PD Control with RBF Neural Network Gravity Compensation for Manipulator

Haitao Zhang, Mengmeng Du, Guifang Wu and Wenshao Bu

***Abstract*—In order to overcome the disadvantages of the traditional PD control in manipulator, a new point to point control of manipulator with interference and gravity uncertainty is proposed. On the Basis of PD control structure, the RBF neural network is introduced to realize the gravity compensation, and a robustness analysis is also given with respect to the approximation error of RBF neural network. The lyapunov method shows that the better performance can be obtained with the new scheme as compared to the present control methods.**

***Index Terms*—manipulator; RBF neural network; lyapunov function**

I. Introduction

DYNAMIC model of manipulator is a highly nonlinear, strong coupling, time-varying model, and its control problem has always been a hot research topic. Domestic and foreign scholars have proposed many control methods [1-2]. When the model of the manipulator is precisely known, combining feedback linearization with passive control, Romeo Ortega and Mark W. Spong [3] put forward two kinds of control schemes, and study the adaptive method of the two schemes, which make the robotic arm to reach the control target. However, due to the manipulator's working environment and the load is always changing, the precise model is hard to get, uncertainty often exists in the model, which makes the controller is very difficult to achieve the ideal effect based on the accurate model [4].

According to the model uncertainty, Le Tien Dung etc. [5] proposed on-line gravity compensation of the PD control, but simple PD control has a low accuracy, it is difficult to meets the requirement of manipulator control performance. Piltan [6] designed a PD-plus-gravity controller which has a nonlinear part to eliminate the term of gravity and fuzzy nonlinear equivalent part to eliminate the nonlinearity part. The proposed method can cancel the terms of gravity. But the fuzzy logic rules and membership functions is difficult to determine. Huang [7] presented a nonlinear PD controller with gravity compensation for robot manipulators. The error of nonlinear function is used to change proportional and derivative gains so as to less the influence of the uncertainty of the dynamic parameters. Ernesto [8] combines iterative learning control with PID control with gravity compensation, and obtain global asymptotic stability.

Neural network has the ability to approximate any nonlinear function, so it is widely used in the control of the manipulator system. Based on the lyapunov stability theory, Lu Lu etc. [9] put forward two BP networks to approximate the manipulator dynamic model, a certain effect is achieved. But BP network is a kind of global approximation network, it has long learning time and slow convergence speed, so it is difficult to meets the real-time requirements of control system. Yang [10] applied the RBF neural network to nonlinear robot manipulators with uncertain dynamics to approximate a desired control target.

In this paper, the RBF neural network is used to approximate the gravity term of the dynamic equation, the adaptive control law is designed to adjust the network parameters. Then we apply the method to the system with the external disturbance. The simulation results prove the effectiveness and correctness of the method.

II. Presentation of Question

A. Ideal PD controller

A standard method for deriving the dynamics equations of a mechanical system is via the Euler-Lagrange equations. Using this method, the dynamics of an n-degree-of-freedom rigid manipulator can be described in the following general form:

$$D(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)+\tau_d=\tau \qquad (1)$$

Where $D(q)$ is an $n\times n$ inertia matrix, which is a positive definite matrix. $C(q,\dot{q})$ is an $n\times n$ matrix containing the centrifugal and coriolis forces. $G(q)$ is an $n\times 1$ vector containing gravity torques. q is joint position, τ is joint drive torques. τ_d denotes disturbance and $\int_0^\infty \tau_d^T\tau_d dt$ is bounded.

The dynamic characteristics of manipulator system are as follows:

Property 1: The matrix $D(q)-2C(q,\dot{q})$ is skew-symmetric.

Property 2: Inertia matrix $D(q)$ is a symmetric positive definite matrix bounded by $d_1\|x\|^2 \le x^T D(q)x \le d_2\|x\|^2$,

Manuscript received January 2, 2018; revised April 27, 2018. This work was supported in part by the key scientific research project of universities and colleges of Henan Province under Grant 15A413003.

Haitao Zhang is with the School of Information Engineering, Henan University of Science and Technology, Luoyang, CO 471023 China (corresponding author: e-mail: zhang_haitao@163.com).

Mengmeng Du is with the School of Information Engineering, Henan University of Science and Technology, Luoyang, CO 471023 China.

Guifang Wu is with the School of Information Engineering, Henan University of Science and Technology, Luoyang, CO 471023 China.

Wenshao Bu is with the School of Electrical Engineering, Henan University of Science and Technology, Luoyang, CO 471023 China.

where d_1 *and* d_2 are known positive constants.

Property 3: The unknown disturbance satisfies $\|\tau_d\|_\infty \le b_d$, where b_d is a known positive constant.

Assuming that there is no external disturbance, the PD control law based on gravity compensation is as follows [12]:

$$\tau = K_p e + K_d \dot{e} + \hat{G}(q) \tag{2}$$

where K_p is proportion coefficient, K_d is differential coefficient, and $\hat{G}(q)$ is the estimate value of the gravity term.

q_d is a constant when the point to point control is used, so $\dot{q}_d = \ddot{q}_d \equiv 0$. Defining the tracking error for $e = q_d - q$. Now, the manipulator equation is as follows:

$$\begin{aligned} &D(q)(\ddot{q}_d - \ddot{q}) + C(q,\dot{q})(\dot{q}_d - \dot{q}) + K_p e \\ &+K_d \dot{e} + \hat{G}(q) - G(q) = 0 \end{aligned} \tag{3}$$

The key to realize the control law (2) is to estimate the gravity term $G(q)$. Assuming that the estimation is accurate, namely $\hat{G}(q) = G(q)$, yield:

$$D(q)\ddot{e} + C(q,\dot{q})\dot{e} + K_p e = -K_d \dot{e} \tag{4}$$

Define the lyapunov function candidate:

$$L = \frac{1}{2}\dot{e}^T D(q)\dot{e} + \frac{1}{2}e^T K_p e \tag{5}$$

Because $D(q)$ and K_p are the positive definite matrices, we can know L is the global positive definite.

