Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Polytopic Quasi-LPV Models Based on Neural State-Space Models and Application to Air Charge Control of a SI Engine

H. Abbas * **H. Werner** *

* *Institute of Control systems, Hamburg University of Technology, 21073, Hamburg, Germany (e-mail: hossam.abbas, h.werner @tu-harburg.de).*

Abstract: This paper is one of two joint papers, each presenting a different representation of a feedforward neural network. Here a discrete-time polytopic quasi linear parameter varying (LPV) model of a nonlinear system based on a neural state-space model is proposed, whereas in the joint paper (Abbas and Werner [2008]) a neural state-space model is transformed into a linear fractional transformation (LFT) representation to obtain a discrete-time quasi-LPV model of the nonlinear system. As a practical application, air charge control of a spark-ignition (SI) engine is used in both papers to illustrate two different synthesis methods for fixed structure low-order discrete-time LPV controllers. In the present paper, the synthesis of a fixed-structure low-order self-scheduled $\mathcal{H}_\infty$ controller is based on linear matrix inequality (LMIs) and evolutionary search. A controller is designed for the nonlinear system and its performance is compared with that achieved when a standard self-scheduled $\mathcal{H}_\infty$ controller is used.

1. INTRODUCTION

Linear parameter-varying (LPV) gain-scheduled control of nonlinear systems has received considerable attention recently as a way of extending linear control techniques to nonlinear systems. However, constructing an LPV model of a nonlinear plant is not trivial. Such a model can be generated from a nonlinear physical plant model; however, a suitable physical model may not always be available. Moreover, when transforming a physical model into LPV form, nonlinear characteristics may have to be approximated by polynomials. A further issue is that a physical model is typically continuous-time, and approximating a controller designed in continuous-time e.g. using Tustin approximation may lead to performance deterioration when the sampling rate is limited.

An alternative approach that avoids these problems is to identify a LPV model directly from experimental input-output data. Neural networks such as multilayer perceptron (MLP) networks can model nonlinear systems to an arbitrary degree of accuracy and have been widely used in nonlinear control applications, see e.g. (v. d. Boom et al. [2003]) and (Norgaard et al. [2000]).

A method for extracting an LPV model from a multilayer perceptron network and controller design based on this model, was proposed in (Bendtsen and Trangbæk [2002]) and extended in (Abbas and Werner [2008]), where a neural state-space model is transformed into LFT form. In (Suykens et al. [1996]), neural network models are interpreted as uncertain linear models in LFT representation, but these representations can only be used for robust controller design, not for synthesis of LPV controllers.

In this paper we propose a way to transform MLP networks into a discrete-time polytopic quasi-LPV models in a nonconservative way. A method that was presented in (Kwiatkowski et al. [2006b]) for an automated generation of affine LPV representations from nonlinear and parameter dependent systems, will be utilized here. The resulting model can be used directly to design a LPV controller from input-output data only.

In this paper we use the method proposed in (Apkarian et al. [1995]) to design a discrete-time LPV controller for the neural network model - represented as a discrete-time quasi-LPV model - and then apply the resulting controller to the original plant. The idea behind this design technique is to solve standard $\mathcal{H}_\infty$ problems at the vertices of a polytope in parameter space that contains the range of operation. Using a single Lyapunov function to show stability and finite $\mathcal{L}_2$-gain at these vertices, one guarantees that these properties will also hold for all points which are linear combinations of the vertices. One can then interpolate between these vertex controller to construct the desired controller.

For several practical reasons, the control designer often wishes to impose constraints on the order and structure of the controller. In this case however, the synthesis problem is non-convex and difficult to solve, and various approaches have been presented in the literature based on sequential LMI optimization (Ghaoui et al. [1997]), branch and bound method (Tuan and Apkairan [2000]), Nonsmooth optimization (Burke et al. [2006]), polynomial optimization by sums of squares relaxation (Hol and Scherer [2004]) and other methods. Most of these methods are used to design linear time invariant controllers only. On the other hand, designing a fixed-structure low-order LPV controller is more difficult, and only few researchers have addressed this issue (Wu and Prajna [2004]). Here we use a hybrid approach to solve this problem, based on LMIs and evolutionary search. An efficient way of initializing the

evolutionary search in this method will also be presented here.

The paper is organized as follows. Section 2 presents a method for transforming a neural state-space model into a discrete-time polytopic quasi-LPV model. The proposed controller synthesis procedure is presented in Section 3. To illustrate the proposed method, it is applied to charge control of a SI engine in Section 4. Finally, conclusions are drawn in Section 5.

