Appeared in Proc. of International Symposium on Intelligent Control, pp. 396-401, Dearborn, Michigan, © IEEE 1996

Temporal Neuro-Control of Idle Engine Speed

Fathi M. Salam and Ammar B. Gharbi

Circuit, Systems and Artificial Neural Networks Laboratory

Michigan State University, East Lansing, MI 48824

salam@ee.msu.edu, gharbi@ee.msu.edu

Abstract

Neural network architectures are proposed to model and control the Spark-Ignition (SI) engine idle speed. Static and dynamic multi-layer neural networks are used to develop plant models and plant inverse function models. A neural network is trained to learn the system's input-output relationship. Another is trained to model the inverse relationship of the plant and is included in the controller. The developed controller, in series with appropriate filters, is then used to control the throttle angle and the spark advance angle control signals to track a desired speed and pressure of the engine. The paper demonstrates how computationally in-expensive neural networks can effectively learn on-line both the modeling and control tasks for this nonlinear, dynamical and complex process.

1 Introduction

The control of a two-stage idle Spark-Ignition (SI) engine has been studied exhaustively in the literature because its performance represents a key feature in the stability, reliability and cost in automotive manufacturing. Conventional nonlinear control schemes [1], recurrent neural networks [2,3] and fuzzy logic [4] approaches have all been considered to address this control problem. In this work, we use neural networks in a feedback control structure which permits the use of static (i.e. non-recurrent) and/or recurrent neural networks as building blocks within the overall feedback structure. It is noted on the outset that since the overall structure includes feedback, and therefore recurrent, it is not critical that recurrent neural networks are exclusively used within the neural controller building blocks. This view can be favorable when the computational effort in training and executing the controller in practice is required to be minimized in order to be implementable onto economical microprocessors. Such economic view is necessarily adopted in the automotive industry as a criterion in eventual deployment.

In this work, we use both static (i.e. feedforward, nonrecurrent) and dynamic/temporal (recurrent) neural network controllers. The temporal neural networks are contrived from feedforward networks by simple iterations, which is a trivial task for microcontrollers. The control structure also includes a filtering stage, in series, to provide (i.e., generate) the necessary state error signals to enable the task of idle engine speed control.

The paper is organized as follows. The problem is defined in section 2. Section 3 presents some review of the basics of feedforward neural networks which will be used to develop neural controllers. In section 4, control structures using static neural network controllers as well as static modeling of the plant are used to accomplish the control task. Simulation results are then presented to show the performance of the proposed structure. A desired level of output signals is tracked robustly (and in a stable manner) using control signals within the permissible range. Also, temporal predictors and temporal plant models are developed as anticipatory approaches with a view towards enhancing the idle engine speed control performance. Concluding remarks are given in section 5.

2 Problem Definition

A two-stage idle engine belongs to a nonlinear class of system described by:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t, d)$$

where **x** is the state vector, **u** is the control signal vector, *d* is the system disturbance and *f* is a nonlinear function. In [5], Powell et al. have developed a mathematical model of the system based on steady-state data map described by the following [2,3,4]

$$\dot{P} = k_N(\dot{m}_{ai} - \dot{m}_{ao}), \quad k_P = 42.40$$
$$\dot{N} = k_N(T_i - T_L), \quad k_N = 54.26$$
$$\dot{m}_{ai} = (1 + 0.9070\theta + 0.0998\theta^2)g(P)$$
$$\dot{m}_{ao} = -0.0005968N - 0.1336P + 0.0005341NP + 0.000001757NP^2$$
$$m_{ao} = \dot{m}_{ao}(t - \tau)/(120N), \quad \tau = 45/N$$
$$T_i = 39.22 + 325024m_{ao} - 0.0112\delta^2 + 0.635\delta + (0.0216 + 0.000675\delta)N(2\pi/60) - 0.000102N^2(2\pi/60)^2$$
$$T_L = (N/263.17)^2 + T_d$$
$$g(P) = \begin{cases} 0 & P > 101.325 \\ 1 & P < 50.6625 \\ 0.0197\sqrt{101.325P - P^2} & \text{Otherwise} \end{cases}$$

where the control signals are the throttle angle θ and the spark advance angle δ, the output signals are the manifold pressure P and the engine speed N, T_d is accessory torque disturbance, $\dot{m}_{ai}$ is the mass flow rate into the manifold, $\dot{m}_{ao}$ is the mass flow rate out from the manifold into the cylinder, T_i is the internally developed torque and T_L is the load torque. It should be noted that this model was used in [3,4] with successful results. We note also that such complex model can only be an approximation to the true model under very restrictive assumptions. The physical system in fact is distributed and includes many unmodeled dynamics. Moreover, it is time and temperature dependent to say the least. Uncertainties in such physical system are abound. Figure 1 below presents a schematic diagram of the system.

