Robust Control Using LMI Transformation and Neural-Based Identification for Regulating Singularly-Perturbed Reduced Order Eigenvalue-Preserved Dynamic Systems

Anas N. Al-Rabadi
Computer Engineering Department, The University of Jordan, Amman Jordan

1. Introduction

In control engineering, robust control is an area that explicitly deals with uncertainty in its approach to the design of the system controller [7,10,24]. The methods of robust control are designed to operate properly as long as disturbances or uncertain parameters are within a compact set, where robust methods aim to accomplish robust performance and/or stability in the presence of bounded modeling errors. A robust control policy is static in contrast to the adaptive (dynamic) control policy where, rather than adapting to measurements of variations, the system controller is designed to function assuming that certain variables will be unknown but, for example, bounded. An early example of a robust control method is the high-gain feedback control where the effect of any parameter variations will be negligible with using sufficiently high gain.
The overall goal of a control system is to cause the output variable of a dynamic process to follow a desired reference variable accurately. This complex objective can be achieved based on a number of steps. A major one is to develop a mathematical description, called dynamical model, of the process to be controlled [7,10,24]. This dynamical model is usually accomplished using a set of differential equations that describe the dynamic behavior of the system, which can be further represented in state-space using system matrices or in transform-space using transfer functions [7,10,24].
In system modeling, sometimes it is required to identify some of the system parameters. This objective maybe achieved by the use of artificial neural networks (ANN), which are considered as the new generation of information processing networks [5,15,17,28,29]. Artificial neural systems can be defined as physical cellular systems which have the capability of acquiring, storing and utilizing experiential knowledge [15,29], where an ANN consists of an interconnected group of basic processing elements called neurons that perform summing operations and nonlinear function computations. Neurons are usually organized in layers and forward connections, and computations are performed in a parallel mode at all nodes and connections. Each connection is expressed by a numerical value called the weight, where the conducted learning process of a neuron corresponds to the changing of its corresponding weights.

When dealing with system modeling and control analysis, there exist equations and inequalities that require optimized solutions. An important expression which is used in robust control is called linear matrix inequality (LMI) which is used to express specific convex optimization problems for which there exist powerful numerical solvers [1,2,6]. The important LMI optimization technique was started by the Lyapunov theory showing that the differential equation $\dot{x}(t) = Ax(t)$ is stable if and only if there exists a positive definite matrix [**P**] such that $A^T P + PA < 0$ [6]. The requirement of $\{ P > 0 ,\ A^T P + PA < 0 \}$ is known as the Lyapunov inequality on [**P**] which is a special case of an LMI. By picking any $Q = Q^T > 0$ and then solving the linear equation $A^T P + PA = -Q$ for the matrix [**P**], it is guaranteed to be positive-definite if the given system is stable. The linear matrix inequalities that arise in system and control theory can be generally formulated as convex optimization problems that are amenable to computer solutions and can be solved using algorithms such as the ellipsoid algorithm [6].

In practical control design problems, the first step is to obtain a proper mathematical model in order to examine the behavior of the system for the purpose of designing an appropriate controller [1,2,3,4,5,7,8,9,10,11,12,13,14,16,17,19,20,21,22,24,25,26,27]. Sometimes, this mathematical description involves a certain small parameter (i.e., perturbation). Neglecting this small parameter results in simplifying the order of the designed controller by reducing the order of the corresponding system [1,3,4,5,8,9,11,12,13,14,17,19,20,21,22,25,26]. A reduced model can be obtained by neglecting the fast dynamics (i.e., non-dominant eigenvalues) of the system and focusing on the slow dynamics (i.e., dominant eigenvalues). This simplification and reduction of system modeling leads to controller cost minimization [7,10,13]. An example is the modern integrated circuits (ICs), where increasing package density forces developers to include side effects. Knowing that these ICs are often modeled by complex RLC-based circuits and systems, this would be very demanding computationally due to the detailed modeling of the original system [16]. In control system, due to the fact that feedback controllers don't usually consider all of the dynamics of the functioning system, model reduction is an important issue [4,5,17].

The main results in this research include the introduction of a new layered method of intelligent control, that can be used to robustly control the required system dynamics, where the new control hierarchy uses recurrent supervised neural network to identify certain parameters of the transformed system matrix [$\tilde{\mathbf{A}}$], and the corresponding LMI is used to determine the permutation matrix [**P**] so that a complete system transformation {[$\tilde{\mathbf{B}}$], [$\tilde{\mathbf{C}}$], [$\tilde{\mathbf{D}}$]} is performed. The transformed model is then reduced using the method of singular perturbation and various feedback control schemes are applied to enhance the corresponding system performance, where it is shown that the new hierarchical control method simplifies the model of the dynamical systems and therefore uses simpler controllers that produce the needed system response for specific performance enhancements. Figure 1 illustrates the layout of the utilized new control method. Layer 1 shows the continuous modeling of the dynamical system. Layer 2 shows the discrete system model. Layer 3 illustrates the neural network identification step. Layer 4 presents the undiscretization of the transformed system model. Layer 5 includes the steps for model order reduction with and without using LMI. Finally, Layer 6 presents various feedback control methods that are used in this research.

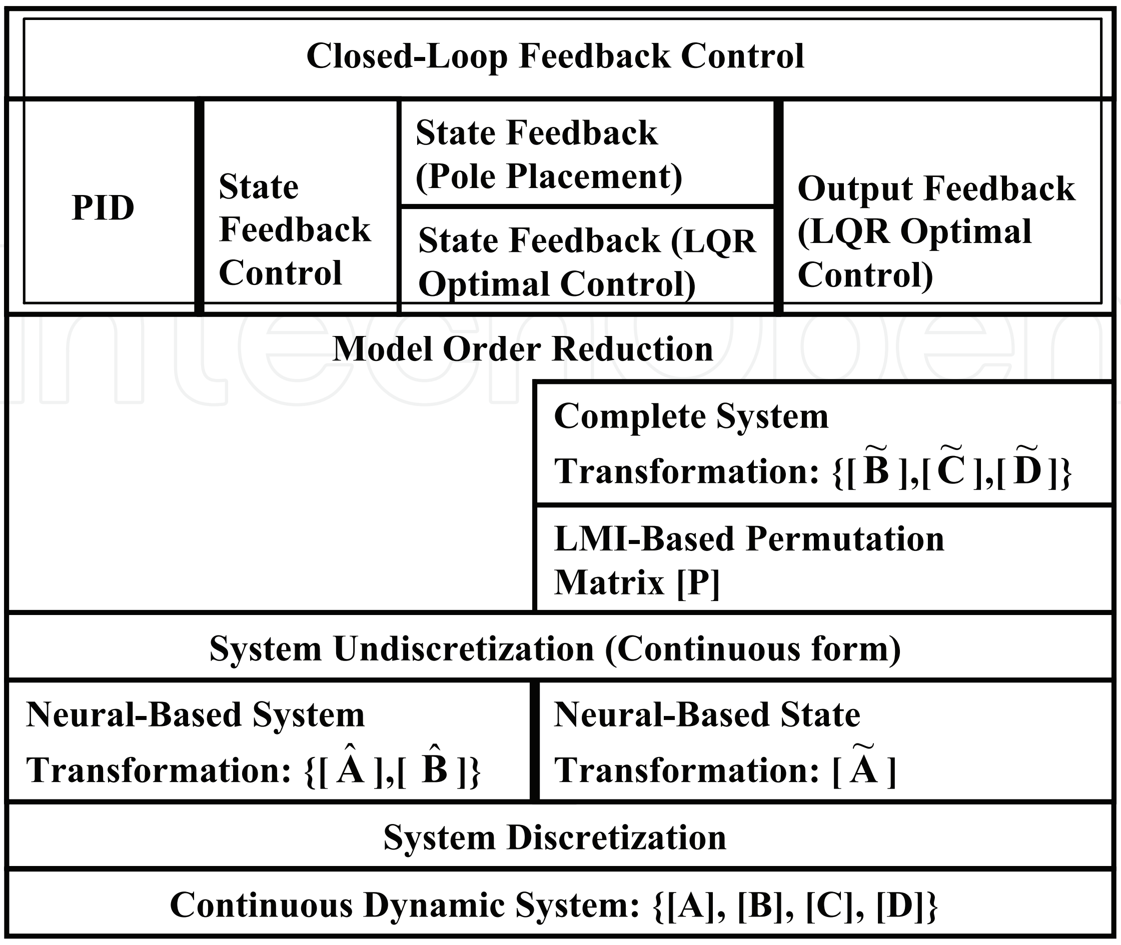

Closed-Loop Feedback Control
PID
State Feedback Control
State Feedback (Pole Placement)
State Feedback (LQR Optimal Control)
Output Feedback (LQR Optimal Control)
Model Order Reduction
Complete System Transformation: $\{[\tilde{B}],[\tilde{C}],[\tilde{D}]\}$
LMI-Based Permutation Matrix [P]
System Undiscretization (Continuous form)
Neural-Based System Transformation: $\{[\hat{A}],[\hat{B}]\}$
Neural-Based State Transformation: $[\tilde{A}]$
System Discretization
Continuous Dynamic System: {[A], [B], [C], [D]}

Fig. 1. The newly utilized hierarchical control method.

While similar hierarchical method of ANN-based identification and LMI-based transformation has been previously utilized within several applications such as for the reduced-order electronic Buck switching-mode power converter [1] and for the reduced-order quantum computation systems [2] with relatively simple state feedback controller implementations, the presented method in this work further shows the successful wide applicability of the introduced intelligent control technique for dynamical systems using various spectrum of control methods such as (a) PID-based control, (b) state feedback control using (1) pole placement-based control and (2) linear quadratic regulator (LQR) optimal control, and (c) output feedback control.

Section 2 presents background on recurrent supervised neural networks, linear matrix inequality, system model transformation using neural identification, and model order reduction. Section 3 presents a detailed illustration of the recurrent neural network identification with the LMI optimization techniques for system model order reduction. A practical implementation of the neural network identification and the associated comparative results with and without the use of LMI optimization to the dynamical system model order reduction is presented in Section 4. Section 5 presents the application of the feedback control on the reduced model using PID control, state feedback control using pole assignment, state feedback control using LQR optimal control, and output feedback control. Conclusions and future work are presented in Section 6.

2. Background

The following sub-sections provide an important background on the artificial supervised recurrent neural networks, system transformation without using LMI, state transformation using LMI, and model order reduction, which can be used for the robust control of dynamic systems, and will be used in the later Sections 3-5.

2.1 Artificial recurrent supervised neural networks

The ANN is an emulation of the biological neural system [15,29]. The basic model of the neuron is established emulating the functionality of a biological neuron which is the basic signaling unit of the nervous system. The internal process of a neuron maybe mathematically modeled as shown in Figure 2 [15,29].

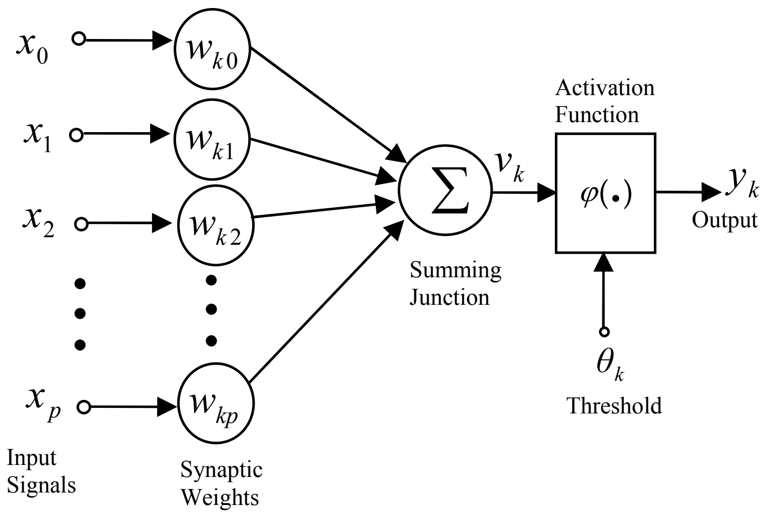


Fig. 2. A mathematical model of the artificial neuron.

