Indirect Training with Error Backpropagation in Gray-Box Neural Model: Application to a Chemical Process

Francisco Cruz Naranjo
Escuela de Informática
Universidad Andrés Bello
Santiago, Chile
fcruz@unab.cl

Gonzalo Acuña Leiva
Departamento de Ingeniería Informática
Universidad de Santiago de Chile
Santiago, Chile
gacuna@usach.cl

***Abstract*—Gray-box neural models mix differential equations, which act as white boxes, and neural networks, used as black boxes, to complete the phenomenological model. These models have been used in different researches proving their efficacy. The aim of this work is to show the training of the gray-box model through indirect backpropagation and Levenberg-Marquardt. The gray-box neural model was tested in the simulation of a chemical process in a continuous stirred tank reactor (CSTR) with 5% noise, responding successfully.**

Keywords: gray-box neural model; neural networks; time-varying parameters; chemical processes

I. INTRODUCTION

Industrial processes are highly relevant to a country, particularly for its economy, and that is why there is a need to monitor and optimize those processes. However, when dealing with complex processes such as those in which a large number of input and output variables which are also represented by nonlinear models and with parameters that vary in time, important hindrances arise.

When the processes to be modeled are complex, the determination of relevant variables or parameters for their improvement is a hard and difficult job. So there is a need to estimate the variables that cannot be measured directly. This requires a software sensor that allows observation of the variables that cannot be measured online making use of other easy to measure variables of the process [1].

An additional problem is that of a model that has parameters that vary in time, since a strategy must be adopted to identify those parameters online and in real time. A methodology that is used in these cases, especially in the field of chemical and biotechnological processes, is that of the so-called gray-box models [2], which are those that include a limited phenomenological model that is complemented with parameters obtained by means of neural networks.

Chemical processes include a large number of variables, many of which can be measured directly online, while others cannot be measured online [3], many times because there are no instruments capable of making the measurement, other times because of their high cost, or because the measurements could take several days, while the aim is to get immediate information.

Learning or training strategies used so far assume the existence of data for the parameters obtained by the neural model [1], but most of the time this is not possible.Training used in this work does not need data for the neural part, therefore working only with the data seen by the phenomenological part of the model, as detailed below. The creation of the proposed model, the learning, and the simulations were made completely in a Matlab environment, using as a basis in the work done in [4]. The contribution of the present work is to incorporate the initial pseudo-random assignment of weights, simulate the process by adding 5% of white or Gaussian noise to the data, the OSA and MPO simulation schemes, and finally incorporating quality measurement indices, all this compared to [4].

II. GRAY-BOX MODELS

Gray-box neural models are used for systems where there is some *a priori* knowledge, i.e., some physical laws are known, however some parameters must be determined from observed data.

Gray-box models, initially called hybrid neural networks or hybrid neural models, have been studied for more than a decade. The basics for a first hybrid neural network for modeling processes are developed in [5], where a comparison is made of standard neural networks and hybrid neural networks, the latter showing better performance for predicting parameters because they have *a priori* knowledge and they make use of it. They also showed that hybrid models can be identified and trained with a much smaller amount of data with respect to an equivalent black box model (neural networks).

Two years later, [6] classified hybrid models in two categories. The first is the series gray-box model (Fig. 1), which consists of a neural network that supports intermediate values which are then introduced in the phenomenological model. The second category is the parallel gray-box model (Fig. 2), which consists of a neural network that compensates the phenomenological model in the sense of modeling the error.

Then in [7] it was shown that in general the series gray-box hybrid scheme achieves better results compared to the parallel scheme, with better performance of series models for MISO (multiple input single output) systems. However, in models of SISO (single input single output) systems the results obtained have similar characteristics.

