Towards Explicit PID Control Tuning using Machine Learning

Andrew Zulu, *member, IEEE*

Department of Mechanical and Marine Engineering
Namibia University of Science and Technology
Windhoek, Namibia
azulu@nust.na

***Abstract*—The PID control algorithm is the most used industrial control method owing to its simplicity and ease of use. However, tuning PID parameters is not trivial and many methods have been reported in literature. This paper seeks to show a machine learning approach using multivariate regression with gradient descent and the normal equation. The first order cruise control system is used as an example and results show good progress towards automatic PID tuning using learned data.**

***Keywords*—PID control, machine learning, regression, gradient descent, normal equation.**

I. Introduction

Control theory has developed from its rudimentary stages during the 1920's to where it stands now. This has mainly been aided by mathematical rigour, computational power from modern fast computers and software algorithms such as Matlab. Concepts of negative feedback and the proportional-integral-derivative (PID) controller were at the heart of early development of the subject. PID controllers originated from the 19th century design of speed governors for steam engines. However, the Russian marine engineer Nicholas Minorsky is credited with writing the first paper on PID control in 1922 [1].

PID controllers are the most pervasive in industrial control systems for the simple reason that their structure is easy and they are also can easily be implemented. However, tuning of PID parameter gains is generally not an easy task. In literature, several methods and algorithms exist (see [2], [3], [4], [5], [6], [7], [8]) to eliminate trial and error but the holy grail of tuning has not been reached.

Most PID tuning methods in literature are hueristic and do not guarantee optimal performance. Astrom and Hagglund propose automatic tuning by varying phase and amplitude margins [2]. Ho *et al* elaborated a frequency domain approach for self-tuning [4]. In some papers, optimal tuning is reported such as in [5] and an extremum seeking approach in [9]. A PD fault-tolerant control for dynamic positioning of ships using Lyapunov theory is found in [10].

A popular textbook tuning method is the Ziegler-Nichols [11], whose algorithm is easy but winding. It works by adjusting the proportional gain first until the system becomes unstable, scaling back by 20% and then adjusting the integral and derivative terms (where necessary) by a proposed scheme. This is reported not to work well especially for process plants where stability issues can not be allowed to be compromised.

With fast computers at hand, very complex tuning methods have been implemented. The *pidtool* or *pidtune* tools in Matlab performs such a function but they rely on linearizing the plant model and is in some cases reported to determine negative PID values. An interesting recent tool, confined to the quadrotor hobbyist community, is called *BlackBox*. It shows a visual representation of the tuning situation applied on a quadrotor. It can be seen in action on Youtube in [12]. In addition, artificial intelligence in general and machine learning in particular are now used by researchers to implement PID tuning [13].

In this paper, we present a machine learning approach that used both gradient descent and the normal equation for modeling PID parameters for a cruise controller. The rest of the paper is divided as follows: Part II gives a brief of the PID controller. Part III discusses the linear regression model, including the normal equation and the alternative gradient descent algorithm. Part IV elaborates how the model was trained and provides sample simulation results. Part V discusses the results and ideas and Part VI concludes the paper.

II. The PID Controller

Figure 1 shows a schematic of a PID controller as applied to a cruise control system. What is beautiful about PID controllers is that the P term accounts for the present error value; I term accounts for the past (accumulated) error

978-1-5386-2775-4/17/$31.00 ©2017 IEEE

values; while the D term accounts for possible future error values. It's mathematical form is given as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \quad (1)$$

where *u(t)* is the control input, *e(t)* is the error, and K_p, K_i and K_d are the parameters to be tuned.

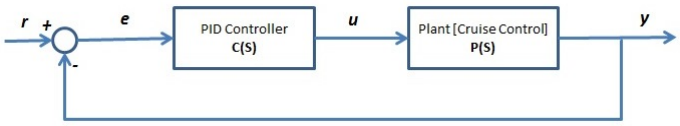

Figure 1: Schematic for PID feedback cruise control.

III. Plant Linear Regression Modeling

The system modelled is a car cruise control system, which can be modelled a first order mass-damper system with the following differential equation:

$$m\frac{dv}{dt} + cv = u \quad (2)$$

where m is the mass of the vehicle, v is the velocity of the vehicle (output), c is the damping coefficient and u is the control input.

The open loop transfer function with the output as the speed of the vehicle is given by:

$$P(s) = \frac{V(s)}{U(s)} = \frac{1}{ms + c} \quad (3)$$

The closed loop transfer function with the PID controller applied as in Figure 1 is given by:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{K_d s^3 + K_p s + K_i}{(m + K_d)\, s^2 + (b + K_p)\, s + K_i} \quad (4)$$

We apply linear regression that relates three input variables to one output variable. For supervised learning, the linear regression model is given by [14]:

$$h_0(X) = \theta^T x = \sum_{i=0}^{n} \theta_i x_i \quad (5)$$

where θ_is are the weights assigned to input variables X_is and $h_0(X)$ is the hypothesis output variables.

