Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-1 (2019) 808–813

Using a neural network for estimating plant gradients in real-time optimization with modifier adaptation

José Matias * **Johannes Jäschke** *

* *Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway (e-mail: jose.o.a.matias@ntnu.no, johannes.jaschke@ntnu.no)*

Abstract: In the presence of structural plant-model mismatch, standard real-time optimization (RTO) schemes are prone to compute an operation point that does not coincide with the plant optimum. Modifier Adaptation (MA) methods are RTO variants that have the ability to reach plant optimality even in the case of structural plant-model mismatch. However, MA implementations require plant gradient information, which is challenging to obtain. This work proposes a method for estimating plant gradients based on neural networks (radial basis function network - RBFN). Our method is applied for obtaining the gradients of a gas lifted oil well network, which is then optimized using MA. The results show that, even with measurement noise, the gradients are estimated within an adequate precision and the MA method is able to increase production of the well network, reaching the plant optimum without any constraint violations despite the presence of plant-model mismatch.

© 2019, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Neural Network, Gradient Estimation, Modifier Adaptation, Real-time Optimization, Gas Lifted Oil Wells.

1. INTRODUCTION

Currently, in standard Real-Time Optimization (RTO) applications, nonlinear rigorous steady-state models are optimized on a regular basis in order to determine the best operating conditions for the plant (e.g maximizing profit) while satisfying process constraints. In order to optimize the system, standard RTO approaches use measurements to estimate the parameters, updating the process model to the current plant state (Darby et al., 2011).

However, even the updated model may not represent the plant accurately. For example, it can overestimate the value of a particular constraint. In these cases, the RTO computes an optimal operation point that does not coincide with the plant optimum, or worse, is not feasible. This phenomenon, which is called plant-model mismatch, has been extensively studied in the RTO literature (Darby et al., 2011). Several proposals for RTO variants deal with process optimization in the presence of plant-model mismatch. Among them, Modifier Adaptation (MA) methods (Marchetti et al., 2016) are especially attractive due to their simplicity and convergence properties.

In spite of guaranteeing that the optimization problem has a local minimum at the plant optimum, implementing MA methods in real situations is a challenging task due to the necessity of estimating the plant gradients (Marchetti et al., 2016). Several methods have been proposed for obtaining the plant gradient information in MA applications. A selection of dynamic and steady state methods for estimating plant gradients is shown in Table 1.

Given all the available options, there is no consensus about the best gradient estimation method. Many authors apply Finite Difference Approximation (FDA), which is the simplest and most straightforward approach, but the method becomes inefficient for noise-contaminated processes or for processes with a large number of inputs. Other methods approximate the plant cost and constraint functions by local quadratic models and are more robust to noise (Gao et al., 2016). However, these methods require probing points in order to guarantee a well-posed regression set to estimate the parameters of the surrogate models.

This paper proposes a new method for estimating plant gradients based on a neural network and applies it to a gas lifted oil network (Krishnamoorthy et al., 2016; Matias et al., 2018). Specifically, a radial basis function network (RBFN) is used to obtain the gradients. The method relies on an adaptive model representation, in which the residual between model and plant predictions of the cost and constraints (plant-model mismatch) is modeled by a RBFN. The plant gradients are estimated exploring the structure of the trained network.

The proposed adaptive method for estimating the gradients proceeds in a similar way as the *Gradients from fitted surfaces method* (Gao et al., 2016). The main difference is the nature of the function that is fitted to past data. Here, a hybrid (gray-box) approach for estimating the gradients combines physical knowledge (model) with data-driven parts (network trained in the residuals using experimental data). Thus, it can take advantage of existing process knowledge, which is not explored if a purely data-driven approach is used (polynomials or spline models).

2405-8963 © 2019, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2019.06.161

Table 1. Gradient estimation methods

Dynamic perturbation methods
1. *Dynamic model identification*, e.g. Mansour and Ellis (2003): Dynamic models are adjusted to the plant data and the steady-state gradients are obtained using the final-value theorem.
2. *Extremum-seeking control*, e.g. Golden and Ydstie (1989): Perturbation signals are added to the inputs and the steady-state gradients are calculated using data-driven methods.
Steady-state perturbation methods
1. *Finite-difference approximation (FDA)*: The inputs are perturbed individually around the current operation point and, by measuring the variation of the outputs, FDA is able to calculate the gradient.
2. *FDA variant* of Brdyś and Tatjewski (1995): It uses past operation points to calculate the gradients instead of further perturbing the plant like in FDA.
3. *Directional Modifier Adaptation* (Singhal et al., 2017): It estimates plant gradients only in privileged directions in the input space to decrease the number of plant perturbations.
4. *Broyden-like methods*, e.g. Rodger and Chachuat (2011): Gradients are estimated from past operation points by a recursive updating scheme.
5. *Gradients from fitted surfaces*, e.g. Gao et al. (2016): Experimental data is fit to polynomials or spline curves and the gradients are evaluated analytically by differentiating the fitted model.

