Transformation Using Neural-Based Identification for Controlling Singularly-Perturbed Eigenvalue-Preserved Reduced Order Systems

Anas N. Al-Rabadi and Othman M.K. Alsmadi

Abstract **- This paper introduces a new hierarchy for controlling dynamical systems. The new control hierarchy uses supervised neural network to identify certain parameters of the transformed system matrix [$\tilde{A}$]. Then, Linear Matrix Inequality (LMI) is used to determine the permutation matrix [P] so that a complete system transformation {[$\tilde{B}$], [$\tilde{C}$], [$\tilde{D}$]} is performed. The transformed model is then reduced using singular perturbation method, and various feedback control schemes are applied to enhance system performance, including PID control, state feedback control using pole assignment, state feedback control using LQR optimal control, and output feedback control. The comparative experimental results between system transformation without using LMI and state transformation via using LMI shows clearly the superiority in system modeling and control using the proposed LMI-based control method. The new control methodology simplifies the system model and thus uses simpler controllers to produce the desired response.**

Index Terms **- Linear Matrix Inequality (LMI), LQR Optimal Control, Neural Networks, Order Model Reduction, Output Feedback Control, PID Control, Parameter Estimation, Pole Placement, Singular Perturbation Methods, State Feedback Control, System Identification.**

1. INTRODUCTION

The general objective of control systems is to cause the output variable of a dynamic process to follow the desired reference variable accurately. This goal can be achieved based on a number of steps. An important one is to develop a mathematical model to be controlled [4,6]. This model is usually done using a set of differential equations that describe the system dynamic behavior.

In system modeling, sometimes it is required to identify certain parameters. This objective maybe achieved by the use of artificial neural networks (ANN) [8,9,15,16]. Artificial neural systems maybe defined as physical cellular systems which have the capability of acquiring, storing and utilizing experiential knowledge [8,16]. It consists of an interconnected group of artificial neurons and processes information using a computational connectionist approach. The basic processing elements of neural networks are called neurons. They perform summing operations and nonlinear functions. Neurons are usually organized in layers and forward connections and computations are performed in a parallel fashion at all nodes and connections.

A. N. Al-Rabadi is with the Computer Engineering Department, The University of Jordan, Amman, Jordan (Phone: + 962-79-644-5364; E-mail: alrabadi@yahoo.com; URL: http://web.pdx.edu/~psu21829/)

O. M.K. Alsmadi is with the Electrical Engineering Department, The University of Jordan, Amman, Jordan (E-mail: othman_mk@yahoo.com)

Each ANN connection is expressed by a numerical value called a weight. The learning process of a neuron corresponds to a way of changing its weights. In fact, they can be used to model complex relationships between inputs and outputs or to find patterns in data [2,8,9,15,16].

When dealing with system modeling and control analysis, some equations and inequalities require optimized solutions. A numerical algorithm called Linear Matrix Inequality (LMI) serves as an optimization technique [1,3]. The LMI 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 $A^T P + PA < 0$ [3]. The requirement $P > 0$, $A^T P + PA < 0$ is what is known as 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 [3].

In practical control design problems, the first step is to obtain a mathematical model in order to examine the system behavior for the purpose of designing a proper controller [1,2,3,4,5,7,9,11,12,13,14]. Sometimes, this mathematical description involves a certain small parameter (perturbation). Neglecting this small parameter results in reducing the order model of the system and thus simplifying the controller design [1,2,5,7,9,11,12,13,14]. A reduced order 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 model reduction leads to controller cost minimization [4,6,7]. In control system, due to the fact that feedback controllers do not usually consider all the dynamics of the system, model reduction is a very important issue [2,9]. One of the model reduction methods is the singular perturbation [5,7,11,14].

Figure 1 illustrates the new control hierarchical methodology introduced in this paper. Section 2 presents background on recurrent supervised NN, LMI, system model transformation using neural identification, and order model reduction. Section 3 presents a detailed illustration of the NN identification with the LMI optimization methods for system order model reduction. A practical implementation of the NN identification and the associated comparative results with and without the use of LMI to the dynamical system order model reduction is presented in Section 4. Section 5 presents the application of the feedback control on the reduced order model using PID control, state feedback control using pole placement, state feedback control using LQR optimal control, and output feedback control. Conclusions are presented in Section 6.

ISBN: 978-988-17012-7-5

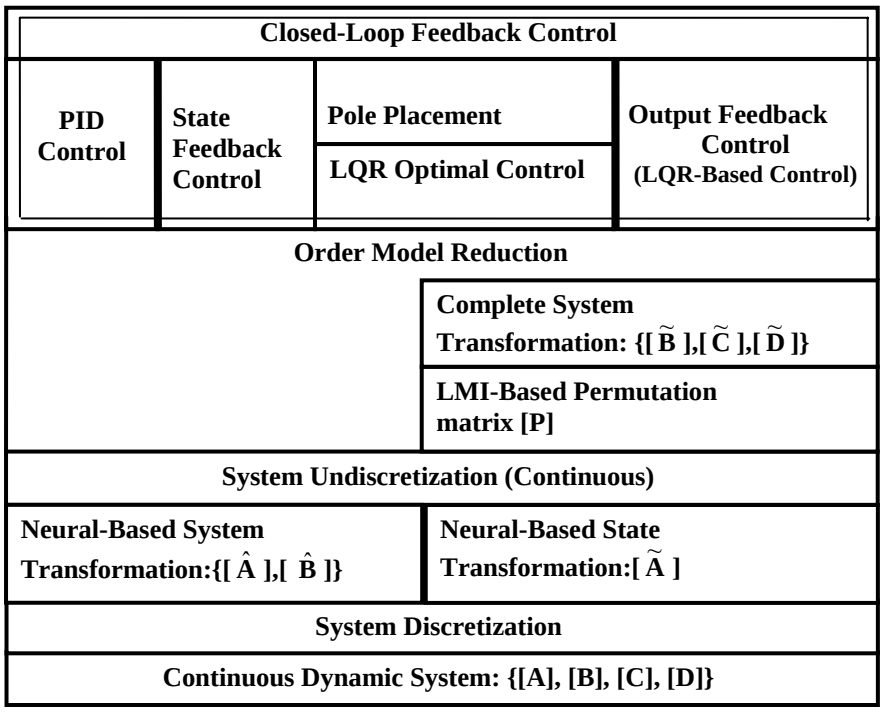


Figure 1. The new hierarchical control methodology used in this paper.

