Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Chemical Engineering 27 (2003) 143–155
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Dynamic neural networks partial least squares (DNNPLS) identification of multivariable processes

Olufemi A. Adebiyi [1], Armando B. Corripio *

Gordon A. and Mary Cain Department of Chemical Engineering, Louisiana State University, Baton Rouge, LA 70803-7303, USA

Accepted 4 March 2002

Abstract

This paper presents the dynamic neural networks partial least squares (DNNPLS) as a strategy for open-loop identification of multivariable chemical processes that circumvent some of the difficulties associated with multivariable process control. The DNNPLS is an extension of the neural networks' partial least squares (NNPLS) developed by Qin and McAvoy (Comp. Chem. Eng. 20 (1992) 379). Here, a dynamic extension to the NNPLS algorithm is proposed in which the static neural network models in the latent space (inner relationship) are replaced by dynamic neural network models. Though this approach has previously been dismissed as being sub-optimal (Am. Inst. Chem. Eng. J. 38 (1992) 1593; Chem. Eng. Sci. 48 (1993) 3447) in terms of the outer relationship (relationship between the residuals), Lakshminarayanan et al. (Am. Inst. Chem. Eng. J. 43 (1997) 2307) have shown that this sub-optimality problem comes into prominence only when no attention is placed on the design of the plant probing signals. As illustrations, the DNNPLS identification strategy is implemented on simulations of a model IV fluid catalytic cracking unit (FCCU) and of an isothermal reactor. In both cases, it is shown that the methodology is capable of modeling the dynamics of the chemical processes and an improved performance is achieved over that of the PLS-ARMA (Comp. Chem. Eng. 20 (1996) 147) for the isothermal reactor.

Keywords: Neural networks; PLS; Dynamics; Latent space scores; Non-linear process modeling

Nomenclature

B	a matrix of sensitivities form the regression (size $f \times f$)
b_h	the regression coefficient for one PLS component
C_B	concentration of B in reactor example
d	the number of samples in the training set
dprr	differential pressure between reactor and regenerator pressure
E_h	the residual of X after subtraction of h components (size $d \times m$)
f	the number of factors used
F_5	flow of fuel to feed furnace
F_h	the residual of Y after subtraction of h components (size $d \times n$)
F_T	air flow rate into regenerator
h	a dummy index for counting factors
i	a dummy index counting samples
I_n	the identity matrix of size $d \times d$
j	a dummy index for counting independent (x) variables
k	a dummy index for counting dependent (y) variables

* Corresponding author. Tel.: +1-225-388-1426; fax: +1-225-388-1476
[1] Currently with Accenture, Atlanta, GA, USA.

0098-1354/02/$ - see front matter
PII: S0098-1354(02)00034-0

m	the number of independent (x) variables
n	the number of dependent (y) variables
p_h^T	a row vector of loadings for the X block, factor h (size $1 \times m$)
p_4	reactor pressure
P^T	a row vector of loadings (size $f \times m$)
q_h^T	a row vector of loadings for the Y block, factor h (size $1 \times n$)
Q^T	the matrix of Y loadings (size $f \times n$)
r	the number of samples in a prediction (test) set
r_h	residual after factor h has been extracted
T	the matrix of X scores (size $d \times f$)
t_h	a column vector of scores for the X block, factor h (size $d \times 1$)
t_r	reactor temperature
U	the matrix of Y scores (size $d \times f$)
u_h	a column vector of scores for the Y block, factor h (size $d \times 1$)
v_{11}	reactor pressure valve position
v_{14}	regenerator stack gas valve position
v_{lift}	lift air blower steam valve position
w_h^T	a row vector of weights for the X block, factor h (size $1 \times m$)
x	a column vector of features for the independent variables (size $m \times 1$)
X	a matrix of features for the independent variables (size $d \times m$)
xt	the number of delayed input loadings used in neural network input
y	a column vector of features for the dependent variables (size $n \times 1$)
Y	a matrix of features for the dependent variables (size $d \times n$)
yt	the number of delayed output loadings used in neural network input
Δt	the sampling interval

1. Introduction

Many chemical processes exhibit nonlinear behavior which creates difficulties in process identification. In multivariable processes, unknown model structures, difficulties in obtaining process and measurement noise characteristics, and high correlation between process variables (MacGregor, Marlin, Kresta & Skagerberg, 1991) are examples of problems that are faced daily. Process identification of a multi input–multi output (MIMO) process could be very difficult, especially when the process exhibits high non-linearities with the presence of several time scales and time delays.

Partial least-squares (PLS) is a linear system identification method that projects the input–output data down into a latent space, extracting a number of principal factors with an orthogonal structure, while capturing most of the variance in the original data (Wold, 1984). PLS was first used in econometrics. Typical examples are in multivariate calibration (Lorber & Kowalski, 1998), structure–activity relationships (Dunn, Wold, Edlund, Hellberg & Gasteiger, 1984) and optimization of complex processes. It has also been applied to chemical engineering problems, such as process monitoring, modeling and fault detection (Ricker, 1988; Wise, 1991; Kresta, 1992; Qin & McAvoy, 1992). The approach also offers potential benefits for dynamic modeling and control (Lakshminarayanan, Shah & Nandakumar, 1997).

When the process exhibits non-linear behavior, it is desirable to extend the PLS model structure to capture non-linearities. Wold, Kettaneh-wold and Skagerberg (1989) incorporated non-linearities into the PLS model for modeling static data using a quadratic relationship between the latent scores. Qin and McAvoy (1992) have demonstrated the modeling of static data using an integrated PLS-neural net structure called NNPLS. The NNPLS uses neural networks to model the latent scores (inner relationship), hence it is able to handle correlated variables and nonlinear relationships directly. The static NNPLS combines the robust properties of the PLS and the continuous nonlinear modeling of neural networks.

Dynamic models are required for process control. Ray and Kaspar (1992) developed a dynamic PLS by filtering the input data and analyzing the major dynamic data removed using PLS. Other attempts at incorporating dynamics have been including large numbers of lagged values of the input variables in the input data matrix, called the PLS-FIR model (Ricker, 1988) and including lagged values of both the inputs and the outputs in the input data matrix, called the PLS-ARMA model. These approaches require a substantial increase in the dimensions of the input data matrix. Due to the presence of lagged variables, resulting models may be cumbersome to manipulate (Ray & Kaspar, 1993). Lakshminarayanan et al. (1997) and Lakshminarayanan, Patwardhan and Shah (1998) employed Hammer-

stein models for the inner relationship for modeling and control purposes. The parameters of the linear dynamic part and the static nonlinear part are identified separately and sequentially. The technique was illustrated using a distillation column and an acid-base neutralization system.

The DNNPLS presented here captures the dynamic and nonlinear relationship between the input and output data using the latent scores. A dynamic neural network models the present output latent score using lagged values of both the input and output latent scores as its inputs. Hence the input data matrix has no lagged values of both the process inputs and outputs as commonly done. This results in models easier to use because of the reduced dimensions of the input and output data matrices.

