Proceedings of the 6th JFPS International
Symposium on Fluid Power, TSUKUBA 2005
November 7-10, 2005

A NEW METHOD TO PREDICT THE OCCURRENCE OF STICK-SLIP IN PNEUMATIC CYLINDERS

Bai-hai ZHANG, Yan-feng MA, Hai-feng CHENG and Guang-zheng PENG

SMC-BIT Pneumatics Center, Department of Automatic Control,
School of Information Science and Technology,
Beijing Institute of Technology
Beijing 100081, China
(E-mail: smczhang@bit.edu.cn)

ABSTRACT

Pneumatic cylinders can not usually retain steady motion when a low speed is required because of the stick-slip effect. Stick-slip is a kind of motion in which the piston stops its motion intermittently because of the friction force. In this paper a neural network (NN) classifier is set up as a new method to predict the occurrence of stick-slip by different dynamic working parameters.

The classifier is made of an NN, and the training samples come from the simulation. It classifies different working conditions into two groups: stick-slip will occur and stick-slip won’t occur, without concerning the process and the model. The NN is based on a back-propagation (BP) neural network. The input variables may be single variables of the dynamic working parameters or the combinations of them and the output variables are the two groups.

KEY WORDS

Pneumatic cylinder, Stick-slip, Neural network, Continuous genetic algorism, Classifier

INTRODUCTION

Pneumatic cylinders play a key role in pneumatic systems. They transform the energy of compressed air into mechanical energy. The steady motion of pneumatic cylinders directly affect whether the driving system works regularly, therefore it is necessary to estimate the occurrence of stick-slip in pneumatic cylinders.

The structure of the pneumatic system is illustrated in Figure 1. It is a meter out circuit. The system is composed of the following parts: air source, a directional control valve, one meter-out speed control valve, a pneumatic cylinder and load mass. In this paper, we just consider the push out process of the cylinder.

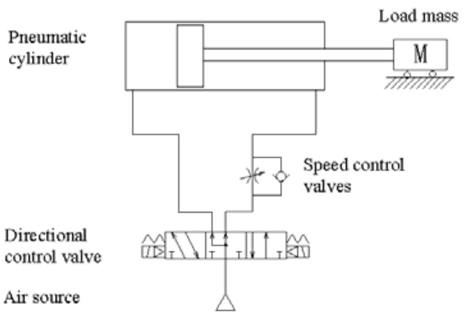

Figure 1   Pneumatic driving system by meter out circuit

We use 20-sim and bond graph as the modeling and simulation tool in our work [1,2].

The criterion of the occurrence of stick-slip in the pneumatic cylinder is whether the piston velocity reaches 0. Since the stick-slip motion has a correlation with many potential pneumatic conditions such as friction forces, air source pressure, lubrication condition, temperature, system stiffness, it is difficult to build a comprehensive fundamental stick-slip model. A black box model, NN is used to classify whether stick-slip motion will occur or not under certain working parameter settings. With this method, input and output variables can be related without any knowledge or assumptions about the mathematical representation. Many experiments and simulations on pneumatic cylinders have been done before and are found in good accordance, so that simulation can make an excellent data set for use with NN as training samples.

Of the many possible NNs, BP is one of the most commonly used NN architectures for classification. In this study, BP is used to classify whether the stick-slip motion will occur or not. The performance of the BP network is affected by the architecture and parameter settings of the network [3]. Users must therefore determine the required numbers of optimal layers, neurons in the hidden layers, weights of the NN and bias of the neurons. However, there has been no clearly defined theory for calculating the ideal parameter settings, and even slight parameters' changes can cause major variations in the behaviors of almost all networks [4]. In this study, in order to optimize the architecture of NN, continuous genetic algorithm is employed to determine the weights and bias.

1. NN INCORPORATING GENETIC ALGORITHMS IN STICK-SLIP PREDICTION

The architecture of NN has considerable influence on their performance. For multi-layered BP, the first thing is to decide the number of layers. The architecture of three or four layers is commonly used. There is also a two-layered architecture whose input layer is connected with the output layer directly. This architecture is used only when the input model is similar to the output model. Otherwise if the input model is different from the output model, hidden layers should be added between them to transmit the information which can make the nonlinear classification problem in the input space be transformed into linear classification in the hidden space. The more hidden layers are in the NN the more effect on the information transmission. However it is not the more hidden layers the better NN. There are two reasons. The first reason is that as the increasing of the number of hidden layers it brings increasing computation of the error's back propagation, and the training time increases sharp. The second reason is that as the increasing of the number of hidden layers local minimal error increases. It is easy to be lost in the local minimal value during the training process.

