Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 729095, 6 pages
http://dx.doi.org/10.1155/2014/729095

Research Article

The LQR Controller Design of Two-Wheeled Self-Balancing Robot Based on the Particle Swarm Optimization Algorithm

Jian Fang

Institute of Electrical Engineering, Jilin Teachers' Institute of Engineering and Technology, Changchun 130052, China

Correspondence should be addressed to Jian Fang; 757314739@qq.com

Received 25 February 2014; Revised 5 May 2014; Accepted 7 May 2014; Published 11 June 2014

Academic Editor: Her-Terng Yau

The dynamics model is established in view of the self-designed, two-wheeled, and self-balancing robot. This paper uses the particle swarm algorithm to optimize the parameter matrix of LQR controller based on the LQR control method to make the two-wheeled and self-balancing robot realize the stable control and reduce the overshoot amount and the oscillation frequency of the system at the same time. The simulation experiments prove that the LQR controller improves the system stability, obtains the good control effect, and has higher application value through using the particle swarm optimization algorithm.

1. Introduction

The two-wheeled and self-balancing robot belongs to a multivariable, nonlinear, high order, strong coupling, and unstable essential motion control system, and it is a typical device of testing various control theories and control methods; therefore, the research has great theoretical and practical significance. Because it has the advantages of simple structure, stable running, high energy utilization rate, and strong environmental adaption, it has the broad application prospects whether in the military field or in the civilian field.

Since 1980s, the scholars of various countries have conducted the system research on the two-wheeled self-balancing robot. The two-wheeled and self-balancing robot control system based on the fuzzy control can overcome the instability and nonlinear nature of the system, but it relies on the expert's experience too much [1, 2]. The optimal LQR controller is designed on the basis of establishing the system structure model; the correctness and effectiveness of the LQR controller are verified, but it is difficult to determine the weighted matrix Q and R [3–5]. The genetic algorithm is successfully applied to the parameter optimization of the LQR controller of the inverted pendulum system, and it achieves the good control effect. However, the parameters are difficult to adjust, and it is easy to fall into the local optimization [6].

This paper concerns the self-designed and two-round self-balancing robot as the research object, which uses the Newtonian mechanics equation method and the linear method near the balance point to establish the linearized mathematical model of the system. In view of the mathematical model of the system, LQR controller is designed based on the particle swarm optimization and makes full use of the searching capability of the particle swarm algorithm to optimize the matrix Q and matrix R of the LQR controller. It gains the global optimal solution of matrix Q and R of the LQR controller so as to design the optimal state feedback control matrix K and overcome the disadvantages of relying on the experience and the trial and error in the selection of matrix Q and R of the general LQR control design, making up for the inadequacy of big workload. This method has better control effect by simulation tests and comparison.

2. The Dynamics Model of the Two-Wheeled Self-Balancing Robot

The two-wheeled and self-balancing robot structure is mainly composed of the body and the two wheels, and the robot is the coaxial two wheels, driven by the independent motor; the parameters of quality, moment of inertia, and radius of

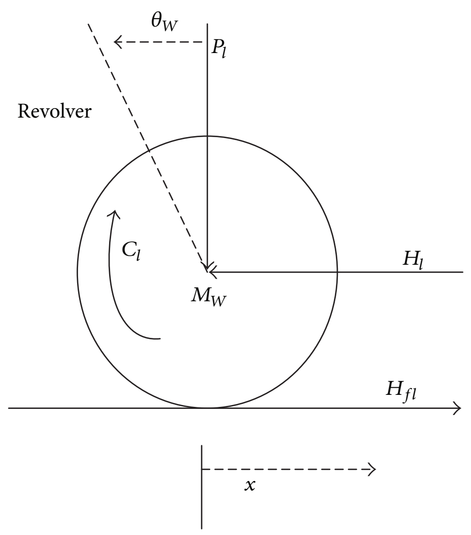

FIGURE 1: The force analysis of revolver.

the two wheels are regarded as the same, so the body center of gravity is inverted above the axletree and it keeps balance through sports.

