

# Neural Modelling of a Yeast Fermentation Process Using Extreme Learning Machines

Maciej Ławryńczuk

**Abstract** This work details development of dynamic neural models of a yeast fermentation chemical reactor using Extreme Learning Machines (ELM). The ELM approach calculates very efficiently, without nonlinear optimisation, dynamic models, but only in the non-recurrent serial-parallel configuration. It is shown that in the case of the considered benchmark the ELM technique gives models which are also quite good recurrent long-range predictors, they work in the parallel configuration (simulation mode). Furthermore, properties of neural models obtained by the ELM and classical (optimisation-based) approaches are compared.

**Keywords** Neural networks • Extreme learning machines

## 1 Introduction

Neural networks [3], due to their excellent approximation ability, are used very frequently as models of nonlinear systems in many fields, e.g. in advanced control algorithms [6, 13], in pattern recognition [12], in interpretation of medical images [11], in fault diagnosis and fault-tolerant control [14] and in optimisation [15].

Typically, determination of parameters (weights) of neural networks (training) needs solving an optimisation problem [3]. Such a problem is nonlinear and it is likely to be non-convex and multi-modal. An alternative is to use Extreme Learning Machines [2]. In the ELM approach the weights of the second layer are determined explicitly, without nonlinear optimisation, while the weights of the first layer are chosen randomly. The ELM method yields non-recurrent serial-parallel models whereas in the case of dynamic systems the objective is to find recurrent models, which give good long-range prediction (the parallel configuration or the simulation mode) [10]. This work reports development of dynamic neural models of a yeast

---

M. Ławryńczuk (✉)

Institute of Control and Computation Engineering, Warsaw University of Technology, Ul. Nowowiejska 15/19, 00-665 Warsaw, Poland  
e-mail: M.Lawrynczuk@ia.pw.edu.pl

© Springer International Publishing Switzerland 2016

R. Szewczyk et al. (eds.), *Challenges in Automation, Robotics and Measurement Techniques*, Advances in Intelligent Systems and Computing 440, DOI 10.1007/978-3-319-29357-8\_2

fermentation reactor using the ELM approach and the classical (optimisation-based) method. Long-range prediction abilities and complexity of both model classes are compared.

## 2 Structures of Neural Model

Let  $u$  and  $y$  denote the input and output variables of a dynamic process, respectively. Figure 1 depicts two possible model configurations [10]. In the non-recurrent serial-parallel model the output signal for the sampling instant  $k$  is a function of the process input and output signal values from some previous instants

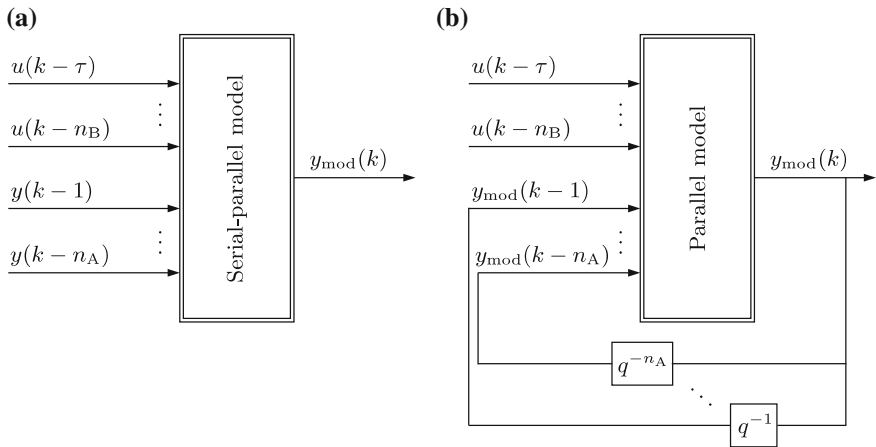
$$y_{\text{mod}}(k) = f(u(k - \tau), \dots, u(k - n_B), y(k - 1), \dots, y(k - n_A)) \quad (1)$$

where the integers  $\tau, n_A, n_B$  determine the order of dynamics. In the recurrent parallel model (the simulation model), the past process outputs are replaced by the model outputs

$$y_{\text{mod}}(k) = f(u(k - \tau), \dots, u(k - n_B), y_{\text{mod}}(k - 1), \dots, y_{\text{mod}}(k - n_A)) \quad (2)$$

The serial-parallel model is a one-step-ahead predictor (the Autoregressive with eXogenous input (ARX) model), the parallel one is a multiple-steps-ahead predictor (the Output Error (OE) model).