As seen in Figure 2, the internal activity of the neuron is produced as:

$$v_k = \sum_{j=1}^{p} w_{kj} x_j \tag{1}$$

In supervised learning, it is assumed that at each instant of time when the input is applied, the desired response of the system is available [15,29]. The difference between the actual and the desired response represents an error measure which is used to correct the network parameters externally. Since the adjustable weights are initially assumed, the error measure may be used to adapt the network's weight matrix [**W**]. A set of input and output patterns, called a training set, is required for this learning mode, where the usually used training algorithm identifies directions of the negative error gradient and reduces the error accordingly [15,29].
The supervised recurrent neural network used for the identification in this research is based on an approximation of the method of steepest descent [15,28,29]. The network tries to match the output of certain neurons to the desired values of the system output at a specific instant of time. Consider a network consisting of a total of *N* neurons with *M* external input connections, as shown in Figure 3, for a 2nd order system with two neurons and one external input. The variable **g**(*k*) denotes the (*M* x 1) external input vector which is applied to the

network at discrete time k, the variable $\mathbf{y}(k + 1)$ denotes the corresponding (N x 1) vector of individual neuron outputs produced one step later at time (k + 1), and the input vector $\mathbf{g}(k)$ and one-step delayed output vector $\mathbf{y}(k)$ are concatenated to form the ((M + N) x 1) vector $\mathbf{u}(k)$ whose i^{th} element is denoted by $u_i(k)$. For Λ denotes the set of indices i for which $g_i(k)$ is an external input, and β denotes the set of indices i for which $u_i(k)$ is the output of a neuron (which is $y_i(k)$), the following equation is provided:

$$u_i(k) = \begin{cases} g_i(k), & \text{if } i \in \Lambda \\ y_i(k), & \text{if } i \in \beta \end{cases}$$

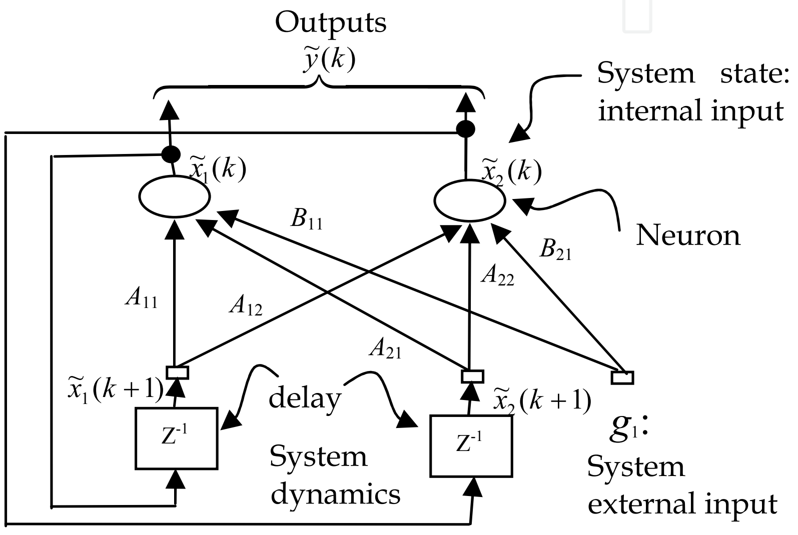


Fig. 3. The utilized 2nd order recurrent neural network architecture, where the identified matrices are given by { $\tilde{A}_d = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $\tilde{B}_d = \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix}$} and that $W = \left[[\tilde{\mathbf{A}}_\mathbf{d}] \quad [\tilde{\mathbf{B}}_\mathbf{d}] \right]$.

The (N x (M + N)) recurrent weight matrix of the network is represented by the variable [$\mathbf{W}$]. The net internal activity of neuron j at time k is given by:

$$v_j(k) = \sum_{i \in \Lambda \cup \beta} w_{ji}(k) u_i(k)$$

where $\Lambda \cup ß$ is the union of sets Λ and $ß$. At the next time step (k + 1), the output of the neuron j is computed by passing $v_j(k)$ through the nonlinearity $\varphi(.)$, thus obtaining:

$$y_j(k+1) = \varphi(v_j(k))$$

The derivation of the recurrent algorithm can be started by using $d_j(k)$ to denote the desired (target) response of neuron j at time k, and $\varsigma(k)$ to denote the set of neurons that are chosen to provide externally reachable outputs. A time-varying (N x 1) error vector $\boldsymbol{e}(k)$ is defined whose j^{th} element is given by the following relationship:

$$e_j(k) = \begin{cases} d_j(k) - y_j(k), & \text{if } j \in \varsigma(k) \\ 0, & \text{otherwise} \end{cases}$$

The objective is to minimize the cost function E_{total} which is obtained by:

$$E_{\text{total}} = \sum_k E(k), \text{ where } E(k) = \frac{1}{2}\sum_{j\in\varsigma} e_j^2(k)$$

To accomplish this objective, the method of steepest descent which requires knowledge of the gradient matrix is used:

$$\nabla_{\mathbf{W}} E_{\text{total}} = \frac{\partial E_{\text{total}}}{\partial \mathbf{W}} = \sum_k \frac{\partial E(k)}{\partial \mathbf{W}} = \sum_k \nabla_{\mathbf{W}} E(k)$$

where $\nabla_{\mathbf{W}} E(k)$ is the gradient of $E(k)$ with respect to the weight matrix [**W**]. In order to train the recurrent network in real time, the instantaneous estimate of the gradient is used $\left(\nabla_{\mathbf{W}} E(k)\right)$. For the case of a particular weight $w_{m\ell}(k)$, the incremental change $\Delta w_{m\ell}(k)$ made at k is defined as $\Delta w_{m\ell}(k) = -\eta \dfrac{\partial E(k)}{\partial w_{m\ell}(k)}$ where η is the learning-rate parameter. Therefore:

$$\frac{\partial E(k)}{\partial w_{m\ell}(k)} = \sum_{j\in\varsigma} e_j(k)\frac{\partial e_j(k)}{\partial w_{m\ell}(k)} = -\sum_{j\in\varsigma} e_j(k)\frac{\partial y_i(k)}{\partial w_{m\ell}(k)}$$

To determine the partial derivative $\partial y_j(k)/\partial w_{m\ell}(k)$, the network dynamics are derived. This derivation is obtained by using the chain rule which provides the following equation:

$$\frac{\partial y_j(k+1)}{\partial w_{m\ell}(k)} = \frac{\partial y_j(k+1)}{\partial v_j(k)}\frac{\partial v_j(k)}{\partial w_{m\ell}(k)} = \dot{\varphi}(v_j(k))\frac{\partial v_j(k)}{\partial w_{m\ell}(k)}, \text{ where } \dot{\varphi}(v_j(k)) = \frac{\partial \varphi(v_j(k))}{\partial v_j(k)}.$$

Differentiating the net internal activity of neuron j with respect to $w_{m\ell}(k)$ yields:

$$\frac{\partial v_j(k)}{\partial w_{m\ell}(k)} = \sum_{i\in\Lambda\cup\beta} \frac{\partial(w_{ji}(k)u_i(k))}{\partial w_{m\ell}(k)} = \sum_{i\in\Lambda\cup\beta}\left[w_{ji}(k)\frac{\partial u_i(k)}{\partial w_{m\ell}(k)} + \frac{\partial w_{ji}(k)}{\partial w_{m\ell}(k)}u_i(k)\right]$$

where $\left(\partial w_{ji}(k)/\partial w_{m\ell}(k)\right)$ equals "1" only when $j = m$ and $i = \ell$, and "0" otherwise. Thus:

$$\frac{\partial v_j(k)}{\partial w_{m\ell}(k)} = \sum_{i\in\Lambda\cup\beta} w_{ji}(k)\frac{\partial u_i(k)}{\partial w_{m\ell}(k)} + \delta_{mj}u_\ell(k)$$

where δ_{mj} is a Kronecker delta equals to "1" when $j = m$ and "0" otherwise, and:

$$\frac{\partial u_i(k)}{\partial w_{m\ell}(k)} = \begin{cases} 0, & \text{if } i \in \Lambda \\ \dfrac{\partial y_i(k)}{\partial w_{m\ell}(k)}, & \text{if } i \in \beta \end{cases}$$

Having those equations provides that:

$$\frac{\partial y_j(k+1)}{\partial w_{m\ell}(k)} = \dot{\varphi}(v_j(k))\left[\sum_{i\in\beta} w_{ji}(k)\frac{\partial y_i(k)}{\partial w_{m\ell}(k)} + \delta_{m\ell}u_\ell(k)\right]$$

The initial state of the network at time (k = 0) is assumed to be zero as follows:

$$\frac{\partial y_i(0)}{\partial w_{m\ell}(0)} = 0\text{, for } \{j \in ß, m \in ß, \ell \in \Lambda \cup \beta\}.$$

The dynamical system is described by the following triply-indexed set of variables ($\pi^j_{m\ell}$):

$$\pi^j_{m\ell}(k) = \frac{\partial y_j(k)}{\partial w_{m\ell}(k)}$$

For every time step k and all appropriate j, m and ℓ, system dynamics are controlled by:

$$\pi^j_{m\ell}(k+1) = \dot{\varphi}(v_j(k))\left[\sum_{i\in\beta} w_{ji}(k)\pi^i_{m\ell}(k) + \delta_{mj}u_\ell(k)\right]\text{, with } \pi^j_{m\ell}(0) = 0\ .$$

The values of $\pi^j_{m\ell}(k)$ and the error signal $e_j(k)$ are used to compute the corresponding weight changes:

$$\Delta w_{m\ell}(k) = \eta \sum_{j\in\varsigma} e_j(k)\pi^j_{m\ell}(k) \tag{2}$$

Using the weight changes, the updated weight $w_{m\ell}$ (k + 1) is calculated as follows:

$$w_{m\ell}(k+1) = w_{m\ell}(k) + \Delta w_{m\ell}(k) \tag{3}$$

Repeating this computation procedure provides the minimization of the cost function and thus the objective is achieved. With the many advantages that the neural network has, it is used for the important step of parameter identification in model transformation for the purpose of model order reduction as will be shown in the following section.

2.2 Model transformation and linear matrix inequality

In this section, the detailed illustration of system transformation using LMI optimization will be presented. Consider the dynamical system:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4}$$

$$y(t) = Cx(t) + Du(t) \tag{5}$$

The state space system representation of Equations (4) - (5) may be described by the block diagram shown in Figure 4.

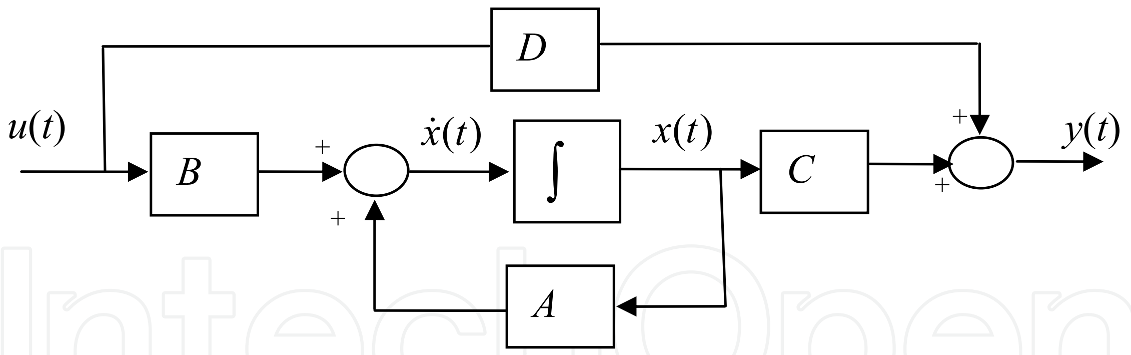


Fig. 4. Block diagram for the state-space system representation.

In order to determine the transformed [**A**] matrix, which is [$\tilde{\mathbf{A}}$], the discrete zero input response is obtained. This is achieved by providing the system with some initial state values and setting the system input to zero ($u(k)$ = 0). Hence, the discrete system of Equations (4) - (5), with the initial condition $x(0) = x_0$, becomes:

$$x(k+1) = A_d x(k) \tag{6}$$

$$y(k) = x(k) \tag{7}$$

We need $x(k)$ as an ANN target to train the network to obtain the needed parameters in [$\tilde{\mathbf{A}}_\mathbf{d}$] such that the system output will be the same for [$\mathbf{A_d}$] and [$\tilde{\mathbf{A}}_\mathbf{d}$]. Hence, simulating this system provides the state response corresponding to their initial values with only the [$\mathbf{A_d}$] matrix is being used. Once the input-output data is obtained, transforming the [$\mathbf{A_d}$] matrix is achieved using the ANN training, as will be explained in Section 3. The identified transformed [$\tilde{\mathbf{A}}_\mathbf{d}$] matrix is then converted back to the continuous form which in general (with all real eigenvalues) takes the following form:

$$\tilde{A} = \begin{bmatrix} A_r & A_c \\ 0 & A_o \end{bmatrix} \rightarrow \tilde{A} = \begin{bmatrix} \lambda_1 & \tilde{A}_{12} & \cdots & \tilde{A}_{1n} \\ 0 & \lambda_2 & \cdots & \tilde{A}_{2n} \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} \tag{8}$$

where λ_i represents the system eigenvalues. This is an upper triangular matrix that preserves the eigenvalues by (1) placing the original eigenvalues on the diagonal and (2) finding the elements $\tilde{A}_{ij}$ in the upper triangular. This upper triangular matrix form is used to produce the same eigenvalues for the purpose of eliminating the fast dynamics and sustaining the slow dynamics eigenvalues through model order reduction as will be shown in later sections.

Having the [**A**] and [$\tilde{\mathbf{A}}$] matrices, the permutation [**P**] matrix is determined using the LMI optimization technique, as will be illustrated in later sections. The complete system transformation can be achieved as follows where, assuming that $\tilde{x} = P^{-1}x$, the system of Equations (4) - (5) can be re-written as:

$$P\dot{\tilde{x}}(t) = AP\tilde{x}(t) + Bu(t)\,,\ \tilde{y}(t) = CP\tilde{x}(t) + Du(t)\,, \text{where } \tilde{y}(t) = y(t)\,.$$

Pre-multiplying the first equation above by [**P**$^{-1}$], one obtains:

$$P^{-1}P\dot{\tilde{x}}(t) = P^{-1}AP\tilde{x}(t) + P^{-1}Bu(t),\ \tilde{y}(t) = CP\tilde{x}(t) + Du(t)$$

which yields the following transformed model:

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}u(t) \tag{9}$$

$$\tilde{y}(t) = \tilde{C}\tilde{x}(t) + \tilde{D}u(t) \tag{10}$$

where the transformed system matrices are given by:

$$\tilde{A} = P^{-1}AP \tag{11}$$

$$\tilde{B} = P^{-1}B \tag{12}$$

$$\tilde{C} = CP \tag{13}$$

$$\tilde{D} = D \tag{14}$$

Transforming the system matrix [**A**] into the form shown in Equation (8) can be achieved based on the following definition [18].