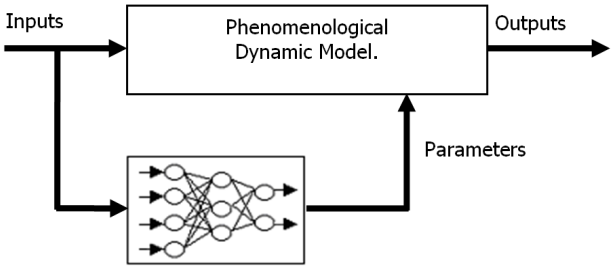

Figure 1. Series gray-box neural model. A neural network that supports intermediate values that are then introduced in the phenomenological model.

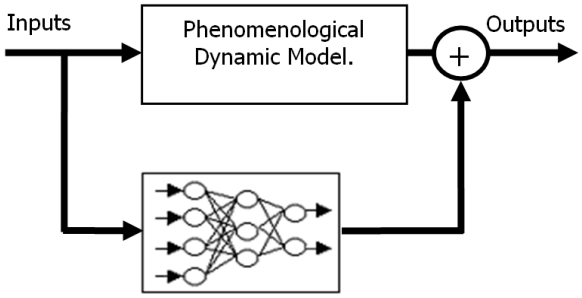

Figure 2. Parallel gray-box neural model. A neural network that compensates the phenomenological model in the sense of modeling the error.

In [8] two forms of training are distinguished. The first form corresponds to direct training (Fig. 3), which uses the error originated at the output of the neural network for the correct determination of their weights. The second form is indirect training (Fig. 4), which uses the error originated at the model's output for learning purposes of the neural network. Indirect training can be made in two forms, one by minimizing an objective function of state variables by means of a nonlinear optimization technique, and the other by backpropagating the output error in the weights of the neural network through the equations of the phenomenological model.

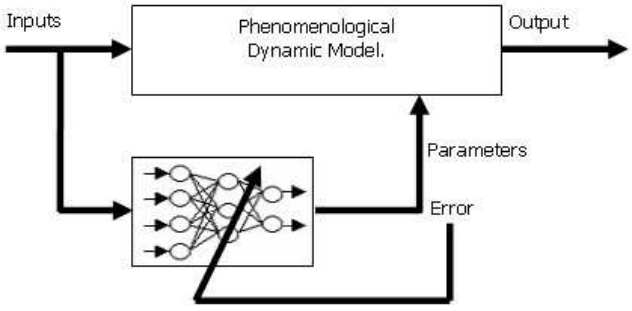

Figure 3. Direct training gray-box model. It uses the error originated at the neural network output for the correct determination of its weight.

In [9] an application based on Matlab is used for developing gray-box neural models using a direct training strategy. This work has the purpose of carrying out a Matlab implementation of a training algorithm based on Levenberg-Marquardt with indirect strategy, backpropagating the error of the model through the phenomenological model to the neural network.

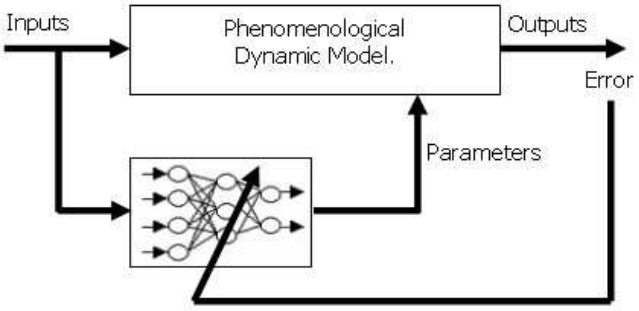

Figure 4. Gray-box model with indirect training. It uses the error originated at the model's output for the neural network learning

III. CSTR PROCESS

The process corresponds to the simulation of the first order exothermic reaction in a Continuous Stirred Tank Reactor (CSTR) [10]. The system's input corresponds to the temperature of the cooling sleeve and the system's output corresponds to the degree of progress of the reaction. The state equations that describe the system are given below, equations (1), (2) and (3).

$$x'_1 = -x_1 + D_a \cdot (1 - x_1) \cdot e^{\left(\frac{x_2}{1 + x_2 / \gamma} \right)} \tag{1}$$

$$x'_2 = -x_2 + B \cdot D_a \cdot (1 - x_1) \cdot e^{\left(\frac{x_2}{1 + x_2 / \gamma} \right)} + \beta \cdot (\mu - x_2) \tag{2}$$

$$y = x_1 \tag{3}$$

where:

x_1 : Degree of progress of the reaction.

x_2 : Nondimensional temperature of the reactor's content.

