Maestría en Ingeniería en Automatización de Procesos Industriales

Title

PD-type self-tuning regulator for trajectory tracking using RBF neural networks

Author

Donaciano Coyotecatl Cuautle

Contributor

Carlos Muñiz Montero

Juan A. Arizaga Silva

September-December 2014

PD-type Self-Tuning Regulator for Trajectory Tracking Using RBF Neural Networks

Donaciano Coyotecatl Cuautle, Carlos Muñiz Montero, Juan A. Arízaga Silva

Maestría en Ingeniería en Automatización de Procesos Industriales

donaciano.coyotecatl@uppuebla.edu.mx

Tercer Carril del Ejido Serrano S/N, San Mateo Cuanalá, Juan C. Bonilla, Puebla, México

1. Introduction

Trajectory tracking control is an important part of the motion control of manipulator robots. It demands robot arm to move in a free space following a desired trajectory without interacting with the environment. Unfortunately, the implementation of that kind of control required a precise knowledge of the dynamic model parameters of the arm, which is not a trivial task.

An alternative to solve trajectory tracking problem is operate the robot arm together a PD position controller with gravity compensation. The resulted closed loop systems is globally asymptotically stable. However, when constant PD gains are used, abrupt changes in the desired trajectories can result in high tracking errors.

In order to improve the performance of PD control in the trajectory tracking problem, a new neural control scheme called PD-N is proposed, which does not requires knowledge of the dynamic model parameters and can modify the gains K_p and K_v depending on the desired positions in the joint space. This goal is achieved using Radial Basis Function Neural Networks (RBFNN´s) .

2. Aim

Build a prototype robot manipulator with two degrees of freedom to incorporate position control and propose a PD-type self-tuning control using Radial Basis Function Neural Networks (RBFNN) for trajectories tracking.

PD-type self-tuning position control targets:

1.- Replace K_p and K_v gains by a radial basis neural network trained offline.

2.- Improve performance in PD position controller for tracking trajectories.

3.- Validate the control law with a specific path.

3. Method

In this section a methodology to calculate the values of the proportional and derivative gains automatically, as a function of the desired position in a PD-type control is presented, through of RBF neural networks, which it call interpolation networks and are trained offline.

a) Design *2n* RBF neural networks corresponding to the gains K_p and K_v in each articular joints, as shown in Figure 1. Where *n* are the degrees of freedom number (*dof*) in a manipulator robot.

b) Select *m* uniformly distributed desired positions into the manipulator workspace, as shown in Figure 3. These positions are in cartesian coordinates and converted via inverse kinematics to rotational space. The data sample form the set of training input $C=\{C_1, C_2, \ldots, C_m\}$, $C_j \in \Re^n$.

c) Tune the K_p and K_v gains manually for each position in the set *C* of previous step b). The set obtained data corresponds to the training output *2n* in step a). $K_i^P = \{K_{i1}^P, K_{i2}^P, \ldots, K_{nm}^P\}$ y $K_i^V = \{K_{i1}^V, K_{i2}^V, \ldots, K_{jm}^V\}$, where $i = 1,2,\ldots,n$, is the i-th *dof* and $j = 1,2,\ldots,m$, is the j-th training data.

d) Set σ value in accordance to attempt that the neurons activation level in the hidden layer is <50 percent, it is

$$\phi_{i,kj}^{P,V} = exp\left[-\left(\frac{\|Q_i-Q_j\|}{\sigma}\right)^2\right] < 0.5, \quad \forall i \neq j$$

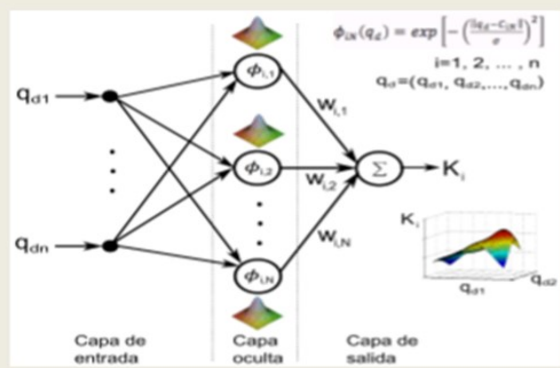

Figure 1. General structure of a Radial Basis Function Neural Network with i-th outputs.

e) Calculate the w_{ij}^P and w_{ij}^V weights of the neural network according to *2n* equations systems.

$$\begin{bmatrix} K_{i1}^{P,V} \\ K_{i2}^{P,V} \\ \vdots \\ K_{im}^{P,V} \end{bmatrix} = \begin{bmatrix} \phi_{i,11}^{P,V} & \phi_{i,12}^{P,V} & \cdots & \phi_{i,1m}^{P,V} \\ \phi_{i,21}^{P,V} & \phi_{i,22}^{P,V} & \cdots & \phi_{i,2m}^{P,V} \\ \vdots & \vdots & \cdots & \vdots \\ \phi_{i,m1}^{P,V} & \phi_{i,m2}^{P,V} & \cdots & \phi_{i,mm}^{P,V} \end{bmatrix} \begin{bmatrix} w_{i1}^{P,V} \\ w_{i2}^{P,V} \\ \vdots \\ w_{im}^{P,V} \end{bmatrix}$$

f) Replace the value data σ, C_j, w_{ij}^P and w_{ij}^V in the equations:

$$k_{pi}(q_d) = \sum_{j=1}^{N} w_{ij}^P \, exp\left[-\left(\frac{\|q_d - C_j\|}{\sigma}\right)^2\right] > 0, \qquad i = 1..n$$

$$k_{vi}(q_d) = \sum_{j=1}^{N} w_{ij}^V \, exp\left[-\left(\frac{\|q_d - C_j\|}{\sigma}\right)^2\right] > 0, \qquad i = 1..n$$

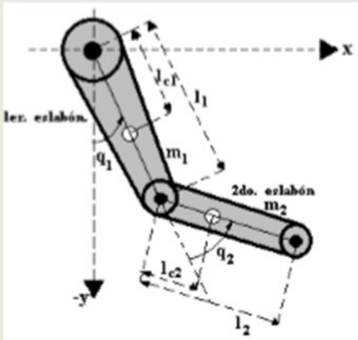

Figure 2. Manipulator robot with 2 dof.

