Adaptive Control For Nonlinear Multivariable Systems Using a Single-Layer Neural Networks And Conventional Linear Controller

A. M. Wahdan
Faculty of Engineering
Ain Shams University, Cairo, Egypt,
A.M_Wahdan@Hotmail.com

T. A. Al-Zohary
Faculty of Science
Al_Azhar University, Cairo, Egypt,
tt_tamer@Yahoo.com

Abstract – In this paper a real time control technique for a nonlinear multivariable system is presented. The proposed technique is a hybrid approach, which combines the ability of a single-layer feedforward neural network for modeling purposes and a linear control law to design the controller, providing a bridge between the field of neural networks and the well known linear adaptive control methods. To solve the control problem in this paper, we consider that the state of the system is accessible. Simulation results are presented toward the end of the paper to show the effectiveness of the proposed methodology.

I. INTRODUCTION

There has been considerable interest in the past few years in exploring the applications of artificial neural networks for identification and adaptive control of dynamical systems [1]-[7].

It has been realized by systems theorists that most real dynamical systems are nonlinear [6]. However, linearizations of such systems around the equilibrium states yield linear models which are mathematically tractable. In particular, based on the superposition principle, the output of the system can be computed for any arbitrary input, and alternately, in control problems, the input which optimizes the output in some sense can also be determined with relative ease. In adaptive control problems, where the plant parameters are assumed to be unknown, the fact that the latter occur linearly makes the estimation procedure straightforward. While the above facts are responsible for the popularity of linear models in theoretical studies, the fact that most nonlinear systems thus far could be approximated satisfactory by such models in their nominal ranges of operation has made them attractive in practical contexts as well. It is this combined effect of ease of analysis and practical applicability that accounts for the greet success of linear models and has then the subject of intensive study for over four decades.

Most of he results reported in the above literatures of the adaptive control of nonlinear dynamical systems are related to a single-input single-output systems. But most practical systems have multiple inputs and multiple outputs (MIMO) then our interest in this paper is to study the problem of controlling nonlinear multivariable systems when the state variables are accessible.

The problem of controlling a plant can be conveniently divided into the regulation and tracking problems. In the former, the main objective is to stabilize the plant around a fixed operating point. In the later, the aim is to make the output of the plant follow a specified signal asymptotically. While our ultimate goal is to determine the control input, u, based only on output measurement for both regulation and tracking. We will confine our attention in this paper to the problem of tracking when the state variables of the multivariable system are accessible.

In this paper, a strategy that combines a single-layer feedforward network model with self tuning indirect adaptive control, is proposed. The proposed control structure is based on the single-layer feedforward network model linearization at every operating point. A standard linear state space model of the form $x(k+1) = Hx(k) + Ku(k)$ is derived and a state space feedback decoupling controller $u(k) = Fr(k) - Gx(k)$ is applied.

With simultaneous online training of the single layer feedforward neural network and control synthesis the resulting algorithm is an indirect adaptive control law.

The proposed method is applied to the nonlinear multivariable system given in [8]. The simulation results given at the end of the paper show the effectiveness of the proposed hybrid approach with respect to set-point tracking.

The paper is organized as follows: In section II the mathematical preliminaries related to control of linear decoupled multivariable systems is given. Section III introduces the problem to solve in detail. In section IV the network architecture used for the identification process is given. Section V study the linearization of the nonlinear process obtained from the neural network. In section VI the algorithm that used to solve the control problem is stated in details. Section VII gives the simulation study on a real example. A comparison with other works is given in section VIII to show the effectiveness of the proposed method.

II. MATHEMATICAL PRELIMINARIES

A discrete-time linear multivariable system with *m* inputs and *m* outputs can be described by the state equations

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{1}$$

where $x(k) \in \Re^n$ is the state vector at instant *k*, $u(k) \in \Re^m$ is the input vector, and $y(k) \in \Re^m$ is the output vector. *A*, *B*, and *C* are constant matrices and are assumed to be known. Our objective is to determine the control input $u(k)$ such that the output $y(k)$ of the plant tracks a desired output vector $y_d(k)$ asymptotically, that is, $lim_{k\to\infty}\|y(k) - y_d(k)\| = 0$. A closely related problem is one of decoupling. In this case, a reference input *m*-vector *r(k)* is defined and a control input has to be generated so that $r_i(k)$ affects only one component $y_i(k)$ of the output for $i = 1,2,\cdots,m$. As seen from the following, weather or not a multivariable system can be decoupled depends upon the relative degrees of the outputs

with respect to the inputs, or alternatively, upon the delay between each input-output pair.