In comparison to FDA and its variants, the proposed adaptive method uses data from points distributed in the operating region, which can decrease the influence of noise (Gao et al., 2016).

The case study shows that despite the presence of noise, our adaptive scheme is able to predict plant gradient accurately and the MA method, which uses only the phenomenological model, is able to drive the model-based optimization to the actual plant optimum in a few iterations.

The paper is organized as follows. Section 2 briefly explains the Modifier Adaptation method and Section 3 describes the problems related to estimating plant gradients in MA applications. In Section 4, the RBFN is introduced. Next, the adaptive model representation is presented in Section 5. Then, the next Sections discuss the case study using the adaptive approach and show the results. The paper is concluded in Section 9.

2. MODIFIER ADAPTATION

Modifier Adaptation was proposed by Marchetti et al. (2009). Instead of estimating the model parameters, MA methods incorporate plant measurements to the model via correction terms (modifiers) for the optimization problem. The modifiers express the difference between actual measurements and predicted values, and the difference between model gradients and plant gradients of the cost and constraint functions. By using the modifiers, MA forces the optimization problem to have a local minimum at the plant optimum even in the presence of structural plant-model mismatch (Marchetti et al., 2016).

Typically, the model-based optimization problem is set to minimize an economic cost function J, while respecting the operation constraints C. In turn, MA modifies the optimization problem using the modifiers as shown below:

$$\begin{aligned} \mathbf{u}^{\star}_{k+1} &= \arg\max_{\mathbf{u}} \quad J_{mod} := J(\mathbf{u}) + \lambda_{J,k}^{T}\mathbf{u} \\ \text{s. t.} \quad C_{mod} &:= C(\mathbf{u}) + \epsilon_{C,k} + \lambda_{C,k}^{T}(\mathbf{u} - \mathbf{u}_k) \end{aligned} \tag{1}$$

where, $\mathbf{u}$ are the process inputs. The subscript k indicates that the values are evaluated at the k^{th} MA iteration. $\lambda_{J,k}$, $\epsilon_{C,k}$ and $\lambda_{C,k}$ are the modifiers. We distinguish between cost function modifiers and constraint modifiers by using the subscripts J and C, respectively. Since the modifiers can be affected by noisy measurements and poor gradient estimates, they need some filtering to avoid overaggressive corrections that may destabilize the system:

$$\begin{aligned} \epsilon_{C,k} &= (I - K_{\epsilon,C})\epsilon_{C,k-1} + K_{\epsilon,C}(C_{p,k} - C_k) \\ \lambda_{C,k} &= (I - K_{\lambda,C})\lambda_{C,k-1} + K_{\lambda,C}(\widehat{\nabla C}_{p,k} - \nabla C_k) \\ \lambda_{J,k} &= (I - K_{\lambda,J})\lambda_{J,k-1} + K_{\lambda,J}(\widehat{\nabla J}_{p,k} - \nabla J_k) \end{aligned} \tag{2}$$

in which, $K_{\epsilon,C}$, $K_{\lambda,C}$ and $K_{\lambda,J}$ are the filter gain matrices, which are square matrices with values between $[0, 1)$ on the main diagonal and zeros elsewhere; I is the identity matrix of appropriate dimension; C_k are the constraint values predicted by the model; ∇C_k and ∇J_k are the model derivatives of the constraints and cost with respect to $\mathbf{u}$ at the current operating point. The plant constraint value C_p is usually directly measured, however, obtaining the plant gradients $\widehat{\nabla C}_p$ and $\widehat{\nabla J}_p$ can be challenging and is the topic of this paper.

Additionally, it can be shown that the selection of the modifiers according to Equation (2) leads to convergence to a plant minimum under standard regularity assumptions (Marchetti et al., 2009).