The two-wheeled and self-balancing robot can be regarded as the vehicle-mounted inverted pendulum, so the dynamic system analysis process is more complex. This paper separates the wheel from the pendulum analysis first in the process of modeling, and then it deduces the dynamics state equation of the two-wheeled self-balancing robot through the simultaneous two parts [7, 8].

The two wheels are regarded as the research object, Figure 1 is a diagram revolver force analysis. According to the revolver force equation can be obtained in the following according to the Newton's law [9] and the rotational torque formula [10]:

$$M_w\ddot{x} = H_{fR} - H_R,$$
$$I_\omega\ddot{\theta}_\omega = C_R - H_{fR}\cdot R. \tag{1}$$

The right wheel force equation is as follows:

$$M_w\ddot{x} = H_{fL} - H_R,$$
$$I_\omega\ddot{\theta}_\omega = C_L - H_{fL}\cdot R. \tag{2}$$

After finishing, it can obtain

$$2\left(M_w + \frac{I_\omega}{R^2}\right)\ddot{x} = \frac{C_R + C_L}{R} - \left(H_R + H_L\right). \tag{3}$$

Among them, M_w is the weight of the wheel; I_ω is the moment of inertia of the wheel; R is the radius of the wheel; $\ddot{X}$ is the wheel acceleration of X axis; C_R and C_L are the right and left wheel torque; H_L and H_R are the Z axis forces of the left and right wheels with the car body; H_{fR} and H_{fL} are the interatomic forces of the right and left wheels with the ground; and θ_ω is the angle of wheel around the Z axis direction.

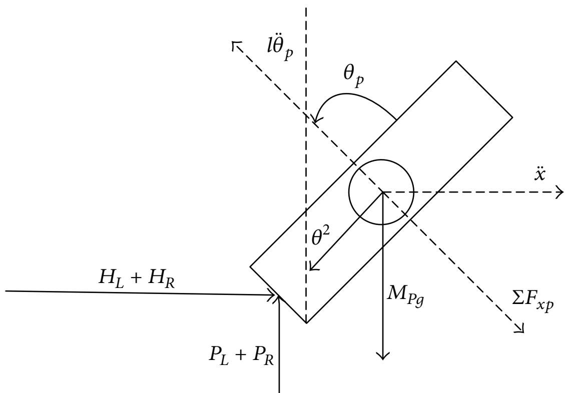

FIGURE 2: The analysis of car body force.

The car body of the two-wheeled and self-balancing robot is modelled as an inverted pendulum; the car body force analysis is shown in Figure 2. Using Newton's second law, the horizontal force is as follows:

$$\sum F_x = \left(H_R + H_L\right) - M_p l\ddot{\theta}_p \sin\theta_p - Ml\ddot{\theta}_p \cos\theta_p$$
$$= M_p\cdot\ddot{x}_p = M_p\left(x + l\sin\ddot{\theta}_p\right). \tag{4}$$

Among them, x_p is the displacement of the car body centre of gravity relative to the ground,

$$x_p = x + l\sin\theta_p. \tag{5}$$

Using Newton's second law, the force of the vertical direction of the car body is as follows:

$$\sum F_{xp} = M_p g\sin\theta - \left(P_R + P_L\right)\sin\theta_p + \left(H_R + H_L\right)\cos\theta_p$$
$$= M_p\left(x + l\sin\ddot{\theta}_p\right)\cos\theta_p. \tag{6}$$

The sum of the torques of the car body mass center is as follows:

$$\sum M_0 = I_p\ddot{\theta}_p = \frac{\left(P_R + P_l\right)}{\sin\theta_p} - \frac{\left(H_H + H_L\right)}{\cos\theta_p}. \tag{7}$$

On the small angle scope, $\theta_p^2 \approx 0$, $\sin\theta_p \approx \theta_p$, $\cos\theta_p \approx 1$, the linearized equations are gotten after the linearization process:

$$\left(M_P + 2M_W\frac{I_W}{R^2}\right)\ddot{x} = \frac{C_R + C_L}{R} - 2M_P l\theta_p,$$
$$\left(2M_P l^2 + I_p\right)\ddot{\theta}_p = M_p g l\theta_p - M_p l\ddot{x}. \tag{8}$$