By the successive application of (1) we obtain

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ x(k+2) &= A^2x(k) + ABu(k) + Bu(k+1) \\ \cdots \quad & \cdots \quad \cdots \\ x(k+n) &= A^n x(k) + \sum_{i=0}^{n-1} A^{n-i-1}Bu(k+i). \end{aligned} \tag{2}$$

If the system is controllable (i.e., the $n \times nm$ matrix $\left[A^{n-1}B, A^{n-2}B, \cdots, B\right]$ is of rank *n*), any initial state can be transferred to any final state in at most *n* steps. Using the relation $y(k) = Cx(k)$, the following equation for the value of the output at time $k+d$ in terms of the states and inputs can be obtained:

$$\begin{aligned} y(k+d) = CA^d x(k) + CA^{d-1}Bu(k) + \\ CA^{d-2}Bu(k+1) + \cdots + CBu(k+d-1). \end{aligned} \tag{3}$$

For the case of an SISO system (when *B* is a vector), it readily follows that if $CB, CAB, \cdots, CA^{d-2}B$ are zero but $CA^{d-1}B$ is not zero, (3) reduces to

$$y(k+d) = CA^d x(k) + CA^{d-1}Bu(k).$$

In this case, the input-output pair is said to have a relative degree *d*. In the multivariable case, we can define, in a similar fashion, the relative degree d_{ij} for each input-output pair (u_j, y_i). If $\min_j d_{ij} = d_i$, d_i is called the relative degree of the *ith* output y_i. It corresponds to the minimum time delay between any one of the inputs and the *ith* output.

Let C_i be the *ith* row of C. Considering the row vector $C_i A^{d_i - 1}B$, only those elements that are nonzero correspond to inputs from which the delay is d_i to the output y_i. Let the matrix E be defined such that its ith row $E_i = C_i A^{d_i - 1}B$. It is well known [9] that the system (1) can be decoupled iff E is nonsingular. Qualitatively, this implies that the delays through the system are such that there is adequate freedom to control one output by one input.

The following representation of linear multivariable systems, which can be derived from the above procedure, is important for the developments given in sections IV and V.

$$\begin{aligned} y_1(k+d_1) &= C_1 A^{d_1} x(k) + E_1 u(k) \\ y_2(k+d_2) &= C_2 A^{d_2} x(k) + E_2 u(k) \\ \cdots \quad & \cdots \quad \cdots \\ y_m(k+d_m) &= C_m A^{d_m} x(k) + E_m u(k) \end{aligned} \tag{4}$$

if the desired output vector — the desired value of the vector on the left-hand side of (4) — is denoted by $r(k)$, it follows that the desired $u(k)$ can be expressed as a linear combination of $r(k)$ and $x(k)$ so that

$$u(k) = E^{-1}\begin{bmatrix} C_1 A^{d_1} \\ C_2 A^{d_2} \\ \cdots \\ C_m A^{d_m} \end{bmatrix} x(k) + E^{-1} r(k) = Gx(k) + Fr(k). \tag{5}$$

hence, using state feedback, the output of the plant can be made to the desired output $r(k)$.

In the following we will show how (5) can be used as an adaptive linear controller for a nonlinear multivariable system.

III. STATEMENT OF THE PROBLEM

Consider a discrete-time nonlinear multivariable system S be described by the state equations

$$\text{S:} \quad \begin{aligned} x(k+1) &= f[x(k), u(k)] \\ y(k) &= C\,x(k) \end{aligned} \tag{6}$$

where $u(k), y(k) \in \Re^m$, and $x(k) \in \Re^n$ are the input, output, and state, respectively, at time t and $f \in C^\infty$.

Consider the reference input $r(k)$ is defined as $r(k) = [r_1(k), r_2(k), \cdots, r_m(k)]^T \in \Re^m$ is specified and the state variables are accessible. The control problem is to determine the input $u(k)$ so that $y_i(k)$ follows the reference input $r_i(k)$. Note that we consider that f is unknown and only input, output state variables are known.

Our primary objective in this paper is to suggest a hybrid method using a single-layer feedforward neural network and a state space feedback controller for dealing with control problem mentioned above.

Some prior information concerning the system is required to determine the controller which are the order of the system and relative degree d_i of each output y_i. For known system some additional conditions are required to determine the controller as given in [8] which are:

1) the linearized system L_S is controllable observable and can be decoubled using state feedback.
2) the plant satisfies the minimum phase condition.

