Time Accounting Artificial Neural Networks for Biochemical Process Models

Petia Georgieva, Luis Alberto Paz Suárez, and Sebastião Feyo de Azevedo

Abstract. This paper is focused on developing more efficient computational schemes for modeling in biochemical processes. A theoretical framework for estimation of process kinetic rates based on different temporal (time accounting) Artificial Neural Network (ANN) architectures is introduced. Three ANNs that explicitly consider temporal aspects of modeling are exemplified: i) Recurrent Neural Network (RNN) with global feedback (from the network output to the network input); ii) Time Lagged Feedforward Neural Network (TLFN) and iii) Reservoir Computing Network (RCN). Crystallization growth rate estimation is the benchmark for testing the methodology. The proposed hybrid (dynamical ANN & analytical submodel) schemes are promising modeling framework when the process is strongly nonlinear and particularly when input-output data is the only information available.

1 Introduction

The dynamics of chemical and biochemical processes are usually described by mass and energy balance differential equations. These equations combine two elements, the phenomena of conversion of one reaction component into another (i.e. the reaction kinetics) and the transport dynamics of the components through the reactor. The identification of such mathematical models from experimental input/output data is still a challenging issue due to the inherent nonlinearity and complexity of this class of processes (for example polymerization or fermentation

Petia Georgieva
Signal Processing Lab, IEETA, DETI
University of Aveiro, 3810-193 Aveiro, Portugal
e-mail: petia@ua.pt

Luis Alberto Paz Suárez · Sebastião Feyo de Azevedo
Department of Chemical Engineering, Faculty of Engineering,
University of Porto, 4200-465 Porto, Portugal
e-mail: sfeyo@fe.up.pt

V. Sgurev et al. (Eds.): Intelligent Systems: From Theory to Practice, SCI 299, pp. 181–199.
springerlink.com

reactors, distillation columns, biological waste water treatment, ect.) The most difficult problem is how to model the reaction kinetics and more particularly, the reaction rates. The traditional way is to estimate the reaction rates in the form of analytical expressions, Bastin and Dochain 1990. First, the parameterized structure of the reaction rate is determined based on data obtained by specially designed experiments. Then the respective parameters of this structure are estimated. Reliable parameter estimation is only possible if the proposed model structure is correct and theoretically identifiable, Wolter and Pronzato 1997. Therefore the reaction rate analytical structure is usually determined after a huge number of expensive laboratory experiments. It is further assumed that the initial values of the identified parameters are close to the real process parameters, Noykove et al. 2002, which is typically satisfied only for well known processes. The above considerations motivated a search for alternative estimation solutions based on computationally more attractive paradigms as are the Artificial Neural Networks (ANNs). The interest in ANNs as dynamical system models is nowadays increasing due to their good non-linear time-varying input-output mapping properties. The balanced network structure (parallel nodes in sequential layers) and the non-linear transfer function associated with each hidden and output nodes allows ANNs to approximate highly non-linear relationships without a priori assumption. Moreover, while other regression techniques assume a functional form, ANNs allow the data to define the functional form. Therefore, ANNs are generally believed to be more powerful than many other nonlinear modeling techniques.

The objective of this work is to define a computationally efficient framework to overcome difficulties related with poorly known kinetics mechanistic descriptors of biochemical processes. Our main contribution is the analytical formulation of a modeling procedure based on time accounting artificial neural networks (ANNs), for kinetic rates estimation. A hybrid (ANN & phenomenological) model and a procedure for ANN supervised training when target outputs are not available are proposed. The concept is illustrated on a sugar crystallization case study where the hybrid model outperforms the traditional empirical expression for the crystal growth rate.

The paper is organized as follows. In the next section a hybrid model of a general chemical or biochemical process is introduced, where a time accounting ANN is assumed to model the process kinetic rates in the framework of a nonlinear state space analytical process model. In section 3 three temporal ANN structures are discussed.

In section 4 a systematic ANN training procedure is formulated assuming that all kinetics coefficients are available but not all process states are measured. The proposed methodology is illustrated in section 5 for crystallization growth rate estimation.

2 Knowledge Based Hybrid Models

The generic class of reaction systems can be described by the following equations, Bastin and Dochain 1990

$$\frac{dX}{dt} = K\varphi(X,T) - DX + U_x \tag{1}$$

$$\frac{dT}{dt} = b\varphi(X,T) - d_0 T + U_T \tag{2}$$

where, for $n, m \in$ N , the constants and variables denote

$X = (x_1(t),.....x_n(t)) \in R^n$	Concentrations of total amounts of *n* process components
$K = [k_1,.....k_m] \in R^{n\times m}$	kinetics coefficients (yield, stoichiometric, or other)
$\varphi = (\varphi_1,.....\varphi_m)^T \in R^m$	Process kinetic rates
T	Temperature
$b \in R^m$	Energy related parameters
q_{in} / V	Feeding flow/Volume
D	Dilution rate
d_o	Heat transfer rate related parameter

