Självständigt arbete på grundnivå

Independent degree project – first cycle

Automation

Improving PID control based on Neural Network

Jun Li

Mittuniversitetet
MID SWEDEN UNIVERSITY

Campus Härnösand Universitetsbacken 1, SE-871 88. Campus Sundsvall Holmgatan 10, SE-851 70 Sundsvall.
Campus Östersund Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

MID SWEDEN UNIVERSITY

Electronics design division

Examiner: Kent Bertilsson, Kent.Bertilsson@miun.se

Supervisor: Johan Sidén, Johan.Siden@miun.se

Author: Jun Li, juli1502@student.miun.se

Degree programme: Automation Engineering, 180 credits

Main field of study: Automation

Semester, year: Spring, 2018

Abstract

PID is a prevalent tool of automatic control in both industry and home environment, and PID parameters are often forced to modify because of systematic service on the machines or systems, which is time-costing. The project aims to investigate the possibility of applying neural network and reducing PID configuration in controlling industry process, by means of establishing control models and comparing control performance between conventional PID method and improved PID control based on neural network where two built neural networks are considered as cores to adjust weights which result in the suggested PID parameters. Adaptive learning rate is also applied which is adjusted by the algorithm based on the error changes. Algorithm program is written in Siemens TIA Portal and simulated in Factory I/O. In general, the simulations after analysis have shown that the proposed model has a better performance than conventional PID in terms of steady state, deviations and consistency of control value except tuning time. In the future the author is dedicated to continue improving the mentioned model through quickening learning process, applying better activation function and modifying variable structure and so on.

Keywords: PID, neural network, PLC, TIA Portal.

Acknowledgements

The author is really grateful for the consistent support from his family during the whole program. Purest love to queen Lei Shi and prince Cheng-An Li. Thanks to Professor Alfonso Gómez Espinosa for the patient and delicate explanation of his works. Thanks to the excellent education platform of Coursera.org.

Table of Contents

Abstract	iv
Acknowledgements	v
Terminology and abbreviations.....	vii
1 Introduction.....	1
1.1 Background and problem motivation	1
1.2 Overall aim	1
1.3 Limitations.....	1
1.4 Concrete objectives.....	2
2 Theory.....	3
2.1 Conventional PID optimization	4
2.2 Artificial neural network	6
2.3 Improved PID control based on neural network.....	8
3 Method	12
4 Results	14
4.1 Auto-tuned PID parameters	14
4.2 Tuning time	15
4.3 Performance of control results	17
4.4 Oscillations of control value	18
5 Discussion.....	21
5.1 Social aspects and ethical aspect	22
6 Conclusions	23
6.1 Future work.....	23
References.....	24
Appendix A: Mathematical deductions for BP of NNC	26
Appendix B: Mathematical deductions for BP of NNP	30

Terminology and abbreviations

PID	Proportional, Integral and Derivative
AI	Artificial Intelligence
NN	Neural Network
PIDNN	improved PID control based on NN
NNC	Neural Network of Controller
NNP	Neural Network of Plant
FP	Forward Propagation
BP	Back Propagation
MSE	Mean Squared Error

1 Introduction

PID is a prevalent tool of automatic control in both industry and home environment, where proportional, integral and derivative parameters should be pre-set to fulfill desired outputs. While as Artificial Intelligence (AI) has been applied in more fields nowadays, Neural Network (NN) tends to be one of the most efficient tools in Machine Learning [1]. This thesis is going to explore potential of employing NN in automatic control and improving PID.

1.1 Background and problem motivation

PID parameters could work well within some ranges, but for the plant which has complex nonlinearity or under extreme situations such as enormous perturbations, the performance is not satisfactory. In practice, PID parameters are often forced to modify because of systematic service on the machines or systems. Sometimes it is not easy to find optimal ones despite of using advanced optimization methods such as Ziegler-Nichols step-response etc. It is considered as a trend that the machines should not be designed to follow human beings' order for every step, but could also be able to perceive the changing environment and take sensible actions.

1.2 Overall aim

The project aims to investigate the possibility of applying NN and reducing PID configuration in controlling industry process, by means of establishing control models and comparing control performance between conventional PID method and improved PID control based on NN (PIDNN).

1.3 Limitations

This report only covers the application of one of the most popular control models in this field. There is possibility of improving the proposed model by means of re-organizing structure and modifying variables.

1.4 Concrete objectives

Objectives of the project are planned to answer following questions:

- Does PIDNN generate and adjust PID parameters automatically, which in turn reduce maintenance load of PID optimization?
- Does PIDNN respond to disturbances and reach steady state more quickly than conventional PID, which leads to better quality?
- Does PIDNN provide less deviations or oscillations when it reaches a steady state?
- Is there smoother output value of actuators from PIDNN in order to reach a steady state, which decreases the risk of shortening actuator's service cycle?
- Generally speaking is PIDNN a better control model comparing with conventional PID?

2 Theory

Improved PID control based on neural network (PIDNN) has been one of the popular methods employed for industry process control. Since already 1993 Xianzhong Cui & Kang G. Shin has shown promising simulations with good performance, simple structure and algorithm of neural network [2]. In a study of self-tuning PID control for underwater vehicles, R. Hernández-Alvarado .el has proved better performance of neural network based model with less energy consumption [3]. Fang, Ming-Chung found that PIDNN avoids inconvenience and insufficiency of searching optimal PID parameters in ship roll reduction [4].

However, there are problems with PIDNN as well:

- there is no systematical way to determine number of hidden layers and units at hidden layers¹. It does not work and improve the performance all the time to add hidden layers and units at hidden layers;
- neural network model for plant system should simulate the reality in a better way, otherwise it will affect the back propagation and weaken the effectiveness of adjusting controller weights;
- it might take a long time for the learning process until the algorithm reaches a steady state;
- there is risk of ending up with local optima despite different methods of initializing weights

Therefore, there is plenty of research over resolving the above difficulties. Kang, Jun el. adopted PSO algorithm to select initial weights in its training for improving the convergent speed and preventing the weights getting trapped in local optima [5]. Milovanović, Miroslav B. el. came up with a complex OENN-OEANFIS structure for online PID tuning which significantly reduced the signal deviations of the payload [6].