The time derivative of the lyapunov function is given by the following equation:

$$\dot{L} = \dot{e}^T D(q)\ddot{e} + \frac{1}{2}\dot{e}^T \dot{D}(q)\dot{e} + \dot{e}^T K_p e \tag{6}$$

According to the Property 1 of the manipulator, we can get $\dot{e}^T \dot{D}\dot{e} = 2\dot{e}^T C\dot{e}$, Substituting it into Equation (6), we can get:

$$\begin{aligned} \dot{L} &= \dot{e}^T D\ddot{e} + \dot{e}^T C\dot{e} + \dot{e}^T K_p e \\ &= \dot{e}^T (D\ddot{e} + C\dot{e} + k_p e) = -\dot{e}^T K_d \dot{e} \le 0 \end{aligned} \tag{7}$$

Because $\dot{L}$ is negative semi-definite, K_d is positive definite, so when $\dot{L} \equiv 0$, we can get $\dot{e} \equiv 0$, $\ddot{e} \equiv 0$. Substituting it into Equation (4), we get $K_p e = 0$, so $e = 0$. By LaSalle theorem, we can know $(e, \dot{e}) = (0,0)$ is the global asymptotic stability equilibrium point of manipulator.

But the manipulator system without any interference and external forces does not exist, the above method can be only considered as the foundation. In view of the above situation, in this paper, a new control method is proposed on the basis of the following premises:

1. Gravity term is uncertain in the manipulator system.
2. The system has the external disturbance.

B. RBF neural network approximation to gravity uncertainty term

Radial Basis Function neural network was proposed by J. Moody and C. Darken in the 1980s, RBF neural network can approximate any continuous function with arbitrary precision. It is a three-layer feed forward network with a single hidden layer, the neuron of output layer has a linear characteristic, and the activation function of the hidden layer uses radial basis functions. It is generally a Gaussian function which is expressed as:

$$h_j = \exp\left(-\frac{\|x - c_j\|^2}{b_j^2}\right) \tag{8}$$

RBF neural network structure is shown in Fig. 1:

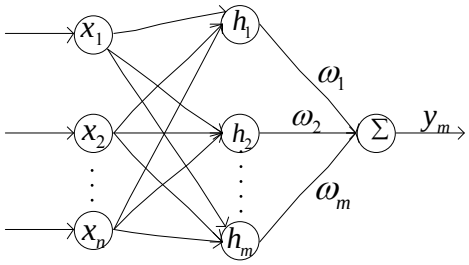


Fig. 1. The RBF neural network structure

The input-output mapping relationship is:

$$y = \omega^T h(x) \tag{9}$$

Where x is the network's input signal, y is the output signal, $h(x)$ is the output of the Gaussian basis function, ω denote neural network weights.

Under the condition of the following assumptions, RBF neural network can approximate continuous functions with any degree of accuracy in a compact set[13].

1. The neural network output $\hat{f}(x, \omega)$ is continuous.
2. The ideal approximation of the neural network output is $\hat{f}(x, \omega^*)$, for a very small positive number ε_0, which has:

$$\max \left\| \hat{f}(x, \omega^*) - f(x) \right\| \le \varepsilon_0 \tag{10}$$

where $\omega^* = \arg \min_{\omega \in \beta(M_\omega)} \{ \sup_{x \in h(M_x)} \left\| \hat{f}(x, \omega) - f(x) \right\| \}$, it is a $n \times n$ matrix, and denotes the best approximation of neural network weights.

Taking the approximation error of ideal neural network, namely

$$\eta = \hat{f}(x, \omega^*) - f(x) \tag{11}$$

By the approximation capability of RBF network, the modeling error η is bounded, assuming that it is η_0.

$$\eta_0 = \sup \left\| \hat{f}(x, \omega^*) - f(x) \right\| \tag{12}$$

And $$\hat{f}(x, \omega^*) = \omega^{*T} h(x)$$

In this paper, RBF neural network is used to approximate the gravity term, and the formula is expressed as $f(x) = G(q)$.

Manipulator equation is shown as Equation (1), a control law is as follows:

$$\tau = K_d \dot{e} + K_p e + \hat{f}(x,\omega) - v \tag{14}$$

where $v = -(\eta_0 + b_d)\,\mathrm{sgn}(\dot{e})$ is the robust term, it is used for overcoming the neural network approximation error.

Substituting Equation (14) into (1), the manipulator equation becomes:

$$\begin{aligned} &D(q)(\ddot{q}_d - \ddot{q}) + C(q,\dot{q})(\dot{q}_d - \dot{q}) + K_p e + K_d \dot{e} \\ &+ \hat{f}(x,\omega) - f(x) - \tau_d - v = 0 \end{aligned} \tag{15}$$

$$\begin{aligned} &D(q)\ddot{e} + C(q,\dot{q})\dot{e} + K_p e \\ &= -K_d \dot{e} - \tilde{\omega}^T h(x) + \eta + \tau_d + v \end{aligned} \tag{16}$$

Where

$$\begin{aligned} \hat{f}(x,\omega) - f(x) &= \hat{f}(x,\omega) - \hat{f}(x,\omega^*) + \hat{f}(x,\omega^*) - f(x) \\ &= \hat{\omega}^T h(x) - \omega^{*T} h(x) + \eta \\ &= -\tilde{\omega}^T h(x) + \eta \end{aligned} \tag{17}$$

An adaptive law is designed as:

$$\dot{\hat{\omega}} = -Fh(x)\dot{e}^T \tag{18}$$

The design of the structure is shown as Fig. 2. In Fig. 2, RBFNN is the abbreviation of RBF neural network.

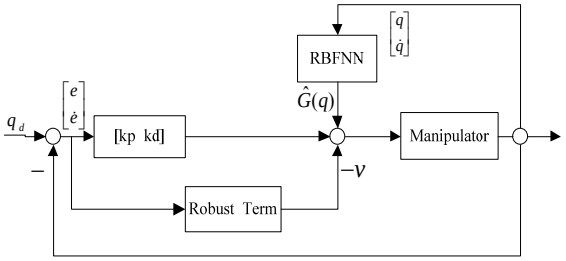


Fig. 2. The system control structure

Let us introduce the candidate lyapunov function:

$$L = \frac{1}{2}\dot{e}^T D\dot{e} + \frac{1}{2}e^T K_p e + \frac{1}{2}tr(\tilde{\omega}^T F^{-1}\tilde{\omega}) \tag{19}$$

where F is a symmetric positive definite matrix, so L is global positive definite.