2. DERIVATION OF A QUASI-LPV MODEL IN POLYTOPIC FORM FROM A NEURAL NETWORK MODEL

Consider a discrete-time nonlinear model

$$x_{k+1} = g(x_k, u_k) \tag{1a}$$
$$y_k = Cx_k \tag{1b}$$

The problem considered in this section is to represent this model in the form of a discrete-time polytopic quasi-LPV model

$$\begin{aligned} x_{k+1} &= \Sigma_i \alpha_i A_i x_k + \Sigma_i \alpha_i B_i u_k = A(\theta_k)x_k + B(\theta_k)u_k \\ y_k &= Cx_k \end{aligned} \tag{2}$$

An LPV system is called polytopic when it can be represented by state-space matrices $A(\theta_k)$, $B(\theta_k)$, $C(\theta_k)$ and $D(\theta_k)$, where the time varying parameter vector $\theta_k \in \mathbb{R}^l$ ranges over a fixed polytope and the dependence of $A(.)$, $B(.)$, $C(.)$ and $D(.)$ on θ_k is affine. The time varying parameter vector is assumed to depend on a vector of measurable signals $\rho_k \in \mathbb{R}^r$ referred to as scheduling signals, according to

$$\theta_k = s(\rho_k) \tag{3}$$

where $s : \mathbb{R}^r \to \mathbb{R}^l$ is a continuous mapping. In the case of quasi-LPV systems, some or all of the scheduling parameters depend on measured system input and output. A matrix polytope is defined as the convex hull of a finite number of matrices N_i with the same dimension. That is,

$$Co\{N_i, i = 1, \ldots, l\} := \left\{ \sum_{i=1}^{l} \alpha_i N_i : \alpha_i \geq 0, \sum_{i=1}^{l} \alpha_i = 1 \right\}$$

The time varying parameter θ_k varies in a polytope Θ, which is assumed to be a compact set, with vertices $v_1, v_2, \ldots, v_r$; that is,

$$\theta_k \in \Theta := Co\{v_1, v_2, \ldots, v_r\}$$

The problem considered in this section is to transform the neural state-space model (1a),(1b) into a polytopic quasi-LPV model (2) which has the above properties with

$$[A(\theta_k)\ B(\theta_k)] \in \mathcal{P}_\theta := Co\left\{ [A_i\ B_i], i = 1, \ldots, r \right\} \tag{4}$$

where $\mathcal{P}_\theta \subset \mathbb{R}^l : \theta \in \mathcal{P}_\theta \forall k > 0$.

The idea in (Kwiatkowski et al. [2006b]), which proposed a method to generate affine LPV representations from nonlinear systems, will be extended here to construct the above quasi-LPV model from a given MLP feedforward neural network containing a single linear hidden layer with l neurons and hyperbolic tangent activation functions. Assume that the nonlinear system dynamics are represented in the following neural state-space form, which has zero bias in the output layer

$$\tilde{x}_{k+1}^{1:p} = W_o \tanh(W_x \tilde{x}_k + W_u \tilde{u} + \tilde{W}_b) \tag{5a}$$
$$\tilde{y}_k = C\tilde{x}_k \tag{5b}$$

where $\tilde{x}_k = [\tilde{y}_k^T \quad \tilde{y}_{k-1}^T \quad \ldots \quad \tilde{y}_{k-ny}^T]^T \in \mathbb{R}^n$ is the state vector. Note that $\tilde{x}_{k+1}$ given in (5a) is just the predicted output vector from the neural network, i.e. $\tilde{x}_{k+1}^{1:p} = \hat{\tilde{y}}_k$, and the rest of the state vector is given as follows

$$\tilde{x}_{k+1} = [\tilde{x}_{k+1}^{1:p} \quad \tilde{y}_k^T \quad \tilde{y}_{k-1}^T \quad \ldots \quad \tilde{y}_{k-ny+1}^T]^T$$

$\tilde{y}_k \in \mathbb{R}^p$ is the output vector of the system, $\tilde{u} = [\tilde{u}_k^T \quad \tilde{u}_{k-1}^T \quad \ldots \quad \tilde{u}_{k-nu}^T]^T$, $\tilde{u}_k \in \mathbb{R}^m$ is a control signal. $W_o \in \mathbb{R}^{p \times l}$ and $W_x \in \mathbb{R}^{l \times n}$, $W_u \in \mathbb{R}^{l \times (m \times n_u)}$ contain the output and hidden layer weights, respectively. $\tilde{W}_b \in \mathbb{R}^l$ contains a set of biases in the hidden layer. Prior to the network training, the user has to select values for l and n_y, n_u, such that a good estimation of the parameter matrices W_o, W_x, W_u and vector $\tilde{W}_b$ is obtained (Norgaard et al. [2000]). Note that the choice of the number of hidden neurons l will determine the number of scheduling parameters of the LPV model. For simplicity we will assume $n_u = n_y$.