IV. NETWORK ARCHITECTURE

The network architecture used in this paper depends on the concept of relative degree of a nonlinear multivariable system S given below.

The vector relative degree of a multivariable system S can be defined as the vector $d = [d_1, d_2, \cdots, d_m]^T$ where d_i denotes the relative degree of the *ith* output. The latter, in turn, is the minimum of the relative degrees of the *ith* output with respect to all the inputs. It can be shown that if the vector relative degree exists for the multivariable system S, in some neighborhood Ω of the equilibrium point $(x=0, u=0)$, the nonlinear system S can be expressed as

$$\begin{aligned} y_1(k+d_1) &= \Psi_1[x(k), u(k)] \\ y_2(k+d_2) &= \Psi_2[x(k), u(k)] \\ \cdots \quad & \cdots \quad \cdots \\ y_m(k+d_m) &= \Psi_m[x(k), u(k)] \end{aligned} \tag{7}$$

Assume that the unknown nonlinear multivariable system to be considered is expressed by (7) where $u(k), y(k+d) \in \Re^m$, and $x(k) \in \Re^n$. The single-layer feedforward network architecture used in this case is shown in Fig. 1, in which the input layer is composed of $(n+m+1)$ nodes for *n* states, *m* inputs all at time *k* and an additional node for input bias its value is always 1 (note that: we consider that this additional input is add to the set

inputs of the system at time *k* and then the network has *m+1* inputs) and the output layer of *m* nodes for the *m* outputs of the system at times $k+d_1, k+d_2, \cdots, k+d_m$. The interconnection matrices are $W^{yx} \in \Re^{m,n}$ and $W^{yu} \in \Re^{m,m+1}$, respectively among the output-state nodes and output-input nodes.

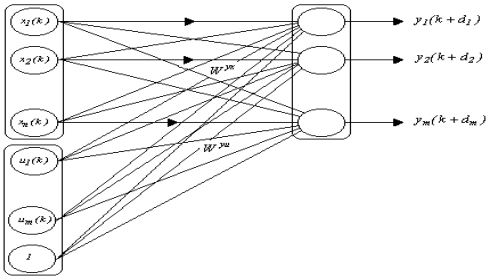

Fig. 1. The neural network structure.

The dynamics of the network is described by the following equations:

$$\hat{y}_j(k+d_j) = \hat{\Psi}[S_j] \tag{8}$$

$$S_j = \sum_{i=1}^{n} w_{ji}^{yx} x_i(k) + \sum_{\ell=1}^{m+1} w_{j\ell}^{yu} u_\ell(k) \tag{9}$$

The activation functions $\hat{\Psi}$ are hyperbolic tangent given by (10)

$$\hat{\Psi}(s) = b\frac{e^{as} - e^{-as}}{e^{as} + e^{-as}} \qquad a,b \in \Re^{+} \tag{10}$$

Then the neuro model for the system (7) can be expressed as

$$\hat{y}_j(k+d_j) = \hat{\Psi}[x(k), u(k)] \tag{11}$$

In section VI we will show the training process used to serve our purpose of controlling the unknown nonlinear multivariable decoupling system (6).

V. LINEARIZATION OF THE NEURAL MODEL

The neural model $\hat{y}_j(k+d_j) = \hat{\Psi}[x(k), u(k)]$ is a nonlinear function. However it is possible to derive a linear model by computing the derivatives from the outputs with respect to the inputs of the network (x(k),u(k)). The linearized system is given by

$$\hat{\hat{y}}_j(k+d_j) = H\, x(k) + K\, u(k) \tag{12}$$

where the two matrices *H* and *K* are of orders $(m \times n)$ and $(m \times m+1)$ respectively which are defined by

$$H = \frac{\partial \hat{\Psi}\{\}}{\partial x(k)} = W^{yx}\hat{\Psi}'\{\} \tag{13}$$

$$K = \frac{\partial \hat{\Psi}\{\}}{\partial u(k)} = W^{yu}\hat{\Psi}'\{\} \tag{14}$$

Experimentally, the linearized system (12) is similar to the linear system given in (4). Then a linear controller similar to (5) can be derived so that outputs of the system at times $k+d_1, k+d_2, \cdots, k+d_m$ become close as possible to a specified corresponding setpoints $r_1(k), r_2(k), \cdots, r_m(k)$.

In the following section, the algorithm that describe simultaneous online training of the single-layer feedforward neural network and control synthesis using the linearization process mentioned above is given in detail. Also we will see that the resulting algorithm is an indirect adaptive control law.