¹ definition of hidden layers and hidden units will be explained more in section 2.2

2.1 Conventional PID optimization

One of the conventional PID optimization methods is called Ziegler-Nichols closed-loop tuning method, or sometimes called Ultimate Cycling method.

To apply the Ziegler-Nichols Closed-Loop method for tuning controllers, follow these steps [7]:

1. Stabilize the process, make sure no process changes are active;
2. If the loop is currently oscillating, make sure that the process variable stops oscillating when the controller is placed in manual mode;
3. Remove Integral action and Derivative action from controller by setting the parameter to zero;
4. Place the controller in automatic control mode if it is in manual mode;
5. Make a set point change and monitor the result;
6. If the process variable does not oscillate at all, double the controller gain;
7. If the process variable oscillates and the amplitude of the peaks decreases, increase the controller gain by 50% (or less if getting close to a constant amplitude);
8. If the process variable oscillates and the amplitude of the peaks increases, decrease the controller gain by 50% (or less if getting close to a constant amplitude);
9. If the process variable or controller output hits its upper or lower limits, decrease the controller gain by 50%. The process variable and controller output must oscillate freely for this method to work;
10. If the oscillations have died out, go to step 6;
11. If the loop is oscillating, but not with a constant amplitude, repeat steps 8, 9, and 10 until oscillations with a constant amplitude are obtained;
12. If the process variable is oscillating with constant amplitude, and neither the process variable nor the controller output hits its limits, take note of the "Ultimate" controller gain (K_u) and the period of the oscillation (T_u);

Figure 2.1: The Ziegler-Nichols closed-loop tuning method [8]
*PV denotes process value and CO for control output

13. Cut the controller gain in half to let the control loop stabilize while you do the calculations;
14. Calculate new controller settings using the equations below, enter them into the controller, and make a set point change to test them.

Use the values of K_u and T_u to determine values of PID gain setting according to the following tuning rule table.

Table 2.2: Tuning rules [8]			
Control type	K_p	T_i	T_d
P	$0.5 * K_u$	-	-
PI	$0.45 * K_u$	$0.85 * T_u$	-
PID	$0.6 * K_u$	$0.5 * T_u$	$0.13 * T_u$

If the output does not exhibit sustained oscillations for whatever value control gain may take, then this method does not apply [8].

2.2 Artificial neural network

Artificial neural network (NN) is a model inspired of how human brain works, and is widely used nowadays in many applications such as voice interpretation on smart phones, image detection, speech recognition and so on.

Figure 2.2: Neural network model

Neurons are basically computational units that take inputs (dendrites) as electrical inputs that are channeled to outputs (axons). In a neural network model, there is at least one input layer, one hidden layer and one output layer. As in Figure 2.2, there are m units at input layer (layer 1), n units at hidden layer (layer 2) and p units at output layer (layer 3). In this model, dendrites are like the input features X_1 to X_m , and the output is the result of our hypothesis function h_1^3 to h_p^3 . Input node X_0 and h_0^2 is called the "bias unit" which is always equal to 1. In neural networks, transfer function g is used as simulating neurons, which is called activation function. In this thesis the sigmoid function is employed, because its value span is (0~1). And these intermediate or hidden layer units h_1^2 to h_n^2 are called activation units. The theta parameters θ^1 , θ^2 are called weights, which are the objectives to be trained in algorithm.

The process where NN receives signals and generate results is called forward propagation (FP), which is expressed as follows in Octave language.

$$g(x) = \frac{1}{1+e^{-x}} \quad (2.1)$$

$$z^2 = \theta^1 * [1 \ X] \quad (2.2)$$

$$h^2 = g(z^2) \quad (2.3)$$

$$z^3 = \theta^2 * [1 \ h^2] \quad (2.4)$$

$$h^3 = g(z^3) \quad (2.5)$$

where equation (2.1) is the sigmoid function, (2.2) and (2.4) generates preliminary units in layer 2 and 4, (2.3) generates hidden units, (2.5) generates output units. X is input matrix of $m \times 1$, z^2 and h^2 are activation matrixes of $n \times 1$, z^3 and h^3 of $p \times 1$, θ^1 of $n \times (m+1)$, θ^2 of $p \times (n+1)$.

For every iteration of FP there is a cost calculated indicating how good the trained hypothesis is. In this thesis the squared error is used as cost function which is aimed to minimize.

$$J = \frac{1}{2} * (y - h^3)^2 \quad (2.6)$$

where y is the actual value, and h^3 is the trained output.

Then the weights are adjusted automatically as follows:

$$\theta_{i,j}^l = \theta_{i,j}^l - \alpha * \frac{\partial J}{\partial \theta_{i,j}^l} \quad (2.7)$$

where $\theta_{i,j}^l$ is the weight in i row j column of layer l , α is the learning rate, $\frac{\partial J}{\partial \theta_{i,j}^l}$ is called gradient descent of $\theta_{i,j}^l$.

In order to obtain all weights' gradient descent, back propagation (BP) should be used. BP is neural-network terminology for minimizing cost function and computing the partial derivative of $J(\theta_{i,j}^l)$ [9].

"Every neuron is given the difference between desired output and actual output, starting from the output layer. It first adjusts its weights. Inputs that led to the output being closer to the desired output are weighted higher next time. For example, if an input was 5, but the actual output was too small, the input's weight increases. Then, it propagates the error back to the previous layer. Each previous neuron gets a new error proportional to its weight."[10]

It is suggested by Coursera to follow these steps for training a NN [11]:

1. Randomly initialize the weights;
2. Implement forward propagation to get trained output for any X_i ;
3. Implement the cost function;
4. Implement backpropagation to compute partial derivatives;
5. Use gradient checking to confirm that your backpropagation works. Then disable gradient checking;
6. Use gradient descent or a built-in optimization function to minimize the cost function with the weights in theta.