In order to remove the bias $\tilde{W}_b$ from (5a), it is assumed that there exists an equilibrium point $(\tilde{x}_k, \tilde{u}) = (\tilde{x}_k^o, \tilde{u}^o)$, such that

$$0 = W_o \tanh(W_x \tilde{x}_k^o + W_u \tilde{u}^o + \tilde{W}_b)$$

This equilibrium can be determined as

$$\begin{bmatrix} \tilde{x}_k^o \\ \tilde{u}^o \end{bmatrix} = -W^+ \tilde{W}_b \tag{6}$$

where $W = [W_x \quad W_u]$ is assumed to have full column rank and W^+ is a right inverse. Then the network coordinates can be changed such that the new coordinates are $x_k = \tilde{x}_k - \tilde{x}_k^o$, $u = \tilde{u} - \tilde{u}^o$. As a result, (5a) can be written as

$$x_{k+1}^{1:p} = W_o \tanh(W_x x_k + W_u u) \tag{7}$$

Now, define the following time varying parameter

$$\theta_k^j = \begin{cases} \dfrac{\tanh(W_x^j x_k + W_u^j u)}{(W_x^j x_k + W_u^j u)}, & (W_x^j x_k + W_u^j u) \neq 0 \\ 1, & (W_x^j x_k + W_u^j u) = 0 \end{cases} \tag{8}$$

for $1 \leq j \leq l$, where W_x^j, W_u^j denote the j^{th} rows in the respective hidden layer weight matrices. Then (7) can be rewritten as

$$x_{k+1}^i = W_o^i \Theta_d (W_x x_k + W_u u_k) \tag{9}$$

for $1 \leq i \leq p$, where $\Theta_d \in \mathbb{R}^{l \times l}$ is a diagonal matrix that contains the variable parameters of the LPV model, and W_o^i denotes the i^{th} row in the outer layer weight matrix. In this way, the neural network model is transformed into a quasi-LPV model (2) where

$$A(\theta_k) = \begin{bmatrix} A_{11}(\theta_k) & A_{12}(\theta_k) & \ldots & A_{1n}(\theta_k) \\ \vdots & \vdots & \vdots & \vdots \\ A_{p1}(\theta_k) & A_{p2}(\theta_k) & \ldots & A_{pn}(\theta_k) \\ 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \tag{10}$$

with $A_{ri}(\theta_k) = \sum_{j=1}^{l} w_o^{rj} \theta_k^j w_x^{ji}$, where, w_o^{rj}, w_x^{ji} and w_u^{ji} are the elements of W_o^i, W_x^j and W_u^j, respectively,

$$B(\theta_k) = \begin{bmatrix} B_{11}(\theta_k) & B_{12}(\theta_k) & \dots & B_{1(m\times n_u)}(\theta_k) \\ \vdots & \vdots & \vdots & \vdots \\ B_{p1}(\theta_k) & B_{p2}(\theta_k) & \dots & B_{p(m\times n_u)}(\theta_k) \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{11}$$

with $B_{ri}(\theta_k) = \sum_{j=1}^{l} w_o^{rj}\theta_k^j w_u^{ji}$, and

$$C = [I_{p\times p}\ 0\ \dots\ 0] \tag{12}$$

The time varying parameters can be collected into a vector $\theta \in \mathbb{R}^l$; its bounds belong to $[\theta^{j,min} \quad \theta^{j,max}]$ where

$$\theta^{j,min} = \min_{0\le k\le T} \theta^j \tag{13a}$$

$$\theta^{j,max} = \max_{0\le k\le T} \theta^j \tag{13b}$$

again for $1 \le j \le l$ where $k \in [0 \quad T]$ is the time interval in which the training data have been acquired. It should be noted that $0 \le \theta^{j,min}, \theta^{j,max} \le 1$ so no scaling is required. This completes the transformation of the neural state-space model (5a), (5b) into a quasi-LPV model in the polytopic representation (2) with the condition (4) where $A(\theta)$, $B(\theta)$, and C are given by (10) - (12). The time varying parameter vector is defined by (8) and its bounds are given by (13a),(13b).