3. PLANT-MODEL MISMATCH AND PLANT GRADIENT ESTIMATION

Mathematical models are approximations of the actual process. Hence, it is unlikely that there is no difference between the model prediction and plant measurements, even for elaborate models that cover a wide range of phenomena. For example, in Equation (2), $C_{p,k}$ and $J_{p,k}$ may be quite different from C_k and J_k for the same input values $\mathbf{u}$. One way of quantifying the plant-model mismatch is to use additive error terms:

$$\begin{aligned} C_p(\mathbf{u}) &= C(\mathbf{u}) + e^{C}_{rr}(\mathbf{u}, \mathbf{n}) \\ J_p(\mathbf{u}) &= J(\mathbf{u}) + e^{J}_{rr}(\mathbf{u}, \mathbf{n}) \end{aligned} \tag{3}$$

in which, the vectors $e^{C}_{rr}(\mathbf{u}, \mathbf{n})$ and $e^{J}_{rr}(\mathbf{u}, \mathbf{n})$ contain the noise, n, in addition to the modeling errors. Since it is relatively easy to obtain the pairs (u, e^{C}_{rr}) and (u, e^{J}_{rr}) using historical data, a data-driven approach can be used for modeling e^{C}_{rr} and e^{J}_{rr} as a function of the input (and possibly measured disturbances). More interesting, depending on the structure of this data-driven model, the gradients related to the residual can be estimated and, consequently, the plant gradients easily computed:

$$\widehat{\nabla C}_p(\mathbf{u}) = \nabla C(\mathbf{u}) + \nabla \widehat{e^{C}_{rr}}(\mathbf{u}) \tag{4}$$

The procedure for $\widehat{\nabla J}_p$ is the same. Essentially, e_{rr} and $\nabla \widehat{e_{rr}}$ combine the zeroth and first order modifiers. How-

ever, as they are not computed using a filter, like in Equation (2), and they are connected to the gradient estimation and not the economic optimization problem, we choose to use a different nomenclature to avoid confusion.

4. RADIAL BASIS FUNCTION NETWORK (RBFN)

Despite the fact that radial basis function networks are commonly used for classification, they can be used for function approximation (for example, for modeling e^C_{rr}) (Johansen and Foss, 1992). Moreover, given RBFN's structure, it is easy to obtain the gradients $\nabla\widehat{e^C_{rr}}$ and $\nabla\widehat{e^J_{rr}}$ after training the network. RBFN is also chosen because its representation is much more intuitive than for other neural networks. RBFN neurons can take the form of Gaussian-like function, in which each neuron is represented by a bell curve centered at given input value, called μ. The center value can be seen as a prototype value for a specific region of the input set. When a new input $\mathbf{u}$ enters the system, the network classifies its similarity to the all the neurons of the network. The similarity is measured by the neuron activation function:

$$\phi_i(\mathbf{u}) = \exp^{-\beta(\mathbf{u}-\mu_i)^T(\mathbf{u}-\mu_i)} \quad i = 1, \dots, n_n \tag{5}$$

where, n_n is the number of neurons and β is the parameter that controls the width of the bell curve. In order to improve the accuracy of the function approximation, the output of a neurons is is normalized between 0 and 1 by diving $\phi_i(u)$ by the sum of all activation functions (Johansen and Foss, 1992):

$$r_i(\mathbf{u}) = \frac{\phi_i(\mathbf{u})}{\sum_{l=1}^{n_n}\phi_l(\mathbf{u})} = \frac{\mathrm{e}^{-\beta(\mathbf{u}-\mu_i)^T(\mathbf{u}-\mu_i)}}{\sum_{l=1}^{n_n}\mathrm{e}^{-\beta(\mathbf{u}-\mu_l)^T(\mathbf{u}-\mu_l)}} \tag{6}$$

The network output is the weighted sum of the all normalized neuron activation functions $r_i(u)$. The weights w_i are tuned in the training step. These coefficients control the influence of the neurons in the output. Figure 4 presents the RBFN in a typical 3-layer neural network scheme, in which the input vector is the first layer, the hidden layer contains the RBFN neurons, and the output layer (third layer) represents the linear combination of the neurons.