U_x and U_T are the inputs by which the process is controlled to follow a desired dynamical behavior. The nonlinear state-space model (1) proved to be the most suitable form of representing several industrial processes as crystallization and precipitation, polymerization reactors, distillation columns, biochemical fermentation and biological systems. Vector (φ) defines the rate of mass consumption or production of components. It is usually time varying and dependent of the stage of the process. In the specific case of reaction process systems φ represents the reaction rate vector typical for chemical or biochemical reactions that take place in several processes, such as polymerization, fermentation, biological waste water treatment, etc. In nonreaction processes as for example crystallization and precipitation, φ represents the growth or decay rates of chemical species. In both cases (reaction or nonreaction systems) φ models the process kinetics and is the key factor for reliable description of the components concentrations. In this work, instead of an exhaustive search for the most appropriate parameterized reaction rate structure, three temporal (time accounting) ANN architectures are applied to estimate the vector of kinetic rates. The ANN sub-model is incorporated in the general dynamical model (1) and the mixed structure is termed knowledge-based hybrid model (KBHM), see Fig.1. A systematic procedure for ANN-based estimation of reaction rates is discussed in the next section.

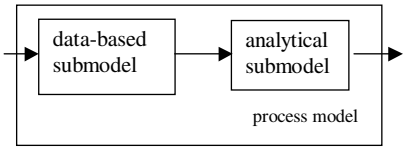

Fig. 1 Knowledge-based hybrid model (KBHM)

3 Time Accounting Artificial Neural Networks

The Artificial Neural Network (ANN) is a computational structure inspired by the neurobiology. An ANN is characterized by its architecture (the network topology and pattern of connections between the nodes), the method of determining the connection weights, and the activation functions that employs. The multi-layer perceptron (MLP), which constitute the most widely used network architecture, is composed of a hierarchy of processing units organized in a parallel-series sets of neurons and layers. The information flow in the network is restricted to only one direction from the input to the output, therefore a MLP is also called Feedforward Neural Network (FNN). FNNs have been extensively used to solve static problems as classification, feature extraction, pattern recognition. In contrast to the FNN, the Recurrent Neural Network (RNN) processes the information in both (feedforward and feedback) directions due to the recurrent relation between network outputs and inputs, Mandic and Chambers 2001. Thus the RNN can encode and learn time dependent (past and current) information which is interpreted as memory. This paper specifically focuses on comparison of three different types of RNNs, namely, i) RNN with global feedback (from the network output to the network input); ii) Time lagged feedforward neural network (TLFN), and iii) Reservoir Computing Network (RCN).

Recurrent Neural Network (RNN) with global feedback
An example of RNN architecture where past network outputs are fed back as inputs is depicted in Fig. 2. It is similar to Nonlinear Autoregressive Moving Average with eXogenios input (NARMAX) filters, Haykin 1999.The complete RNN input consists of two vectors formed by present and past network exogenous inputs (r) and past fed back network outputs (p) respectively.

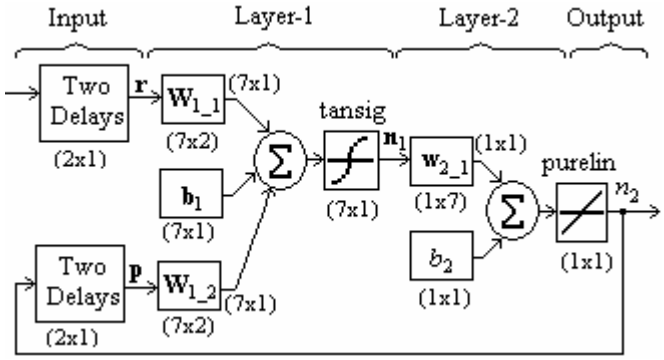

Fig. 2 RNN architecture

The RNN model implemented in this work is the following

$$\mathbf{u}_{NN} = [\mathbf{r}, \mathbf{p}] \text{ (complete network input)} \tag{3}$$

$$\mathbf{r} = [r_1(k),....r_1(k-l),.....r_c(k),....r_c(k-l)] \text{ (network exogenous inputs)} \tag{4}$$

$$\mathbf{p} = \left[n_2(k-1),....n_2(k-h)\right] \quad \text{(recurrent network inputs)} \tag{5}$$

$$\mathbf{x} = \mathbf{W}_{11} \cdot \mathbf{r} + \mathbf{W}_{12} \cdot \mathbf{p} + \mathbf{b}_1 \quad \text{(network states)} \tag{6}$$

$$\mathbf{n}_1 = \left(e^{\mathbf{x}} - e^{-\mathbf{x}}\right)/\left(e^{\mathbf{x}} + e^{-\mathbf{x}}\right) \quad \text{(hidden layer output)} \tag{7}$$

$$n_2 = \mathbf{w}_{21} \cdot \mathbf{n}_1 + b_2 \quad \text{(network output)}, \tag{8}$$

where $\mathbf{W}_{11} \in R^{m\times 2}$, $\mathbf{W}_{12} \in R^{m\times 2}$, $\mathbf{w}_{21} \in R^{1\times m}$, $\mathbf{b}_1 \in R^{m\times 1}$, $b_2 \in R$ are the network weights (in matrix form) to be adjusted during the ANN training, m is the number of nodes in the hidden layer. l is the number of past exogenous input samples and h is the number of past network output samples fed back to the input. The RNNs are a powerful technique for nonlinear dynamical system modeling, however their main disadvantage is that they are difficult to train and stabilize. Due to the simultaneous spatial (network layers) and temporal (past values) aspects of the optimization, the static Backpropagation (BP) learning method has to be substituted by the Backpropagation through time (BPTT) learning. BPTT is a complex and costly training method, which does not guarantee convergence and often is very time consuming, Mandic and Chambers 2001.

