UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
Más que un estado sólido
https://revistas.udistrital.edu.co/index.php/visele

VISIÓN ELECTRÓNICA
A RESEARCH VISION

Optimization of a linear controller using dynamic back-propagation

Optimización de un controlador lineal empleando dynamic back-propagation

Helbert Eduardo Espitia-Cuchango[1], ***Iván Machón-González***[2], ***Hilario López-García***[3]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 21/02/2019
Recibido: 16/03/2019
Aceptado: 02/05/2019

Keywords:
Controller
DC motor
Dynamic
Optimization

ABSTRACT

This paper presents the optimization of a linear controller for a DC motor using the dynamic back-propagation algorithm. This algorithm is commonly employed for neural networks training; however, it can be used for optimization of a linear controller. The results show a satisfactory controller optimization.

Palabras clave:
Controlador
Motor DC
Dinámico
Optimización

RESUMEN

En este documento se presenta la optimización de un controlador lineal para un motor DC mediante el algoritmo "*Dynamic Back-Propagation*". Este algoritmo es comúnmente utilizado para el entrenamiento de redes neuronales, sin embargo, puede ser empleado para la optimización de un controlador lineal. Los resultados muestran que la optimización del controlador es satisfactoria.

[1]BSc. In Electronic engineering, PhD. In Computer and systems engineering. Current position: Professor at Universidad Distrital Francisco José de Caldas, Bogotá, Colombia. E-mail: heespitiac@udistrital.edu.co.
[2]BSc. In Industrial engineering, PhD. In Industrial engineering. Current position: Professor at Universidad de Oviedo, Gijón, España. E-mail: machonivan@uniovi.es
[3]BSc. In Industrial engineering, PhD. In Industrial engineering. Current position: Professor at Universidad de Oviedo, Gijón, España. E-mail: hilario@uniovi.es

Citar este artículo como: H. E. Espitia-Cuchango, I. Machón-González and H. López-García, "Optimization of a linear controller using dynamic back-propagation", Visión electrónica, algo más que un estado sólido, vol. 13, no. 2, p.p. 273-279 july-december 2019, https://doi.org/10.14483/22484728.15183

1. Introduction

In general, many applications like robotics, servomechanisms, control, and automation normally use motors with Direct Current (DC). For instance, an application for implementing a solar tracker with a DC motor can be found in [1] and [2]. On the other hand, the dynamic back-propagation (DBP) algorithm is used for training neural networks by incorporating the same characteristics as those of dynamic systems [3, 4]. Such an algorithm is suitable in the application of the chain rule of the descending gradient method characterized for having a fast convergence rate [4].

In addition to using the DBP algorithm, [5] presents an adaptive processing system consisting of a digital filter based on a neural network. This work focuses on the online training algorithms to achieve an association between the characteristics of the input signal of the neural network and the dynamic responses of the digital filter. For this, a DBP algorithm is developed to train the closed loop network between the output of the digital filter and the inputs to the neural network.

In contrast, [6] describes a single input-output adaptive neuronal network controller scheme and the training algorithm. For the implementation of the system, a modification of the traditional back-propagation algorithm is developed. The proposal is made considering applications for time-variant systems.

A paper that considers the dynamic Lyapunov stability of the neural network during the training process is [7]. In this work, to avoid unstable phenomena during the learning process, multiplier and restricted rate learning schemes are proposed. With the multiplier method, explicit stability conditions are introduced in the iterative error index and the update equations contain a set of inequality constraints. With the restricted learning rate algorithm, rates are updated at each iterative instant by an equation derived from the stability conditions.

According to [8], adaptive control with reference model is commonly used in the design of controllers based on traditional neural networks, where it often requires a plant emulator when the neural controller is connected to the plant. In this work, the authors propose a plant emulator using a neuro-fuzzy system and a variation of the DBP algorithm to train the neuronal controller. The system is employed to the control of a DC-to-DC converter.

On the other hand, [9] presents a general framework for dynamic neural networks by reviewing two general algorithms for the calculation of gradients and Jacobians for these dynamic networks: backward propagation through time (BPTT) and real time recurrent learning (RTRL). The results show that the BPTT algorithm is more efficient for gradient calculations, while the RTRL algorithm is more efficient for Jacobian calculations.

