An adaptive PID controller with an online auto-tuning by a pretrained neural network

An adaptive PID controller with an online auto-tuning by a pretrained neural network

P A Chertovskikh[1,2]**, A V Seredkin**[1,2]**, O A Gobyzov**[1]**, A S Styuf**[1,2]**, M G Pashkevich**[3] **and M P Tokarev**[1,2]

[1]Kutateladze Institute of Thermophysics SB RAS, 1, Lavrentyev Ave., Novosibirsk, 630090, Russia
[2]Novosibirsk National Research State University, 1 Pirogov Street, Novosibirsk, 630090, Russia
[3]Novosibirsk State University of Economics and Management, 52/1 Kamenskaya Street, Novosibirsk, 630099, Russia

E-mail: mtokarev@itp.nsc.ru

Abstract. This paper describes an intelligent adaptive PID controller design procedure. The controller consists of a discrete time PID and an auto-tuning neural network unit. First system identification with a nonlinear autoregressive model (NARX) was performed. This model was then used to train the neural PID tuner. A special MATLAB toolbox "SmatPID Toolbox" was developed to automate the process of controller synthesis. The resulting controller was tested in a laboratory coal-gas furnace control system to track specified air flow rates.

1. Introduction

Application of adaptive control is a very potent direction in emerging technologies. Examples include heat engineering equipment [1], turbulence control for drag reduction [2], nonlinear servomechanism control [3], chemical reactor [4] and even human glucose-regulatory system control [5]. Since processes underlying of these systems exhibit nonlinear behavior and parameter drift a proper adaptive controller design is paramount to achieve efficiency and reliability.

PID controller is the most popular controller design in all industries. It has several important properties: feedback, integral and derivative action. However, being linear it is not suited for nonlinear systems in general case.

Gain scheduling is one of the most popular nonlinear controller design. It yields excellent results for slowly time-varying parameter systems. For example in [6] a gain-scheduled PID controller is proposed. However, the problem of robust stability of arbitrary time-varying systems is proved to be NP-hard [7]. On the other hand, classic gain-scheduled approach uses gains obtained from analysis of linearized plant which may render controller suboptimal in case of large operation regions.

Fuzzy control is often considered a modern alternative to PID. The main advantages of fuzzy control are that it is easy to add nonlinearities, logic and input signals. Fuzzy systems can be also used for nonlinear system identification [8]. However, fuzzy system design is either based on heuristic information or revolves around training/optimization.

Neural networks (NN) are one of the most popular field of study nowadays. From the perspective of control engineering, neural networks should be viewed as tunable nonlinearities. Neural networks are great for nonlinear system identification since even simple networks are proved to be able to estimate a vast variety of functions [9]. While fuzzy systems are tuned by changing rules and membership functions neural networks are tuned by adjusting weights which makes neural networks more suitable for black-box modeling which is more general approach for system identification. Neural networks can also be used to design controllers, for example, in [10] recurrent neural network control is provided.

Combining techniques mentioned above yields more controller designs. Fuzzy gain scheduling is a common technique in modern control engineering [8]. In [11] PID controllers are scheduled by feedforward neural network. In [3] and [12] recurrent neural networks are used to tune PID gains.

This paper reports on development of an adaptive type PID controller for control of an air supply channel of a coal-gas furnace. In the second paragraph methods used for development and analysis of the controller are described. The third paragraph presents results of testing the controller in a real laboratory coal-gas furnace. The last section summarizes the results.

2. Methods

A block-scheme of the used adaptive controller with a controllable system or plant is shown in Fig. 1. The controller itself consists of two main parts: a feedforward neural network (NN) unit used for auto-tuning and a PID controller block in the parallel form. Here we consider only single-input single-output (SISO) plants. The auto-tuning NN unit gets two inputs as control error e and sensor data y and calculates current PID coefficients at every control cycle. Training of the NN is done once at the design stage and then the NN model remains unchanged during the operation. The architecture of the NN model is also fixed before the training. In order to save time, training is used in a virtual environment by the model of the plant, which is obtained with system identification.