The feedforward neural network with two layers is the most popular structure [3]. Taking into account Eq. (1) or Eq. (2), the network has  $n_A + n_B - \tau + 1$  input nodes,  $K$  nonlinear hidden neurons with the nonlinear transfer function  $\varphi: \mathbb{R} \rightarrow \mathbb{R}$



**Fig. 1** Structures of dynamic models: **a** the serial-parallel model, **b** the parallel model

(e.g.  $\varphi = \tanh$ ), one linear output element (sum) and one output  $y_{\text{mod}}(k)$ . The weights of the first layer are denoted by  $w_{i,j}^1$ , where  $i = 1, \dots, K, j = 0, \dots, n_A + n_B - \tau + 1$ , the weights of the second layer are denoted by  $w_i^2$ , where  $i = 0, \dots, K$ . The output signal of the neural ARX or OE model is

$$y_{\text{mod}}(k) = w_0^2 + \sum_{i=1}^K w_i^2 v_i(k)$$

where  $v_i(k)$  denotes the output of the  $i^{\text{th}}$  hidden node. Let  $z_i(k)$  denote the sum of input signals of the  $i^{\text{th}}$  hidden node. Hence,  $v_i(k) = \varphi(z_i(k))$ . For the ARX model defined by Eq. (1), one has

$$z_i(k) = w_{i,0}^1 + \sum_{j=1}^{n_B-\tau+1} w_{i,j}^1 u(k - \tau + 1 - j) + \sum_{j=1}^{n_A} w_{i,n_B-\tau+1+j}^1 y(k - j)$$

whereas for the OE model defined by Eq. (2), one has

$$z_i(k) = w_{i,0}^1 + \sum_{j=1}^{n_B-\tau+1} w_{i,j}^1 u(k - \tau + 1 - j) + \sum_{j=1}^{n_A} w_{i,n_B-\tau+1+j}^1 y_{\text{mod}}(k - j)$$

### 3 Training of Neural Models

#### 3.1 Classical Approach

Neural models are trained using a series of input-output samples. The objective is to find the values of the weights which result in good accuracy of the model. Accuracy is defined by the following Sum of Squared Errors (SSE) cost-function

$$\text{SSE} = \sum_{k=k_{\min}}^{k_{\max}} (y_{\text{mod}}(k) - y(k))^2 \quad (3)$$

where  $y_{\text{mod}}(k)$  is the output signal calculated by the model for the sampling instant  $k$  (in the serial-parallel or parallel configuration),  $y(k)$  is the real value of the recorded process output,  $k_{\min}$  and  $k_{\max}$  define the training samples. Training consists in solving the unconstrained optimisation problem

$$\min_{w_{1,0}^1, \dots, w_{K,n_A+n_B-\tau+1}^1, w_0^2, \dots, w_K^2} \{ \text{SSE} \} \quad (4)$$

It is necessary to emphasise the fact that for training by means of the classical approach a nonlinear, possibly multi-modal, non-convex optimisation problem (4) must be solved. For this purpose a number of classical nonlinear optimisation algorithms may be used, e.g. the steepest descent method, the conjugated gradients methods, the quasi-Newton variable metrics methods, the Levenberg-Marquardt algorithm or heuristic optimisation approaches, e.g. evolutionary algorithms [3].

### 3.2 Extreme Learning Machines

In the ELM approach [2] the structure of the network the same as in the classical approach, but training does not need solving a computationally demanding nonlinear optimisation problem (4). The weights are calculated from the following procedure:

1. The weights of the first layer are assigned randomly.
2. The outputs of all hidden nodes (i.e.  $v_1(k), \dots, v_K(k)$ ) for all training data samples (i.e. for  $k = k_{\min}, \dots, k_{\max}$ ) are calculated.
3. The weights of the second layer are calculated analytically.