Finally, [10] reviews the different techniques for the training of neural networks, particularly the DBP algorithm for recurrent neural networks.

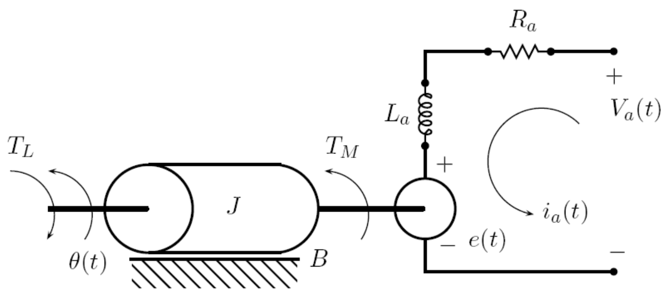
This paper presents linear controller optimization for a DC motor through the DBP algorithm. A model of the plant is obtained as a transfer function that later is transformed into discreet time by implementing the optimization algorithm to determine the result of the process, where the simulation shows a satisfactory controller optimization.

This document is organized as follows: first, the dynamic model for DC motors and the DBP algorithm theory are revised; second, the architecture of the implemented controller is also reviewed. The subsequent section is related to the model for the motor in discreet time which is used for implementing the DBP algorithm. Lastly, results and conclusions are discussed.

2. DC motor model

The model of the motor mainly consists of a mechanical and an electrical part [11], as shown in Figure 1.

Figure 1: DC Motor scheme. Source: adapted from [11].



For the model in Figure 1, the motor torque is proportional to the armature current as presented in equation (1).

$$T_M = K_T i_a(t) \tag{1}$$

The counter-electromotive force is proportional to the motor angular velocity, according to equation (2).

$$e(t) = K_e\omega(t) = K_e\frac{d\theta}{dt} \tag{2}$$

Equation (3) represents the electrical component.

$$V_a(t) - e(t) = L_a\frac{di_a(t)}{dt} + R_a i_a(t) \tag{3}$$

The mechanical component is given by equation (4).

$$T_M(t) - T_L(t) = J\frac{d^2\theta(t)}{dt^2} + B\frac{d\theta(t)}{dt} \tag{4}$$

A complete plant dynamic model may be impractical in some applications given the difficulty in establishing the parameters. An alternative approach consists of a simplified model via plant parameter identification [12].

Equation (5) describes the plant simplified transfer function corresponding to a first-order system with an integrator, where the variable s is associated with the Laplace transform.

$$G(s) = \frac{K}{s(\tau \cdot s + 1)} \tag{5}$$

The parameter identification to this plant is performed considering time and velocity average values as shown in [1] and [12].

In equation (5), K corresponds to the input-output relation in stable state that for the specific case corresponds to $K = \Delta\omega/\Delta V$, where ω is the angular velocity; with a 12- voltage power supply, the average time taken by the system to rotate an angle of $\pi/2$ is 0.06 seconds; thus, the corresponding K parameter value is 0.005 rad/(Vsec) [12].

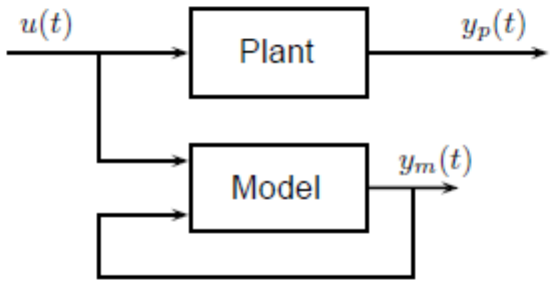
The parameter τ can be defined as a quarter part of the time to obtain a steady-state output, which is 1.53 seconds; thus, for τ, 0.38 seconds are taken [12]. In this way, the model of the plant is given by equation (6).

$$G(s) = \frac{0{,}005}{s(0{,}38s + 1)} \tag{6}$$

3. Dynamic Back-Propagation Algorithm

This algorithm is used in neural networks for identification and control of dynamic systems, specifically when having a parallel-type scheme [13]. Figure (2) provides a representation of this architecture for identification process.

