PDEs from various domains, e.g. Navier-Stokes [10], Darcy flow problem [9], and 1D & 2D Coupled Burgers' Equation [8]. In these papers it is shown that PINNs could provide better or similar quality results than classical solvers in these various application domains with generally less computational time.

Using ANNs as a physical phenomenon estimator, by incorporating prior physical knowledge is not a new idea. Psichogios et al.[16] investigated the use of ANNs to model a fedbatch bioreactor. Since then several works have used ANNs to address different physical phenomena. Raissi et al.[17] recently modelled fluid motions using Navier-Stokes equations and ANNs. In the prementioned works, ANNs are trained on two types of data: observation points, representing discrete observations of the physical phenomenon, and collocation points, representing points of the domain where PDE should be satisfied. Zabaras et al [9] discuss PINNs framework where ANNs are trained only on collocation points and starting from the initial solution.

This study focuses on applying PINNs to CRMs. Before introducing how PINNs and CRMs can be coupled, we detail separately the mathematics of each of these models.

3.1-The Mathematics of Capacitance Resistive Models

The name CRMs came from the similarity between the CRMs Ordinary Differential Equation (ODE) and the governing equations of electrical capacitor resistors models. CRMs consist in modelling the variation of the rate production over time $\boldsymbol{q(t)}$ for each producer, given the injections $\boldsymbol{I(t)}$ of all

injectors and the producer bottom hole pressure $\boldsymbol{p_{wf}(t)}$. CRMs are based on differential equations: the material balance equation (1) and the deliverability equation (2).

$$\boxed{C_t V_p \frac{d\bar{p}}{dt} = I(t) - q(t)} \quad (1)$$

$$\boxed{q(t) = J\left(\bar{p}(t) - p_{wf}(t)\right)} \quad (2)$$

where $\boldsymbol{C_t}$ is total compressibility, $\boldsymbol{V_p}$ is the control volume, $\boldsymbol{\bar{p}}$ is the volume averaged pressure, $\boldsymbol{I(t)}$ is the total injection at time $\boldsymbol{t}$, and $\boldsymbol{q(t)}$ is the rate at time $\boldsymbol{t}$.
$\boldsymbol{p_{wf}}$ is the bottom hole pressure, and $\boldsymbol{J}$ is the productivity index of the given producer. Replacing (2) in (1) yields the following CRMs ODE equation:

$$\boxed{\tau \frac{dq(t)}{dt} + q(t) = I(t) - \tau * J * \frac{dp_{wf}(t)}{dt}} \quad (3)$$

Where $\boldsymbol{\tau}$ is the time constant defined as:

$$\boxed{\tau = \frac{C_t V_p}{J}} \quad (4)$$

Let us consider $\boldsymbol{N}$ injectors and $\boldsymbol{M}$ producers, then (3) can be written as:

$$\boxed{\forall j \in [1..M], \tau_j \frac{dq_j(t)}{dt} + q_j(t) = \sum_{i=1}^{N} f_{ij} I_i(t) - \tau_j * J_j * \frac{dp_j(t)}{dt}} \quad (5)$$

From (5) $\boldsymbol{f_{ij}}$ represents the connectivity between the injector $\boldsymbol{i}$ and the producer $\boldsymbol{j}$. for each injector we have:

$$\boxed{\sum_{j=1}^{M} f_{ij} \leq 1} \quad (6)$$

then we can rewrite (5):

$$\boxed{\begin{array}{c} \forall j \in [1..M], \quad \tau_j \frac{dq_j(t)}{dt} + q_j(t) = \sum_{i=1}^{N} f_{ij} I_i(t) - \tau_j * J_j * \frac{dp_j(t)}{dt} \\ \forall i \in [1..N], \sum_{j=1}^{M} f_{ij} \leq 1 \\ \forall i \in [1..N], \forall j \in [1..M], f_{ij} \geq 0, \ \tau_j \geq 0, \ J_j \geq 0 \end{array}} \quad (7)$$

Example. Figure 1 represents a case with N = 4 injectors and M = 1 producer, where the edges are weighted by the amount of injection. $I_i(t)$ represents the total injection of injector i at time t. f_{ij} represents the connectivity between injector i and producer j.

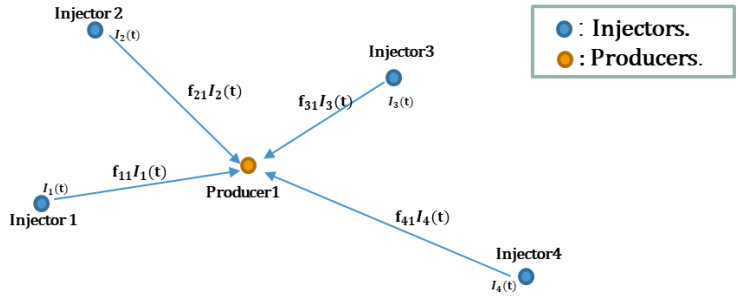


Figure 1 *Example of producers-injectors connectivity*

In [4] the authors show that in case of constant injection and linear variation of BHP, (5) can be solved analytically:

$$q_j(t_n) = q_j(t_0)e^{\left(-\frac{t_n-t_0}{\tau_j}\right)} + \sum_{k=1}^{n}\left\{e^{\left(-\frac{t_n-t_k}{\tau_j}\right)}\left(1 - e^{\frac{-\Delta t_k}{\tau_j}}\right)\left[\sum_{i=1}^{N}\left[f_{ij}I_i^{(k)}\right] - \tau_j * J_j\frac{\Delta p_{wf,j}^{(k)}}{\Delta t_k}\right]\right\} \quad (8)$$

where $q_j(t_n)$ is the rate of the producer j at time t_n. $I_i^{(k)}$ is the rate of the injector i for the k time interval. In the case of linear injection, and linear BHP we have the following solution:

$$\begin{aligned} q_j(t_n) = {} & q_j(t_0)e^{\left(-\frac{t_n-t_0}{\tau_j}\right)} + \sum_{i=1}^{N}\left[f_{ij}\left(I_i(t_n) - i_i(t_0)e^{\left(-\frac{t_n-t_0}{\tau_j}\right)}\right)\right] \\ & + \sum_{k=1}^{n}\left\{\tau_j * e^{\left(-\frac{t_n-t_k}{\tau_j}\right)}\left(1 - e^{\frac{-\Delta t_k}{\tau_j}}\right)\left[\sum_{i=1}^{N_i}\left[f_{ij}\frac{\Delta i_i^k}{\Delta t_k}\right] + J_j\frac{\Delta p_{wf,j}^{(k)}}{\Delta t_k}\right]\right\} \end{aligned} \quad (9)$$

Knowing the closed form solution, and having available observations, the parameters of the CRMs can be obtained using a nonlinear multivariate regression:

$$(f_{ij}, \tau_j, J_j)_{j=1\ldots M,\ i=1\ldots N} = argmin\left(\sum_{j=1}^{M}\sum_{n=1}^{T}\left(q_j(t_n) - \widehat{q_j(t_n)}\right)^2\right) \quad (10)$$

where $\widehat{q_j(t_n)}$ is the observed rate of the j producer at time t_n.

3.2-The Mathematics of Physics informed Neural Networks Theory

Let u be our quantity of interest (QOI) satisfying the following ODE:

$$\mathcal{N}_t(u(t)) = f(t), \qquad t \in \mathcal{D} \quad (11)$$

Where $\mathcal{N}_t$ is a differential operator, $\mathcal{D}$ is the time domain and $f(t)$ is a known function. The aim of PINNs is to solve (11) by approximating the QOI using ANNs.

ANNs are a computational method, composed of many neurons connected to each other. Each neuron carries out a part of the total computation. Thanks to such connections ANNs can learn complex nonlinear functions.