μ : Input that corresponds to a nondimensional flow rate of the heat transfer fluid through the cooling sleeve.

To carry out the simulation the following values were used for the model's constants:

$$D_a = 0,072$$
$$\mathrm{B} = 8,0$$
$$\beta = 0,3$$
$$\gamma = 20,0$$

The gray-box neural models consist of two parts, the first composed of a phenomenological model represented by differential equations and the second composed of an empirical model represented by a neural network. The gray-box phenomenological model is represented by equations (4), (5) and (6), where ρ corresponds to the parameter that is hard to model, which will be estimated by the neural network.

$$x'_1 = -x_1 + 0,072 \cdot (1 - x_1) \cdot \rho \tag{4}$$

$$x'_2 = -x_2 + 8 \cdot 0,072 \cdot (1 - x_1) \cdot \rho + 0,3 \cdot (\mu - x_2) \tag{5}$$

$$y = x_1 \tag{6}$$

However, for experimental and data generation purposes, the expression shown in equation (7) will be used.

$$\rho = e^{\left(\frac{x_2}{1 + x_2 / 20} \right)} \tag{7}$$

Therefore, the parameter is assumed to be completely unknown and will be estimated using the indirectly trained neural network. Then, integrating equations (4) and (5) in the time interval between t and t+1, the discrete equations (8) and (9) are obtained.

$$x_{1(t+1)} = x_{1(t)} + \left(-x_{1(t)} + 0,072 \cdot \left(1 - x_{1(t)} \right) \cdot \rho \right) \cdot \Delta t \tag{8}$$

$$x_{2(t+1)} = x_{2(t)} + (-x_{2(t)} + 8 \cdot 0,072 \cdot \left(1 - x_{1(t)} \right) \cdot \rho + 0,3 \cdot \left(u - x_{2(t)} \right)) \cdot \Delta t \tag{9}$$

Developing equations (8) and (9) for the construction of the gray-box neural model, the phenomenological part is represented by equations (10), (11) and (12).

$$x_{1(t+1)} = (1 - \Delta t) x_{1(t)} + 0,072 \rho \Delta t - 0,072 \rho \Delta t x_{1(t)} \tag{10}$$

$$x_{2(t+1)} = (1 - 1.3\Delta t) x_{2(t)} + 8 \cdot 0,072 \rho \Delta t - 8 \cdot 0,072 \rho \Delta t x_{1(t)} + 0,3 u \Delta t \tag{11}$$

$$y = x_1 \tag{12}$$

IV. Proposed Solution

The proposed solution is a gray-box neural model whose phenomenological part can be described together with its empirical part, for which a neural network that contains both parts of the gray-box model is designed (Fig. 5). This hybrid neural network has the ability of setting weights in the training stage, so that it can act as a gray-box model. The weights in Fig. 5 that have a given value correspond to the model's phenomenological part, and in that way the weights will not be modified during the training. The weights for which no values are given correspond to the neural part of the model, and it is also seen that they present bias connections. These weights were initiallly assigned with pseudorandom values obtained by the initialization method of Nguyen-Widrow [11].

In Fig. 5 it is clearly seen that the neural part of the model is estimated by a multilayer perceptron that is inserted in the gray-box neural model, which has a 1-4-1 architecture. Its input corresponds only to X_2 because, as seen in the mathematical model, the parameter that is hard to obtain depends only on this state variable.