Definition. *A* matrix $A \in M_n$ is called reducible if either:

a. $n = 1$ and $A = 0$; or
b. $n \geq 2$, there is a permutation matrix $P \in M_n$, and there is some integer r with $1 \leq r \leq n-1$ such that:

$$P^{-1}AP = \begin{bmatrix} X & Y \\ \mathbf{0} & Z \end{bmatrix} \tag{15}$$

where $X \in M_{r,r}$, $Z \in M_{n-r,n-r}$, $Y \in M_{r,n-r}$, and $\mathbf{0} \in M_{n-r,r}$ is a zero matrix.

The attractive features of the permutation matrix [**P**] such as being (1) orthogonal and (2) invertible have made this transformation easy to carry out. However, the permutation matrix structure narrows the applicability of this method to a limited category of applications. A form of a similarity transformation can be used to correct this problem for $\{f : R^{n\times n} \to R^{n\times n}\}$ where f is a linear operator defined by $f(A) = P^{-1}AP$ [18]. Hence, based on [**A**] and [$\tilde{\mathbf{A}}$], the corresponding LMI is used to obtain the transformation matrix [**P**], and thus the optimization problem will be casted as follows:

$$\min_{P} \left\| P - P_o \right\| \quad Subject\ to\ \left\| P^{-1}AP - \tilde{A} \right\| < \varepsilon \tag{16}$$

which can be written in an LMI equivalent form as:

$$\begin{aligned} &\min_{S}\ trace(S)\ \ Subject\ to\ \begin{bmatrix} S & P-P_o \\ (P-P_o)^T & I \end{bmatrix} > 0 \\ &\qquad\qquad\qquad\qquad\quad \begin{bmatrix} \varepsilon_1^2 I & P^{-1}AP-\tilde{A} \\ (P^{-1}AP-\tilde{A})^T & I \end{bmatrix} > 0 \end{aligned} \tag{17}$$

where S is a symmetric slack matrix [6].

2.3 System transformation using neural identification

A different transformation can be performed based on the use of the recurrent ANN while preserving the eigenvalues to be a subset of the original system. To achieve this goal, the upper triangular block structure produced by the permutation matrix, as shown in Equation (15), is used. However, based on the implementation of the ANN, finding the permutation matrix [**P**] does not have to be performed, but instead [**X**] and [**Z**] in Equation (15) will contain the system eigenvalues and [**Y**] in Equation (15) will be estimated directly using the corresponding ANN techniques. Hence, the transformation is obtained and the reduction is then achieved. Therefore, another way to obtain a transformed model that preserves the eigenvalues of the reduced model as a subset of the original system is by using ANN training without the LMI optimization technique. This may be achieved based on the assumption that the states are reachable and measurable. Hence, the recurrent ANN can identify the $[\hat{\mathrm{A}}_{\mathrm{d}}]$ and $[\hat{\mathrm{B}}_{\mathrm{d}}]$ matrices for a given input signal as illustrated in Figure 3. The ANN identification would lead to the following $[\hat{\mathrm{A}}_{\mathrm{d}}]$ and $[\hat{\mathrm{B}}_{\mathrm{d}}]$ transformations which (in the case of all real eigenvalues) construct the weight matrix [**W**] as follows:

$$W = \left[[\hat{A}_d]\ [\hat{B}_d] \right] \quad \rightarrow \quad \hat{A} = \begin{bmatrix} \lambda_1 & \hat{A}_{12} & \cdots & \hat{A}_{1n} \\ 0 & \lambda_2 & \cdots & \hat{A}_{2n} \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix},\ \hat{B} = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_n \end{bmatrix}$$

where the eigenvalues are selected as a subset of the original system eigenvalues.

2.4 Model order reduction

Linear time-invariant (LTI) models of many physical systems have fast and slow dynamics, which may be referred to as singularly perturbed systems [19]. Neglecting the fast dynamics of a singularly perturbed system provides a reduced (i.e., slow) model. This gives the advantage of designing simpler lower-dimensionality reduced-order controllers that are based on the reduced-model information.

To show the formulation of a reduced order system model, consider the singularly perturbed system [9]:

$$\dot{x}(t) = A_{11}x(t) + A_{12}\xi(t) + B_1u(t), \quad x(0) = x_0 \tag{18}$$

$$\varepsilon\dot{\xi}(t) = A_{21}x(t) + A_{22}\xi(t) + B_2u(t), \quad \xi(0) = \xi_0 \tag{19}$$

$$\mathrm{y}(t) = C_1x(t) + C_2\xi(t) \tag{20}$$

where $x \in \Re^{m_1}$ and $\xi \in \Re^{m_2}$ are the slow and fast state variables, respectively, $u \in \Re^{n_1}$ and $y \in \Re^{n_2}$ are the input and output vectors, respectively, $\{[\mathbf{A_{ii}}], [\mathbf{B_i}], [\mathbf{C_i}]\}$ are constant matrices of appropriate dimensions with $i \in \{1, 2\}$, and ε is a small positive constant. The singularly perturbed system in Equations (18)-(20) is simplified by setting $\varepsilon = 0$ [3,14,27]. In doing so, we are neglecting the fast dynamics of the system and assuming that the state variables ξ have reached the quasi-steady state. Hence, setting $\varepsilon = 0$ in Equation (19), with the assumption that [$\mathbf{A_{22}}$] is nonsingular, produces:

$$\xi(t) = -A_{22}^{-1}A_{21}x_r(t) - A_{22}^{-1}B_1u(t) \tag{21}$$

where the index *r* denotes the remained or reduced model. Substituting Equation (21) in Equations (18)-(20) yields the following reduced order model:

$$\dot{x}_r(t) \; = A_r x_r(t) + B_r u(t) \tag{22}$$

$$y(t) = C_r x_r(t) + D_r u(t) \tag{23}$$

where { $A_r = A_{11} - A_{12}A_{22}^{-1}A_{21}$, $B_r = B_1 - A_{12}A_{22}^{-1}B_2$, $C_r = C_1 - C_2A_{22}^{-1}A_{21}$, $D_r = -C_2A_{22}^{-1}B_2$ }.

3. Neural network identification with lmi optimization for the system model order reduction

In this work, it is our objective to search for a similarity transformation that can be used to decouple a pre-selected eigenvalue set from the system matrix [**A**]. To achieve this objective, training the neural network to identify the transformed discrete system matrix [$\tilde{\mathbf{A}}_\mathbf{d}$] is performed [1,2,15,29]. For the system of Equations (18)-(20), the discrete model of the dynamical system is obtained as:

$$x(k+1) = A_d x(k) + B_d u(k) \tag{24}$$

$$y(k) = C_d x(k) + D_d u(k) \tag{25}$$

The identified discrete model can be written in a detailed form (as was shown in Figure 3) as follows:

$$\begin{bmatrix} \tilde{x}_1(k+1) \\ \tilde{x}_2(k+1) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \tilde{x}_1(k) \\ \tilde{x}_2(k) \end{bmatrix} + \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix} u(k) \tag{26}$$

$$\tilde{y}(k) = \begin{bmatrix} \tilde{x}_1(k) \\ \tilde{x}_2(k) \end{bmatrix} \tag{27}$$

where *k* is the time index, and the detailed matrix elements of Equations (26)-(27) were shown in Figure 3 in the previous section.

The recurrent ANN presented in Section 2.1 can be summarized by defining Λ as the set of indices *i* for which $g_i(k)$ is an external input, defining β as the set of indices *i* for which $y_i(k)$ is an internal input or a neuron output, and defining $u_i(k)$ as the combination of the internal and external inputs for which $i \in \beta \cup \Lambda$. Using this setting, training the ANN depends on the internal activity of each neuron which is given by:

$$v_j(k) = \sum_{i \in \Lambda \cup \beta} w_{ji}(k) u_i(k) \tag{28}$$

where w_{ji} is the weight representing an element in the system matrix or input matrix for $j \in ß$ and $i \in ß \cup \Lambda$ such that $W = \left[[\tilde{\mathbf{A}}_{\mathbf{d}}] \quad [\tilde{\mathbf{B}}_{\mathbf{d}}] \right]$. At the next time step (k +1), the output (internal input) of the neuron j is computed by passing the activity through the nonlinearity $\varphi(.)$ as follows:

$$x_j(k+1) = \varphi(v_j(k)) \tag{29}$$

With these equations, based on an approximation of the method of steepest descent, the ANN identifies the system matrix $[\mathbf{A_d}]$ as illustrated in Equation (6) for the zero input response. That is, an error can be obtained by matching a true state output with a neuron output as follows:

$$e_j(k) = x_j(k) - \tilde{x}_j(k)$$

Now, the objective is to minimize the cost function given by:

$$E_{\text{total}} = \sum_k E(k) \ \text{ and } \ E(k) = \tfrac{1}{2} \sum_{j \in \varsigma} e_j^2(k)$$

where ς denotes the set of indices j for the output of the neuron structure. This cost function is minimized by estimating the instantaneous gradient of $E(k)$ with respect to the weight matrix $[\mathbf{W}]$ and then updating $[\mathbf{W}]$ in the negative direction of this gradient [15,29]. In steps, this may be proceeded as follows:

- Initialize the weights $[\mathbf{W}]$ by a set of uniformly distributed random numbers. Starting at the instant (k = 0), use Equations (28) - (29) to compute the output values of the N neurons (where $N = ß$).
- For every time step k and all $j \in ß$, $m \in ß$ and $\ell \in ß \cup \Lambda$, compute the dynamics of the system which are governed by the triply-indexed set of variables:

$$\pi_{m\ell}^{j}(k+1) = \dot{\varphi}(v_j(k)) \left[\sum_{i \in ß} w_{ji}(k) \pi_{m\ell}^{i}(k) + \delta_{mj} u_\ell(k) \right]$$

  with initial conditions $\pi_{m\ell}^{j}(0) = 0$ and δ_{mj} is given by $\left(\partial w_{ji}(k) / \partial w_{m\ell}(k) \right)$, which is equal to "1" only when {$j = m,\ i = \ell$ } and otherwise it is "0". Notice that, for the special case of a sigmoidal nonlinearity in the form of a logistic function, the derivative $\dot{\varphi}(\cdot)$ is given by $\dot{\varphi}(v_j(k)) = y_j(k+1)[1 - y_j(k+1)]$.
- Compute the weight changes corresponding to the error signal and system dynamics:

$$\Delta w_{m\ell}(k) = \eta \sum_{j \in \varsigma} e_j(k) \pi_{m\ell}^{j}(k) \tag{30}$$

- Update the weights in accordance with:

$$w_{m\ell}(k+1) = w_{m\ell}(k) + \Delta w_{m\ell}(k) \tag{31}$$

- Repeat the computation until the desired identification is achieved.

As illustrated in Equations (6) - (7), for the purpose of estimating only the transformed system matrix [$\tilde{\mathbf{A}}_{\mathbf{d}}$], the training is based on the zero input response. Once the training is completed, the obtained weight matrix [**W**] will be the discrete identified transformed system matrix [$\tilde{\mathbf{A}}_{\mathbf{d}}$]. Transforming the identified system back to the continuous form yields the desired continuous transformed system matrix [$\tilde{\mathbf{A}}$]. Using the LMI optimization technique, which was illustrated in Section 2.2, the permutation matrix [**P**] is then determined. Hence, a complete system transformation, as shown in Equations (9) - (10), will be achieved. For the model order reduction, the system in Equations (9) - (10) can be written as:

$$\begin{bmatrix} \dot{\tilde{x}}_r(t) \\ \dot{\tilde{x}}_o(t) \end{bmatrix} = \begin{bmatrix} A_r & A_c \\ 0 & A_o \end{bmatrix} \begin{bmatrix} \tilde{x}_r(t) \\ \tilde{x}_o(t) \end{bmatrix} + \begin{bmatrix} B_r \\ B_o \end{bmatrix} u(t) \tag{32}$$

$$\begin{bmatrix} \tilde{y}_r(t) \\ \tilde{y}_o(t) \end{bmatrix} = \begin{bmatrix} C_r & C_o \end{bmatrix} \begin{bmatrix} \tilde{x}_r(t) \\ \tilde{x}_o(t) \end{bmatrix} + \begin{bmatrix} D_r \\ D_o \end{bmatrix} u(t) \tag{33}$$

The following system transformation enables us to decouple the original system into retained (*r*) and omitted (*o*) eigenvalues. The retained eigenvalues are the dominant eigenvalues that produce the slow dynamics and the omitted eigenvalues are the non-dominant eigenvalues that produce the fast dynamics. Equation (32) maybe written as:

$$\dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + A_c\tilde{x}_o(t) + B_r u(t) \text{ and } \dot{\tilde{x}}_o(t) = A_o\tilde{x}_o(t) + B_o u(t)$$