ANNs can be seen as a direct acyclic graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ whose nodes are the neurons, and each edge connects the output of a neuron to the input of another neuron. Each neuron applies a simple function on the weighted sum of its inputs (12 and 13). This function $\boldsymbol{\sigma}: \mathbb{R} \to \mathbb{R}$ also called the activation function is chosen from a set of predefined functions; *e.g.* Rectified Linear Unit (RELU) $\boldsymbol{\sigma(t) = \mathbf{max}\,(t, 0)}$ or Sigmoid $\boldsymbol{\sigma(t) = \frac{1}{1+\exp(-t)}}$. More practically neurons can be stacked into layers by decomposing the set of neurons $\boldsymbol{V = \cup_{n=0}^{L} V_n}$. We denote $\boldsymbol{v_{n,i}}$ the $\boldsymbol{i^{th}}$ neuron of the $\boldsymbol{n^{th}}$ layer and $\boldsymbol{o_{n,i}(t)}$ its output for a given input $\boldsymbol{t}$.
Suppose we have calculated the output of layer $\boldsymbol{n}$, then the output of the layer $\boldsymbol{n+1}$ is as follows:

$$\boxed{a_{n+1,i}(t) = \sum_{k:(v_{n,k}, v_{n+1,i}) \in E} \left(\left((v_{n,k}, v_{n+1,i}) \right) \right) o_{n,k}(t)} \quad \textbf{(12)}$$

$$\boxed{o_{n+1,i}(t) = \sigma\left(a_{n+1,i}(t)\right)} \quad \textbf{(13)}$$

The $\boldsymbol{m}$ final outputs of the ANNs are given by:

$$\boxed{h_{w,i}(t) = a_{L,i}(t), \;\; i = 1..m} \quad (14)$$

Without any loss of generality, we can assume that $\boldsymbol{m = 1}$ yielding:

$$\boxed{h_w(t) = a_L(t)} \quad (15)$$

Figure 2 shows an example of ANNs with one input $\boldsymbol{t}$ (representing time in this case) and one output $\boldsymbol{q_j(t)}$ (representing the production of producer $\boldsymbol{j}$ at time $\boldsymbol{t}$). The ANNs has 6 hidden layers and 5 neurons per hidden layer.

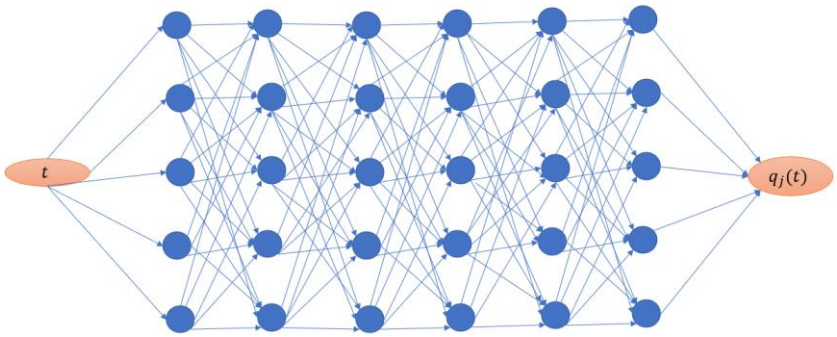


Figure 2 *Example of Artificial Neural Network with 6 hidden layers and 5 neurons per layer*

The weight function is $\boldsymbol{w}$: $\boldsymbol{E} \to \mathbb{R}$. Suppose we have $\boldsymbol{n}$ inputs and $\boldsymbol{m}$ outputs and a given weight function $\boldsymbol{w}$, we denote the function calculated by the networks as $\boldsymbol{h_w}$: $\mathbb{R}^{\boldsymbol{n}} \to \mathbb{R}^{\boldsymbol{m}}$. We define $\boldsymbol{\Delta(h_w(t), y)}$ the loss for predicting $\boldsymbol{h_w(t)}$ when the true (or target) value is $\boldsymbol{y}$. $\boldsymbol{\Delta}$ is a norm function, *e.g.* $\boldsymbol{l^1, l^2}$ ... for a given $\boldsymbol{\mathcal{H}}$ examples, the total loss of the networks is:

$$\boxed{\mathcal{L}_{\mathcal{H}}(w) = \mathbf{E}_{(t,y)\in\mathcal{H}}[\Delta(h_w(t), y)]} \quad \textbf{(16)}$$

In the rest of this paper $\boldsymbol{\mathcal{L}_{\mathcal{H}}}$ will be called the Multi-Layer-Perceptron (MLP) loss.
The output of the ANNs, approximating the QOI, should also satisfy the ODE **(11)** resulting in the next physical loss function:

$$\boxed{\mathcal{L}_{\mathcal{D}}(w) = \mathcal{N}_t(h_w(t)) - f(t) = 0, \forall\, t \in \mathcal{D}} \quad (17)$$

Thus, the final loss to minimize to get the optimal ANNs weight function w is:

$$\boxed{\mathcal{L}(w) = \mathcal{L}_{\mathcal{H}}(w) + \mathcal{L}_{\mathcal{D}}(w)} \quad (18)$$

3.3- CRMs + PINNs = Deep-CRMs

In order to integrate CRMs into PINNs, we propose a new architecture (Figure 3) composed of small architectures corresponding to each producer. All these small architectures have the same input which is the time t. In Figure 3 we can identify two ANNs, representing the producer BHP and the producer rate. Indeed, we decided to add one ANNs for producer BHP in addition to the one that approximates the producer rate. The interest of using these additional ANNs is twofold:

- ANNs naturally smooths the data. It is important for reducing the noise which can exist particularly in raw data (e.g. outliers).

- In the CRMs (7) we need to compute the derivative of the BHP over time. This latter value can directly be obtained thanks to the additional ANNs that approximates the BHP.