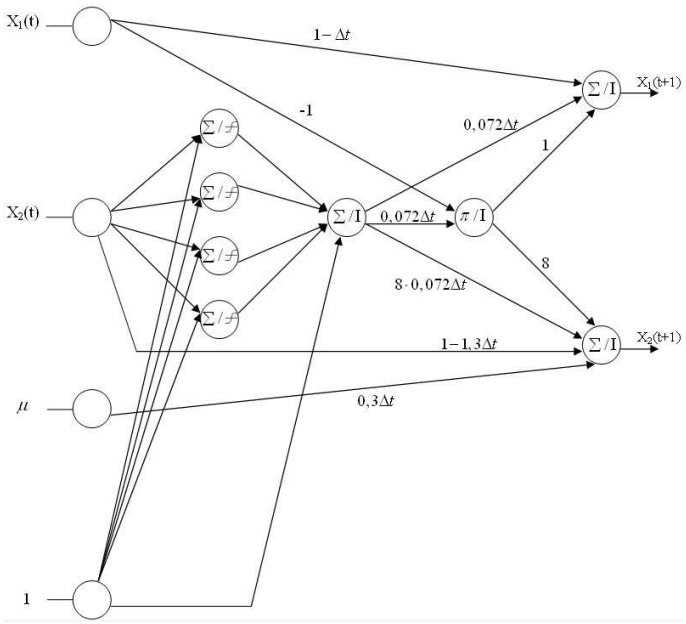

Figure 5. Gray-box neural model for the simulation of the CSTR process with fixed weights.

Consequently, the multilayer perceptron estimates the value of the hard to obtain parameter, which in turn is mixed with the phenomenological part of the model, which in that way get its output.

For the neural part of the model hyperbolic tangent was used as transfer function in the intermediate layer and the identity function was used for the output layer, while for the phenomenological part the identity function was used as transfer function. The activation function used was the summation of the weighted inputs, except for the neuron immediately after the output of the neural part, where a product was used, due to the form of the phenomenological equations.

For the validation of the proposed gray-box neural model quality indices such as the IA (index of agreement), RMS (root mean square) and RSD (residual standard deviation) were calculated; the values considered acceptable for these indices are IA > 0.9, RMS < 0.1 and RSD < 0.1. The quality indices are described in equation (13),
where o_i and p_i are the observed and predicted values, respectively, in time i, and N is the total number of data. Therefore, $p_i'=p_i-o_m$ and $o_i'=o_i-o_m$, where o_m is the mean value of the observations.

V. TRAINING

In the network the root mean squared is used as a performance function, and it is calculated at the output of the phenomenological model, from where it is backpropagated toward the weights of the network that are modifiable. In the training phase the input used is a noisy signal (Fig. 6), while in the simulation stage the input used is a sinusoidal signal (Fig. 7).

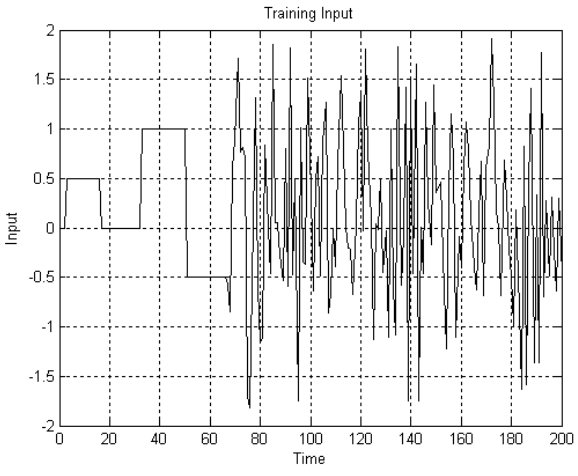

Figure 6. Input used in the gray-box neural model for the training stage.

The training algorithm corresponds to the Levenberg-Marquardt backpropagation method, which is a second order algorithm that presents a slight modification of the traditional Newton method. The algorithm also has the ability to modify only the weights that are indicated, leaving a group of fixed weights in the training phase which, as already mentioned, represent the model's phenomenological part.