Among them, θ_p is the angle of the car body deviating from the Z axis direction; I_p is the moment of inertia of the car body; M_P is the weight of the car body; and l is the height of which the car body is apart from the shaft. The output torque of the wheel is $C_R = C_L = I_R(d\omega/dt) = (k_m/R)U_a - (-k_m k_e/R)\dot{\theta}_\omega$; the state equation of the two-wheeled self-balancing robot is obtained:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta}_p \\ \ddot{\theta}_p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{2k_m k_e \left(M_p lr - I_p - M_p l^2\right)}{Rr^2 A} & \dfrac{M_p^2 g l^2}{A} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{2k_m k_e \left(rB - M_p l\right)}{Rr^2 A} & \dfrac{M_p g l \beta}{A} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta_p \\ \dot{\theta}_p \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{2k_m \left(I_p + M_p l^2 - M_p lr\right)}{RrA} \\ 0 \\ \dfrac{2k_m \left(M_p l - rB\right)}{RrA} \end{bmatrix} U_a. \quad (9)$$

Among them, $A = [I_p\beta + 2M_p l^2(M_w + (I_w/r^2))]$; $B = (2M_w + (2I_w/r^2) + M_p)$.

The output equation is as follows:

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta_p \\ \dot{\theta}_p \end{bmatrix}. \quad (10)$$

3. The LQR Controller Parameter Optimization Based on the Particle Swarm Optimization Algorithm

3.1. The Design of the Self-Balancing Robot LQR Controller. The LQR method is the most mature controller design method in the development of modern control theory [11]; LQR optimal control is to seek the control amount $u^*(t)$ to make the system reach the steady state and guarantee the performance index J to take the minimum value:

$$J = \int_0^\infty \left(X^T Q X + u^T R u\right) dt. \quad (11)$$

Among them, $u^* = -R^{-1}B^T Px = -Kx$, and P is the solution of the algebraic equation $PA + A^T P + Q - PBR^{-1}B^T P = 0$ of the matrix Riccati [12]. In (13), the matrix Q and matrix R are mutually restricted, and the size of the Q value is proportional to the anti-interference ability of the system; increasing the Q value, the anti-interference ability of the system is enhanced, and the adjustment time of the system is shortened. However, at the same time, the oscillation of the system is strengthened, and the consumption of energy increases. The increase of R value makes the energy consumed by the system less, but the adjustment time increases. Therefore, the design key is to find the right weight matrix Q and matrix R. As long as we make sure of the matrix Q and matrix R, the state feedback matrix K is the only confirmation. However, the selection of the Q matrix and R matrix entirely depends on the experience and trial and error method in the process of LQR controller design, so the subjectivity is larger, resulting in the imperfection of the controller design and affecting the control effect.

3.2. The Parameter Optimization Principle of the LQR Controller Based on the Particle Swarm Optimization Algorithm. As a new optimization algorithm has been developed in recent years, the particle swarm optimization algorithm is abbreviated as the PSO. The particle swarm optimization algorithm is a kind of evolutionary algorithms, and it starts from the random solutions and searches for the optimal solution through the iterative algorithm [13].

Assuming X particles are composed of a group in n dimensional space, among them, the position and velocity of i particle in the space are $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, $v_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{iD})$, $i = 1, 2, \ldots, m$, the best position that the i particle experiences is denoted by $p_{\mathrm{bset}}(i)$, and the best position that all the particles in the group experience is denoted by $g_{\mathrm{bset}}(i)$. The whole particle swarm updates the velocity and position through tracking the individual extreme value and the optimal value [14]. The particle optimization process is expressed in

$$\begin{aligned} v_{id}^{k+1} &= w v_{id}^k + c_1 \partial \left(p_{id}^k - x_{id}^k\right) + c_2 \beta \left(p_{gd}^k - x_{id}^k\right), \\ x_{id}^{k+1}(t+1) &= x_{id}^k(t) + v_{id}^k(t). \end{aligned} \quad (12)$$