In order to simplify calculations it is assumed that there is a sufficient number of hidden nodes. Since the weights of the first layer are chosen randomly, the training optimisation problem (4) becomes

$$\min_{\mathbf{w}^2 = [w_0^2 \dots w_K^2]^T} \{\text{SSE}\} \quad (5)$$

Although the classical minimised objective function (3) may be used, it is more practical to use

$$\text{SSE} = \sum_{k=k_{\min}}^{k_{\max}} (y_{\text{mod}}(k) - y(k))^2 + \alpha \sum_{i=0}^K (w_i^2)^2 \quad (6)$$

where the regularisation term  $\sum_{i=0}^K (w_i^2)^2$  minimises values of the weights of the second layer,  $\alpha > 0$ . The cost-function (6) may be expressed in a vector notation

$$\text{SSE} = (\mathbf{y}_{\text{mod}} - \mathbf{y})^T (\mathbf{y}_{\text{mod}} - \mathbf{y}) + \alpha (\mathbf{w}^2)^T \mathbf{w}^2 = \|\mathbf{y}_{\text{mod}} - \mathbf{y}\|^2 + \alpha \|\mathbf{w}^2\|^2$$

where  $\mathbf{y}_{\text{mod}} = [y_{\text{mod}}(k_{\min}) \dots y_{\text{mod}}(k_{\max})]^T$ ,  $\mathbf{y} = [y(k_{\min}) \dots y(k_{\max})]^T$ . The outputs of the hidden nodes for all training samples give a matrix of dimensionality  $(k_{\max} - k_{\min} + 1) \times (K + 1)$

$$\mathbf{v} = \begin{bmatrix} v_0(k_{\min}) & \dots & v_K(k_{\min}) \\ \vdots & \ddots & \vdots \\ v_0(k_{\max}) & \dots & v_K(k_{\max}) \end{bmatrix}$$

Hence, in the case of the ARX configuration, the model output vector is  $\mathbf{y}_{\text{mod}} = \mathbf{v}\mathbf{w}^2$  and the minimised cost-function (6) becomes

$$\text{SSE} = \|\mathbf{v}\mathbf{w}^2 - \mathbf{y}\|^2 + \alpha \|\mathbf{w}^2\|^2 \quad (7)$$

As the minimised cost-function (7) is a second-order polynomial of the weights of the second layer,  $\mathbf{w}^2$ , they may be determined analytically, without nonlinear optimisation, by zeroing the derivative vector  $\frac{d\text{SSE}}{d\mathbf{w}^2} = 2\mathbf{v}^T(\mathbf{v}\mathbf{w}^2 - \mathbf{y}) + 2\alpha\mathbf{w}^2$ , which gives

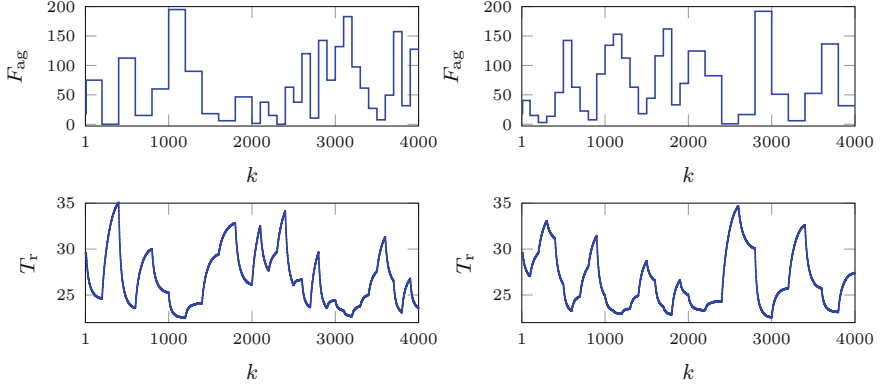
$$\mathbf{w}^2 = (\mathbf{v}^T\mathbf{v} + \alpha\mathbf{I}_{(K+1)\times(K+1)})^{-1}\mathbf{v}^T\mathbf{y} \quad (8)$$

## 4 Simulation Results

The considered process is a yeast fermentation reactor (*Saccharomyces cerevisiae*). Yeast is commonly used in many branches of the food industry, in particular in: bakeries, breweries, wineries and distilleries. The reactor manifests significantly nonlinear behaviour. It cannot be modelled precisely by means of linear models with constant parameters and it cannot be controlled efficiently by the classical linear control schemes [6, 8]. Neural networks may be successfully used to approximate behaviour of the process as described in [6–9]. Different nonlinear controllers may be used for the process, including a fuzzy-PI controller [1], an inverse neural-network controller [4], a reinforcement learning algorithm [5] and nonlinear Model Predictive Control (MPC) strategies [6–9].