2. FUNDAMENTALS

The following sections provide an important background which will be used Sections 3-5.

2.1 Recurrent Supervised Neural Network

An artificial NN is an emulation of a biological neural system. The process of a neuron may be mathematically modeled as shown in Figure 2 [8,16].

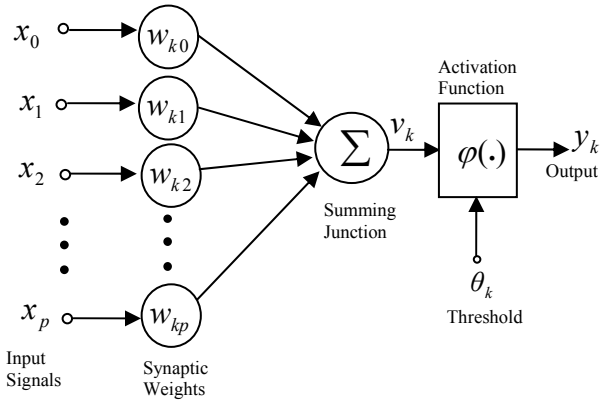


Figure 2. Mathematical model of an artificial neuron.

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

$$v_k = \sum_{j=1}^{p} w_{kj} x_j \qquad (1)$$

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 [8,16]. The difference between the actual and the desired response represents an error measure and 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 is required for this learning mode. The training algorithm estimates the negative error gradient directions and reduces the resulting error [8,16].

The supervised NN used for identification in this paper is based on an approximation of the method of steepest descent [8,15,16]. The network tries to match the output of certain neurons to the desired values of the system output at 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 $\mathbf{g}(k)$ denotes the (M x 1) external input vector 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). 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 $u_i(k)$. If 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 is true:

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

The (N x ($M + N$)) recurrent weight matrix of the network is represented by the variable [**W**]. The net internal activity of

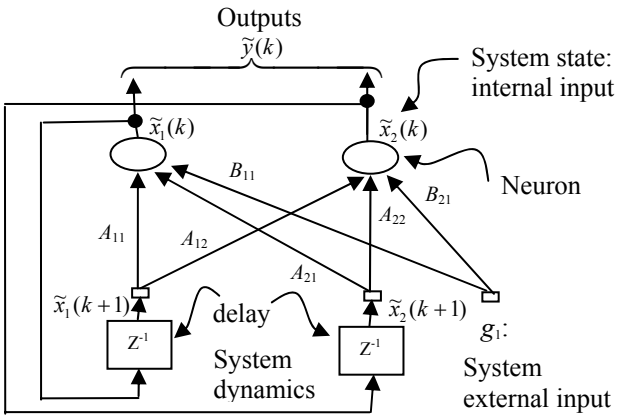


Figure 3. A second order recurrent neural network architecture, where the estimated matrices are given by: $\tilde{A}_d = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $\tilde{B}_d = \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix}$ and that $W = \left[\left[\tilde{\mathbf{A}}_\mathbf{d}\right] \ \left[\tilde{\mathbf{B}}_\mathbf{d}\right]\right]$.

neuron j at time k is given by:

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

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

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

To accomplish this objective, the steepest descent 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 time k is defined as:

$$\Delta w_{m\ell}(k) = -\eta \frac{\partial E(k)}{\partial w_{m\ell}(k)}$$

where η is the learning-rate parameter. Hence:

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

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

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

where $\dot{\varphi}(v_j(k)) = \dfrac{\partial \varphi(v_j(k))}{\partial v_j(k)}$.

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 $i = \ell$; otherwise, it is "0". 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 equal 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:

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

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

$$\pi_{m\ell}^j(k) = \frac{\partial y_j(k)}{\partial w_{m\ell}(k)}$$

For every time step k and all appropriate j, m and ℓ, and $\pi_{m\ell}^j(0) = 0$, system dynamics are controlled by:

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

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

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

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

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

Repeating this computation procedure provides the objective of minimizing the cost function.

With the many advantages that the NN has, it is used for parameter identification in model transformation for the purpose of order model reduction.

2.2 LMI and Model Transformation

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

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

$$y(t) = Cx(t) + Du(t) \quad (5)$$

In order to determine the transformed [**A**] matrix, which is [$\widetilde{\mathbf{A}}$], 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) and (5), with $x(0) = x_0$, becomes:

$$x(k+1) = A_d x(k) \quad (6)$$

$$y(k) = x(k) \quad (7)$$

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

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

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 $\widetilde{A}_{ij}$ in the upper triangle. This upper triangular matrix form is used to produce same eigenvalues for the purpose of eliminating the fast dynamics and preserving the slow dynamics eigenvalues through order model reduction.

