TWO APPLICATIONS OF ENG-GENES BASED NONLINEAR IDENTIFICATION

Patrick Connally, Kang Li, George W. Irwin

Intelligent Systems & Control Group, School of Electrical & Electronic Engineering
Queen's University Belfast, UK

Abstract: Nonlinear identification using a novel neural network paradigm, namely eng-genes, is investigated. A set of MATLAB functions for the training and simulation of eng-genes based neural models are described. These functions are then used to investigate the effectiveness of the technique applied to two nonlinear dynamical systems. Experimental data from a pH neutralisation plant and simulation data from a physical model of a CSTR process are used to generate 'eng-genes' models. The results are compared with conventional neural models of these plants, showing that simple neural models with better performance and improved transparency are obtainable using the eng-genes paradigm. *Copyright © 2005 IFAC*

Keywords: Neural networks, neural network models, nonlinear systems, system identification.

1. INTRODUCTION TO ENG-GENES

1.1 Nonlinear System Modelling

Current system modelling techniques fall into three broad categories: white-box models, black-box models and grey-box models. White-box modelling techniques, such as Computational Fluid Dynamics (CFD), use the physical equations governing the system in order to predict its behaviour under given conditions. While this method is very accurate provided the system equations are sufficiently well-known, it is very computationally intensive for all but the simplest of systems, rendering it unsuitable for real-time use and unwieldy for many offline applications (Thompson and Li, 2003). Black-box models, such as linear and nonlinear autoregressive models and conventional neural networks, are derived from plant data and may be as simple or complex as necessary, with obvious trade-offs in accuracy and complexity. However, due to the absence of a mechanism for including 'a priori' physical knowledge, the accuracy of a black-box model of a given complexity is very strongly dependent on the quality of the training data used.

Grey-box techniques such as fundamental grey-box (FGB) modelling (Li, *et al*., 2004), attempt to include the advantages of both of the above concepts by incorporating some 'a priori' system knowledge into a black-box type model structure. In some grey-box techniques, physical modelling and system identification are combined to form two intersecting paths (Bohlin, 1994). However, such methods would assume that the model structure is at least partially known a priori. The modelling task is therefore to identify the unmodelled dynamics or unknown parameters using black-box methods such as neural networks (Psichogios and Ungar, 1992). The eng-genes concept, on the other hand, incorporates known system nonlinearities into a novel neural network structure.

1.2 Eng-genes concept

As universal approximators, artificial neural networks (ANN) such as the multi-layer perceptron (MLP) and radial basis function (RBF) networks have many successful applications. These neural networks utilise a set of hidden nodes with homogeneous nonlinear activation functions in order

to give the network its capability to approximate nonlinear systems. Thus, a logical way to incorporate a priori system information into a neural network structure is to use the system's nonlinearities, or variations more suitable for neural network manipulation, as activation functions. This leads to a novel neural network paradigm – 'eng-genes' networks. These could be heterogeneous networks, in which different hidden layer nodes have different activation functions.

The most pressing problem in implementing the eng-genes concept is to select activation functions which reflect the specific nonlinearities present in the system. It is proposed to do so by encoding the set of possible functions and parameters into chromosome representations, and to evolve these solutions using genetic algorithms to produce an 'optimum' solution. This idea is behind the name 'eng-genes'.

2. ENG-GENES FRAMEWORK AND SOFTWARE

2.1 Eng-genes modelling framework

In a conventional neural network structure, once the number of hidden layers and nodes has been selected, the only adjustment made to the network is the modification of the intra-neuron connection weight and bias values. This may be achieved through standard back-propagation algorithms and is relatively straightforward. However, in an eng-genes network, not only the values of the weights, but also the type of activation function of each hidden layer neuron, may be changed in order to best approximate plant the behaviour. Thus, the following series of steps are required for eng-genes construction:

Selection of potential activation functions: These must be derived from the first principle laws for the engineering system in question. Appropriate ranges are required for the parameters appearing in these functions.