The *gradient descent* algorithm minimizes the error between the hypothesis output variable and the actual output y as given by the cost function $J(\theta)$:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \quad (6)$$

where m is the number of training examples.

To minimize the cost function $J(\theta)$, the gradient descent update rule is given by:

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(X)^{(i)} - y^{(i)}) X_j^{(i)} \quad (7)$$

where α is the learning rate.

However, when the number of features is not too large and not many iterations are required for learning, the *normal equation* can be used. The weights can be calculated explicitly using the pseudo-inverse of the features as given by this equation:

$$\theta = \left(X^T X\right)^{-1} X^T y \quad (8)$$

IV. Model Training and Simulation Results

A learning algorithm is required to train the cruise controller to be able to perform repeated or similar tasks. Hence, obtaining large quantities of accurate data is key to the success of this training.

A. Model training

The model is trained using desired input variables to obtain output variables. The input variables are chosen as the desired step variables, which are the rise time (t_r), settling time (t_s), and overshoot (e_{os}) while the output variables are the PID parameters. For the three output variables, this can be written as:

$$\begin{aligned} K_p &= t_r\theta_1 + t_s\theta_2 + e_{os}\theta_3 \\ K_i &= t_r\theta_4 + t_s\theta_5 + e_{os}\theta_6 \\ K_d &= t_r\theta_7 + t_s\theta_8 + e_{os}\theta_9 \end{aligned} \quad (9)$$

where K_p, K_i, and K_d are the proportional, integral and derivative gains respectively; θ_1/θ_4/θ_7; θ_2/θ_5/θ_8, and θ_3/θ_6/θ_9 are the weights for K_p; K_i; and K_d for the rise time, settling time, and overshoot respectively.

B. Results

Various step variables are obtained for various combinations of PID parameters. The following input variables as in [15] are used:

mass of the car = 1000 kg;
damping coefficient = 50 Ns/m;
reference speed = 10 m/s;
rise time $<$ 5 s;
settling time $<$ 10 s;
overshoot $<$ 5%;

The system response is known to react quickly to K_p and slowly to K_i and almost never to K_d, which case a PI

controller would be sufficient. Consequently, K_p is varied quickly while K_i and K_d are varied slowly. An algorithm was generated that varied K_p values from 1 exponentially to 10,000 in steps of 10; and K_i and K_d values were each varied linearly from 0 to 100 in steps of 1. This generated over a million training values. Table I shows a sample of representative training results.

The gradient descent algorithm was applied on the training examples and produced numerous results for analysis. However, due to the immense variability of input data, gradient descent did not produce a global minimum and predicted hypothesis function outputs were inconclusive. The normal equation approach produced consistent results despite the wide variability in the inputs and outputs. Table II shows a sample of training values that meet the design step information criteria.

TABLE I
SAMPLE TRAINING RESULTS - VARYING K_i AND K_d AT K_p = 1

K_p	K_i	K_d	t_r	t_s	e_{os}
1	0	0	43.0786	76.7073	0
	0	25	44.1555	78.625	24.3902
	0	50	45.2325	80.5427	142.8571
	0	75	46.3095	82.4604	255.814
	0	100	47.3864	84.3781	363.6364
	25	0	7.4881	145.841	59.8091
	25	25	7.5731	148.0102	58.7653
	25	50	7.6563	150.0843	57.7045
	25	75	7.7377	152.1089	56.6295
	25	100	7.8174	168.9083	55.5427
	50	0	5.0968	144.81	69.7161
	50	25	5.157	146.8172	68.3141
	50	50	5.2159	160.5161	66.9698
	50	75	5.2737	163.2004	65.6795
	50	100	5.3303	165.4612	64.4399

Using the normal equation and for a selected results range that meets the design criteria, the design matrix X was obtained as:

$$\begin{pmatrix} 1 & 2.377 & 8.568 & 0 \\ 1 & 2.199 & 3.916 & 0 \\ 1 & 1.991 & 9.966 & 0 \\ 1 & 0.2234 & 0.4176 & 0 \\ 1 & 0.223 & 0.4147 & 0 \end{pmatrix}$$

For each of the parameters K_p, K_p and K_p as vectorized output y, the weighting factors are obtained as follows:

$$\begin{aligned} \theta_i[K_p] &= \begin{pmatrix} 5020 & -93.65 & -36.20 & 3.812 \end{pmatrix}^T \\ \theta_i[K_i] &= \begin{pmatrix} 27.284 & -6.364 & 5.378 & 0 \end{pmatrix}^T \\ \theta_i[K_d] &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T \end{aligned} \quad (10)$$

This brings the explicit form of the parameter equations to the following:

$$\begin{aligned} K_p &= 5020 - 93.65t_r - 36.2t_s + 3.812e_{os} \\ K_i &= 27.2838 - 6.3639t_r + 5.3781t_s \\ K_d &= 1.00 \end{aligned} \quad (11)$$

The step response as obtained by the calculated PID parameters is shown in Figure 2.