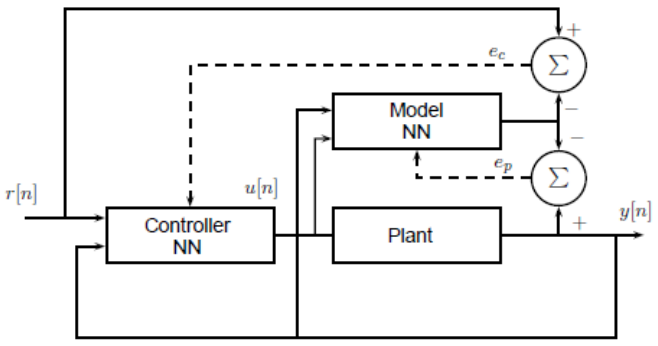
Figure 2: Parallel identification scheme. Source: adapted from [13].



The parallel identification scheme uses plant inputs and the output feedback from the network itself.

In control applications, one of the schemes of neural networks uses one network to model the plant and another to model the controller. This approach allows the plant identification first, and later the controller training is performed by aiming the output of the system to follow the reference [3]. Figure (3) displays the two neural networks used in this process.

Figure 3: Control scheme using neural networks. Source: adapted from [3].



Thus, for the plant model, there is an input $u[n]$ and output $y[n]$ such that the result is a structure given by equation (7).

$$y[n] = f_p(y[n-1], y[n-2], \ldots, y[n-p], u[n], u[n-1], \ldots, [n-q]) \tag{7}$$

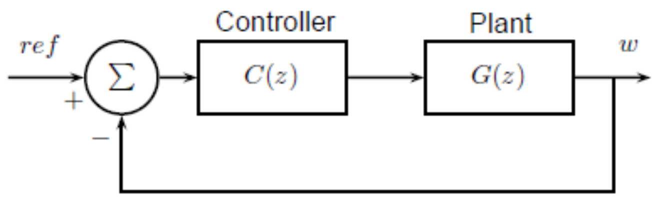
Meanwhile, the inputs for the controller are given by the reference $r[n]$ and the process signal is measured by $y[n]$, while the output is given by the control action $u[n]$ as shown in equation (8), where p, q and m corresponds to the number of outputs, inputs and reference delays, respectively. Typically, the number of delays increases according to the order in the plant [3].

$$u[n] = f_c(y[n], \dots, y[n-p], r[n], ..., r[n-m], u[n], ..., u[n-q]) \tag{8}$$

4. Controller architecture

Figure (4) represents the considered control scheme; both the plant and the controller are discrete-time models.

Figure 4: System simplified architecture.



Source: own.

In discrete time, the transfer function of the controller is given by equation (9).

$$C(z) = \frac{U(z)}{E(z)} = \frac{B_0 + B_1 z^{-1} + B_2 z^{-2}}{1 + A_1 z^{-1} + A_2 z^{-2}} \tag{9}$$

The controller difference equation is (10). Under this approach, the general action of the controller corresponds to equation (11).

$$u[n] = B_0 e[n] + B_1 e[n-1] + B_2 e[n-2] - A_1 u[n-1] - A_2 u[n-2] \tag{10}$$

$$u[n] = f_c(e[n], ..., e[n-N_e], u[n-1], ..., u[n-N_u], H_c) \tag{11}$$

Similarly, the output for the controller training is given by equation (12).

$$y[n] = f_p(y[n-1], ..., y[n-N_y], u[n], ..., u[n-N_u], H_p) \tag{12}$$

Here, N_y corresponds to the total number of output samples, N_u the number of samples in the inputs, and H_p the parameters vector of the plant model. Meanwhile, the controller parameters set is given by equation (13).

$$H_c = [B_0, B_1, B_2, A_1, A_2] \tag{13}$$

The adaptation (optimization) of parameters in the controller is given by equation (14) using a learning rate η.

$$H_c(k+1) = H_c(k) - \eta \frac{\partial J(k)}{\partial H_c(k)} \tag{14}$$

The adjustment function J used in equation (14) is defined by (15). The variation of J with respect to the controller parameters can be calculated using equation (16).