Having the [**A**] and [$\widetilde{\mathbf{A}}$] 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: assuming that $\widetilde{x} = P^{-1}x$, the system of Equations (4) and (5) can be re-written as:

$$P\dot{\widetilde{x}}(t) = AP\widetilde{x}(t) + Bu(t),\ \widetilde{y}(t) = CP\widetilde{x}(t) + Du(t),$$

where $\widetilde{y}(t) = y(t)$. Pre-multiplying the first equation above by [$\mathbf{P^{-1}}$], we get:

$$P^{-1}P\dot{\widetilde{x}}(t) = P^{-1}AP\widetilde{x}(t) + P^{-1}Bu(t)$$

$$\widetilde{y}(t) = CP\widetilde{x}(t) + Du(t)$$

which yields the following transformed model:

$$\dot{\widetilde{x}}(t) = \widetilde{A}\widetilde{x}(t) + \widetilde{B}u(t) \quad (9)$$

$$\widetilde{y}(t) = \widetilde{C}\widetilde{x}(t) + \widetilde{D}u(t) \quad (10)$$

where the transformed system matrices are given by:

$$\widetilde{A} = P^{-1}AP \quad (11)$$

$$\widetilde{B} = P^{-1}B \quad (12)$$

$$\widetilde{C} = CP \quad (13)$$

$$\widetilde{D} = D \quad (14)$$

Transforming the system matrix [**A**] into the form shown in Equation (8) can be done based on the following definition [10].

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:

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

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 orthogonal and invertible have made this transformation easy to carry out. Some form of a similarity transformation maybe used as follows $f : R^{n\times n} \rightarrow R^{n\times n}$, where f is a linear operator defined by $f(A) = P^{-1}AP$ [10]. Hence, based on the [**A**] and [$\widetilde{\mathbf{A}}$], LMI is used to obtain the transformation matrix [**P**]. The optimization problem is performed as follows:

$$\min_P \left\| P - P_o \right\| \quad Subject\ to \quad \left\| P^{-1}AP - \widetilde{A} \right\| < \varepsilon \quad (16)$$

which maybe written in an LMI equivalent form as:

$$\min_S trace(S) \quad Subject\ to$$

$$\begin{bmatrix} S & P - P_o \\ (P - P_o)^T & I \end{bmatrix} > 0 \quad (17)$$

$$\begin{bmatrix} \varepsilon_1^2 I & P^{-1}AP - \widetilde{A} \\ (P^{-1}AP - \widetilde{A})^T & I \end{bmatrix} > 0$$

where S is a symmetric slack matrix [3].

2.3 System Transformation via Neural Identification

A different type of transformation can be obtained using NN with the condition of preserving the eigenvalues as 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, in the new method when using NN, finding the permutation matrix [**P**] does not have to be performed, instead, [**X**] and [**Z**] in

Equation (15) will be selected as the eigenvalues and [**Y**] in Equation (15) is to be estimated directly using NN identification. Thus, the transformation is obtained and the reduction is then achieved. Therefore, another way to obtain a transformed model preserving the eigenvalues of a reduced order model as a subset of the original system is by using NN training without LMI. This may be achieved by assuming that the states are measurable. Hence, the NN can estimate $[\hat{\mathbf{A}}_\mathbf{d}]$ and $[\hat{\mathbf{B}}_\mathbf{d}]$ for a given input signal as illustrated in Figure 3. The NN identification would lead to the following $[\hat{\mathbf{A}}_\mathbf{d}]$ and $[\hat{\mathbf{B}}_\mathbf{d}]$ transformations which (in the case of all real eigenvalues) construct the weight matrix [**W**] as follows:

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

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

2.4 Order Model Reduction

Linear Time-Invariant (LTI) models of many physical systems have fast and slow dynamics, which may be referred to as singularly perturbed systems [11]. Neglecting the fast dynamics of a singularly perturbed system provides a reduced (slow) model. This gives the advantage of designing simpler lower-dimensionality reduced order controllers based on the reduced model.

To show the formulation of a reduced order model, consider the singularly perturbed system [5]:

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

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

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

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 $\varepsilon = 0$ [1,7,13]. In doing so, we neglect the fast system dynamics and assume that the state variables ξ have reached their quasi-steady state. Hence, setting $\varepsilon = 0$ in Equation (19), with the assumption that $[\mathbf{A_{22}}]$ is nonsingular, produces:

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

where the index *r* denotes the reduced model. Substituting Equation (21) in Equations (18)-(20) gives the reduced model

$$\dot{x}_r(t) = A_rx_r(t) + B_ru(t) \quad (22)$$

$$y(t) = C_rx_r(t) + D_ru(t) \quad (23)$$

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

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 NN to estimate the transformed discrete system matrix $[\tilde{\mathbf{A}}_\mathbf{d}]$ is performed [1,8]. For the system of Equations (18)-(20), the discrete model of the system is given as:

$$x(k+1) = A_dx(k) + B_du(k) \quad (24)$$

$$y(k) = C_dx(k) + D_du(k) \quad (25)$$

The estimated discrete model can be written as:

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

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

where *k* is the time index. The detailed matrix elements of Equations (26)-(27) are shown in Figure 3.

The NN 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, and by defining β as the set of indices *i* for which $y_i(k)$ is an internal input or a neuron output. Also, by defining $u_i(k)$ as the combination of the internal and external inputs for which $i \in \beta \cup \Lambda$. Training the network depends on the internal activity of each neuron which is:

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

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

$$x_j(k+1) = \varphi(v_j(k)) \quad (29)$$

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

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

The objective is to minimize the cost function given by:

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

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 [**W**] and then updating [**W**] in the negative direction of this gradient [8,16]. In steps, this may be proceeded as follows:

- Initialize the weights, [**W**], by a set of uniformly distributed random numbers. Starting at the instant *k* = 0, use Equations (28) and (29) to compute the output values of the *N* neurons (where $N = \beta$).

- For every time step k and all $j \in \beta$, $m \in \beta$ and $\ell \in \beta \cup \Lambda$, compute the dynamics of the system which are governed by the triply indexed set:

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

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 $j = m$ and $i = \ell$; 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:

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

- Update the weights in accordance with:

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

- Repeat the computation until the desired estimation is achieved.