Determination of neural network structure: As in a conventional neural network, the number and size of hidden layers and the number of inputs to the network must be chosen. The set of activation functions and parameters to be used by the eng-genes network must also be selected. Due to the size of the potential solution space, genetic algorithms will be used to select functions and optimise parameters.

Training: The network created above must be trained, ie. its weights adjusted so as to provide an acceptable model. This can also be done using genetic optimisation in conjunction with structure selection if desired. Training can also be done using revised neural network training algorithms specifically designed for eng-genes models. In this paper, MATLAB functions for training eng-genes models using back-propagation principles are applied.

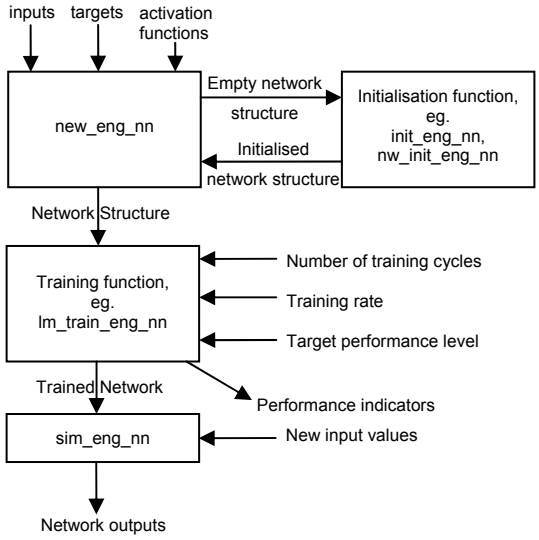

Fig. 1 Flowchart showing usage of eng-genes software (Version 1, Sept. 2004)

2.2 Software Implementation

To date, software has been implemented in the MATLAB environment to create an eng-genes network structure and to train the network from given input/output plant data. The fundamental difference between this set of functions, and others designed to manipulate neural networks in the same environment, is the capability of using different nonlinear functions in a heterogeneous hidden layer.

Fig. 1 illustrates typical usage of the new software in the form of a flowchart.

Initialisation: A new eng-genes network is created using the function *new_eng_nn*. This takes as its inputs an input matrix, a target output vector, and a vector of integers representing the hidden layer activation functions. It returns an *eng_nn* data structure.

This function may call one of two other functions to initialise the network weights. The most basic of these, *init_eng_nn*, simply initializes the members of the weight matrices to random numbers between zero and one. The other, *nw_init_eng_nn*, uses the Nguyen-Widrow method (Nguyen and Widrow, 1990) to scatter the hidden nodes evenly across the input space. This is done by initialising the weight matrix W such that

$$W = kh^{\frac{1}{n}}R \qquad (1)$$

where W is the matrix of weights, h is the number of hidden nodes, n is the number of input nodes, R is a matrix of random values with normalised rows, and $0 < k < 1$ is used to provide some overlap between neuron responses in the input space (usually k = ~0.7).

Training: The Levenberg-Marquardt algorithm (Hagan and Menhaj, 1994) is a very commonly used second-order training method which relies on a Jacobian matrix, given in equation (2).

$$\mathbf{J}(\mathbf{w}) = \begin{bmatrix} \frac{\partial e_1(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial e_1(\mathbf{w})}{\partial w_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_N(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial e_N(\mathbf{w})}{\partial w_m} \end{bmatrix} \quad (2)$$

where $\mathbf{w}$ is the vector of weights (both hidden layer and output layer), e_i are the instantaneous errors at each sample, N is the number of training samples, and m is the number of weights in the network.