2. Neural networks

This section presents the structure of the neural network models used in this work. For a more comprehensive review, the reader is referred to the paper by Lippman (1987).

In general, neural networks are characterized by the networks' topology, computational characteristics of its processing elements and the training rule. The network consists of processing elements called neurons, arranged in layers and exhaustively interconnected. These interconnections produce the robust predictive qualities of neural networks. This study uses the feed-forward neural network commonly called multi-layer preceptron, which means that the output of the network is not included as input to any of the network's neurons.

A typical network consists of an input layer where each neuron has a single input, the external input, one or more hidden layers with each neuron having inputs from all the neurons in the previous layer, plus a bias, and an output to all the neurons in the following layer and an output layer in which each neuron has inputs from the last hidden layer and its output is one of the external outputs of the network. Fig. 1 shows a three-layer network—input, hidden and output—with m inputs and n outputs.

The purpose of a neural network is to obtain a mapping from an input vector X of length m to an output vector Y of length n. The user specifies the number of neurons in the hidden layer. It has been shown that the accuracy of the mapping is relatively insensitive to the number of neurons in this layer as long as some minimum number is used.

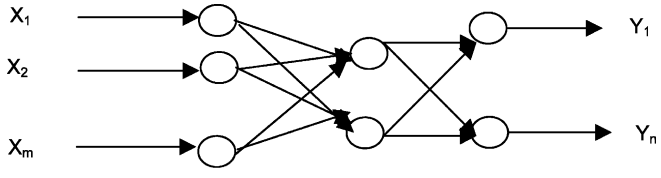

Fig. 1. Topology of a 3-layer neural network.

A neuron acts as a combining function and an activation function. The combining function produces the weighted sum of the neuron inputs, while the activation function performs a nonlinear transformation of this sum. This work uses a tangent sigmoid activation function for the neurons in the hidden layer and a linear function for the output layer.

Since the purpose of the network is to model a nonlinear dynamical system, dynamics has to be incorporated into the network by letting the input vector X be a combination of N_u previous inputs and N_y previous outputs and requiring that the network output match the system output at one sampling time into the future. Hence

$$X = [y(k-1), \ldots, y(k-N_y), u(k-d-1), \ldots, u(k-d-N_u)]^T \quad (1)$$

where d is the process time delay and k is the time index. Therefore, the network general model structure is given as

$$\hat{Y}(k) = \phi(X) \quad (2)$$

where ϕ is the nonlinear mapping function provided by the network.

The final part of the neural network definitions is to specify the training rule that is an algorithm necessary to determine the connecting weights of the model. Initially, the weights are usually set to small random values. During a training cycle, the network is presented with training vectors that consist of sampled values of input X and target Y. The output of the network is the predicted value of the plant output, $\hat{Y}$.

Training produces the optimal connection weights for the network by minimizing a quadratic cost function of the errors between the neural network output and the plant output over the entire set of samples. The training algorithm used here is the Levenberg–Marquandt optimization routine of MATLAB, which is more efficient and faster than the popularly used back propagation algorithm because it takes implicitly into account second derivatives.

3. DNNPLS identification methodology

3.1. PLS identification

A brief description of the PLS methodology and its extension to the DNNPLS is outlined below. Detailed descriptions of the PLS can be found in Geladi and Kowalski (1986) and Martens and Naes (1989).

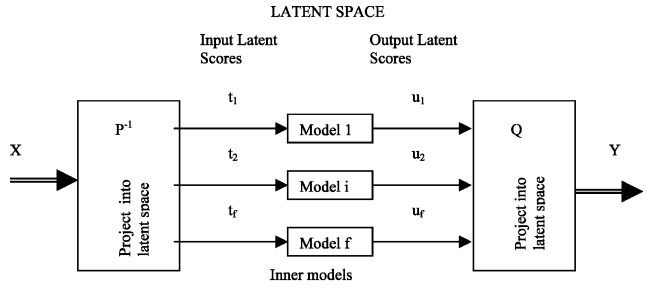

Fig. 2. Schematics of the PLS model.

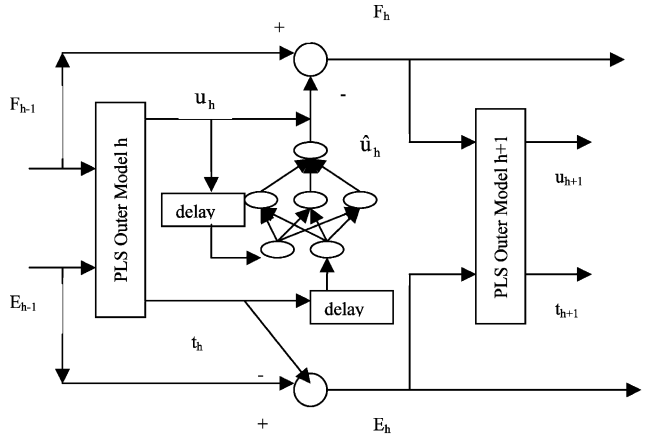

Fig. 3. Schematic of the DNNPLS training algorithm (Qin and McAvoy, 1996).

Assume we have n output variables, y_i $(i=1, 2, \ldots, n)$ and m input variables, x_i $(i=1, 2, \ldots, m)$. Also assume d samples of data are observed, then two matrices can be formulated, one for the output Y, and one for the input X, as follows:

$$y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \cdots & & & \\ y_{d1} & y_{d2} & \cdots & y_{dn} \end{bmatrix} \in \Re^{dxn} \quad (3)$$

$$x = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & & & \\ x_{d1} & x_{d2} & \cdots & x_{dm} \end{bmatrix} \in \Re^{dxm} \quad (4)$$

where each row is composed of observations from one sample instant and each column contains the dynamic observations of a variable from time instant 1 to time instant d. The MIMO system identification problem involves extracting all the relevant information in matrix X (excluding correlations) represented by some function for predicting Y. The matrices X and Y are decomposed into bilinear models similar to principal components (projections/outer relations of the PLS model):

$$X = t_1 p_1^T + t_2 p_2^T + \cdots + t_f p_f^T + E_f \quad (5)$$

$$Y = u_1 q_1^T + u_2 q_2^T + \cdots + u_f q_f^T + F_f \quad (6)$$

where t and u are the latent scores, p and q represent the latent loadings, f the number of factors extracted in the latent space, E_f and F_f are residuals for X and Y matrices, respectively after f factors have been extracted. Appropriate dimensions are defined in the nomenclature. The number of factors (f) is selected either statistically or heuristically and each factor is extracted sequentially from factor 1 to factor f. The outer relationship is defined as the relationship between E_i and F_i, $(i=1, \ldots, f)$ where initially $E_0 = X$ and $F_0 = Y$.