3. DISCRETE-TIME CONTROLLER DESIGN

This section describes the discrete-time design of a fixed-structure, low-order, self-scheduled $\mathcal{H}_\infty$ controller, based on a model obtained by the method of the previous section. A hybrid evolutionary-algebraic approach for solving the non-convex problem of fixed structure and low-order controller design that was proposed in (Farag and Werner [2004]) is utilized here for this purpose. The method is based on the following assumptions (Apkarian et al. [1995]):

- The discrete-time polytopic LPV system has the form
$$\begin{aligned} x_{k+1} &= A(\theta_k)x_k + B_1(\theta_k)w_k + B_2(\theta_k)u_k \\ zk &= C_1(\theta_k)x_k + D_{11}(\theta_k)w_k + D_{12}(\theta_k)u_k \\ y_k &= C_2(\theta_k)x_k + D_{21}(\theta_k)w_k + D_{22}(\theta_k)u_k \end{aligned} \tag{14}$$
and the system matrices are assumed to belong the polytope $\mathcal{P}_\theta$ defined by
$$\mathcal{P}_\theta := Co\left\{ \begin{bmatrix} A_i & B_{1i} & B_{2i} \\ C_{1i} & D_{11i} & D_{12i} \\ C_{2i} & D_{21i} & D_{22i} \end{bmatrix}, i = 1, 2, \dots, l \right\} \tag{15}$$
where A_i, B_{1i}, ..., denote the values of the matrices $A(\theta_k)$, $B_1(\theta_k)$, ... at the vertices v_i of the parameter polytope Θ.
- in (14) the system matrices $B_2(\theta_k)$, $C_2(\theta_k)$, $D_{12}(\theta_k)$ and $D_{21}(\theta_k)$ are parameter dependent.
- $D_{22}(\theta_k) = 0$.
- the pairs $(A(\theta_k), B_2)$ and $(A(\theta_k), C_2)$ are quadratically stabilizable and quadratically detectable over Θ respectively.
- The discrete-time LPV controller which establishes quadratic $\mathcal{H}_\infty$ performance for the closed-loop system has the following state space representation
$$\Omega(\theta) := \begin{bmatrix} A_K(\theta_k) & B_K(\theta_k) \\ C_K(\theta_k) & D_K(\theta_k) \end{bmatrix} \tag{16}$$
where A_K, B_K, C_K, D_K are affine in θ.
- The closed-loop system can be compactly written in the form
$$\begin{aligned} x_{k+1}^{cl} &= A_{cl}(\theta_k)x_k^{cl} + B_{cl}(\theta_k)w_k \\ z_k &= C_{cl}(\theta_k)x_k^{cl} + D_{cl}(\theta_k)w_k \end{aligned} \tag{17}$$

In the closed-loop system (17) the system matrices can be represented as $A_{cl}(\theta_k) = A_o(\theta_k) + \tilde{B}\Omega(\theta_k)\tilde{C}$, $B_{cl}(\theta_k) = B_o(\theta_k) + \tilde{B}\Omega(\theta_k)\tilde{D}_{21}$, $C_{cl}(\theta_k) = C_o(\theta_k) + \tilde{D}_{12}\Omega(\theta_k)\tilde{C}$, $D_{cl}(\theta_k) = D_{11}(\theta_k) + \tilde{D}_{12}\Omega(\theta_k)\tilde{D}_{21}$ where

$$A_o(\theta_k) = \begin{bmatrix} A(\theta_k) & 0 \\ 0 & 0_{n_k\times n_k} \end{bmatrix}, \quad B_o = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}, \tilde{D}_{21} = \begin{bmatrix} 0 \\ D_{21} \end{bmatrix}$$

$$C_o = [C_1\ 0], \qquad \tilde{B} = \begin{bmatrix} 0 & B_2 \\ I_{n_k} & 0 \end{bmatrix},$$

$$\tilde{C} = \begin{bmatrix} 0 & I_{n_k} \\ C_2 & 0 \end{bmatrix}, \qquad \tilde{D}_{12} = [0\ D_{12}]$$

where $n_k < n$ is the order of the controller.

The synthesis is based on the following result given in (Apkarian et al. [1995]) and (Apkarian et al. [1996]).