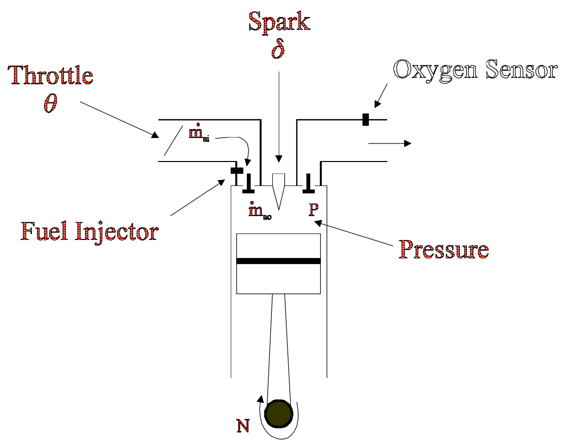


Fig. 1: Idle Engine Speed System

The goal of the work is to develop neural network controllers that would produce the necessary control inputs of throttle and spark advance angles in order to track a desirable setting of engine speed and pressure. Moreover, for the application at hand, the controller must be computationally inexpensive and realizable on standard economical microcontrollers.

3 Neural Networks

Neural networks [6] can be used to design models that represent input/output relationships. The main building component of a neural network is the *neuron*. Each neuron has a number of parameters called *weights* and *biases*. The interactions of the individual neurons and the change of the values of these parameters are what defines the functionality of the whole network. For simplicity the neuron is primarily linear with a single sigmoidal nonlinearity. A network is easily constructed to any order of complexity depending on the problem at hand. The goal is then to choose the weights and biases of the network to achieve the desired static input/output relationship. More details and variations are available in standard textbook such as the one referenced here [6].

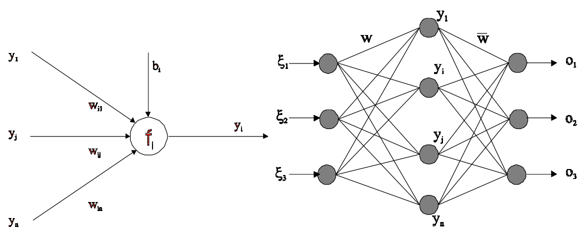


Fig. 2: Feedforward Neural Networks

Figure 2 shows the architecture of such network. The activity of each neuron is defined by:

$$o_i = f_i(\sum_{j=1}^{n} w_{ij} y_j + b_i)$$

where o_j is the j^{th} neuron's output, b_j is the j^{th} neuron's bias, w_{ij} is the weight connecting the j^{th} neuron's output to the i^{th} neuron's input and f_i is the i^{th} neuron's activation function.

A neural network is constructed by connecting a set of neurons to each other. Most commonly used architecture to design such neural networks is the multilayer feedforward architecture where the neurons are lined up in columns (or layers) and every neuron within a layer is connected only to the neurons from the previous layers.

The parameters (the weights and biases) are then updated using the gradient descent method [6]:

$$\Delta \overline{w}_{ij} = \eta \sum_{\mu} \overline{\delta}_i^{\mu} y_j^{\mu} \text{ and } \Delta \overline{b}_i = \eta \sum_{\mu} \overline{\delta}_i^{\mu}$$

$$\Delta w_{ij} = \eta \sum_{\mu} \delta_i^{\mu} \xi_j \text{ and } \Delta b_i = \eta \sum_{\mu} \delta_i^{\mu}$$

$$\text{where} \quad \overline{\delta}_i^{\mu} = (o_i^{\mu} - \zeta_i^{\mu}) f'(\overline{h}_i^{\mu})$$

$$\delta_i^{\mu} = f'(h_i^{\mu}) \sum_k \overline{\delta}_k^{\mu} \overline{w}_{ki}$$

and η is the learning rate. The above learning rule is commonly known in the literature as the standard back-propagation learning. The back-propagation learning algorithm is based on the (discrete) gradient descent method which can be very slow if η is small and may oscillate or diverge if η is too large. Possible ways of dealing with this problem are the use adaptive learning rates and the inclusion of the momentum term [6]. Other more sophisticated techniques include the use of more appropriate energy functions and structure.