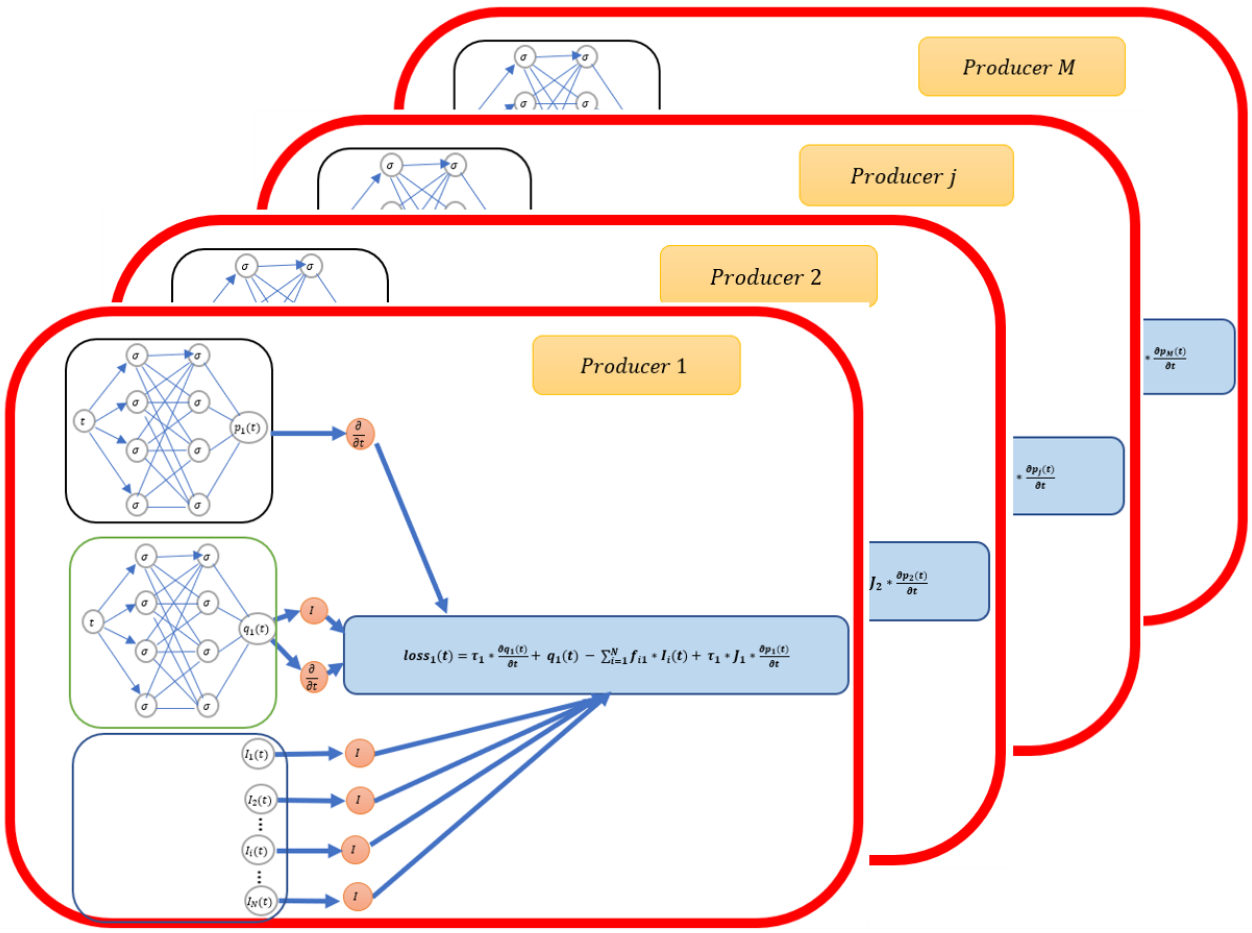


Figure 3 *Deep-CRMs Architecture*

The proposed architecture (Figure 3) allows us to compute the physical loss function for all the producers which is detailed in (19).

$$\mathcal{L}_{\mathcal{D}}(w_1, w_2, \dots, w_M) = \sum_{j=1}^{M} E_{(t)\in\mathcal{D}_j}\left[\tau_j \frac{dq_j(t)}{dt} + q_j(t) - \sum_{i=1}^{N} f_{ij} * I_i(t) + \tau_j * J_j * \frac{dp_j(t)}{dt}\right] \quad (19)$$

Where w_j is the weight function of the j^{th} producer ANNs for $j = 1,\dots,M$. $\mathcal{D}_j$ represents the set of points (time) where the j^{th} producer CRMs ODE should be satisfied.

At the same time, we compute for each producer the MLP loss for the ANNs dedicated to q_j , $j = 1..M$ resulting in the global MLP loss defined in (20).

$$\mathcal{L}_{\mathcal{H}}(w_1, w_2, \dots, w_M) = \sum_{j=1}^{M} \mathbf{E}_{(t,y_j)\in\mathcal{H}_j}\left[\Delta\left(h_{w_j}(t), y_j(t)\right)\right] \quad (20)$$

$\mathcal{H}_j$ contains the observations of the j^{th} producer, for $j = 1 \dots M$.

In the global architecture, we train the producers' BHP ANNs at the same time as the producers ANNs. This is done by incorporating the MLP loss for producers BHPs (21) in the global loss (22):

$$\mathcal{L}_{\mathcal{H}_p}(z_1, z_2, \dots, z_M) = \sum_{j=1}^{M} \mathbf{E}_{(t,\breve{p}_j)\in\mathcal{H}_{p_j}}\left[\Delta\left(h_{z_j}(t), \breve{p}_j(t)\right)\right] \quad (21)$$

z_j, $j = 1..M$ represents the weight function of the j^{th} producer BHP and $\mathcal{H}_{p_j}$ its corresponding set of observations.

At the same time, the producers should also respect the different constraints on the physical parameters (23), after we can define the global loss of Deep-CRMs:

$$\mathcal{L}_{Deep-CRMs}(w, z, f, \tau, J) = \mathcal{L}_{\mathcal{H}}(w) + \mathcal{L}_{\mathcal{D}}(w) + \mathcal{L}_{\mathcal{H}_p}(z) + \mathcal{L}_{injector}(f) + \mathcal{L}_{Connectvity}(f) + \mathcal{L}_{time\ constant}(\tau) + \mathcal{L}_{index\ of\ productivity}(J) \quad (22)$$

Where $f = (f_{ij})_{(i,j)\in 1..N\times 1..M}$, $\tau = (\tau_j)_{j=1\dots M}$ **and** $J = (J_j)_{j=1\dots M}$.

After defining $\mathcal{L}_{Deep-CRMs}$ we need to define an optimizer to minimize it giving us though the best weight functions and the best physical parameter (connectivity, time constant and index of productivity).

$$\begin{aligned}
\mathcal{L}_{injector} &= \sum_{i=1}^{N} max\left(0, \sum_{j=1}^{m} f_{ij} - 1\right) \\
\mathcal{L}_{Connectvity} &= \sum_{i=1}^{N}\sum_{j=1}^{M} max(0, f_{ij}) \\
\mathcal{L}_{time\ constant} &= \sum_{j=1}^{M} max(0, \tau_j) \\
\mathcal{L}_{index\ of\ productvity} &= \sum_{j=1}^{M} max(0, J_j)
\end{aligned} \tag{23}$$

In our work we have used RMSprop a gradient based method to minimize $\mathcal{L}_{Deep-CRMs}(w, z, f, \tau, J)$.

4- Experiments

The objective of the experimentations is to validate the efficiency and the robustness of our model on a synthetic dataset as well as on a real dataset. Section 4.1 describes the two datasets. Section 4.2 details the experimental protocol and gives the objectives. Finally, in section 4.3 the results are discussed.

4.1 Datasets

The two datasets are rescaled in order to make the learning faster and avoid the networks optimizer from stacking in local optima. Moreover, the real dataset has been smoothed in order to reduce noise and remove outliers from the dataset.

4.1.1 Synthetic Dataset

The synthetic dataset called Sondous was simulated using SIMAGE® and ECLIPSE® (see Figure 4 and Figure 5). It contains 97 observations irregularly distributed between 1/1/1988 and 16/11/1998. Figure 4 represents the field in 3D picture. We can observe the presence of a fault in the middle of the field (black line). The scale of colour corresponds to the net-to-gross (NTG). Figure 5 illustrates the same field in a 2D view, where the 3 producers (P1, P2 and P3) and 2 injectors (I1 and I2) are positioned. We can notice the presence of the fault separating the producer one (P1) from the rest of producers (P2 and P3) and injectors (I1 and I2). In Figure 5 the colour scale represents the reservoir pressure BHP of the producers.

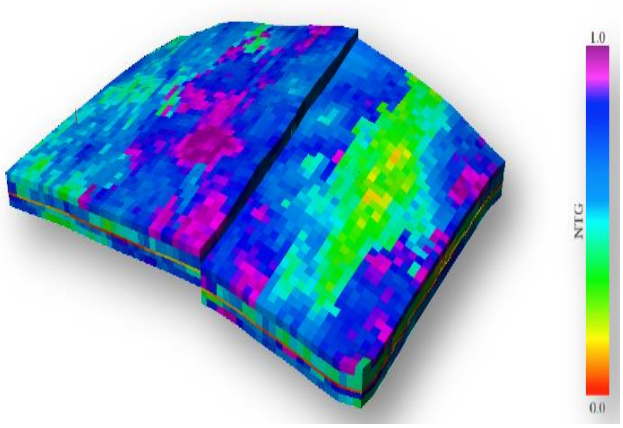


Figure 4 *Sondous Field NTG scale*

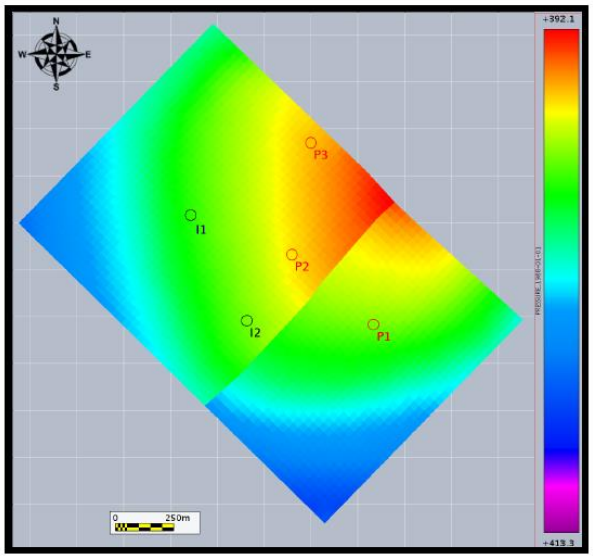


Figure 5 *Sondous Injectors and producers pressure scale.*

Figure 6 shows the liquid rates for the three producers, while Figure 7 shows the injection rate for the two injectors. We can observe that curve of P1 is significantly lower than those of P2 and P3. In Figure 7 we also note that the injection curves are piecewise constant.