Theorem 1. Consider the discrete-time polytopic LPV system (14). There exists a discrete-time LPV controller guaranteeing quadratic $\mathcal{H}_\infty$ performance γ along all parameter trajectories in the parameter polytope Θ if and only if there exist a symmetric matrix X_{cl} in $\mathbb{R}^{(n+n_k)\times(n+n_k)}$ satisfying the system of $l+1$ LMIs

$$\begin{bmatrix} A_{cli}^T X A_{cli} - X & A_{cli}^T X B_{cli} & C_{cli}^T \\ B_{cli}^T X A_{cli} & B_{cli}^T X B_{cli} - \gamma I & D_{cli}^T \\ C_{cli} & D_{cli} & -\gamma I \end{bmatrix}, i = 1, ..., l+1 \tag{18}$$

The condition in (18) - for a given controller - is linear in the variables γ and X_{cl}. To determine the optimal performance measure, one needs to find the solution to the problem

$$\min_{X_{cl}} \gamma \quad \text{subject to (18)} \tag{19}$$

To solve the non-convex problem of finding a controller of order $n_k < n$ that minimizes the performance measure, we follow the approach in Farag and Werner [2004] and split the original problem into a convex subproblem - the minimization problem (19) - and a nonconvex one - the search for the controller parameters. The former one can be solved with LMI solvers and the latter one with evolutionary search techniques. Let $\mathcal{K}$ denote a set of local controllers $\{K_1, \dots, K_l\}$ for each vertex. The synthesis procedure can be summarized as follows.

- **Generate** an initial random population of controllers for all vertices $\{\mathcal{K}_1, \dots, \mathcal{K}_\mu\}$.
- **Evaluate** the objective function
$$f(\mathcal{K})_i = \begin{cases} \hat{\gamma} & \text{if } A_{cli} \text{ is stable for each vertex,} \\ \kappa(A_{cli}) + \beta_u & \text{if } A_{cli} \text{ is unstable for any vertex,} \\ \beta_s & \text{if } A_{cli} \text{ stable for each vertex, (19) infeasible for any vertex} \end{cases}$$
where $\hat{\gamma}$ is the solution to the minimization problem (19). The term $\kappa(A_{cli})$ denotes the maximum real

part of the eigenvalues of A_{cli} for any vertex, and $\beta_u \gg \beta_s$ represents a penalty for destabilizing controllers and infeasible inequalities, respectively.
- **Evolve** the current population, i.e. use ranking to evaluate the fitness and apply evolutionary operators (mutation and crossover).
- **Repeat** the steps *Evaluate* and *Evolve* until a stopping criterion is met.

Next we propose a way to initialize the population of the evolutionary search. First, the problem is solved for each vertex, i.e. (19) subject to a single LMI. Then the populations associated with each vertex are fused into a single population whose size equals the number of decision variables. This larger population is then taken as the initial population of the original problem of synthesizing a self-scheduled $\mathcal{H}_\infty$ controller.

The selection of the structure (n_y, n_u, l) of the neural state-space model is not trivial. Moreover, the assessment of the quality of a neural network model should not be done based on cross validation by simulation alone - a model should be assessed in terms of achievable control performance when the synthesis is based on this model. For this purpose, a whole set of neural network models should be trained, with different structure and using different training signals. The modelling and synthesis procedure described above can then be applied to all models in the set, and the achieved control performance can be compared.

Since the number of the time varying parameters θ_k, (8) depends on the number of neurons in the hidden layer of the neural state-space model, the efficiency of the synthesis approach is improved as that number is reduced, because the number of the LMIs ($2l + 1$ and $l + 1$ in the case of full-order and fixed-structure low-order controllers, respectively) is decreased, as well as the number of decision variables. This should be taken into consideration when the number of the hidden neurons is chosen during the modelling stage.

4. APPLICATION TO CHARGE CONTROL OF AN SI ENGINE

In the following, the design procedure described in the previous section is applied to a continuous-time nonlinear Simulink model of the intake manifold of a SI engine (Kwiatkowski et al. [2006a]) for air charge control, for more details about the physical model refer to (Kwiatkowski et al. [2006a]). The nonlinear dynamics of the system will be represented as a discrete time quasi-LPV model based on a neural network model obtained from the input-output data only. Since the engine is controlled by a digital controller and the model is in discrete-time form, a discrete-time LPV controller will be designed.

4.1 The Intake Manifold of an SI Engine

The intake manifold is not an isolated system but it is part of the overall system of the vehicle, Fig. 1. The nonlinear engine block generates the torque T_e from the normalized air charge m_{nac} and the motor speed N, and the vehicle model generates the motor speed from the torque and some fixed environmental conditions. The opening of the

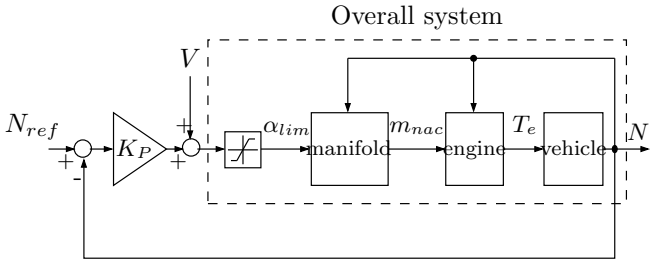

Fig. 1. Overall system in the closed loop system identification

throttle valve in the intake manifold α_{lim} is used to control the amount of the normalized air charge. It should be noted here that the speed of the engine influences the internal dynamics of the intake manifold and the engine.