VI. ALGORITHM DESCRIPTION

Assume that the unknown nonlinear system to be considered is expressed by (7). The purpose of our control algorithm is to select a control signals $u_1(k), u_2(k), \cdots, u_m(k)$, such that the the outputs of the system at times $k+d_1, k+d_2, \cdots, k+d_m$, are made as close as possible to a prespecified setpoints $r_1(k), r_2(k), \cdots, r_m(k)$.

Fig. 2 shows the adaptive control structure using a single-layer feedforward neural network which consists of

1) the system (7).
2) a single-layer feedforward network which estimates $\Psi = [\Psi_1, \Psi_2, \cdots, \Psi_m]$.
3) a controller realized by the linearization of the neural model.

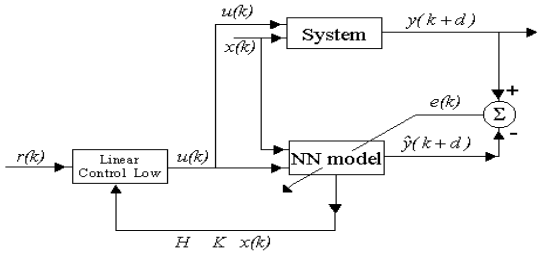

Fig. 2. Adaptive control structure using single layer network.

The neuro model for the unknown system (7) can be expressed as (11)

$$\hat{y}_j(k+d_j) = \hat{\Psi}[x(k), u(k)]$$

where $\hat{y}_j(k+d_j)$ is the *jth* output of the neural network and $\hat{\Psi}$ is the estimate of the functons $\Psi_1, \Psi_2, \cdots, \Psi_m$. The training algorithm used guarantees that

$$\sum_{j=1}^{m} [y_j(k+d_j) - \hat{y}_j(k+d_j)]^2 = min. \tag{15}$$

$\hat{y}_j(k+d_j)$ is also referred to as a predicted *jth* output of the system (7) at time $k+d_j$. The objective function *J* is defined as follows:

$$J = \frac{1}{2}\sum_{j=1}^{m} e_j^2(k) \tag{16}$$

where

$$e_j(k) = y_j(k+d_j) - \hat{y}_j(k+d_j). \tag{17}$$

The elements of the gradient of *J* with respect to both W^{yx} and W^{yu} is given by the following two equations

$$\frac{\partial J}{\partial W_{ji}^{yx}} = -e_j.\Psi'[S_j].x_i \tag{18}$$

$$\frac{\partial J}{\partial W_{ji}^{yu}} = -e_j.\Psi'[S_j].u_i \tag{19}$$

The weight changes may be made along the negative gradient of of *J* by means of the equations

$$\Delta W_{ji}^{yx} = -\eta\frac{\partial J}{\partial W_{ji}^{yx}} \tag{20}$$

$$\Delta W_{ji}^{yu} = -\eta \frac{\partial J}{\partial W_{ji}^{yu}} \quad (21)$$

Once change of the weights occur, the linearization process start using the process stated in section V. Then the linearized system of the form (12)

$$\hat{\hat{y}}_j(k+d_j) = H\,x(k) + K\,u(k)$$

is obtained .

Having a linear model describe the process the state feedback control low can be defined by

$$u(k) = Fr(k) - Gx(k) \quad (22)$$

more details of the process can be find in the simulation results.

Then equation (22) can now be used in computer program for real time control.

A pesudocode outline of the algorithm is presented below:

Select initial weights
Select the state
Select the values of the desired set points r(k)
u(k) = any random value (very small) or equal to zero.
Repeat up to approximately no change occur in the control signal

{
produce $\hat{y}_j(k+d_j)$ using (11);
find J using (16);
update the weights using (18), (19), (20) and (21);
find the linearized model (12);
compute new control signals u(k+1) using (22);
$u(k) = u(k+1)$;
}

VII. SIMULATION RESULTS

In this section, we describe the application of the proposed model on a real problem. Note that the learning used for the network used in this example takes the value 0.001. The values of both *a* and *b* in the activation function takes the values 1 and 8 respectively and random initial weights in the range $(-0.1, 0.1)$.