During fermentation the reactor temperature must be maintained within a narrow range because temperature greatly influences process operation. Imprecise temperature control is likely to result in a reduction of fermentation yield [1]. Hence, the problem is to find a sufficiently precise model of the reactor temperature, which may be next used for developing a control system [6–9]. From the perspective of a control algorithm the reactor is a single-input single-output process: the coolant flow rate ( $F_{\text{ag}}$ ) is the input (the manipulated variable), the reactor temperature ( $T_r$ ) is the output (the controlled variable).

The first-principle model consists of a set of nonlinear differential equations [6–9]. It is treated as the “real” process. The first-principle model is simulated open-loop (without any controller) in order to obtain data sets necessary for model identification. Figure 2 depicts training and validation data sets. Each set has 4000 samples. The sampling period is 30 min [9]. The output signal contains small measurement noise. The training data set is used only for model training, i.e. the training error is minimised for that set. The validation data set is used only to calculate the validation error after training of different model structures. The validation error indicates generalisation abilities of the models. The models are finally compared taking into account the validation error.



**Fig. 2** The training data set (*left*) and the validation data set (*right*)

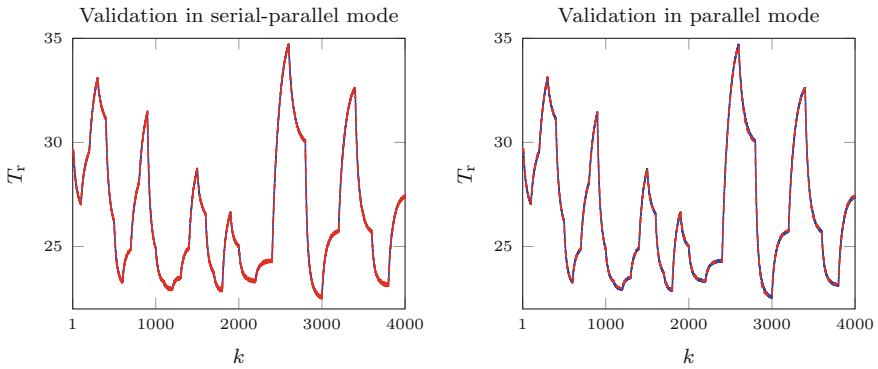
The hyperbolic tangent transfer function  $\varphi = \tanh$  is used in the hidden layer. Because input and output process variables have different orders of magnitude, they are scaled:  $u = 0.01(F_{\text{ag}} - F_{\text{ag,nom}})$ ,  $y = 0.1(T_r - T_{r,\text{nom}})$ , where  $F_{\text{ag,nom}} = 18$  l/h,  $T_{r,\text{nom}} = 29.573212$  °C correspond to the nominal operating point.

#### 4.1 Training of Classical Neural Models

Accuracy of models is assessed taking into account the SSE index (3). For model training the efficient Broyden-Fletcher-Goldfarb-Shanno (BFGS) nonlinear optimisation algorithm is used, training is carried out in the parallel (recurrent) configuration. The second-order dynamics is assumed (i.e.  $\tau = 1$ ,  $n_A = n_B = 2$ ) as in [6, 8]. In order to find a neural model with good accuracy and generalisation abilities the networks with  $K = 1, \dots, 7$  hidden nodes are trained and compared. For each structure training is repeated 10 times (because of possible shallow local minima), all weights are initialised randomly. Table 1 presents properties of the best obtained models. Increasing the number of model parameters leads to reducing the training error ( $\text{SSE}_{\text{OE}}^{\text{train}}$ ). On the other hand, when  $K > 3$ , the models have too many weights and the validation error ( $\text{SSE}_{\text{OE}}^{\text{val}}$ ) increases, which means that the generalisation ability deteriorates. Hence, the model with as few as 3 hidden nodes is finally chosen. The model has only 19 weights. Figure 3 compares the validation data set and the output of the chosen neural model. The model works fine both in serial-parallel and parallel configurations.