As illustrated in Equations (6) and (7), for the purpose of estimating only the transformed $[\tilde{\mathbf{A}}_\mathbf{d}]$, the training is based on the zero input response. Once the training is complete, the obtained weight matrix [**W**] is the discrete estimated transformed system matrix $[\tilde{\mathbf{A}}_\mathbf{d}]$. Transforming the estimated system back to the continuous form yields the desired continuous transformed system matrix $[\tilde{\mathbf{A}}]$. Using LMI, the permutation matrix [**P**] is determined. Hence, a complete system transformation, as shown in Equations (9) and (10) is achieved.

For order model reduction, the system in Equations (9) and (10) will be written as:

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

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

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. Hence,

$$\dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + A_c\tilde{x}_o(t) + B_r u(t),$$

$$\dot{\tilde{x}}_o(t) = A_o\tilde{x}_o(t) + B_o u(t)$$

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 based on the singular perturbation method (by setting $\varepsilon = 0$). Doing so, the following is obtained:

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

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

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

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

Thus, the overall reduced order model is:

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

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

The details of the $\{[\mathbf{A_{or}}], [\mathbf{B_{or}}], [\mathbf{C_{or}}], [\mathbf{D_{or}}]\}$ overall reduced matrices are shown in Equations (35) and (36).

4. Dynamical System Model Reduction Using Neural Identification

This section investigates the proposed method of reduced system modeling using NN with and without LMI.

4.1 Model Reduction Using Neural Identification

Based on the assumption that the states are reachable and measurable, the recurrent NN is utilized to estimate $[\hat{\mathbf{A}}_\mathbf{d}]$ and $[\hat{\mathbf{B}}_\mathbf{d}]$ for a given input as shown in Figure 3. Based on this transformation, a desired reduced model is obtained.

Example 1. Consider the 3rd order system:

$$\dot{x} = \begin{bmatrix}-20 & 5 & -18\\ 8 & -30 & 4\\ 2 & 5 & -40\end{bmatrix}x + \begin{bmatrix}1\\ 1\\ 0.5\end{bmatrix}u$$

$$y = \begin{bmatrix}1 & 0.2 & 1\end{bmatrix}x$$

Since the system is a 3rd order, there are three eigenvalues which are {-25.2822, -22, -42.717}. After proper transformation and training, the following desired diagonal transformed model is obtained:

$$\dot{x} = \begin{bmatrix}-25.28221 & 0.4534 & 12.8339\\ 0 & -22 & 13.4343\\ 0 & 0 & -42.7178\end{bmatrix}x + \begin{bmatrix}0.3343\\ 0.7721\\ 0.8777\end{bmatrix}u$$

$$y = \begin{bmatrix}1.0405 & 0.1856 & 0.8581\end{bmatrix}x + \begin{bmatrix}0.0019\end{bmatrix}u$$

This transformed model was simulated with an input signal that has different functions to capture most of the system dynamics as seen in Figure 4, which presents the system states while training and converging.

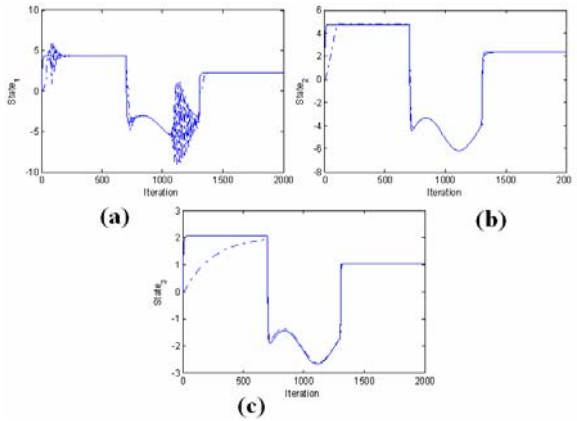


Figure 4. System state response for the three states for a sequence of inputs (1) step, (2) sinusoidal and (3) step (___original state. -.-.-. state while convergence).

It is important to notice that the eigenvalues of the original system are preserved in the transformed model as seen in the above diagonal system matrix. Reducing the 3rd order transformed model to a 2nd order model yields:

$$\dot{x}_r = \begin{bmatrix} -25.2822 & 10.4534 \\ 0 & -22 \end{bmatrix} x_r + \begin{bmatrix} 0.5979 \\ 1.0482 \end{bmatrix} u$$
$$y_r = \begin{bmatrix} 1.0405 & 0.1856 \end{bmatrix} x_r + \begin{bmatrix} 0.0195 \end{bmatrix} u$$

with the dominant eigenvalues preserved as desired. However, by comparing this result to the result of the singular perturbation without transformation which is:

$$\dot{x}_r = \begin{bmatrix} -20.9 & 2.75 \\ 8.2 & -29.5 \end{bmatrix} x_r + \begin{bmatrix} 0.775 \\ 1.05 \end{bmatrix} u$$
$$y_r = \begin{bmatrix} 1.05 & 0.325 \end{bmatrix} x_r + \begin{bmatrix} 0.0125 \end{bmatrix} u$$

it is seen that the eigenvalues {-18.79, -31.60} are totally different from the original system eigenvalues. The three different models were tested for a step input signal and the results are shown in Figure 5.

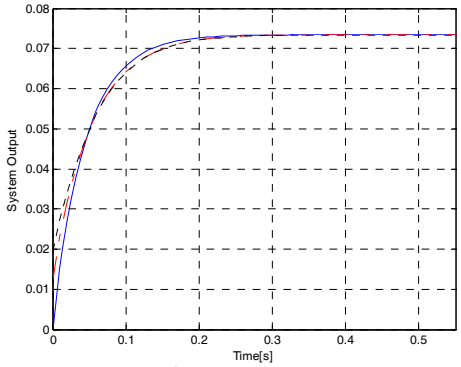


Figure 5. Reduced 2nd order models (…. transformed, -.-.- non-transformed) output responses to a step input along with the non-reduced model (___ original) 3rd order system output response.