Example:

The plant considered here is a third-order system given in [8] with two inputs and two outputs and is described by the state equations

$$x_1(k+1) = 0.9x_1(k)\sin[x_2(k)] + [2 + 1.5\frac{x_1(k)u_1(k)}{1+x_1^2(k)}]u_1(k) + x_1(k) + \frac{2x_1(k)}{1+x_1^2(k)}]u_2(k)$$

$$x_2(k+1) = x_3(k)\{1 + \sin[4x_3(k)]\} + \frac{x_3(k)}{1+x_3^2(k)}$$

$$x_3(k+1) = \{3 + \sin[2x_1(k)]\}u_2(k)$$

$$y_1(k) = x_1(k); \qquad y_2(k) = x_2(k)$$

where $x(k) = [x_1(k), x_2(k), x_3(k)]^T$ represents the state, $u(k) = [u_1(k), u_2(k)]^T$ the input, and $y(k) = [y_1(k), y_2(k)]^T$ the output, at instant *k*. The delay from either of the inputs to y_1 is unity, and the delay to y_2 is three from input u_1 and two from input u_2. Hence, $d_1 = 1, d_2 = 2$. The linearized system at $x = 0$, $u = 0$ is described by

$$\delta x_1(k+1) = 2\delta u_1(k)$$
$$\delta x_2(k+1) = 2\delta x_3(k)$$
$$\delta x_3(k+1) = 3\delta u_2(k)$$
$$\delta y_1(k) = \delta x_1(k); \qquad \delta y_2(k) = \delta x_2(k)$$

which can also be described by the input-output equations

$$\delta y_1(k+1) = 2\delta u_1(k)$$
$$\delta y_2(k+2) = 6\delta u_2(k).$$

The linearized system is controllable, observable, and is of minimum phase. Hence, a nonlinear decoupling controller exists for the nonlinear plant in a neighborhood of the origin. Our objective is to realize this nonlinear controller using neural networks to make the two outputs $y_1(k)$ and $y_2(k)$ follow two independent reference signals $y_1^*(k)$ and $y_2^*(k)$, respectively. We assume that $y_1^*(k+d_1) = r_1(k)$ and $y_2^*(k+d_2) = r_2(k)$ are specified at instant *k*. Hence, the control problem is to determine a control input *u(k)* so that $lim_{k\to\infty}|y_i(k+d_i) - r_i(k)| = 0 \ (i = 1,2)$.

Using the network architecture and Algorithm given in sections IV and VI it was found that:

The state input output neural model obtained in this case is given by:

$$y_1(k+1) = \hat{\Psi}[w_{00}^{yx}x_0 + w_{01}^{yx}x_1 + w_{02}^{yx}x_2 + w_{00}^{yu}u_0 + w_{01}^{yu}u_1 + w_{02}^{yu}u_2].$$

$$y_2(k+2) = \hat{\Psi}[w_{10}^{yx}x_0 + w_{11}^{yx}x_1 + w_{12}^{yx}x_2 + w_{10}^{yu}u_0 + w_{11}^{yu}u_1 + w_{12}^{yu}u_2].$$

where $u_2 = 1$ is the additional input used for bias.

After a weight change obtained using (20) and (21) the linearization process started using the method given in section IV which in our example is given by

$$\begin{bmatrix}\hat{\hat{y}}_1(k+1)\\ \hat{\hat{y}}_2(k+2)\end{bmatrix} = \begin{bmatrix}\frac{\partial\hat{\Psi}_0}{\partial x_0} & \frac{\partial\hat{\Psi}_0}{\partial x_1} & \frac{\partial\hat{\Psi}_0}{\partial x_2}\\ \frac{\partial\hat{\Psi}_1}{\partial x_0} & \frac{\partial\hat{\Psi}_1}{\partial x_1} & \frac{\partial\hat{\Psi}_1}{\partial x_2}\end{bmatrix}\begin{bmatrix}x_0\\x_1\\x_2\end{bmatrix} + \begin{bmatrix}\frac{\partial\hat{\Psi}_0}{\partial u_0} & \frac{\partial\hat{\Psi}_0}{\partial u_1} & \frac{\partial\hat{\Psi}_0}{\partial u_2}\\ \frac{\partial\hat{\Psi}_1}{\partial u_0} & \frac{\partial\hat{\Psi}_1}{\partial u_1} & \frac{\partial\hat{\Psi}_1}{\partial u_2}\end{bmatrix}\begin{bmatrix}u_0\\u_1\\u_2\end{bmatrix} \quad (23)$$