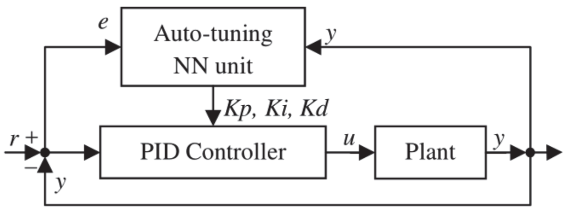


Figure 1. Block-scheme of the adaptive PID controller.

In this work we present a general procedure for the design of an adaptive PID SISO controller. The first and the last phase of the procedure is performed on a target system. The intermediate phase is done on the virtual environment. The general sequence is following:

1) Initial phase:
    a) Recording a system response to a test signal.
2) Intermediate simulation phase:
    a) System identification of a plant model.
    b) Setting up the PID coefficients for initialization of the auto-tuning NN unit.
    c) Training of the PID auto-tuning NN unit.
3) Final phase:
    a) Operation of the designed controller on the target system after deployment.

In the next subsections the above phases will be described in more details.

2.1. Recording of a system response

To perform system identification of the plant model at first it is necessary to record a system response to a known input control signal u. The test signal should be within the range for plant's input and have a broad frequency range to emulate possible transient processes. Here we used amplitude modulated pseudo-random binary signal (APRBS) [13], which satisfies the mentioned requirements. Additional benefit of the selected form of the test signal is that it can be used for characterization of nonlinear plants due to amplitude modulation of the signal.

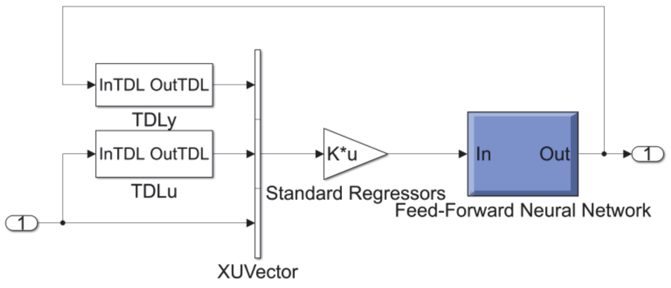
2.2. *Simulation phase*

The next phase is the most computationally intensive, and it is done separately from the target plant and the controller. Below all examples will be illustrated with the help of objects from MATLAB software package. For system identification a nonlinear autoregressive model (NARX) of the plant was utilized, which in general case is described as:

$$y(t) = h(y(t-1), y(t-2), \ldots, y(t-p), u(t), u(t-1), \ldots, u(t-k)) + e(t), \tag{1}$$

where h is a nonlinear function and e represents noise.

A recurrent behavior of the model and also an effect of memorizing the states is provided by delay lines. Fig. 2 shows the block diagram of the NARX model used. It consists of delay units of the input signal (TDL – tapped delay line) TDLu and feedback signal TDLy, two blocks that commutate signals from the delay lines to the input of a neural network called XUVector and Standard Regressors, as well as the model's nonlinear unit provided by the neural network (see Fig. 2-b)

(a)


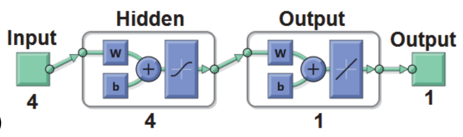
(b)


Figure 2. Block diagram of the NARX model plant (a) and nonlinearity of the plant (b).

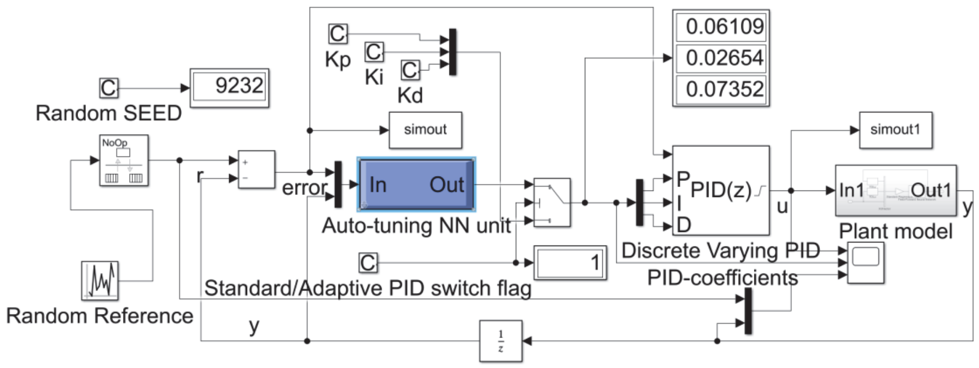


Figure 3. Block diagram of the dynamic system with the adaptive PID controller for control simulations.

