ADVANCED ITERATIVE LEARNING CONTROLLERS FOR ROBOTIC SYSTEMS

Nicolae Constantin

University Politehnica of Bucharest, Faculty of Control and Computers,
Dept. of Automatic Control and Systems Engineering
313, Spl. Independentei, sector 6 RO-77206 Bucharest,
E -mail: nicu@cib.pub.ro

ABSTRACT- In this paper it is proposed an extended memory iterative learning technique. The knowledge of the iterative learning controller can be built by using the previous tasks of the iterative learning controller in tracking various desired trajectories in terms of a database of input and output data. For a new desired trajectory, iterative learning controller can predict the initial control input from this database and the tracking error converges to an acceptable level in less number of iterations.

Keywords: iterative learning, trajectory tracking, control, robotic systems.

INTRODUCTION

Iterative learning control is now a well-known technique for improving the tracking response in systems that repeat a given task or operation over and over again. Examples of such systems are robot manipulators that are required to repeat a given task with high precision, chemical batch processes, or, more generally, the class of repetitive systems.

Motivated by human learning, the basic idea of iterative learning control is to use information from the previous executions of the task in order to improve the performance from trial to trial in the sense that the tracking error is sequentially reduced.

In its principle, the iterative learning control technique [1,7] aims to improve the transient response and the tracking performance of systems that execute the same trajectory or operation over and over again. It consists in finding an adequate rule which allows the controller to learn from the tracking errors of the previous operations and perform progressively better with every new operation in order to achieve accurate tracking when the number of iterations increase.

Iterative learning control of continuous time systems, can be categorized into D-type in which the modification term is construct based on the error rate data of previous iteration $\delta u_i(t) = g(\dot{e}_i(t))$ and P-type in which $\delta u_i(t) = g(e_i(t))$. The function g (.) can be linear, i.e. for D-type iterative learning controller , $g(\dot{e}_i(t)) = \Gamma \dot{e}_i(t)$ and for P-type iterative learning controller $g(\dot{e}_i(t)) = \Phi e_i(t)$, where Γ and Φ are the learning gains.

The output of the system converges to the desired trajectory and tracking error converges to zero as number of iterations increases and goes to ∞. Acceptable bound of the tracking error can be defined by the user and iterative learning control terminates when the tracking error reduces below that bound.

For a new trajectory tracking task, iterative learning controller starts with no knowledge about the system. The tracking error is always large in the initial iterations. The number of iterations required to achieve a certain accuracy in the tracking is always high for every new desired trajectory.

It is proposed in this paper that knowledge of the iterative learning controller can be built by remembering the experience of the iterative learning controller in tracking various desired trajectories in terms of a database of input and output data.

ITERATIVE LEARNING CONTROL

A basic configuration of iterative learning control is shown in Figure 1. Let $q_d(t)$ be the desired trajectory. A control input $u_i(t)$ is applied to the system and the output of the system $q_i(t)$ is recorded in the *i*th iteration.

The error $e_i(t)$, difference between the system's output $q_i(t)$ and the desired trajectory $q_d(t)$ is used to generate the modification in the control input $\delta u_i(t)$.

The control input $u_{i+1}(t)$ for the (i+1)th iteration is generated by adding the modification term $\delta u_i(t)$ to the previous control input $u_i(t)$.

Hence iterative learning control takes the form

$$u_{i+1}(t) = u_i(t) + \delta u_i(t) \qquad (1)$$

where

$$\delta u_i(t) = g(\dot{e}_i(t)). \qquad (2)$$

Iterative learning scheme proposed in [1] by Arimoto et al. is of the form

$$u_{i+1}(t) = u_i(t) + \Gamma \dot{e}_i(t) \ , \forall t \in [0, t_f] \qquad (3)$$

The dynamic equation of a robotic manipulator having *n* degree of freedom can be described as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u \qquad (4)$$

where $q(t) \in R^n$ is the angular joint positions, M(q) is the inertia matrix, $C(q,\dot{q})\dot{q}$ is the Coriolis and centripetal forces, G(q) are the gravitational torques and $q(t) \in R^n$ are the input torques applied to the joints of manipulators.

