Proceedings of the American Control Conference
Anchorage, AK May 8-10, 2002

Stochastic learning methods for dynamic neural networks: simulated and real-data comparisons [1]

Krzysztof Patan [2] and Thomas Parisini [3]

Abstract

In the paper some stochastic methods for dynamic neural network training are presented and compared. The considered network is composed of dynamic neurons, which contain inner feedbacks. This network can be used as a part of fault diagnosis system to generate residuals. Classical optimisation techniques, based on back propagation idea, suffer from many well–known drawbacks. Two stochastic algorithms are tested as training algorithms to overcome these difficulties. Efficiency of proposed learning methods is checked on two examples: modelling of an unknown linear dynamic system basing on simulated data and modelling of the actuator behaviour in the first section of the evaporation station in the Sugar Factory, Lublin using real data measurements. In these two significant examples, the stochastic learning algorithms are extensively compared from many different perspectives.

1 Introduction

With increasing demands on the reliability and safety of technical processes, various Fault Detection and Isolation (FDI) approaches have been proposed. The quantitative model-based approach, most frequently is based on the idea of analytical redundancy [2, 5, 6] that requires of an analytical mathematical model of the system. However, in practice it is difficult to meet the demands of such a method due to the inevitable model mismatch, noise, disturbances and inherent nonlinearity. In this case, the use of knowledge-model-based techniques, i.e. artificial intelligence [4, 12] either in the framework of diagnosis expert systems [3] or in combination with a human operator is the only feasible way. One of the possible solutions is to use neural networks. Neural networks under consideration belong to the class of locally recurrent globally feed-forward [9]. They have an architecture that is similar to the feed-forward multi-layer perceptron and dynamic characteristics included in their processing units. The networks under consideration are designed using the Dynamic Neuron Models (DNM). A single dynamic neuron consists of an adder module, linear dynamic system – Infinite Impulse Response (IIR) filter, and nonlinear activation module. When such neurons are connected into a multi-layer structure, a powerful approximating tool may be obtained. Taking into account that neuron by itself has dynamic characteristics, it is not required to introduce any global feedback to the network structure. In this way a simple architecture is obtained, what makes it relatively easier to elaborate a proper learning algorithm and contrary to recurrent networks to keep stability of the neural model. Taking into consideration dynamic characteristics of the network, it is possible to apply it for modelling and identification of nonlinear systems. It is especially useful when there are no mathematical models of the modelled system, and analytical models and parameter-identification algorithms cannot be applied [4, 11]. Based on the dynamic neural networks, the fault detection and isolation system for diagnosis of industrial processes can be designed [2, 11]. In recent few years this kind of neural networks was successfully applied in several fault diagnosis applications such as fault detection and isolation in a two tank laboratory system or fault detection of the instrumental faults in chosen parts of the sugar factory [7, 10]. The fundamental training method for the considered dynamic networks is the Extended Dynamic Back-Propagation (EDBP) algorithm [9]. This is very simple algorithm utilizing the back-propagation error scheme. This algorithm may have both on-line and off-line forms, and therefore it can be widely used in the control theory. The identification of dynamic systems, however, is an example where training of the neural network is not a trivial problem. The error (cost) function is strongly multimodal, and during training, the EDBP often gets stuck in local minima. Even multi-starting of the EDBP cannot yield the expected results. Therefore other methods that belong to the class of global optimisation should be applied. To overcome above problems, it is proposed to use a stochastic method, so called Adaptive Random Search (ARS) [16]. Additionally, to carry out more detailed analysis and comparison, it is applied another stochastic approach Simultaneous Perturbation Stochastic Approximation (SPSA) [1, 14]. The paper is organised as follows: in Section 2 the brief description of the neural structure is presented. Sections 3 and 4 present the ARS and SPSA learning methods and in Section 5

[1] This research has been partially supported by the EU RTN Project "DAMADICS".

[2] Institute of Control and Computation Engineering, University of Zielona Góra, ul. Podgórna 50, 65-246 Zielona Góra, Poland, k.patan@issi.uz.zgora.pl.

[3] Dept. of Electrical, Electronic and Computer Engineering, University of Trieste, Via Valerio 10, 34127 Trieste, Italy, parisini@univ.trieste.it.

extensive simulations and comparisons are reported.