Time lagged feedforward neural network (TLFN)
TLFN is a dynamical system with a feedforward topology. The dynamic part is a linear memory, Principe et al. 2000. TLFN can be obtained by replacing the neurons in the input layer of an MLP with a memory structure, which is sometimes called a tap delay-line (see Fig. 3). The size of the memory layer (the tap delay) depends on the number of past samples that are needed to describe the input characteristics in time and it has to be determined on a case-by-case basis. When the memory is at the input the TLFN is also called Focused Time delay Neural Network (TDNN). There are other TLFN topologies where the memory is not focused only at the input but can be distributed over the next network layers. The main advantage of the TDNN is that it can be trained with the static BP method.

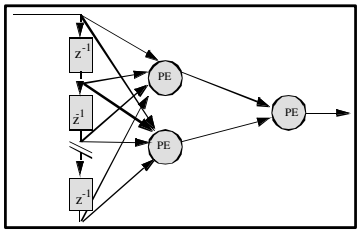

Fig. 3 RNN architecture

Reservoir Computing Network (RCN).
RCN is a concept in the field of machine learning that was introduced independently in three similar descriptions, namely, Echo State Networks (Jaeger 2001), Liquid State Machines (Maass et al. 2002) and Backpropagation-Decorelation

learning rule (Steil 2004). All three techniques are characterized by having a fixed hidden layer usually with randomly chosen weights that is used as a reservoir of rich dynamics and a linear output layer (termed also readout layer), which maps the reservoir states to the desired outputs (see Fig. 4). Only the output layer is trained on the response to input signals, while the reservoir is left unchanged (except when making a reservoir re-initialization). The concepts behind RCN are similar to ideas from both kernel methods and RNN theory. Much like a kernel, the reservoir projects the input signals into a higher dimensional space (in this case the state space of the reservoir), where a linear regression can be performed. On the other hand, due to the recurrent delayed connections inside the hidden layer, the reservoir has a form of a short term memory, called the fading memory which allows temporal information to be stored in the reservoir. The general state update equation for the nodes in the reservoir and the readout output equation are as follows:

$$x(k+1) = f\left(W_{res}^{res} x(k) + W_{inp}^{res} u(k) + W_{out}^{res} y(k) + W_{bias}^{res}\right) \tag{9}$$

$$y(k+1) = W_{res}^{out} x(k+1) + W_{inp}^{out} u(k) + W_{out}^{outr} y(k) + W_{bias}^{out} \tag{10}$$

Where: *u(k)* denotes the input at time *k*; *x(k)* represents the reservoir state; *y(k)* is the output; and *f* () is the activation function (with the hyperbolic tangent *tanh*() as the most common type of activation function). The initial state is usually set to *x(0)*=0. All weight matrices to the reservoir (denoted as W^{res}) are initialized randomly , while all connections to the output (denoted as W^{out}) are trained. In the general state update equation (3), it is assumed a feedback not only between the reservoir neurons expressed by the term $W_{res}^{res} x(k)$, but also a feedback from the output to the reservoir accounted by $W_{out}^{res} y(k)$. The first feedback is considered as the short-term memory while the second one as a very long term memory. In order to simplify the computations Following the idea of Antonelo et al. 2007, for the present study the second feedback is discarded and a scaling factor α is introduced in the state update equation

$$x(k+1) = f\left((1-\alpha)x(k) + \alpha W_{res}^{res} x(k) + W_{inp}^{res} u(k) + W_{bias}^{res}\right) \tag{11}$$

Parameter α serves as a way to tune the dynamics of the reservoir and improve its performance. The value of α can be chosen empirically or by an optimization. The output calculations are also simplified (Antonelo et al. 2007) assuming no direct connections from input to output or connections from output to output

$$y(k+1) = W_{res}^{out} x(k+1) + W_{bias}^{out} \tag{12}$$

Each element of the connection matrix W_{res}^{res} is drawn from a normal distribution with mean 0 and variance 1. The randomly created matrix is rescaled so that the spectral radius $|\lambda_{\max}|$ (the largest absolute eigenvalue) is smaller than 1. Standard settings of $|\lambda_{\max}|$ lie in a range between 0.7 and 0.98. Once the reservoir topology is

set and the weights are assigned, the reservoir is simulated and optimized on the training data set. It is usually done by linear regression (least squares method) or ridge regression, Bishop 2006. Since the output layer is linear, regularization can be easily applied by adding a small amount of Gaussian noise to the RCN response.

The main advantage of RCN is that it overcomes many of the problems of traditional RNN training such as slow convergence, high computational requirements and complexity. The computational efforts for training are related to computing the transpose of a matrix or matrix inversion. Once trained, the resulting RCN-based system can be used for real time operation on moderate hardware since the computations are very fast (only matrix multiplications of small matrices).