Based on the above analysis a three-layered architecture with one input layer, one hidden layer and one output layer is chosen in this study. The number of neurons in the input layer, hidden layer and output layer are set at 4, 3(4, 5, 6, 7), and 1, respectively [5]. These numbers of neurons in the hidden layer n_h is set by Eq. 1.

$$n_h = sqrt\left(n_i + n_o\right) + n \qquad (1)$$

n_i is the number of neurons in the input layer; n_o is the number of neurons in the output layer; n is an integer and its range is 0~7. The architecture of the NN is shown in Figure 2.

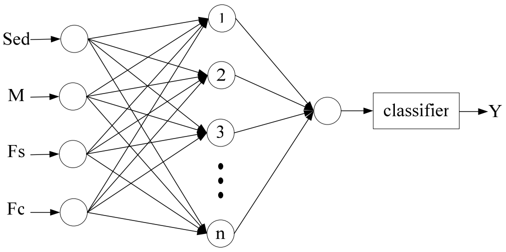

Figure 2 Architecture of the neural network

In the input layer, each neuron corresponds to a working parameter. In the hidden layer and the output layer each neuron's output is calculated using Eq. (2).

$$net = x_1\omega_1 + x_2\omega_2 + \cdots + x_n\omega_n + bias_i \qquad (2)$$

Usually a neuron's output is modified by the transfer function. The sigmoid function as shown in Eq. (3) is used for this transfer function.

$$f\left(net\right) = \frac{1.0}{1.0 + e^{-net}} \qquad (3)$$

Here x_i is the value of the input variable, ω_i is the connection weight between the input and hidden neurons and between the hidden and output neurons, $bias_i$ is bias term for the hidden and output neurons, respectively.

In Fig.1, a classifier is used to classify the output of the NN into two types:

$$Y = f(x) = \begin{cases} 1 & x \geq 0.5 \\ 0 & x < 0.5 \end{cases} \qquad (4)$$

Here $Y=1$ means the stick-slip motion will occur, otherwise won't occur, x represents the output of the output layer.

2. OPTIMIZING NN'S PARAMETERS BY USING GENETIC ALGORITHMS

A continuous genetic algorism application process for optimizing the parameters (the weights and the biases) of NN is used. This will avoid being lost in the local minimal value during the searching process. The continuous GA has the advantage of requiring less storage than the binary GA because a single floating-point number represents the variable instead of *N* bits binary number. The continuous GA is also inherently faster than the binary GA, because the chromosomes do not have to be decoded prior to the evaluation of the cost function [6].

The momentum of the chromosomes has a lower bound of 0 and an upper bound of 1. The number of chromosomes (population size) and the number of generations are set to 100 respectively. Every chromosome in a generation evolves into new chromosomes for 100 generations unless the break condition is satisfied.

The flow chart of the NN classifier can be expressed in Figure 3.

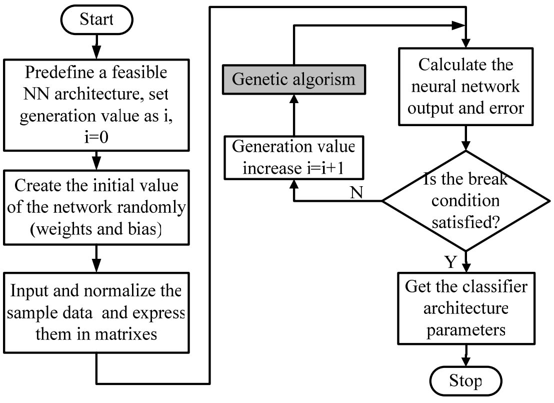
Start
Predefine a feasible NN architecture, set generation value as i, i=0
Create the initial value of the network randomly (weights and bias)
Input and normalize the sample data and express them in matrixes
Genetic algorism
Generation value increase i=i+1
Calculate the neural network output and error
Is the break condition satisfied?
N
Y
Get the classifier architecture parameters
Stop

Figure 3 The flow chart of the NN classifier

It is reasonable to express a continuous variable by a float-point number, so the *m* chromosomes with *n* vectors can be expressed as a matrix named *A* with *m* rows and n columns. a_{ij} is the j^{th} variables of the i^{th} chromosome.