2.3 Improved PID control based on neural network

Improved PID control based on neural network (PIDNN) has been one of the popular methods employed for industry process control. The structure of PIDNN is shown in Figure 2.3. Neural network controller (NNC) denotes the model that generates preliminary PID parameters automatically under process. NNC has a structure of $(8,4,3)^2$ where 8 input units include set point $sp(t)$, process value $y(t)$, control error $e(t)$, flow meter $ft(t)$ and their historical data for one second earlier $sp(t-1)$, $y(t-1)$, $e(t-1)$, $ft(t-1)$ since sampling time is chosen 1 second in the thesis.

² (a,b,c) indicates NN structure where there are one input layer, one hidden layer, one output layer, a input units, b hidden units and c output units.

In the output layer there are 3 units which corresponds preliminary PID parameters. Because these output values are between 0 and 1, scaling function is needed.

$$f = \begin{cases} Kp = cp * h_1^3 \\ Ki = ci * h_2^3 \\ Kd = cd * h_3^3 \end{cases} \quad (2.8)$$

where cp , ci and cd are the scaling factors³, h_j^3 is the output of NNC.

In order to carry out an effective BP for NNC, a neural network plant model (NNP) is necessary to simulate plant system's properties mainly the interaction between control value and the coming process value. NNP has a structure of (4,4,1) where 4 input units include control value $cv(t)$, real-time process value $y(t)$ and the historical values $cv(t-1)$, $y(t-1)$, and the simulated coming process value $yp(t+1)$ is generated in output layer.

Figure 2.3: Control system scheme of PIDNN

³ Scaling factors have default values of 1 in the thesis

In the discrete time domain, the digital PID algorithm can be expressed as follows [3]:

$$\begin{aligned}\Delta u &= u(t) - u(t-1) \\ &= K_p * (e(t) - e(t-1)) + K_i * e(t) \\ &\quad + K_d * (e(t) - 2e(t-1) + e(t-2))\end{aligned}\quad (2.9)$$

where K_p , K_i and K_d are PID parameters, $e(t)$ is the control error between set point and process value.

Cost function of NNP is

$$J_1 = \frac{1}{2} * (y - yp)^2 = \frac{1}{2} * (y - h^6)^2 \quad (2.10)$$

where y is process value, both yp and h^6 is the simulated process value.

Gradient descent of NNP is as follows in Octave language (see detailed mathematical deduction in Appendix C):

$$\frac{\partial J_1}{\partial \theta^5} = -(y - yp) * \frac{\partial yp}{\partial \theta^5} = -(y - yp) * g'(z^6) * [1 \ h^5'] \quad (2.11)$$

$$\begin{aligned}\frac{\partial J_1}{\partial \theta^4} &= -(y - yp) * \frac{\partial yp}{\partial \theta^4} \\ &= -(y - yp) * g'(z^6) * (\theta^5(2:5)' .* g'(z^5)) * [1 \ Xp']\end{aligned}\quad (2.12)$$

where Xp is the input units of NNP.

Cost function of NNC is

$$J_2 = \frac{1}{2} * (sp - y)^2 \quad (2.13)$$

where y is process value, and sp is the set point.

Gradient descent of NNC is as follows in Octave language (see detailed mathematical deduction in Appendix B):

$$\frac{\partial J_2}{\partial \theta^i} = -(sp - y) * \frac{\partial yp}{\partial \Delta u} * \frac{\partial \Delta u}{\partial PID} * \frac{\partial PID}{\partial h^3} * \frac{\partial h^3}{\partial \theta^i} \quad (2.14)$$

$$\begin{aligned} \text{delta}a_{temp} &= -(sp - y) * \frac{\partial yp}{\partial \Delta u} \\ &= -(sp - y) * g'(z^6) * (\theta_{11}^5 * g'(z_1^5) * \theta_{13}^4 + \theta_{12}^5 * g'(z_2^5) * \theta_{23}^4 \\ &\quad + \theta_{13}^5 * g'(z_3^5) * \theta_{33}^4 + \theta_{14}^5 * g'(z_4^5) * \theta_{43}^4) \end{aligned} \quad (2.15)$$

$$\frac{\partial J_2}{\partial \theta_{1j}^2} = \text{delta}a_{temp} * (e(t) - e(t - 1)) * cp * g'(z_1^3) * [1 \ h^2'] \quad (2.16)$$

$$\frac{\partial J_2}{\partial \theta_{2j}^2} = \text{delta}a_{temp} * e(t) * ci * g'(z_2^3) * [1 \ h^2'] \quad (2.17)$$

$$\frac{\partial J_2}{\partial \theta_{3j}^2} = \text{delta}a_{temp} * (e(t) - 2e(t - 1) + e(t - 2)) * cd * g'(z_3^3) * [1 \ h^2'] \quad (2.18)$$

$$\begin{aligned} \frac{\partial J_2}{\partial \theta^1} &= \text{delta}a_{temp} * ((e(t) - e(t - 1)) * cp * \theta_{1j}^2(2:5)' * g'(z^2) * g'(z_1^3) * [1 \ X'] \\ &\quad + e(t) * ci * \theta_{2j}^2(2:5)' * g'(z^2) * g'(z_2^3) * [1 \ X'] \\ &\quad + (e(t) - 2e(t - 1) + e(t - 2)) * cd * \theta_{3j}^2(2:5)' * g'(z^2) * g'(z_3^3) * [1 \ X']) \end{aligned} \quad (2.19)$$

To resolve the problem of slow convergence with conventional PID, adaptive learning rate is applied which is adjusted by the algorithm based on the error changes. The procedure is borrowed from the paper of L. Zhao [12]: if the present error is 0.9 times smaller than the last error, the learning rate should be multiplied by 1.1; if the present error is 1.05 times larger than the last error, the learning rate should be multiplied by 0.9.

3 Method

In order to carry out PIDNN, theory of NN is a critical part. An intensive internet course of ML was taken at Coursera which includes different categories of ML algorithms such as neural networks, support vector machines, kernels and useful strategies of improving algorithms. NN model and online learning process has been chosen due to its robust applicability and properties of level control process.

In the very beginning, an independent NN model with one output value was built which after test did not give promising results. Through literature review, it is found that PIDNN has become a prevalent model type that has been investigated and applied in automatic control process with complex nonlinearity. However, more paper studied the application in machines such as robots, underwater vehicles and ships. One common PIDNN model has been employed in the report where NNC and NNP are considered as cores to adjust weights which result in the suggested PID parameters. Besides literature review, contact with paper writers has been made as well to clarify the relevant model structure.