The equation (4) can be described in state space form as follows:

$$\begin{aligned} \dot{x}_i(t) &= \varphi(x_i(t),t) + Bu_i(t) \\ y_i(t) &= Cx_i(t) \end{aligned} \qquad (5)$$

where **B** and **C** are the input and output matrices respectively, the state vector x(t) is $[q(t) \ \ \dot{q}(t)]$ and $\varphi(x_i(t),t) : R^n \times [0, t_f] \mapsto R^n$ is a nonlinear, piecewise continuous function which satisfies the Lipschitz condition.

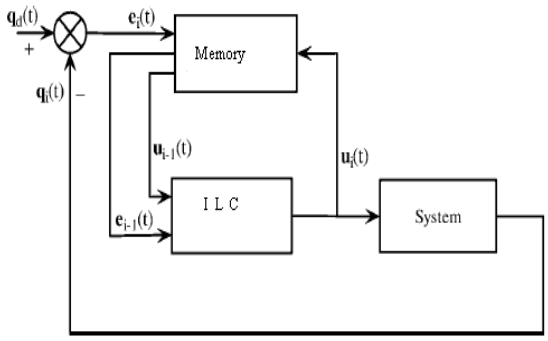

Fig.1

EXTENDED MEMORY ILC

A simple iterative learning control algorithm can be represented in series form as

$$u_{i+1}(t) = u_0(t) + \sum_{j=0}^{i} g(\dot{e}_j(t), e_j(t)) \ , \forall t \in [0, t_f] \qquad (6)$$

In many applications, it is not necessary that the desired trajectory be fixed and a controller may be required to track various desired trajectories.

In this situation, iterative learning control will start learning from the zero knowledge for each desired trajectory and will take almost the same time for the convergence of tracking error to an acceptable level.

Hence, it is desirable that controller should be capable of utilizing the controller's previous tasks into a new trajectory tracking task. As the number of tasks grows, controller should converge to the desired trajectory in less number of iterations.

The method proposed here can reduce the error in the initial iteration by utilizing previous tasks information. The data of the system's states, system's output and the corresponding control input for all iterations are stored in the form of a database.

A new desired trajectory is divided into many query points and for each query point, the control input is predicted by fitting a local model near the querypoint. The quality of the prediction of the control input depends on the population of the points near the query point in the database.

If the database is dense near the query point, a good prediction will be obtained, otherwise it will be poor. The search space of the database will depend on the type of trajectories, the system has tracked previously.

Locally weighted learning is becoming popular and many researchers are doing research in the application of the locally weighted learning in nonlinear control, more specifically in robotics applications [3, 5].

The main theme of the locally weighted learning is that any nonlinearity can be approximated by many local models which may be constant or linear ones. Therefore, for a complex nonlinear system like (4), instead of looking for a complex global model, it is easy to approximate the nonlinear function by using simple local models.

A new desired trajectory can be represented as many query points for which the desired control input has to be calculated. Hence as many local models as the query points can be approximated by searching the database for the nearby data points for each query point. Local models are fitted using weighted kernel regression technique near every query point. For each query point, nearby data

points are defined according to the euclidean distance of the data points from the query point.

These data points are weighted according to their distance from the query point. To search the nearby data points, *k*-nearest neighbor bandwidth selection as in [2] or as in [4] can be used.

The desired control input u_d for the query point x is computed as

$$u_d(x) = \sum_{i=1}^{k} w_i(d/h) f(p_i) \qquad (7)$$

where $w_i(d/h)$ is is the normalized kernel. Some weighting functions are Gaussian kernel, tricube kernel and quadratic kernel.