The coupling term $A_c\tilde{x}_o(t)$ maybe compensated for by solving for $\tilde{x}_o(t)$ in the second equation above by setting $\dot{\tilde{x}}_o(t)$ to zero using the singular perturbation method (by setting $\varepsilon = 0$). By performing this, the following equation is obtained:

$$\tilde{x}_o(t) = -A_o^{-1}B_o u(t) \tag{34}$$

Using $\tilde{x}_o(t)$, we get the reduced order model given by:

$$\dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + [-A_cA_o^{-1}B_o + B_r]u(t) \tag{35}$$

$$y(t) = C_r\tilde{x}_r(t) + [-C_oA_o^{-1}B_o + D]u(t) \tag{36}$$

Hence, the overall reduced order model may be represented by:

$$\dot{\tilde{x}}_r(t) \ = A_{or}\tilde{x}_r(t) + B_{or}u(t) \tag{37}$$

$$y(t) = C_{or}\tilde{x}_r(t) + D_{or}u(t) \tag{38}$$

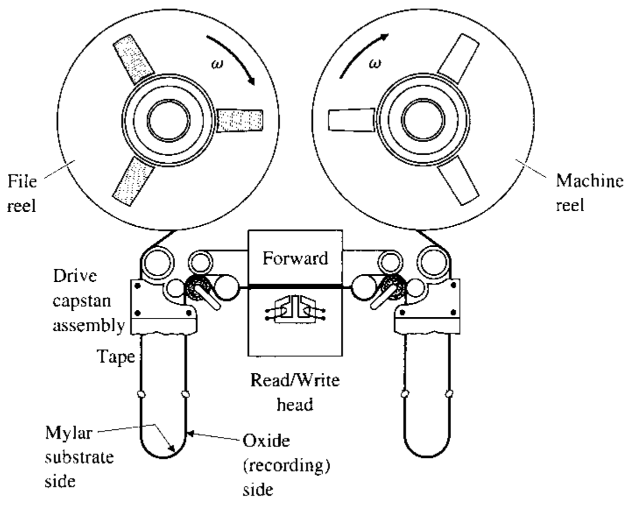
where the details of the {[$\mathbf{A}_{\mathbf{or}}$], [$\mathbf{B}_{\mathbf{or}}$], [$\mathbf{C}_{\mathbf{or}}$], [$\mathbf{D}_{\mathbf{or}}$]} overall reduced matrices were shown in Equations (35) - (36), respectively.

4. Examples for the dynamic system order reduction using neural identification

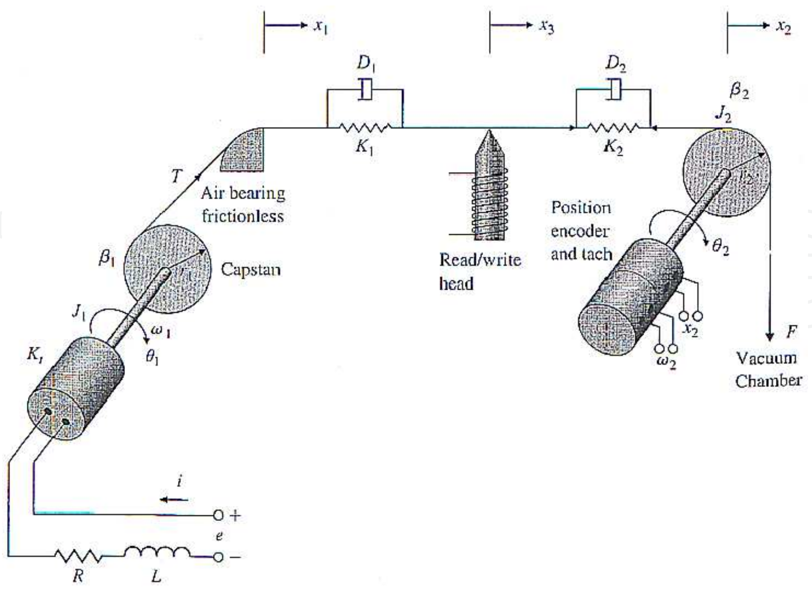
The following subsections present the implementation of the new proposed method of system modeling using supervised ANN, with and without using LMI, and using model order reduction, that can be directly utilized for the robust control of dynamic systems. The presented simulations were tested on a PC platform with hardware specifications of Intel Pentium 4 CPU 2.40 GHz, and 504 MB of RAM, and software specifications of MS Windows XP 2002 OS and Matlab 6.5 simulator.

4.1 Model reduction using neural-based state transformation and lmi-based complete system transformation

The following example illustrates the idea of dynamic system model order reduction using LMI with comparison to the model order reduction without using LMI. Let us consider the system of a high-performance tape transport which is illustrated in Figure 5. As seen in Figure 5, the system is designed with a small capstan to pull the tape past the read/write heads with the take-up reels turned by DC motors [10].



(a)



(b)

Fig. 5. The used tape drive system: (a) a front view of a typical tape drive mechanism, and (b) a schematic control model.

As can be shown, in static equilibrium, the tape tension equals the vacuum force ($T_o = F$) and the torque from the motor equals the torque on the capstan ($K_t i_o = r_1 T_o$) where T_o is the tape tension at the read/write head at equilibrium, *F* is the constant force (i.e., tape tension for vacuum column), *K* is the motor torque constant, i_o is the equilibrium motor current, and r_1 is the radius of the capstan take-up wheel.
The system variables are defined as deviations from this equilibrium, and the system equations of motion are given as follows:

$$J_1 = \frac{d\omega_1}{dt} + \beta_1\omega_1 - r_1 T + K_t i\,,\ \dot{x}_1 = r_1\omega_1$$

$$L\frac{di}{dt} Ri + K_e\omega_1 = e\,,\ \dot{x}_2 = r_2\omega_2$$

$$J_2\frac{d\omega_2}{dt} + \beta_2\omega_2 + r_2 T = 0$$

$$T = K_1(x_3 - x_1) + D_1(\dot{x}_3 - \dot{x}_1)$$

$$T = K_2(x_2 - x_3) + D_2(\dot{x}_2 - \dot{x}_3)$$

$$x_1 = r_1\theta_1\,,\ x_2 = r_2\theta_2\,,\ x_3 = \frac{x_1 - x_2}{2}$$

where $D_{1,2}$ is the damping in the tape-stretch motion, *e* is the applied input voltage (*V*), *i* is the current into capstan motor, J_1 is the combined inertia of the wheel and take-up motor, J_2 is the inertia of the idler, $K_{1,2}$ is the spring constant in the tape-stretch motion, K_e is the electric constant of the motor, K_t is the torque constant of the motor, *L* is the armature inductance, *R* is the armature resistance, r_1 is the radius of the take-up wheel, r_2 is the radius of the tape on the idler, *T* is the tape tension at the read/write head, x_3 is the position of the tape at the head, $\dot{x}_3$ is the velocity of the tape at the head, β_1 is the viscous friction at take-up wheel, β_2 is the viscous friction at the wheel, θ_1 is the angular displacement of the capstan, θ_2 is the tachometer shaft angle, ω_1 is the speed of the drive wheel $\dot{\theta}_1$, and ω_2 is the output speed measured by the tachometer output $\dot{\theta}_2$.
The state space form is derived from the system equations, where there is one input, which is the applied voltage, three outputs which are (1) tape position at the head, (2) tape tension, and (3) tape position at the wheel, and five states which are (1) tape position at the air bearing, (2) drive wheel speed, (3) tape position at the wheel, (4) tachometer output speed, and (5) capstan motor speed. The following sub-sections will present the simulation results for the investigation of different system cases using transformations with and without utilizing the LMI optimization technique.

4.1.1 System transformation using neural identification without utilizing linear matrix inequality

This sub-section presents simulation results for system transformation using ANN-based identification and without using LMI.
Case #1. Let us consider the following case of the tape transport:

$$\dot{x}(t) = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ -1.1 & -1.35 & 1.1 & 3.1 & 0.75 \\ 0 & 0 & 0 & 5 & 0 \\ 1.35 & 1.4 & -2.4 & -11.4 & 0 \\ 0 & -0.03 & 0 & 0 & -10 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t)\,,$$

$$y(t)=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ -0.2 & -0.2 & 0.2 & 0.2 & 0 \end{bmatrix}x(t)$$

The five eigenvalues are {-10.5772, -9.999, -0.9814, -0.5962 ± j0.8702}, where two eigenvalues are complex and three are real, and thus since (1) not all the eigenvalues are complex and (2) the existing real eigenvalues produce the fast dynamics that we need to eliminate, model order reduction can be applied. As can be seen, two real eigenvalues produce fast dynamics {-10.5772, -9.999} and one real eigenvalue produce slow dynamics {-0.9814}. In order to obtain the reduced model, the reduction based on the identification of the input matrix [$\hat{\mathbf{B}}$] and the transformed system matrix [$\hat{\mathbf{A}}$] was performed. This identification is achieved utilizing the recurrent ANN.
By discretizing the above system with a sampling time T_s = 0.1 sec., using a step input with learning time T_l = 300 sec., and then training the ANN for the input/output data with a learning rate η = 0.005 and with initial weights w = [[$\hat{\mathrm{A}}_\mathrm{d}$] [$\hat{\mathrm{B}}_\mathrm{d}$]] given as:

$$w=\begin{bmatrix} -0.0059 & -0.0360 & 0.0003 & -0.0204 & -0.0307 & 0.0499 \\ -0.0283 & 0.0243 & 0.0445 & -0.0302 & -0.0257 & -0.0482 \\ 0.0359 & 0.0222 & 0.0309 & 0.0294 & -0.0405 & 0.0088 \\ -0.0058 & 0.0212 & -0.0225 & -0.0273 & 0.0079 & 0.0152 \\ 0.0295 & -0.0235 & -0.0474 & -0.0373 & -0.0158 & -0.0168 \end{bmatrix}$$

produces the transformed model for the system and input matrices, $[\hat{\mathbf{A}}]$ and $[\hat{\mathbf{B}}]$, as follows:

$$\dot{x}(t)=\begin{bmatrix} -0.5967 & 0.8701 & -0.1041 & -0.2710 & -0.4114 \\ -0.8701 & -0.5967 & 0.8034 & -0.4520 & -0.3375 \\ 0 & 0 & -0.9809 & 0.4962 & -0.4680 \\ 0 & 0 & 0 & -9.9985 & 0.0146 \\ 0 & 0 & 0 & 0 & -10.5764 \end{bmatrix}x(t)+\begin{bmatrix} 0.1414 \\ 0.0974 \\ 0.1307 \\ -0.0011 \\ 1.0107 \end{bmatrix}u(t)$$

$$y(t)=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ -0.2 & -0.2 & 0.2 & 0.2 & 0 \end{bmatrix}x(t)$$

As observed, all of the system eigenvalues have been preserved in this transformed model with a little difference due to discretization. Using the singular perturbation technique, the following reduced 3rd order model is obtained as follows:

$$\dot{x}(t)=\begin{bmatrix} -0.5967 & 0.8701 & -0.1041 \\ -0.8701 & -0.5967 & 0.8034 \\ 0 & 0 & -0.9809 \end{bmatrix}x(t)+\begin{bmatrix} 0.1021 \\ 0.0652 \\ 0.0860 \end{bmatrix}u(t)$$

$$y(t)=\begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0.5 \\ -0.2 & -0.2 & 0.2 \end{bmatrix}x(t)+\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}u(t)$$

It is also observed in the above model that the reduced order model has preserved all of its eigenvalues {-0.9809, -0.5967 ± j0.8701} which are a subset of the original system, while the reduced order model obtained using the singular perturbation without system transformation has provided different eigenvalues {-0.8283, -0.5980 ± j0.9304}.
Evaluations of the reduced order models (transformed and non-transformed) were obtained by simulating both systems for a step input. Simulation results are shown in Figure 6.

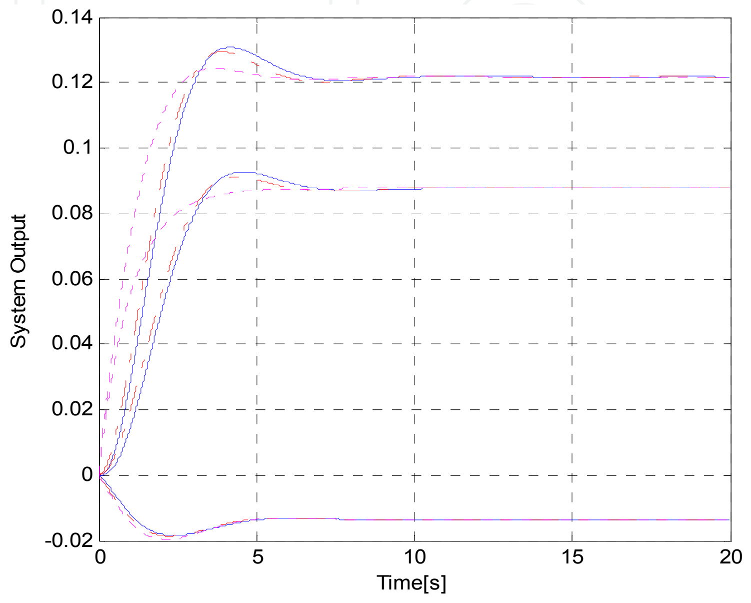


Fig. 6. Reduced 3rd order models (.... transformed, -.-.-.- non-transformed) output responses to a step input along with the non-reduced model (____ original) 5th order system output response.