It is observed that the transformed reduced model has achieved (1) preserving the original system dominant eigenvalues and (2) performing well as compared with the original system response.

4.2 Order Model Reduction via NN and LMI-Based System Transformation

The following example illustrates the new idea of system order model reduction using LMI with comparison to the order model reduction without using LMI.

Example 2. Consider the system of a high-performance tape transport shown in Figure 6. 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 [6].

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 = tape tension at the read/write head at equilibrium,
F = constant force (tape tension for vacuum column),
K = motor torque constant,
i_o = equilibrium motor current,
r_1 = radius of the capstan take-up wheel.

The variables are defined as deviations from this equilibrium. The system motion equations are:

$$J_1 = \frac{d\omega_1}{dt} + \beta_1 \omega_1 - r_1 T + K_t i \,,\ \dot{x}_1 = r_1 \omega_1$$
$$L\frac{di}{dt} Ri + K_e \omega_1 = e \,,\ \ \dot{x}_2 = r_2 \omega_2$$
$$J_2 \frac{d\omega_2}{dt} + \beta_2 \omega_2 + r_2 T = 0$$
$$T = K_1(x_3 - x_1) + D_1(\dot{x}_3 - \dot{x}_1)$$
$$T = K_2(x_2 - x_3) + D_2(\dot{x}_2 - \dot{x}_3)$$
$$x_1 = r_1\theta_1 \,,\ \ x_2 = r_2\theta_2 \,,\ \ x_3 = \frac{x_1 - x_2}{2}$$

where:
$D_{1,2}$ = damping in the tape-stretch motion,
e = applied input voltage V,
i = current into capstan motor,
J_1 = combined inertia (wheel and take-up) motor,
J_2 = inertia of the idler,
$K_{1,2}$ = spring constant in the tape-stretch motion,
K_e = electric constant of the motor,
K_t = torque constant of the motor,
L = armature inductance,
R = armature resistance,
r_1 = radius of take-up wheel,
r_2 = radius of the tape on the idler,
T = tape tension at the read/write head,
x_3 = position of the tape at the head,
$\dot{x}_3$ = velocity of the tape at the head,
β_1 = viscous friction at take-up wheel,
β_2 = viscous friction at the wheel,
θ_1 = angular displacement of the capstan,
θ_2 = tachometer shaft angle,
ω_1 = speed of the drive wheel $\dot{\theta}_1$,
ω_2 = output speed measured by the tachometer output $\dot{\theta}_2$.

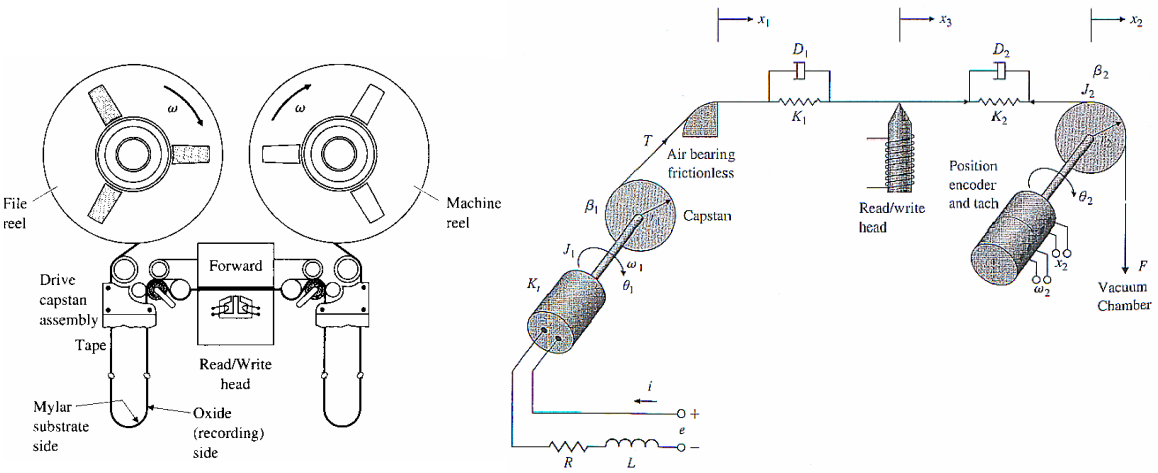
The state space form is derived from the system equations, where there is one input, which is the applied voltage, three outputs, which are: (1) tape position at the head, (2) tape tension, and (3) tape position at the wheel, and five states: (1) tape position at the air bearing, (2) drive wheel speed, (3) tape position at the wheel, (4) tachometer output speed, and (5) capstan motor speed. The following subsections present the simulation results for the investigation of different system cases using transformations with and without the use of LMI.

4.2.1 System Transformation Using Neural Identification without LMI

This subsection presents simulations of system transformation using neural identification without LMI use.

Case #1. Consider the following case of the tape transport:

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



(a) **(b)**

Figure 6. Tape-drive system: (a) Front view of a typical tape-drive mechanism and (b) schematic control model.

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

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, order model reduction may be applied. As can be seen, two real eigenvalues produce fast dynamics {-10.5772, -9.999} and one real eigenvalue produces slow dynamics {-0.9814}. We performed reduction based on the estimation of the input matrix $[\hat{\mathbf{B}}]$ and the transformed system matrix $[\hat{\mathbf{A}}]$. This estimation is achieved using NN.

By discretizing the above system with a sampling time $T_s = 0.1$ s, using a step input with learning time $T_l = 300$ s, training the NN for the input output data with a learning rate $\eta = 0.005$, the transformed model for $[\hat{\mathbf{A}}]$ and $[\hat{\mathbf{B}}]$ is:

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

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

As observed, all 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:

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

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

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 transformation has provided different eigenvalues {-0.828, -0.598± j0.930}.

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 7.