4 The Control Structure

In the context of this paper, the idle engine speed control is to provide *robustly* the correct throttle and spark advance inputs angles to drive the engine to the desired engine speed and pressure outputs. The objective is then to train a neural network controller to produce the correct input to *stably* drive the plant to the desired output. A number of different approaches for training a controller have been described in the literature [7,8] such as the reinforcement learning. In this work, we use the generic control structure shown in Figure 3 below.

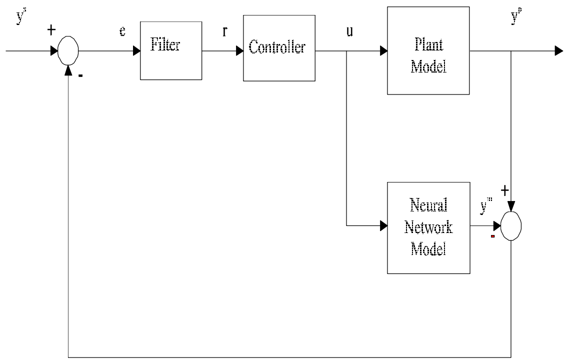
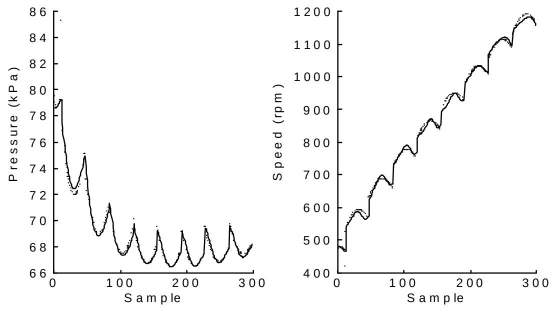


Fig. 3: Control Structure

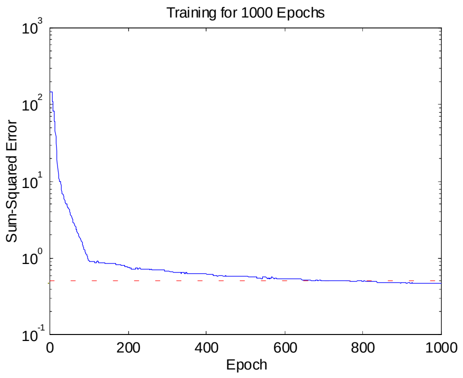
In [8], the authors describe a control structure such as the one shown in figure 3 to control a nonlinear plant P. They proposed a three step-procedure to carry out this task: *(i)* design a neural network model, M, that represents the plant, *(ii)* design a neural network model, C, that represents the inverse functionality of the plant, and *(iii)* use the structure in Fig. 3 with an appropriate choice of the filter, F, to control the plant. This control structure will be used to observe the plant performance for the cases of static and temporal modeling of the controller. It should be repeated that since the overall control structure includes feedback, and therefore recurrence, it is not critical that each neural building block be recurrent. Note that eliminating redundant recurrent loops translates to computational savings (and hence cost) which is more critical for the automotive industry.

4.1 Using Static Controllers

Using the mathematical model developed in [5], a set of data was obtained to represent the input-output relationship of the idle engine speed system. The collected data represent a training set and was used to develop static neural network models and controllers of the system. Simulations of the performance of plant model and the corresponding training errors as a function of the number of epochs are shown in Figure 4.