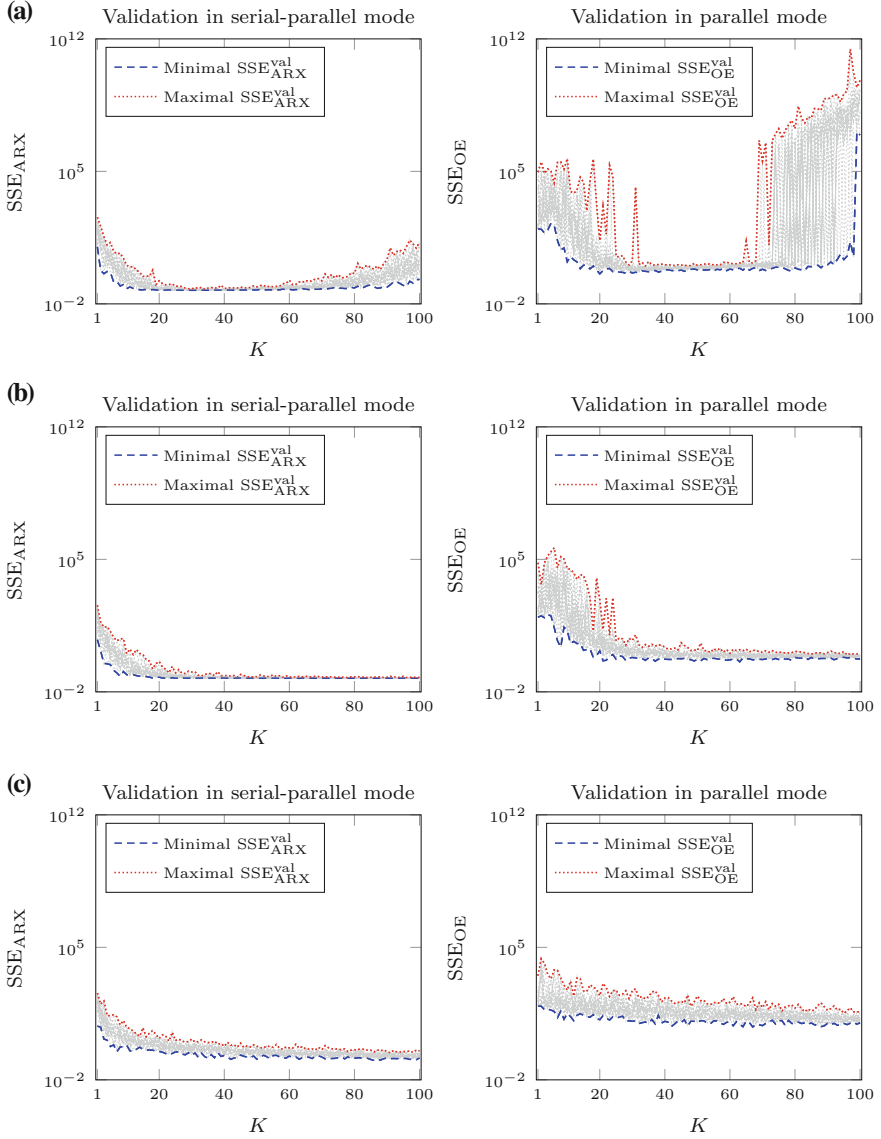
**Table 1** Comparison of the best neural models trained in the parallel configuration and in the classical approach; NP—the number of parameters

$K$	NP	$SSE_{OE}^{train}$	$SSE_{OE}^{val}$
1	7	$1.1649 \times 10^1$	$1.3895 \times 10^1$
2	13	$3.2821 \times 10^{-1}$	$3.2568 \times 10^{-1}$
3	19	$2.0137 \times 10^{-1}$	$1.8273 \times 10^{-1}$
4	25	$1.9868 \times 10^{-1}$	$1.9063 \times 10^{-1}$
5	31	$1.3642 \times 10^{-1}$	$1.9712 \times 10^{-1}$
6	37	$1.3404 \times 10^{-1}$	$2.0440 \times 10^{-1}$
7	43	$1.2801 \times 10^{-1}$	$2.9391 \times 10^{-1}$