In a conventional neural network, derivative values to be used in the Jacobian are back-propagated through the network according to:

$$\frac{\partial e_q}{\partial w^k(i,j)} = \frac{\partial e_q}{\partial y^k(j)} \frac{\partial f^k(x^k(j))}{\partial x^k(j)} y^{k-1}(i) \quad (3)$$

where $q = 1,\ldots,N$, $w^k(i,j)$ is the weight between node i in layer $(k-1)$ and node j in layer k, $y^k(j)$ is the output of node j in layer k, $x^k(j)$ is the sum of weighted inputs to node j of layer k, and f^k is the activation function used in layer k.

However, for the eng-genes structure, where hidden node activation functions are not necessarily homogeneous, the back-propagation formula becomes:

$$\frac{\partial e_q}{\partial w^k(i,j)} = \frac{\partial e_q}{\partial y^k(j)} \frac{\partial f_j^k(x^k(j))}{\partial x^k(j)} y^{k-1}(i) \quad (4)$$

where f_j^k is the activation function of the j^{th} node in layer k.

The weight update for each node is then given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mathbf{e}(\mathbf{w})^T \mathbf{J}(\mathbf{w}) \left(\mathbf{J}^T(\mathbf{w}) \mathbf{J}(\mathbf{w}) + \mu \mathbf{I} \right)^{-1} \quad (5)$$

where $\mathbf{e}(\mathbf{w})$ is the vector of instantaneous errors, I is an $m \times m$ identity matrix, and μ is a small scalar.

This algorithm is implemented for eng-genes networks in the function *lm_train_eng_nn*.

3. TWO APPLICATIONS OF THE ENG-GENES METHOD

3.1 Continuously Stirred Tank Reactor (CSTR)

This is a simulated continuously stirred tank reactor (CSTR) (Morningred, *et al.*, 1990), a chemical engineering process exhibiting a high degree of nonlinearity.

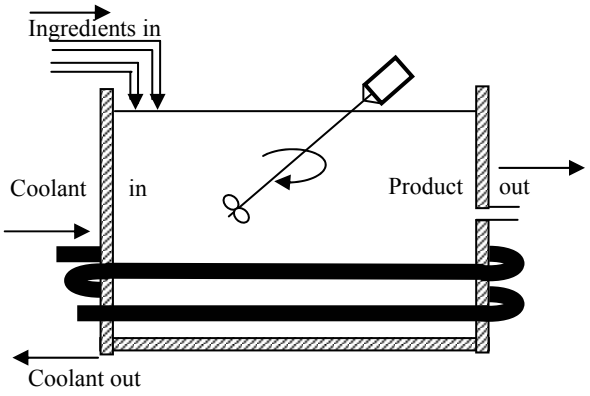

Fig. 2 Schematic of a CSTR plant

Physical Characteristics: The system is single-input, single-output, where the input variable is the flowrate of a coolant $q_c(t)$, and the output variable is the concentration of the product $C_a(t)$. The reaction is exothermic; if uncooled, the heat it generates acts to slow it down. $T(t)$ is the temperature of the solution. Fig. 2 shows a simple schematic of the plant, and equations (6) and (7) define the physical relationship between $q_c(t)$, $C_a(t)$ and $T(t)$.

$$\dot{C}_a(t) = \frac{q}{v}(C_{ao} - C_a(t)) - k_o C_a(t) \exp\left(-\frac{E}{R \cdot T(t)}\right) \quad (6)$$

$$\dot{T}(t) = \frac{q}{v}(T_o - T(t)) + k_1 C_a(t) \exp\left(-\frac{E}{R \cdot T(t)}\right) + k_2 q_c(t) \left(1 - \exp\left(-\frac{k_3}{q_c(t)}\right)\right)(T_{co} - T(t)) \quad (7)$$

where

$$k_1 = -\frac{\Delta H k_0}{\rho C_p}, \qquad k_2 = \frac{\rho_c C_{pc}}{\rho C_p v}, \qquad k_3 = \frac{h_a}{\rho_c C_{pc}}$$

A simulation of the physical plant was created within the Simulink environment using the nominal values given in Table 1.