$$J = \frac{1}{2} P(r[n] - y[n])^2 + Q(u[n])^2 \tag{15}$$

$$\frac{\partial J}{\partial H_c[n]} = \frac{\partial J}{\partial y[n]} \frac{\partial y[n]}{\partial H_c[n]} \tag{16}$$

5. Plant implementation in discreet time

The transfer function of the plant is described by equation (17).

$$G(s) = \frac{0{,}005}{s(0{,}38s + 1)} \tag{17}$$

Thus, transforming this transfer function to discreet time considering a sampling time of $T_s = 0{,}1s$, and using the method of bilinear transformation, equation (18) is obtained. In general, this transfer function can be represented using equation (19).

$$G(z) = 10^{-5} \frac{2{,}907z^2 + 5{,}814z + 2{,}907}{z^2 - 1{,}767z + 0{,}7674} \tag{18}$$

$$G(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{19}$$

Finally, expression (20) represents the difference equation of the plant, which is utilized to establish the parameter expressions to perform the controller training.

$$y[n] = b_0 u[n] + b_1 u[n-1] + b_2 u[n-2] - a_1 y[n-1] - a_2 y[n-2] \tag{20}$$

6. Equations in discrete time to optimization algorithm implementation

Implementing the DBP algorithm requires the difference equations of each parameter to be used in equations (14), (15), and (16). In [14], the application of this algorithm for the training of neural networks can be seen.

First, using (20), the error equation is (21).

$$e[n] = r[n] + a_1 r[n-1] + a_2 r[n-2] - a_1 e[n-1] - a_2 e[n-2] - b_0 u[n] - b_1 u[n-1] - b_2 u[n-2] \tag{21}$$

Meanwhile, the controller difference equation is (22).

$$u[n] = -A_1 u[n-1] - A_2 u[n-2] + B_0 e[n] + B_1 e[n-1] + B_2 e[n-2] \tag{22}$$

Consequently, the training equations of the parameter A_1 correspond to (23).

$$\begin{aligned}\frac{du}{dA_1}[n] &= -u[n-1] - A_1\frac{du}{dA_1}[n-1] - A_2\frac{du}{dA_1}[n-2] + B_0\frac{de}{dA_1}[n] + B_1\frac{de}{dA_1}[n-1] + B_2\frac{de}{dA_1}[n-2]\\ \frac{de}{dA_1}[n] &= -a_1\frac{de}{dA_1}[n-1] - a_2\frac{de}{dA_1}[n-2] - b_0\frac{du}{dA_1}[n] - b_1\frac{du}{dA_1}[n-1] - b_2\frac{du}{dA_1}[n-2]\end{aligned} \tag{23}$$

Similarly, for parameter A_2, the training equations are given by (24).

$$\begin{aligned}\frac{du}{dA_2}[n] &= -u[n-2] - A_1\frac{du}{dA_2}[n-1] - A_2\frac{du}{dA_2}[n-2] + B_0\frac{de}{dA_2}[n] + B_1\frac{de}{dA_2}[n-1] + B_2\frac{de}{dA_2}[n-2]\\ \frac{de}{dA_2}[n] &= -a_1\frac{de}{dA_2}[n-1] - a_2\frac{de}{dA_2}[n-2] - b_0\frac{du}{dA_2}[n] - b_1\frac{du}{dA_2}[n-1] - b_2\frac{du}{dA_2}[n-2]\end{aligned} \tag{24}$$

For parameter B_0, the training equations are (25).

$$\begin{aligned}\frac{du}{dB_0}[n] &= e[n] - A_1\frac{du}{dB_0}[n-1] - A_2\frac{du}{dB_0}[n-2] + B_0\frac{de}{dB_0}[n] + B_1\frac{de}{dB_0}[n-1] + B_2\frac{de}{dB_0}[n-2]\\ \frac{de}{dB_0}[n] &= -a_1\frac{de}{dB_0}[n-1] - a_2\frac{de}{dB_0}[n-2] - b_0\frac{du}{dB_0}[n] - b_1\frac{du}{dB_0}[n-1] - b_2\frac{du}{dB_0}[n-2]\end{aligned} \tag{25}$$

For parameter B_1, the training equations are given by (26).