Differentiating (19), we get

$$\dot{L} = \dot{e}^T D\ddot{e} + \frac{1}{2}\dot{e}^T \dot{D}\dot{e} + \dot{e}^T K_p e + tr(\tilde{\omega}^T F^{-1}\dot{\tilde{\omega}}) \tag{20}$$

Substituting $\dot{e}^T \dot{D}\dot{e} = 2\dot{e}^T C\dot{e}$ into Equation (20), yield:

$$\dot{L} = \dot{e}^T D\ddot{e} + \dot{e}^T C\dot{e} + \dot{e}^T K_p e + tr(\tilde{\omega}^T F^{-1}\dot{\tilde{\omega}}) \tag{21}$$

Substituting Equation (16) into Equation (21), yield:

$$\begin{aligned} \dot{L} &= \dot{e}^T(-K_d\dot{e} - \tilde{\omega}^T h + \eta + \tau_d + v) + tr(\tilde{\omega}^T F^{-1}\dot{\tilde{\omega}}) \\ &= -\dot{e}^T K_d \dot{e} - \dot{e}^T\tilde{\omega}^T h + tr(\tilde{\omega}^T F^{-1}\dot{\tilde{\omega}}) + \dot{e}^T(\eta + \tau_d + v) \\ &= -\dot{e}^T K_d \dot{e} + tr\tilde{\omega}^T(F^{-1}\dot{\tilde{\omega}} - h\dot{e}^T) + \dot{e}^T(\eta + \tau_d + v) \end{aligned} \tag{22}$$

Substituting adaptive law (18) into Equation (22), we get:

$$\dot{\tilde{\omega}} = \dot{\omega}^* - \dot{\hat{\omega}} = 0 - \dot{\hat{\omega}} = Fh(x)\dot{e}^T \tag{23}$$

Then

$$\dot{L} = -\dot{e}^T K_d \dot{e} + \dot{e}^T(\eta + \tau_d + v) \tag{24}$$

where

$$\begin{aligned} \dot{e}^T(\eta + \tau_d + v) &= \dot{e}^T(\eta + \tau_d) + \dot{e}^T v \\ &= \dot{e}^T(\eta + \tau_d) - \|\dot{e}\|(\eta_0 + b_d) \le 0 \end{aligned}$$

where $\|\eta\| \le \eta_0, \|\tau_d\| \le b_d$.

From the above analysis, we can obtain $\dot{L} \le 0$, so the equilibrium state of system is uniform stable.

III. SIMULATION

A. *The model of manipulator*

In this section, a simulation study is conducted to demonstrate the performance of our algorithm. A simple two degree of freedom manipulator is used in the simulation. The dynamic equation has been given in Equation (1), that is,

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + \tau_d = \tau$$

For two degree of freedom manipulator,

$$D(q) = \begin{pmatrix} p_1 + p_2 + 2p_3\cos q_2 & p_2 + p_3\cos q_2 \\ p_2 + p_3\cos q_2 & p_2 \end{pmatrix}$$

$$C(q,\dot{q}) = \begin{pmatrix} -p_3\dot{q}_2\sin q_2 & -p_3(\dot{q}_1 + \dot{q}_2)\sin q_2 \\ p_3\dot{q}_1\sin q_2 & 0 \end{pmatrix}$$

$$G(q) = \begin{pmatrix} p_4 g\cos q_1 + p_5 g\cos(q_1 + q_2) \\ p_5 g\cos(q_1 + q_2) \end{pmatrix}$$

$$p_1 = (m_1 + m_2)l_1^2$$

$$p_2 = m_2 l_2^2$$

$$p_3 = m_2 l_1 l_2$$

$$p_4 = (m_1 + m_2)l_1$$

$$p_5 = m_2 l_2$$

The manipulator parameters are: $m_1 = 2.04kg$, $m_2 = 1kg$, $l_1 = 1m$, $l_2 = 0.87m$. So we get the parameter $P = [p_1\ p_2\ p_3\ p_4\ p_5]^T = [2.9\ 0.76\ 0.87\ 3.04\ 0.87]^T$,

The structure of RBF neural network is $2\times5\times1$, and its input is $z = [q\ \ \dot{q}]$. The parameters of Gaussian basis function is $c = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0.01 & 0.01 & 0.01 & 0.01 \end{pmatrix}$, $b = 0.54$.

The initial state of system is $[0\ \ 0\ \ 0\ \ 0]$.

Position command is $q_d(0) = [1.5\ \ 3.0]^T$. For the robust term, $\eta_0 = 0.5$, $b_d = 0.2$. SIMULINK and S function is used to design the control system. In order to compare the advantages of proposed control method, we also simulate the PID control method and PD control based fixed gravity compensation.

B. *PD control based on fixed gravity compensation without interference*

For the PD control based on fixed gravity compensation, we can set up different gravity compensation items so as to understand the shortage of fixed gravity compensation.

First, we set up the interference $\tau_d = 0$. The control

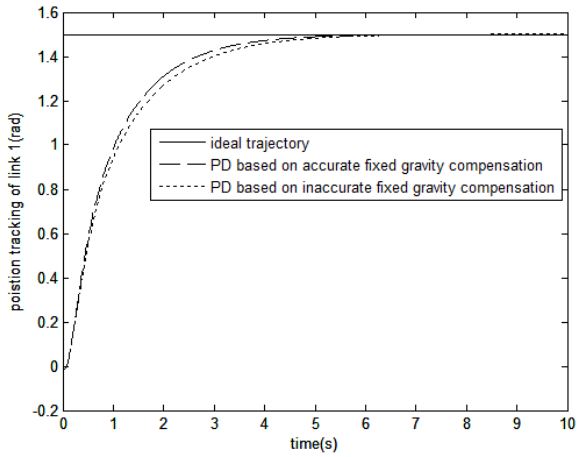
parameters are $K_p = diag\{90,90\}$, and $K_d = diag\{70,70\}$.