Figure 3.1: Factory I/O scene and SCADA

Algorithm program is written in Siemens TIA Portal where PLC 1500 series are used. Factory I/O is the simulation software. The simulation scene and SCADA of an assignment from previous course “Advanced Process Control” has been slightly modified to fit in this project (see Figure 3.1). In the scene, there is a tank whose level we need to regulate. It has been equipped with filling and discharging control valve, a level sensor and a volume flow meter. Discharging valve is a hand valve, which is used as a disturbance factor in the system, while the filling valve receive control signal from PLC.

Simulations are organized in the following manner:

1. Start with empty tank, 0 cm;
2. Scenario A: Change set point to 30 cm (10%) and hand valve CV1D to 30%;
3. Scenario B: Change set point to 75 cm (25%) and hand valve CV1D to 40%;
4. Scenario C: Change set point to 150 cm (50%) and hand valve CV1D to 50%;
5. Scenario D: Change set point to 225 cm (75%) and hand valve CV1D to 60%;
6. Scenario E: Change set point to 270 cm (90%) and hand valve CV1D to 70%;

Under simulation, SCADA logs useful data which is processed later in Octave which is also used to build the NN prototype. All the data management and visualization has been done with help of Octave.

4 Results

Following the steps of Ziegler-Nichols Closed-Loop method for tuning PID controllers, it is found that $K_u \approx 30, T_u \approx 5\text{ s}$. Therefore, the suggested PID parameters are

$$K_p = 0.6 * K_u = 18, K_i = \frac{1}{0.5 * T_u} = 10, K_d = 0.13 * T_u = 0.65$$

4.1 Auto-tuned PID parameters

Figure 4.1: Auto-tuned PID parameters

Figure 4.1 shows the auto-tuned PID parameters by PIDNN. They vary through all the scenarios, significantly when the set point and disturbances from discharging valve have changed. While scenario drifts from A to E, proportional gain is descending while integral gain is ascending. Generally speaking, derivative gain is keeping in the same level and responding to the scenario changes significantly.

4.2 Tuning time

The control process of simulations in 5 scenarios is visualized in Figure 4.2-4.6:

Figure 4.2: Control process PIDNN vs. PID in scenario A

Figure 4.3: Control process PIDNN vs. PID in scenario B

Figure 4.4: Control process PIDNN vs. PID in scenario C

Figure 4.5: Control process PIDNN vs. PID in scenario D

Figure 4.6: Control process PIDNN vs. PID in scenario E

In all the scenarios, PID responds to the changes more quickly and reaches its end state earlier than PIDNN.

4.3 Performance of control results

Cost values or Mean squared error (MSE⁴) is summarized in the table below. PIDNN gives less MSE than conventional PID in all scenarios except B.

Table 4.1: Cost values PIDNN vs. PID

	A	B	C	D	E
PIDNN	3.9174e-4	4.7045e-4	9.9868e-4	8.3340e-4	6.3615e-4
PID	4.0002e-4	1.5187e-4	17.036e-4	15.366e-4	6.7874e-4

Putting MSE aside, PIDNN gives a steady state in all scenarios while PID leaves infinite oscillations around set point in the end state (see Figure 4.2-4.6).

⁴ It is calculated as $MSE=0.5*(sp/300-pv/300)^2$

4.4 Oscillations of control value

The real trajectory of control values is visualized in Figure 4.7-4.11. It shows that PIDNN has less oscillation than PID in all scenarios. And control value of PIDNN is tending to reach a constant point through time while PID keeps an infinite oscillation.

Figure 4.7: Control value PIDNN vs. PID in scenario A

Figure 4.8: Control value PIDNN vs. PID in scenario B

Figure 4.9: Control value PIDNN vs. PID in scenario C

Figure 4.10: Control value PIDNN vs. PID in scenario D

Figure 4.11: Control value PIDNN vs. PID in scenario E

The standard deviations of control values have been summarized in Table 4.2. It shows that PIDNN has smoother control value for filling valve.

Table 4.2: Standard deviation of control values PIDNN vs. PID

	A	B	C	D	E
PIDNN	7.4417	11.998	15.182	8.7081	2.7933
PID	17.875	26.014	40.086	22.835	5.0665

5 Discussion

The auto-tuned PID parameters have bigger oscillations between scenarios and then tend to bend into a steady state. It seems that PIDNN tries to adjust parameters and strive against the disturbances. After a short period, when the process value approaching set point and control error have become less and less, all the parameters continue to stay at a stable level, which means that PIDNN is working actively and has potential of reducing maintenance load of PID optimization.

It has been mentioned by many studies that PIDNN has the problem of long learning period, and that has been verified in this report as well. Conventional PID shows a better capacity to get to an end state in a shorter time. Even though the end state of PID is still with oscillations, the control error is small enough for some uncritical processes.

In all scenarios, PIDNN has shown a perfect steady state after a learning period, while conventional PID keeps small oscillations within a deviation of 3~8 cm in most occasions. Therefore, PIDNN is considered to have better performance, which is also demonstrated by index MSE except scenario B where PIDNN produced a very big oscillation in the beginning and PID kept a constant degree of oscillations. What is interesting is that PID gives much smaller oscillations in scenario E than in other scenarios, of which the deviation is only round 1.5 cm.

The simulations show that control values of PIDNN approaches a steady state after a period in all scenarios while PID keeps infinite oscillations, which essentially results in corresponding performance. That has been numerically demonstrated in the contrast of standard deviations as well. Thus PIDNN gives less risk of wearing out actuators than conventional PID.

In one word, through result analysis, the project's goal has been achieved. The thesis has verified the possibility of applying NN and reducing PID configuration in controlling simulated industry process, by means of establishing control models and comparing control performance between conventional PID and PIDNN. And PIDNN has been proved better than PID as expected. On the other hand, what is unexpected is that Ziegler-Nichols closed-loop tuning method does not even

give acceptable results and reach a steady state, which implies it might neither work well in practice. What's more, it requires quite a time to find the ultimate oscillation.