$$\begin{aligned}\frac{du}{dB_1}[n] &= e[n-1] - A_1\frac{du}{dB_1}[n-1] - A_2\frac{du}{dB_1}[n-2] + B_0\frac{de}{dB_1}[n] + B_1\frac{de}{dB_1}[n-1] + B_2\frac{de}{dB_1}[n-2]\\ \frac{de}{dB_1}[n] &= -a_1\frac{de}{dB_1}[n-1] - a_2\frac{de}{dB_1}[n-2] - b_0\frac{du}{dB_1}[n] - b_1\frac{du}{dB_1}[n-1] - b_2\frac{du}{dB_1}[n-2]\end{aligned} \tag{26}$$

Finally, the training equations of the parameter B_2 correspond to (27).

$$\begin{aligned}\frac{du}{dB_2}[n] &= e[n-2] - A_1\frac{du}{dB_2}[n-1] - A_2\frac{du}{dB_2}[n-2] + B_0\frac{de}{dB_2}[n] + B_1\frac{de}{dB_2}[n-1] + B_2\frac{de}{dB_2}[n-2]\\ \frac{de}{dB_2}[n] &= -a_1\frac{de}{dB_2}[n-1] - a_2\frac{de}{dB_2}[n-2] - b_0\frac{du}{dB_2}[n] - b_1\frac{du}{dB_2}[n-1] - b_2\frac{du}{dB_2}[n-2]\end{aligned} \tag{27}$$

Table 1: Controller parameter values.

Configuration	B_0	B_1	B_2	A_1	A_2
Initial	10	0	0	0	0
Final	10.0314	0.0306	0.0297	−0.4702	−0.4339

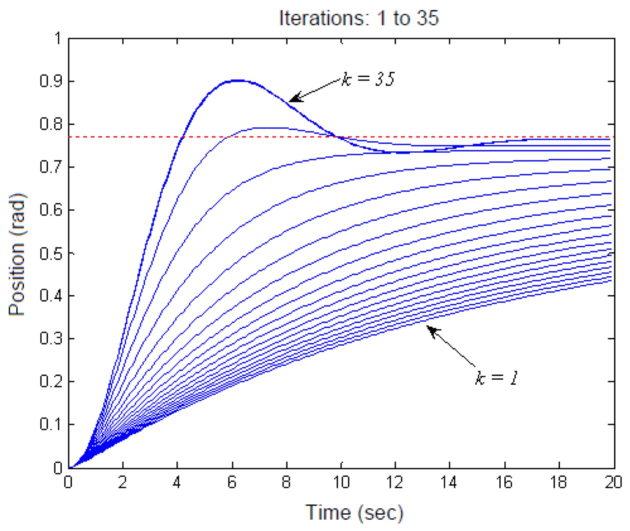
Source: own.

7. Results

The reference of the system is taken for 45° which is an angular position of $\pi/4$ radians. The learning rate used is $\eta = 0{,}01$ and 35 iterations are performed for the training process of the controller parameters. A proportional controller behavior is considered as initial configuration; therefore, its parameters are assigned values of zero excluding B_0, which, following previous experimentation, is set to 10. Initial and final (optimized) values of the controller parameters can be seen in Table 1.

Figure (5) presents different responses of the control system during the training process. This figure also shows the system output getting closer to the reference. Meanwhile, Figure (5) displays the values obtained from the objective function for each iteration; this demonstrates that the objective function tends toward no variation after iteration 20.

Figure 5: Responses of the system during the training process.

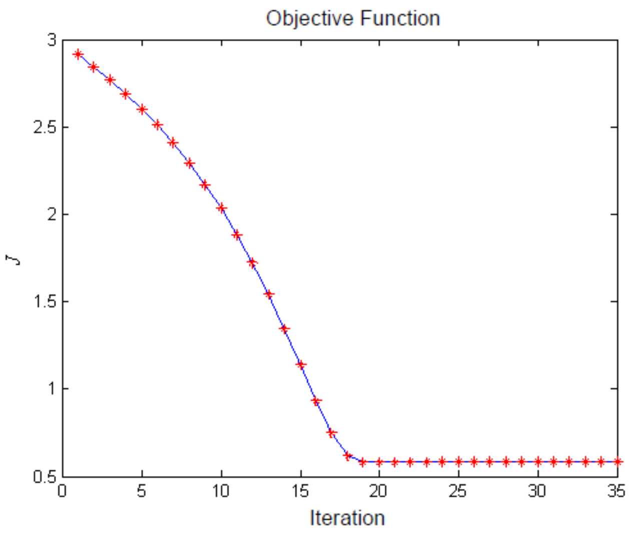


Source: own.