The simulation results are shown from Fig. (3-1) to Fig. (3-6). For the inaccurate fixed gravity compensation, the gravity item is $G_p(q) = 1.2G(q)$; For the accurate fixed gravity compensation, the gravity item is $G_p(q) = G(q)$.

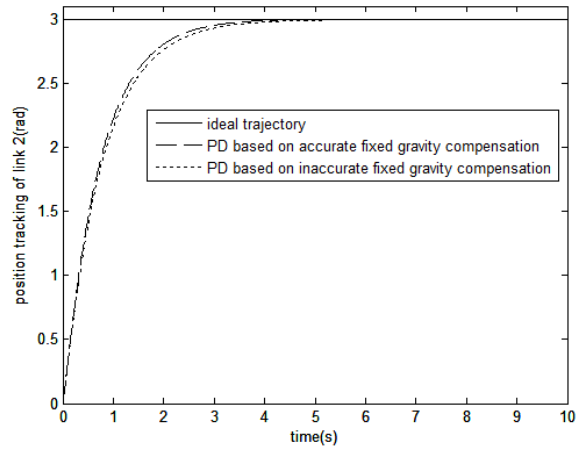
Fig. (3-1) and Fig. (3-2) are respectively the position tracking trajectories of link 1 and link 2. Their adjusting time is similar, and they both reach a stable state after about 5s. However, compared with PD based accurate fixed gravity compensation, PD based on inaccurate fixed gravity compensation has a longer response time, and a larger error before the system becomes stable.

Fig. (3-3) and Fig. (3-4) are respectively the tracking trajectory error of link 1 and link 2. The steady-state error of two methods is about zero. The error of beginning stage of PD based on accurate fixed gravity compensation is less that of PD based on inaccurate fixed gravity compensation.

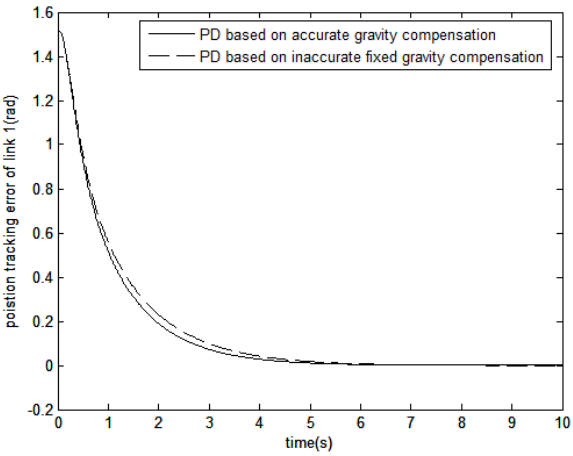
Fig. (3-5) and Fig. (3-6) are respectively the output torque of link 1 and link 2. The both output torque curves are basically same, which shows that the small difference in gravity compensation has little influence on the output torque.



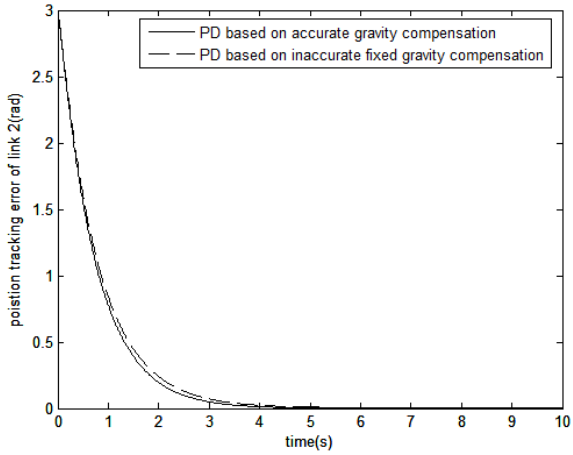
(3-1) tracking trajectory of link 1



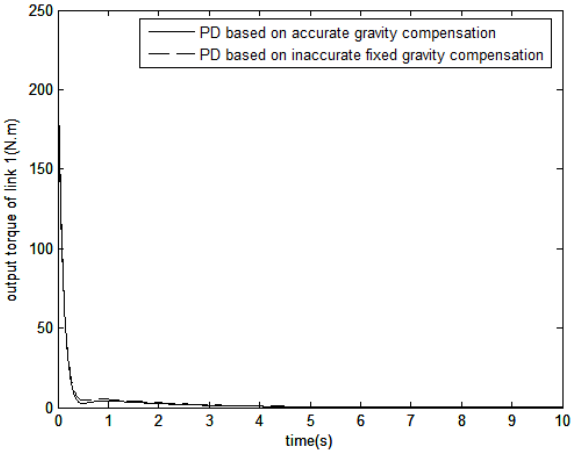
(3-2) tracking trajectory of link 2



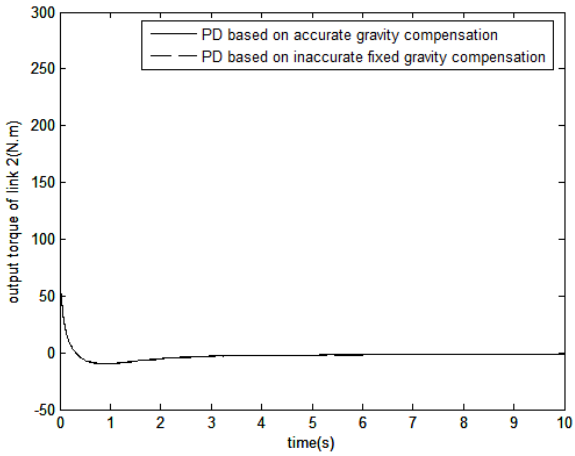
(3-3) tracking trajectory error of link 1



(3-4) tracking trajectory error of link 2



(3-5) output torque of link 1



(3-6) output torque of link 2

Fig. 3 Inaccurate fixed gravity compensation for $G_p(q)=1.2G(q)$ with the interference $\tau_d=0$

If we set up the gravity compensation item $G_p(q)=0.8G(q)$, we can get similar simulation diagram with Fig.3.