There are different versions of PIDNN in this field, such as using system sign instead of a numeric number generated by NNP to implement BP. Also are there handful techniques of improving model performance including quickening learning process, applying better activation function and modifying variable structure and so on. Therefore the mentioned model has potential of getting even better.

The project has executed algorithm in Siemens PLC and vectorization operation is not possible in TIA Portal, which makes programming really un-efficiency. It is suggested to accomplish algorithm with professional software such as Octave at OPC than locally, which on the other side will burden the communication of network.

5.1 Social aspects and ethical aspect

Better performance shown by PIDNN will obviously lead to better process and consequently product quality, and higher sustainability. What's more, the new method is going to liberate work force a great deal so that productivity will be increased as a result.

PIDNN is a sign that there is a large potential of NN and AI being applied in not only industry but also other branches such as administration, finance, medical care etc. As the life quality will be enhanced by AI, many think AI is a treat to work force market. Since production and service efficiency would be upgraded to completely another level, there would not be such a demand of work force as today, and unemployment would be a major concern for the society. However, the author holds a point of view that AI is going to create more opportunities in other existed and new branches. With retrospect of human history, every technology revolution brought a massive productivity improvement, but our life quality has been rising all the time. History is not planned but created.

AI in another way might be a completely threat against personal integrity and civil society if it is abused to undermine others' privacy or maintain a sort of governance such as surveillance.

6 Conclusions

In general, PIDNN has shown a better performance than conventional PID, or strictly speaking better than Ziegler-Nichols closed-loop tuning method, in terms of steady state, deviations and consistency of control value. However, PIDNN is defeated in tuning time even though an adaptive learning rate has been applied in the model.

The project has explored possibility of applying NN in industry process control, and carried out algorithm in PLC language, which is the most important contribution to the related research.

Proposed PIDNN will wipe out maintenance work of PID optimization, and requires a learning period to reach a steady state. Therefore, it is applicable for those processes where couple of beginning oscillations is tolerable and it is feasible to change some work procedures like documentation over PID parameters in medicine industry.

6.1 Future work

There are different variations of PIDNN in this field, and also useful techniques of improving model performance including quickening learning process, applying better activation function and modifying variable structure and so on. The author is dedicated to continue improving the mentioned model and exploring new ML tools in the future.

References

- [1] Coursera.org, “Neural Networks: Representation”,
<https://www.coursera.org/learn/machine-learning/home/week/4>
Published unknown date. Retrived 2018-06-05.
- [2] Xianzhong Cui; Kang G. Shin, “Direct control and coordination using neural networks”, *IEEE Transactions on Systems, Man, and Cybernetics*, May-June 1993, Vol.23(3), pp.686-697
- [3] Rodrigo Hernández-Alvarado; Luis Govinda García-Valdovinos; Tomás Salgado-Jiménez; Alfonso Gómez-Espinosa; Fernando Fonseca-Navarro, “Neural Network-Based Self-Tuning PID Control for Underwater Vehicles”, *Sensors*, 01 September 2016, Vol.16(9), p.1429
- [4] Fang, Ming-Chung; Zhuo, Young-Zoung; Lee, Zi-Yi, “The application of the self-tuning neural network PID controller on the ship roll reduction in random waves”, *Ocean Engineering*, 2010, Vol.37(7), pp.529-538
- [5] Kang, Jun; Meng, Wenjun; Abraham, Ajith; Liu, Hongbo, “An adaptive PID neural network for complex nonlinear system control”, *Neurocomputing*, 5 July 2014, Vol.135, pp.79-85
- [6] Milovanović, Miroslav B.; Antić, Dragan S.; Milojković, Marko T.; Nikolić, Saša S.; Perić, Staniša Lj.; Spasić, Miodrag D., “Adaptive PID control based on orthogonal endocrine neural networks”, *Neural Networks*, December 2016, Vol.84, pp.80-90
- [7] Control Notes, “Ziegler-Nichols Closed-Loop Tuning Method”,
<http://blog.opticontrols.com/archives/131>, Published 2010-03-31.
Retreived 2018-05-14.
- [8] Kurser.lha.dk, “Ziegler Nichols method”,
http://kurser.lha.dk/m/mtpri1/control/3_PID/f_ziegler/ziegler_nichols_method.html#Ziegler-Nichols_method, Published unknown date. Retreived 2018-05-14.
- [9] Coursera.org, “backpropagation-algorithm”,

<https://www.coursera.org/learn/machine-learning/supplement/pjdBa/backpropagation-algorithm>
Published unknown date. Retrived 2018-05-15.

- [10] The Blog, "Implementing Back Propagation in a Neural Network", <https://theblog.github.io/post/neural-networks-implementing-backpropagation/>
Published 2016-10-19. Reteived 2018-05-16.
- [11] Coursera.org, "Putting it together", <https://www.coursera.org/learn/machine-learning/supplement/Uskwd/putting-it-together>
Published unknown date. Retrived 2018-05-16.
- [12] Zhao, Lei, "Single tank liquid level control based on improved BP neural network PID control algorithm", *2013 IEEE Third International Conference on Information Science and Technology*, March 2013, pp.230-232

19

$$J_1 = \frac{1}{2} (y - y_p)^2$$

II. Adjust NNC weights

$$J_2 = \frac{1}{2} (sp - Y)^2$$

I:

$$\frac{\partial J}{\partial \theta_5} = - \frac{(Y - Y_p) \cdot g'(z_6) \cdot [1 \ h^5]}{\text{delta } 6}$$

$$\frac{\partial J}{\partial \theta_4} = \frac{\theta_5 (z=5)' \cdot g'(z_5) \cdot \text{delta } 6 \cdot [1 \ x'']}{\text{delta } 5}$$