Table 1 Nominal values for CSTR parameters

Parameter	Nominal Value
Process flowrate q	100 l/min
Reactor volume v	100 l
Reaction rate constant k_0	7.8×10^{10} min^{-1}
Activation energy E/R	1×10^4 K
Feed temperature T_0	350 K
Inlet coolant temperature T_{co}	350 K
Heat of reaction ΔH	-2×10^5 cal/mol
Specific heats C_p, C_{pc}	1 cal/g/K
Liquid densities ρ, ρ_c	1×10^3 g/l
Heat transfer coefficient h_a	7×10^5 cal/min/K
Inlet feed concentration C_{ao}	1.0 mol/l

Fig. 3 illustrates the nonlinear nature of this plant – note the increasingly oscillatory behaviour of the output as the magnitude of the input steps increases, until saturation finally occurs.

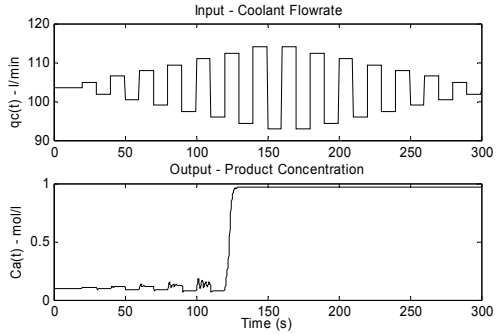

Fig.3 Nonlinearity evident in the CSTR system.

Generation of modelling data: For a steady-state output concentration of $C_a(t)$ = 0.1 mol/l, equations (6), (7) yield values of $T(t)$ = 438.54 K and $q_c(t)$ = 103.41 l/min. The system was simulated with a sampling interval of 0.2 seconds, and subjected to input consisting of uniformly distributed random perturbations in the input $q_c(t)$ over the range [-10, 10] l/min from the operating point with a zero-order-hold of 2 seconds. A normally distributed random signal was added to the output to simulate measurement noise. Statistics of the data compared to their nominal values are shown in Table 2.

Table 2 Statistics of CSTR data

Data	Nominal	Mean	Variance
$q_c(t)$ (l/min)	103.41	103.42	23.41
$C_a(t)$ (mol/l)	0.1000	0.1029	0.0005

The plant data was split into two sets; the first 100 samples were used for modelling, the remaining 2900 as test data to analyse the generalization performance of the models produced. All data was zero meaned and normalized such that its variance is equal to 1.

3.2 The pH neutralisation process

While the CSTR data was generated from physical equations, actual plant data is available for the pH neutralisation process (Hall and Seborg, 1989; Brown, *et al.*, 1997).

In this process, acid, buffer, and base solutions are mixed in a tank, similar to the CSTR plant shown in figure 1. With a constant buffer flow rate, the flow rate of the base input to the tank is used to control the pH value of the effluent.

This plant is highly nonlinear, as gain and dynamics vary significantly with operating point. Equations (8) – (11) govern the system behaviour.

$$\dot{h} = \frac{1}{A}(q_1 + q_2 + q_3 - C_v h^{0.5}) \tag{8}$$

$$\dot{W}_{a4} = \frac{1}{Ah}[(W_{a1} - W_{a4})q_1 + (W_{a2} - W_{a4})q_2 + (W_{a3} - W_{a4})q_3] \tag{9}$$

$$\dot{W}_{b4} = \frac{1}{Ah}[(W_{b1} - W_{b4})q_1 + (W_{b2} - W_{b4})q_2 + (W_{b3} - W_{b4})q_3] \tag{10}$$

$$W_{a4} + 10^{pH_4 - 14} + W_{b4}\frac{1 + 2\times 10^{pH_4 - pK_2}}{1 + 10^{pK_1 - pH_4} + 10^{pH_4 - pK_2}} - 10^{-pH_4} = 0 \tag{11}$$

Here h is the liquid level, W_{a4} and W_{b4} are the reaction invariants of the effluent stream, and q_1, q_2 and q_3 are the acid, buffer and base flowrates respectively. The pH measurement is assumed to be delayed by a transportation lag β so $pH_{4m} = pH_4(t - \beta)$. The output time delay is 0.5min and the sampling rate used is 0.25 min.