2 Dynamic neural network

The artificial neural network under consideration, has structure similar to feed-forward multi-layer perceptron. The neurons are organized in layers, where the last one is called the output layer and other layers are called the hidden layers. The information flows in one direction only, from successive layer to the following one. Dynamics is introduced to the neuron in such a way that the neuron activation depends on its internal states. It is done by introducing a linear dynamic system – the IIR filter – to the neuron structure [9]. Therefore this kind of neural network is often called locally recurrent globally feed-forward. Each neuron in the dynamic network reproduces the past signal values with the input $u_p(k)$, for $p = 1, 2, \ldots, P$, where P is the number of inputs, and the output $y(k)$. Figure 1 shows the structure of this neuron with many inputs, which is called the Dynamic Neuron Model. In such models one can distinguish three main parts: a weight adder, a filter block and an activation block. The behaviour of the dynamic neuron model is described by the following set of equations:

$$\begin{aligned} x(k) &= \sum_{p=1}^{P} w_p u_p(k) \\ \widetilde{y}(k) &= -\sum_{i=1}^{n} a_i \widetilde{y}(k-i) + \sum_{i=0}^{n} b_i x(k-i) \\ y(k) &= F(g \cdot \widetilde{y}(k) + c) \end{aligned} \tag{1}$$

where w_p denotes the input weights, $\widetilde{y}(k)$ is the filter output, $a_i,\ i = 1, \ldots, n$ and $b_i,\ i = 0, \ldots, n$ are the feedback and feed-forward filter parameters, respectively, n denotes the filter order, and g and c are the slope parameter and the threshold of the nonlinear activation function $F(\cdot)$, respectively. In this dynamic structure the slope parameter g can change. Thus, the dynamic neuron can model the biological neuron better. Introduction of the slope parameter g to the activation operation can be very helpful, particularly in the case of non-linear squashing activation functions, i.e.: sigmoidal or hyperbolic tangent. This kind of activation function are characterized by saturation ranges. Let us assume that the non-linear model is the logistic function, e.g. sigmoidal one. This is a limited function, for large positive $x(k)$ the function tends to one, and for large negative $x(k)$ the function tends to zero. In the case when activation data are large, the activation function drives into its own saturation range and response of the neuron could become a constant value or a signal with negligible small amplitude. This is a very undesirable effect, which can be compensated by application of the slope parameter g. In limit, when

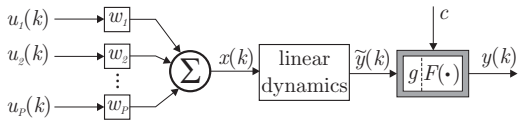

Figure 1: Structure of the DNM with P inputs.

$g \to \infty$, the sigmoidal function tends to become the sign function with an angle equal to $\frac{\Pi}{2}$ for $x(k) = 0$. On the other hand, when $g \to 0$ the sigmoidal function tends to become the constant function $y(k) = 0.5$.

Taking into account the fact that this network has no recurrent links between neurons, to adapt the network parameters, a training algorithm based on the back-propagation idea can be elaborated. The calculated output is propagated back to the inputs through the hidden layers containing dynamic filters. As a result the Extended Dynamic Back Propagation can be defined [7]. This algorithm can work in both the off–line and the on-line modes. The choice of the suitable algorithm depends on the specific problem.

3 Adaptive Random Search

In this section, algorithms which have iterative character are considered. Assuming that the sequence of solutions $\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_k$ is already appointed, a way of achieving next point $\hat{\boldsymbol{\theta}}_{k+1}$ is formulated as follows [16]:

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k + \mathbf{r}_k \tag{2}$$

where $\boldsymbol{\theta}_k$ is the estimate of the $\boldsymbol{\theta}^\star$ at the k-th iteration, and $\mathbf{r}_k$ is the perturbation vector generated randomly according to the normal distribution $\mathcal{N}(0, \sigma)$. New solution $\hat{\boldsymbol{\theta}}_{k+1}$ is accepted when the cost function $J(\hat{\boldsymbol{\theta}}_{k+1})$ is less than $J(\hat{\boldsymbol{\theta}}_{\boldsymbol{k}})$ otherwise $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k$. To start the optimization procedure, it is necessary to determine the initial point $\hat{\boldsymbol{\theta}}_0$ and the variance σ. Let $\boldsymbol{\theta}^\star$ be a global minimum to be located. When $\hat{\boldsymbol{\theta}}_k$ is far from $\boldsymbol{\theta}^\star$, $\mathbf{r}_k$ should have a large variance to allow large displacements, which are necessary to escape the local minima. On the other hand, when $\hat{\boldsymbol{\theta}}_k$ is close $\boldsymbol{\theta}^\star$, $\mathbf{r}_k$ should have a small variance to allow exact exploration of parameter space. The idea of the ARS is to alternate two phases: variance-selection and variance-exploitation [16]. During the variance-selection phase, several successive values of σ are tried for a given number of iteration of the basic algorithm. The competing σ_i is rated by their performance in the basic algorithm in terms of cost reduction starting from the same initial point. Each σ_i is computed according to the formula:

$$\sigma_i = 10^{-i}\sigma_0, \quad \text{for} \quad i = 1, \ldots, 4 \tag{3}$$

and it is allowed for $100/i$ iterations to give more trails to larger variances. σ_0 is the initial variance and can be determined, e.g. as a spread of the parameters domain:

$$\sigma_0 = \theta_{max} - \theta_{min} \tag{4}$$

where θ_{max} and θ_{min} are the largest and lowest possible values of parameters, respectively. The best σ_i in terms the lowest value of the cost function is selected for the variance-exploitation phase. The best parameter set $\hat{\boldsymbol{\theta}}_k$ and the variance σ_i are used in the variance-exploitation phase, whilst the algorithm (2) is run typically for one hundred iterations. The algorithm can be terminated when the maximum number of algorithm iteration n_{max} is reached or when assumed accuracy J_{min} is obtained. Taking into account local minima, algorithm can be stopped when σ_4 has been selected a given number of times. It means that algorithm get stucks in local minimum and cannot escape its basin of attraction. Apart from its simplicity, the algorithm possesses the property of global convergence. Moreover, adaptive parameters of the algorithm, cause that a chance to get stuck in local minima is decreased.

4 Simultaneous Perturbation Stochastic Approximation

In recent years has been observed a growing interest in stochastic optimisation algorithms that do not depend on gradient information or measurements. This class of algorithms is based on an approximation of the gradient of the loss function. The general form of Stochastic Approximation (SA) recursive procedure is as follows [14]:

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) \tag{5}$$

where $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ is the estimate of the gradient $\partial J/\partial \hat{\boldsymbol{\theta}}$ based on the measurements of the loss function $L(\cdot)$. The essential part of this equation is the gradient approximation. The SPSA has all elements of $\hat{\boldsymbol{\theta}}$ randomly perturbed to obtain two measurements $L(\cdot)$, but each component $\hat{\boldsymbol{g}}_{ki}(\hat{\boldsymbol{\theta}}_k)$ if formed from a ratio involving the individual components in the perturbation vector and the difference in the two corresponding measurements. For two-sided simultaneous perturbation, estimation of gradient is obtained according to the formula [15]:

$$\hat{\boldsymbol{g}}_{ki}(\hat{\boldsymbol{\theta}}_k) = \frac{L(\hat{\boldsymbol{\theta}}_k + c_k\Delta_k) - L(\hat{\boldsymbol{\theta}}_k - c_k\Delta_k)}{2c_k\Delta_{ki}} \tag{6}$$

where the distribution of the user-specified p-dimensional random perturbation vector, $\Delta_k = (\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp})^T$ is independent and symmetrically distributed about 0 with finite inverse moments $E(|\Delta_{ki}|^{-1})$ for all k, i. One of the possible distributions that satisfies these conditions is the symmetric Bernoulli ± 1. Two commonly used distributions that not satisfy these conditions are the uniform and normal ones. The rich bibliography presents sufficient conditions for convergence of the SPSA ($\hat{\boldsymbol{\theta}}_k \to \boldsymbol{\theta}^\star$ in the stochastic almost sure sense). However, the efficiency of the SPSA depends on the shape of the $J(\boldsymbol{\theta})$, the values of gain sequences $\{a_k\}$ and $\{c_k\}$ and distribution of the $\{\Delta_{ki}\}$. The choice of the gain sequences is critical to the performance of the algorithm. In the SPSA the gain sequences are calculated as follows [14]:

$$a_k = \frac{a}{(A+k)^\alpha}, \quad c_k = \frac{c}{k^\gamma} \tag{7}$$

where a, c, A, α and γ are non-negative coefficients. To apply the SPSA to global optimisation, it is needed to use a stepwise (slowly decaying) sequence $\{c_k\}$ [15].