$$\begin{aligned} \text{II: } \because Z_p &= \theta_{10}^5 + \theta_{11}^5 \cdot h_1^5 + \theta_{12}^5 \cdot h_2^5 + \theta_{13}^5 \cdot h_3^5 + \theta_{14}^5 \cdot h_4^5 \\ &= \theta_{10}^5 + \theta_{11}^5 \cdot g(\theta_{10}^4 + \theta_{11}^4 \cdot Y + \theta_{12}^4 \cdot Y' + \theta_{13}^4 \cdot \Delta U + \theta_{14}^4 \cdot \Delta U') \\ &\quad + \theta_{12}^5 \cdot g(\theta_{20}^4 + \theta_{21}^4 \cdot Y + \theta_{22}^4 \cdot Y' + \theta_{23}^4 \cdot \Delta U + \theta_{24}^4 \cdot \Delta U') \\ &\quad + \theta_{13}^5 \cdot g(\theta_{30}^4 + \theta_{31}^4 \cdot Y + \theta_{32}^4 \cdot Y' + \theta_{33}^4 \cdot \Delta U + \theta_{34}^4 \cdot \Delta U') \\ &\quad + \theta_{14}^5 \cdot g(\theta_{40}^4 + \theta_{41}^4 \cdot Y + \theta_{42}^4 \cdot Y' + \theta_{43}^4 \cdot \Delta U + \theta_{44}^4 \cdot \Delta U') \end{aligned}$$

$$\therefore \frac{\partial Z_p}{\partial \Delta U} = \theta_{11}^5 \cdot g'(z_5) \cdot \theta_{13}^4 + \theta_{12}^5 \cdot g'(z_2) \cdot \theta_{23}^4 + \theta_{13}^5 \cdot g'(z_3) \cdot \theta_{33}^4 + \theta_{14}^5 \cdot g'(z_4) \cdot \theta_{43}^4$$

P2

$$\begin{aligned}
 \Delta U &= kp(e-e') + ki \cdot e + kd(e - ze' + e'') \quad e = sp - y; \quad f = \begin{cases} kp = cp \cdot h_1^3 \\ ki = ci \cdot h_2^3 \\ kd = cd \cdot h_3^3 \end{cases} \\
 \frac{\partial \Delta U}{\partial h_3} &= \frac{\partial \Delta U}{\partial kp} \cdot \frac{\partial kp}{\partial h_3} + \frac{\partial \Delta U}{\partial ki} \cdot \frac{\partial ki}{\partial h_3} + \frac{\partial \Delta U}{\partial kd} \cdot \frac{\partial kd}{\partial h_3} \\
 &= (e-e') \cdot cp \cdot \frac{\partial}{\partial h_1^3} + e \cdot ci \cdot \frac{\partial}{\partial h_2^3} + (e - ze' + e'') \cdot cd \cdot \frac{\partial}{\partial h_3^3} \\
 \Rightarrow \frac{\partial \Delta U}{\partial \theta} &= (e-e') \cdot cp \cdot \frac{\partial h_1^3}{\partial \theta} + e \cdot ci \cdot \frac{\partial h_2^3}{\partial \theta} + (e - ze' + e'') \cdot cd \cdot \frac{\partial h_3^3}{\partial \theta} \\
 \frac{\partial h_1^3}{\partial \theta_{1j}^2} &= g'(z_1^3) \cdot [1 \quad h^2]' \quad , \quad \frac{\partial h_2^3}{\partial \theta_{2j}^2} = g'(z_2^3) \cdot [1 \quad h^2]' \quad , \quad \frac{\partial h_3^3}{\partial \theta_{3j}^2} = g'(z_3^3) \cdot [1 \quad h^2]' \\
 \frac{\partial h_1^3}{\partial \theta'} &= \theta_{1j}^2(z=5') \cdot g'(z_1^3) \cdot g'(z_1^3) \cdot [1 \quad x'] \\
 \frac{\partial h_2^3}{\partial \theta'} &= \theta_{2j}^2(z=5') \cdot g'(z_2^3) \cdot g'(z_2^3) \cdot [1 \quad x'] \\
 \frac{\partial h_3^3}{\partial \theta'} &= \theta_{3j}^2(z=5') \cdot g'(z_3^3) \cdot g'(z_3^3) \cdot [1 \quad x']
 \end{aligned}$$

p3

$$\begin{aligned}
 \Rightarrow \frac{\partial \Delta U}{\partial \theta_{1j}^2} &= (e - e') \cdot cp \cdot g'(z_1^3) \cdot [1 \ k'] \\
 \frac{\partial \Delta U}{\partial \theta_{2j}^2} &= e \cdot ci \cdot g'(z_2^3) \cdot [1 \ k'] \quad , \quad \frac{\partial \Delta U}{\partial \theta_{3j}^2} = (e - ze' + e'') \cdot cd \cdot g'(z_3^3) \cdot [1 \ k'] \\
 \\
 \frac{\partial \Delta U}{\partial \theta_1^2} &= (e - e') \cdot cp \cdot \theta_{1j}^2(z=5)' \cdot * g'(z_2) \cdot g'(z_1^3) \cdot [1 \ x'] \\
 &\quad + e \cdot ci \cdot \theta_{2j}^2(z=5)' \cdot * g'(z_2) \cdot g'(z_2^3) \cdot [1 \ x'] \\
 &\quad + (e - ze' + e'') \cdot cd \cdot \theta_{3j}^2(z=5)' \cdot * g'(z_2) \cdot g'(z_3^3) \cdot [1 \ x']
 \end{aligned}$$

P4

Appendix B: Mathematical deductions for BP of NNP

$$J = \frac{1}{2m} \sum (h_5 - y)^2 \quad // \text{ Cost Function}$$

$$\begin{aligned} \frac{\partial J}{\partial \theta_{10}^4} &= \frac{1}{2m} \sum 2 \cdot (h_5 - y) \cdot \frac{\partial h_5}{\partial \theta_{10}^4} = \frac{1}{m} \sum (h_5 - y) \cdot \frac{\partial (g(z_5))}{\partial \theta_{10}^4} \\ &= \frac{1}{m} \sum (h_5 - y) \cdot g'(z_5) \cdot \frac{\partial z_5}{\partial \theta_{10}^4} \end{aligned}$$

$$\because z_5 = \theta^4 \cdot [1; h_4] = \theta_{10}^4 \cdot 1 + \theta_{11}^4 \cdot h_1^4 + \theta_{12}^4 \cdot h_2^4 + \theta_{13}^4 \cdot h_3^4 + \theta_{14}^4 \cdot h_4^4$$