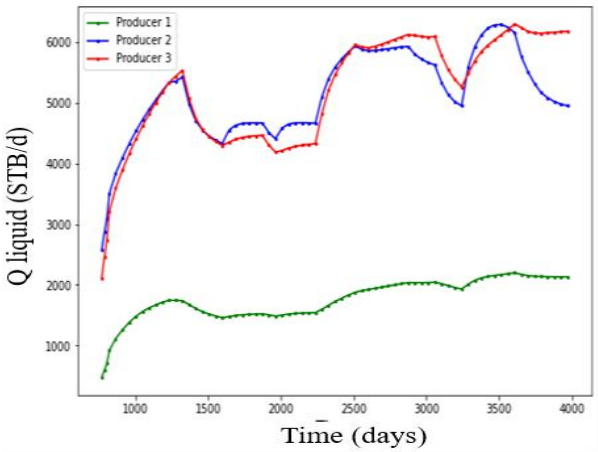


Figure 6 *Producers*

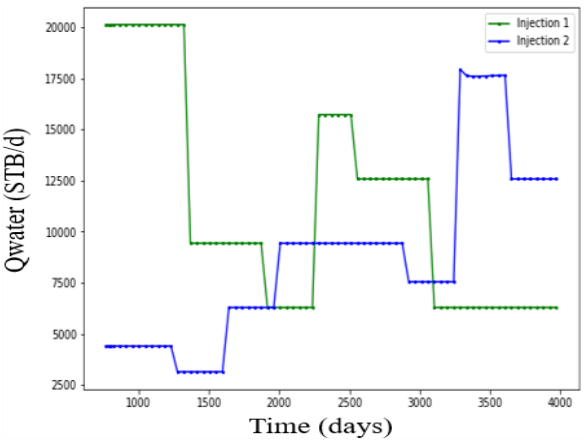


Figure 7 *Injectors*

4.1.2 Real Dataset

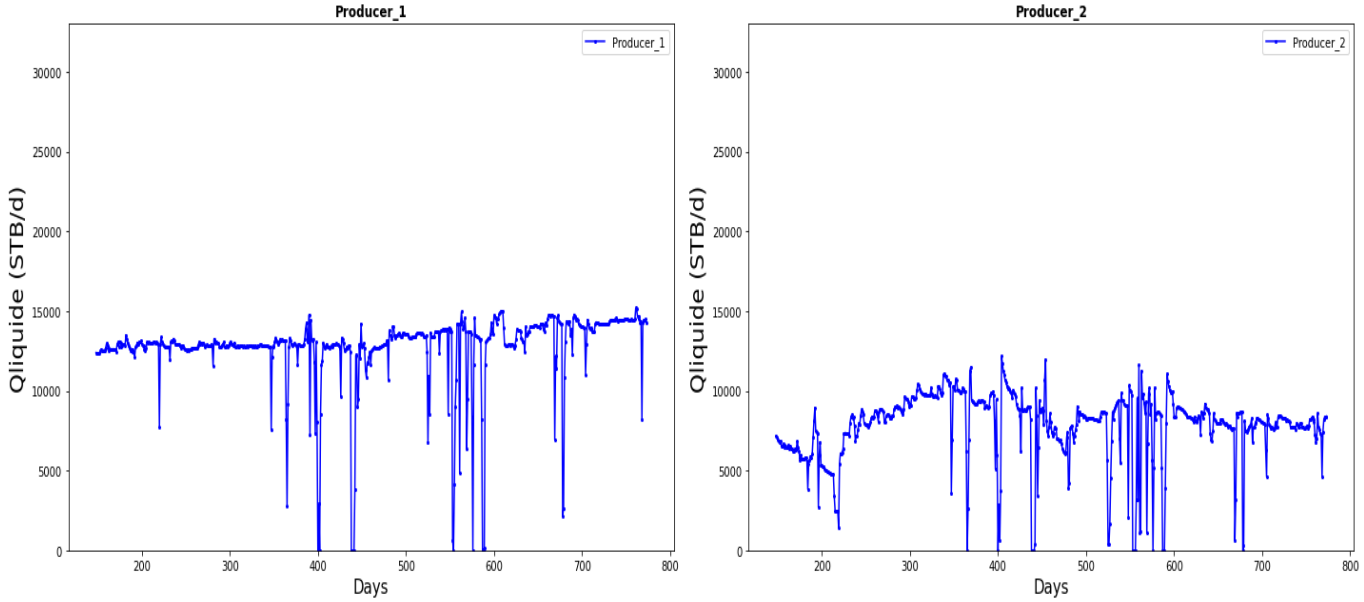
The real dataset provided by one of our affiliates, represents an offshore oil field. It contains 6 producers and 5 injectors (see Figure 8). Some connections between injectors and producers have already been proven by our affiliate as shown in Table 1. Different techniques were used by our affiliate to study connection between injectors and producers e.g. interference test, pressure response at producer wells when injection is on/off, salinity test, tracer test and 4D seismic images. In Table 1 green colour indicates proven or probable connection whereas red colour proves non-existence connection. Since all cases have not been studied by our affiliate; we cannot conclude on other "No Information" connectivity.

In the Figure 8 (a), (b) et (c) we show the daily production rates of the 6 producers over 700-time daily observations starting from April 30th, 2013 to June 12th, 2015. In comparison to the synthetic data (see Figure 6) we have more noisy data in the real dataset.

	Producer 1	Producer 2	Producer 3	Producer 4	Producer 5	Producer 6
Injector 1	No Information	No Information	No Information	Proven	No Information	No Information
Injector 2	No Information	Probable	No Information	No Information	No Information	No Information
Injector 3	Proven	Non-existence Proven	No Information	No Information	No Information	Probable
Injector 4	No Information	No Information	No Information	Proven	Proven	Proven
Injector 5	Proven	Proven	No Information	No Information	No Information	Proven

Table 1 *Real Dataset Connectivity Table*

(a)



(b)