A data set of 2100 samples was available from the experimental plant, from which 500 samples were used for training, and 1600 for validation. Nominal values for all parameters are given in Table 3. All input data was zero meaned and normalised.

Table 3 Nominal pH system operating conditions

A = 207 cm^2	$W_{b2} = 3\times10^{-2}$ M
C_v = 8.75 ml cm^{-1} s^{-1}	$W_{b3} = 5\times10^{-5}$ M
pK_1 = 6.35	q_1 = 16.6 ml s^{-1}
pK_2 = 10.25	q_2 = 0.55 ml s^{-1}
$W_{a1} = 3\times10^{-3}$ M	q_3 = 15.6 ml s^{-1}
$W_{a2} = -3\times10^{-2}$ M	h = 14.0 cm
$W_{a3} = -3.05\times10^{-3}$ M	pH_4 = 7.0
W_{b1} = 0	θ = 0.5 min

4. RESULTS

4.1 CSTR results

On inspection of equations (6) and (7) governing the CSTR system, it becomes evident that the plant derives its nonlinear nature from the Arrhenius terms of the form given in equation (12).

$$k = A\times\exp\left(\frac{-E_a}{RT}\right) \tag{12}$$

Here k is the rate coefficient, A is a constant, E_a is the activation energy, R is the universal gas constant, and T is the temperature (in degrees Kelvin). The activation function given in equation (13) was chosen for use in the hidden-layer neurons.

$$y = \exp\left(\frac{-a}{x+b}\right) \tag{13}$$

where y is the neuron output and x is the weighted sum of neuron inputs. The set of values for the constant parameters a and b which have been chosen by genetic optimisation for this case study are given in Table 4.

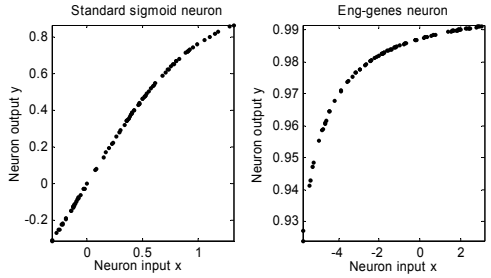

Fig. 4 Standard MLP neuron vs. eng-genes neuron

Table 4 CSTR activation function parameters

Neuron	*a* value	*b* value
1	8.8469×10^{-2}	6.8697
2	8.4694×10^{-3}	7.6782
3	2.0781×10^{-1}	6.2040

The set of inputs to the neural networks to be created was also chosen via genetic algorithms, yielding the set $q_c(t-2)$, $q_c(t-4)$, $q_c(t-5)$, $C_a(t-1)$, $C_a(t-5)$. Neural networks with two and three hidden neurons were created, of both eng-genes and MLP types. The eng-genes networks used hidden nodes with activation functions of the form given in equation (13). The two hidden node eng-gene network used parameters from rows 1 and 2 in Table 4, and the three hidden node network used parameters from rows 1, 2 and 3.

Figure 4 shows the distribution of input-output pairs from a standard sigmoidal MLP neuron and a neuron of type 1 in Table 4 after the networks had been trained for 5 epochs. Because of the less bounded nature of the eng-gene response, the networks were initialised using the Nguyen-Widrow method with the value of k (see equation (1)) reduced to 0.1 in order to begin training from smaller weight values.

Five single hidden layer networks of both types (eng-genes and MLP) with 2 or 3 hidden neurons were trained for five epochs each, and evaluated for both one-step-ahead and long-term prediction on both the modelling and validation sets. Table 5 gives the average rooted mean squared errors (RMSE) for the predictions for each of the two network paradigms.