Based on Figure 6, it is seen that the non-transformed reduced model provides a response which is better than the transformed reduced model. The cause of this is that the transformation at this point is performed only for the [**A**] and [**B**] system matrices leaving the [**C**] matrix unchanged. Therefore, the system transformation is further considered for complete system transformation using LMI (for {[**A**], [**B**], [**D**]}) as will be seen in subsection 4.1.2, where LMI-based transformation will produce better reduction-based response results than both the non-transformed and transformed without LMI.

Case #2. Consider now the following case:

$$\dot{x}(t)=\begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ -1.1 & -1.35 & 0.1 & 0.1 & 0.75 \\ 0 & 0 & 0 & 2 & 0 \\ 0.35 & 0.4 & -0.4 & -2.4 & 0 \\ 0 & -0.03 & 0 & 0 & -10 \end{bmatrix} x(t)+\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t)\,,\; y(t)=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ -0.2 & -0.2 & 0.2 & 0.2 & 0 \end{bmatrix} x(t)$$

The five eigenvalues are {-9.9973, -2.0002, -0.3696, -0.6912 ± j1.3082}, where two eigenvalues are complex, three are real, and only one eigenvalue is considered to produce fast dynamics {-9.9973}. Using the discretized model with T_s = 0.071 sec. for a step input with learning time T_l = 70 sec., and through training the ANN for the input/output data with η = 3.5 x 10^{-5} and initial weight matrix given by:

$$w=\begin{bmatrix} -0.0195 & 0.0194 & -0.0130 & 0.0071 & -0.0048 & 0.0029 \\ -0.0189 & 0.0055 & 0.0196 & -0.0025 & -0.0053 & 0.0120 \\ -0.0091 & 0.0168 & 0.0031 & 0.0031 & 0.0134 & -0.0038 \\ -0.0061 & 0.0068 & 0.0193 & 0.0145 & 0.0038 & -0.0139 \\ -0.0150 & 0.0204 & -0.0073 & 0.0180 & -0.0085 & -0.0161 \end{bmatrix}$$

and by applying the singular perturbation reduction technique, a reduced 4th order model is obtained as follows:

$$\dot{x}(t)=\begin{bmatrix} -0.6912 & 1.3081 & -0.4606 & 0.0114 \\ -1.3081 & -0.6912 & 0.6916 & -0.0781 \\ 0 & 0 & -0.3696 & 0.0113 \\ 0 & 0 & 0 & -2.0002 \end{bmatrix}x(t)+\begin{bmatrix} 0.0837 \\ 0.0520 \\ 0.0240 \\ -0.0014 \end{bmatrix}u(t)$$

$$y(t)=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ -0.2 & -0.2 & 0.2 & 0.2 \end{bmatrix}x(t)$$

where all the eigenvalues {-2.0002, -0.3696, -0.6912 ± j1.3081} are preserved as a subset of the original system. This reduced 4th order model is simulated for a step input and then compared to both of the reduced model without transformation and the original system response. Simulation results are shown in Figure 7 where again the non-transformed reduced order model provides a response that is better than the transformed reduced model. The reason for this follows closely the explanation provided for the previous case.

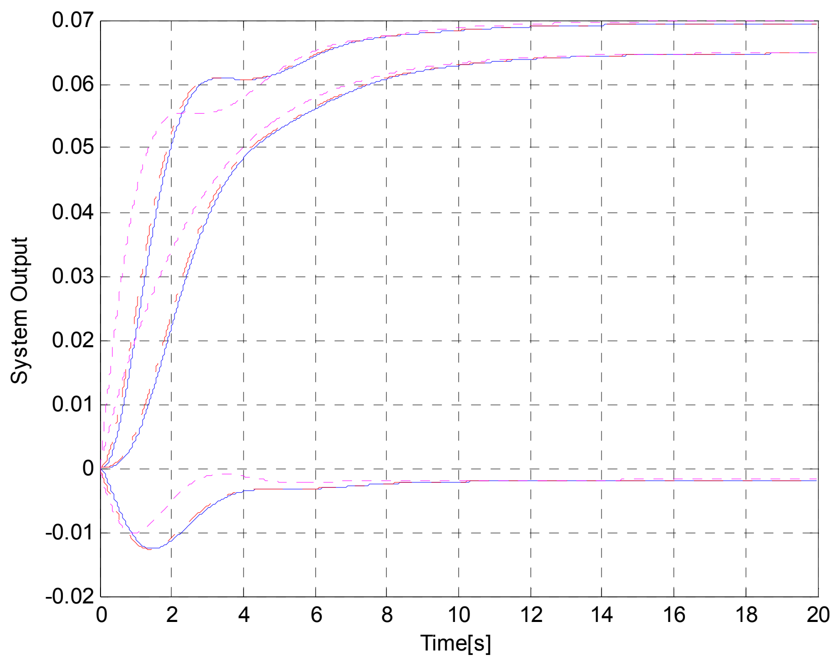


Fig. 7. Reduced 4th order models (…. transformed, -.-.-.- non-transformed) output responses to a step input along with the non-reduced (____ original) 5th order system output response.

Case #3. Let us consider the following system:

$$\dot{x}(t) = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ -0.1 & -1.35 & 0.1 & 04.1 & 0.75 \\ 0 & 0 & 0 & 5 & 0 \\ 0.35 & 0.4 & -1.4 & -5.4 & 0 \\ 0 & -0.03 & 0 & 0 & -10 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t),\ y(t) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ -0.2 & -0.2 & 0.2 & 0.2 & 0 \end{bmatrix} x(t)$$

The eigenvalues are {-9.9973, -3.9702, -1.8992, -0.6778, -0.2055} which are all real. Utilizing the discretized model with T_s = 0.1 sec. for a step input with learning time T_l = 500 sec., and training the ANN for the input/output data with η = 1.25 x 10^{-5}, and initial weight matrix given by:

$$w = \begin{bmatrix} 0.0014 & -0.0662 & 0.0298 & -0.0072 & -0.0523 & -0.0184 \\ 0.0768 & 0.0653 & -0.0770 & -0.0858 & -0.0968 & -0.0609 \\ 0.0231 & 0.0223 & -0.0053 & 0.0162 & -0.0231 & 0.0024 \\ -0.0907 & 0.0695 & 0.0366 & 0.0132 & 0.0515 & 0.0427 \\ 0.0904 & -0.0772 & -0.0733 & -0.0490 & 0.0150 & 0.0735 \end{bmatrix}$$

and then by applying the singular perturbation technique, the following reduced 3rd order model is obtained:

$$\dot{x}(t) = \begin{bmatrix} -0.2051 & -1.5131 & 0.6966 \\ 0 & -0.6782 & -0.0329 \\ 0 & 0 & -1.8986 \end{bmatrix} x(t) + \begin{bmatrix} 0.0341 \\ 0.0078 \\ 0.4649 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0.5 \\ -0.2 & -0.2 & 0.2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0.0017 \end{bmatrix} u(t)$$

Again, it is seen here the preservation of the eigenvalues of the reduced-order model being as a subset of the original system. However, as shown before, the reduced model without system transformation provided different eigenvalues {-1.5165,-0.6223,-0.2060} from the transformed reduced order model. Simulating both systems for a step input provided the results shown in Figure 8.
In Figure 8, it is also seen that the response of the non-transformed reduced model is better than the transformed reduced model, which is again caused by leaving the output [**C**] matrix without transformation.

4.1.2 LMI-based state transformation using neural identification

As observed in the previous subsection, the system transformation without using the LMI optimization method, where its objective was to preserve the system eigenvalues in the reduced model, didn't provide an acceptable response as compared with either the reduced non-transformed or the original responses.
As was mentioned, this was due to the fact of not transforming the complete system (i.e., by neglecting the [**C**] matrix). In order to achieve better response, we will now perform a

complete system transformation utilizing the LMI optimization technique to obtain the permutation matrix [**P**] based on the transformed system matrix [$\tilde{\mathbf{A}}$] as resulted from the ANN-based identification, where the following presents simulations for the previously considered tape drive system cases.

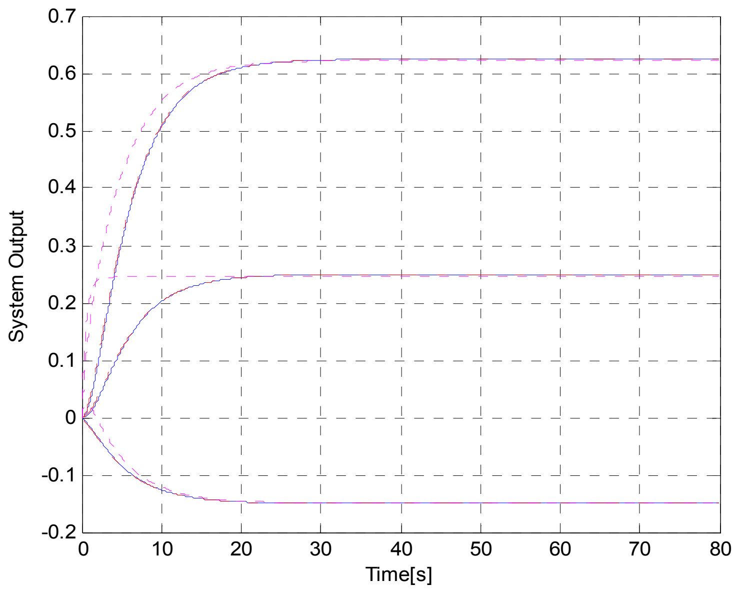


Fig. 8. Reduced 3rd order models (…. transformed, -.-.-.- non-transformed) output responses to a step input along with the non-reduced (____ original) 5th order system output response.

Case #1. For the example of case #1 in subsection 4.1.1, the ANN identification is used now to identify only the transformed [$\tilde{\mathbf{A}}_{\mathbf{d}}$] matrix. Discretizing the system with T_s = 0.1 sec., using a step input with learning time T_l = 15 sec., and training the ANN for the input/output data with η = 0.001 and initial weights for the [$\tilde{\mathbf{A}}_{\mathbf{d}}$] matrix as follows:

$$w = \begin{bmatrix} 0.0286 & 0.0384 & 0.0444 & 0.0206 & 0.0191 \\ 0.0375 & 0.0440 & 0.0325 & 0.0398 & 0.0144 \\ 0.0016 & 0.0186 & 0.0307 & 0.0056 & 0.0304 \\ 0.0411 & 0.0226 & 0.0478 & 0.0287 & 0.0453 \\ 0.0327 & 0.0042 & 0.0239 & 0.0106 & 0.0002 \end{bmatrix}$$

produces the transformed system matrix:

$$\tilde{A} = \begin{bmatrix} -0.5967 & 0.8701 & -1.4633 & -0.9860 & 0.0964 \\ -0.8701 & -0.5967 & 0.2276 & 0.6165 & 0.2114 \\ 0 & 0 & -0.9809 & 0.1395 & 0.4934 \\ 0 & 0 & 0 & -9.9985 & 1.0449 \\ 0 & 0 & 0 & 0 & -10.5764 \end{bmatrix}$$

Based on this transformed matrix, using the LMI technique, the permutation matrix [**P**] was computed and then used for the complete system transformation. Therefore, the transformed {[$\tilde{\mathbf{B}}$], [$\tilde{\mathbf{C}}$], [$\tilde{\mathbf{D}}$]} matrices were then obtained. Performing model order reduction provided the following reduced 3rd order model:

$$\dot{x}(t)=\begin{bmatrix}-0.5967 & 0.8701 & -1.4633\\ -0.8701 & -0.5967 & 0.2276\\ 0 & 0 & -0.9809\end{bmatrix}x(t)+\begin{bmatrix}35.1670\\ -47.3374\\ -4.1652\end{bmatrix}u(t)$$

$$y(t)=\begin{bmatrix}-0.0019 & 0 & -0.0139\\ -0.0024 & -0.0009 & -0.0088\\ -0.0001 & 0.0004 & -0.0021\end{bmatrix}x(t)+\begin{bmatrix}-0.0025\\ -0.0025\\ 0.0006\end{bmatrix}u(t)$$

where the objective of eigenvalue preservation is clearly achieved. Investigating the performance of this new LMI-based reduced order model shows that the new *completely transformed system* is better than all the previous reduced models (transformed and non-transformed). This is clearly shown in Figure 9 where the 3rd order reduced model, based on the LMI optimization transformation, provided a response that is almost the same as the 5th order original system response.