Here, a tricube kernel function that smoothly descends to zero and has continuous derivative is used

$$K(z) = \begin{cases} (1-z^3)^3 & ,|z| \le 1 \\ 0 & ,|z| > 1 \end{cases} \qquad (8)$$

The desired control input can be calculated for each query point using Equation (6). This control input is then used as the initial control input for the iterative learning controller. In this method, the data from previous tasks of the iterative learning controller for a system can be incorporated to guess the initial control input for a new desired trajectory. Initial control input is predicted from the previous data tasks of iterative learning control.

The error in the predicted control input will be corrected by the iterative learning controller. For good prediction, the set of query points should be inside the region of approximation to ensure that the weighted kernel regressor always interpolate.
To solve this problem, a supervisor can be used. Whenever a new desired trajectory is presented for the tracking, the supervisor checks whether the set of query points lies within the region or not. The initial control input is only generated when query points lie within the region of the database. If the query points lie outside, the supervisor forces the iterative learning controller to start from the zero knowledge.

The tracking error will converge smoothly when an iterative learning controller is applied to the nonlinear system given in Equation (4) and the initial control input is generated by fitting local models to the experience of the controller, as is shown in [2] with the following theorem

Theorem. Let an iterative learning control law mentioned in Equation (6) is applied to the nonlinear repetitive uncertain system like given in Equation (4). The initial control input $\mathbf{u}_0(t)$ for the iterative learning controller is calculated by fitting as many local models as the number of query points using weighted kernel regression technique to the database of previous tasks. The final tracking error will converge to zero as iteration i tends to ∞ , if the condition

$$\left\| I - CB\Gamma \right\|_\infty < 1 \,,\ \forall t \in [0, t_f] \qquad (9)$$

is fulfilled.

SIMULATION RESULTS

In the simulation, a nonlinear model of a two link robot is considered. The dynamic equation of the system is given in Equation (4).

For two link robot as shown in Figure 2, the elements of inertia matrix M(q)are

$M_{11} = m_2 l_1^2 + 2m_2 l_1 l_{c2} \cos q_2 + m_1 l_{c1}^2 + m_2 l_{c2}^2 + I_1 + I_2$

$M_{12} = M_{21} = I_2 + m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos q_2$,

$M_{22} = I_2 + m_2 l_{c2}^2$;

where m_i , I_i , l_{ci} and l_i are the mass, moment of inertia, length from center of the mass and total length of the link i, respectively.

The friction term is ignored and the system is considered as gravity compensated. The parameters of the two link robot are $m_1 = m_2 = 0.5$kg, $l_1 = l_2 = 0.5$m, $l_{c1} = l_{c2} = 0.25$m, $I_1 = I_2 = 0.011$kgm^2.

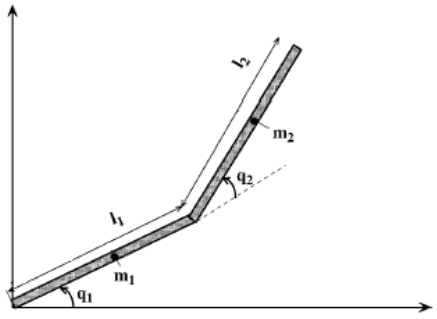

Fig. 2

In order to evaluate the performance the following index J is defined:

$$J = \sqrt{\frac{1}{T}\int_0^T e^2(t)dt} = \sqrt{\frac{1}{N}\sum_{k=0}^{N} e_k^2} \qquad (10)$$

where $e(t) = q_d(t) - q(t)$.

The sampling period is set to be 0.01 s. A D-type iterative learning controller is used where Γ is the learning gain vector and is equal to [0.01 0.01] for the smooth error convergence.

Standard D-type iterative learning controller is applied to the system for the desired trajectories for the joint angles of the two link robot manipulator shown in Figure 3.