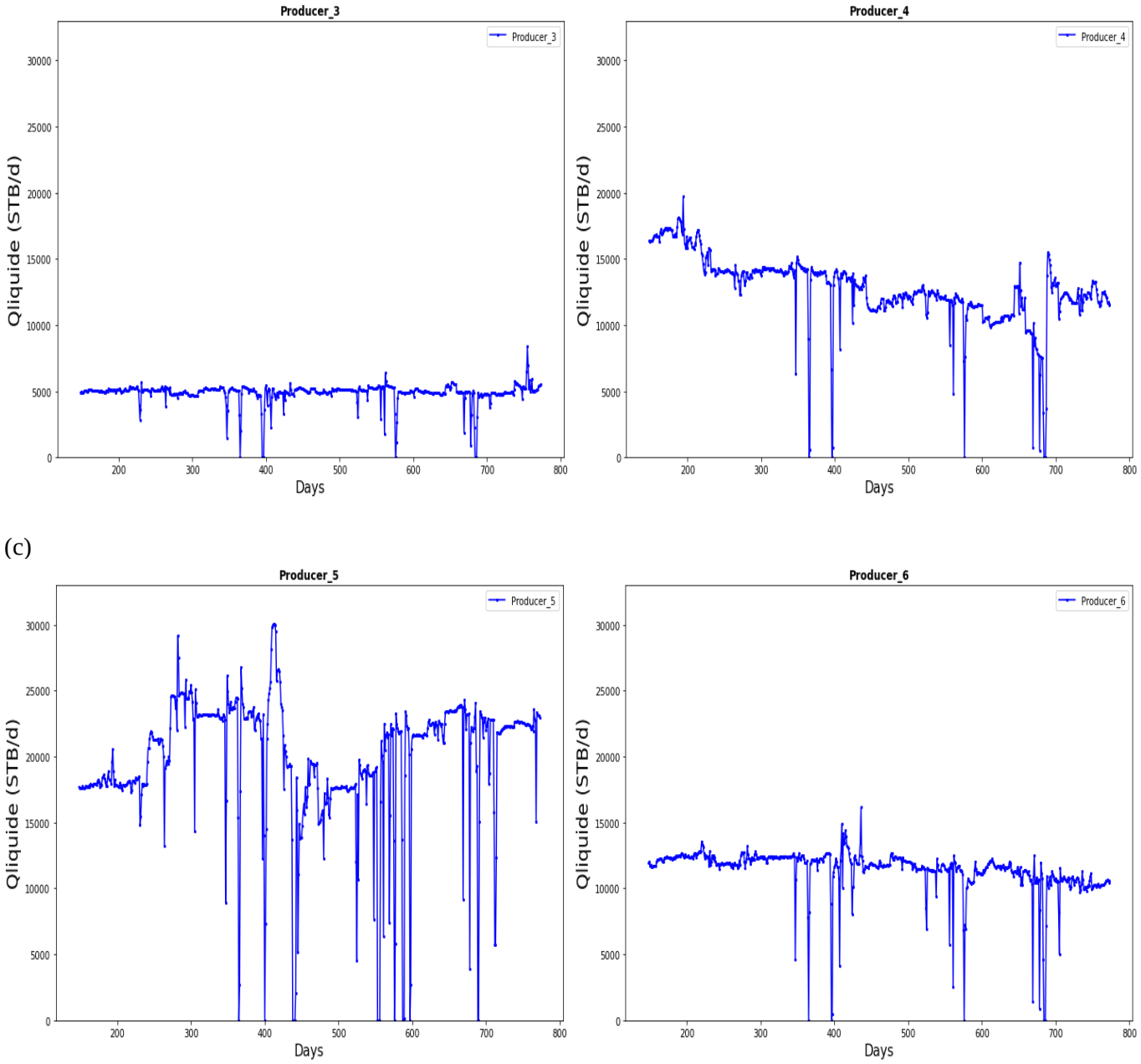


Figure 8 *Real Dataset producers*

4.1.3 Data Preparation

Data preparation represents the different transformations applied to the raw data, before presenting it to the model. First, a rescaling is needed to improve the training process by making the datasets comparable within the same scale. Secondly, in real dataset we often have numerous outliers, that need to be removed. It is important to remove the noise from the data by applying smoothing principles; in our case we have used ANNs to smooth the real dataset.

Data rescaling. Before applying Deep-CRMs we have rescaled the injections and the rates production by the maximum rate production computed over all the producers; the pressure was rescaled with respect to the maximum pressure. Rescaling the data aims at making the learning faster and avoids the network from stacking in local optima.

Data Smoothing with ANNs. Data smoothing techniques are usually independent from the learning model. In our case data smoothing is performed at the same time as the data training process. We directly used the ANNs of Deep-CRMs to smooth the producer bottom hole pressure (BHP).
This is done by incorporating in the global model loss (22) the ANNs loss of the BHP ((21), $\mathcal{L}_{\mathcal{H}_p}(z_1, z_2, \dots, z_M)$, is incorporated in the (22) of the global loss). An interesting consequence of this idea is that we do not need to compute manually the BHP gradients (as in classical approaches); in our case the gradients can be computed directly from the ANN approximation of BHP.
Moreover, ANNs smoothing facilitate the gradient computation thanks to the automatic differentiation and the derivability of the ANNs.

In the same way, the liquid rate is smoothed implicitly by the ANNs of Deep-CRMs (the formula $\mathcal{L}_{\mathcal{H}}(w)$ and $\mathcal{L}_{\mathcal{D}}(w)$ are incorporated in (22)).

For instance, in Figure 9 we illustrate the BHP of producer 1 before smoothing (a) and after ANNs smoothing (b). We superpose the cases before and after on the same figure in (c). The red curve is directly obtained after training Deep-CRMs.

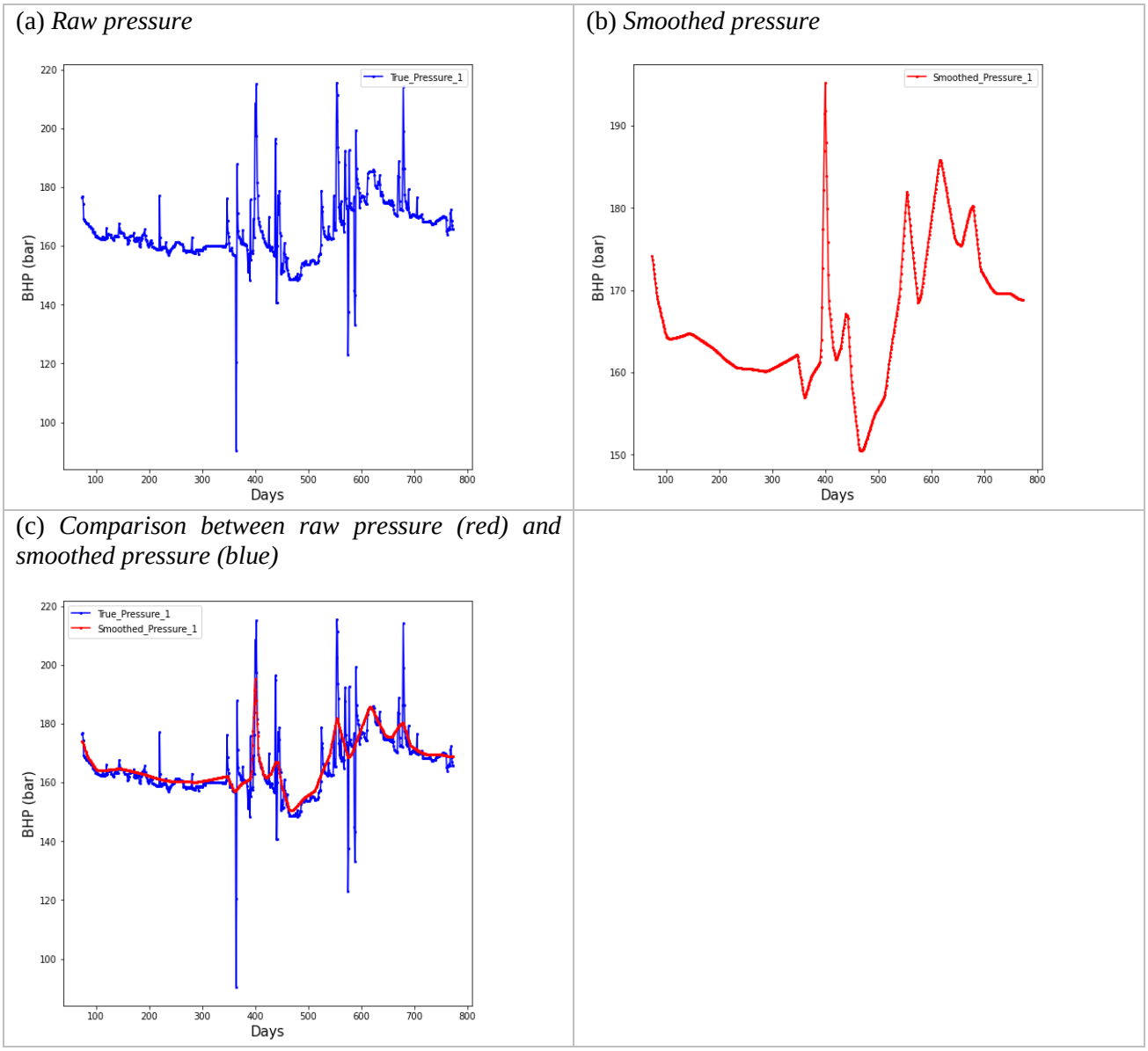


Figure 9 *Real dataset pressure for producer 1 (a) before and (b) after filtering. A comparison between raw and smoothed pressure is illustrated in (c)*