**Fig. 3** The validation data set (*solid line*) versus the output of the best neural model with 3 hidden nodes trained in the parallel configuration and in the classical approach (*dashed line*),  $SSE_{ARX}^{val} = 1.2320 \times 10^{-1}$ ,  $SSE_{OE}^{val} = 1.8273 \times 10^{-1}$ 

## 4.2 Training of Extreme Learning Machines

In the classical approach to training, i.e. when the cost-function is minimised by means of a nonlinear optimisation algorithm, the model may be trained in both serial-parallel and parallel configurations. When the model is trained in the ELM manner, training is possible only in the first mode. Nevertheless, since the objective is to obtain good dynamic models, the ELM neural models are validated in the parallel configuration after training. The models with  $K = 1, 2, \dots, 100$  hidden nodes are considered, for each model structure the weights are calculated 20 times. The SSE error with the regularisation term (6) is minimised during training, the models are evaluated using the error (3), i.e. without that term. Validation errors of all determined models are shown in Fig. 4, both serial-parallel and parallel configurations are considered. Additionally, validation errors of the best selected neural models are given in Table 2. When  $\alpha = 0$  (no regularisation during training), the models with too few and too many hidden nodes frequently give huge errors, particularly in the



**Fig. 4** Validation errors of all models trained in the serial-parallel configuration and in the ELM approach for different values of the regularisation parameter  $\alpha$ : **a**  $\alpha = 0$ , **b**  $\alpha = 0.01$ , **c**  $\alpha = 10$

parallel configuration, which is illustrated in Fig. 4a. Moreover, there are huge differences of accuracy between the best and the worst models, even for the same number of hidden nodes. As the regularisation parameter  $\alpha$  increases, the differences between models of the same structure become smaller and smaller and the effect of



**Table 2** Comparison of the best neural models trained in the serial-parallel configuration and in the ELM approach; NP—the number of parameters

K	NP	$SSE_{OE}^{val}$				
		$\alpha = 0$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$
10	61	$1.6836 \times 10^0$	$1.6927 \times 10^1$	$1.7547 \times 10^1$	$2.9732 \times 10^1$	$3.2173 \times 10^1$
20	121	$3.8370 \times 10^{-1}$	$1.4811 \times 10^0$	$4.6646 \times 10^0$	$8.3131 \times 10^0$	$1.5030 \times 10^1$
30	181	$4.2560 \times 10^{-1}$	$6.5626 \times 10^{-1}$	$1.2046 \times 10^0$	$4.0492 \times 10^0$	$1.2573 \times 10^1$
40	241	$6.2061 \times 10^{-1}$	$5.3584 \times 10^{-1}$	$1.0773 \times 10^0$	$4.6231 \times 10^0$	$1.2441 \times 10^1$
50	301	$6.2207 \times 10^{-1}$	$5.1635 \times 10^{-1}$	$7.5840 \times 10^{-1}$	$3.3643 \times 10^0$	$1.0516 \times 10^1$
60	361	$6.9239 \times 10^{-1}$	$4.9123 \times 10^{-1}$	$7.8591 \times 10^{-1}$	$3.5661 \times 10^0$	$1.1595 \times 10^1$
70	421	$6.7860 \times 10^{-1}$	$5.1162 \times 10^{-1}$	$8.8329 \times 10^{-1}$	$2.8592 \times 10^0$	$7.3480 \times 10^0$
80	481	$8.0274 \times 10^{-1}$	$6.6283 \times 10^{-1}$	$7.9178 \times 10^{-1}$	$2.3020 \times 10^0$	$9.0120 \times 10^0$
90	541	$1.1602 \times 10^{-1}$	$5.7329 \times 10^{-1}$	$6.8762 \times 10^{-1}$	$2.3021 \times 10^0$	$9.7736 \times 10^0$
100	601	$8.3138 \times 10^{-1}$	$5.5840 \times 10^{-1}$	$5.7272 \times 10^{-1}$	$1.9188 \times 10^0$	$1.0406 \times 10^1$