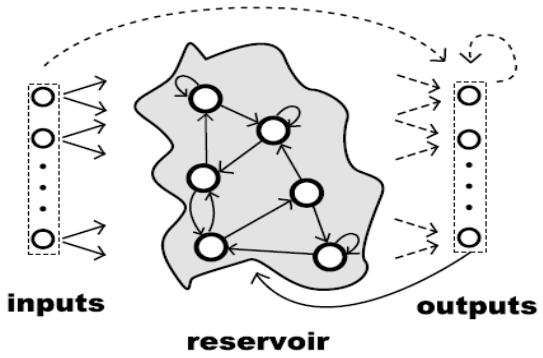

Fig. 4 Reservoir Computing (RC) network with fixed connections (solid lines) and adaptable connections (dashed lines)

4 Kinetic Rates Estimation by Time Accounting ANN

The ANNs are a data-based modeling technique where during an optimization procedure (termed also learning) the network parameters (the weights) are updated based on error correction principle. At each iteration, the error between the network output and the corresponding reference has to be computed and the weights are changed as a function of this error. This principle is also known as supervised learning. However, the process kinetic rates are usually not measured variables, therefore targets (references) are not available and the application of any data-based modeling technique is questionable. A procedure is proposed in the present work to solve this problem. The idea is to propagate the ANN output through a fixed partial analytical model (*Anal. model*) until it comes to a measured process variable (see Fig.5). The proper choice of this *Anal. model* and the formulation of the error signal for network updating are discussed below. The procedure is based on the following assumptions:

(A1) Not all process states of model (1) are measured.
(A2) All kinetics coefficients are known, that is b and all entries of matrix K are available.

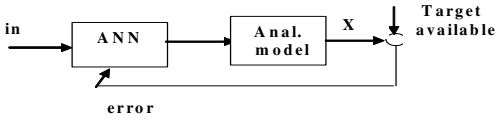

Fig. 5 Hybrid ANN training structure

For more convenience, the model (1) is reformulated based on the following augmented vectors

$$X_{aug} = \begin{bmatrix} X \\ T \end{bmatrix}, \qquad X_{aug} \in R^{n+1}, \qquad K_{aug} = \begin{bmatrix} K \\ b \end{bmatrix}, \qquad K_{aug} \in R^{(n+1)\times m}. \tag{13}$$

Then (1) is rewritten as

$$\frac{dX_{aug}}{dt} = K_{aug}\varphi\left(X_{aug}\right) - \overline{D}X_{aug} + U \quad \text{with} \quad \overline{D} = \begin{bmatrix} D & 0 \\ 0 & d_0 \end{bmatrix}, \quad U = \begin{bmatrix} U_x \\ U_T \end{bmatrix} \tag{14}$$

Step 1: State vector partition A

The general dynamical model (8) represents a particular class of nonlinear state-space models. The nonlinearity lies in the kinetics rates $\varphi\left(X_{aug}\right)$ that are nonlinear functions of the state variables X_{aug}. These functions enter the model in the form $K_{aug}\varphi\left(X_{aug}\right)$, where K_{aug} is a constant matrix, which is a set of linear combinations of the same nonlinear functions $\varphi_1(X_{aug}),.....\varphi_m(X_{aug})$. This particular structure can be exploited to separate the nonlinear part from the linear part of the model by a suitable linear state transformation. More precisely, the following nonsingular partition is chosen, Chen and Bastin 1996.

$$LK_{aug} = \begin{bmatrix} K_a \\ K_b \end{bmatrix}, \; rank\left(K_{aug}\right) = l, \tag{15}$$

where $L \in R^{n\times n}$ is a quadratic permutation matrix, K_a is a *lxm* full row rank submatrix of K_{aug} and $K_b \in R^{(n-l)\times m}$. The induced partitions of vectors X_{aug} and U are

$$LX_{aug} = \begin{bmatrix} X_a \\ X_b \end{bmatrix}, \quad LU = \begin{bmatrix} U_a \\ U_b \end{bmatrix}, \tag{16}$$

with $X_a \in R^l$, $U_a \in R^l$, $X_b \in R^{n-l}$, $U_b \in R^{n-l}$.

According to (9), model (8) is also partitioned into two submodels

$$\frac{dX_a}{dt} = K_a\varphi(X_a, X_b) - \overline{D}X_a + U_a \tag{17}$$

$$\frac{dX_b}{dt} = K_b\varphi(X_a, X_b) - \overline{D}X_b + U_b \tag{18}$$

Based on (9), a new vector $Z \in R^{n+1-l}$ is defined as a linear combination of the state variables

$$Z = A_0 X_a + X_b, \tag{19}$$

where matrix $A_0 \in R^{(n+1-l)\times l}$ is the unique solution of

$$A_0 K_a + K_b = 0, \tag{20}$$

that is

$$A_0 = -K_b K_a^{-1}, \tag{21}$$

Note that, a solution for A_0 exist if and only if K_a is not singular. Hence, a necessary and sufficient condition for the existence of a desired partition (9), is that K_a is a *pxm* full rank matrix, which was the initial assumption. Then, the first derivative of vector Z is

$$\begin{aligned}\frac{dZ}{dt} &= A_0\frac{dX_a}{dt} + \frac{dX_b}{dt} \\ &= A_0\left[K_a\varphi(X_a, X_b) - \overline{D}X_a + U_a\right] + K_b\varphi(X_a, X_b) - \overline{D}X_b + U_b \\ &= (A_0K_a + K_b)\varphi(X_a, X_b) - \overline{D}(A_0X_a + X_b) + A_0U_a + U_b\end{aligned} \tag{22}$$

Since matrix A_0 is chosen such that eq. (13) holds, the term in (15) related with φ is cancelled and we get

$$\frac{dZ}{dt} = -\overline{D}Z + A_0U_a + U_b \tag{23}$$

The state partition A results in a vector Z whose dynamics, given by eq. (15), is independent of the kinetic rate vector φ. In general, (9) is not an unique partition and for any particular case a number of choices are possible.