4.2 Objectives and Protocols

Deep-CRMs have dual objectives, first discovering the underlaying connectivity between injectors and producers and secondly performing forecasting giving the amount of injected water and the bottom hole pressure BHP of each producer. In these experiments we are interested in those dual objectives. At the same time, it is also interesting to compare Deep-CRMs to classical optimizer relying on the closed form solution.

For neural networks, there is no universal configuration (optimizer, number of neurons, number of layers, activation function) that works for every case. Each case study is different, though we need to test several configurations and validate them on a validation set. From which we retain the best configuration providing the minimum loss. this process is called grid search. To effectively scan the different configurations, we define a set of values for each hyper parameter and we check all the possible combinations. This systematic evaluation is called a grid search. This process will be adopted in the rest of this paper.
For Sondous test case, training was performed on 50% of the data, 10% of the data was used as validation for selecting the best model and 40% for testing. For the real dataset 60% of the dataset was used for training and 40% for testing.

The following metrics were used for model selection and comparison with classical optimizer:

$$\boldsymbol{MSE}\ (\boldsymbol{y},\hat{\boldsymbol{y}}) = \mathbb{E}\left[(\boldsymbol{y}-\hat{\boldsymbol{y}})^2\right] = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{y_i}-\hat{\boldsymbol{y}}_{\boldsymbol{i}})^2 \quad (\mathbf{24})$$

The Mean Squared Error (MSE) measures the average square error of approximating y by $\hat{y}$. The bigger the MSE the bigger the error of approximation. When dealing with no rescaled data it is better to use the normalized MSE:

$$\boldsymbol{NMSE}\ (\boldsymbol{y},\hat{\boldsymbol{y}}) = \frac{\boldsymbol{MSE}(\boldsymbol{y},\hat{\boldsymbol{y}})}{\mathbf{Max}(\boldsymbol{y})} \quad (\mathbf{25})$$

Another widely used metric for regression problem is the $\boldsymbol{R^2}$. The close $\boldsymbol{R^2}$ to 1, the close $\boldsymbol{y}$ to $\hat{\boldsymbol{y}}$.

$$\boldsymbol{R^2}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \mathbf{1} - \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{y_i}-\hat{\boldsymbol{y}}_{\boldsymbol{i}})^2}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^2} \quad (\mathbf{26})$$

4.3 Results

4.3.1 Synthetic Dataset: Sondous

4.3.1.1 First objective: forecasting

The first objective of Deep-CRMs is to predict future liquid production given the historical production data, the injection data, and the BHP data. Deep-CRMs use those data to minimize its loss (22) by adjusting the ANN weights (learning the weight function). After that Deep-CRMs can forecast producer's production for future time knowing the future injection and BHP.
Figure 10 represents the evolution of Deep-CRMs main loss functions over the number of iterations. Training loss is the general loss (22) to minimize (blue curve), containing Physics loss (green curve) on the CRMs ODE, Multi-Layer Perceptron loss (red curve) on the observation points, plus the ODE constraints losses. We can observe that our optimizer RMSprop minimizes the different loss over the number of iterations. Moreover, we can see that the different loss start getting constant after the 20.000 th iteration. Thus, we have fixed the number of iterations at 30.000. the validation loss is particularly important since its variation controls the generalization of the model. A very known problem in machine

learning is the model overfitting. It means that the model cannot generalize outside the learning dataset in such case the validation loss will increase although the model general loss is decreasing. In our case the validation loss is decreasing which indicates that our model is not overfitting and it can generalize outside the learning dataset.

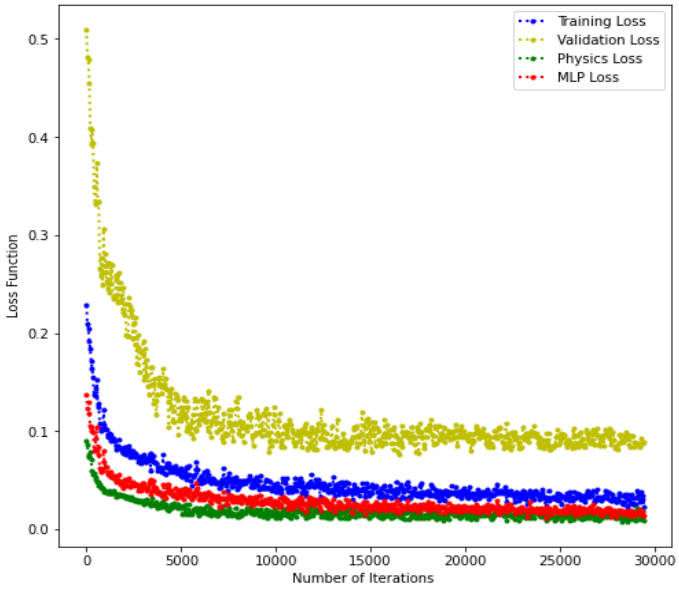


Figure 10 *Training loss and validation loss*

Figure 11 shows the different models, the best model (red curve), and the analytical solution (dark curve). Before the red vertical line, data are used for training, between the red line and the dark-olive line we have the validation data and after the dark-olive line we have the data used for testing the model. First observation is that all the different models are quite similar in terms of forecasting. Meaning that with 6 hidden layers and 300 neurons per hidden layer (1800 parameters) we have quite similar performance as with 5 neurons and 200 layers (1000 parameters). This is important because generally in deep learning the famous architectures are those with millions of parameters. One explanation could be that in classical deep learning models we are trying to capture the underlying phenomenon with only observations data. This is why we need a lot of observations data and a very sophisticated architecture (e.g. millions of parameters) without forgetting the amount of time needed to train such architecture. whereas in the case of PINNs we already have a modelling of the underlying phenomenon in our case CRMs ODE, and we are only using deep learning architecture as an approximator of the underlying phenomenon thus we do not need a lot of data neither a huge time of computation. Second observation is that the different models capture the same variation as in the real dataset. In CRMs model we assume that the variation of rate production comes only from its relationship with injectors and producers. Thus, we can see (Figure 11) that our models have learned this relationship since they can forecast future rate production knowing only time, corresponding injection, and corresponding BHP. The third observation is that Deep-CRMs does not need a huge amount of data to be trained, in our case 50% of the total data was sufficient to perfectly train the model.

Model selection is performed on the dark part of the dataset (validation) using Normalized Mean Squared Error (NMSE). Table 2 shows the values of NMSE per model producer, where Model 4 is the best model in terms of NMSE.