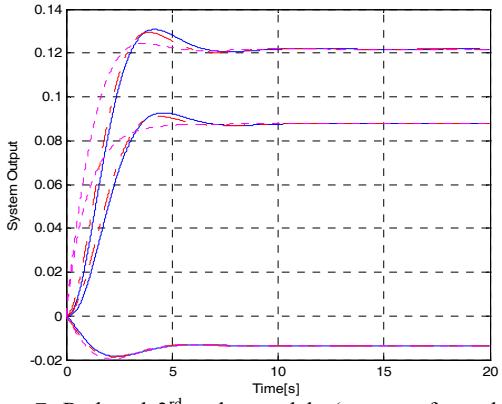


Figure 7. Reduced 3rd order models (…. transformed, -.-.-.- non-transformed) output responses to a step input along with the non-reduced model (____ original) 5th order system output response.

Based on Figure 7, it can be seen that the non-transformed reduced order model provides a response that is better than the transformed reduced model. This is due to the transformation performed only for [**A**] and [**B**] leaving [**C**] unchanged. Hence, system transformation is further considered for complete system transformation using LMI (for [**A**], [**B**], and [**D**]), where LMI-based transformation will produce better reduction-based response results than both the non-transformed and transformed without LMI.

Case #2. Consider the following 2nd case:

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

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

The eigenvalues are {-9.9973, -3.9702, -1.8992, -0.6778, -0.2055} which are all real. Utilizing the discretized model (T_s = 0.1 s) for a step input with learning time T_l = 500 s, and training the NN for the input output data with η = 1.25 x 10^{-5}, 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 reduced order model eigenvalues being as a subset of the original system. However, as shown before, the reduced order 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.

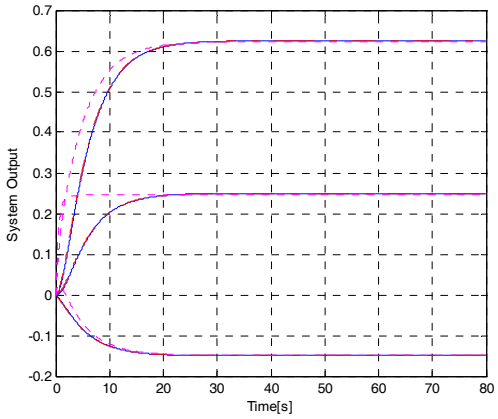


Figure 8. Reduced 3rd order models (…. transformed, -.-.-.- non-transformed) output responses to a step input along with the non-reduced (____ original) 5th order system output response.

In Figure 8, it is also seen that the response of the non-transformed reduced order model is better than the transformed reduced model, which is again caused by leaving the output [**C**] matrix without transformation.

4.2.2 LMI-Based State Transformation via Neural Identification

As previously observed, the system transformation without using LMI, where its objective was to preserve the system eigenvalues in the reduced order model, did not provide an accurate response as compared with either the reduced non-transformed or the original responses. As was explained, 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 using LMI to obtain the permutation matrix [**P**]. The following presents simulations for the previously considered tape drive system cases.

Case #1. For the example of case #1 in subsection 4.2.1, the NN estimation is used now to estimate only the transformed [$\widetilde{\mathbf{A}}_\mathbf{d}$] matrix. Discretizing the system with T_s = 0.1 s, using a step input with learning time T_l = 15 s, and training the NN for the input output data with η = 0.001, produces the transformed system matrix:

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

Based on this transformed matrix, using LMI, the permutation matrix [**P**] was computed and then used for the complete system transformation. Thus, the transformed {[$\widetilde{\mathbf{B}}$], [$\widetilde{\mathbf{C}}$], [$\widetilde{\mathbf{D}}$]} matrices were then obtained. Performing order model reduction provided the 3rd order reduced 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 clearly that the new completely transformed system is better than all of the previous reduced models. This is shown in Figure 9. As seen in the figure, the 3rd order reduced model, based on LMI transformation, provided a response that is almost the same as the 5th order original system response.

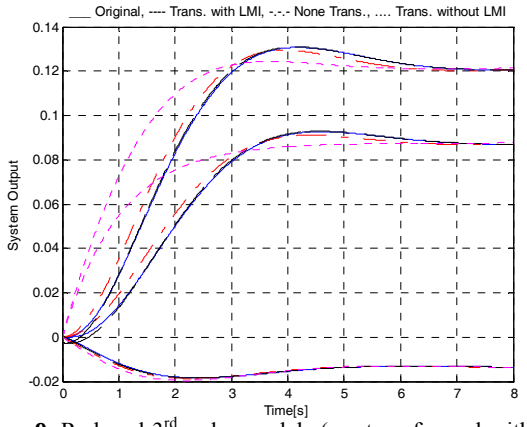


Figure 9. Reduced 3rd order models (…. transformed without LMI, -.-.-.- non-transformed, ---- transformed with LMI) output responses to a step input along with the non reduced (____ original) system output response. The LMI-transformed curve fits almost exactly on the original response.

Case #2. Investigating the example of case #2 in subsection 4.2.1, for T_s = 0.1 s, 200 input/output data points, and η = 1 x 10^{-4}, 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 10.

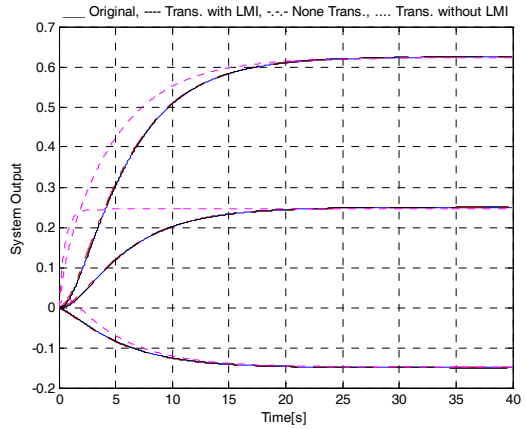


Figure 10. Reduced 3rd order models (…. transformed without LMI, -.-.-.- non-transformed, ---- transformed with LMI) output responses to a step input along with the non reduced (____ original) system output response. The LMI-transformed curve fits almost exactly on the original response.