Step 2: State vector partition B (measured & unmeasured states)

Now a new state partition is defined as sub-vectors of measured and unmeasured states X_1, X_2, respectively. The model (8) is also partitioned into two submodels

$$\frac{dX_1}{dt} = K_1\varphi(X_1, X_2) - \overline{D}X_1 + U_1 \tag{24}$$

$$\frac{dX_2}{dt} = K_2\varphi(X_1, X_2) - \overline{D}X_2 + U_2 \tag{25}$$

From state partitions A and B, vector Z can be represented in the following way

$$Z = A_0 X_a + X_b = A_1 X_1 + A_2 X_2 . \tag{26}$$

The first representation is defined in (12), then applying linear algebra transformations A_1 and A_2 are computed to fit the equality (18). **The purpose of state partitions A and B is to estimate the unmeasured states (vector X_2) independently of the kinetics rates (vector φ).** The recovery of X_2 is defined as state observer .

Step 3: State observer

Based on (16) and starting with known initial conditions, Z can be estimated as follow (in this work the estimations are denoted by hat)

$$\frac{d\hat{Z}}{dt} = -D\hat{Z} + A_0(F_{in\_a} - F_{out\_a}) + (F_{in\_b} - F_{out\_b}) \tag{27}$$

Then according to (18) the unmeasured states X_2 are recovered as

$$\hat{X}_2 = A_2^{-1}(\hat{Z} - A_1 X_1) \tag{28}$$

Note that, estimates $\hat{X}_2$ exist if and only if A_2 is not singular, Bastin and Dochain 1990. Hence, **a necessary and sufficient condition for observability of the unmeasured states is that A_2 is a full rank matrix.**

Step 4: Error signal for NN training

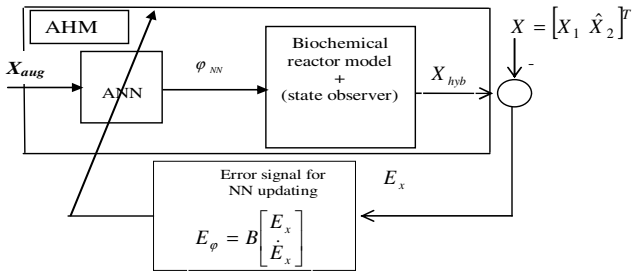

Fig. 3 Hybrid NN-based reaction rates identification structure

The hybrid structure for NN training is shown in Fig. 3, where the adaptive hybrid model (AHM) is formulated as

$$\frac{dX_{hyb}}{dt} = K_{aug}\varphi_{NN} - \overline{D}X_{hyb} + U + \Omega(X_{aug} - X_{hyb}) \tag{29}$$

The true (but unknown) process behavior is assumed to be represented by (8). Then the error dynamics is modeled as the difference between (8) and (21)

$$\frac{d(X_{aug} - X_{hyb})}{dt} = K_{aug}(\varphi - \varphi_{NN}) - \overline{D}(X_{aug} - X_{hyb}) + \Omega(X_{aug} - X_{hyb}) \tag{30}$$

The following definitions take place:

$E_x = (X_{aug} - X_{hyb})$ is termed as the observation error,

$E_\varphi = \varphi - \varphi_{NN}$ is the error signal for updating the ANN parameters.

X_{aug} consists of the measured (X_1) and the estimated ($\hat{X}_2$) states. Thus, (22) can be rearranged as follows

$$\frac{dE_x}{dt} = K_{aug}E_\varphi - (\overline{D} - \Omega)E_x \tag{31}$$

and from (23) the error signal for NN training is

$$E_\varphi = K_{aug}^{-1}\left[\overline{D} - \Omega \quad 1\right]\begin{bmatrix} E_x \\ \dot{E}_x \end{bmatrix} = B\begin{bmatrix} E_x \\ \dot{E}_x \end{bmatrix}, \quad B = K_{aug}^{-1}\left[\overline{D} - \Omega \quad 1\right] \tag{32}$$

Ω is a design parameter which defines the speed of the observation error convergence. The necessary identifiability condition for the kinetic rate vector is the non singularity of matrix K_{aug}. Note that, the error signal for updating the network parameters is a function of the observation error (E_x) and the speed of the observation error ($\dot{E}_x$). The intuition behind is that the network parameters are changed proportionally to their effect on the prediction of the process states and the prediction of their dynamics.