The basic algorithms in continuous variables GA can be expressed as follows:

2.1. CROSS-OVER

Choose the k^{th} and the l^{th} chromosomes in the *m* chromosomes group stochastically. The cross-over is realized through Eq. (5) and Eq. (6):

a_{kj} is variable of the k^{th} chromosomes

$$\xrightarrow{\text{Chiasma}}$$

$$a_{kj}^{'} = a_{kj}(1-\beta_j)+\beta_j a_{lj}\,(j=1\sim n) \tag{5}$$

a_{lj} is variable of the l_{th} chromosomes

$$\xrightarrow{\text{Chiasma}}$$

$$a_{lj}^{'} = a_{lj}(1-\beta_j)+\beta_j a_{kj}\,(j=1\sim n) \tag{6}$$

Here $0<\beta_j<1$ is the cross-over parameter, and a_{kj}' and a_{lj}' present the new values after cross-over. Since cross-over is the information exchange of the chromosomes, the sum of the variables in the cross-over chromosomes keeps the same value as shown in Eq. (7):

$$a_{kj} + a_{lj} = a_{kj}^{'} + a_{lj}^{'}\,(j=1\sim n) \tag{7}$$

Different from the scattered variables' cross-over, the continuous variables' cross-over exchange every variable in the chromosome. It is similar to the n section exchange in binary variables' cross-over which will enrich the exchange information.

2.2. MUTATION

Choose *m* variables in the chromosome group, and choose $0<\xi,\ \eta<1$ stochastically, operate the mutation according to Eq. (8):

$$\begin{cases} a_{ij}^{''} = a_{ij} - (a_{ij} - \underline{a}_j)\eta & 0<\xi\le 0.5 \\ a_{ij}^{''} = a_{ij} + (a_{ij} - \bar{a}_j) & 0.5<\xi<1 \end{cases} \tag{8}$$

Here $\underline{a}_j$ and $\bar{a}_j$ is the lower limit and the upper limit of the j^{th} variable, a_{ij}' is the new value after mutation. The mutation variables get increased when $0<\xi\leqslant 0.5$ and decreased when $0.5<\xi<1$. And the extent is controlled by η.

Generalize the continuous GA's step as follows:

1) Build the initial chromosome group as mother generation, named as array *A*. The row number is *m*, which will affect the searching result. In this study, *m* is set to 100.
2) Extend the chromosome group to 2 times. *m* variables are chosen from the *2m* variables to do cross-over stochastically. n variables are chosen to do mutation stochastically. *n* is 1% of the whole variables.
3) Connect the original array and the new array after mutation, which is 3 times extension of the original array. Calculate the NN output and error.

Choose the variables corresponding to the *m* least errors as the new mother generation.

4) Finish propagates. There are 3 conditions to break the evolutional cycle. Firstly, when the error gets 0 the cycle will break. Secondly, if it has reached 100 generations the cycle will break. Thirdly, if the NN output error hasn't improved for more than 10 generations, the cycle will break and return the optimal value. Otherwise return to step 2.

The continuous variables genetic algorism shown in the Figure 3 is expressed in Figure 4.

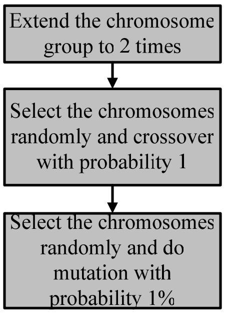
Extend the chromosome group to 2 times
Select the chromosomes randomly and crossover with probability 1
Select the chromosomes randomly and do mutation with probability 1%

Figure 4 The continuous variables genetic algorism

3. SAMPLE PREPARATION

Stick-slip is a kind of motion in which a body stops its motion momentarily or definitively because of the action of friction force. The friction force depends on a number of factors, such as lubrication condition, the velocity of pneumatic cylinders and the load mass. In order to derive how the parameters affect the stick-slip, the model of a pneumatic system by meter out circuit is built by using bond graph method. Simulations are carried out with different working parameters such as the friction characteristics, the effective area of the outlet speed control valve and load mass [1]. The training samples used in the NN come from the simulations of the stick-slip in pneumatic system. Simulations are carried out by running multiple simulations with parameter sweep.

Through the simulations under different working parameters, stick-slip will occur or not. It can be seen that the different working parameters such as friction characteristics, the effective area of the outlet speed control valve and load mass have some nonlinear relations with stick-slip occurrence in pneumatic cylinders. The working parameters or their combinations can be set as the input parameters of input layer of the NN. Whether the stick-slip will occur is set as the output of output layer of the NN. To realized this, a new variable Y is set as the output to express the state of stick-slip. If the stick-slip occurs, *Y=1*; otherwise, *Y=0*. In this study, four input variables: the effective area of the outlet speed control valve *Sed*, load mass *M* , the maximum static friction force *Fs* and coulomb friction force *Fc* are set as the input of NN, so that there are totally four neurons in the input layer. If more working parameters or their combinations are expected, the neurons in the input layer should be increased, and the neurons in the hidden layer should also be changed accordingly.