Though the steady-state error is almost zero in Fig. (3-3) and Fig. (3-4), the difference is still large. Table I is the data of steady-state error of two links in three cases. The error is almost zero for PD based on accurate fixed gravity compensation, but the error of links under other two cases is very large, especially for link 2.

From the above analysis, PD based on accurate fixed gravity compensation has better dynamic performance than PD based on inaccurate fixed gravity compensation.

TABLE I
THE STEADY-STATE ERROR OF TWO LINKS

Control method	Link 1(m)	Link 2(m)
PD based on accurate fixed gravity compensation	0.0001	0
PD based on inaccurate fixed gravity compensation for $G_p(q)=1.2G(q)$	0.0007	0.0041
PD based on inaccurate fixed gravity compensation for $G_p(q)=0.8G(q)$	0.0007	0.0041

C. PD control based on fixed gravity compensation with interference

However, it is impossible to get accurate gravity compensation because of the existence of interference.

We set up the interference $\tau_d=d_1+d_2\|e\|+d_3\|\dot{e}\|$, $d_1=1.5$, $d_2=2.0$, $d_3=5$. The control parameters are also $K_p=diag\{90,90\}$, and $K_d=diag\{70,70\}$.

The simulation results are shown from Fig. (4-1) to Fig. (4-6). For the inaccurate fixed gravity compensation, the gravity item is $G_p(q)=1.2G(q)$; for the accurate fixed gravity compensation, the gravity item is $G_p(q)=G(q)$.

From Fig. (4-1) to (4-4), we know that the steady-state error is obviously increased due to the existence of the interference. Table II is the data of steady-state error of two links in three cases. For PD based on accurate fixed gravity compensation, it still has better performance, but it generates much larger steady-state error because of the interference.

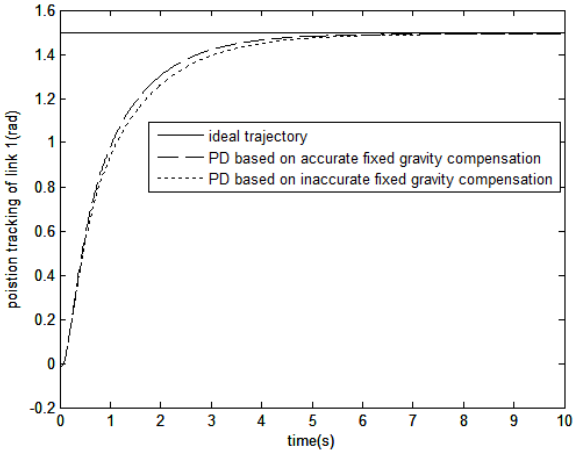
From Fig. (4-5) and Fig. (4-6), we know that the output torque doesn't have great change.

If we set up the gravity compensation item $G_p(q)=0.8G(q)$, we can get similar simulation diagram with Fig.4.

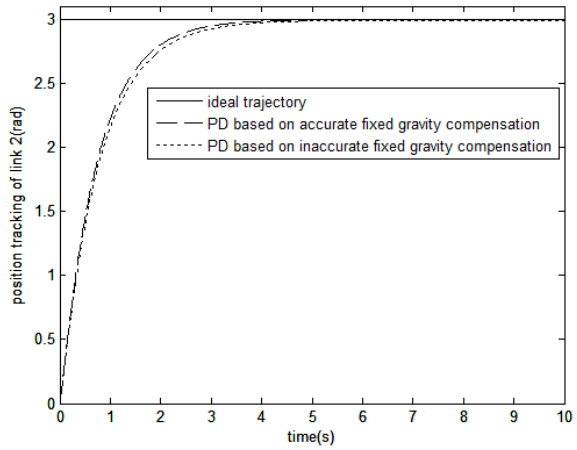
All in all, the inaccurate gravity item will result in poor dynamic response of system. Since the accurate gravity item is difficult to get, and the system always exists interference, it is necessary to use RBF neural network to compensate the gravity item.

TABLE II
THE STEADY-STATE ERROR OF TWO LINKS

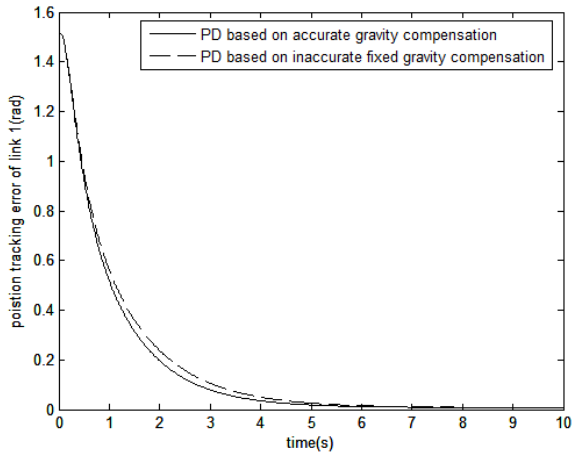
Control method	Link 1(m)	Link 2(m)
PD based on accurate fixed gravity compensation	0.0237	0.0161
PD based on inaccurate fixed gravity compensation for $G_p(q)=1.2G(q)$	0.0262	0.022
PD based on inaccurate fixed gravity compensation for $G_p(q)=0.8G(q)$	0.0262	0.022