5 Simulation studies

All training methods are implemented in Borland C++ Builder™ Enterprise Suite Ver. 5.0. Simulations are performed using a PC Computer with Athlon K7 550 processor and 128 MB RAM. To check efficiency of the training methods, two examples are studied: modelling of an unknown linear dynamic system and modelling of the actuator behaviour in the first section of the sugar evaporation station in the Lublin Sugar Factory.

5.1 Modelling of an unknown dynamic system

The second order linear process under consideration is described by the following transfer function [13]:

$$G(s) = \frac{\omega}{(s+a)^2 + \omega^2} \tag{8}$$

Its discrete form is given by:

$$y_d(k) = A_1 y_d(k-1) + A_2 y_d(k-2) + B_1 u(k-1) + B_2 u(k-2) \tag{9}$$

Assuming that parameters of the process (8) are $a = 1$, and $\omega = 2\pi/2.5$, and the sampling time $T = 0.5$ s, the coefficients of the equation (9) are: $A_1 = 0.374861$, $A_2 = -0.367879$, $B_1 = 0.200281$ and $B_2 = 0.140827$. Taking into account the structure of the DNM, only one neuron with the second order IIR filter, and the linear activation function is required to model this process. Training of the dynamic neuron network was carried out using the off-line EDBP, ARS and SPSA algorithms. In order to compare different learning methods, assumed accuracy is set to 0.01 and several performance indices such as the Sum of Squared Errors (SSE), the number of Floating Operations (FO), the number of Network Evaluations (NE) and training time are observed. Learning data is generated feeding a random signal of the uniform distribution $|u(k)| \leqslant (a^2+\omega^2)$ to the process, and recording its output signal. In this way, the training set containing 200 patterns is generated. After training, the behaviour of the neural model was checked using the step signal $u(k) = (a^2 + \omega^2)/\omega$.

EDBP algorithm. The learning process was carried out off-line. To speed up the convergence of the learning, the adaptive learning rate is used. The initial value of the learning rate η is 0.005. Initial network parameters were chosen randomly using uniform distribution

from the interval $[-0.5; 0.5]$. Figure 2 shows the course of the output error. The assumed accuracy is reached after 119 algorithm iterations.

ARS algorithm. The next experiment was performed using the ARS algorithm. As in the previous example, the initial network parameters are generated randomly using uniform distribution in the interval $[-0.5; 0.5]$. The initial variance σ_0 is 0.1. In Fig. 2, one can see the error course for this example. The assumed accuracy is achieved after 9 iterations. The initial value of σ_0 is very important for convergence of the learning. When this value is too small, e.g. 0.0001, the convergence is very slow. On the other hand when the value of σ_0 is too large, e.g. 10, many cost evaluations are performed at very large variances and this results in too chaotic search. These steps are not effective for performance of the algorithm and cause that the learning time is much longer.

SPSA algorithm. The last training example uses the SPSA algorithm. The initial parameters were generated randomly with uniform distribution in the interval $[-0.5; 0.5]$. After some experiments, the algorithm parameters which assure quite fast convergence are as follows: $a = 0.001$, $A = 0$, $c = 0.02$, $\alpha = 0.35$ and $\gamma = 0.07$. The learning results are shown in Fig. 2. The assumed accuracy is obtained after 388 iterations.

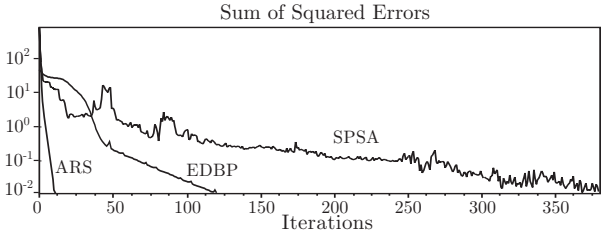

Figure 2: Learning error for different algorithms.