Among them, ω is the inertia weight; c_1 is the weight coefficient of the optimal value that the particle tracks its history; c_2 is weight coefficients that particle track the optimal value; and ∂ and β are the random numbers changing in $[0, 1]$. p_{id}^k is the individual optimal solution of the particle after the k iterations; p_{gd}^k is the global optimal solution of the group in the k iterations. To make the algorithm a more accurate search scope, the movement speed of the particle is limited in $[-v_{\max}, v_{\max}]$; if $v_{\max}$ is too large, the particle will fly over the optimal solution; if it is too small, it is easy to fall into the local optimum. Assuming that the particle position is defined as the interval $[-v_{\max}, v_{\max}]$, the two-wheeled self-balancing robot state variables $[x, \dot{x}, \theta_p, \dot{\theta}_p]^T$ are regarded as the particles, and the particle's position and the initial value of the speed are produced at random in a certain range. The fitness function is an important link in using the particle swarm optimization algorithm, and it is the standard of the whole particle swarm algorithm iterative evolution. Because what we have designed is the quadratic optimal control regulator, we adopt the linear quadratic performance index formula (10) as the fitness function. The Q is a symmetric positive semidefinite matrix of $6 * 6$; R is a constant positive definite matrix. In order to simplify the problem and make the weighted matrix a clear physical meaning, we choose

Table 1: The parameter of the robot.

Symbol	Actual value
k_m	0.0136 Nm/A
k_e	0.01375 V/(rad/s)
R	1.6 Ω
M_P	0.52 kg
M_w	0.02 kg
l	0.16 m
I_p	0.0038 kg·m^2
g	9.8 m/s^2
I_w	0.0032 kg·m^2
r	0.025 m

the weighted matrix Q as the diagonal matrix, so that the performance index can be represented as

$$J = \int_0^\infty \left(q_1x_1^2 + q_2x_2^2 + q_3x_3^2 + q_4x_4^2 + Ru^2\right) dt. \quad (13)$$

Among them, q_1, q_2, q_3, and q_4 are the weights of the position, speed, angle, and angular velocity of the two-wheeled self-balancing robot, respectively. R is the square weight of the control amount u in the objective function.

3.3. The Parameter Optimization Steps of the LQR Controller Based on the Practical Swarm Optimization Algorithm

Step 1. Initialize the particle swarm. Set the speed coefficients c_1, c_2, the maximum evolution algebra gen, the size of the group pop, and the location of the initial search point and its speed, and each particle has the value of the current position.

Step 2. Calculate the fitness value $F_{id}[i]$ of each particle.

Step 3. Compare the fitness value $F_{id}[i]$ with the individual extremum $p_{\text{best}}(i)$ of each particle; if $F_{id}[i] > p_{\text{best}}(i)$, use $F_{id}[i]$ to replace $p_{\text{best}}(i)$.

Step 4. Compare the fitness value $F_{id}[i]$ with the global extremum g_{best} of each particle; if $F_{id}[i] > p_{\text{best}}(i)$, use $F_{it}[i]$ to replace g_{best}.

Step 5. Update the particle's speed v_i and position x_i according to formula (12).

Step 6. If it meets the end condition (the error is good enough or it reaches the maximum cycle times), it will exit, or it returns to Step 2.

4. The Simulation Experiment of the LQR Controller Based on the Particle Swarm Algorithm

The parameter symbols, description, and the actual value of the two-wheeled self-balancing robot are shown in Table 1.

The actual parameters of this system are substituted into the state equation; the actual state equation is obtained as follows:

$$\begin{bmatrix}\dot{X}\\ \ddot{X}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -0.1038 & 25.5862 & 0\\ 0 & 0 & 0 & 1\\ 0 & -0.5015 & 238.4685 & 0\end{bmatrix}\begin{bmatrix}X\\ \dot{X}\\ \phi\\ \dot{\phi}\end{bmatrix} + \begin{bmatrix}0\\ 0.4891\\ 0\\ 2.3634\end{bmatrix}V_a. \quad (14)$$

The initial state of self-balancing robot system is as follows: $[x, \dot{x}, \theta_p, \dot{\theta}_p]^T = [0, 0, 0, 0]^T$, selecting $c_1 = c_2 = 1.2$ to make experiment, at the same time, selecting the inertia weight formula [15]:

$$\omega^k = \omega_{\max} - \frac{k}{\text{gen}}\left(\omega_{\max} - \omega_{\min}\right). \quad (15)$$

Among them, $\omega_{\max} = 1$, $\omega_{\min} = 0.3$, and gen $= 30$; it indicates the iterative number of the algorithm evolution, and k is the current evolution algebra.