In matrix form Eq. (5) and Eq. (6) are written as:

$$X = TP^T + E_f \quad (7)$$

$$Y = UQ^T + F_f \quad (8)$$

where T and U are the score matrices, and P and Q are the loading matrices of matrices X and Y, respectively.

Fig. 2 shows a schematic of the PLS model. It shows that the input X is projected into the latent space by the input-loading matrix P obtaining the input scores T. Similarly, the output Y is projected into the latent space by the output-loading matrix Q obtaining the output scores U. A linear model captures the relationship between the input and output latent scores for each factor.

The loading vectors p_i $(i=1, 2, \ldots, f)$ and q_i $(i=1, 2, \ldots, f)$ represent the cosines of the dominant directions and the projection of the X and Y data on p_i and q_i, respectively results in t_i and u_i $(i=1, 2, \ldots, f)$. The linear combinations must account for much of the variation of X and must correlate well with the variables in the Y space to achieve the objectives of model stability and goodness of fit (Lakshminarayanan et al., 1997).

The procedure of determining the scores and loading vectors is carried out sequentially from the first factor to the fth factor. Scores and loading vectors for each factor are calculated from the previous residuals (outer relationship). This is performed until the number of required factors is extracted or the residual is below some threshold.

Fig. 3 shows the schematic of the model development algorithm for the DNNPLS. The schematic is equivalent to the PLS except that a neural network model replaces the linear model. Initially $(h=1)$, the residuals are the matrices X and Y because nothing has been extracted. Hence, $E_0 = X$ and $F_0 = Y$, as previously defined. The hth factor is extracted by projecting the residuals into the latent space, obtaining u_h and t_h (the latent scores of factor h).

A link between matrices X and Y is established by linear regression (least squares) between u_h with t_h. This link is known as the inner relationship.

$$\hat{u}_h = b_h t_h \tag{9}$$

In matrix notation

$$\hat{U} = TB \tag{10}$$

Hence, from Eq. (8)

$$\hat{Y} = TBQ^T + F_f \tag{11}$$

where B is a diagonal matrix with elements expressing the relationship between individual u_h and t_h scores. B is estimated such as to minimize the norm $||F_f||_2$.

The advantage of PLS over other linear regressions is that the outer model projects the original data into latent factors, by which the orthogonality of the score vectors is achieved (Höskuldsson, 1998).

3.2. DNNPLS identification

When modeling the dynamic behavior of non-linear processes it is desirable to capture the dynamics and the non-linearities in the DNNPLS model. This can be achieved by modifying the link of the inner relationships (Eq. (9)). The score vectors are visualized as realizations of certain functions of discrete time $t_i(k\Delta t)$ and $u_i(k\Delta t)$, where Δt is the sampling interval and k is the sample number (Ray & Kaspar, 1993). Hence k is equivalent to the row number of X or Y. Modeling the inner relationship by a dynamic neural network instead of a linear regression equation incorporates nonlinear dynamical information into the PLS model. The inner relationships are modeled by the nonlinear regressor as:

$$\hat{u}_h(k) = \Phi[t_h(k-1),\ t_h(k-2), \ldots,\ t_h(k-xt),\ u_h(k-1),\ u_h(k-2), \ldots,\ u_h(k-yt)] + \varepsilon \tag{12}$$

where the function $\Phi(.)$ represents the neural network model function and ε is the residual.

The network models the present output score using lagged values of the input and output scores, where xt is the number of lagged input scores used and yt is the number of lagged output scores used. MATLAB's Levenberg–Marquandt optimization routine, which is faster and more robust than the commonly used back-propagation method, was used to train the network. Parameters of the neural network model are chosen by cross validation (Section 3.4). The main advantage of using a dynamic neural network model is that it reduces the multivariable problem to a set of single-variable problems. Because of this, single-input single-output model-based controllers can easily be incorporated into the structure.

The residuals can now be calculated from the equations:

$$E_h = E_{h-1} - t_h p_h^T \tag{13}$$

$$F_h = F_{h-1} - \hat{u}_h q_h^T \tag{14}$$

where $\hat{u}_h$ is the predicted output score of the model. The process is performed recursively until the fth factor is extracted. The present loadings and scores of a factor have to be obtained before the present residuals are estimated.

The dynamic approach here differs from other approaches in that the dynamic has been moved from the original space into the latent space and also no filtering of input or pre/post compensators are used. By using neural networks as the inner regressors, only one SISO network is trained at a time, which is easier to train than MIMO networks and also prevents over-parameterization and local minima problems by reducing the complexity of the optimization problem.

In conclusion, this section presents the DNNPLS identification structure that decomposes the original MIMO problem into multiple SISO problems (Eqs. (5) and (6)) in the latent space. The SISO problems are modeled using neural networks, which capture the non-linearities and dynamics of the original MIMO identification problem (Eq. (12)). Eq. (8) predicts the output Y from the outputs of the neural networks in the latent space.

3.3. Training (estimation) algorithm (Geladi & Kowalski, 1986)

The algorithm is implemented after the data have been pre-processed, i.e. scaling around zero mean and unit variance. Proper scaling prevents the latent variables from being biased towards variables with larger magnitude.

For each factor h

Step one: Initialization:
Set: $u_h = y_j$
$t_h^{\text{old}} = x_i$
where i and j are any column of the input matrix X and output matrix Y, respectively, used just for the initialization. These are often taken as the first column.
In the X block:

Step two: Define w_h as:

$$w_h^T = \frac{u_h^T X}{u_h^T u_h} \tag{15}$$

Step three: Normalize w_h to norm of one:

$$w_h = \frac{w_h}{\|w_h\|} \tag{16}$$

Step four: Calculate the X matrix scores:

$$t_h = \frac{Xw_h}{w_h^T w_h} \tag{17}$$

In the Y block:

Step five: Calculate Y matrix loadings:

$$q_h^T = \frac{t_h^T Y}{t_h^t t_h} \tag{18}$$

Step six: Normalize the loadings to norm of one:

$$q_h = \frac{q_h}{\|q_h\|} \tag{19}$$

Step seven: Calculate the Y matrix scores:

$$u_h = \frac{Yq_h}{q_h^T q_h} \tag{20}$$

Step eight: Check for convergence:

$$\frac{\text{abst}(t_h - t_h^{\text{old}})}{t_h^{\text{old}}} \leq \varepsilon \tag{21}$$

where ε is a stopping criterion.

If condition Eq. (21) is satisfied, continue to step 9 otherwise repeat from step 2 with $t_h^{\text{old}} = t_h$.