Step 5: Optimization porcesure - Levenberg-Marquardt Quasi-Newton algorithm

The cost function to be minimized at each iteration of network training is the sum of squared errors, where *N* is the time instants over which the optimization is performed (batch mode of training)

$$J_k = \frac{1}{N}\sum_{i=1}^{N}\left[E_\varphi(i)\right]^2 \tag{33}$$

A number of algorithms have been proposed to update the network parameters (*w*). For this study the Levenberg-Marquardt (LM) Quasi Newton method is the chosen algorithm due to its faster convergence than the steepest descent or

conjugate gradient methods, Hagan et al. 1996. One (k) iteration of the classical Newton's method can be written as

$$w_{k+1} = w_k - H_k^{-1} g_k \,,\; g_k = \frac{\partial J_k}{\partial w_k} \,,\; H_k = \frac{\partial J_k^2}{\partial w_k \partial w_k} \tag{34}$$

where g_k is the current gradient of the performance index (25) H_k is the Hessian matrix (second derivatives) of the performance index at the current values (k) of the weights and biases. Unfortunately, it is complex and expensive to compute the Hessian matrix for a dynamical ANN. The LM method is a modification of the classical Newton method that does not require calculation of the second derivatives. It is designed to approach second-order training speed without having to compute directly the Hessian matrix. When the performance function has the form of a sum of error squares (25), at each iteration the Hessian matrix is approximated as

$$H_k = J_k^T J_k \tag{35}$$

where J_k is the Jacobian matrix that contains first derivatives of the network errors (e_k) with respect to the weights and biases

$$J_k = \frac{\partial E_{\varphi k}}{\partial w_k} \,, \tag{36}$$

The computation of the Jacobian matrix is less complex than computing the Hessian matrix. The gradient is then computed as

$$g_k = J_k E_{\varphi k} \tag{37}$$

The LM algorithm updates the network weights in the following way

$$w_{k+1} = w_k - \left[J_k^T J_k + \mu I \right]^{-1} J_k^T E_{\varphi k} \tag{38}$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

5 Case Study - Estimation of Sugar Crystallization Growth Rate

Sugar crystallization occurs through mechanisms of nucleation, growth and agglomeration that are known to be affected by several not well-understood operating conditions. The search for efficient methods for process description is linked both to the scientific interest of understanding fundamental mechanisms of the

crystallization process and to the relevant practical interest of production requirements. The sugar production batch cycle is divided in several phases. During the first phase the pan is partially filled with a juice containing dissolved sucrose. The liquor is concentrated by evaporation, under vacuum, until the supersaturation reaches a predefined value. At this point seed crystals are introduced into the pan to induce the production of crystals (crystallization phase). As evaporation takes place further liquor or water is added to the pan. This maintains the level of supersaturation and increases the volume contents. The third phase consists of tightening which is controlled by the evaporation capacity, see Georgieva et al. 2003 for more details. Since the objective of this paper is to illustrate the technique introduced in section 4, the following assumptions are adopted:

i) Only the states that explicitly depend on the crystal growth rate are extracted from the comprehensive mass balance process model;
ii) The population balance is expressed only in terms of number of crystals;
iii) The agglomeration phenomenon is neglected.

The simplified process model is then

$$\frac{dM_s}{dt} = -k_1 G + F_f \rho_f B_f Pur_f \tag{39}$$

$$\frac{dM_c}{dt} = k_1 G \tag{40}$$

$$\frac{dT_m}{dt} = k_2 G + bF_f + cJ_{vap} + d \tag{41}$$

$$\frac{dm_0}{dt} = k_3 G \tag{42}$$

where M_s is the mass of dissolved sucrose, M_c is the mass of crystals, T_m is the temperature of the massecuite, m_0 is the number of crystals. Pur_f and ρ_f are the purity (mass fraction of sucrose in the dissolved solids) and the density of the incoming feed. F_f is the feed flowrate, J_{vap} is the evaporation rate and b, c, d are parameters incorporating the enthalpy terms and specific heat capacities. They are derived as functions of physical and thermodynamic properties. The full state vector is $X_{aug} = [M_s \quad M_c \quad T_m \quad m_0]^T$, with $K_{aug} = [-k_1 \quad k_1 \quad k_2 \quad k_3]^T$. Now we are in a position to apply the formalism developed in sections 2.2 and 2.3 for this particular reaction process.

We chose the following state partition A : $X_a = M_c$, $X_b = [M_s \quad T_m \quad m_0]^T$ and the solution of equation (13) is

$$A_0 = \begin{bmatrix} 1 & -\frac{k_2}{k_1} & -\frac{k_3}{k_1} \end{bmatrix}^T \tag{43}$$

M_c and T_m are the measured states, then the unique state partition B is

$$X_1 = [M_c \quad T_m]^T, \; X_2 = [M_s \quad m_0]^T,$$

Taking into account (32), the matrices of the second representation of vector Z in (18) are computed as

$$A_1 = \begin{bmatrix} 1 & -\frac{k_2}{k_1} & -\frac{k_3}{k_1} \\ 0 & 1 & 0 \end{bmatrix}^T, \; A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T$$

For this case D=0, then the estimation of the individual elements of Z are

$$\hat{Z}_1 = M_c + \hat{M}_s, \quad \hat{Z}_2 = -\frac{k_2}{k_1} M_c + T_m, \quad \hat{Z}_3 = -\frac{k_3}{k_1} M_c + \hat{m}_0 \tag{44}$$