$$\therefore \frac{\partial z_5}{\partial \theta_{10}^4} = 1, \frac{\partial z_5}{\partial \theta_{11}^4} = h_1^4, \frac{\partial z_5}{\partial \theta_{12}^4} = h_2^4, \frac{\partial z_5}{\partial \theta_{13}^4} = h_3^4, \frac{\partial z_5}{\partial \theta_{14}^4} = h_4^4$$

$$\begin{aligned} \Rightarrow \frac{\partial J}{\partial \theta_{ij}^4} &= \frac{1}{m} \sum (h_5 - y) \cdot g'(z_5) \cdot h_j^4 \Rightarrow \frac{\partial J}{\partial \theta^4} = \frac{1}{m} \sum \delta^5 \cdot h^4 \cdot g'(z_5) \\ &= \frac{1}{m} \sum \delta^5 \cdot g'(z_5) \cdot [1 \ h^4] \end{aligned}$$

$$\frac{\partial J}{\partial \theta_{10}^3} = \frac{1}{m} \sum (h_5 - y) \cdot g'(z_5) \cdot \frac{\partial z_5}{\partial \theta_{10}^3}$$

$$\begin{aligned} \because z_5 &= \theta_{10}^4 \cdot 1 + \theta_{11}^4 \cdot h_1^4 + \theta_{12}^4 \cdot h_2^4 + \theta_{13}^4 \cdot h_3^4 + \theta_{14}^4 \cdot h_4^4 \\ &= \theta_{10}^4 + \theta_{11}^4 \cdot g(z_1^4) + \theta_{12}^4 \cdot g(z_2^4) + \theta_{13}^4 \cdot g(z_3^4) + \theta_{14}^4 \cdot g(z_4^4) \\ &= \theta_{10}^4 + \theta_{11}^4 \cdot g(\theta_{10}^3 \cdot 1 + \theta_{11}^3 \cdot h_1^3 + \theta_{12}^3 \cdot h_2^3 + \theta_{13}^3 \cdot h_3^3 + \theta_{14}^3 \cdot h_4^3) \\ &\quad + \theta_{12}^4 \cdot g(\theta_{20}^3 \cdot 1 + \theta_{21}^3 \cdot h_1^3 + \theta_{22}^3 \cdot h_2^3 + \theta_{23}^3 \cdot h_3^3 + \theta_{24}^3 \cdot h_4^3) \\ &\quad + \theta_{13}^4 \cdot g(\theta_{30}^3 \cdot 1 + \theta_{31}^3 \cdot h_1^3 + \theta_{32}^3 \cdot h_2^3 + \theta_{33}^3 \cdot h_3^3 + \theta_{34}^3 \cdot h_4^3) \\ &\quad + \theta_{14}^4 \cdot g(\theta_{40}^3 \cdot 1 + \theta_{41}^3 \cdot h_1^3 + \theta_{42}^3 \cdot h_2^3 + \theta_{43}^3 \cdot h_3^3 + \theta_{44}^3 \cdot h_4^3) \end{aligned}$$

$$\therefore \frac{\partial z_5}{\partial \theta_{10}^3} = \theta_{11}^4 \cdot g'(z_1^4) \cdot 1, \frac{\partial z_5}{\partial \theta_{11}^3} = \theta_{11}^4 \cdot g'(z_1^4) \cdot h_{11}^3, \dots, \frac{\partial z_5}{\partial \theta_{14}^3} = \theta_{11}^4 \cdot g'(z_1^4) \cdot h_{14}^3$$

$$\frac{\partial z_5}{\partial \theta_{20}^3} = \theta_{12}^4 \cdot g'(z_2^4) \cdot 1, \frac{\partial z_5}{\partial \theta_{21}^3} = \theta_{12}^4 \cdot g'(z_2^4) \cdot h_1^3, \dots, \frac{\partial z_5}{\partial \theta_{24}^3} = \theta_{12}^4 \cdot g'(z_2^4) \cdot h_4^3$$

...

...

...

...

$$\frac{\partial z_5}{\partial \theta_{40}^3} = \theta_{14}^4 \cdot g'(z_4^4) \cdot 1, \frac{\partial z_5}{\partial \theta_{41}^3} = \theta_{14}^4 \cdot g'(z_4^4) \cdot h_1^3, \dots, \frac{\partial z_5}{\partial \theta_{44}^3} = \theta_{14}^4 \cdot g'(z_4^4) \cdot h_4^3$$

$$\Rightarrow \frac{\partial z_5}{\partial \theta^3} = \theta^4(z_5)' \cdot g'(z^4) \cdot [1 \ h^3]$$

$$\Rightarrow \frac{\partial J}{\partial \theta^3} = \frac{1}{m} \sum (h_5 - y) \cdot g'(z_5) \cdot \theta^4(z_5)' \cdot g'(z^4) \cdot [1 \ h^3]$$

Pz

$$\frac{\partial J}{\partial \theta^2} = \frac{1}{m} \cdot \sum (h_5 - x) \cdot g'(z_5) \cdot \frac{\partial z_5}{\partial \theta^2}$$

$$\therefore \frac{\partial z_5}{\partial \theta^2} = \theta_{11}^4 \cdot g'(z_1^4) \cdot \frac{\partial z_1^4}{\partial \theta^2} + \theta_{12}^4 \cdot g'(z_2^4) \cdot \frac{\partial z_2^4}{\partial \theta^2} + \theta_{13}^4 \cdot g'(z_3^4) \cdot \frac{\partial z_3^4}{\partial \theta^2} + \theta_{14}^4 \cdot g'(z_4^4) \cdot \frac{\partial z_4^4}{\partial \theta^2}$$