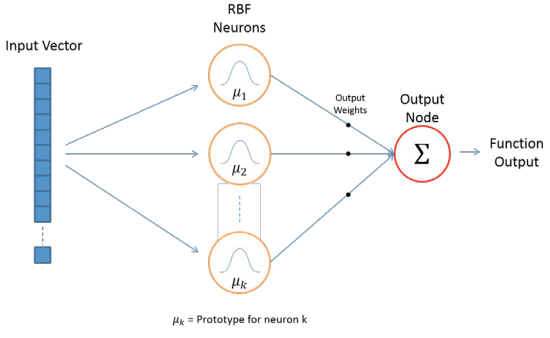

Fig. 1. RBFN as a 3-layer neural network (input vector, layer of RBF neurons and output layer. Adapted from: McCormick (2013)

Despite the appealing fact that RBFN can be classified as a neural network, it can be also seen as simple linear regressions in relation to the coefficients w_i, where the value of the unknown function $e_{rr}(u)$ is approximated by a weighted sum of n_n nonlinear functions:

$$e_{rr,j} = w_0 + w_1 \cdot r_1(u_j) + \cdots + w_{n_n} \cdot r_{n_n}(u_j) \tag{7}$$

where, the u_j and $e_{rr,j}$ are the input and residual measurement of the j^{th} operation point, respectively. A bias term, w_0, is added to the problem in order to represent the cases where all the neurons have a zero activation value. Note that, for our purposes, $e_{rr,j}$ can be either $e^C_{rr,j}$ or $e^J_{rr,j}$. Then, for all the available measurements (n_m in this case) the following matrix can be obtained:

$$\begin{bmatrix} e_{rr,1} \\ e_{rr,2} \\ \vdots \\ e_{rr,n_m} \end{bmatrix} = \begin{bmatrix} 1 & r_1(u_1) & r_2(u_1) & \dots & r_{n_n}(u_1) \\ 1 & r_1(u_2) & r_2(u_2) & \dots & r_{n_n}(u_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_1(u_{n_m}) & r_2(u_{n_m}) & \dots & r_{n_n}(u_{n_m}) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{n_m} \end{bmatrix}$$
$$\mathbf{Y} = \mathbf{R} \cdot \mathbf{W} \tag{8}$$

The coefficients can be easily estimated by a simple ordinary least square estimator, for example:

$$\mathbf{W} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}\mathbf{Y} \tag{9}$$

5. RBFN FOR ESTIMATING PLANT GRADIENTS

After training the network, $\nabla\widehat{e_{rr}}$ can be easily obtained for new values of the input $\mathbf{u}$ by taking advantage of the network structure. Here, we present the derivation for only one dimension. However, it can be easily extended to the multidimensional case. For obtaining the gradient, Equation (6) is differentiate with respect to u:

$$\begin{aligned}
\frac{\partial r_i(u)}{\partial u} =& \frac{\partial}{\partial u}\left(\frac{\mathrm{e}^{-\beta(u-\mu_i)^2}}{\sum_{l=1}^{n_n}\mathrm{e}^{-\beta(u-\mu_l)^2}}\right) \\
=& \frac{-2\beta(u-\mu_i)\mathrm{e}^{-\beta(u-\mu_i)^2}}{\sum_{l=1}^{n_n}\mathrm{e}^{-\beta(u-\mu_l)^2}} \\
& - \frac{\mathrm{e}^{-\beta(u-\mu_i)^2}(\sum_{l=1}^{n_n} -2\beta(u-\mu_l)\mathrm{e}^{-\beta(u-\mu_i)^2})}{(\sum_{l=1}^{n_n}\mathrm{e}^{-\beta(u-\mu_l)^2})^2} \\
=& \frac{-2\beta\mathrm{e}^{-\beta(u-\mu_i)^2}}{\sum_{l=1}^{n_n}\mathrm{e}^{-\beta(u-\mu_i)^2}}\left((u-\mu_i)^T - \frac{\sum_{l=1}^{n_n}(u-\mu_i)\mathrm{e}^{-\beta(u-\mu_l)^2})}{(\sum_{l=1}^{n_n}\mathrm{e}^{-\beta(u-\mu_l)^2})}\right) \\
=& -2\beta r_i(u)\left((u-\mu_i) - \sum_{l=1}^{n_n}(u-\mu_l)r_l(u)\right)
\end{aligned} \tag{10}$$

Substituting the results for every r_i, we obtain:

$$\frac{\partial e_{rr}(u)}{\partial u} = w_1\frac{\partial r_1(u)}{\partial u} + w_2\frac{\partial r_2(u)}{\partial u} + \cdots + w_k\frac{\partial r_{n_n}(u)}{\partial u} \tag{11}$$