(a) Training performance



(b) Training error

Fig. 4: Model Performance

A filter F to achieve the control of the plant using the structure described in Figure 3 was designed. A second order filter to adjust the gain in series with an integrator to minimize the backpropagation error were used. The filter is of the form:

$$F(s) = K_p \frac{w_n^2}{s^2 + 2\zeta w_n s + w_n^2} + \frac{K_e}{s}$$

With an appropriate choice of parameters, the output of the plant will match a desired reference that is set by the user. By considering $K_p = 1.1$, $w_n = 1000$, $\zeta = 0.7$ and $K_e = 500$, the closed-loop system is able to track the set reference. Figures 5 and 6 represent the performance of the controller for a sample of the trained data. It can be observed that the plant reaches the desired target within 2msec in simulation with nominal step size of 0.5msec. In this experiment, it is desired to produce the necessary control signal vector u that will drive the plant to the following reference (0.0360, -0.5351). This normalized data corresponds to pressure of 76.2239kPa and an engine speed of 545.6933 rpm. As it is observed in Figures 5 and 6, the plant has tracked these reference signals within 2msec. A (normalized) control signal of (-0.5574, 0.5994) was necessary to accomplish this task. When these values are converted to their physical domain, they correspond to the following angles (10.2130, 40.6122). This pattern is not included in the training or testing data sets which were generated from the original mathematical model. This illustrates that the designed control has the capacity to generalize (i.e., interpolate) to other input/output patterns.

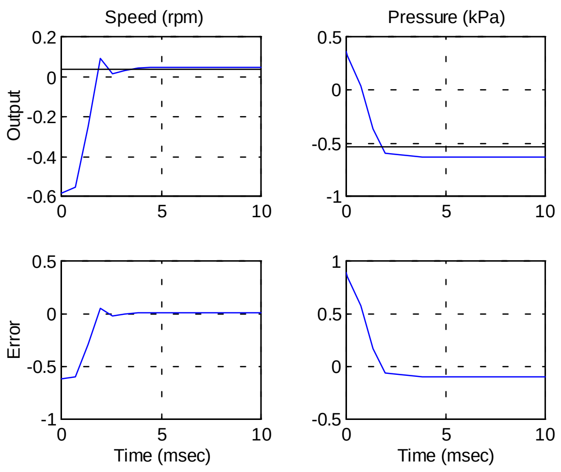


Fig. 5: Output Tracking

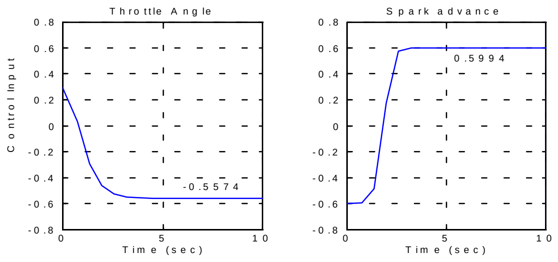


Fig. 6: Control signals

4.2 Using Temporal Controller

Here, a temporal (i.e. recurrent) neural network architecture, as depicted in Figure 7(b), was also used to develop a controller. Successful training of such controller was completed and is demonstrated in Figure 8.

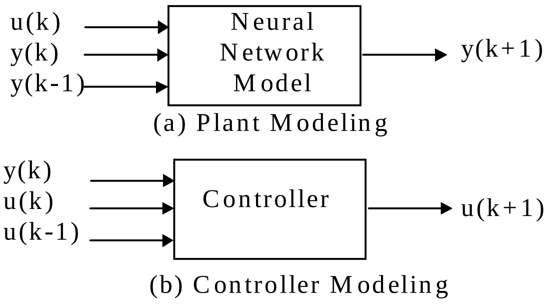


Fig. 7: Temporal Neural Network Modeling

Using the control structure shown in Figure 3 along with the temporal neural network controller, the idle engine speed tracking was achieved as shown in Figure 9. The filter F was selected to have the following parameters $K_p = 1.5$, $w_n = 1000$, $\zeta = 0.7$ and $K_e = 5000$.