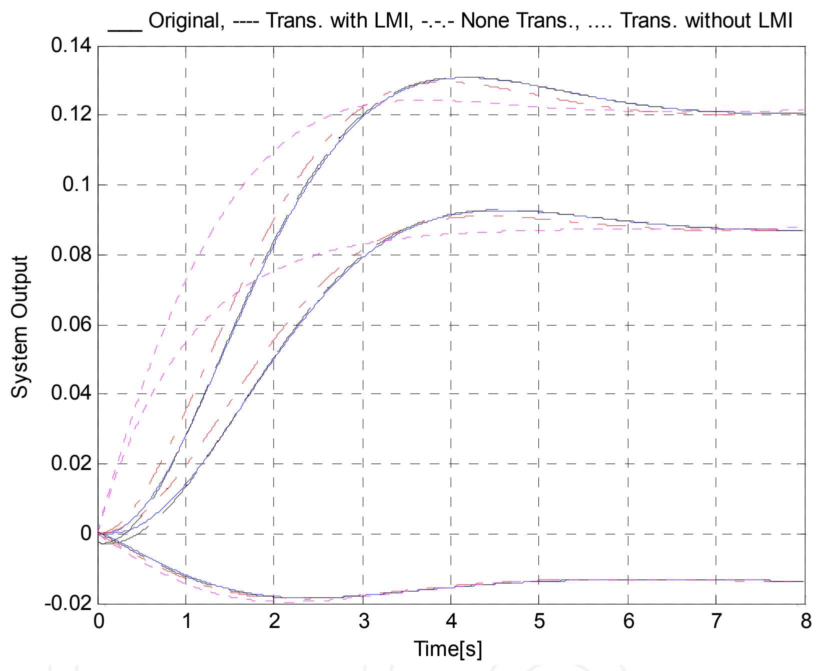


Fig. 9. Reduced 3rd order models (…. transformed without LMI, -.-.-.- non-transformed, ---- transformed with LMI) output responses to a step input along with the non reduced (____ original) system output response. The LMI-transformed curve fits almost exactly on the original response.

Case #2. For the example of case #2 in subsection 4.1.1, for T_s = 0.1 sec., 200 input/output data learning points, and η = 0.0051 with initial weights for the [$\tilde{\mathbf{A}}_{\mathbf{d}}$] matrix as follows:

$$w=\begin{bmatrix}0.0332 & 0.0682 & 0.0476 & 0.0129 & 0.0439\\ 0.0317 & 0.0610 & 0.0575 & 0.0028 & 0.0691\\ 0.0745 & 0.0516 & 0.0040 & 0.0234 & 0.0247\\ 0.0459 & 0.0231 & 0.0086 & 0.0611 & 0.0154\\ 0.0706 & 0.0418 & 0.0633 & 0.0176 & 0.0273\end{bmatrix}$$

the transformed [$\tilde{\mathbf{A}}$] was obtained and used to calculate the permutation matrix [**P**]. The complete system transformation was then performed and the reduction technique produced the following 3rd order reduced model:

$$\dot{x}(t)=\begin{bmatrix}-0.6910 & 1.3088 & -3.8578\\ -1.3088 & -0.6910 & -1.5719\\ 0 & 0 & -0.3697\end{bmatrix}x(t)+\begin{bmatrix}-0.7621\\ -0.1118\\ 0.4466\end{bmatrix}u(t)$$

$$y(t)=\begin{bmatrix}0.0061 & 0.0261 & 0.0111\\ -0.0459 & 0.0187 & -0.0946\\ 0.0117 & 0.0155 & -0.0080\end{bmatrix}x(t)+\begin{bmatrix}0.0015\\ 0.0015\\ 0.0014\end{bmatrix}u(t)$$

with eigenvalues preserved as desired. Simulating this reduced order model to a step input, as done previously, provided the response shown in Figure 10.

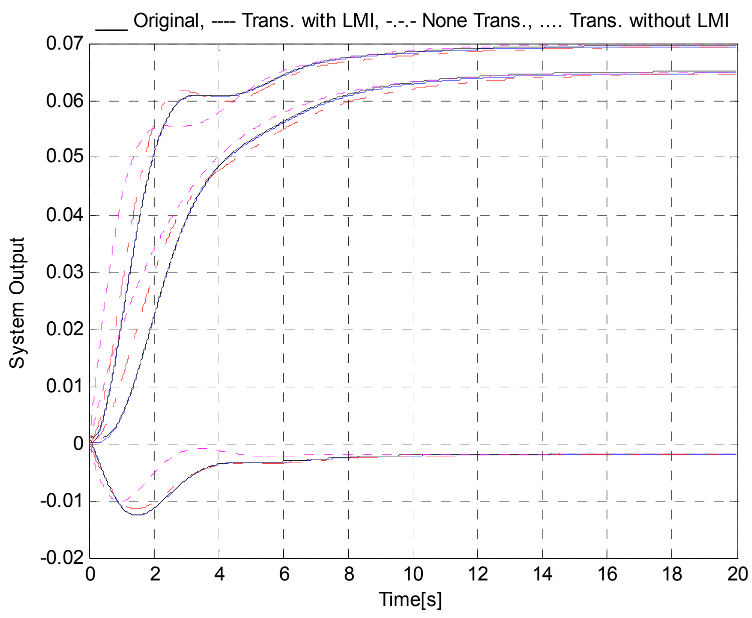


Fig. 10. Reduced 3rd order models (…. transformed without LMI, -.-.-.- non-transformed, ---- transformed with LMI) output responses to a step input along with the non reduced (_____ original) system output response. The LMI-transformed curve fits almost exactly on the original response.

Here, the LMI-reduction-based technique has provided a response that is better than both of the reduced non-transformed and non-LMI-reduced transformed responses and is almost identical to the original system response.

Case #3. Investigating the example of case #3 in subsection 4.1.1, for T_s = 0.1 sec., 200 input/output data points, and $\eta = 1 \times 10^{-4}$ with initial weights for $[\tilde{\mathbf{A}}_{\mathbf{d}}]$ given as:

$$w = \begin{bmatrix} 0.0048 & 0.0039 & 0.0009 & 0.0089 & 0.0168 \\ 0.0072 & 0.0024 & 0.0048 & 0.0017 & 0.0040 \\ 0.0176 & 0.0176 & 0.0136 & 0.0175 & 0.0034 \\ 0.0055 & 0.0039 & 0.0078 & 0.0076 & 0.0051 \\ 0.0102 & 0.0024 & 0.0091 & 0.0049 & 0.0121 \end{bmatrix}$$

the LMI-based transformation and then order reduction were performed. Simulation results of the reduced order models and the original system are shown in Figure 11.

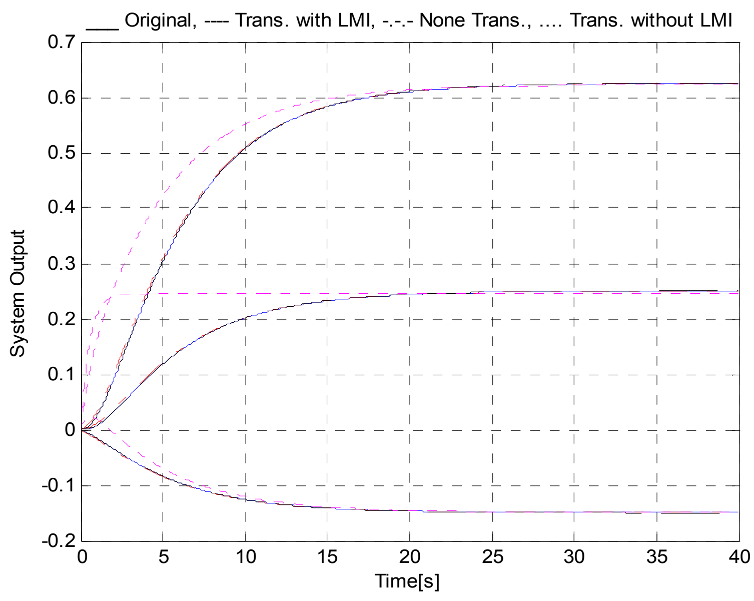


Fig. 11. Reduced 3rd order models (…. transformed without LMI, -.-.-.- non-transformed, ---- transformed with LMI) output responses to a step input along with the non reduced (____ original) system output response. The LMI-transformed curve fits almost exactly on the original response.

Again, the response of the reduced order model using the complete LMI-based transformation is the best as compared to the other reduction techniques.

5. The application of closed-loop feedback control on the reduced models

Utilizing the LMI-based reduced system models that were presented in the previous section, various control techniques – that can be utilized for the robust control of dynamic systems - are considered in this section to achieve the desired system performance. These control methods include (a) PID control, (b) state feedback control using (1) pole placement for the desired eigenvalue locations and (2) linear quadratic regulator (LQR) optimal control, and (c) output feedback control.

5.1 Proportional–Integral–Derivative (PID) control

A PID controller is a generic control loop feedback mechanism which is widely used in industrial control systems [7,10,24]. It attempts to correct the error between a measured process variable (output) and a desired set-point (input) by calculating and then providing a corrective signal that can adjust the process accordingly as shown in Figure 12.

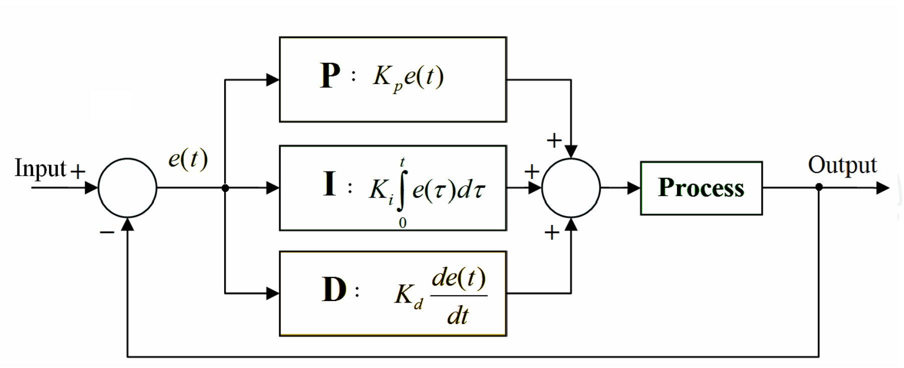


Fig. 12. Closed-loop feedback single-input single-output (SISO) control using a PID controller.

In the control design process, the three parameters of the PID controller $\{K_p, K_i, K_d\}$ have to be calculated for some specific process requirements such as system overshoot and settling time. It is normal that once they are calculated and implemented, the response of the system is not actually as desired. Therefore, further tuning of these parameters is needed to provide the desired control action.

Focusing on one output of the tape-drive machine, the PID controller using the reduced order model for the desired output was investigated. Hence, the identified reduced 3rd order model is now considered for the output of the tape position at the head which is given as:

$$G(s)_{\text{original}} = \frac{0.0801s + 0.133}{s^3 + 2.1742s^2 + 2.2837s + 1.0919}$$

Searching for suitable values of the PID controller parameters, such that the system provides a faster response settling time and less overshoot, it is found that $\{K_p = 100, K_i = 80, K_d = 90\}$ with a controlled system which is given by:

$$G(s)_{\text{controlled}} = \frac{7.209s^3 + 19.98s^2 + 19.71s + 10.64}{s^4 + 9.383s^3 + 22.26s^2 + 20.8s + 10.64}$$

Simulating the new PID-controlled system for a step input provided the results shown in Figure 13, where the settling time is almost 1.5 sec. while without the controller was greater than 6 sec. Also as observed, the overshoot has much decreased after using the PID controller.

On the other hand, the other system outputs can be PID-controlled using the cascading of current process PID and new tuning-based PIDs for each output. For the PID-controlled output of the tachometer shaft angle, the controlling scheme would be as shown in Figure 14. As seen in Figure 14, the output of interest (i.e., the 2nd output) is controlled as desired using the PID controller. However, this will affect the other outputs' performance and therefore a further PID-based tuning operation must be applied.

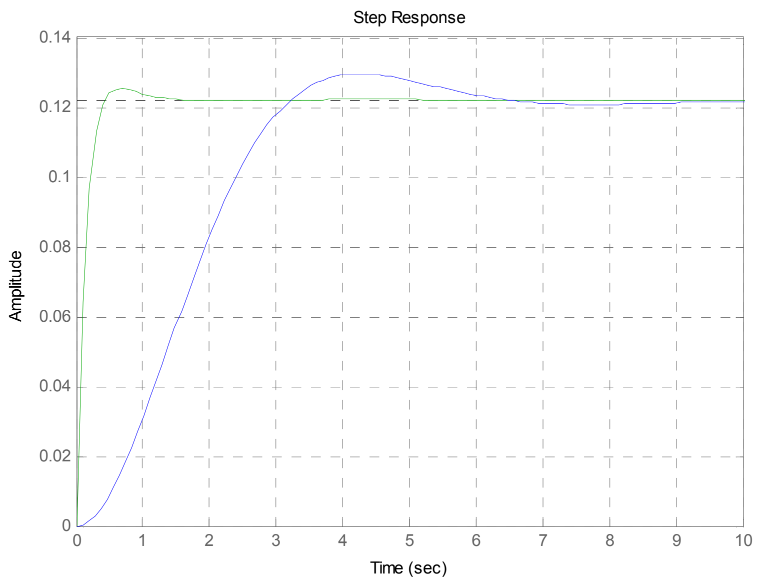


Fig. 13. Reduced 3rd order model PID controlled and uncontrolled step responses.