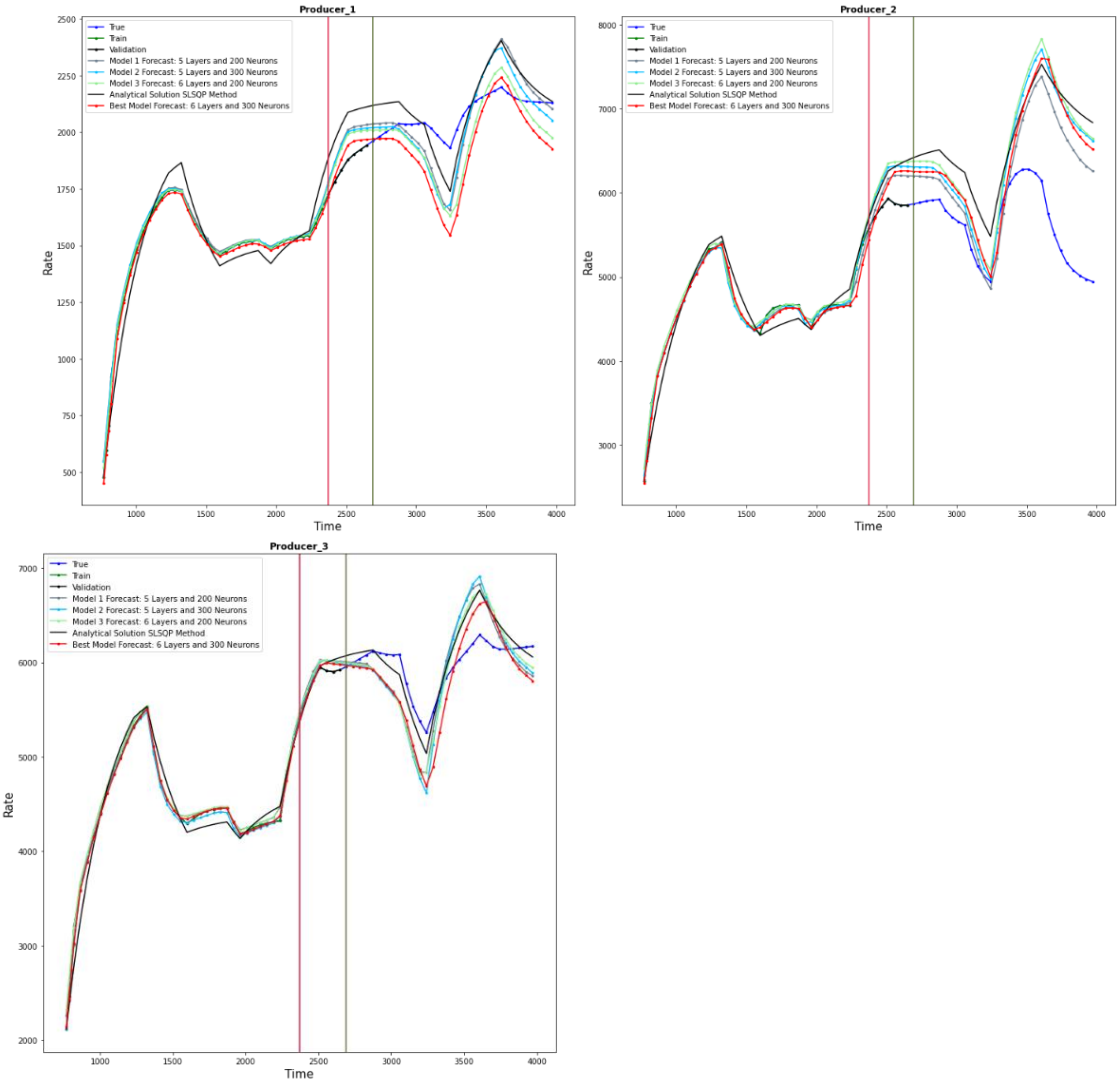


Figure 11 *Deep-CRMs liquid rates (STB/d) test for the 3 producers (Sondous data)*

	Producer 1	Producer 2	Producer 3	Total NMSE
Model 1	1.72	10.22	1.22	13.16
Model 2	1.46	21.3	0.79	23.55
Model 3	1.15	26.76	1.04	28.95
Model 4 (The Best)	0.3	12.17	0.48	**12.95**

Table 2 *NMSE Validation*

The Sondous dataset represents a synthetic particular case, where we have stepwise variation of injection rates and linear variation of BHP; in such conditions CRMs analytical solution is valid and parameter optimization can be done using a nonlinear multivariate regression.
We have compared Deep-CRMs solution to the nonlinear multivariate regression with SLQP [18] optimizer. Table 3 shows that Deep-CRMs has an NMSE 13% lower than the analytical solution. This proves that even in a very simple case, where we can apply the analytical solution, Deep-CRMs can give better results than the analytical solution with a classical optimizer.

	NMSE
Deep-CRMs	**188.37**
Nonlinear Multivariate Regression (NMR)	217.11

Table 3 *NMSE comparison between Deep-CRMs vs NMR*

4.3.1.2 Second objective: physical parameter

During the optimization process Deep-CRMs will update the values of each parameter: function weight $\boldsymbol{w}, \boldsymbol{z}$ and physical parameter $\{\boldsymbol{f}_{ij\,(i=1..N, j=1..M)}, \boldsymbol{\tau}_{j\,j=1..M}\}$. In our case we have used RMSprop optimizer to obtain these parameters.

In Table 4 we show the obtained values for the physical parameters: wells connectivity and time Constants.

Connectivity	Producer 1	Producer 2	Producer 3
Injector 1	**0.07**	0.21	0.22
Injector 2	**0.11**	0.36	0.32
Time Constant (days)	204.35	208.39	211.54

Table 4 *Connectivity and Time Constant*

Table 4 shows that, connectivity between Injector 1 and Producer 1 is nearly equal to zero, and the connectivity between Injector 2 and Producer 1 is smaller than the connectivity of Injector 2 with other producers. These results could confirm the presence of a partially sealing fault separating Producer 1 from other Injectors.

4.3.2 Real Dataset

Testing Deep-CRMs on a synthetic dataset shows that Deep-CRMs can satisfy the two objectives: forecasting and parameter discovery. However, in real fields we do not necessarily have stepwise variation in injection or linear variation in BHP, plus that the rate production is often more complicated to model and thus to forecast. To prove the ability of Deep-CRMs we have used a real field dataset provided by one of our affiliates.

4.3.2.1 First objective: forecasting

Figure 12 shows the results of Deep-CRMs on the real dataset. The vertical yellow line separates 60% of the data used for training and 40% used for testing. For Producer 3, 4 and 6 the model forecasts are quite similar to the real dataset. For Producer 1, Producer 2 and Producer 5 forecasts are affected by the high variations (complete and small shut in production rate during training data) in the training data as well as in injections and BHP, but still close to the real data.