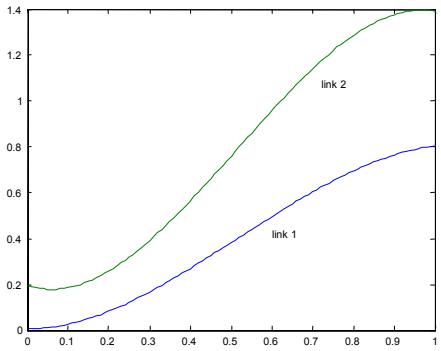

Fig. 3

The task was to move the two link manipulator from the initial joint angles [0 0.2] to the final joint angles [0.8 1.4] following a smooth trajectory. The data received from the system for 100 iterations is stored in a database.

The initial control input is generated by this database for another set of desired trajectories for the joint angles of the two links as shown in Fig. 4.

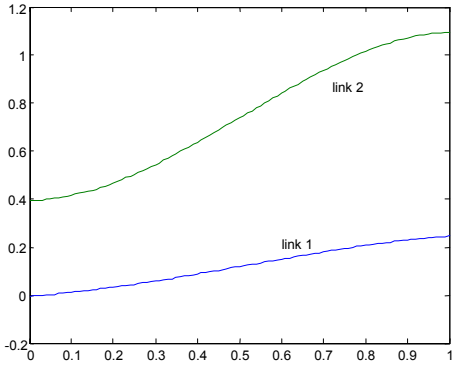

Fig. 4

This time the task is to move the manipulator from the initial joint angles [0 0.2] to the final joint angles [0.8 1.4] following a smooth trajectory.

The performance index J is plotted in Figure 5 for the standard D-type iterative learning controller without extended memory mentioned in Equation (3) and extended memory iterative learning controller. It can be observed from the profile of both performance indices that extended memory iterative learning controller has performed considerably better than standard Dtype iterative learning controller

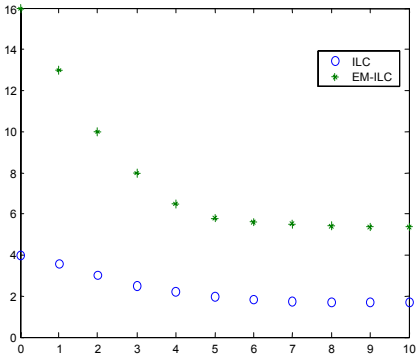

Fig. 5

CONCLUSIONS

In this paper, an extended memory iterative learning controller is proposed. The previous tracking tasks of the controller are stored in the database.

For a new desired trajectory, iterative learning controller can predict the initial control input form this database and hence the tracking error converges to an acceptable level in less number of iterations. Simulation results have shown the effectiveness of the proposed method.

REFERENCES

1. Arimoto S., and Miyazaki, F.: Bettering operation of robots by learning operation of robots by learning, Journal of Robotic Systems, vol. 1, 1984, pp. 123-140.
2. Arif, M., Ishihara, T., and Inooka, H., Experience-Based Iterative Learning Controllers. Journal of Intelligent and Robotic Systems, vol.35, 2002, pp. 381-396.
3. Atkeson, C. G., Moore, A. W., and Schaal, S.: Locally weighted learning for control, *Artificial Intell. Rev.* **11**(1), 1997, pp. 75–113.
4. Constantin N., A New Locally Weighted Regression Model , Proc. of Int. Conference CSCS14, Bucharest vol.1, 2003, pp.400-404.
5. Seidl, T. and Kriegel, H., Optimal multi-step k-nearest neighbor search, in: Proc. of ACM Int. Conf. On Management of Data,1998,pp. 55-69.
6. Owens, D. H., Rogers, E., and Moore, K., Analysis of linear iterative learning scheme using repetitive process theory. Asian Journal of Control, Vol.4, 2002, pp. 68–91.
7. Xu, J.X., and Viswanathan, B., Adaptive robust iterative learning control with dead zone scheme. *Automatica*, **36**, 2000, pp.91-99.