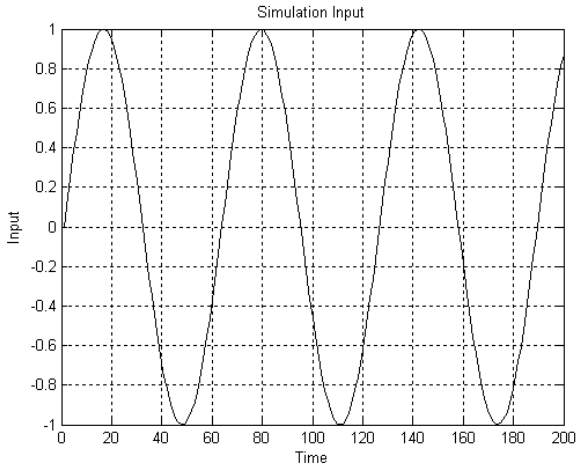

Figure 7. Input used in the gray-box neural model for the simulation stage.

This training scheme corresponds to the second indirect training method proposed by [8], in which the error at the output of the gray-box model is calculated and from there it is backpropagated to the empirical or neural part of the model. This has the advantage that it is not necessary to have measurements of the hard-to-obtain parameter.

The hybrid neural network or the gray-box model were trained for 1000 epochs, several times, getting around 500 different trainings, and then the best one of them was selected, using as a selection criterion the mean squared error obtained at the model's output. Then, in the simulation phase, the IA, RMS and RSD error indices are calculated. It is important to note that the number of training epochs can increase in this type of model, because having fixed weights is a restriction that is added to the optimization algorithm.

$$IA = 1 - \frac{\sum_{i=1}^{n}(o_i - p_i)^2}{\sum_{i=1}^{n}\left(|o_i'| + |p_i'|\right)^2} \quad RMS = \sqrt{\frac{\sum_{i=1}^{n}(o_i - p_i)^2}{\sum_{i=1}^{n} o_i^2}} \quad RSD = \sqrt{\frac{\sum_{i=1}^{n}(o_i - p_i)^2}{N}} \tag{13}$$

VI. SIMULATION AND RESULTS

For the simulation, tests were made for data with 5% error. The simulation was made for the 200 available data. The Δt value used for the simulation was 0.005, while the initial values for the state variables were $X_1 = 0.1$ and $X_2 = 0.8$. The simulation was made in an MPO (multiple predictive output) scheme, i.e., from the initial state vector all the values are backfed to the input.

Fig. 8 shows the response of the network to the OSA (one step ahead) simulation. The solid line shows the system's real output and the dashed line shows the output estimated by the model. Both outputs agree, so the model responds satisfactorily.

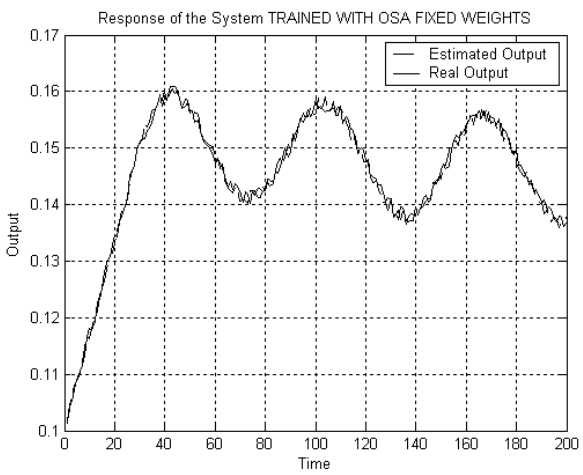

Figure 8. Response of the trained model with fixed weights, and simulated with 5% of OSA noise.

Fig. 9 shows the MPO simulation. The solid line shows the sytem's real output and the dashed line shows the simulated output. In this case the estimated output does not present noise, acting as a filter. This happens because the MPO simulations consider only the initial vector and the model is iterated from it to perform all the simulations.