For an initial population of 40 $*$ 8 matrix, the four dimensions in the front represent particle updated location and the four dimensions in the latter represent particle updated speed. The particle swarm updated position curve is shown in Figure 3. The particle motion curve shown in Figure 3 is not lost regularity, so there is only one particle position in critical condition. The PSO updated rate curve is shown in Figure 4. Figure 4 shows the particle movement speed can be controlled, not beyond the intended scope. Selecting the inertia weight formula is appropriate.

PSO algorithm in this case has a total of 50 times iterative and adaptive values. The number of iterations with the curve is shown in Figure 5.

Seen from Figure 5, Q and R parameters after 21 iterations to achieve the optimization.

The global optimal solution can be gotten through the particle swarm algorithm programming:

$$Q = \begin{bmatrix}221.5326 & 0 & 0 & 0\\ 0 & 169.2376 & 0 & 0\\ 0 & 0 & 121.2542 & 1\\ 0 & 0 & 0 & 187.6532\end{bmatrix}, \quad (16)$$

$$R = 1.7682.$$

With the aid of MATLAB function $K = \text{lqr}(A, B, Q, R)$ to work out the optimal feedback matrix: $K = [-11.1932, -16.2145, -72.4045, -17.7329]$.

The dynamic response curves of the two kinds of algorithms of the LQR controller and the LQR controller based on the particle swarm algorithm are shown, respectively, in Figure 6.

The simulation curves of Figure 6 show that when the initial conditions of self-balancing robot are zero, the LQR

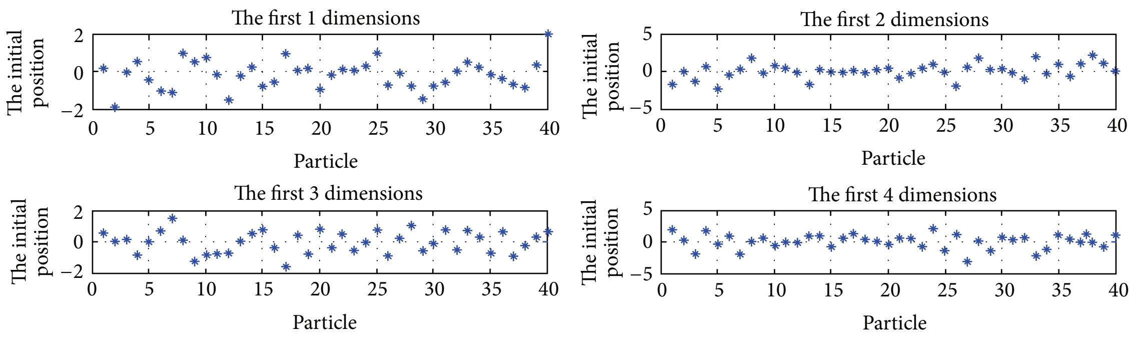

Figure 3: Particle swarm position initialization.

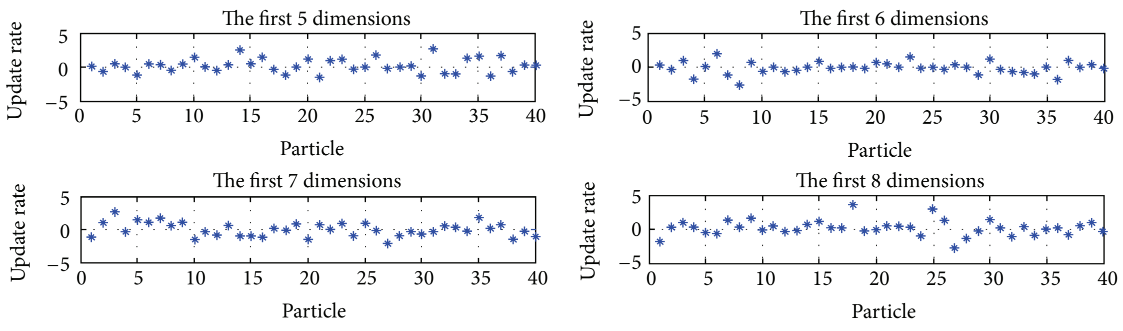

Figure 4: Particle swarm speed initialization.