$$\begin{aligned} z_1^4 &= \theta_{10}^3 + \theta_{11}^3 \cdot g(\theta_{10}^2 \cdot 1 + \theta_{11}^2 \cdot h_1^2 + \theta_{12}^2 \cdot h_2^2 + \theta_{13}^2 \cdot h_3^2 + \theta_{14}^2 \cdot h_4^2) \quad z_1^3 \\ &\quad + \theta_{12}^3 \cdot g(\theta_{20}^2 \cdot 1 + \theta_{21}^2 \cdot h_1^2 + \theta_{22}^2 \cdot h_2^2 + \theta_{23}^2 \cdot h_3^2 + \theta_{24}^2 \cdot h_4^2) \quad z_2^3 \\ &\quad + \theta_{13}^3 \cdot g(\theta_{30}^2 \cdot 1 + \theta_{31}^2 \cdot h_1^2 + \theta_{32}^2 \cdot h_2^2 + \theta_{33}^2 \cdot h_3^2 + \theta_{34}^2 \cdot h_4^2) \quad z_3^3 \\ &\quad + \theta_{14}^3 \cdot g(\theta_{40}^2 \cdot 1 + \theta_{41}^2 \cdot h_1^2 + \theta_{42}^2 \cdot h_2^2 + \theta_{43}^2 \cdot h_3^2 + \theta_{44}^2 \cdot h_4^2) \quad z_4^3 \end{aligned}$$

$$z_2^4 = \theta_{20}^3 + \theta_{21}^3 \cdot g(z_1^3) + \theta_{22}^3 \cdot g(z_2^3) + \theta_{23}^3 \cdot g(z_3^3) + \theta_{24}^3 \cdot g(z_4^3)$$

$$z_3^4 = \theta_{30}^3 + \theta_{31}^3 \cdot g(z_1^3) + \theta_{32}^3 \cdot g(z_2^3) + \theta_{33}^3 \cdot g(z_3^3) + \theta_{34}^3 \cdot g(z_4^3)$$

$$z_4^4 = \theta_{40}^3 + \theta_{41}^3 \cdot g(z_1^3) + \theta_{42}^3 \cdot g(z_2^3) + \theta_{43}^3 \cdot g(z_3^3) + \theta_{44}^3 \cdot g(z_4^3)$$

$$\Rightarrow \frac{\partial z_5}{\partial \theta_{10}^2} = \theta_{11}^4 \cdot g'(z_1^4) \cdot \theta_{11}^3 \cdot g'(z_1^3) \cdot 1 + \theta_{12}^4 \cdot g'(z_2^4) \cdot \theta_{21}^3 \cdot g'(z_1^3) \cdot 1 \\ + \theta_{13}^4 \cdot g'(z_3^4) \cdot \theta_{31}^3 \cdot g'(z_1^3) \cdot 1 + \theta_{14}^4 \cdot g'(z_4^4) \cdot \theta_{41}^3 \cdot g'(z_1^3) \cdot 1$$

$$\frac{\partial z_5}{\partial \theta_{11}^2} = \theta_{11}^4 \cdot g'(z_1^4) \cdot \theta_{11}^3 \cdot g'(z_1^3) \cdot h_1^2 + \theta_{12}^4 \cdot g'(z_2^4) \cdot \theta_{21}^3 \cdot g'(z_1^3) \cdot h_1^2 \\ + \theta_{13}^4 \cdot g'(z_3^4) \cdot \theta_{31}^3 \cdot g'(z_1^3) \cdot h_1^2 + \theta_{14}^4 \cdot g'(z_4^4) \cdot \theta_{41}^3 \cdot g'(z_1^3) \cdot h_1^2$$

$$\frac{\partial z_5}{\partial \theta_{14}^2} = \theta_{11}^4 \cdot g'(z_1^4) \cdot \theta_{11}^3 \cdot g'(z_1^3) \cdot h_4^2 + \theta_{12}^4 \cdot g'(z_2^4) \cdot \theta_{21}^3 \cdot g'(z_1^3) \cdot h_4^2 \\ + \theta_{13}^4 \cdot g'(z_3^4) \cdot \theta_{31}^3 \cdot g'(z_1^3) \cdot h_4^2 + \theta_{14}^4 \cdot g'(z_4^4) \cdot \theta_{41}^3 \cdot g'(z_1^3) \cdot h_4^2$$

$$\frac{\partial z_5}{\partial \theta_{40}^2} = \theta_{11}^4 \cdot g'(z_1^4) \cdot \theta_{14}^3 \cdot g'(z_4^3) \cdot 1 + \theta_{12}^4 \cdot g'(z_2^4) \cdot \theta_{24}^3 \cdot g'(z_4^3) \cdot 1 \\ + \theta_{13}^4 \cdot g'(z_3^4) \cdot \theta_{34}^3 \cdot g'(z_4^3) \cdot 1 + \theta_{14}^4 \cdot g'(z_4^4) \cdot \theta_{44}^3 \cdot g'(z_4^3) \cdot 1$$

$$\frac{\partial z_5}{\partial \theta_{44}^2} = \theta_{11}^4 \cdot g'(z_1^4) \cdot \theta_{14}^3 \cdot g'(z_4^3) \cdot h_4^2 + \theta_{12}^4 \cdot g'(z_2^4) \cdot \theta_{24}^3 \cdot g'(z_4^3) \cdot h_4^2 \\ + \theta_{13}^4 \cdot g'(z_3^4) \cdot \theta_{34}^3 \cdot g'(z_4^3) \cdot h_4^2 + \theta_{14}^4 \cdot g'(z_4^4) \cdot \theta_{44}^3 \cdot g'(z_4^3) \cdot h_4^2$$

$$\dots \dots \dots \\ \Rightarrow \frac{\partial z_5}{\partial \theta^2} = \theta^3(:, z-5)' \cdot g'(z^3) \cdot (\theta^4(z-5)' \cdot g'(z^4)) \cdot [1 \ h^2']$$

P3

$$\Rightarrow \frac{\partial J}{\partial \theta^2} = \frac{1}{m} \sum^5 \delta^5 \cdot g'(z^5) \cdot (\theta^3(:, z=5))' \cdot g'(z^3) \cdot (\theta^4(z=5))' \cdot g'(z^4) \cdot [1 \ x']$$

In the same principle,

$$\Rightarrow \frac{\partial J}{\partial \theta^1} = \frac{1}{m} \sum^5 \delta^5 \cdot g'(z^5) \cdot (\theta^2(:, z=5))' \cdot g'(z^2) \cdot (\theta^3(:, z=5))' \cdot g'(z^3) \cdot (\theta^4(z=5))' \cdot g'(z^4) \cdot [1 \ x']$$

P4.