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. Closed Loop Feedback Control For The Reduced Order Models

Using the LMI-based reduced system models presented in the previous section, different control techniques are considered in this section to obtain the desired system performance.

5.1 PID Control

A PID controller is a control feedback method which is widely used in industrial control systems [4,6]. 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 as shown in Figure 11.

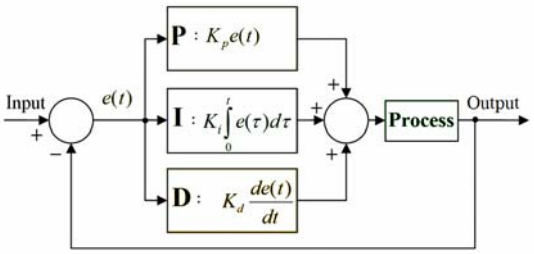


Figure 11. Single-input single-output (SISO) closed-loop feedback control using a PID controller.

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.

For the system of LMI-based case #1, focusing on one output of the tape-drive machine, we investigated the PID controller using the reduced order model for the desired output that we are interested in. Thus, the estimated reduced 3rd order model is now considered for the output of the tape position at the head represented by:

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

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, and K_d = 90 with a controlled system 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 PID controlled system for a step input shows the results in Figure 12, where the settling time is 1.5 s while without the controller was 6.5 s. Also as observed, the overshoot has much decreased after using the PID controller.

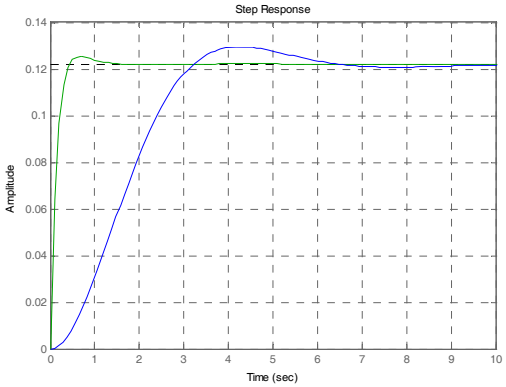


Figure 12. Reduced 3rd order model PID controlled and uncontrolled step responses.

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 13.

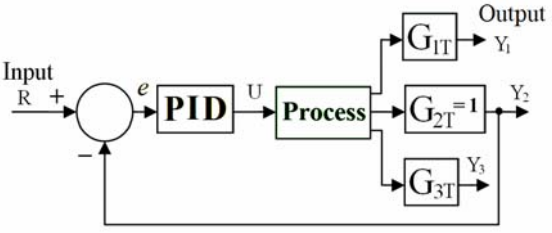


Figure 13. Single-input multiple-output (SIMO) closed loop feedback system with a PID controller.

As seen in Figure 13, the output of interest (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. As shown in Figure 13, the tuning process is accomplished using G_{1T} and G_{3T}. For example, for the 1st output G_1:

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

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

where Y_2 is the Laplace transform of the 2nd output. Similarly, G_{3T} can be obtained for the 3rd output G_3.

5.2 State Feedback Control

In this section, we will present the state feedback control (SFC) using pole placement and the LQR optimal control.

5.2.1 SFC via Pole Placement

For the reduced order model in the system of Equations (37) and (38), an LQR-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, this 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 as shown in Figure 14.

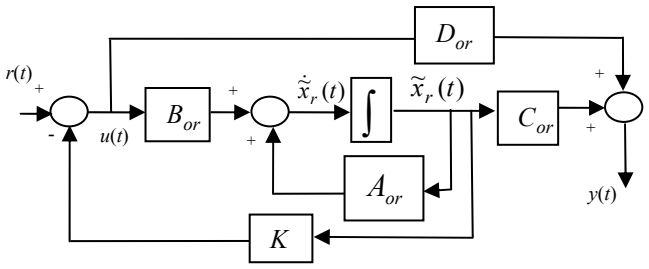


Figure 14. Block diagram of a state feedback control (SFC) with {[$\mathbf{A_{or}}$], [$\mathbf{B_{or}}$], [$\mathbf{C_{or}}$], [$\mathbf{D_{or}}$]} overall reduced order system matrices.

Replacing the control input $u(t)$ in Equations (37) and (38) by the above new control input in Equation (41) gives 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:

$$\begin{aligned}\dot{\tilde{x}}_r(t) &= A_{or}\tilde{x}_r(t) - B_{or}K\tilde{x}_r(t) + B_{or}r(t)\\ &= [A_{or} - B_{or}K]\tilde{x}_r(t) + B_{or}r(t)\\ y(t) &= C_{or}\tilde{x}_r(t) - D_{or}K\tilde{x}_r(t) + D_{or}r(t)\\ &= [C_{or} - D_{or}K]\tilde{x}_r(t) + D_{or}r(t)\end{aligned}$$

The overall closed-loop system model can 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 #2 using LMI, the state feedback was used to replace the eigenvalues by {-1.89, -1.5, -1}. The feedback control was found K = [-1.2098 0.3507 0.0184], which placed the eigenvalues as desired and enhanced the system performance as observed in Figure 15.

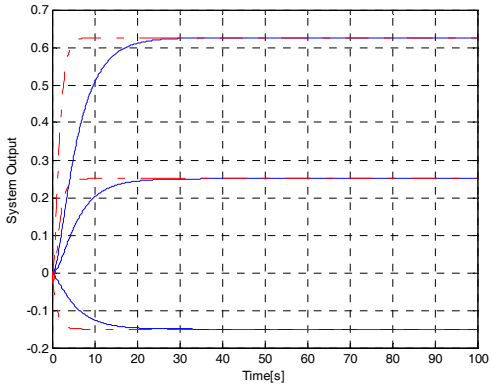


Figure 15. Reduced 3rd order state feedback control (for pole placement) output step response .-.-.- compared with the original ____ full order system output step response.