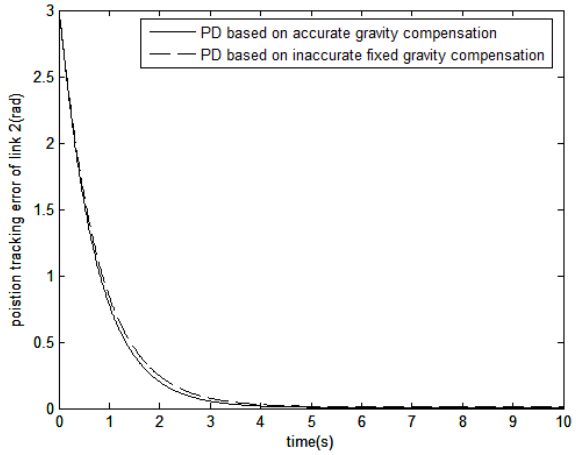
(4-1) tracking trajectory of link 1



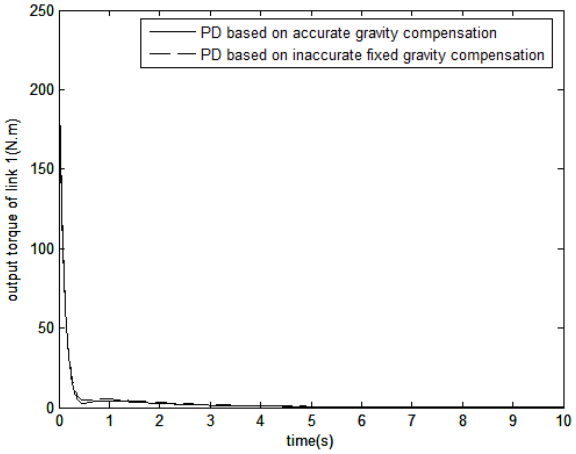
(4-2) tracking trajectory of link 2



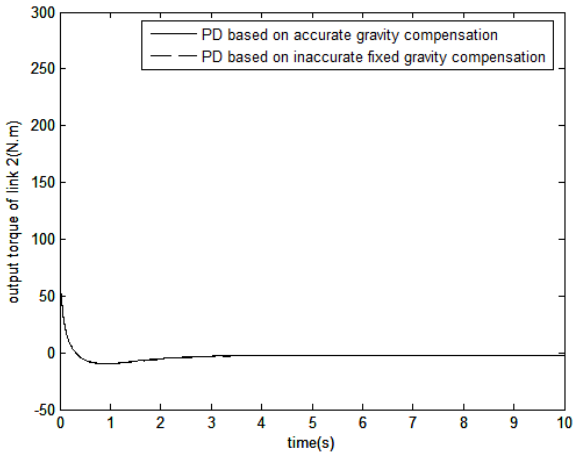
(4-3) tracking trajectory error of link 1



(4-4) tracking trajectory error of link 1



(4-5) output torque of link 1



(4-6) output torque of link 2

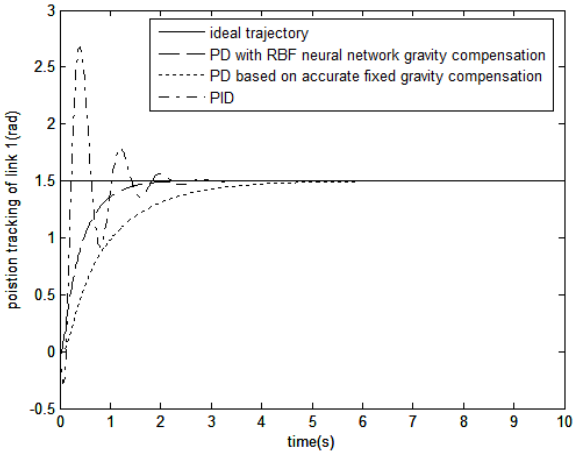
Fig. 4 Inaccurate fixed gravity compensation for $G_p(q)=1.2G(q)$ with the interference $\tau_d \neq 0$

D. PD control based on RBF neural network gravity compensation

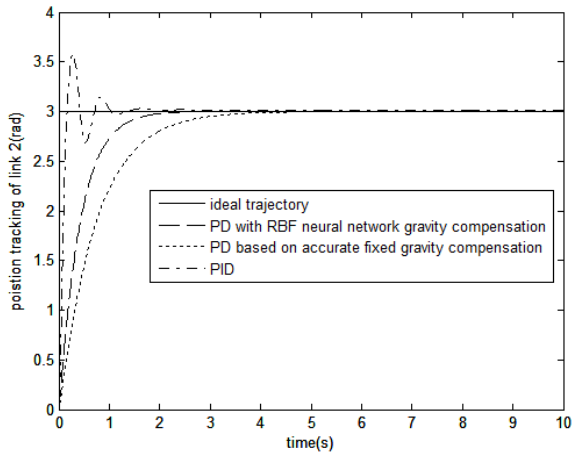
RBF neural network is introduced to compensate the gravity item of system. The inputs of RBF neural network are the system position error and error change rate, so RBF neural network can quickly correct the position error, and track input position trajectory, which makes the system get better dynamic performance.

For PD with RBF neural network gravity compensation, the control parameters $K_p = diag\{400,400\}$, $K_d = diag\{170,170\}$, $F = diag\{10,10\}$. For PID control, the control parameters $K_p = diag\{400,400\}$, $K_d = diag\{7,7\}$, $K_i = diag\{3,3\}$ $F = diag\{10,10\}$.

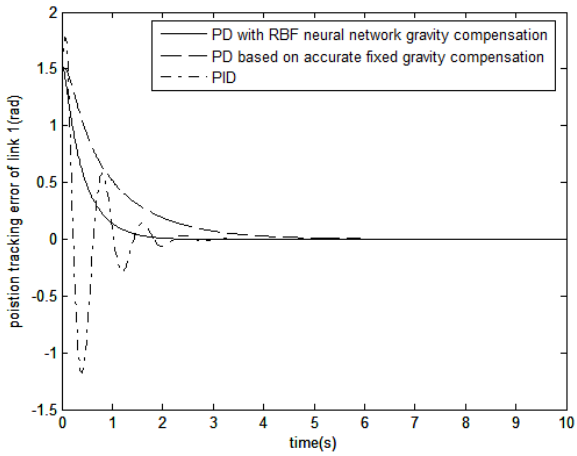
Fig. 5 is the simulation result. It compares the position tracking, velocity trajectory, tracking error, and output torque e of three control methods.