Because our objective is to determine a control input *u(k)* so that outputs of the network i.e $y_1(k+1)$ and $y_2(k+2)$ follow a prespecified setpoints $r_1(k)$ and $r_2(k)$. We have to find a control low that match our purpose. In our method we use a control low similar to that given in (5). Then we replace $\hat{\hat{y}}_1(k+1)$ and $\hat{\hat{y}}_2(k+2)$ by the desiresd prespecified setpoints $r_1(k)$ and $r_2(k)$. Then we have to find a control signals u_1 and u_2 in terms of the states and desiresd prespecified setpoints. The process is shown in the following equations:

$$\begin{bmatrix} r_1(k) \\ r_2(k) \end{bmatrix} = \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial x_0} & \frac{\partial \hat{\Psi}_0}{\partial x_1} & \frac{\partial \hat{\Psi}_0}{\partial x_2} \\ \frac{\partial \hat{\Psi}_1}{\partial x_0} & \frac{\partial \hat{\Psi}_1}{\partial x_1} & \frac{\partial \hat{\Psi}_1}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial u_0} & \frac{\partial \hat{\Psi}_0}{\partial u_1} & \frac{\partial \hat{\Psi}_0}{\partial u_2} \\ \frac{\partial \hat{\Psi}_1}{\partial u_0} & \frac{\partial \hat{\Psi}_1}{\partial u_1} & \frac{\partial \hat{\Psi}_1}{\partial u_2} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \tag{24}$$

$$\begin{bmatrix} r_1(k) \\ r_2(k) \end{bmatrix} = \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial x_0} & \frac{\partial \hat{\Psi}_0}{\partial x_1} & \frac{\partial \hat{\Psi}_0}{\partial x_2} \\ \frac{\partial \hat{\Psi}_1}{\partial x_0} & \frac{\partial \hat{\Psi}_1}{\partial x_1} & \frac{\partial \hat{\Psi}_1}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial u_0} & \frac{\partial \hat{\Psi}_0}{\partial u_1} \\ \frac{\partial \hat{\Psi}_1}{\partial u_0} & \frac{\partial \hat{\Psi}_1}{\partial u_1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} + \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial u_2} \\ \frac{\partial \hat{\Psi}_1}{\partial u_2} \end{bmatrix} u_2 .$$

because $u_2 = 1$ then

$$\begin{bmatrix} r_1(k) - \frac{\partial \hat{\Psi}_0}{\partial u_2} \\ r_2(k) - \frac{\partial \hat{\Psi}_1}{\partial u_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial x_0} & \frac{\partial \hat{\Psi}_0}{\partial x_1} & \frac{\partial \hat{\Psi}_0}{\partial x_2} \\ \frac{\partial \hat{\Psi}_1}{\partial x_0} & \frac{\partial \hat{\Psi}_1}{\partial x_1} & \frac{\partial \hat{\Psi}_1}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial u_0} & \frac{\partial \hat{\Psi}_0}{\partial u_1} \\ \frac{\partial \hat{\Psi}_1}{\partial u_0} & \frac{\partial \hat{\Psi}_1}{\partial u_1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}.$$

$$\begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial u_0} & \frac{\partial \hat{\Psi}_0}{\partial u_1} \\ \frac{\partial \hat{\Psi}_1}{\partial u_0} & \frac{\partial \hat{\Psi}_1}{\partial u_1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} r_1(k) - \frac{\partial \hat{\Psi}_0}{\partial u_2} \\ r_2(k) - \frac{\partial \hat{\Psi}_1}{\partial u_2} \end{bmatrix} - \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial x_0} & \frac{\partial \hat{\Psi}_0}{\partial x_1} & \frac{\partial \hat{\Psi}_0}{\partial x_2} \\ \frac{\partial \hat{\Psi}_1}{\partial x_0} & \frac{\partial \hat{\Psi}_1}{\partial x_1} & \frac{\partial \hat{\Psi}_1}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}.$$

$$\begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial u_0} & \frac{\partial \hat{\Psi}_0}{\partial u_1} \\ \frac{\partial \hat{\Psi}_1}{\partial u_0} & \frac{\partial \hat{\Psi}_1}{\partial u_1} \end{bmatrix}^{-1} \left(\begin{bmatrix} r_1(k) - \frac{\partial \hat{\Psi}_0}{\partial u_2} \\ r_2(k) - \frac{\partial \hat{\Psi}_1}{\partial u_2} \end{bmatrix} - \begin{bmatrix} \frac{\partial \hat{\Psi}_0}{\partial x_0} & \frac{\partial \hat{\Psi}_0}{\partial x_1} & \frac{\partial \hat{\Psi}_0}{\partial x_2} \\ \frac{\partial \hat{\Psi}_1}{\partial x_0} & \frac{\partial \hat{\Psi}_1}{\partial x_1} & \frac{\partial \hat{\Psi}_1}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \right). \tag{25}$$

equation (25) represent the state feedback control low used which is similar to that given in (5).