5.2.2 SFC via LQR Optimal Control

Another method for designing SFC may be achieved by minimizing the cost function given by [6]:

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

which is defined for the system $\dot{x}(t) = Ax(t) + Bu(t)$, where Q and R are weight matrices for the system states and inputs. This is known as the LQR problem, [4,6]. The feedback control law that minimizes the values of the cost is given by:

$$u(t) = -K\,x(t) \tag{47}$$

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

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

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 $Q = C^T C$.

The LQR optimization is applied to the reduced 3rd order model in case #2 of subsection 4.2.2. The state feedback optimal control gain 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 16.

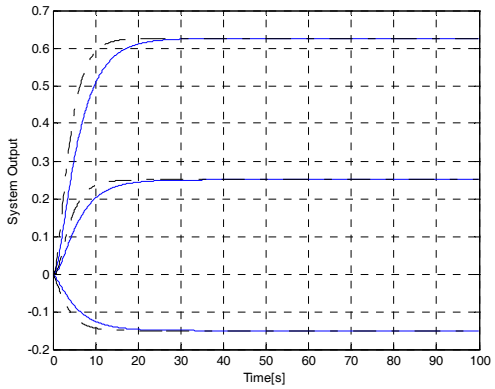


Figure 16. Output step response for the LQR state feedback control reduced 3rd order model -.-.-.- and original ____ full order system.

5.3 Output Feedback Control

The output feedback control (OFC) is another way of controlling the system for certain desired system performance as shown in Figure 17 where the feedback is directly taken from the output. This type of control is important since the output is usually measurable while the states (in the case of SFC) may not be measurable.

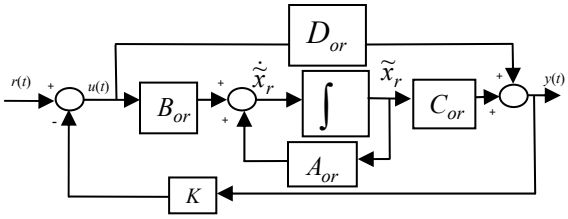


Figure 17. Block diagram of an output feedback control (OFC).

The control input is now given by $u(t) = -K\,y(t) + r(t)$, where $y(t) = C_{or}\tilde{x}_r(t) + D_{or}u(t)$. Applying this control to the considered system, the system equations become [4]:

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

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

To obtain the OFC gain K, we used the LQR-based control and then the pseudo-inverse technique with partial tuning for K was performed. Focusing on the reduced model of case #2 in subsection 4.2.2, using the OFC, the feedback control is found K = [0.5799 -2.6276 -11]. The normalized controlled system step response is shown in Figure 18, where system performance enhancement can be seen.

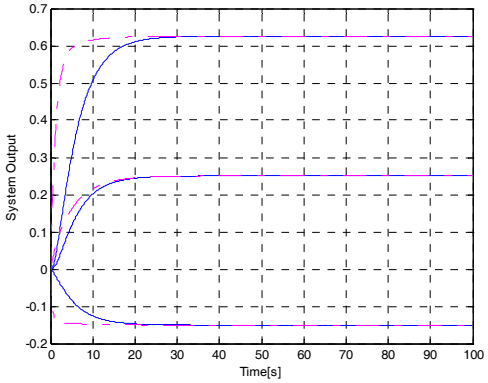


Figure 18. Reduced 3rd order output feedback controlled step response -.-.-.- compared with the original ____ full order system uncontrolled output step response.

6. Conclusion

A new method of dynamical system control is introduced in this paper. In order to achieve this control, the model order of the dynamical system was reduced. This reduction was achieved by the implementation of a NN 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 (slow dynamics) and [$\mathbf{A_o}$] contains the non-dominant eigenvalues (fast dynamics). To obtain the transformed matrix [$\tilde{\mathbf{A}}$], the zero input response ($u(t)$ = 0) was used in order to obtain output data related to the state dynamics, based only on the system matrix [**A**]. After the transformed system matrix was obtained, the optimization algorithm of Linear Matrix Inequality (LMI) was used to determine the permutation matrix [**P**], which is required to complete system transformation matrices {[$\tilde{\mathbf{B}}$], [$\tilde{\mathbf{C}}$], [$\tilde{\mathbf{D}}$]}. The reduction process was then performed 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 results show clearly that modeling and controlling dynamical systems using LMI is superior to that without using LMI.

References

[1] A. N. Al-Rabadi and O. MK. Alsmadi, "Supervised Neural Computing and LMI Optimization for Order Model Reduction-Based Control of the Buck Switching-Mode Power Supply," submitted.

[2] 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 Int. Association of Science and Tech.*, Artificial Intelligence and Application, Innsbruck, Austria, 2004.

[3] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM, 1994.

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

[5] J. H. Chow and Peter V. Kokotovic, "A Decomposition of Near-Optimal Regulators for Systems with Slow and Fast Modes," *IEEE Transactions on Automatic Control*, Vol. AC-21, pp. 701-705, October 1976.

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

[7] G. Garsia, J. Dfouz, and J. Benussou, "*H2* Guaranteed Cost Control for Singularly Perturbed Uncertain Systems," *IEEE Transactions on Automatic Control*, Vol. 43, pp. 1323-1329, September 1998.

[8] S. Haykin, *Neural Networks*: *a Comprehensive Foundation*, Macmillan College Publishing Company, New York, 1994.

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

[10] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge, 1985.

[11] S. H. Javid, "Observing the Slow States of a Singularly Perturbed Systems," *IEEE Transactions on Automatic Control,* Vol. Ac-25, pp. 277-280, April 1980.

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

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

[14] 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.

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

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