Comparison of the algorithms. All the considered algorithms have reached the assumed accuracy. It must be taken into account, however, that these methods need different numbers of floating operations per one iteration as well as different numbers of network evaluations, in order to calculate values of the cost function. The characteristic of algorithms are shown in Table 1. The ARS has reached the assumed accuracy at the lowest number of iteration but during one algorithm step it is required much more floating operations than for the others algorithms. It is caused by a large number of network evaluations (see Table 1). Therefore, the learning time for this algorithm is the greatest. In turn, the EDBP uses the least number of network evaluations, but calculating of a gradient is time consuming operation. In result, the simplest algorithm, taking into account the number of floating operations per one iteration, is the SPSA. However, the SPSA approximates the gradient and it is needed to perform more algorithm steps to obtain a similar accuracy as for the gradient based one. For this simple example, the EDBP is the most effective algorithm. But it should be keep in mind that the examined system is a linear one and the error surface has only one minimum. The next example shows behaviour of the learning methods for a nonlinear dynamic case.

Table 1: Characteristics of the learning methods

Characteristics	EDBP	ARS	SPSA
Learning time	2,67 sek	10,06 sek	3,995 sek
Iterations	119	9	388
SSE	0.0099	0.006823	0.00925
FO	$2.26 \cdot 10^6$	$7.5 \cdot 10^6$	$2.9 \cdot 10^6$
NE	119	2772	776
FO/iteration	$1.89 \cdot 10^4$	$8.33 \cdot 10^5$	$8.58 \cdot 10^3$
NE/iteration	1	308	2

5.2 Sugar Factory actuator

In this section, the sugar evaporation station in the Lublin Sugar Factory (Poland) is presented [10]. In a sugar factory, sucrose juice is extracted by diffusion. This juice is concentrated in a multiple-stage evaporator to produce a syrup. The liquor goes through a series of five stages of vapourisers, and in each passage its sucrose concentration increases. The first three sections are of Roberts type with a bottom heater-chamber, while the last two are of Wiegends type with a top heater-chamber. The sugar evaporation control should be performed in such a way that the energy used is minimised to achieve the required quality of the final product. The main inconvenient features that complicate the control of the evaporation process are [8]:

- a highly complex evaporation structure (large number of interacting components),
- large time delays and responses (configuration of the evaporator, their number, capacities),
- strong disturbances caused by violent changes in the steam,
- many constrains on several variables.

Figure 3 shows the first evaporation stage with four preheaters. The actuator to be modelled is the valve marked by dotted square. For this valve, $LC51\_03.CV$ denotes the control value to the valve on the juice inlet to the evaporation section (the actuator input), and $F51\_01$ is the juice flow on the inlet to the evaporation station (the actuation). With these two signals the neural model of the actuator can be defined as:

$$F51\_01 = F_N(LC51\_03.CV) \tag{10}$$

where F_N denotes the nonlinear function.

Experiment. During the experiment, the neural

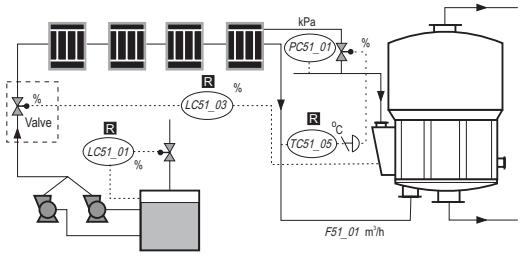

Figure 3: The first section of the evaporator.

model of the structure $N^2_{1,5,1}$ (two processing layers, one input, five neurons in hidden layer and one output), is trained using in turn the EDBP, ARS and SPSA methods. Taking into account dynamic behaviour of the valve each neuron in the network structure possesses first order filter. The model of the valve is identified using real process data from the sugar factory recorded during the sugar campaign in October 2000. In the sugar factory control system, the sampling time is equal to 10 s. Thus, during one work shift (6 hours) approximately 2 160 training samples per one monitored process variable are collected. For many industrial processes, measurement noise is of a high frequency [8]. Therefore, to eliminate this noise, a low pass filter of the Butterworth type of the second order was used. Moreover, the input samples were normalized to zero mean and unit standard deviation. In turn, the output data should be transformed taking into consideration the response range of the output neurons. For the hyperbolic tangent activation function, this range is [-1;1]. To perform such kind of transformation, the simple linear scalling can be used. Additionally, to avoid saturation of the activation functions, the output was transformed into the range [-0.8;0.8]. It is necessary to notice that, if the network will be used with other data sets, it is required to memorise maximum and minimum values of the training sequence. To perform experiments two data sets were used. The first set, containing 500 samples, was used for training and another one, containing 1 000 samples, was used to check the generalisation ability of the networks.