The last equation can be written in matrix form:

$$\frac{\partial e_{rr}(u)}{\partial u} = -2\beta\left[W_p \cdot diag(M_R(u)) \cdot M(u) - (M_R(u) \cdot M(u)^T) \cdot (W_p K(u)^T)\right] \tag{12}$$

where:

$$\begin{aligned}
M(u) &= [(u-\mu_1), (u-\mu_2), \cdots, (u-\mu_{n_n})]^T \\
M_R(u) &= [r_1(u), r_2(u), \cdots, r_{n_n}(u)] \\
W_p &= [w_1, w_2, \cdots, w_{n_n}]
\end{aligned}$$

In order to improve the quality of information used to train the network, two extra steps are included in the RTO method. First, on every iteration (i.e. whenever new

steady-state information is available), the neurons are repositioned in the input space and the networks are retrained. To determine the new values of μ, the k-Means clustering strategy is chosen due to its simplicity and effectiveness (Schwenker et al., 2001). The idea is based on partitioning the n_d data points from the training data into n_n groups. The data points are assigned to the nearest cluster, which is determined based on the squared distance between the data point and the cluster center. Then, a new cluster center is computed as the average of all points inside the cluster. This process is repeated until the centers of all clusters remain equal for at least two iterations, suggesting that the algorithm has converged.

The second extra step is a probing policy, which is developed to obtain extra plant information in order to ensure that the regressor matrix ($\mathbf{R}$ in Equation (8)) contains well-distributed and sufficiently distant points from the current iterate. The idea is based on *Steps 4 and 5* of the methodology proposed by Gao et al. (2016). Two parameters Δu and n_{past} control the probing policy. The former determines the neighboring region around the current operating point, while the latter determines the number of past data points that are used for analyzing the regression region. The policy shrinks the regression region in order to improve the accuracy of the gradient estimates near the vicinity of the optimum. For the sake of brevity, the probing policy is not explained here. Please refer to the original paper for a detailed explanation.

A diagram containing a detailed description of steps of the Modifier Adaptation method using neural networks for estimating plant gradients is shown in Figure 2.

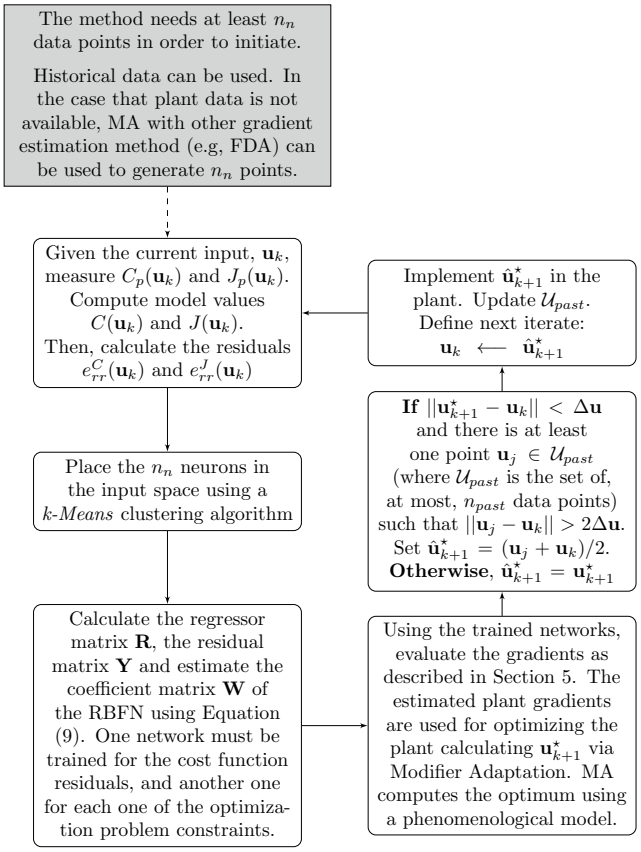

Fig. 2. MA iteration using a radial basis function network for estimating plant gradients

6. CASE STUDY: GAS LIFTED OIL WELL NETWORK

Gas lift is an artificial lift method applied in offshore oil and gas production. In cases that the reservoir pressure is low, gas is injected at the well bottom in order to economically lift the fluids from the reservoir to the surface (Krishnamoorthy et al., 2016). The gas injection reduces the fluid mixture density, which decreases the pressure at the bottom of the well, resulting in an larger inflow from the reservoir. A simplified flowsheet of the process can be seen in Figure 3.