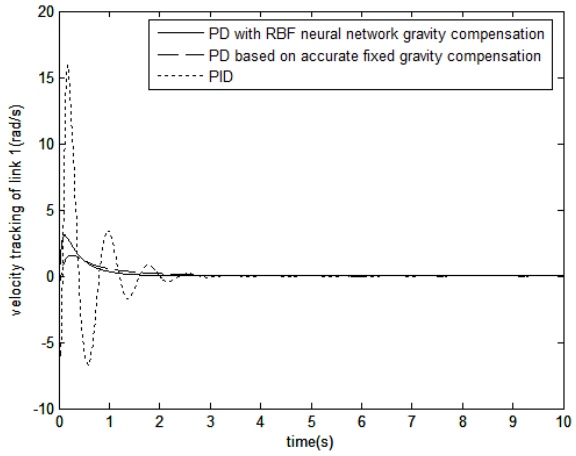
(5-1) tracking trajectory of link 1



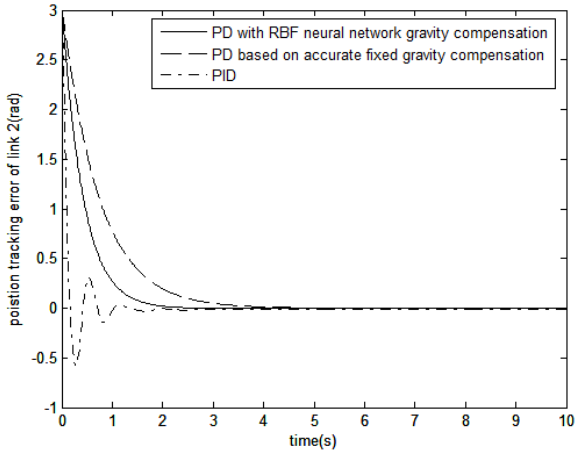
(5-2) tracking trajectory of link 2



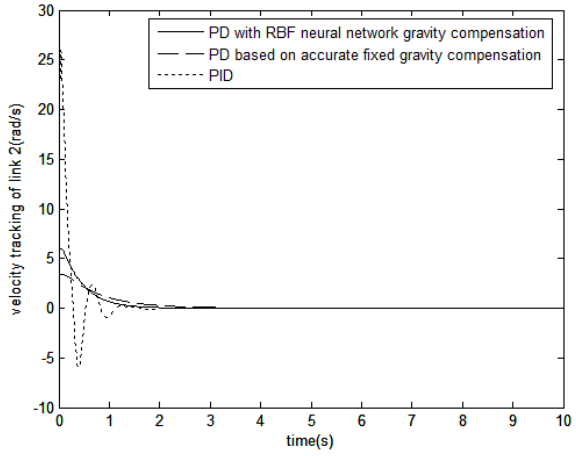
(5-5) tracking trajectory error of link 1



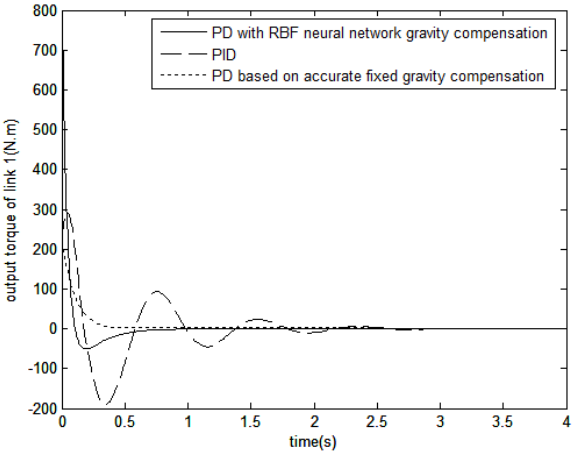
(5-3) velocity trajectory of link 1



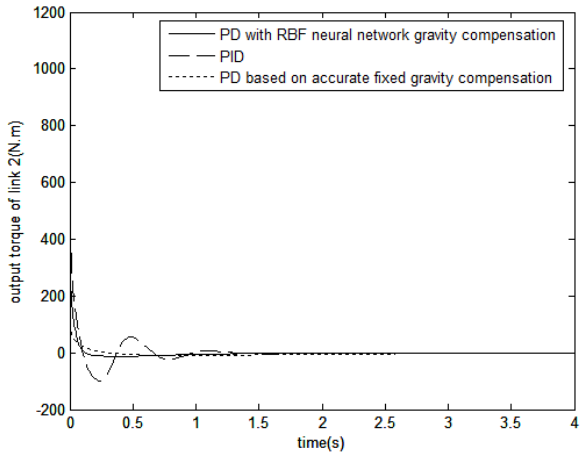
(5-6) tracking trajectory error of link 2



(5-4) velocity trajectory of link 2



(5-7) output torque of link 1



(5-8) output torque of link 2

Fig. 5 PD with RBF neural network gravity compensation

Fig. (5-1) and Fig. (5-2) are respectively the position tracking trajectories of link 1 and link 2. The traditional PID controller has very larger overshoot, which even reaches 80% for link 1. The larger overshoot is disadvantageous for manipulator system. For PD based on accurate fixed gravity compensation, the adjusting time of link 1 and link 2 is both about 4s, and is longer than that of other two methods. PD with RBF neural network gravity compensation has not overshoot, and its adjusting time is about 2s. So PD with RBF neural network gravity compensation has good dynamic performance.

Fig. (5-3) and Fig. (5-4) are respectively the velocity trajectories of link 1 and link 2. The traditional PID controller can generate a larger velocity at the beginning stage. Other two methods also have an initial velocity, but it is acceptable.

Fig. (5-5) and Fig. (5-6) are respectively the tracking trajectory error of link 1 and link 2. The steady-state error of three methods is almost about zero. The error of beginning stage of PD with RBF neural network gravity compensation is less that of PD based on accurate fixed gravity compensation.

For PD with RBF neural network gravity compensation, it quickly response to the tracking trajectory error caused by inaccurate gravity and uncertainty. So it can acquire better dynamic performance than PD based on accurate fixed gravity compensation.