In this work we used feedforward type NN with one or two hidden layers and their sizes optimized by a grid search. The best plant model is defined according to its highest normalized root mean square (NRMSE) metrics. Nonlinearity was realized by sigmoid activation functions in the hidden layers of the NN.

The next step b of the simulation phase is the tuning of the PID coefficients for initialization of the auto-tuning NN unit. During training of the auto-tuning NN unit a gradient descent optimization of the model coefficients is used. This implies an appropriate starting point, from which it is possible to get closer to the extremum.

For tuning of the PID coefficients the same cost function as during the auto-tuning NN unit was used. According to our tests a combined cost function in the form of: $J = u(t), 0 \le t \le t_c; e(t), t > t_c$ was utilized where $t = 0$ is the time of change of a setpoint. It allowed us to achieve a smooth approximation of the

NARX시스템모델의 non-linearity에 대응하는
adaptive non-linear PID controller출력값을 initialization값으로 해서,
control error를 줄이는 방향으로 optimization(GD+Levenberg-Marquardt)

controlled parameter y to a given setpoint value r at a suitable convergence time t_c in case of gradient descent optimization. Therefore in the beginning of the setpoint change, the optimization limits the value of the control signal and then it switches to minimize the control error. In case of using only the control error for the cost function the controller parameters are found such that $y(t)$ approaches a setpoint r fast with an overshot and periodic damped oscillations.

For the control system simulation we used a parameterized dynamic system with a Simulink model shown in Fig. 3. As it can be seen, depending on the PID switch flag, the model uses either the standard linear PID controller or adaptive nonlinear PID controller. At the step b of the procedure coefficients of the linear PID are found and the auto-tuning NN unit is pretrained such that for the typical input (e, r) it outputs three PID coefficients (Kp, Ki, Kd) close to that were found.

Fig. 4 shows the structure of the auto-tuning NN unit. It is a feedforward NN with only one hidden layer, which by its three neurons provides the nonlinear behavior of the unit. In total the unit has 21 tunable parameters.

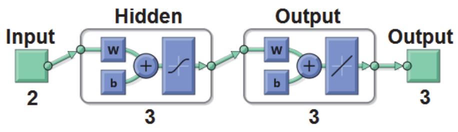


Figure 4. Architecture of the neural network used for the PID auto-tuning unit.

Now we come up to next step c of the simulation phase that includes training of the auto-tuning NN model with similar procedure described for the step b. The model parameters were optimized by the Levenberg-Marquardt algorithm for the same r signal kept unchanged until a gradient descent step is done at the current iteration of the optimization. Then the setpoint history r is randomly changed to a new one. The Jacobian of the described cost function $J(\boldsymbol{\theta})$ was assessed numerically using forward difference scheme to decrease the number of the simulations. In total it took to run n+1 simulations of the dynamic system to obtain the Jacobian of the cost function with n free parameters for only one point $\boldsymbol{\theta}$ by this scheme. It is also important that every simulation finishes fast enough in order to complete the training in a reasonable time.

2.3. *Operation of the designed controller on a target system*

As an example of a target system we used a coal-gas furnace controlled by a home-made SCADA called Inferno developed for research purposes. A discrete time PID controller using the incremental algorithm [13] was integrated into the software. PID gains are adjusted by PID tuner NN on each program cycle.

2.4. *Implementation notes*

The simulation phase was implemented as a MATLAB toolbox called “SmatPID Toolbox”. It can be downloaded by this link https://github.com/mikhailpt/smartpid-toolbox-matlab. During the training of the auto-tuning NN unit the next parameters of the Simulink model that realized the control cost function were used: discrete solver ode3 (Bogacki-Shampine) with fixed timestep and period of 300 sec. Stopping criteria for the nonlinear optimization were set as 10^{-12} for step tolerance and 10^{-6} for function tolerance. In order to save time for simulation the automatically generated NN unit of the plant model was changed into built-in Simulink blocks for neural networks. The auto-tuning NN unit was also constructed from the same built-in NN blocks. Additionally the fast restart mode was enabled in order to skip the Simulink model recompilation for every simulation run. All these measures helped to achieve reasonable simulation time close to 0.8 sec and several minutes for training in total. As the Inferno software was written on Python interfacing between Python code of the SCADA and the MATLAB realization of the auto-tuning NN unit was done via MATLAB Engine API for Python binding library.