EDBP algorithm. The algorithm was run over 20 times with different initial points. The learning process was carried out off-line performing 5 000 steps. To speed up the convergence of learning, the adaptive learning rate is used. The initial value of the learning rate η is 0.005. The obtained accuracy is 0.098. To check quality of the modelling, the neural model is tested using another data set of 1 000 samples. Figure 4 shows the testing phase of the neural model. As it can be seen, the generalization abilities of the dynamic network are quite good.

ARS algorithm. Many experiments were performed to find the best value of initial variance σ_0. Eventually, this value was found to be $\sigma_0 = 0.05$. With this initial

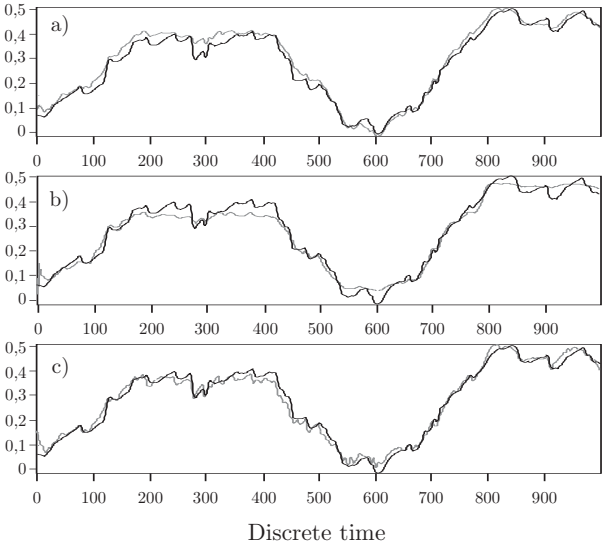

Figure 4: Testing phase: EDBP (a), ARS (b) and SPSA (c). Valve (black), neural model (grey).

variance algorithm was carried out for 200 iterations. The modelling results for the testing set are presented in Fig. 4. The characteristics of the algorithm are included in Table 2. The ARS is time consuming, but it can find a better solution than the EDBP. The influence of the initial network parameters are examined also. The most frequently used range of the parameters values is $[-1; 1]$. The simulations show that more narrow intervals e.g. [-0.7;0.7] or [-0.5;0.5] assure faster convergence.

SPSA algorithm. This algorithm is a simple and very fast procedure. However, choice of the proper parameters is not a trivial problem. There are 5 parameters which have a crucial influence on convergence of the SPSA. In spite of speed of the algorithm, a user should spend a lot of time to select proper values. Sometimes it is very difficult to find good values and algorithm fails. The experiment was carried out for 7 500 iterations using the following parameters $a = 0.001$, $A = 100$, $c = 0.01$, $\alpha = 0.25$ and $\gamma = 0.05$. The modelling results for the testing set are presented in Fig. 4. The parameter γ controls the decreasing ratio of sequence $\{c_k\}$ and is set to a small value to enable a property of global optimisation. Parameter a is set to a very small value to assure convergence of the algorithm. The dynamic neural network is very sensitive to large changes in parameters values (dynamic filters) and large values of a like 0.4 can cause that learning process will be divergent. Taking into account that the first value of the sequence a_k is small, the parameter α is set to 0.25 (the optimal value is 1, Spall [15] proposes to use 0.602). In spite of difficulties in selection of the network parameters, the modelling results are quite good. Moreover, generalisation ability for this case is better than for both the EDBP and the ARS.