The vehicle model as shown in Fig. 1 has integral behavior; for this reason the loop is closed between the engine speed and α_{lim} through a proportional gain K_p as shown in Fig. 1 in order to stabilize the engine speed during the input-output data collection for system identification.

4.2 Nonlinear System Identification with Neural Networks

Generating an input signal to excite the different dynamics of the nonlinear system is a crucial step in system identification. The required operating ranges, the sampling frequency and the bandwidth of the system are important information to design a rich training signal. As shown in Fig. 1, the intake manifold system has two inputs α_{lim} and N, and a single output m_{nac}. The aim here is to construct a single-input single-output black box model that is robust against variation of the engine speed. This can be achieved if the training signal excites the system at different levels of the engine speed. For this purpose, a random signal, N_{ref}, in the low frequency range is designed to cover the whole range of the engine speed and another pseudo-random multilevel sequence V in the high frequency range (Norgaard et al. [2000]), up to the Nyquist frequency, is designed to excite the input-output dynamics of the system (from α_{lim} to m_{nac}). The way the signals enter the loop is shown in Fig. 1. The input-output data are collected and divided into training and validation sets. The required operating ranges for the different variables to design the input signal are

$$m_{nac} \in [10\ 80]\%,\ \alpha_{lim} \in [0\ 100]\%,\ N \in [760\ 6250]rpm$$

Here NARX model structures are used for the identification with different values of n_y, n_u, l. Several feedforward networks with different structure were separately trained using the Levenberg-Marquardt optimization algorithm. Cross validation confirmed that each model had learned the behavior of the intake manifold system and was robust against the variations of the engine speed. In this way a set of neural state-space models was determined.

4.3 Controller Implementation and Simulation Results

The first step in the proposed design approach is to convert the neural network models into polytopic discrete-time quasi-LPV models using the method of Section 2. Next, controllers are designed for these models. The design objectives considered here are

- Rise time $t_r < 0.3s$
- Zero steady state error
- Small overshoot
- Constraint on actuator usage

To meet these objectives, two shaping filters (weights) were used to tune the controller: W_S for sensitivity and W_{KS} for control sensitivity. These were adjusted by changing their gains and bandwidths until the required performance was achieved with the following values:

$$W_S = 0.00178\frac{1-z^{-1}}{1-0.995z^{-1}} \quad (20)$$

$$W_{KS} = 0.0001\frac{2.97-0.2z^{-1}}{1+0.9802z^{-1}} \quad (21)$$

Using the hybrid evolutionary-algebraic approach discussed in Section 2, a discrete-time LPV-PID controller for the intake manifold system was computed. For comparison, a full-order LPV controller was also designed by solving the associated LMI problem.

The full order controller was designed first. Because the input matrix B_2 in (14) depends on time varying parameters, pre-filtering of the control input u_k is required to remove this dependence as discussed in (Apkarian et al. [1995]). Thus, a first-order pre-filter with bandwidth larger than the desired system bandwidth is added ($P.F2$ in Fig. 2). The design algorithm which is proposed in Section 4 is applied to obtain a full-order controller for each model. These controllers are then tested on the nonlinear model of the intake manifold. To avoid windup effects after actuator saturation, an integral anti-windup with fixed gain K_{aw} Campo and Morari [1990] is applied as shown in Fig. 2 (the solid part). By comparing the achieved performance (in terms of the induced $\mathcal{L}_2$ gain), the best controller and the corresponding model are determined. Here the controller algorithm was applied to 450 neural state-state space models with different structure and trained with different signals. Nearly half of these models turned out to be not quadratic stabilizable or detectable. The best model has $l = 3$, $n_u = 1$ and $n_y = 2$ and the smallest induced $\mathcal{L}_2$ achieved is $\gamma \approx 2.35$. The corresponding discrete-time controller is then connected to the continuous-time nonlinear model through digital-to-analog (DA) and analog-to-digital (AD) converters as shown in Fig. 2, together with the anti-windup and another pre-filter (P.F 1), which is used to shape the step response.