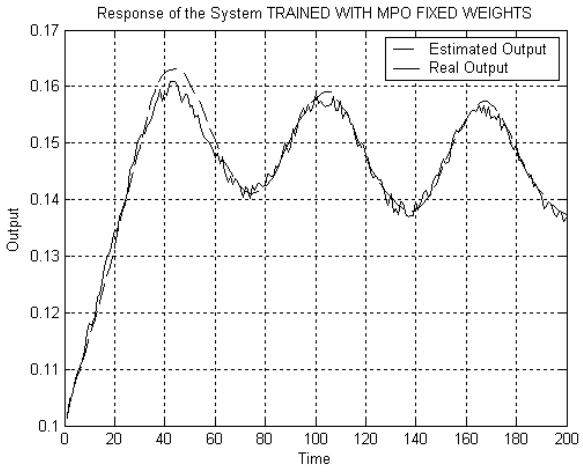

Figure 9. Response of the trained model with fixed weights, and simulated with 5% MPO noise.

To validate the estimated output with respect to the system's real output with noise, the quality indices were used, which are shown in Table 1. All the resultant quality indices of these simulations are within acceptable ranges, so the gray-box neural model fits adequately the desired output for the case of 5% noise in the data.

VII. CONCLUSIONS

Gray-box neural models are a real alternative for modeling real world processes. They have advantages over black box models, because they are supported by the *a priori* knowledge available on the process.

TABLE I. QUALITY INDICES OF THE ESTIMATED OUTPUT, FOR OSA AND MPO, SIMULATED WITH 5% NOISE.

	OSA	**MPO**
IA	0.9975	0.9948
RMS	0.0080	0.0118
RSD	0.0012	0.0017

The model proposed in this paper combines the phenomenological knowledge of the process with a neural network of the multilayer perceptron type for the estimation of the parameter within a higher order neural network to carry out the backpropagation process.

The indirect training method with backpropagation and Levenberg-Marquardt used in this paper shows good results in the learning phase and later simulation only with the measurable variables as software sensors.

ACKNOWLEDGEMENTS

The authors acknowledge the financial support of FONDECYT under Project 1090316.

REFERENCES

[1] James, S., Legge, R., Budman, H., Comparative study of black-box and hybrid estimation methods in fed-batch fermentation. Journal of Process Control 12, 113-121 (2000)

[2] Hensen, R., Angelis, G., van de Molengraft, M., de Jager, A., Kok, J., Gray-box modeling of friction: An experimental case-study. European Journal of Control 6, 258-267 (2000)

[3] Chen, L., Bernard, O., Bastin, G., Angelov, P., Hybrid modelling of biotechnological processes using neural networks. Control Engineering Practice 8, pp. 821-827 (2000)

[4] Cruz, F., Acuña, G., Entrenamiento indirecto de modelos de caja gris utilizando Levenberg-Marquardt en Matlab®, Actas en CD de las XIII Jornadas Chilenas de Computación, Universidad Austral de Chile, Valdivia (2005)

[5] Psichogios, D., Ungar, L., A Hybrid Neural Network-First Principles Approach to Process Modeling, Vol. 38, Nº10, 1499-1511 (1992)

[6] Thompson, M.; Kramer, M., Modeling Chemical Processes Using Prior Knowledge and Neural Networks, Vol. 40, Nº8, 1328-1340 (1994)

[7] Van Can, H., Hellinga, C., Luyben, K., Heijnen, J., Strategy for Dynamic Process Modeling Based on Neural Network in Macroscopic Balance", AIChE 42, 3403-3418 (1996)

[8] Acuña, G., Cubillos, F., Thibault, J., Latrille, E., Comparison of methods for training gray-box neural network models, Computers and Chemicals Engineering Supplement, 561-564 (1999)

[9] Pinto, E., Acuña, G., Desarrollo de un Toolbox Matlab para la elaboración de modelos neuronales de caja gris. Actas en CD del XVI Congreso de la Asociación Chilena de Control Automático, Universidad de las Américas, Santiago (2004)

[10] Hernández, E.; Arkun, Y., Study of the Control-Relevant Properties of Backpropagation Neural Network Models of Nonlinear Dynamical Systems, Computers Chemical Engineering, Vol. 16, n° 4, 227-240 (1992)

[11] Nguyen, D., Widrow, B., Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. Proceedings IEEE IJCNN, 21-26, San Diego (1990)