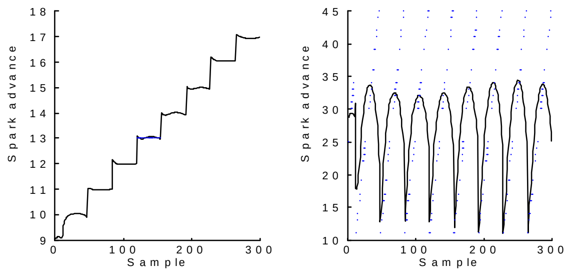


Fig. 8: Temporal Controller Training Performance

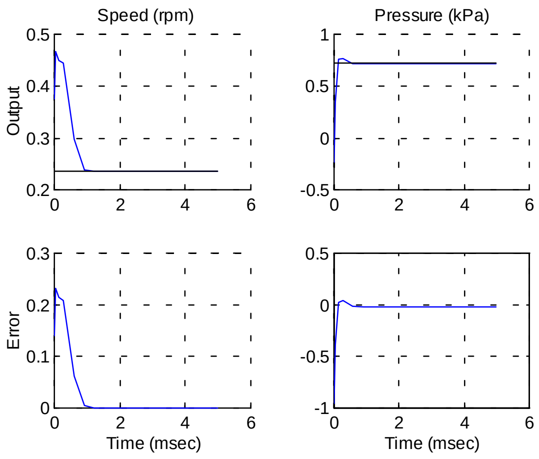


Fig. 9: Temporal Control Performance

The architecture shown in Figure 7(a) can be used to design a neural network predictor which would guide the neural network controller shown in Figure 7(b) to produce an enhanced anticipatory control signals. The training of such model is shown in Figure 10. A suitable control structure can be developed to incorporate the prediction model. The control signal at time k is applied to the plant and the predictive model simultaneously. Assume that the corresponding plant output is also at time k. The predictive model, however, produces its estimate of the "next" output, say, at time k+1. The controller then uses the "anticipated" output to augment (e.g., additatively) its control action.

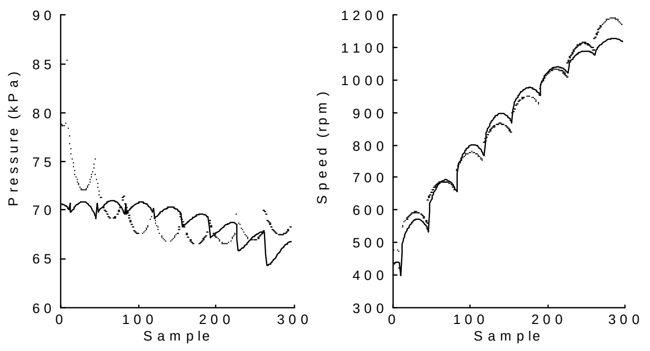


Fig. 10: Predictor Modeling

5 Conclusion

Static and temporal Neural networks were successfully employed to develop a controller for the idle engine speed control. Computer simulations show the successful output tracking and present the necessary control signals produced by the neural network controllers to accomplish the control task. Such neural networks are not necessarily required to be recurrent. Moreover, the control structure uses basic filters to contract a state error signals to enable adequate stable control.

ACKNOWLEDGMENT**:**
This work was supported in part by the Michigan Research Excellence Fund (REF).

6 References

[1] D. Cho and K. Hedrick, “A Nonlinear controller Design Method for Fuel Injected Engines”, J. Engineering for Gas Turbines and Power, vol. 110, pp. 313-320, July 1988

[2] L. A. Fedlkamp, G. V. Puskorius, L. I. Davis, Jr., and F. Yuan, “Neural Control Systems trained by Dynamic gradient Methods for Automotive Applications”, International Joint Conference on Neural Networks, 1992, vol. 2, pp. 798-804

[3] G. V. Puskorius and L. A. Feldkamp, “Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks”, IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 279-297, March 1994

[4] G. Vachtsevanos, S. S. Farinwata and D. K. Pirovolou, “Fuzzy Logic Control of an Automotive Engine”, IEEE Control Systems Magazine, vol. 13, no. 3, pp. 62-68, 1993

[5] K. Powell and J. A. Cook, “Nonlinear Low Frequency Phenomenological Engine Modeling and Analysis”, Proceedings of the 1987 American Control Conference, vol. 1, pp. 332-340, 1987

[6] N. Bose and P.Ling, "Neural Network Fundamentals with Graphs, Algorithms and Applications, McGraw-Hill, 1996

[7] Widrow, N. K. Gupta and S. Maitra, “Punish/Reward: Learning with a Critic in Adaptive Threshold Systems”, IEEE Transactions on Control Systems Magazine, , October 1989.

[8] Anderson, “Learning to Control an Inverted Pendulum Using Neural Networks”, IEEE Control Systems Magazine, vol. 9, no. 3, April 1989.

[9] K. Narendra and K. Parthsarathy, “Identification and Control of Dynamical Systems Using Neural Networks”, IEEE Transactions on Neural Networks, vol. 1, no. 1, pp. 4-27, 1990.