Table 5 CSTR modelling results

	One step ahead		Long-term	
	Mod.	Val.	Mod.	Val.
Eng (5,2,1)	0.0046	0.0063	0.0057	0.0078
MLP(5,2,1)	0.0047	0.0065	0.0062	0.0082
Eng (5,3,1)	0.0044	0.0058	0.0048	0.0065
MLP(5,3,1)	0.0045	0.0070	0.0063	0.0091

As can be seen in Table 5, the eng-genes network shows an improvement over the MLP in this application, with only a few hidden nodes used. Moreover, as the activation function for the eng-genes representation was extracted from the engineering equations governing the CSTR, this a priori information was preserved in the eng-genes network.

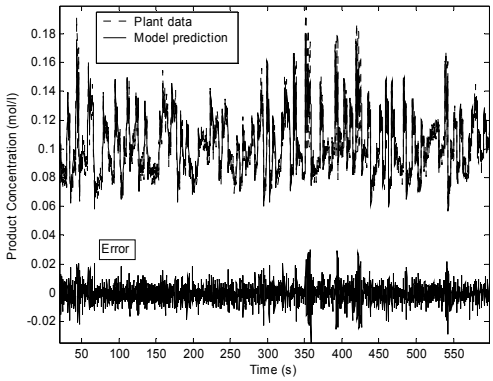

Fig. 5 Long-term prediction performance of a (5,3,1) eng-genes network

Fig. 5 shows the long-term prediction performance of the eng-genes network over the validation data.

4.2 pH neutralisation process results

Inspection of equations (8) – (11) resulted in the selection of two activation functions. The function given in equation (14) reflects the power function present in the system in equation (8).

$$y = \left(\frac{x + a}{20} \right)^b \tag{14}$$

As before, y is the neuron output, x is the neuron input, and a and b are constant parameters. The second activation function is given in equation (15), and is used to reflect the relationship of the output pH_4 with the other variables.

$$y = \log_{10}\left(\frac{x + c}{6} \right) \tag{15}$$

Here c is a constant parameter.

Sets of a, b and c values were again obtained using genetic optimisation, and are given in table 6. The network inputs were $q_3(t-1)$, $q_3(t-4)$ and $pH_4(t-1)$.

Table 6 Activation functions for the pH process

Neuron	Type	*a* value	*b* value	*c* value
1	Log	-	-	10.084
2	Power	12.835	1.128	-
3	Power	13.529	0.560	-

Again, both eng-genes and MLP networks were created, containing both 2 and 3 hidden nodes. Stability considerations in this case required the use of small initial weight values.

Five networks of each type were trained until the one-step-ahead root-mean-square error fell below 0.5. Table 7 gives the average rooted mean squared errors (RMSE) for each type of network for one-step-ahead and long-term prediction using both the modelling and validation datasets.

Table 7 pH Process modelling results

	One step ahead		Long-term	
	Mod.	Val.	Mod.	Val.
Eng(3,2,1)	0.4930	0.7889	0.7834	1.2502
MLP(3,2,1)	0.4063	0.6813	1.0617	1.3553
Eng(3,3,1)	0.4451	0.5852	0.7003	0.9498
MLP(3,3,1)	0.3965	0.7317	0.8681	1.3444

While the MLP generally performs better in the one-step-ahead case, the eng-genes network is consistently more accurate in the long-term case, which is a more rigorous test of modelling accuracy. Figure 6 shows the long-term prediction performance of the (3,3,1) eng-genes network and the (3,3,1) MLP network over the validation data.

5. CONCLUSION AND FUTURE WORK

This paper has described briefly the concept behind the eng-genes modelling framework, before giving a detailed description of a MATLAB software toolbox for creating and manipulating eng-genes heterogeneous neural networks. Two nonlinear engineering systems were then introduced and described, and data from these systems obtained for use in a case study to verify the modelling potential of the eng-genes technique. Activation functions were derived from known system equations, and these were used to construct eng-genes networks to be used in long-term modelling of the systems. Standard multi-layer perceptron networks were also constructed to act as a point of comparison.