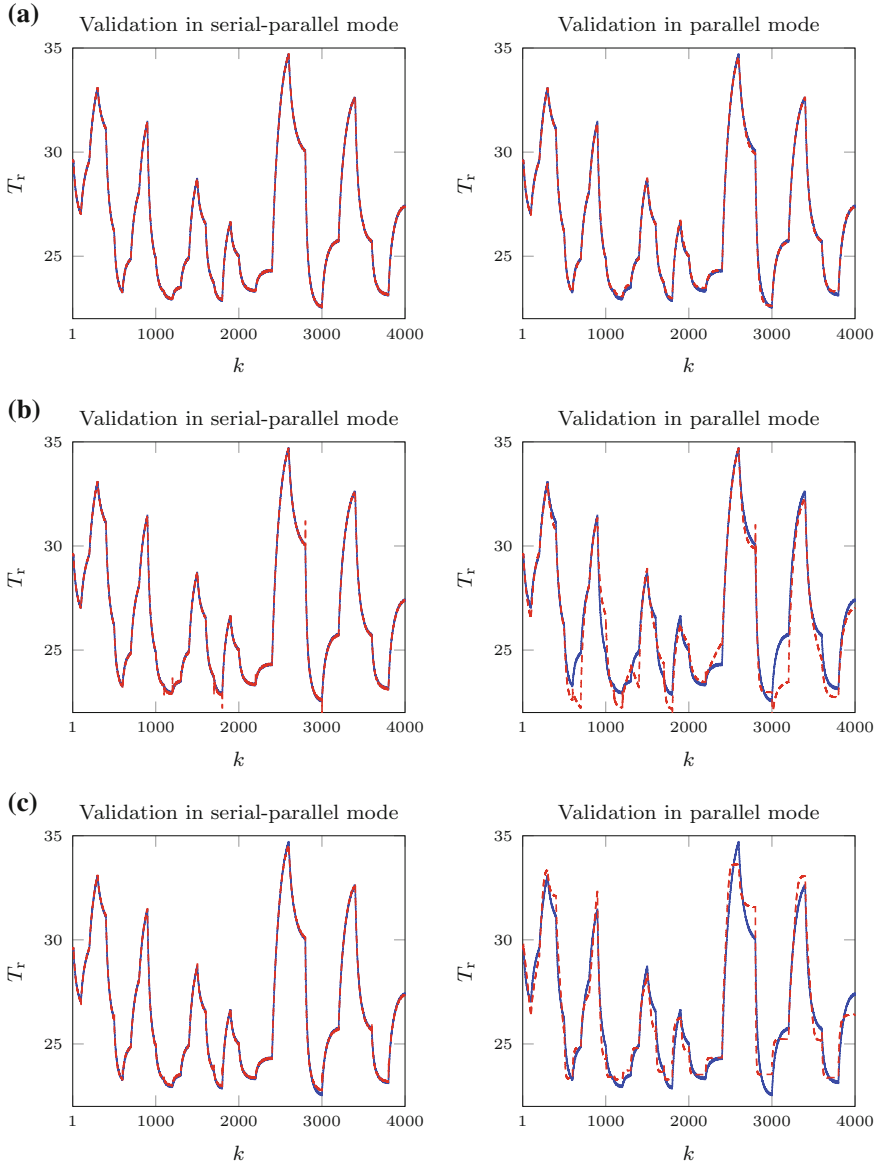
overparameterisation is not present (for  $K \leq 100$ ). Unfortunately, the bigger the parameter  $\alpha$ , the bigger the errors.

The best model trained using the ELM approach is obtained for 20 hidden nodes (it has 121 weights). Its validation error is  $SSE_{OE}^{val} = 3.8370 \times 10^{-1}$  whereas the model trained by means of the classical approach needs only 3 hidden nodes (19 weights) and it is characterised by the error  $SSE_{OE}^{val} = 1.8273 \times 10^{-1}$ .

Figure 5 compares the validation data set and the output of three neural models with 20 hidden nodes trained in the serial-parallel configuration and in the ELM approach. The best model (minimal  $SSE_{OE}^{val}$  for  $\alpha = 0$ ) works fine both in serial-parallel and parallel configurations. Conversely, the worst neural model (maximal  $SSE_{OE}^{val}$  for  $\alpha = 0$ ) fails to give good long-range prediction in the parallel configuration. When the regularisation factor is big ( $\alpha = 10$ ), the best model (minimal  $SSE_{OE}^{val}$ ) also gives a significant error in the parallel configuration.