The nonlinear steady-state model of the process used in this case study consists of mass balances for oil and gas and relations for calculating density, flow and pressure. The model considers constant temperatures, frictional pressure drop, ideal gas behavior, and simple linear relations to calculate the reservoir outlet flows. For more details to the gas lift well model, refer to Krishnamoorthy et al. (2016).

The system model is divided in three control volumes: *Wells*, each well model is composed by two section, before and after gas injection. In this case study, the network contains two wells; *Annulus*, a void between the product pipeline and the external tubing in which the lift gas is injected; and *Riser*, which encompasses a manifold that connects the wells to the main pipeline that transports the oil/gas mixture to the the surface.

The optimal operation of the 2 wells network is achieved by maximizing the profit using the system decision variables,

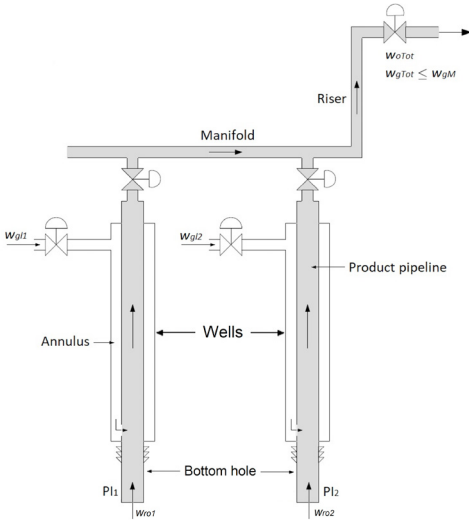

Fig. 3. Network containing two gas lifted wells. Adapted from: Krishnamoorthy et al. (2016).

which are the mass flow rate of gas lift in each well $\mathbf{u} = [w_{gl,1}, w_{gl,2}]^T$. At first, increasing the gas lift flowrate to its maximum capacity might appear as the obvious optimal solution, but larger gas injection flowrates also increase frictional pressure drop, decreasing the well production. Therefore, there is a trade-off to find the optimal gas injection. In addition, process constraints, like well total gas production capacity, need to be taken into account. Hence, the process is well suited for RTO applications.

The system optimization problem can be written as:

$$\begin{aligned} \max_{\mathbf{u}=[w_{gl,1},w_{gl,2}]^T} \quad & J := w_{oTot}^2 - 0.5\sum_{i=1}^{2} w_{gl,i}^2 \\ \text{s. t.} \quad & C := \begin{bmatrix} w_{gTot} \\ w_{gl,1} \\ w_{gl,2} \end{bmatrix} \leq \begin{bmatrix} w_{gM} \\ w_{glM} \\ w_{glM} \end{bmatrix} \end{aligned} \tag{13}$$

Here, w_{oTot} is the total oil production and w_{gTot} the total gas production of the well network; w_{gM} and w_{glM} are the maximum gas processing capacity of the system and maximum gas lift flowrate for the well, respectively. The total oil and gas production of the well are calculated using the input-output mapping, $[w_{oTot}, w_{gTot}]^T = y_{model}(w_{gl,1}, w_{gl,2})$, which represents the solution of the nonlinear steady-state model.

In order to consider plant-model mismatch, a model for the plant and a model for the optimization layer are developed. The difference between them lies in the reservoir model $w_{ro} = PI(p_r - p_{bh})$, which uses the productivity index (PI) to relate the oil flowrate leaving the reservoir, w_{ro}, with the difference between well bottom hole and reservoir pressures, (p_r - p_{bh}). The plant-model mismatch is then achieved using different values for PI ($PI_{plant} = 7$ and $PI_{model} = 5$). As a consequence, the model and the plant optimum are different and, moreover, they lie at a different set of constraints.

7. CASE STUDY AND RBFN SET-UP

In this case study, only the steady-state behavior of the plant is analyzed. Random noise was added to the measurements, the noise values are drawn from an uniform distribution in the interval of ±1% of the current measurement value. The noise is assumed to be neither correlated in time nor between measurements. The values of the model parameter values are not shown here for the sake of brevity. They can be found in Krishnamoorthy et al. (2016). Table 2 shows the tuning parameters of MA and RBFN.