Calculate the X loadings and rescale the scores and weights accordingly:

Step nine: Calculate the X matrix loadings:

$$p_h^T = \frac{t_h^T X}{t_h^T t_h} \tag{22}$$

Step ten: Re-scale the X matrix scores accordingly:

$$t_h = t_h \cdot \|p_h^T\| \tag{23}$$

Step eleven: Normalize the loadings to norm of one:

$$p_h = \frac{p_h}{\|p_h\|} \tag{24}$$

Step twelve: Re-scale w_h accordingly:

$$w_h = w_h \cdot \|p_h\| \tag{25}$$

p_h, q_h and w_h are saved for prediction purposes.

Step thirteen: Inner model training using neural networks

t_h and u_h are taken as dynamic realizations as explained previously, so using delayed values of t_h and u_h, the present value of u_h is modeled using the neural network Eq. (12). The number of delayed values is based on cross validation Section 3.4. The training is carried out such that $\|u_h - \hat{u}_h\|$ is minimized and the weights are saved for prediction purposes.

Step fourteen: Calculation of residuals

For each factor h, Eq. (13) gives the general outer relation for the X block with $E_0 = X$ and h is the current factor. For the Y block,

$$F_h = F_{h-1} - \Phi(\cdot) q_h^T \tag{26}$$

where $F_0 = Y$ and the inner neural network model has been used in calculating the residual of the output block in Eq. (26).

Step fifteen: Final step

The procedure is repeated recursively with each factor with X and Y replaced by the corresponding residuals E_h and F_h in the algorithm. The required number of factors and hidden neurons in the network are obtained by cross validation (Section 3.4).

3.4. Cross validation (Schenker & Agarwal, 1996)

Because of fear of over-fitting and flexibility of nonlinear models, cross validation is very important for the DNNPLS. It should be used to check the number of factors needed.

The number of factors is that which gives the lowest residual on the test data. If too many factors are chosen, then the prediction property of the model is poor, so a balance has to be made on when the error reduction is minimal compared with the increase in the number of factors. Eventually, after all important factors are extracted, the residual will contain mostly random noise and the next latent factors derived from the residuals will be uncorrelated.

3.5. Prediction algorithm

This section outlines how the DNNPLS model is used for prediction of an MIMO process.

Step one: Initial condition calculations.
Set $k = 0$

1. Create matrix $Q = [q_1^T,\ q_2^T, \ldots,\ q_f^T]$
2. Calculate the inverse of Q^{-1}. If Q is not a square matrix, the generalized inverse is calculated.
3. Scale initial measured inputs and outputs to zero mean and unit variance using the scaling parameters used during the model training.
4. For each factor h extracted during training, calculate the initial latent scores using the loading weights (Step 12, Section 3.3) saved during training:

$$t_h(k) = E_{h-1}(k) w_h(k) \tag{27}$$

$$E_h(k) = E_{h-1}(k) - t_h(k) p_h^T(k) \tag{28}$$

$$U(k) = Y(k) Q^{-1} \tag{29}$$

The initial calculated scores are used by the neural network models at time step one ($k = 1$) to predict the present output scores. In the above equations, k represents the time instant, hence $Y(k)$ has dimensions of $1 \times n$ and $E_h(k)$ has dimensions of $1 \times m$, where n and m are the number of output and input variables, respectively.

Step two: For the current time step until the end of the process ($k > 0$)

1. Load the saved neural network weights for each h factor.
2. Using previous (delayed) scores and the neural network models predict present output score at time k using Eq. (12).

For control purposes, these models are used where the manipulated variables are t_h and the controlled variables are u_h. For example, consider a three-input, three-output multivariable system with two factors extracted. Since $h = 2$, two different controllers are designed in the latent space using the dynamic models for each factor. The first controller has t_1 and u_1 as its manipulated and controlled variables, respectively while the second controller has t_2 and u_2. The set point of the controllers is obtained by projecting the original set points into the latent space. An upcoming paper expands on the usage of the DNNPLS for control.

Step three: Calculate the predicted outputs from the model outputs in the latent space.

$$\hat{U}(k) = [\hat{u}_1(k),\ \hat{u}_2(k), \ldots,\ \hat{u}_f(k)] \tag{30}$$

$$\hat{Y}(k) = \hat{U}(k) Q(k) \tag{31}$$

Step four: Take process measurement
Scale present output measurements appropriately and projects the scaled measurements into the latent space to be used for next time step prediction.

The difference between the model output and the measurement at the present time step is assumed to account for plant/model mismatch when prediction is made into the future.

4. Simulation examples

4.1. Case study I: model IV fluidized catalytic cracking unit (FCCU)

A model IV fluid catalytic cracking unit (FCCU) is used here to study the effectiveness of the DNNPLS methodology. The mathematical model of the FCCU proposed by McFarlane, Reineman, Bartee and Georgakis (1993) is simulated without modification using the *s*-function of MATLAB. A FCCU receives heavier hydrocarbons from several other refinery process units and cracks these streams into lighter and more valuable components. FCCUs play a very important role because products from the unit can be blended into other products, such as gasoline. They are multi-unit processes with significant nonlinear and time-varying behavior and hence have become prime candidates for process control and optimization. Data generated by the FCCU simulation are used for implementing the DNNPLS algorithm. McFarlane's model captures the major dynamics of the reactor/regenerator section including non-linearities and interactions.

Pre-heated fresh feed, mixed with hot slurry recycle from the bottom of the main fractionator, is injected into the reactor riser, where it mixes with hot regenerated catalyst from the regenerator. This completely vaporizes the feed. Endothermic cracking reactions take place in the riser and a side reaction forms coke, which is deposited on the catalyst surface. The coke reduces the activity of the catalyst, hence continuous regeneration of the catalyst is needed. Separation of the catalyst and product gases occurs in the disengaging zone of the reactor. The product gases are sent to the main fractionator for separation into various product

streams. Air is injected into the bottom of the regenerator to assist circulation of the catalyst and to burn the coke deposited on it. Carbon and hydrogen on the catalyst react with the oxygen in the air to produce carbon monoxide, carbon dioxide and water, which are removed from the top of the regenerator.

The input variables to the model are the wet gas compressor suction valve position (v_{11}), the stack gas valve position (v_{14}), the lift air blower steam valve position (v_{lift}) and the flow of fuel to the feed preheat furnace (F_5). The output variables are the temperature of reactor riser (t_r), the reactor pressure (p_4), the differential pressure between the regenerator and the reactor (dprr) and the air flow into the regenerator (F_T). Three hundred open-loop data points are generated at a sample time of 1 min, using random amplitude signals as inputs into the unit. Another 100 data points are generated separately for cross validation to determine the optimal number of factors. The objective is to establish a non-linear dynamic model between the input and output variables. A dynamic neural network models each factor and three neurons are used in the hidden layer. The neural network model inputs consisted of two previous input scores and two previous output scores to model the present output score, hence a 4-3-1 three layer neural network. This is a one step ahead of prediction network. Four factors are extracted. Fig. 4 shows the performance of the trained neural network models in the latent space on the testing data. Only 25 data points are shown in Fig. 4. As the figure shows, the neural network models are able to predict the latent scores for each factor and the first two factors are adequate to predict the output variables. This is because two factors were sufficient for extracting the relevant information between the input and output variables. The sum of square errors between the model and the test data, given by factor in Table 1, also indicates that two factors are sufficient; hence the model for prediction purposes uses only two factors.