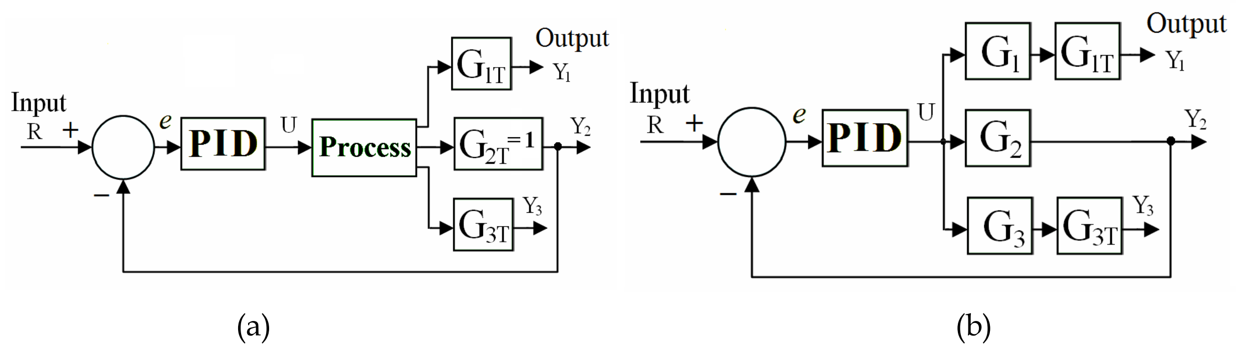


Fig. 14. Closed-loop feedback single-input multiple-output (SIMO) system with a PID controller: (a) a generic SIMO diagram, and (b) a detailed SIMO diagram.

As shown in Figure 14, the tuning process is accomplished using G_{1T} and G_{3T}. For example, for the 1st output:

$$Y_1 = G_{1T}G_1\text{PID}(R - Y_2) = Y_1 = G_1 R \tag{39}$$

$$\therefore\ G_{1T} = \frac{R}{\text{PID}(R - Y_2)} \tag{40}$$

where Y_2 is the Laplace transform of the 2nd output. Similarly, G_{3T} can be obtained.

5.2 State feedback control

In this section, we will investigate the state feedback control techniques of pole placement and the LQR optimal control for the enhancement of the system performance.

5.2.1 Pole placement for the state feedback control

For the reduced order model in the system of Equations (37) - (38), a simple pole placement-based state feedback controller can be designed. For example, assuming that a controller is needed to provide the system with an enhanced system performance by relocating the eigenvalues, the objective can be achieved using the control input given by:

$$u(t) = -K\tilde{x}_r(t) + r(t) \tag{41}$$

where K is the state feedback gain designed based on the desired system eigenvalues. A state feedback control for pole placement can be illustrated by the block diagram shown in Figure 15.

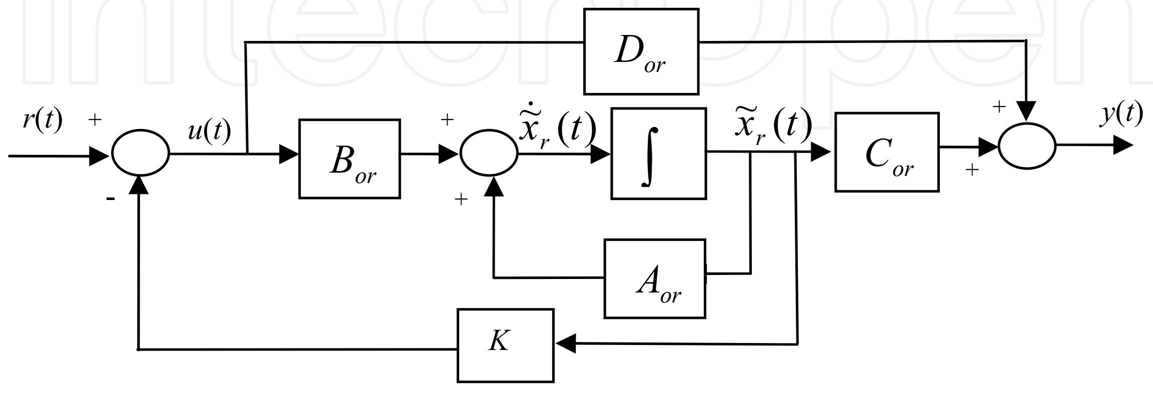


Fig. 15. Block diagram of a state feedback control with {[$\mathbf{A_{or}}$], [$\mathbf{B_{or}}$], [$\mathbf{C_{or}}$], [$\mathbf{D_{or}}$]} overall reduced order system matrices.

Replacing the control input $u(t)$ in Equations (37) - (38) by the above new control input in Equation (41) yields the following reduced system equations:

$$\dot{\tilde{x}}_r(t) = A_{or}\tilde{x}_r(t) + B_{or}[-K\tilde{x}_r(t) + r(t)] \tag{42}$$

$$y(t) = C_{or}\tilde{x}_r(t) + D_{or}[-K\tilde{x}_r(t) + r(t)] \tag{43}$$

which can be re-written as:

$$\dot{\tilde{x}}_r(t) = A_{or}\tilde{x}_r(t) - B_{or}K\tilde{x}_r(t) + B_{or}r(t) \;\rightarrow\; \dot{\tilde{x}}_r(t) = [A_{or} - B_{or}K]\tilde{x}_r(t) + B_{or}r(t)$$

$$y(t) = C_{or}\tilde{x}_r(t) - D_{or}K\tilde{x}_r(t) + D_{or}r(t) \;\rightarrow\; y(t) = [C_{or} - D_{or}K]\tilde{x}_r(t) + D_{or}r(t)$$

where this is illustrated in Figure 16.

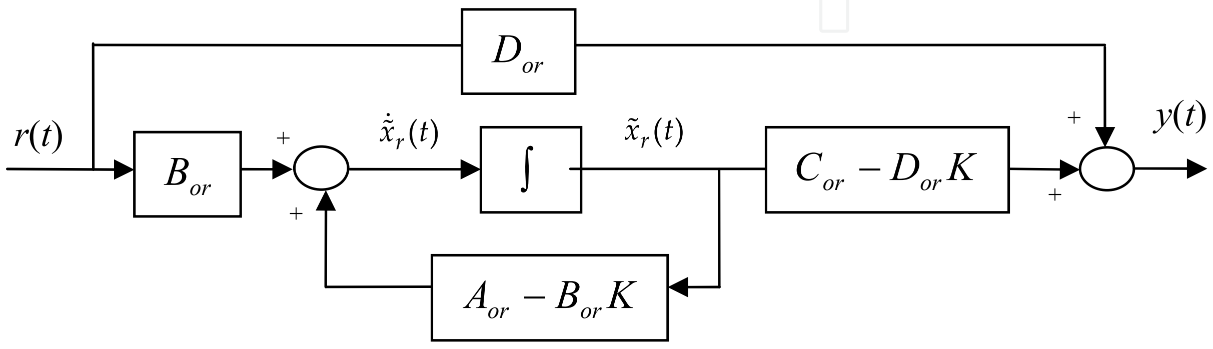


Fig. 16. Block diagram of the overall state feedback control for pole placement.

The overall closed-loop system model may then be written as:

$$\dot{\tilde{x}}(t) = A_{cl}\tilde{x}_r(t) + B_{cl}r(t) \tag{44}$$

$$y(t) = C_{cl}\tilde{x}_r(t) + D_{cl}r(t) \tag{45}$$

such that the closed loop system matrix [$\mathbf{A_{cl}}$] will provide the new desired system eigenvalues.
For example, for the system of case #3, the state feedback was used to re-assign the eigenvalues with {-1.89, -1.5, -1}. The state feedback control was then found to be of K = [-1.2098 0.3507 0.0184], which placed the eigenvalues as desired and enhanced the system performance as shown in Figure 17.

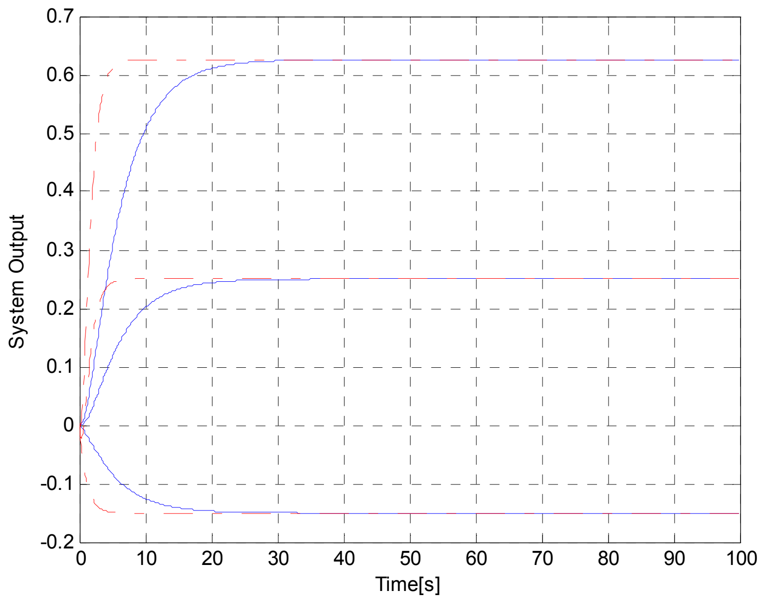


Fig. 17. Reduced 3rd order state feedback control (for pole placement) output step response -.-.-.- compared with the original ____ full order system output step response.

5.2.2 Linear-Quadratic Regulator (LQR) optimal control for the state feedback control

Another method for designing a state feedback control for system performance enhancement may be achieved based on minimizing the cost function given by [10]:

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{46}$$

which is defined for the system $\dot{x}(t) = Ax(t) + Bu(t)$, where Q and R are weight matrices for the states and input commands. This is known as the LQR problem, which has received much of a special attention due to the fact that it can be solved analytically and that the resulting optimal controller is expressed in an easy-to-implement state feedback control [7,10]. The feedback control law that minimizes the values of the cost is given by:

$$u(t) = -Kx(t) \tag{47}$$

where K is the solution of $K = R^{-1}B^{T}q$ and **[q]** is found by solving the algebraic Riccati equation which is described by:

$$A^{T}q + qA - qBR^{-1}B^{T}q + Q = 0 \tag{48}$$

where **[Q]** is the state weighting matrix and **[R]** is the input weighting matrix. A direct solution for the optimal control gain maybe obtained using the MATLAB statement $K = \text{lqr}(A, B, Q, R)$, where in our example R = 1, and the **[Q]** matrix was found using the output **[C]** matrix such as $Q = C^{T}C$.

The LQR optimization technique is applied to the reduced 3rd order model in case #3 of subsection 4.1.2 for the system behavior enhancement. The state feedback optimal control gain was found K = [-0.0967 -0.0192 0.0027], which when simulating the complete system for a step input, provided the normalized output response (with a normalization factor γ = 1.934) as shown in Figure 18.

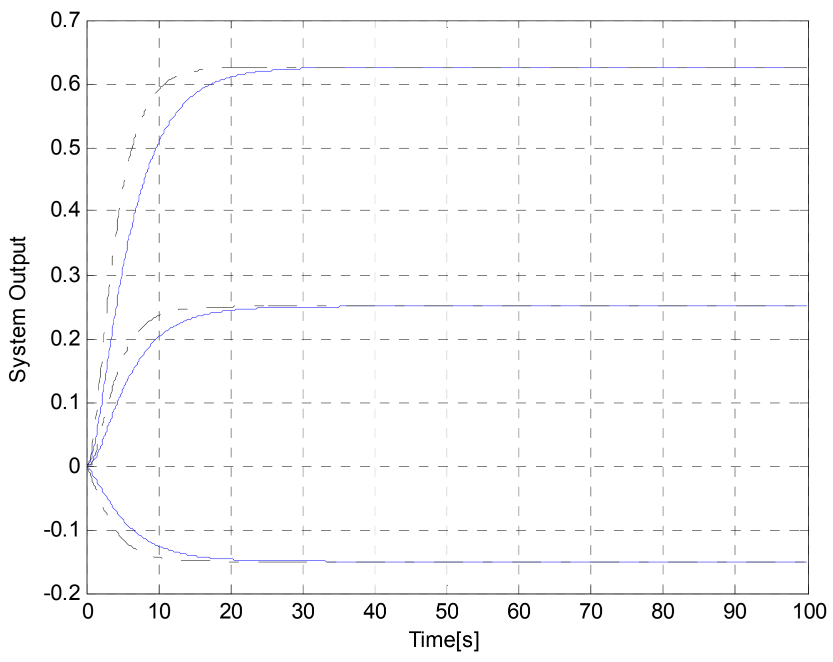


Fig. 18. Reduced 3rd order LQR state feedback control output step response -.-.-.- compared with the original ____ full order system output step response.

As seen in Figure 18, the optimal state feedback control has enhanced the system performance, which is basically based on selecting new proper locations for the system eigenvalues.

5.3 Output feedback control

The output feedback control is another way of controlling the system for certain desired system performance as shown in Figure 19 where the feedback is directly taken from the output.