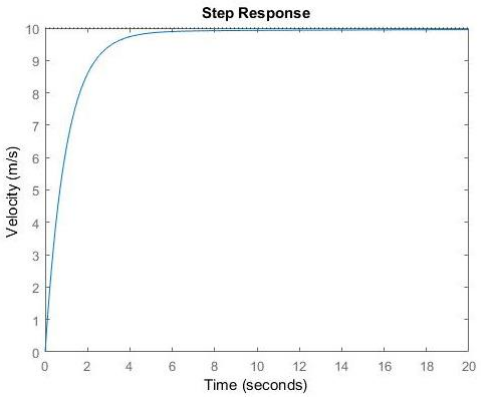

Figure 2: System step response.

V. DISCUSSION

The data-driven (machine learning) approach is becoming popular and growing into a formidable method in control system design. This papers shows how the approach can be used to drive PID tuning parameters through data.

Where the system becomes complex, gradient descent and normal equation forms are not sufficient for analysis deep learning take over from there as they represent a more accurate brain-like way of data transmission with back propagation error-correcting strategies. The general idea is to train a model to gather enough learning data with the hope to self tune PID parameters using an fast looping algorithm.

This work would be more enhanced with a deep learning algorithm that would predict PID parameter values with more input features in the step information not analysed in this work like steady-state error, peak value and peak time. Ultimately, machines and process systems will be able to learn to do PID self-tuning without much human intervention.

VI. CONCLUSION

This paper presents a first step in the automatic tuning of PID parameters using learned data from a regression model. This approach can be extended to process plants with the availability of data and electro-mechanical systems that generate sufficient data for supervised learning. The approach is not suitable for complex non-linear systems for which PID is unsuitable. However, non-linear systems can

also be trained. The scope for future work includes training on deep neural networks and designing a PID controller that guarantees optimal performance.

ACKNOWLEDGMENT

The author would like to thank the Namibia University of Science and Technology (NUST) for their financial support towards attending IEEE AFRICON 2017.

REFERENCES

[1] N. Minorsky, "Directional stability of automatically steered bodies," *J. Amer. Soc. Navel Engineers*, vol. 42, no. 2, pp. 280–309, 1922.

[2] K. J. Åström and T. Hägglund, *Automatic tuning of PID controllers*. Instrument Society of America (ISA), 1988.

[3] H. Rasmussen, "Automatic tuning of pid-regulators," *Aalborg University, Dept. of Control Engineering*, 2002.

[4] W. K. Ho, O. Gan, E. B. Tay, and E. Ang, "Performance and gain and phase margins of well-known pid tuning formulas," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 4, pp. 473–477, 1996.

[5] M. Zhuang and D. Atherton, "Automatic tuning of optimum pid controllers," in *IEE Proceedings D-Control Theory and Applications*, vol. 140, no. 3. IET, 1993, pp. 216–224.

[6] S. Skogestad, "Probably the best simple pid tuning rules in the world," in *AIChE Annual Meeting, Reno, Nevada*, 2001.

[7] M. Shahrokhi and A. Zomorrodi, "Comparison of pid controller tuning methods," *Department of Chemical & Petroleum Engineering Sharif University of Technology*, 2013.

[8] F. Haugen, "The good gain method for pi (d) controller tuning," *Tech Teach*, pp. 1–7, 2010.

[9] N. J. Killingsworth and M. Krstic, "Pid tuning using extremum seeking: online, model-free performance optimization," *IEEE control systems*, vol. 26, no. 1, pp. 70–79, 2006.

[10] Y. Su, C. Zheng, and P. Mercorelli, "Nonlinear pd fault-tolerant control for dynamic positioning of ships with actuator constraints," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1132–1142, 2017.

[11] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[12] J. Bardwell, "The best tuned quadcopter ... in the world," *https://www.youtube.com/watch?v=m0f1tPHxfOs*, 2017 [accessed May 3, 2017].

[13] R. Parvathy and R. R. Devi, "Gradient descent based linear regression approach for modeling pid parameters," in *2014 International Conference on Power Signals Control and Computations (EPSCICON)*, Jan 2014, pp. 1–4.

[14] A. Ng, "Machine learning (open classroom)," *http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning*, 2012 [accessed May 1, 2017].

[15] W. C. Messner, D. M. Tilbury, and A. P. R. Hill, *Control Tutorials for MATLAB® and Simulink®*. Addison-Wesley, 1999.