For comparison, a MIMO neural network was developed using six hidden layer nodes and the same data for training as was used for the DNNPLS. The MIMO network input consist of two previous inputs of the process and two previous outputs of the process to predict the present outputs, hence it was a 16-6-4, one step ahead, three layer neural network. It required 130 parameters to be identified compared to a total of 38 parameters of the DNNPLS networks. The structure of the MIMO network is chosen based on Theorem 1 (Appendix A), with the number of hidden layer nodes equal to the total of the hidden layer nodes of the SISO networks of the DNNPLS model. Hence, the MIMO neural network structure is equivalent to the DNNPLS structure. It took $\approx$75 min to train on a Pentium II, 100 MHz processor compared to 5 min for each SISO network of the DNNPLS model. An argument might

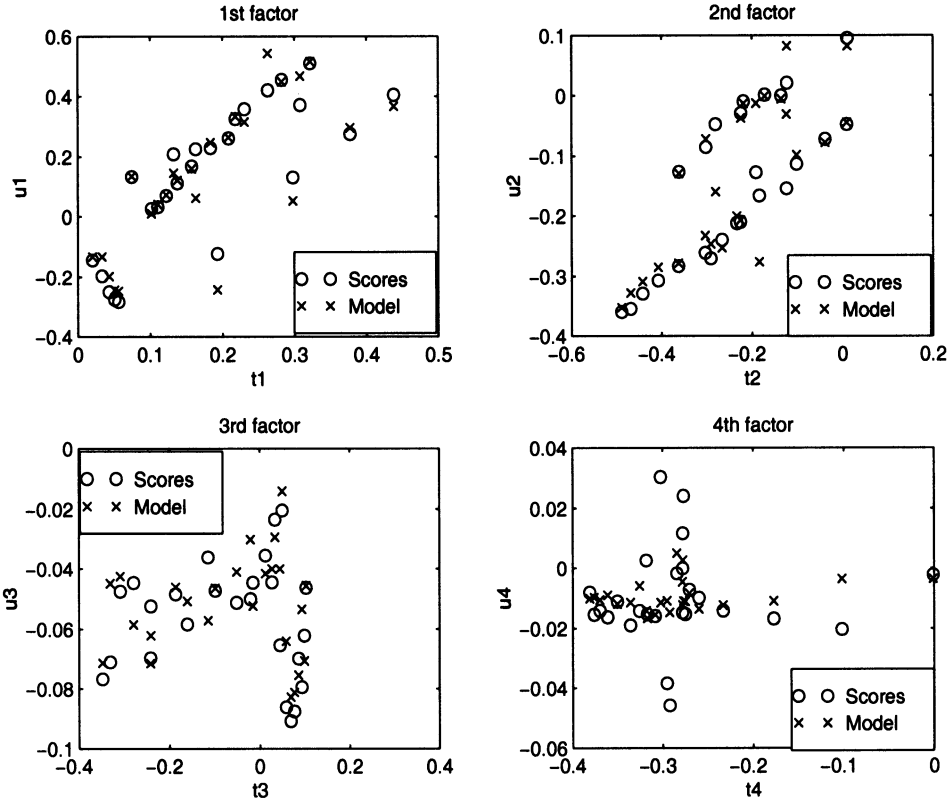

Fig. 4. Performance of the neural network models on the testing data for the four factors extracted.

Table 1
Sum of squared errors between model and test data

No of factors	Sum of squared errors
1	27.4793
2	6.5921
3	2.5648
4	2.5185

be made that the MIMO network can be broken down into SISO networks, but the MIMO model is needed to directly account for interactions in the sensitivity analysis calculations.

The model is used to predict unseen data, which was generated using different random amplitude signals as inputs. These data are different from the testing data generated for cross validation. Fig. 5 compares the results of the DNNPLS model with two factors and the MIMO neural network model in predicting the reactor temperature and Table 2 shows the sum of squared errors of the DNNPLS model and the MIMO neural network model for the reactor temperature and pressure, the differential pressure and the air flow. Both models predict adequately except for the regenerator air flow, for which the DNNPLS out-performs the MIMO neural model. Notice that only two factors are extracted by the DNNPLS model for prediction. This means the dimension of the MIMO problem in the original space has been reduced from a four-input by four-output to two SISO problems in the latent space.

By using the DNNPLS model, the training time and the number of parameters needed are reduced and it performs as good as a MIMO neural network model. It is also more suitable for control purposes, as decoupling takes place in the latent space while indirectly accounting for interactions. Though the dynamics remain decoupled, the coupling of the input constraints in the latent space can lead to problems. Hence, not only is the DNNPLS able to model the complex dynamics, the properties of the PLS can still be used to reduce the dimensionality of the problem.

Table 2
Comparison of sum of squared errors between the DNNPLS model and the MIMO neural network model

Variable	DNNPLS model	MIMO neural network model
Reactor temperature	6.821×10^{-3}	6.688×10^{-3}
Reactor pressure	0.9748	0.29845
Differential pressure	0.9598	0.9838
Regenerator air flow	3.8332	12.5805

4.2. Case study II: isothermal reactor

The reactor of Li and Biegler (1988) is used for comparison between the DNNPLS and PLS-ARMA approach to modeling dynamics. Fig. 6 shows a schematic of the reactor. The following reaction occurs in an ideal stirred tank reactor:

$$A + B \rightarrow P \tag{32}$$

where A is in excess. The reaction rate equation is given by:

$$r_B = \frac{K_1 C_B}{(1.0 + K_2 C_B)^2} \tag{33}$$

where C_B is the concentration of component B. The concentration of B in the two inlet streams are assumed constant at $C_{B1} = 24.9$ mol/l and $C_{B2} = 0.1$ mol/l and contain an excess amount of A. The flow of liquid from the tank is determined by

$$F(h) = 0.2 \cdot h^{0.5}. \tag{34}$$

The cross-sectional area of the tank is 1.0 m^2 and the sampling time is 1.0 min. The values of parameters used are listed in the Table 3.

