Engineering Applications of Artificial Intelligence 23 (2010) 283–289

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Robust air/fuel ratio control with adaptive DRNN model and AD tuning

Yu-Jia Zhai [a], Ding-Wen Yu [b], Hong-Yu Guo [c], D.L. Yu [a,*]

[a] Control Systems Research Group, Liverpool John Moores University, UK
[b] Northeast University at Qinhuangdao, China
[c] Zhejiang Gong Shang University, China

ARTICLE INFO

Article history:
Received 17 November 2008
Received in revised form
4 December 2009
Accepted 17 December 2009

Keywords:
Air/fuel ratio control
SI engines
Recurrent neural networks
Adaptive neural networks
Model predictive control

ABSTRACT

Current production engines use look-up table and proportional and integral (PI) feedback control to regulate air/fuel ratio (AFR), which is time-consuming for calibration and is not robust to engine parameter uncertainty and time varying dynamics. This paper investigates engine modelling with the diagonal recurrent neural network (DRNN) and such a model-based predictive control for AFR. The DRNN model is made adaptive on-line to deal with engine time varying dynamics, so that the robustness in control performance is greatly enhanced. The developed strategy is evaluated on a well-known engine benchmark, a simulated mean value engine model (MVEM). The simulation results are also compared with the PI control.

1. Introduction

Automotive engine emission is a major source of urban air pollution. The spark-ignition (SI) engine exhaust gases contain oxides of nitrogen (nitric oxide NO and small amounts of nitrogen dioxide, NO_2—collectively known as NO_x), carbon monoxide (CO) and organic compounds which are unburned or partially burned hydrocarbons (HC) (Heywood, 1988). Governments set strict emission standards for engines to reduce the air pollution. These regulations provide control challenge on their own, but when combined with other regulations and system objectives, such as fuel economy and performance, the challenges become significant.

For SI engines one of the most important variables in determining emissions is the AFR. In term of control engineering, the target is to maintain the AFR at stoichiometric value (14.7) for both steady state and transient operation, which is the best solution to minimum emission and a widely accepted balance between power output and fuel consumption. The stoichiometric value ensures the maximum efficiency of three-way catalysts (TWC) so that minimum emissions. Variations of greater than 1% below 14.7 can result in significant increase of CO and HC emissions. An increase of more than 1% will produce more NO_x up to 50% (Manzie et al., 2001, 2002). However, the dynamics of air manifold and fuel injection of SI engines are very fast, severely nonlinear and with constraints imposed on the states and inputs (Balluchi et al., 2000; Vinsonneau et al., 2003). Therefore, they present a challenge problem to control engineers. The production automotive engine control schemes use mainly look-up tables and feedback controller, which take a huge effort and labours for engine calibration and is not robust to mechanical wear of engine parts and engine-to-engine dynamic differences.

Different control strategies have been developed for AFR in the past two decades, including the sliding mode control (Wang and Yu, 2007), feed-forward feedback control (Zhai and Yu, 2008), model predictive control (MPC) using RBF models, and integral sliding mode with RBF network compensation (Wang et al., 2006). However, the MPC using RBF model, a feed-forward neural networks, uses back-propagation or its other variations for training (Shayler et al., 2000). The main drawback of this approach is that it can only provide accurate predictions for a predetermined finite number of steps, in most cases, only one step ahead (Al Seyab and Cao, 2007). When prediction horizon is increased, the prediction accuracy is greatly reduced. This drawback makes such models not well suitable for predictive control, where multi-step predictions are desired.

In this paper, we study the applicability and the effectiveness of DRNN-based mode predictive control to regulate the air/fuel ratio. To model engine parameter uncertainty and severe non-linear dynamics in different operating regions, the DRNN is on-line adapted by dynamic back-propagation algorithm that is implemented using automatic differentiation (AD) technique. The control result is compared with traditional feedback control. Finally, the robustness of the system performance to the system uncertainty is evaluated under engine operating condition.

2. DRNN modelling

Recurrent NNs (RNN) have important capabilities that are not found in the static feed-forward networks, such as attractor dynamics

* Corresponding author.
E-mail address: D.Yu@ljmu.ac.uk (D.L. Yu).

0952-1976/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2009.12.006

and the ability to store information for later use (Lapedes and Farber, 1986). Of particular interest is their ability to deal with time varying input or output through their own natural temporal operation. Thus, the RNN is a dynamic mapping and is better suited for dynamic systems modelling than the static feed-forward networks. Considering the computation burden of MPC for fast dynamic system, the diagonal recurrent neural network, instead of fully connected recurrent neural networks (FRNN), is used in this study. The DRNN has one hidden layer that is comprised of self-recurrent neurons from their own output only. The signals within the hidden layer are the feedback signals from the output layer and with different time-delays. This gives possibility of the DRNN to model the dynamic system with different input orders and different output orders. Since there is no inter-links among neurons in the hidden layer, DRNN has considerably fewer weights than FRNN and the network is simplified considerably (Ku and Lee, 1995).