3. Results

The described above adaptive PID controller design procedure was tested in a setup that included a control of an air supply system of a laboratory coal-gas furnace with a maximum thermal power of 50 kW. The setup with the laboratory scale burner was described in [14].

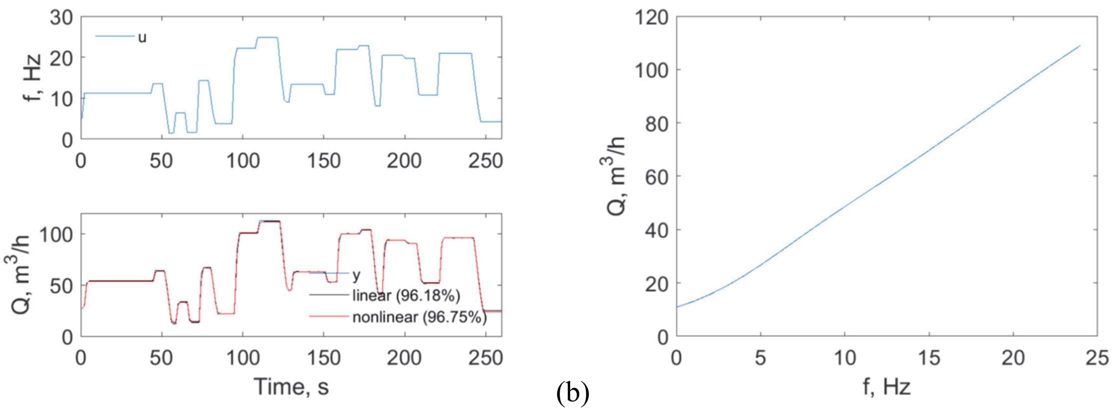


Figure 5. System identification for an air supply system of a coal-gas furnace (a) and dependence of the stationary air flow rate on the frequency setpoint for the nonlinear model (b).

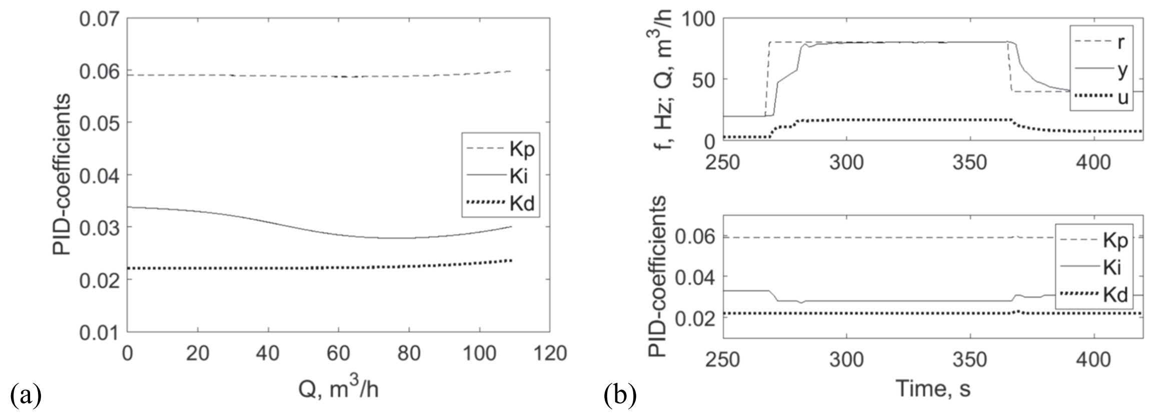


Figure 6. Variation of the PID-coefficients depending on the air flow rate at the stationary state (a), a control sequence of the developed adaptive PID controller in an air supply line of a laboratory coal-gas furnace (b).