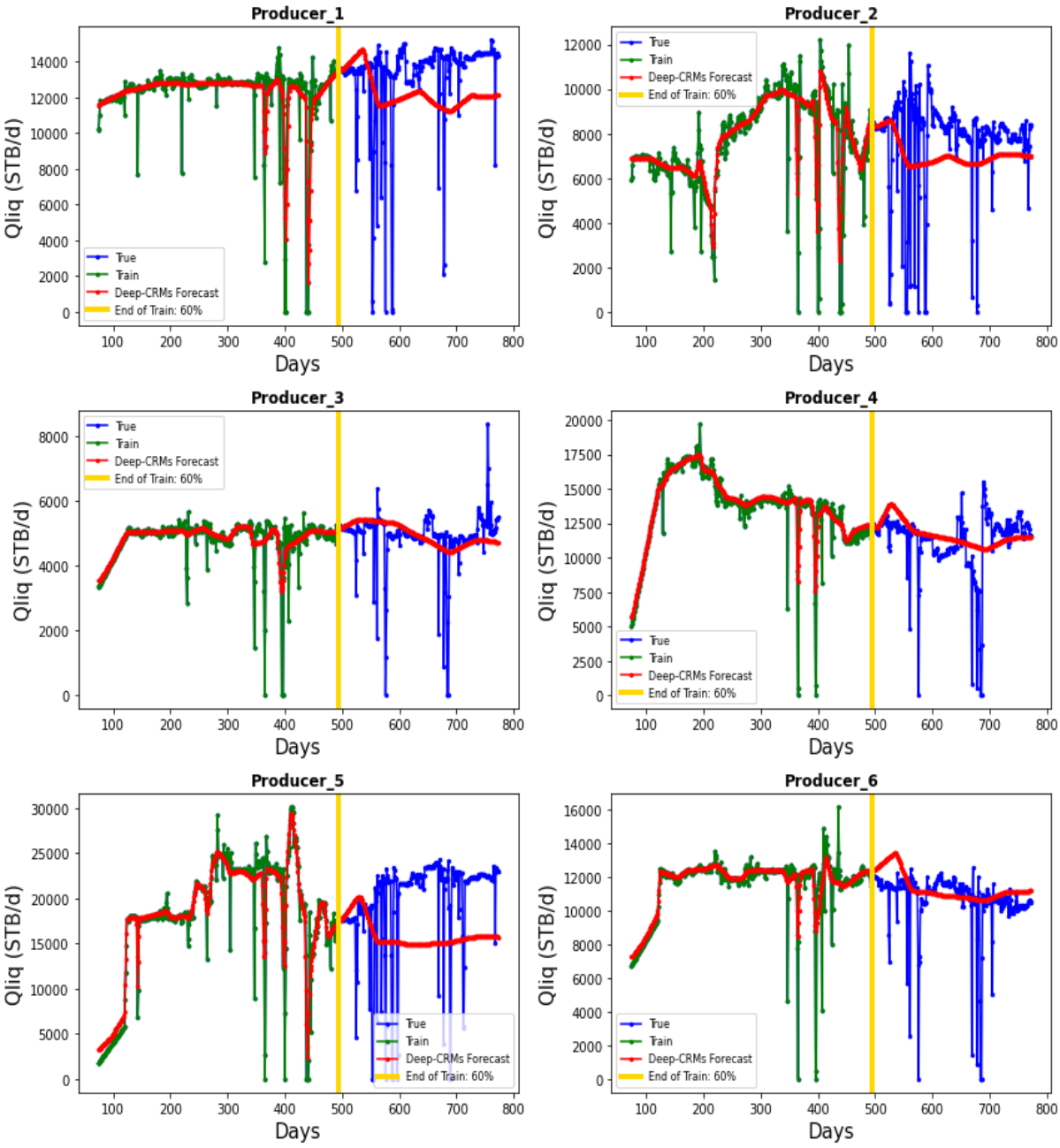


Figure 12 *Deep-CRMs*

4.3.2.2 Second objective: physical parameter

Table 4 presents the physical parameter (i.e. connectivity per couple producer-injector) obtained after Deep-CRMs optimization. If we observe these results, we can see that in the case where our affiliate analysis found a proven, probable, or no-existence connection our model found a coherent connection with (Table 1). In Table 4, we recall the conclusions of Table 1. Thus, green colour indicates proven or probable connection between injector and producer. The red colour shows non-existence connection. We can remark that the model affects values to connection that were not studied by our affiliate. One explanation could be that our model has identified connections that our affiliate did not identify or consider not worth to be studied. We still be careful with this explanation and we are waiting the feedback of our affiliate to confirm or not such connections.

	Produce 1	Produce 2	Produce 3	Produce 4	Producer 5	Producer 6
Injector 1	**0.30**	**0.12**	**0.03**	**0.23**	**0.19**	**0.09**
Injector 2	**0.15**	**0.07**	**0.01**	**0.20**	**0.35**	**0.17**
Injector 3	**0.21**	**0.05**	**0.15**	**0.0**	**0.15**	**0.25**
Injector 4	**0.14**	**0.06**	**0.12**	**0.17**	**0.26**	**0.23**
Injector 5	**0.11**	**0.19**	**0.07**	**0.21**	**0.24**	**0.15**

Table 5 *Computed Connectivity between Producers and Injectors on real dataset*

5- Conclusions and future work

In this work, we have presented a new approach called Deep-CRMs to identify CRMs' parameters and to perform production rate forecasting, without the need of any prior assumptions on injections and on producers' bottom hole pressure or a closed form solution as in the state-of-the-art on CRMs. Deep-CRMs is a fully ANN-based approach, including data preparation and data modelling. Deep-CRMs was tested on two datasets: the first is a synthetic dataset, where we have showed that Deep-CRMs can explain the underlying geology (e.g. presence of a fault in SONDOUS) and perform better forecasting than the analytical solution with nonlinear multivariate regression with the SLSQP [18] optimizer. The second dataset is a real field dataset, provided by one of our affiliates, for which Deep-CRMs gave quite satisfactory results particularly in terms of forecasting and parameter identification. In this work, uncertainty was not treated and was left to future work, but different solutions can be used to get uncertainty with Deep-CRM such as Dropout [19] or Bayesian Neural Networks.

6- References

1. Rafael Wanderley de Holanda, Eduardo Gildin, Jerry L. Jensen, Lary W.Lake, C.shah Kabir State-of-the-Art Literature Review on Capacitance Resistive Models for Reservoir Characterization and Performance Forecasting, 2018.
2. Ali Abdallah Youssef, Pablo Hugo Gentil, Jerry L Jensen, Lary Lake. A Capacitance Model to Infer Interwell Connectivity from Production and Injection Rate Fluctuations. SPE Reserv. Eval. Eng. **2006**, 9, 630–646. [CrossRef]
3. Ali Abdallah Al-Yousef, investigating statistical techniques to infer interwell connectivity from injection and production rate fluctuations. Ph.D. Dissertation, University of Texas, Austin, TX, USA, 2006.
4. Morteza Sayarpour, Lary Lake, C.Shah Kabir, Elizabeth Zuluaga, The use of capacitance resistive models for rapid estimation of waterflood performance and optimization, 2007.
5. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep learning, MIT Press, 2016.
6. Maziar Raissi, Paris Perdikaris, GE Karniadakis , Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations - arXiv preprint arXiv:1711.10561, 2017
7. Maziar Raissi, Paris Perdikaris, GE Karniadakis, Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations arXiv preprint arXiv:1711.10561, 2017.
8. Nicholas Geneva, Nicholas Zabaras, Modelling the dynamics of PDE systems with physics constrained Deep Auto-Regressive Networks, 2019.
9. Yinhao Zhu, Nicolas Zabaras, Phaedon-Stelios Koutsourelakis, Paris Perdikaris, Physics-Constrained Deep learning for high dimensional surrogate modelling and uncertainty quantification without labelled data, 2019.
10. Cheng Yang, Xubo Yang, Xiangyun Xiao, Data-driven projection method in fluid simulation, Computer Animation and Virtual Worlds 27 (3-4) (2016) 415–424.

arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav. 1695, doi:10.1002/cav.1695. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1695

11. Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar, Foundations of Machine Learning, MIT Press 2012.
12. Robert E.Schapire, Yoav Freund, Boosting Foundations and Algorithms, MIT Press 2012.
13. Steve Cannon, Reservoir Modelling A Practical Guide, 2018.
14. L.P Dake, Fundamentals of Reservoir Engineering.
15. Morteza Sayarpour, Development and applications of Capacitance Resistive Models to water CO2/Floods. Ph.D. Dissertation, University of Texas, Austin, TX, USA, 2008.
16. Dimitris C.Psichogios, Lyle H.Ungar. (1992). A hybrid neural network-first principle approach to process modeling. AIChE Journal, 38(10), 1499–1511. doi:10.1002/aic.690381003
17. Maziar Raissi, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, https://maziarraissi.github.io/research/09_hidden_fluid_mechanics/
18. Richard H. Byrd, Nicolas I.M Gould, JORGE Nocedal, RICHARD A. Waltz, On the convergence of successive linear-quadratic programming algorithms - SIAM Journal on Optimization, 2005 – SIAM
19. Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.