Table 2. Tuning parameters

Constraint bias modifier	$K_{\epsilon,C}$	0.7
Constraint gradient modifier	$K_{\lambda,C}$	0.5
Cost function gradient modifier	$K_{\lambda,J}$	0.5
Input filter	K_u	0.4
Width of neurons	β	0.4
Number of neurons	n_n	6
Neighboring region parameter	Δ_u	0.3
Number of setpoins in $\mathcal{U}_{past}$	n_{past}	15

The first three parameters are the filters used in the zero- and first-order modifiers terms (Equation (2)), which are added to the cost and constraint function of the MA optimization (Equation (1)). Additionally, after obtaining the solution of the economic optimization problem, an input filter $\mathbf{u}_{k+1} = \mathbf{u}_k + K_u(\mathbf{u}^\star_{k+1} - \mathbf{u}_k)$ is also used for mitigating the effect of noise and errors in the gradient predictions. Small values of the K_u represent a more conservative updating strategy, for example: if the measurements are unreliable, it is better to avoid large changes in the inputs $\mathbf{u}_{k+1}$ in order to not deteriorate plant performance.

As only one constraint (the total gas production of the well network constraint) is modified in the economic optimization problem, two RBFN are trained, one for the constraint and one for the optimization problem cost function. The neurons of both RBFN have the β value shown in Table 2. β is the coefficient of Equation (5) that controls the width of the neuron. This value may be selected differently for each neuron. However, a single value was used for all the neurons in this case study, i.e. all neurons have the same width.

The number of neurons is an important parameter for the network. As the number of neurons increases, the accuracy of the networks increases on the training data, but there is also a risk of overfitting. Also, there is a trade-off in efficiency, more RBF neurons means more computational time. For our case study, a network with 6 neurons was found to achieve a good accuracy. Finally, Δu and n_{past} are parameters connect to the probing policy shown in Figure 2. They are tuned based on Matias et al. (2018), where the authors implement the methodology proposed by Gao et al. (2016) in a gas lifted oil well network.

8. RESULTS AND DISCUSSION

In order to train the neural network, an initial set of data points needs to be generated. Instead of using data from previous operation runs, the first 6 operation points are calculated as in Gao and Engell (2005), i.e. first using finite-difference approximation (FDA) and afterwards Iterative Gradient-Modification Optimization (IGMO). Then the neural network is trained on this data set. Note that in the number of initial points is chosen based on the number of neurons.

Figure 4 displays the iterations using the MA scheme with gradient estimation via RBFN and the extra plant probing points. The figure shows the contour lines of the plant profit surface and the setpoints moves from the initial point, $\mathbf{u}(0) = [1, 1]$ [kg/s], to the plant optimum. The optimization points are connected by the blue line. The extra probing points are also shown. The results confirm that the calculated optimum converged to the plant optimum in few iterations and no constraints are violated.

The final step is the analysis of the quality of gradient estimation of both profit and constraints, which can be seen in Figure 5. The figures show the gradient values for both inputs u_1 and u_2. The results show that the methodology is able to estimate the gradients accurately. They are in good agreement with the plant gradients in both cases. The deviation is relatively small if compared to the absolute value of the gradient.

9. CONCLUSION

In this work, radial-basis function network (RBFN) has been used to model the deviation between model and plant. Taking advantage of the network structure, the study provided a framework to estimate the plant gradients. Using RBFN to model plant-model mismatch was already proposed by (Johansen and Foss, 1992). However, the main contribution of the paper lies in the application of neural network to obtain the plant gradients in

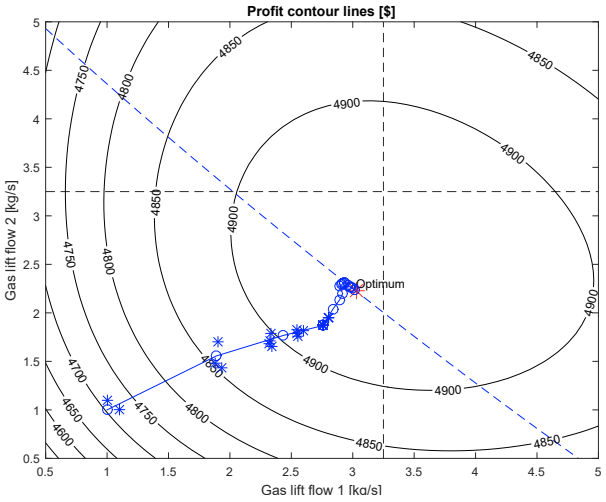

Fig. 4. Set-point moves from the optimization run and profit contour lines for the plant. Circles represent optimization steps, while asterisks represent probing steps. Dashed blue line is the maximum gas production constraint and dashed black lines are the maximum gas lift flow rate for each well.