Note that in the CSTR case, as the data were generated from the physical equations, the long-term prediction performances of both the eng-genes and MLP are satisfactory. With the pH plant, as the data were acquired from the real system, many unaccounted factors will have coloured the data, thus producing a satisfactory neural model is more difficult. In both cases, however, the eng-genes networks generally performed better than the MLP's.

Future work will include refining the activation functions for the eng-genes framework and testing its on-line performance.

ACKNOWLEDGEMENTS

Patrick Connally wishes to acknowledge the financial support of the European Social Fund. Dr K. Li wishes to acknowledge the financial support of the UK Engineering and Physical Sciences Research Council (EPSRC Grant GR/S85191/01).

REFERENCES

Bohlin, T. (1994). A case study of grey-box identification. *Automatica,* **30,** pp. 307-318.

Brown, M.D., G. Lightbody, and G.W. Irwin (1997). Nonlinear internal model control using local model networks. *IEE Proc. Control Theory Appl.,* **144**, pp. 505-514.

Hagan, M.T. and M.B. Menhaj (1994). Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks,* **5,** pp. 989-993.

Hall R.C. and D.E. Seborg (1989). Modelling and self-tuning control of a multivariable pH neutralisation process. Part I: Modelling and multiloop control, *Proc. of American Control Conf*, pp. 1822-1827.

Li, K., S. Thompson and J. Peng (2004). Modelling and prediction of NOx emission in a coal-fired power generation plant. *Control Engineering Practice*, **12,** pp. 707-723.

Morningred J.D. *et al* (1990). An adaptive nonlinear predictive controller. *Proc. of the American Control Conference*, **2**, pp. 1614-1619.

Nguyen, D., and B. Widrow (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *1990 International Joint Conference on Neural Networks.* **3,** pp. 21-26.

Psichogios, D.C., and L.H. Ungar (1992). A Hybrid Neural Network – First Principles Approach to Process modelling. *AIChE Journal*, **38**, pp. 1499-1511.

Thompson, S. and K. Li (2003). Modelling of NOx emission, In: *Thermal power plant simulation, monitoring and control.* (D. Flynn. (Ed)) pp: 243-268. IEE, London.

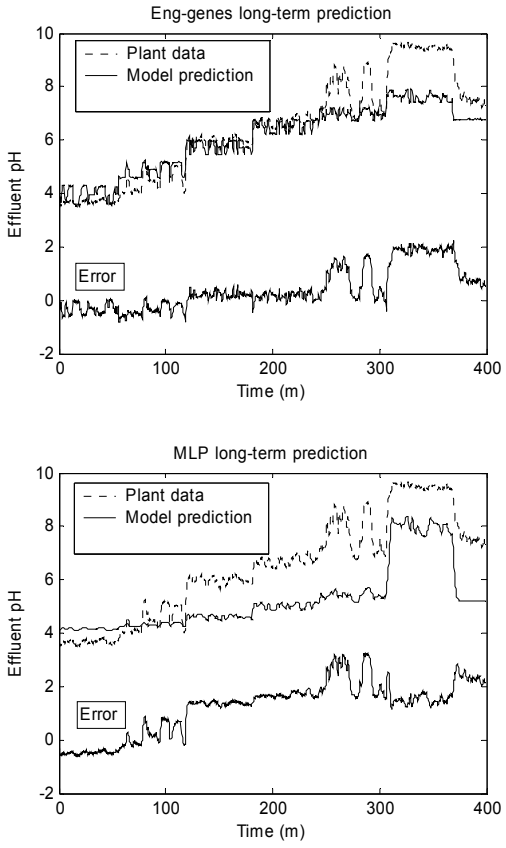

Fig. 6 Long-term prediction performance comparison of a (3,3,1) eng-genes network and a (3,3,1) MLP network