## 5 Conclusions

The ELM approach makes it possible to very efficiently find values of the weights of neural networks since they are calculated explicitly, without nonlinear optimisation. Although the ELM technique trains networks only in the non-recurrent serial-parallel configuration, for the considered yeast fermentation reactor they perform well also in the recurrent parallel mode. As the input weights are chosen randomly, the ELM approach gives the best network with 20 hidden nodes whereas in the classical approach only 3 nodes are sufficient.



**Fig. 5** The validation data set (*solid line*) versus the output of the neural model with 20 hidden nodes trained in the serial-parallel configuration and in the ELM approach (*dashed line*): **a** the best neural model (minimal  $SSE_{OE}^{val}$  for  $\alpha = 0$ ),  $SSE_{ARX}^{val} = 5.2808 \times 10^{-2}$ ,  $SSE_{OE}^{val} = 3.8370 \times 10^{-1}$ ; **b** the worst neural model (maximal  $SSE_{OE}^{val}$  for  $\alpha = 0$ ),  $SSE_{ARX}^{val} = 1.6595 \times 10^{-1}$ ,  $SSE_{OE}^{val} = 2.2227 \times 10^1$ , **c** the best neural model (minimal  $SSE_{OE}^{val}$  for  $\alpha = 10$ ),  $SSE_{ARX}^{val} = 3.3633 \times 10^{-1}$ ,  $SSE_{OE}^{val} = 1.5030 \times 10^1$

## References

1. Fonseca, R.R., Schmitz, J.E., Fileti, A.M.F., da Silva, F.V.: A fuzzy-split range control system applied to a fermentation process. *Bioresour. Technol.* **142**, 475–482 (2013)
2. Huang, G.-B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **42**, 513–529 (2012)
3. Haykin, S.: *Neural Networks and Learning Machines*. Prentice Hall, Englewood Cliffs (2009)
4. Imtiaz, U., Assadzadeh, A., Jamuar, S.S., Sahu, J.N.: Bioreactor temperature profile controller using inverse neural network (INN) for production of ethanol. *J. Process Control* **23**, 731–742 (2013)
5. Li, D., Qian, L., Jin, Q., Tan, T.: Reinforcement learning control with adaptive gain for a *Saccharomyces cerevisiae* fermentation process. *Appl. Soft Comput.* **11**, 4488–4495 (2011)
6. Ławryńczuk, M.: Computationally efficient model predictive control algorithms. In: *A Neural Network Approach, Studies in Systems, Decision and Control*, vol. 3, Springer, Heidelberg (2014)
7. Ławryńczuk, M.: Online set-point optimisation cooperating with predictive control of a yeast fermentation process: a neural network approach. *Eng. Appl. Artif. Intell.* **24**, 968–982 (2011)
8. Ławryńczuk, M.: Modelling and nonlinear predictive control of a yeast fermentation biochemical reactor using neural networks. *Chem. Eng. J.* **145**, 290–307 (2008)
9. Nagy, Z.K.: Model based control of a yeast fermentation bioreactors using optimally designed artificial neural networks. *Chem. Eng. J.* **127**, 95–109 (2007)
10. Nelles, O.: Nonlinear system identification. In: *From Classical Approaches to Neural Networks and Fuzzy Models*. Springer, Berlin (2001)
11. Ogiela, M., Tadeusiewicz, R.: Modern computational intelligence methods for the interpretation of medical images. In: *Studies in Computational Intelligence*, vol. 84. Springer, Heidelberg (2008)
12. Ripley, B.D.: *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge (1996)
13. Tatjewski, P.: Advanced control of industrial processes. In: *Structures and Algorithms*. Springer, London (2007)
14. Witczak, M.: Fault diagnosis and fault-tolerant control strategies for non-linear systems: analytical and soft computing approaches. In: *Lecture Notes in Electrical Engineering*, vol. 266. Springer, Heidelberg (2014)
15. Yan, Z., Wang, J.: Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks. *IEEE Trans. Neural Networks Learn. Syst.* **25**, 457–469 (2014)

Challenges in Automation, Robotics and Measurement  
Techniques

Proceedings of AUTOMATION-2016, March 2-4, 2016,  
Warsaw, Poland

Szewczyk, R.; Zieliński, C.; Kaliczyńska, M. (Eds.)

2016, XVI, 923 p. 459 illus., 147 illus. in color.,

Softcover

ISBN: 978-3-319-29356-1