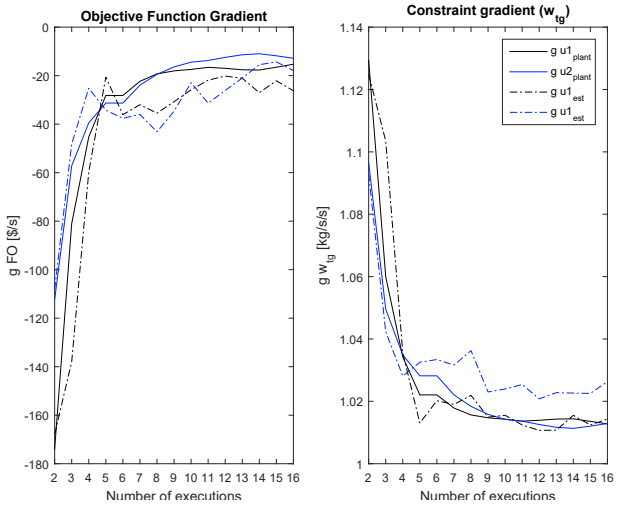

Fig. 5. Difference between the estimated gradient (based on RBFN) and the actual plant gradients.

a Modifier Adaptation (MA) scheme. The method was applied in the optimization of a gas lifted oil well network. The simulation results show that, despite the presence of plant-model mismatch and a considerable amount of noise measurement, the MA approach is able to reach the plant optimum. The cost and constraints gradients of the plant were successfully obtained using the RBFN. In addition, no constraints are violated even before convergence to the plant optimum, which is important for practical applications.

ACKNOWLEDGEMENTS

The authors would like to thank M.Sc. Dinesh Krishnamoorthy for providing the process models used in this work. The authors acknowledge Financial Support from the Norwegian research council /Intpart, SUBPRO

REFERENCES

Brdyś, M. and Tatjewski, P. (1995). An algorithm for steady-state optimizing dual control of uncertain plants. In *New Trends in Design of Control Systems 1994*, 215–220. Elsevier.

Darby, M.L., Nikolaou, M., Jones, J., and Nicholson, D. (2011). Rto: An overview and assessment of current practice. *Journal of Process Control*, 21(6), 874–884.

Gao, W. and Engell, S. (2005). Iterative set-point optimization of batch chromatography. *Computers & Chemical Engineering*, 29(6), 1401–1409.

Gao, W., Wenzel, S., and Engell, S. (2016). A reliable modifier-adaptation strategy for real-time optimization. *Computers & Chemical Engineering*, 91, 318–328.

Golden, M.P. and Ydstie, B.E. (1989). Adaptive extremum control using approximate process models. *AIChE journal*, 35(7), 1157–1169.

Johansen, T. and Foss, B. (1992). Nonlinear local model representation for adaptive systems. In *Intelligent Control and Instrumentation, 1992. SICICI'92. Proceedings., Singapore International Conference on*, volume 2, 677–682. IEEE.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2016). Real-time optimization under uncertainty applied to a gas lifted well network. *Processes*, 4(4), 52.

Mansour, M. and Ellis, J. (2003). Comparison of methods for estimating real process derivatives in on-line optimization. *Applied Mathematical Modelling*, 27(4), 275–291.

Marchetti, A., François, G., Faulwasser, T., and Bonvin, D. (2016). Modifier adaptation for real-time optimizationmethods and applications. *Processes*, 4(4), 55.

Marchetti, A., Chachuat, B., and Bonvin, D. (2009). Modifier-adaptation methodology for real-time optimization. *Industrial & engineering chemistry research*, 48(13), 6022–6033.

Matias, J.O., Le Roux, G.A., and Jäschke, J. (2018). Modifier adaptation for real-time optimization of a gas lifted well network. *IFAC-PapersOnLine*, 51(8), 31–36.

McCormick, C. (2013). Radial basis function network (rbfn) tutorial. `http://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/`. Accessed: 2018-04-27.

Rodger, E.A. and Chachuat, B. (2011). Design methodology of modifier adaptation for on-line optimization of uncertain processes. *IFAC Proceedings Volumes*, 44(1), 4113–4118.

Schwenker, F., Kestler, H.A., and Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural networks*, 14(4-5), 439–458.

Singhal, M., Marchetti, A.G., Faulwasser, T., and Bonvin, D. (2017). Improved directional derivatives for modifier-adaptation schemes. *IFAC-PapersOnLine*, 50(1), 5718–5723.