For practical implementation in the electronic control unit of the car, a low-order controller is required. An LPV-PID controller was designed using the hybrid algorithm presented in Section 3. The controller has the structure

$$K(\theta_k) = K_P(\theta_k)+K_I(\theta_k)\frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}+\frac{K_D(\theta_k)}{1+\frac{\alpha T}{2}\frac{1+z^{-1}}{1-z^{-1}}} \quad (22)$$

where T is the sampling time and α is a suitable constant required to make the PID controller realizable. The LPV-PID controller is designed for the best model. The synthesis procedure is then applied to find and tune the coefficients of $K(\theta_k)$ in (22) and the resulting controller is tested on the nonlinear system. Again, an anti-windup, this time consisting only of a fixed gain K_{aw} as shown in Fig. 2 (the dotted part) is applied to the integral part of the LPV-PID controller as proposed in (Kwiatkowski et al. [2006a]) to avoid windup effects. As before, the controller is connected to the nonlinear system through AD and DA converters and the two pre filters (P.F 1 and P.F 2) are added for the same reasons as above.

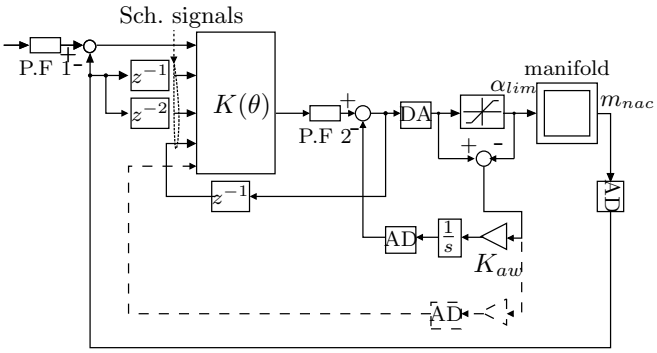

Fig. 2. Block diagram of the closed-loop system with controller and anti-wind up, solid loop for full order and dashed one for LPV-PID controllers, respectively

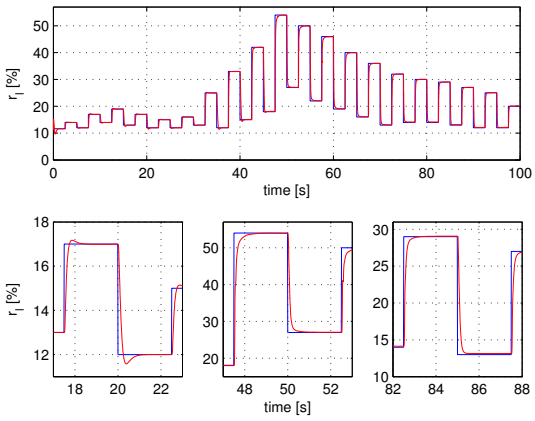

Fig. 3. Tracking of m_{nac} with a full-order controller

The closed-loop tracking performance is illustrated in Fig. 3 and Fig. 4 for the full-order and the LPV-PID controllers, respectively. The full-order controller as well as the LPV-PID controller follow the reference trajectory in a satisfactory manner and both achieved the required objectives. Table 1 shows a comparison between the full-order and the LPV-PID controller synthesis and the performance of the intake manifold. Table 1, Fig. 3 and Fig. 4 show how the output of the intake manifold with the LPV-PID controller successfully tracks the required trajectory and with approximately the same performance as the full-order controller. It should be kept in mind that the LPV controller synthesis based on a fixed Lyapunov function for the whole range of operation - which was used here - introduces conservatism into the design. This explains the improvement in the controller performance reported in (Abbas and Werner [2008]), where LPV controllers are designed for the same system but with a parameter dependent Lyapunov function.

Remark 1: The method proposed in Section 3 to find an initial population for the controller synthesis problem worked successfully in the present problem where the number of decision variables is (3×2^l=24) with population size 60 and 100 generations. First, the problem (19) is

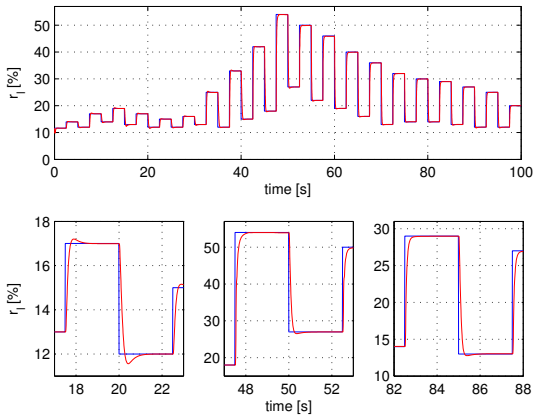

Fig. 4. Tracking of m_{nac} with a LPV-PID controller

Table 1. Comparison between full-order controller and LPV-PID controller

	Full-order	LPV-PID
Order	5^{th}	2^{nd}
t_r	< 0.16s	< 0.15s
Anti-wind up type	integrator with static gain	static gain
Max. overshoot	1%	1.3%
Max. undershoot	3.8%	4.3%

solved for each vertex subject to a single LMI condition. In this case the number of decision variables is only 3 and a feasible solution at each vertex is easily found; the size of the population associated with each vertex is 60. All vertex populations are then fused into a single population of 60 individuals by concatenating vertex individuals representing 3 decision variables into individuals representing candidate solutions to the original problem with 3×2^l variables.