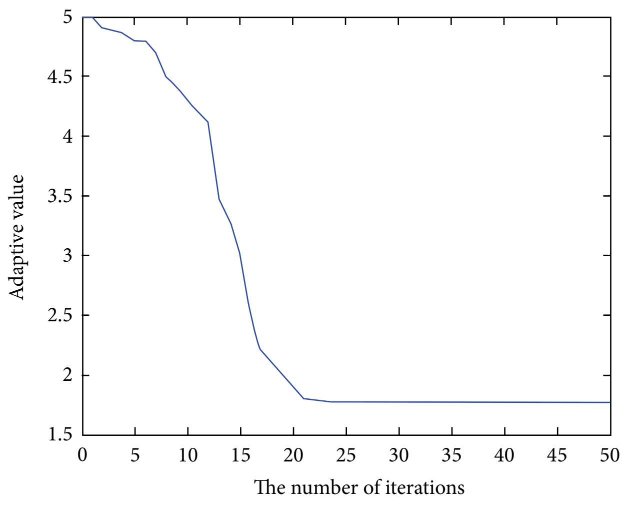

Figure 5: Q and R iterative optimization Figure based on PSO algorithm.

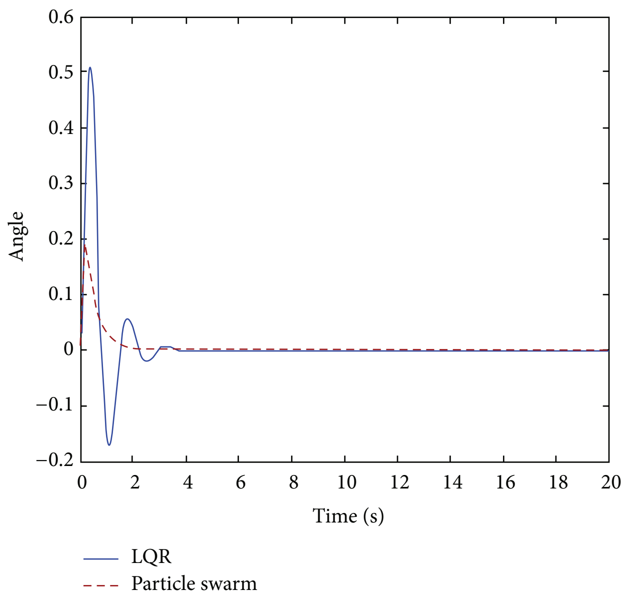

Figure 6: Based on self-balancing robot angle PSO optimized response.

control and LQR control after the particle swarm optimization algorithm can make the system stable. However, the latter algorithm has the advantages of stable short time, less overshoot with fewer shocks.

In order to verify the LQR algorithm based on PSO optimization is better than that LQR algorithm in the literature [4] and literature [7]. LQR algorithm comparison and each algorithm control indicators are shown in Table 2.

According to the above results, three kinds of algorithms can all achieve the stability of the system. The proposed algorithm on the overshoot and oscillation frequency is out of the more obvious advantages. That is because the algorithm

Table 2: The effective comparison table between other methods and proposed methods.

Symbol	Stability	Overshoot	Regulation time	Oscillation frequency
Literature [4]	Stable	62%	1.8 s	1
Literature [7]	Stable	21%	3.2 s	3
Proposed method	Stable	18%	2.5 s	1

can find the optimal solution of the LQR matrix Q and matrix R.