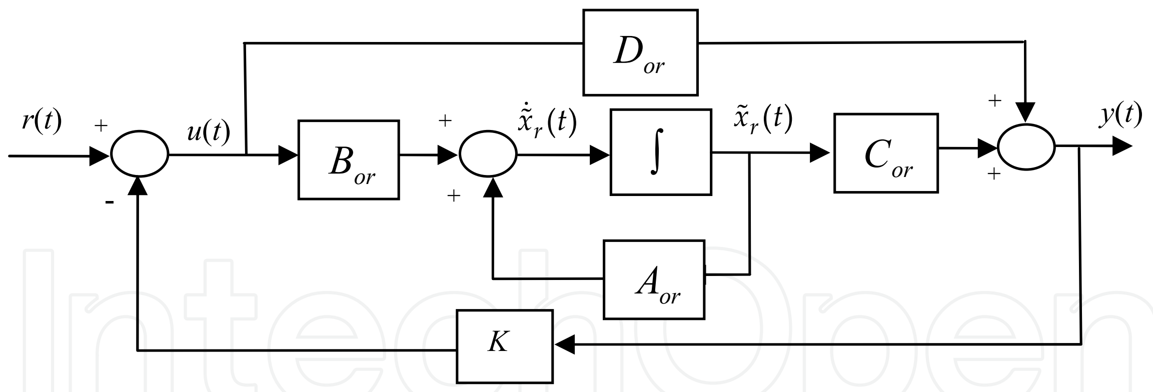


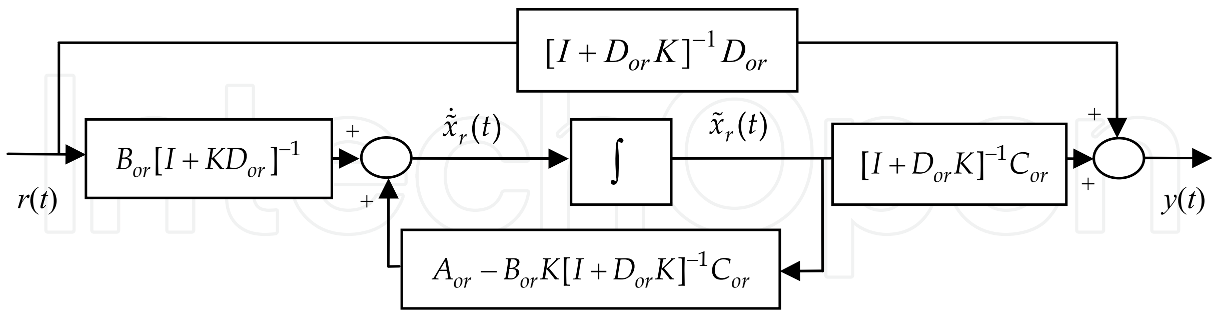
Fig. 19. Block diagram of an output feedback control.

The control input is now given by $u(t) = -Ky(t) + r(t)$, where $y(t) = C_{or}\tilde{x}_r(t) + D_{or}u(t)$. By applying this control to the considered system, the system equations become [7]:

$$\begin{aligned}\dot{\tilde{x}}_r(t) &= A_{or}\tilde{x}_r(t) + B_{or}[-K(C_{or}\tilde{x}_r(t) + D_{or}u(t)) + r(t)] \\ &= A_{or}\tilde{x}_r(t) - B_{or}KC_{or}\tilde{x}_r(t) - B_{or}KD_{or}u(t) + B_{or}r(t) \\ &= [A_{or} - B_{or}KC_{or}]\tilde{x}_r(t) - B_{or}KD_{or}u(t) + B_{or}r(t) \\ &= [A_{or} - B_{or}K[I + D_{or}K]^{-1}C_{or}]\tilde{x}_r(t) + [B_{or}[I + KD_{or}]^{-1}]r(t)\end{aligned} \tag{49}$$

$$\begin{aligned}y(t) &= C_{or}\tilde{x}_r(t) + D_{or}[-Ky(t) + r(t)] \\ &= C_{or}\tilde{x}_r(t) - D_{or}Ky(t) + D_{or}r(t) \\ &= [[I + D_{or}K]^{-1}C_{or}]\tilde{x}_r(t) + [[I + D_{or}K]^{-1}D_{or}]r(t)\end{aligned} \tag{50}$$

This leads to the overall block diagram as seen in Figure 20.



$[I + D_{or}K]^{-1}D_{or}$
$B_{or}[I + KD_{or}]^{-1}$
$r(t)$
+
+
$\dot{\tilde{x}}_r(t)$
$\int$
$\tilde{x}_r(t)$
$[I + D_{or}K]^{-1}C_{or}$
+
+
$y(t)$
$A_{or} - B_{or}K[I + D_{or}K]^{-1}C_{or}$

Fig. 20. An overall block diagram of an output feedback control.

Considering the reduced 3rd order model in case #3 of subsection 4.1.2 for system behavior enhancement using the output feedback control, the feedback control gain is found to be K = [0.5799 -2.6276 -11]. The normalized controlled system step response is shown in Figure 21, where one can observe that the system behavior is enhanced as desired.

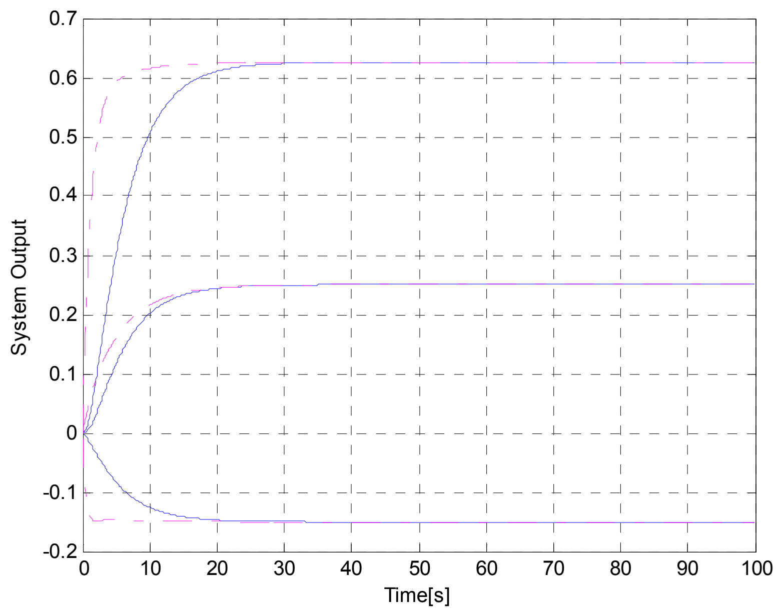


Fig. 21. Reduced 3rd order output feedback controlled step response -.-.-.- compared with the original ____ full order system uncontrolled output step response.

6. Conclusions and future work

In control engineering, robust control is an area that explicitly deals with uncertainty in its approach to the design of the system controller. The methods of robust control are designed to operate properly as long as disturbances or uncertain parameters are within a compact set, where robust methods aim to accomplish robust performance and/or stability in the presence of bounded modeling errors. A robust control policy is static - in contrast to the adaptive (dynamic) control policy - where, rather than adapting to measurements of variations, the system controller is designed to function assuming that certain variables will be unknown but, for example, bounded.

This research introduces a new method of hierarchical intelligent robust control for dynamic systems. In order to implement this control method, the order of the dynamic system was reduced. This reduction was performed by the implementation of a recurrent supervised neural network to identify certain elements $[\mathbf{A}_c]$ of the transformed system matrix $[\tilde{\mathbf{A}}]$, while the other elements $[\mathbf{A}_r]$ and $[\mathbf{A}_o]$ are set based on the system eigenvalues such that $[\mathbf{A}_r]$ contains the dominant eigenvalues (i.e., slow dynamics) and $[\mathbf{A}_o]$ contains the non-dominant eigenvalues (i.e., fast dynamics). To obtain the transformed matrix $[\tilde{\mathbf{A}}]$, the zero input response was used in order to obtain output data related to the state dynamics, based only on the system matrix $[\mathbf{A}]$. After the transformed system matrix was obtained, the optimization algorithm of linear matrix inequality was utilized to determine the permutation matrix $[\mathbf{P}]$, which is required to complete the system transformation matrices $\{[\tilde{\mathbf{B}}], [\tilde{\mathbf{C}}], [\tilde{\mathbf{D}}]\}$. The reduction process was then applied using the singular perturbation method, which operates on neglecting the faster-dynamics eigenvalues and leaving the dominant slow-dynamics eigenvalues to control the system. The comparison simulation results show clearly that modeling and control of the dynamic system using LMI is superior

to that without using LMI. Simple feedback control methods using PID control, state feedback control utilizing (a) pole assignment and (b) LQR optimal control, and output feedback control were then implemented to the reduced model to obtain the desired enhanced response of the full order system.
Future work will involve the application of new control techniques, utilizing the control hierarchy introduced in this research, such as using fuzzy logic and genetic algorithms. Future work will also involve the fundamental investigation of achieving model order reduction for dynamic systems with all eigenvalues being complex.

7. References

[1] A. N. Al-Rabadi, "Artificial Neural Identification and LMI Transformation for Model Reduction-Based Control of the Buck Switch-Mode Regulator," *American Institute of Physics* (*AIP*), In: *IAENG Transactions on Engineering Technologies*, *Special Edition of the International MultiConference of Engineers and Computer Scientists* 2009, AIP Conference Proceedings 1174, Editors: Sio-Iong Ao, Alan Hoi-Shou Chan, Hideki Katagiri and Li Xu, Vol. 3, pp. 202 – 216, New York, U.S.A., 2009.

[2] A. N. Al-Rabadi, "Intelligent Control of Singularly-Perturbed Reduced Order Eigenvalue-Preserved Quantum Computing Systems via Artificial Neural Identification and Linear Matrix Inequality Transformation," *IAENG Int. Journal of Computer Science* (*IJCS*), Vol. 37, No. 3, 2010.

[3] P. Avitabile, J. C. O'Callahan, and J. Milani, "Comparison of System Characteristics Using Various Model Reduction Techniques," 7th *International Model Analysis Conference*, Las Vegas, Nevada, February 1989.

[4] P. Benner, "Model Reduction at ICIAM'07," *SIAM News*, Vol. 40, No. 8, 2007.

[5] A. Bilbao-Guillerna, M. De La Sen, S. Alonso-Quesada, and A. Ibeas, "Artificial Intelligence Tools for Discrete Multiestimation Adaptive Control Scheme with Model Reduction Issues," *Proc. of the International Association of Science and Technology*, *Artificial Intelligence and Application*, Innsbruck, Austria, 2004.

[6] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Society for Industrial and Applied Mathematics (SIAM), 1994.

[7] W. L. Brogan, *Modern Control Theory*, 3rd Edition, Prentice Hall, 1991.

[8] T. Bui-Thanh, and K. Willcox, "Model Reduction for Large-Scale CFD Applications Using the Balanced Proper Orthogonal Decomposition," 17th *American Institute of Aeronautics and Astronautics* (*AIAA*) *Computational Fluid Dynamics Conf.*, Toronto, Canada, June 2005.

[9] J. H. Chow, and P. V. Kokotovic, "A Decomposition of Near-Optimal Regulators for Systems with Slow and Fast Modes," *IEEE Trans. Automatic Control*, AC-21, pp. 701-705, 1976.

[10] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 3rd Edition, Addison-Wesley, 1994.

[11] K. Gallivan, A. Vandendorpe, and P. Van Dooren, "Model Reduction of MIMO System via Tangential Interpolation," *SIAM Journal of Matrix Analysis and Applications*, Vol. 26, No. 2, pp. 328-349, 2004.

[12] K. Gallivan, A. Vandendorpe, and P. Van Dooren, "Sylvester Equation and Projection-Based Model Reduction," *Journal of Computational and Applied Mathematics*, 162, pp. 213-229, 2004.

[13] G. Garsia, J. Dfouz, and J. Benussou, "H_2 Guaranteed Cost Control for Singularly Perturbed Uncertain Systems," *IEEE Trans. Automatic Control*, Vol. 43, pp. 1323-1329, 1998.

[14] R. J. Guyan, "Reduction of Stiffness and Mass Matrices," *AIAA Journal*, Vol. 6, No. 7, pp. 1313-1319, 1968.

[15] S. Haykin, *Neural Networks*: *A Comprehensive Foundation*, Macmillan Publishing Company, New York, 1994.

[16] W. H. Hayt, J. E. Kemmerly, and S. M. Durbin, *Engineering Circuit Analysis,* McGraw-Hill, 2007.

[17] G. Hinton, and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, pp. 504-507, 2006.

[18] R. Horn, and C. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985.

[19] S. H. Javid, "Observing the Slow States of a Singularly Perturbed Systems," *IEEE Trans. Automatic Control,* AC-25, pp. 277-280, 1980.

[20] H. K. Khalil, "Output Feedback Control of Linear Two-Time-Scale Systems," *IEEE Trans. Automatic Control,* AC-32, pp. 784-792, 1987.

[21] H. K. Khalil, and P. V. Kokotovic, "Control Strategies for Decision Makers Using Different Models of the Same System," *IEEE Trans. Automatic Control,* AC-23, pp. 289-297, 1978.

[22] P. Kokotovic, R. O'Malley, and P. Sannuti, "Singular Perturbation and Order Reduction in Control Theory – An Overview," *Automatica*, 12(2), pp. 123-132, 1976.

[23] C. Meyer, *Matrix Analysis and Applied Linear Algebra*, Society for Industrial and Applied Mathematics (SIAM), 2000.

[24] K. Ogata, *Discrete-Time Control Systems*, 2nd Edition, Prentice Hall, 1995.

[25] R. Skelton, M. Oliveira, and J. Han, *Systems Modeling and Model Reduction*, Invited Chapter of the Handbook of Smart Systems and Materials, Institute of Physics, 2004.

[26] M. Steinbuch, "Model Reduction for Linear Systems," 1st *International MACSI-net Workshop on Model Reduction,* Netherlands, October 2001.

[27] A. N. Tikhonov, "On the Dependence of the Solution of Differential Equation on a Small Parameter," *Mat Sbornik* (*Moscow*), 22(64):2, pp. 193-204, 1948.

[28] R. J. Williams, and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation,* 1(2), pp. 270-280, 1989.

[29] J. M. Zurada, *Artificial Neural Systems,* West Publishing Company, New York, 1992.