8. Conclusions

It is clear that the DBP algorithm, which is conventionally used for the training of neural networks, can also be used to optimize a linear controller. This mechanism offers an alternative for controller tuning in both linear and non-linear controllers and can also be used for schemes of supervised control. The obtained results are satisfactory in showing the response of the system during the training process as well as the evolution of the objective function. This technique for a neuro-fuzzy type controller training will be implemented in a follow-up paper.

Figure 6: Objective function values during the training process.



Source: own.

References

[1] H. Espitia and F. Sierra, “Optimización de un controlador análogo para un seguidor solar empleando algoritmos genéticos y enjambres de partículas”, IEEE SIFAE Simposio Internacional en Fuentes Alternativas de Energía y Calidad Energética, 2012. https://doi.org/10.1109/SIFAE.2012.6478905

[2] H. Espitia and F. Sierra, “Controller optimization for a solar tracking system using differential evolution”, TECCIENCIA, vol. 10, no. 18, pp. 7-13, 2015. https://doi.org/10.18180/tecciencia.2015.18.2

[3] H. Nguyen, N. Prasad, C. Walker and F. Walker, “A First Course in fuzzy and neural control”, Washington, D.C.: CHAPMAN & HALL/CRC, 2003. https://doi.org/10.1201/9781420035520

[4] M. Singh, I. Singh and A. Verma, “Identification on Non Linear Series-Parallel Model Using Neural Network”, *MIT International Journal of Electrical and Instrumentation Engineering*, vol. 3, no. 1, pp. 21-23, 2003.

[5] J. Sztipanovits, “Dynamic backpropagation algorithm for neural network controlled

resonator-bank architecture", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* vol. 39, no. 2, pp. 99-108, 1992. https://doi.org/10.1109/82.205813

[6] Z. Qiuping, F. Zhihao and H. Yinbiao, "Dynamic back-propagation neural net for adaptive controller", *Wuhan University Journal of Natural Sciences*, vol. 3, no. 2, pp. 196-200, 1998. https://doi.org/10.1007/BF02827551

[7] L. Jin and M. M. Gupta, "Stable dynamic backpropagation learning in recurrent neural networks", *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1321-1334, 1999. https://doi.org/10.1109/72.809078

[8] S. G. Kadwane, A. Kumar and B. M. Karan, "Dynamic Back Propagation based MRAC with Fuzzy Emulator for DC-DC Converter", *Elektronika Ir Elektrotechnika*, vol. 73, no. 1, pp. 49-54, 2007.

[9] O. De Jesús and M. T. Hagan, "Backpropagation Algorithms for a Broad Class of Dynamic Networks", *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 14-27, 2007. https://doi.org/10.1109/TNN.2006.882371

[10] M. T. Hagan, H. B. Demuth, M. H. Beale and O. De Jesus, "Neural Network Design", Martin Hagan; Edition: 2, 2014.

[11] H. Espitia, L. Morales and J. Soriano, "Diseño y simulación de un controlador difuso para un motor DC basado en relaciones booleanas", *Visión Electrónica: Algo Más Que Un Estado sólido*, vol. 4, no 1, pp. 12-22, 2010. https://doi.org/10.14483/udistrital.jour.tecnura.2012.2.a02

[12] H. Espitia and J. Sierra, "Diseño e implementación de controladores análogos para un seguidor solar", *Visión Electrónica: Algo Más Que Un Estado sólido*, vol. 7, no. 1, pp. 118-135, 2013.

[13] S. Garrido, "Identificación, estimación y control de sistemas no-lineales mediante RGO", thesis PhD., Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III, Madrid, España, 1999.

[14] M. Hagan, O. De Jesús and R. Schultz, "Training recurrent networks for filtering and control", Eds. Boca Raton, FL: CRC Press, pp. 325-354, 2000.