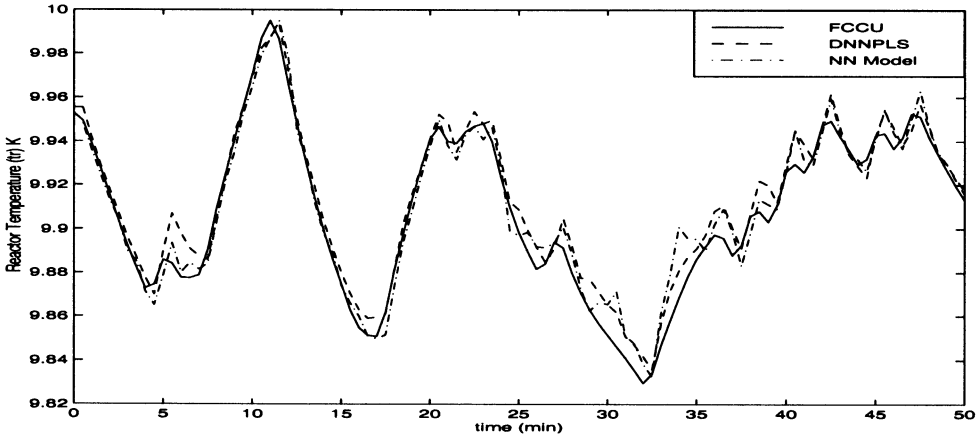

Fig. 5. Prediction of the reactor temperature.

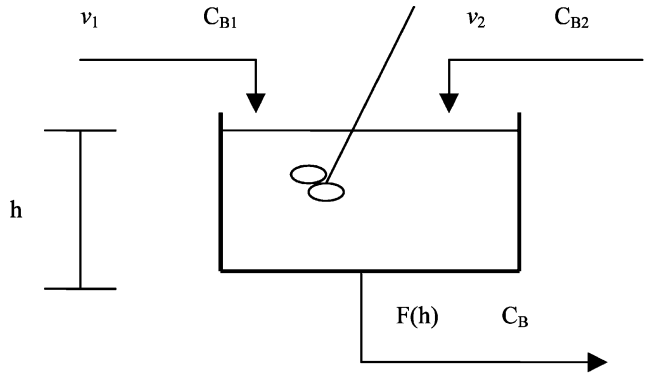

Table 3
Parameters for the isothermal reactor

Kinetic parameters	Concentrations of B	Initial conditions
$K_1 = 1.0\ s^{-1}$	$C_{B1} = 24.9$ mol/l	$x_{10} = 40$ m
$K_2 = 1.0$ l/mol	$C_{B2} = 0.1$ mol/l	$x_{20} = 0.1$ mol/l

After simplification the model becomes:

$$dt = \frac{dc_1}{dt} = v_1 + v_2 - 0.2c_1^{0.5} \tag{35}$$

$$\frac{dc_2}{dt} = (24.9 - c_2)v_1c_1^{-1} + (0.1 - c_2)v_2c_1^{-1} - \frac{c_2}{(1 + c_2)^2} \tag{36}$$

$$\begin{cases} y_1 = c_1 \\ y_2 = c_2 \end{cases} \tag{37}$$

where c_1 is the liquid height in tank in meters, c_2, the concentration of B in the reactor in mol/l, v_1, the inlet flow rate with condensed B in m^3/s and v_2, the inlet flow rate with dilute B in m^3/s. A factor of 1000 l/m^3 cancels out in Eq. (36).

Open loop data (input variables v_1 and v_2, and output variables y_1 and y_2) used for training the inner neural networks models of the DNNPLS and PLS-ARMA structure, are obtained using random amplitude signals as inputs into the simulation of the process. Two hundred data points are generated separately for training and 100 for testing.

For the DNNPLS the input matrix $X = [v_1(k)\ v_2(k)]$ and $Y = [y_1(k)\ y_2(k)]$ while for the PLS-ARMA, the input matrix $X = [v_1(k-1)\ v_2(k-1)\ v_1(k-2)\ v_2(k-2)]$. The input matrix X for the PLS-ARMA approach consist of two delayed input and output variables because the same structure is used for the neural network models of the DNNPLS. Hence, for training the dimensions of X are 200×2 and 200×4, respectively and the dimension of Y is 200×2 for both methods. The dimension of matrix X is twice for the PLS-ARMA approach compared to the DNNPLS. In the PLS-ARMA models, neural networks have been used to model the inner relationship between the latent scores for fair comparison. An argument might be made that the DNNPLS methodology equally increases the number of parameters to be estimated but this is much easier to handle than the complexities of non-square loading matrix inversions, that may result from introducing lagged values in the input matrix X.

Fig. 7 shows the response of the reactor concentration to a different random amplitude signal for both approaches using the same number of factors. The two factors are used by both approaches for predicting the reactor liquid level and concentration. The DNNPLS outperforms the PLS-ARMA approach for both the level and concentration. While the PLS-ARMA is able to model the trend using two factors, the DNNPLS gives more accurate predictions. The performance of the PLS-ARMA approach can be improved by increasing the number of factors.

Table 4 shows the sum of squared errors for the DNNPLS model and PLS-ARMA model.

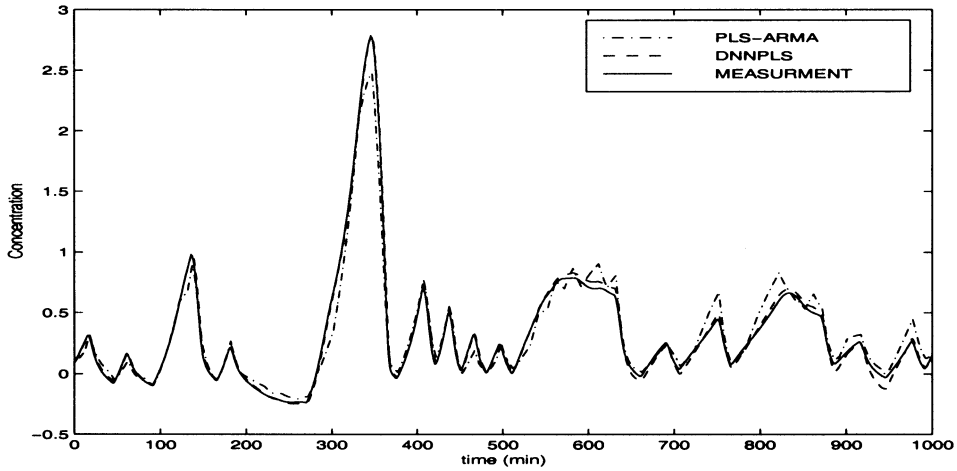

Fig. 7. Prediction using DNNPLS and PLS-ARMA of concentration.

Table 4
Comparison of the sum of squared errors between the DNNPLS model and PLS-ARMA model

Variable	DNNPLS model	PLS-ARMA model
Level	9.425×10^{-3}	2.3670
Concentration	7.929×10^{-2}	0.6723

Figs. 8 and 9 show the prediction of the first 25 factor scores on the test data for each factor in the latent space using both approaches. u_h and t_h are the output and input scores for the hth factor.