100 training samples are used to train the NN. 40 test samples are used to test the training result. They both come from the simulation of stick-slip. Table 1 shows part of the samples of the parameters extracted from the simulation results.

Table 1 Extracted parameters

	Sed(m^2)	*M*(kg)	*Fs*(N)	*Fc*(N)	*Y*
1	1.0e-8	8	18	13	0
2	1.0e-8	20	28	7	1
3	1.5e-8	12	24	16	0
4	2.0e-8	20	36	10	1
5	4.0e-8	40	36	10	0

Parameters dimension in the first column of table 1 *Sed* is much smaller than the others. The weights and the biases of the network are extracted from a generation's array of stochastic numbers whose elements are uniformly distributed in the interval (0, 1). Then the effect brought by *Sed* will be much less than the other parameters, which will decrease the precision of the output or lead to the float of the optimize value. Hence the samples should be preprocessed by normalizing the inputs and outputs so that they have means of 0 and standard deviations of 1 before training.

4. TRAINING RESULT

The training process for the NN classifier to predict the occurrence of the stick-slip is carried out when the number of neurons in hidden layer is 3, 4, 5, 6 and 7 respectively. The result shows that when there are 3 neurons in the hidden layer optimal result can be found with the least time. The optimal architecture of NN is build by one hidden layer with 3 neurons. The NN's parameters which are trained by the optimal architecture are used to test the 40 test samples. Figure 5 shows the test result of the NN classifier.

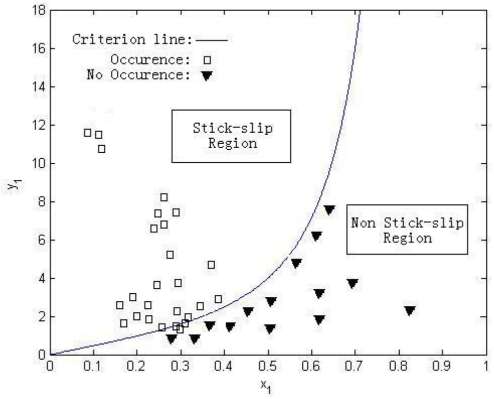

Figure 5 The result of the classifier

In Figure 5, x_1 axis and y_1 axis represents two dimensionless values calculated through the model of pneumatic system, which is presented in reference [7]. The curve in Figure 5 is the criterion curve which separates the whole area into stick-slip region and non stick-slip region. Test samples are marked in Figure 5. It shows that most test samples located in the right region. The error rate of the classifier is less than 4%.

CONCLUSION

A classifier made of the NN based on continuous GA is realized to predict the occurrence of stick-slip in a pneumatic cylinder. Continuous GA ensures the precision of the classifier. To complete the training process, simulations are carried out with different working parameters to acquire the training and test samples. The prediction results of the classifier are in agreement with former analytical solution. The error rate of the classifier is acceptable.

REFERENCES

1. Zhang, B.H. and Cheng, H. F.,Bond Graph Modeling and Simulation of Stick-slip Characteristics of a Pneumatic System, The Sixth International Conference on Fluid Power Transmission and Control, 2005,pp.569-572.
2. Controllab Products B.V. Getting Started with 20-sim 3.4, Controllab Products B. V., Enschede, 2003
3. Kim, G.H., Yoon, J.E., An S.H., Cho, H.H. and Kang, K. I., Neural Network Model in Cooperating a Genetic Algorithm in Estimating Construction Costs, Building and Environment, 2004, **39** , pp.1333-1340.
4. Finnie, G.R., Witting, G.E. and Desharnais, J.M., A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-based Reasoning and Regression Models, Systems and Software, 1997, **39**-3, pp.281-289.
5. Cui, R., Hong, B. ,On Constructing a Hidden Layer for Three-layered Feed forward Neural Networks ,Computer Research and Development, 2004, **41**-4.
6. Haupt, R., Haupt, S. Practical Genetic Algorithms 2nd Edition, John Wiley & Sons, Inc., New Jersey, 2004,pp.51-64.
7. Tokashi, L.R.,Fujita, T.,Kagawa,T. and Ikegami,T., Stick-slip Motion in Pneumatic Cylinders Driven by Meter Out Circuit, Transactions of the Japan Hydraulics & Pneumatics Society, 1999,**30**-4, pp.22~29.