Once the control signals are obtained using (25) it can be used instead of the old value.

The processes of the training, linearization, and obtaining the control signals are continued until no change occur in the control signals.

The performance of the proposed method for controlling the system given in this section was tested with the sinusoidal reference inputs given by

$$r_1(k) = 0.75 \sin\left[\frac{2\pi k}{50}\right] + 0.75 \sin\left[\frac{2\pi k}{10}\right]$$
$$r_1(k) = 0.75 \sin\left[\frac{2\pi k}{30}\right] + 0.75 \sin\left[\frac{2\pi k}{20}\right] \tag{26}$$

is shown in figures 3, 4, and the control signals are shown in Fig. 5.

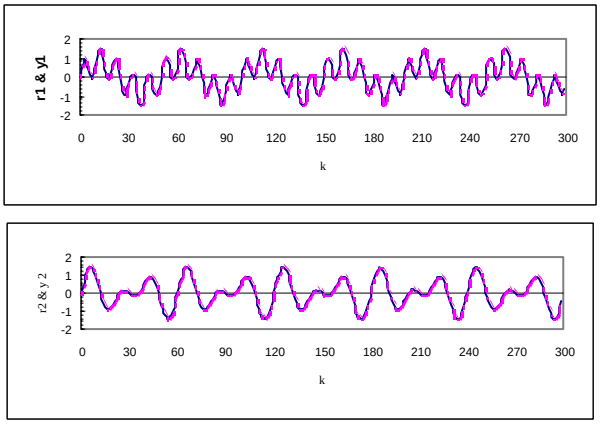

Fig. 3. Performance of linear controller when the plant equations are unknown; solid line is the desired output, and dotted line is the actual output of the plant.

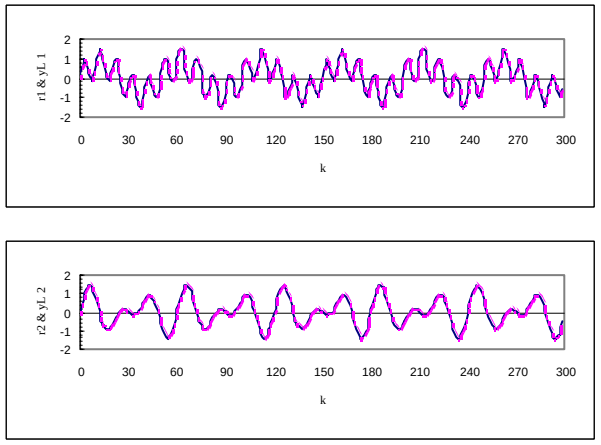

Fig. 4. Performance of linear controller on a linearized model; solid line is the desired output, and dotted line is the output of the linear model.

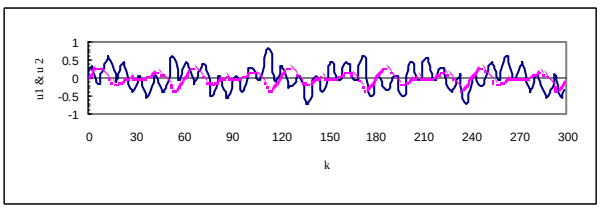

Fig. 5. The required control signals; solid line represent u_1 and dotted line represent u_2.

VIII. COMPARISON WITH OTHER WORKS

In this section, we will make a comparison between the model given above to obtain a multivariable decoupling controller with that given in [9] (section 5.3) to solve the same problem.

As was stated in [9], for the identification of the multivariable plant as well as for the design of the nonlinear decoupling controller, three-stage procedure is usually employed.

1) Identification;
2) Off-line controller design;
3): On-line adjust of identifier and controller.

Both identifier and controller are updated at every time instant based on their respective errors, the former using static back propagation and the latter using dynamic back propagation.

From above, it is clear that, dynamic back propagation needs to be used to adjust the controller parameters which is quite slow and computationally intensive as compared to their static counterparts for computation of *u(k).*

The performance of the controller in this case was tested with the sinusoidal reference inputs given in equation (26) and it is shown in Fig. 6 and the control signals are given in Fig. 7.