Table III is the data of steady-state error of two links in five cases. Owing to the large fluctuation, PID control method uses the average value from 9s to 10s as its steady-state error. For other methods, the error of tenth second is regarded as steady-state error

TABLE III

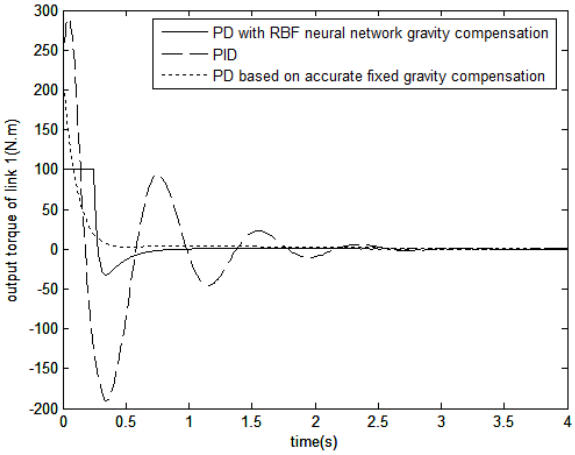
THE STEADY-STATE ERROR OF TWO LINKS

Control method	Link 1(m)	Link 2(m)
Traditional PID control	0.0061	0.002
PD based on accurate fixed gravity compensation	0.0237	0.0161
PD based on inaccurate fixed gravity compensation for $G_p(q)=1.2G(q)$	0.0262	0.022
PD based on inaccurate fixed gravity compensation for $G_p(q)=0.8G(q)$	0.0262	0.022
PD with RBF neural network gravity compensation	0.0015	0.0006

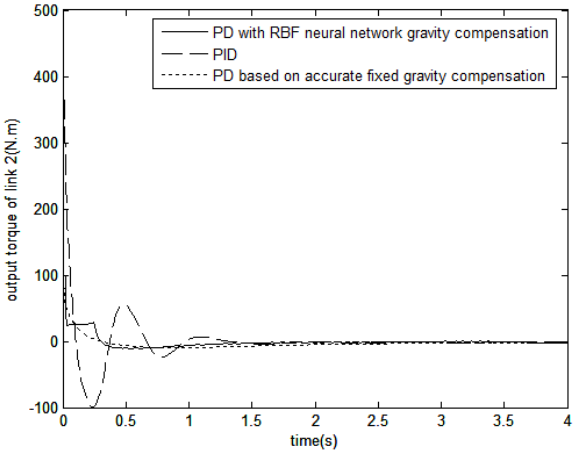
For PD with RBF neural network gravity compensation, its steady-state error is the smallest. This is because the method not only compensates the gravity item, but also compensates the interference.

Fig. (5-7) and Fig. (5-8) are respectively the output torque of link 1 and link 2. PD with RBF neural network gravity compensation has a larger initial torque than PD based on accurate fixed gravity compensation. So PD with RBF neural network gravity compensation has a high requirement for the actuator. It requires system to provide a larger initial torque, which is advantageous to the system.

In order to lessen the initial torque of PD with RBF neural network gravity compensation, we can limit its maximum output toque to 100N.m. Then the simulation is done under same simulation parameters. The simulation figures of the tracking trajectory, the velocity trajectory and the tracking trajectory error are basically same. However, shown as Fig. 6, the initial output torque of PD with RBF neural network gravity compensation can be greatly reduced.



(6-1) output torque of link 1



(6-2) output torque of link 2

Fig. 6 PD with RBF neural network gravity compensation after limiting the maximum output torque

IV. CONCLUSION

This paper puts forward a kind of gravity compensation PD control method based on RBF neural network for the manipulator with external disturbance and uncertainty. The

proposed algorithm doesn't need accurate dynamics model of the manipulator, and can adjust the network weights by online adaptive method. Through MATLAB simulation, the effectiveness of the RBF neural network in gravity compensation is confirmed.

REFERENCES

[1] Haitao Zhang, Mengmeng Du and Wenshao Bu. Sliding Mode Controller with RBF Neural Network for Manipulator Trajectory Tracking. IAENG International Journal of Applied Mathematics, 2015, 45(4): 334-342

[2] Hanlei Wang. Adaptive Control of Robot Manipulators with Uncertain Kinematics and Dynamics. IEEE Transactions on Automatic Control, 2017, 62(2): 948-954

[3] Romeo Ortega, Mark W. Spong. Adaptive motion control of rigid robots: a tutorial. Automatica, 1989, 25(6):877-888.

[4] Mingming Li, Yanan Li, S.S. Ge, and Tong Heng Lee. Adaptive control of robotic manipulators with unified motion constraints. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017, 47(1): 184-194

[5] Haitao Zhang, Xiaofeng Liu and Guifang Wu. A Sliding Mode Controller based on Backstepping for Manipulator. Engineering Letters, 2017, 25(4): 464-473

[6] Le Tien Dung, Hee-Jun Kang, Young-Shick Ro. Robot manipulator modeling in Matlab-Simmechanics with PD control and online Gravity compensation. Strategic Technology(IFOST),2010: 446-449.

[7] F. Piltan, M. J. Rafaati, F. Khazaeni, A. Hosainpour and S. Soltani. A design high impact Lyapunov fuzzy PD-plus-gravity controller with application to rigid manipulator. International Journal of Information Engineering and Electronic Business. 2013, 5(1): 17-25

[8] Jianfeng Huang, Chengying Yang and Jun Ye. Nonlinear PD Controllers with Gravity Compensation for Robot Manipulators. Cybernetics and Information Technologies 2014, 14(1): 141-150

[9] H. Ernesto and J. O. Pedro. Iterative learning control with desired gravity compensation under saturation for a robotic machining manipulator. Mathematical Problems in Engineering. 2015, p187948(13pp.)

[10] Lu Lu, Tianshi Li, Weixiang Shi. Affine nonlinear system research of the neural network adaptive controller and the application of the manipulator. Robot, 1999, 21(3):68-71

[11] Runxian Yang, Chengguang Yang, Mou Chen and Andy SK Annamalai. he RBF neural network. Neurocomputing, 2017, 234: 107-115

[12] Wei Huo. The robot dynamics and control. Beijing：Higher Education Press, 2005.

[13] Hongrui Wang, Congna Liu, Yongxing Zhang. A neural network robust control based on the dissipative theory for robot. Control Engineering China, 2010,17(6):853-855.

Haitao Zhang received the B. Sc. And M. Sc. degrees in mechanical engineering from Henan University of Science and Technology, PRC in 1994 and 1997, respectively, and the Ph. D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, PRC in 2006. He studied as a visiting scholar at University of Cincinnati in the United States in 2017. He is currently a professor in School of Information Engineering at Henan University of Science and Technology. His research interests include intelligent control and computer application technology.

E-mail: zhang haitao@163.com (Corresponding author)