The analytical expression for the estimation of the unmeasured states is then

$$\begin{bmatrix} \hat{M}_s \\ \hat{m}_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \hat{Z}_1 \\ \hat{Z}_2 \\ \hat{Z}_3 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ -\frac{k_2}{k_1} & 1 \\ -\frac{k_3}{k_1} & 0 \end{bmatrix} \begin{bmatrix} M_c \\ T_m \end{bmatrix} \right) \tag{45}$$

The observation error is defined as

$$E_x = \begin{bmatrix} \hat{M}_s - M_{shyb} \\ M_c - M_{chyb} \\ \hat{m}_0 - m_{0hyb} \\ T_m - T_{mhyb} \end{bmatrix} \tag{46}$$

In the numerical implementation the first derivative of the observation error is computed as the difference between the current $E_x(k)$ and the previous value $E_x(k-1)$ of the observation error divided by the integration step (Δt)

$$\dot{E}_x = \frac{E_x(k) - E_x(k-1)}{\Delta t} \tag{47}$$

The three types of time accounting ANNs were trained with the same training data coming from six industrial batches (training batches). The physical inputs to all networks are (M_c, T_m, m_0, M_s), the network output is G_{NN}. Two of the inputs (M_c, T_m) are measurable, the others (m_0, M_s) are estimated. In order to improve the comparability between the different networks a linear activation function is located at the single output node (see Fig. 2, Layer 2- purelin) and hyperbolic

tangent functions are chosen for the hidden nodes (Fig. 2, Layer 1 - tansig). Though other S-shaped activation functions can be also considered for the hidden nodes, our choice was determined by the symmetry of the hyperbolic tangent function into the interval (-1, 1).

The hybrid models are compared with an analytical model of the sugar crystallization, reported in Oliveira et al. 2009, where G is computed by the following empirical correlation

$$G = K_g \exp\left[-\frac{57000}{R(T_m + 273)}\right](S-1)\exp\left[-13.863(1-P_{sol})\right]\left(1+2\frac{V_c}{V_m}\right), \qquad (48)$$

where S is the supersaturation, P_{sol} is the purity of the solution and V_c / V_m is the volume fraction of crystals. K_g is a constant, optimized following a non-linear least-squares regression.

The performance of the different models is examined with respect to prediction quality of the crystal size distribution (CSD) at the end of the process which is quantified by two parameters - the final average (in mass) particle size (AM) and the final coefficient of particle variation (CV). The predictions given by the models are compared with the experimental data for the CSD (Table 1), coming from 8 batches not used for network training (validation batches). The results with respect to different configurations of the networks are summarized in Tables 2, 3, and 4. All hybrid models (eqs. 31 +RNN/TLNN/RCN) outperform the empirical model (37) particularly with respect to predictions of CV. The predictions based on TLFN and TCN are very close especially for higher reservoir dimension. Increasing the RCN hidden nodes (from 100 to 200) reduces the AM and CV prediction errors, however augmenting the reservoir dimension from 200 to 300 does not bring substantial improvements. The hybrid models with RNN exhibit the best performance though the successful results reported in Table 2 were preceded by a great number of unsuccessful (not converging) trainings. As with respect to learning efforts the RCN training takes in average few seconds on an Intel Core2 Duo processor based computer and by far is the easiest and fastest dynamical regressor.

Table 1 Final CSD – experimental data versus analytical model predictions

batch No.	experimental data		analytical model (eqs. 31+ eq. 37)	
	AM[mm]	CV [%]	AM[mm]	CV [%]
1	0.479	32.6	0.583	21.26
2	0.559	33.7	0.542	18.43
3	0.680	43.6	0.547	18.69
4	0.494	33.7	0.481	14.16
5	0.537	32.5	0.623	24.36
6	0.556	35.5	0.471	13.642
7	0.560	31.6	0.755	34.9
8	0.530	31.2	0.681	27.39
av. err			**13.7%**	**36.1%**

Table 2 Final CSD – hybrid model predictions (eqs. 31+RNN)

RNN	batch No.	AM[mm]	CV[%]
exogenous input	1	0.51	29.6
delay: 2	2	0.48	30.7
	3	0.58	33.6
Recurrent input	4	0.67	31.7
delay: 2	5	0.55	29.5
	6	0.57	34.5
Total Nº of inputs: 14	7	0.59	29.6
hidden neurons:5	8	0.53	32.2
Average error (%)		**4.1**	**7.5**
exogenous input	1	0.59	30.7
delay: 1	2	0.55	41.5
	3	0.59	39.3
Recurrent input	4	0.51	35.9
delay: 3	5	0.49	32.1
	6	0.58	31.7
Total Nº of inputs: 11	7	0.56	30.5
hidden neurons:5	8	0.53	36.8
Average error (%)		**5.2**	**9.2**
exogenous input	1	0.51	30.9
delay:3	2	0.56	31.1
	3	0.59	37.2
Recurrent input	4	0.48	29.8
delay:1	5	0.52	34.8
	6	0.51	32.4
Total Nº of inputs: 17	7	0.59	30.6
hidden neurons:5	8	0.50	33.5
Average error (%)		**3.6**	**6.9**

Table 3 Final CSD – hybrid model predictions (eqs. 31+TLNN)