In comparison between the mode proposed in this paper and that given in [9] it was found that:

1) the model presented in this paper use a simple training algorithm. In the method present in [9], the training process involves dynamic gradient methods which are computationally intensive.

2) the performances given by the model given in this paper are more accurate than that obtained by the method given in [9].

IX. CONCLUSION

In this paper a strategy of using a single-layer feedforward neural network with a conventional linear controller is developed. The result strategy is an indirect adaptive control scheme. The ability of a single layer a conventional decoupling technique was used to derive the adaptive control law.neural network, with on line learning, was used to the identification of an arbitrary dynamic nonlinear system and a conventional decoupling technique was used to derive the adaptive control law.

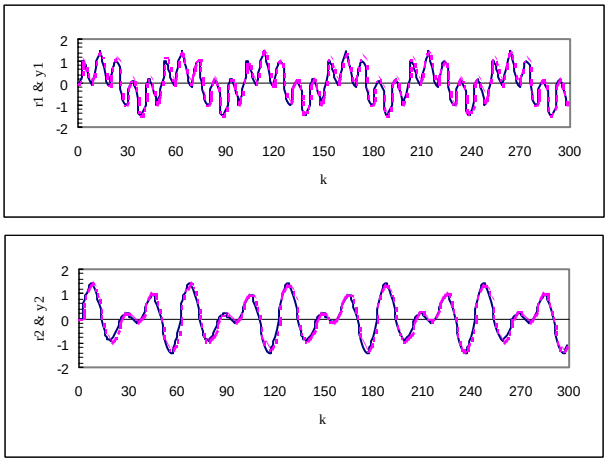

Fig. 6. Performance of neural network controller when the state variables are accessible, but the plant equations are unknown; solid line is the desired output, and dotted line is the actual output of the plant.

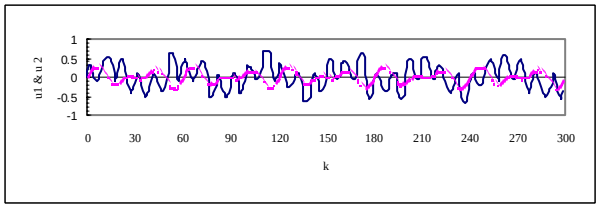

Fig. 7. The required control signals; solid line represent u_1 and dotted line represent u_2.

The effectiveness of this strategy was validated in a multivariable real system. A comparison between the model proposed and that given in [9] is made in section VIII.

From the simulation studies shown in section VII it was found that the performances of the controller described is very perfect than that given in section VII to solve the same problem.

The advantage obtained from using the strategy given in this paper than that obtained in section VII are (1) the control input u(k) can be obtained easily from the linearization process, (2) the identification model use a simple training process with a simple architecture network.

XI. REFERENCES

[1] M. Saerens and A. Soquet, “A neural controller,” in Proc. 1st Int. Conf. Artifical Neural Networks, London, Oct. 1989, pp. 211-215.

[2] M. I. Jordan and R. A. Jacobs, “Learning to control an unstable system with forward modeling,” Advances in Neural Inform. Processing Syst. Vol. 2, pp. 324-331, 1990.

[3] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” IEEE Tran. Neural Networks, vol. 1, pp. 4-27, 1990.

[4] A. U. Levin and K. S. Narendra, “control of nonlinar dynamical systems using neural networks: Controllability and stabilization,” IEEE Tran. Neural Networks, vol. 4, no. 2. pp. 192-205, 1993.

[5] A. U. Levin and K. S. Narendra, “Control of nonlinear dynamical systems using neural networks — Pare II: Observability, Identification, and control,” IEEE Tran. Neural Networks, vol. 7, no. 1, pp. 30-42, 1996.

[6] K. S. Narendra and S. Mukhopadhyay, “Adaptive control of dynamical using neural networks and approximate models,” IEEE Tran. Neural Networks, vol. 8, no. 3, pp. 475-485, 1997.

[7] J. R. Noriega and H. W. Wang, “A direct adaptive neural-network control for unknown nonlinear systems and its application,” IEEE Tran. Neural Networks, vol. 9, pp. 27-33, 1998.

[8] K. S. Narendra and S. Mukhopadhyay, “Adaptive control of nonlinear multivariable systems using neural networks,” Neural Networks, vol. 7, no. 5, pp. 737-752, 1994.

[9] P. L. Falb and W. A. Wolovich, “Decoupling in the design and synthesis of multivariable control systems,” IEEE Tran. On Automatoc control, vol. 12, pp. 651-659, 1967.