Dynamic model of the manipulator robot

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f(\dot{q}) = \tau$$

PD position control

$$\tau = K_p\tilde{q} - K_v\dot{q} + g(q)$$

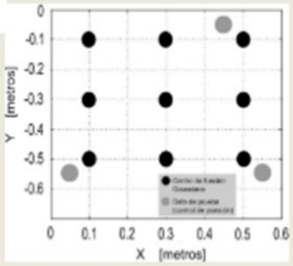

Figure 3. Interpolation Network with 9 training data in the Cartesian space.

PD-type self-tuning position control

$$\tau = diag\{k_{pi}(q_d)\}\tilde{q} - diag\{k_{vi}(q_d)\}\dot{q} + g(q)$$

4. Results

According to the methodology presented in Section 3, 9 samples were taken from cartesian space to train the interpolation network, see Figure 3. Further *2nm = 36* gains were tuned, for *n = 2 (dof)* and *m = 9* input data. The tuning criterion was: overshoot <1%, time response <1sec and in order to avoid saturation in the actuators the applied torque were $\tau_1 < 150Nm$ and $\tau_2 < 15Nm$. The sigma calculated value was σ=0.6. Finally to get the weights w_{ij}^P and w_{ij}^V value, four simultaneous equations was solved, as depicted in step e).

The result of the training process to achieve self-tuning of the proportional and derivative gains were four neural networks (see Figure 4) corresponding to K_p and K_v gains for the articulation one and two of the typical two-link manipulator robot shown in Figure 2.

In Figure 4 shows the behavior of the proportional and derivative gains approximated by RBF neural networks, which are functions of the desired positions q_1, q_2 in joint space and its values increased in points away from de robot base.

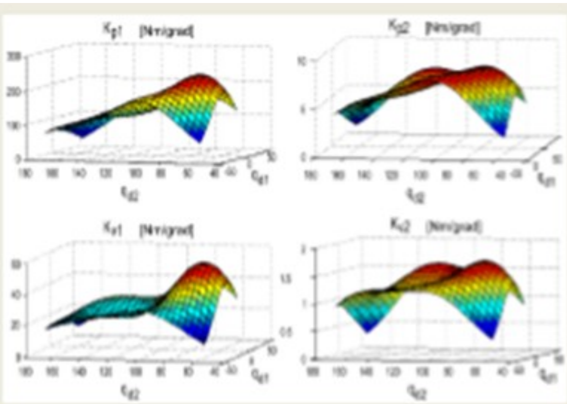

Figure 4. Proportional and derivative gains approximated by RBF Neural Network

In order to validate the new control scheme called PD-N, simulations for a specified trajectory as 8- petal flower was carried out. The results are compared with the corresponding PD control by taking into account the same conditions. The simulations were carried out using MatLab 2012b. The used parametric equations of the trajectory are presented below:

$$x = 0.3 + 0.25\cos(0.4t)\cos(0.1t)$$
$$y = -0.3 + 0.25\cos(0.4t)\sin(0.1t)$$

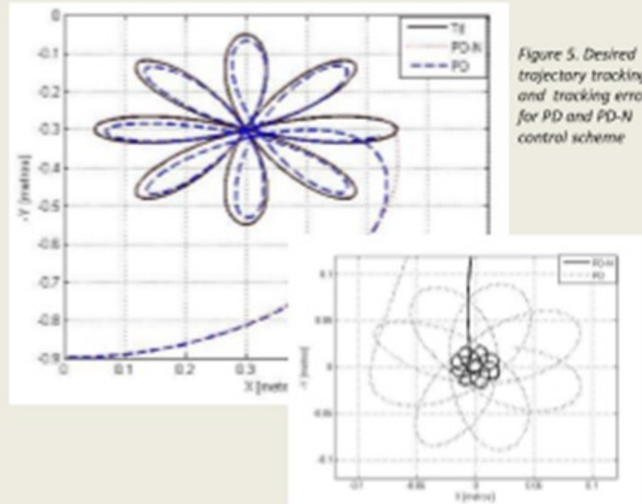

Figure 5. Desired trajectory tracking and tracking errors for PD and PD-N control scheme

In Figure 5 the trajectory tracking "eight-petal flower" with PD and PD-N controllers is presented. The simulation time is 63 seconds with a sampling period of 0.01 seconds, corresponding to 6300 samples. It is observed that PD-N control follow the desired trajectory with better performance than the PD control scheme even at points where the manipulator inertia increases, i.e. when the trajectory direction changes abruptly. Furthermore, it is observed that the tracking error is limited to a square region of 9 cm in the PD control and 2.5 cm for the proposed PD-N control. The average square error is 33.69 and 6.64 respectively .

5. Conclusion

The problem of trajectory tracking based on position control with automatic tuning of the proportional and derivative gains using RBF neural networks has been addressed in this poster. Unlike other authors the neural networks have been treated to create a training network as functions of desired positions, instead of approximating system unknown parameters. The new approach advantage is does not require the model knowledge to its implementation. Simulation results show better performance with the proposed PD-N control scheme in a desired trajectory tracking compared to the classical PD.

Acknowledgements

The completion of this work was possible thanks to the support of CONACYT, through a scholarship for graduate studies awarded to the first author of this article.

2014