There are probably two reasons why the DNNPLS outperforms the PLS-approach:

1) Because the data are dynamic, the scores become inherently dynamic by nature and hence modeling the scores by a static model does not perform as well as a dynamic model. This can be seen by the prediction accuracy of the inner models when using a dynamic neural network and a static neural network during training (see Figs. 8 and 9 for comparison).
2) Increasing the dimensions of the X matrix means increasing the number of factors extracted to capture the underlying information. Though in the example two factors for the PLS-ARMA gave less accurate performance to the DNNPLS, the number of factors for the PLS-ARMA needed for comparable performance may have to be increased. Fig. 10

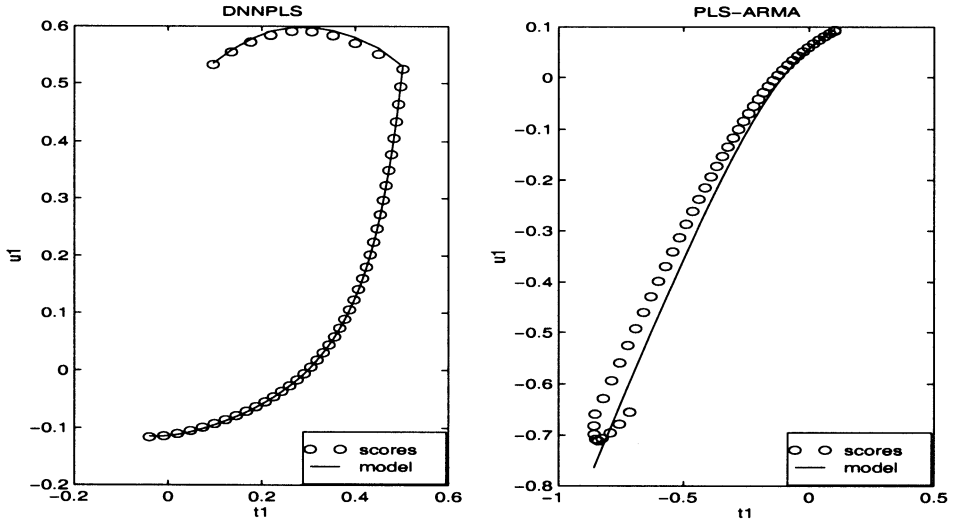

Fig. 8. First factor predictions of the inner models.

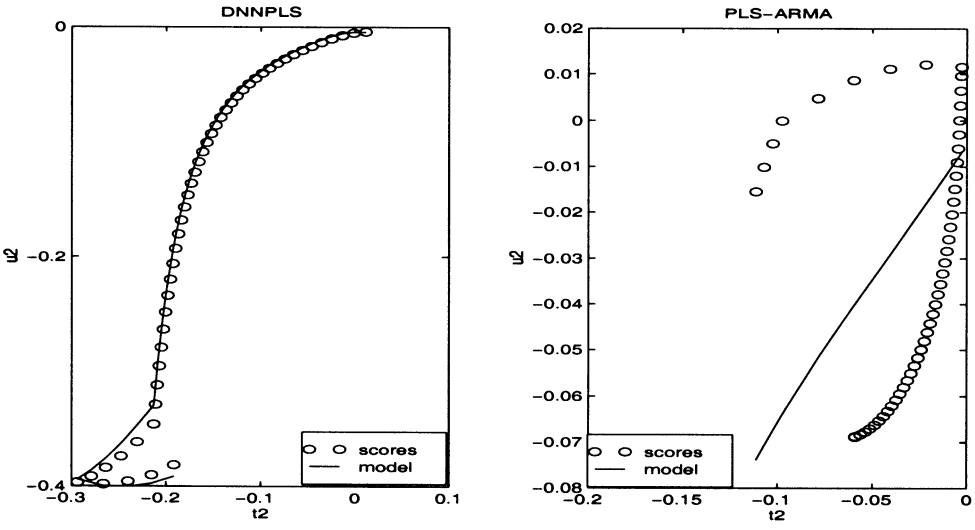

Fig. 9. Second factor predictions of the inner models.

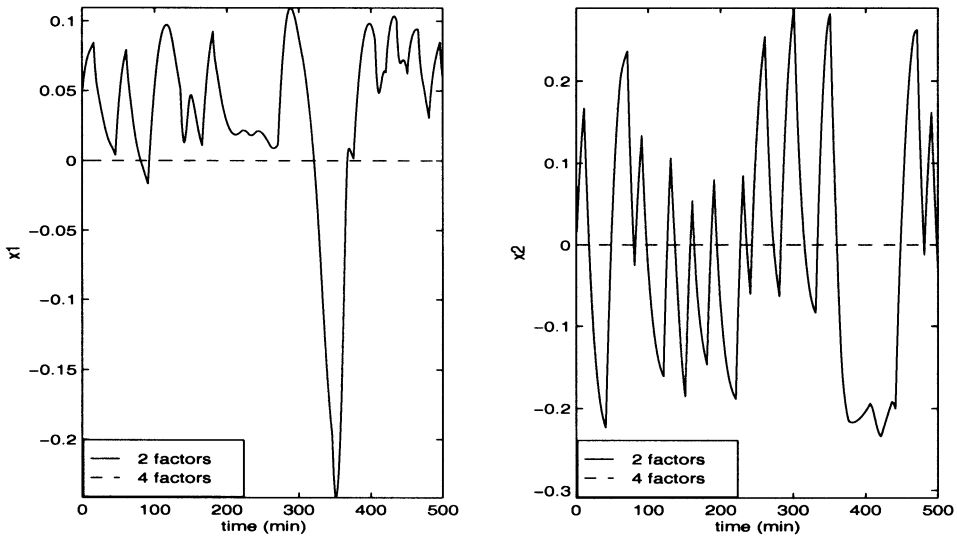

Fig. 10. Residuals after using PLS-ARMA.

shows the residual information after factors have been extracted during training using the PLS-ARMA. $x1$ is the residual of the scaled inlet flow rate of condensed B and $x2$ is the residual of the scaled inlet flow rate of dilute B. It is obvious from Fig. 10 that not all the relevant information is extracted with two factors and better results would be obtained if four factors were extracted. When the number of factors extracted is increased to four, most of the information needed is extracted. Notice in Fig. 10 that the residuals are very small after four factors are extracted.

5. Conclusions

A dynamic NNPLS (DNNPLS) modeling approach for multivariable systems is introduced. This is a direct extension of the NNPLS, which is static. The dynamic relation has been moved from the outer model into the inner model of the PLS. The advantages of moving the lagged details into the inner model include reduction in the size of the input X matrix and a simpler model structure suitable for process control.

The performance of the approach is shown on the highly nonlinear fluidized catalytic cracking unit to model the dynamics. It was shown that with less parameters and difficulties, the process predicts as good as a MIMO neural network. Though this approach has been dismissed as sub-optimal in the past, no reason has been given why it should not be used. The superior performance over an approach of modeling the dynamics using the outer model (PLS-ARMA) is also shown. Some reasons for this superiority are also given. Future research would address using the developed model for process control.