5. Conclusion

Using the particle swarm algorithm to optimize the selection of weighted matrix Q and matrix R can overcome the blindness of selecting matrix Q and matrix R in the traditional LQR optimal control. This paper uses the characteristics that the particle swarm optimization algorithm can achieve an intelligent search, gradual optimization, and rapid convergence. Therefore, it is not easy to fall into local optimum, but easy to be implemented on the basis of the linear model of the two-wheeled and self-balancing robot to obtain the global optimal solution of the Q, R, achieving the optimal LQR controller design through the MATLAB simulation experiments. It can be found that the design response speed of the LQR optimal controller is faster with less overshoot amount, and it can keep the steady-state error zero, so the control effect is better.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] J. Wu and W. Zhang, "Design of fuzzy logic controller for two-wheeled self-balancing robot," in *Proceedings of the 6th International Forum on Strategic Technology (IFOST '11)*, pp. 1266–1270, August 2011.

[2] Y. Qin, Y. Liu, X. Zang, and J. Liu, "Balance control of two-wheeled self-balancing mobile robot based on TS fuzzy model," in *Proceedings of the 6th International Forum on Strategic Technology (IFOST '11)*, pp. 406–409, August 2011.

[3] X. Ruan, J. Liu, H. Di, and X. Li, "Design and LQ control of a two-wheeled self-balancing robot," in *Proceedings of the 27th Chinese Control Conference (CCC '08)*, pp. 275–279, July 2008.

[4] J. Zhao and X. Ruan, "The LQR control and design of dual-wheel upright self-balance robot," in *Proceedings of the 7th World Congress on Intelligent Control and Automation (WCICA '08)*, pp. 4859–4863, June 2008.

[5] L. Qiang, K.-K. Wang, and G.-S. Wang, "Research of LQR controller based on two-wheeled self-balancing robot," in *Proceedings of the Chinese Control and Decision Conference (CCDC '09)*, pp. 2343–2348, June 2009.

[6] Y. Li, C. Yue, and M. Wang, "The inverted pendulum system based on genetic algorithm of multi-stage control research," *Journal of North China University of Technology*, pp. 19–24, 2009.

[7] J. Wu and W. Zhang, "Design of fuzzy logic controller for two-wheeled self-balancing robot," in *Proceedings of the 6th International Forum on Strategic Technology (IFOST '11)*, pp. 1266–1270, August 2011.

[8] J.-X. Cai, X.-G. Ruan, and J.-F. Gan, "Modeling of two-wheeled self-balancing robot and fuzzy self-adjusting PID control," *Journal of Beijing University of Technology*, vol. 35, no. 12, pp. 1603–1607, 2009.

[9] A. M. Bloch and P. E. Crouch, "Newton's law and nonholonomic systems," in *Proceedings of the 37th IEEE Conference on Decision and Control (CDC '98)*, pp. 3569–3574, December 1998.

[10] E. R. Wuori and J. H. Judy, "Rotational hysteresis for domain wall motion in the presence of demagnetizing fields," *IEEE Transactions on Magnetics*, vol. 21, no. 5, pp. 1602–1603, 1985.

[11] L. Cheng, H. Dewen, P. Yaodong, Z. Xiaocai, and D. Guohua, "Fuzzy control of a quintuple inverted pendulum with the LQR method and 2-ary fuzzy piecewise interpolation function," in *Proceedings of the 45th IEEE Conference on Decision and Control (CDC '06)*, pp. 6307–6312, December 2006.

[12] F. M. Ham and E. G. Collins, "Neurocomputing approach for solving the algebraic matrix Riccati equation," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '96)*, pp. 617–622, June 1996.

[13] J. Xu and C. Huiyou, "The discrete binary version of the improved particle swarm optimization algorithm," in *Proceedings of the International Conference on Management and Service Science (MASS '09)*, pp. 1–6, September 2009.

[14] W. L. Tan, "An improved method for rectilinear double inverted pendulum LQR controller parameter optimization," *Journal of Chongqing University of Technology (Natural Science)*, pp. 85–88, 2012.

[15] X. M. Wu, *Double Inverted Pendulum Based on Particle Swarm Optimization Control Research*, Qufu Normal University, Qufu, China, 2012.