TLNN	batch No.	AM[mm]	CV[%]
Tap delay: 1	1	0.49	30.8
	2	0.51	37.1
	3	0.62	31.5
Total Nº of inputs: 8	4	0.60	35.5
hidden neurons:5	5	0.57	36.2
	6	0.52	28.7
	7	0.55	38.6
	8	0.54	32.4
Average error (%)		**6.02**	**11.0**
Tap delay: 2	1	0.51	37.5
	2	0.49	31.6
	3	0.59	34.6
	4	0.53	40.3
Total Nº of inputs: 12	5	0.60	35.2
hidden neurons:5	6	0.49	31.5
	7	0.51	29.6
	8	0.54	30.3
Average error (%)		**5.9**	**10.8**
Tap delay: 3	1	0.479	30.3
	2	0.559	41.2
	3	0.680	39.4
	4	0.494	35.7
Total Nº of inputs: 16	5	0.537	35.4
hidden neurons:5	6	0.556	30.3
	7	0.560	29.9
	8	0.530	28.3
Average error (%)		**5.8**	**10.3**

Table 4 Final CSD – hybrid model predictions (eqs. 31+RCN)

RCN	batch No.	AM[mm]	CV[%]
Reservoir	1	0.53	31.2
dimension: 100 nodes	2	0.49	28.1
	3	0.57	43.6
Total Nº of inputs: 4	4	0.61	41.7
	5	0.59	39.6
	6	0.60	36.1
	7	0.51	30.4
	8	0.54	40.2
Average error (%)		**6.8**	**12.0**
Reservoir	1	0.56	40.1
dimension: 200 nodes	2	0.51	37.4
	3	0.61	36.2
	4	0.56	38.6
Total Nº of inputs: 4	5	0.49	28.9
	6	0.59	34.7
	7	0.61	30.4
	8	0.54	39.2
Average error (%)		**5.9**	**10.2**
Reservoir	1	0.59	33.9
dimension: 300 nodes	2	0.48	28.8
	3	0.57	39.7
	4	0.51	29.6
Total Nº of inputs: 4	5	0.53	31.8
	6	0.51	33.9
	7	0.49	30.7
	8	0.57	36.9
Average error (%)		**5.9**	**9.8**

6 Conclusions

This work is focused on presenting a more efficient computational scheme for estimation of process reaction rates based on temporal artificial neural network (ANN) architectures. It is assumed that the kinetics coefficients are all known and do not change over the process run, while the process states are not all measured and therefore need to be estimated. It is a very common scenario in reaction systems with low or medium complexity.

The concepts developed here concern two aspects. On one side we formulate a hybrid (temporal ANN+ analytical) model that outperforms the traditional reaction rate estimation approaches. On the other side a procedure for ANN supervised training is introduced when target (reference) outputs are not available. The network is embedded in the framework of a first principle process model and the error signal for updating the network weights is determined analytically. According to the procedure, first the unmeasured states are estimated independently of the reaction rates and then the ANN is trained with the estimated and the measured

data. Ongoing research is related with the integration of the hybrid models proposed in this work in the framework of a model based predictive control.

Acknowledgements. This work was financed by the Portuguese Foundation for Science and Technology within the activity of the Research Unit IEETA-Aveiro, which is gratefully acknowledged.

References

1. Antonelo, E.A., Schrauwen, B., Campenhout, J.V.: Generative modeling of autonomous robots and their environments using reservoir computing. Neural Processing Letters 26(3), 233–249 (2007)
2. Bastin, G., Dochain, D.: On-line estimation and adaptive control of bioreactors. Elsevier Science Publishers, Amsterdam (1990)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
4. Chen, L., Bastin, G.: Structural identifiability of the yeals coefficients in bioprocess models when the reaction rates are unknown. Mathematical Biosciences 132, 35–67 (1996)
5. Georgieva, P., Meireles, M.J., Feyo de Azevedo, S.: Knowledge Based Hybrid Modeling of a Batch Crystallization When Accounting for Nucleation, Growth and Agglomeration Phenomena. Chem. Eng. Science 58, 3699–3707 (2003)
6. Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design. PWS Publishing, Boston (1996)
7. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, NJ (1999)
8. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001a)
9. Maass, W., Natschlager, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14(11), 2531–2560 (2002)
10. Mandic, D.P., Chambers, J.A.: Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability (Adaptive & Learning Systems for Signal Processing, Communications & Control). Wiley, Chichester (2001)
11. Noykove, N., Muller, T.G., Gylenberg, M., Timmer, J.: Quantitative analysis of anaerobic wastewater treatment processes: identifiably and parameter estimation. Biotechnology and bioengineering 78(1), 91–103 (2002)
12. Oliveira, C., Georgieva, P., Rocha, F., Feyo de Azevedo, S.: Artificial Neural Networks for Modeling in Reaction Process Systems. In: Neural Computing & Applications, vol. 18, pp. 15–24. Springer, Heidelberg (2009)
13. Principe, J.C., Euliano, N.R., Lefebvre, W.C.: Neural and adaptive systems: Fundamentals through simulations, New York (2000)
14. Steil, J.J.: Backpropagation-Decorrelation: Online recurrent learning with O(N) complexity. In: Proc. Int. Joint Conf. on Neural Networks (IJCNN), vol. 1, pp. 843–848 (2004)
15. Walter, E., Pronzato, L.: Identification of parametric models from experimental data. Springer, UK (1997)