Appendix A

A.1. Theorem 1

The DNNPLS model is equivalent to an m-input p-output dynamic network with one hidden layer given by:

$$Y(k) = \Psi_2\sigma(\Psi_1\zeta(k) + \kappa_1) + \kappa_2 + F_f \tag{A1}$$

where

$$\zeta(k) = [X(k-1), \ldots, X(k-xt), Y(k-1), \ldots, Y(k-yt)]^T \tag{A2}$$

$$\Psi_2 = [q_1 W_{21}, \ldots, q_f W_{2f}] \tag{A3}$$

$$\Psi_1 = [\tilde{W}_{11}, \ldots, \tilde{W}_{2f}] \tag{A4}$$

$$\kappa_1 = [B_{11}, \ldots, B_{2f}]^T \tag{A5}$$

$$\kappa_2 = [B_{21}q_1, \ldots, B_{2f}q_1] \tag{A6}$$

and σ is the sigmoid function.

A.1.1. Proof

Eq. (8) can be written as:

$$Y(k) = \sum_{h=1}^{f} \hat{u}_h(k) q_h^T + F_h \tag{A7}$$

where

$$\hat{u}_h(k) W_{2h}\sigma(W_{1h} IN_h(k) + B_{1h}) + B_{2h} \tag{A8}$$

$$IN_h(k) = [t_h(k-1), \ldots, t_h(k-xt), u_h(k-1), \ldots, u_h(k-yt)]^T \tag{A9}$$

Hence

$$IN_h(k) = [X(k-1)\tilde{w}_h, \ldots, X(k-xt)\tilde{w}_h, Y(k-1)\tilde{c}_h, \ldots, Y(k-yt)\tilde{c}_h]^T \tag{A10}$$

Define

$$\gamma_{Ah} = \text{diagonal}\,(\tilde{w}_h) \in \Re^{xtx(m-xt)} \quad \text{(A11)}$$

$$\gamma_{Bh} = \text{diagonal}\,(\tilde{c}_h) \in \Re^{ytx(n-yt)} \quad \text{(A12)}$$

$$\gamma_h = \begin{bmatrix} \gamma_{Ah} & 0 \\ 0 & \gamma_{Bh} \end{bmatrix} \in \Re^{(xt+yt)x(m-xt+n-yt)} \quad \text{(A13)}$$

$$\zeta(k) = [X(k-1), \ldots, X(k-xt), Y(k-1), \ldots, Y(k-yt)]^T \quad \text{(A14)}$$

Then

$$IN_h(k) = \gamma_h \zeta(k) \quad \text{(A15)}$$

Hence:

$$Y(k) = \sum_{h=1}^{f} (W_{2h}\sigma(W_{1h}\gamma_h\zeta(k) + B_{1h}) + B_{2h})q_h^T + F_f \quad \text{(A16)}$$

$$Y(k) = \Psi_2\sigma(\Psi_1\zeta(k) + \kappa_1) + \kappa_2 + F_f \quad \text{(A17)}$$

Q.E.D.

where

$$\tilde{W}_{1h} = W_{1h}\gamma_h \quad \text{(A18)}$$

Hence the weight of a MIMO neural network model can be obtained from a DNNPLS model.

References

Dunn, W. J., III, Wold, S., Edlund, U., Hellberg, S., & Gasteiger, J. (1984). Multivariate structure–activity relationships between data from a battery of biological test and an ensemble of structure descriptors: the PLS method. *Quant.-Struc.-Act. Relat. 3*, 131–137.

Geladi, P., & Kowalski, B. R. (1986). Partial least squares regression: a tutorial. *Anal. Chim. Acta 185*, 1–17.

Höskuldsson, A. (1998). PLS regression methods. *J. Chemomet. 2*, 211–228.

Kresta, J. Applications of partial least squares regression. PhD Thesis, McMaster University, Hamilton, Ont., Canada, 1992.

Lakshminarayanan, S., Shah, S. L., & Nandakumar, K. (1997). Modeling and control of multivariable processes: dynamic PLS approach. *Am. Inst. Chem. Eng. J. 43*, 2307–2322.

Lakshminarayanan, S., Patwardhan, R. S., & Shah, S. L. (1998). Constrained nonlinear MPC using Hammerstein and Wiener models: PLS framework. *Am. Inst. Chem. Eng. J. 44* (7), 1611–1622.

Li, W. C., & Biegler, L. T. (1988). Process control strategies for constrained nonlinear systems. *Ind. Eng. Chem. Res. 27*, 1421–1433.

Lippman, R.P. An introduction to computing with neural nets. IEEE ASSP Magazine, April 1987, p. 4.

Lorber, A., & Kowalski, B. R. (1998). The effects of interferences and calibration design on accuracy: implications for sensor and sample selection. *J. Chemomet. 2*, 67–79.

MacGregor, J.F., Marlin, T.E., Kresta, J., Skagerberg, B. Multivariate statistical methods in process analysis and control. Proceedings of the CPC-IV Conference, South Padre Island, TX, 1991, pp. 18–22.

Martens, H., & Naes, T. (1989). *Multivariate Calibration*. New York: Wiley.

McFarlane, R. C., Reineman, R. C., Bartee, J. F., & Georgakis, C. (1993). Dynamic simulator for a model IV fluid catalytic cracking unit. *Comp. Chem. Eng. 17* (3), 275–300.

Qin, S. J., & McAvoy, T. J. (1992). Nonlinear PLS modeling using neural networks. *Comp. Chem. Eng. 16* (4), 379–391.

Qin, S. J., & McAvoy, T. J. (1996). Nonlinear FIR modeling via a neural net PLS approach. *Comp. Chem. Eng. 20* (2), 147–159.

Ray, W. H., & Kaspar, M. H. (1992). Chemometric methods for process monitoring and high-performance controller design. *Am. Inst. Chem. Eng. J. 38*, 1593.

Ray, W. H., & Kaspar, M. H. (1993). Dynamic PLS modeling for process control. *Chem. Eng. Sci. 48*, 3447–4346.

Ricker, N. L. (1988). The use of biased least squares estimators for parameters in discrete-time pulse response models. *Ind. Eng. Chem. Res. 27*, 343–350.

Schenker, B., & Agarwal, M. (1996). Cross-validated structure selection for neural networks. *Comp. Chem. Eng. 20*, 175–186.

Wise, B.M. Adapting multivariate analysis for monitoring and modeling dynamic systems. PhD Thesis, University of Washington, Seattle, 1991.

Wold, H. (1984). Partial least squares. In *Encyclopedia of Statistical Sciences*, vol. 6 (pp. 581–591). New York: Wiley.

Wold, S., Kettaneh-wold, N., & Skagerberg, B. (1989). Nonlinear PLS modeling. *Chemometr. Intell. Lab. Sys. 7*, 53–65.