The DRNN output can be calculated from its input and weights as follows.

$$\hat{y}(k) = W^y \begin{bmatrix} h(k) \\ 1 \end{bmatrix} \tag{1}$$

$$h(k) = f\left[z(k)\right] \tag{2}$$

$$z(k) = W^h \begin{bmatrix} x(k) \\ 1 \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} w^d_{1,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w^d_{1,q} \end{bmatrix} & \cdots & \begin{bmatrix} w^d_{v,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w^d_{v,q} \end{bmatrix} \end{bmatrix} \begin{bmatrix} h^d(k-1) \\ \vdots \\ h^d(k-v) \end{bmatrix} \tag{3a}$$

$$= W^h \begin{bmatrix} x(k) \\ 1 \end{bmatrix} + \begin{bmatrix} diag(w^d_1) & \cdots & diag(w^d_v) \end{bmatrix} \begin{bmatrix} h^d(k-1) \\ \vdots \\ h^d(k-v) \end{bmatrix} \tag{3b}$$

$$h^d(k) = h(k) \in \Re^q, \quad \text{feedbackafteractivationfunction} \tag{4}$$

where $f(\cdot)$ is the nonlinear activation function in hidden layer, and the other variables and parameters can be referred to Ref. Ku and Lee (1995).

The DRNN is trained using the dynamic back-propagation algorithm (Wang and Zhang, 1997). Because the weights in the feedback loop are nonlinearly related to the network output, optimization algorithms for linear systems cannot be used. Here, training of these weights is achieved using a so-called dynamic back-propagation algorithm. Let $y(k)$ and $\hat{y}(k)$ be the actual responses of the plant and the output of the DRNN model, then an error function for a training cycle for DRNN can be defined as

$$E_m = \tfrac{1}{2}[y(k) - \hat{y}(k)]^2 \tag{5}$$

The gradient of error simply becomes

$$\frac{\partial E_m}{\partial W} = -e_m(k)\frac{\partial \hat{y}(k)}{\partial W} \tag{6}$$

where $e_m(k) = y(k) - \hat{y}(k)$ is the output error between the plant and the DRNN. The output gradients with respect to output, recurrent and input weights, respectively, are given by

$$\frac{\partial \hat{y}(k)}{\partial W^y} = h(k) \tag{7}$$

$$\frac{\partial \hat{y}(k)}{\partial W^d} = W^y P(k) \tag{8}$$

$$\frac{\partial \hat{y}(k)}{\partial W^h} = W^y_j Q(k) \tag{9}$$

where $P(k) \equiv (\partial h(k)/\partial W^d)$ and $Q(k) \equiv (\partial h(k)/\partial W^h)$ and satisfy

$$P(k) = f'(z)[h(k-1) + W^d P(k-1)], \quad P(0) = 0 \tag{10}$$

$$Q(k) = f'(z)[x(k) + W^d Q(k-1)], \quad Q(0) = 0 \tag{11}$$

The weights can now be adjusted following a gradient method, i.e., the update rule of the weights becomes

$$W(k+1) = W(k) + \eta\left(-\frac{\partial E_m}{\partial W}\right) \tag{12}$$

where η is the learning rate and $\eta = \eta^h, \eta^d, \eta^y$, respectively, for the corresponding weight matrix. Eqs. (5)–(12) define the dynamic back-propagation algorithm (DBP) for DRNN.

The update rules call for a proper choice of the learning rate η. If we let η^h, η^d and η^y be the learning rate for DRNN weights W^h, W^d and W^y, respectively, then, the DBP algorithm converges if $0 < |W^d_j| < 1, \quad j = 1, 2, \ldots, v$ and the learning rate are chosen as

$$0 < \eta^y < 2/q \tag{13}$$

$$0 < \eta^d < \frac{2}{q}\left[\frac{1}{W^y_{\max}}\right]^2 \tag{14}$$

$$0 < \eta^h < \frac{2}{(n+q)}\left[\frac{1}{W^y_{\max} \cdot x_{\max}}\right]^2 \tag{15}$$

Here $W^y_{\max} := \max_k ||W^y(k)||$, $x_{\max} := \max_k ||x(k)||$ and $||\cdot||$ is the sup-norm.

Except when the function is fairly simple, hand-coding of the derivative function is a tedious and sometimes error prone job in practical applications. And even if symbolic software is used, the resulting expression of $(\partial \hat{y}(k)/\partial W)$ in Eq. (9) is very complex and hard to work with. Automatic differentiation (AD) techniques has been employed in this study to give an accurate result; i.e., there are no truncation errors, no human errors. Given a vector function$F : \Re^n \to \Re^m$, the derivative matrix $A = F'$ can be obtained by employing the following four-step strategy:

1) Compute the sparsely pattern of $A \in \Re^{m \times n}$.
2) Compute a seed matrix $S \in \{0, 1\}^{n \times p}$ with the smallest p.
3) Compute the compressed matrix $B = AS$.
4) Recover the non-zeros of A from B.

The matrix S partitions the columns of A:

$$s_{jk} = \begin{cases} 1, & \textit{if column } a_j \textit{ belongs to group } k \\ 0, & \textit{otherwise} \end{cases} \tag{16}$$

$$\sum_{k=1}^{p} s_{jk} = 1 \quad \forall j : 1 \le j \le n$$

This approach can be realized successively by automatic differentiation using the chain rule. Moreover, the computation efficiency and advantages of using AD in the engineering applications was demonstrated in Bucker et al. (2002), the study show that the technology of AD is not only applicable to small codes but scales up to computer models consisting of thousands of lines of code. Therefore, the output gradients with respect to output, recurrent and input weights $(\partial \hat{y}(k)/\partial W)$ is computed by AD technique, which is implemented in C++ language using the software package *ADOL-C* and interfaced to *MATLAB* with a *MEX* warp.

3. MPC with DRNN model

Model predictive control applications have in the past been mostly used in the process and chemical industries with