Table 2: Characteristics of the learning methods

Characteristics	EDBP	ARS	SPSA
Learning time	12.79 min	33.9 min	10.1 min
Iterations	5 000	200	7 500
SSE – training	0.098	0.07	0.0754
SSE – testing	0.564	0.64	0.377
FO	$3.1 \cdot 10^9$	$1.6 \cdot 10^{10}$	$2.5 \cdot 10^9$
NE	5 000	61 600	15 000
FO/iteration	$6.2 \cdot 10^5$	$8 \cdot 10^7$	$3.3 \cdot 10^5$
NE/iteration	1	308	2

Comparison of the methods. The characteristics of the learning methods are shown in Table 2. The best accuracy was obtained using the ARS. Slightly worse result was achieved using the SPSA and the worst quality was obtained using the EDBP. In this example, the actuator is described by nonlinear dynamic relation and the algorithms belonging to the global optimisation techniques performed their task with better quality than gradient based one. Simultaneously, the SPSA is much faster than the ARS. Moreover, concerning the same accuracy, the generalisation ability of the neural model trained by the SPSA is better than the neural model trained by the ARS.

6 Concluding remarks

The main objective of this work was to perform comparative studies between some stochastic methods and the off-line version of the gradient based algorithm to train dynamic neural networks using both simulated and real data. The performed simulations show that stochastic approaches can be an effective alternative to gradient based methods. The ARS is a very simple algorithm. This algorithm can be very useful in practice for engineers, because a user should only determine one parameter to start the optimisation procedure. In contrary, the SPSA is much faster than the ARS, but to start the optimisation process, five parameters should be determined. To define these values, a user should possesses a quite large knowledge about this method, to use it properly. Taking into account the property of global optimisation, both stochastic approaches can be effectively used for modelling of nonlinear processes.

References

[1] A. Alessandri and T. Parisini. Nonlinear modelling of complex large-scale plants using neural networks and stochastric approximation. *IEEE Trans. Systems, Man, and Cybernetics – A*, (27):750–757, 1997.

[2] J. Chen and R. J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic Publishers, Berlin, 1999.

[3] Z. Fathi, W. F. Ramirez, and J. Korbicz. Analytical and knowledge-based redundancy for fault diagnosis in process plants. *AIChE J.*, 39(1):42–56, 1993.

[4] P. M. Frank and B. Köppen-Seliger. New developments using AI in fault diagnosis. *Artificial Intelligence*, 10(1):3–14, 1997.

[5] J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, Inc., New York, 1999.

[6] R. Isermann, editor. *Supervision, Fault Detection and Diagnosis of Technical Systems – Special Section Control Engineering Practice* **5**(5). 1997.

[7] J. Korbicz, K. Patan, and A. Obuchowicz. Dynamic neural networks for process modelling in fault detection and isolation systems. *International Journal of Applied Mathematics and Computer Science*, 9(3):519–546, 1999.

[8] S. Lissane Elhaq, F. Giri, and H. Unbehauen. Modelling, identification and control of sugar evaporation – theoretical design and experimental evaluation. *Control Engineering Practice*, (7):931–942, 1999.

[9] K. Patan. *Artificial Dynamic Neural Networks and Their Application in Modelling of Industrial Processes*. Ph.D. dissertation., Warsaw University of Technology, Faculty of Mechatronics, Warszawa, 2000.

[10] K. Patan and J. Korbicz. Application of Dynamic Neural Networks in an Industrial Plant. In *Proc. SAFEPROCESS'2000, Budapest, Hungary*, pages 186–191, 2000.

[11] R. J. Patton, P. M. Frank, and R.N. Clark. *Issues of Fault Diagnosis for Dynamic Systems*. Springer-Verlag, Berlin, 2000.

[12] R. J. Patton and J. Korbicz, editors. *Advances in Computational Intelligence* – Special Issue International Journal of Applied Mathematics and Computer Sceince **9**(3). 1999.

[13] D. T. Pham and X. Liu. Training of Elman networks and dynamic system modelling. *International Journal of Systems Science*, 27:221–226, 1996.

[14] J.C. Spall. Multivariate stochastic aproximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Automatic Control*, (37):332–341, 1992.

[15] J.C. Spall. Stochastic optimization, stochastic approximation and simulated annealing. In J.G. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering*, New York, 1999. John Wiley & Sons.

[16] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Springer, London, 1996.