The test signal with the duration of 1000 sec and its response were first recorded for the air supply line. Then the system identification was done and two examples of responses of a linear and a nonlinear plant model to a validation part of the signal are shown in Fig. 5-a. The first model was a transfer function of the third order and the second model was NARX model defined by its parameters (*na*, *nb*, *nk*) as (2, 2, 1) and the NN unit with one hidden layer including four neurons. Both models worked relatively well with close NRMSE values higher than 96%, while having a simple structure. If we look at the dependence of the stationary input and output parameters of the nonlinear plant model in Fig. 5-b, it explains why the error is low even for the linear model. The dependence is almost linear except for the region of the lower frequencies.

Tuning of the PID coefficients for initialization of the auto-tuning NN unit was done in 16 iterations with obtained values for (*Kp*, *Ki*, *Kd*) as (0.06, 0.04, 0.02). These values were used to pretrain the auto-tuning NN unit with a random noise additive by the amount of the possible signal variation at the input and output. Final training on simulated random control sequences after 76 simulations gave a dependence of the PID-coefficients on flow rate values for a steady state shown in Fig. 6-a. In the next step the auto-tuning NN unit was tested to control flow rate in the laboratory coal-gas furnace with cycle time 2 sec. The time-history of a control sequence for changing flow rate from 20 to 80 and then back to 40 m^3/h similar to those used during the optimization is presented in Fig. 6-b. It can be seen that during operation of the real equipment the PID-coefficients were adjusted according to the current setpoint and the control error change. The highest variation approximately 0.01 had the integral coefficient as it was expected. For the future tests, it is planned to apply the technique to synthesize an adaptive PID controller for control of a process with higher nonlinearity, for example, temperature control inside seven muffled sections of the furnace depending on the air flow rate.

Conclusion

This article reports on a method for designing of an adaptive PID controller that uses a preliminary trained neural network for an online auto-tuning of PID-coefficients. The obtained controller can be used for trajectory tracking control of linear and nonlinear processes. The method was implemented as a MATLAB toolbox which is available publicly in the Internet. Using this toolbox the adaptive PID controller of an air supply channel for the laboratory coal-gas furnace was designed and tested. According to the obtained results the maximum observed variation among the PID-coefficients within the working operating range was for integral gain. The considered dynamical system appeared to be a weakly nonlinear system with nonlinearity in the beginning of the working range at low air flow rates.

Acknowledgements

The work was supported by RSF (grant № 19-79-30075) via IT SB RAS.

References

[1] Skorospeshkin M V, Sukhodoev M S, Skorospeshkin V N and Rymashevskiy P O 2017 *J. Phys.: Conf. Ser.* **803** 012153
[2] Changhoon L, Kim J, Babcock D and Goodman R 1997 *Physics of Fluids* **9** 1740–7
[3] Jafari R and Dhaouadi R 2011 *Advances in reinforcement learning A Mellouk (InTech)* 275–96
[4] Dostál P, Kubalčík M, Bobál V and Vojtěšek J 2011 *9th Mediterranean Conf. on Control & Automation* (Corfu: Greece) 600–5
[5] Pagurek B, Riordon J S and Mahmoud S 1972 *Med. & Biol. Eng.* **10** 752–61
[6] Donyanavard B, Rahmani A M, Muck T, Moazemmi K, and Dutt N 2018 *Automation and Test in Europe Conf. Exhibition* 2018 (DATE'18) 921–4
[7] Toker O 1997 *Automatica* **33** 2015–7
[8] Passino K M and Yurkovich S 1998 *Fuzzy Control* ed T Hyde et al. (Addison Wesley Longman, Inc) p 502
[9] Hornik K 1989 *Neural Networks* **2** 359–66
[10] Chih-Hong L 2013 *Mathematical Problems in Engineering* 753756
[11] Henriques J, Gil P, Cardoso A and Dourado A 2002 *Proc. of the 2002 Int. Joint Conf. on Neural Networks* (Honolulu, USA) **1** 311–16
[12] Reichensdörfer E, Günther J and Diepold K 2017 *Technical report Department of Electrical and Computer Engineering Technical University of Munich* 84
[13] Åström K J and Hägglund T 1995 *PID controllers: theory, design, and tuning* (Research Triangle Park N.C.: International Society for Measurement and Control) 343
[14] Gorelikov E U, Litvinov I V, Kulikov D V, Rahmanov V V, Shtork S I 2018 *AIP Conference Proceedings* AIP Publishing **2027** 040055