Remark 2: The fact that the integral part explicitly appears in the PID controller facilitates the use of anti-windup. For the full-order controller, an anti-windup scheme proposed in (Campo and Morari [1990]) was used that works well in simulation but increases the complexity of the controller.

5. CONCLUSIONS

A procedure for transforming neural state-space model into a nonconservative discrete-time polytopic quasi-LPV model has been proposed. Based on this model, a hybrid evolutionary-algebraic synthesis procedure is used to design a discrete-time, fixed structure and low-order self-scheduled $\mathcal{H}_\infty$ controller that guarantees stability and performance for a wide range of operation. In the design procedure the interdependence of the modelling step and controller synthesis step is taken into account. A number of different neural state-space models are assessed in terms of the performance achieved with the corresponding controllers. The proposed method was successfully applied to design the air charge controller of a SI engine, leading to a discrete-time LPV-PID controller based only on input-output data of the nonlinear plant. Simulation studies showed satisfactory performance similar to the performance of a full-order LPV controller that was designed for comparison.

REFERENCES

H. Abbas and H. Werner. LPV design of charge control for an SI engine based on LFT neural state-space models. In *Proc. IFAC World Congress 2008*, 2008.

P. Apkarian, P. Gahinet, and G. Becker. Self-scheduled H_∞ control of linear parameter-varying systems: a design example. *Automatica*, 31(9):1251–1261, 1995.

P. Apkarian, G. Becker, P. Gahinet, and H. Kajiwara. LMI techniques in control engineering from theory to practice. In *Workshop notes CDC*, Kobe, Japan, 1996. IEEE.

J. Bendtsen and K. Trangbæk. Roubst quasi-LPV control based on neural state space models. *IEEE Transactions on Neural Networks*, 13(2):355–368, 2002.

J. Burke, D. Henrion, A. Lewis, and M. Overton. HIFOO - a Matlab package for fixed-order controller design and H_∞ optimization. In *IFAC Symposium on Robust Control Design*, 2006.

P. J. Campo and M. Morari. Robust control of processes subject to saturation nonlinearities. *Computers and Chemical Engineering*, 14(4/5):343–358, 1990.

A. Farag and H. Werner. A Riccati - genetic algorithms approach to fixed-structure controller synthesis. In *Proc. of the American Control Conference*, 2004.

L. Ghaoui, F. Oustry, and M. AitRami. A cone complementarity linearization algorithm for static output feedback and related problems. *IEEE Transactions on Automatic Control*, 42(8):1171–1176, 1997.

C. Hol and C. Scherer. Computing optimal fixed order h_∞-synthesis values by matrix sum of squares relaxations. In *43rd Proc. of the IEEE Conference on Desision and Control*, 2004.

A. Kwiatkowski, J. Blath, H. Werner, and M. Schultalbers. Application of LPV gain scheduling to charge control of a SI engine. In *Proc. of IEEE International Symposium on Computer-Aided Control Systems Design*, Munich, Germany, 2006a.

A. Kwiatkowski, M. Boll, and H. Werner. Automated generation and assessment of affine LPV models. In *Proc. of the IEEE Conference on Decision and Control*, San Diego, CA, 2006b.

M. Norgaard, O. Ravn, N. Poulsen, and L. Hansen. *Neural Networks for Modelling and Control of Dynamic Systems*. Springer-Verlag, London, UK, 2000.

J. A. K. Suykens, J. P. L. Vandewalle, and B. L. R. D. Moor. *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

H. D. Tuan and P. Apkairan. Low nonconvexity-rank bilinear matrix inequalities: Algorithms and applications in robust controller and strucure design. *IEEE Transactions on Automatic Control*, 45(8):2111–2117, 2000.

T. v. d. Boom, M. A. Botto, and J. S. d. Costa. Robust control of dynamical systems using neural networks with input-output feedback linearization. *International Journal of Control*, 76(18):1783–1789, 2003.

F. Wu and S. Prajna. A new solution approach to polynomial LPV system analysis and synthesis. In *Proc. of the American Control Conference*, 2004.