Using Experience to Get Better Convergence in Iterative Learning Control

Muhammad Arif, Tadashi Ishihara and Hikaru Inooka
Tohoku University, 01 Aramaki-aza, Aoba-ku, Sendai 980-8579, Japan

Abstract: A method of incorporating experience in iterative learning controllers is proposed in this paper. It is proposed that if the previous experience of the controller can be used in the selection of the initial control input for a new desired trajectory tracking task, the convergence of the iterative learning controller can be improved without modifying the structure of the controllers. Therefore the method proposed in this paper is very general and is applicable to most of the iterative learning control algorithms.

1 Introduction
Main objective of the tracking problems in the control theory is to force the output of the system to a desired trajectory as accurately as possible. The performance of the controller can be illustrated in terms of some specific norm of error or some other measures of optimality. Unmodelled dynamics or parameter uncertainties makes the job of achieving certain performance level difficult. A robot manipulator very well illustrates the problem of tracking the same desired trajectory for multiple times. Iterative learning control is one of the solutions to the problems of tracking the same desired trajectory for multiple times. Iterative learning controller utilizes the repetition of the task as an experience to improve the control quality. The beauty of the iterative learning control lies in its structural simplicity. It is different from the adaptive control because it does not require the identification of the system. The adaptation of iterative learning control is by modification of the control input in a way that the output of the system converges to the desired trajectory.

It is possible to get better convergence of the system's output to the desired trajectory, if the initial control input \( u_0(t) \) can be selected properly and efficiently. We can make a database of the system's states, system's output and the corresponding control input by using the computer memory. When a new desired trajectory is required to be tracked, the desired trajectory can be decomposed into many query points. A locally weighted regression method can be applied to predict the control input for the desired trajectory by using the information stored in the database. This predicted control input then can be used as an initial control input by the iterative learning controller for the new desired trajectory. In this way we can get better convergence of the system's output to the desired trajectory in lesser number of iterations.

2 Iterative Learning Control - Preliminaries
The iterative learning control algorithm takes the form,

\[
 u_{i+1}(t) = u_i(t) + g(e_i(t), e_i(t)) \quad 0 \leq t < T \tag{1}
\]

The first iterative learning scheme was proposed by Arimoto et al. [1]. He used the derivative of error \( e_i(t) = y_d(t) - y_i(t) \), where \( i \) is the iteration number. The algorithm is of the form,

\[
 u_{i+1}(t) = u_i(t) + \Gamma \dot{e}_i(t), \forall t \in [0, T] \tag{2}
\]

A non-linear time varying dynamic system can be described as,

\[
 x_i(t) = f(x_i(t), t) + B(t)u_i(t) \tag{3}
\]

\[
 y_i(t) = C(t)x_i(t)
\]

where \( i \) corresponds to the \( i \)th iteration of the system; \( x_i(t) \in \mathbb{R}^n, u_i(t) \in \mathbb{R}^m \) and \( y_i(t) \in \mathbb{R}^r \) are the states, control input and output of the system respectively for \( t \in [0, T] \); \( f(x_i(t), t) : R^n \times [0, T] \rightarrow R^n \) is a non-linear, piecewise continuous function of \( x_i(t) \) and \( t \).

\( f(.) \) satisfies the Lipschitz condition as follows

\[
 \| f(x_2(t), t) - f(x_1(t), t) \| \leq L_f \| x_2(t) - x_1(t) \| \tag{4}
\]

where \( L_f > 0 \) is Lipschitz constant.

**Theorem 1** (Kawamura et al. 1986[2]) By applying the iterative learning control law mentioned in equation (2) to the non-linear time varying repetitive system given in equation (3) and incorporating the condition,

\[
 \rho < 1 \quad \text{for} \quad \forall t \in [0, T] \tag{5}
\]

PR0001-3/99/0000-1211 $4.00 © 1999 SICE
where
\[ \rho \equiv \|I_s - CB\|_\infty \] (6)

the final tracking error will be bounded for sufficiently large \( \lambda \).

3 Annexation of Experience in Iterative Learning Control

A simple iterative learning control algorithm can be represented in series form as,
\[ u_{i+1}(t) = u_0(t) + \sum_{j=1}^{i} g(e_j(t), e_j(t)), \forall t \in [0, T] \] (7)

All the research till now is concentrated on the second part of equation (7), i.e. how to design a better function \( g(.) \) to get a faster and smooth convergence of the error to zero, which is certainly the main structure of the iterative learning control. But the first part of the equation (7) i.e. the initial control input \( u_0(t) \) also plays an important rule in the convergence of the error and should be taken into consideration.

Iterative learning control learns through iteration and if the convergence condition given in equation (5) is satisfied, the error will converge to zero as the iteration number \( i \) tends to infinity, but in practical applications, we usually terminate the process when an acceptable performance level is achieved. Iterative learning controller does not require a priori knowledge of the system dynamics, therefore large error results in early iterations and the error converges slowly to zero as the iteration number \( i \) tends to infinity. In the situations, when it is required to perform the tracking task for multiple desired trajectories, it is desirable that the controller can learn the model of the complex non-linear functions so that this learning can be incorporated to get a faster convergence to the desired trajectory. We propose a scheme of iterative learning control which can improve its learning rate from its previous encounters with the system and can better understand the system.

In our approach, the data about the system’s states, system’s output and the corresponding control input for all iterations will be stored in the computer memory. The idea is to remember all the previous experience and whenever a new desired trajectory is required to be tracked, we will predict the control input for this desired trajectory by using the previous experience and will use it as an initial control input for the iterative learning algorithm. This prediction can be done by dividing the desired trajectory into many query points and for each query point (a data point for which the control input is desirable), the control input is predicted by fitting a linear model near the query point.

The convergence condition quoted by many researchers in the field of iterative learning control as,
\[ \|e_{i+1}\|_\lambda \leq \rho \|e_i\|_\lambda \] for \( \rho < 1 \) (8)

To illustrate the effectiveness of our proposed method, the above inequality can be rewritten as,
\[ \|e_{i+1}\|_\lambda \leq \rho^i \|e_0\|_\lambda \] for \( \rho < 1 \) (9)

Hence the error at \( i^{th} \) iteration depends on two parameters. One is the selection of the parameter \( \rho \) which controls the convergence rate and the other parameter is the initial error \( e_0(t) \). In our proposed method, by selecting the initial control input \( u_0(t) \) using the previous experience of the controller, we are going to decrease the initial error \( e_0(t) \) which in turn will effectively reduce the error at \( i^{th} \) iteration.

3.1 Locally weighted Learning

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. A good survey can be found in [3]. Locally weighted learning is a memory based learning system. The main theme of the locally weighted learning is that any nonlinearity can be approximated by a linear model, if the output surface is smooth. Therefore instead of looking for a complex global model, it is easy to approximate the nonlinear functions by using simple local models.

Let a nonlinear system is defined as,
\[ y(t) = z(x(t), u(t)) \] (10)

where \( z(.) \) is a nonlinear function.

The task is the proper choice of the control input \( u(t) \) such that the output of the system \( y(t) \) follows the desired trajectory \( y_d(t) \). If the desired output \( y_d(t) \) and states \( x_d(t) \) are given then it is possible to compute the desired control input \( u_d(t) \) by using the inverse model [3] of the system as,
\[ u_d(t) = z^{-1}(x_d(t), y_d(t)) = \mu(x_d(t), y_d(t)) \] (11)

This inverse model can be represented by a database of experiences i.e. the system states \( x(t) \), system’s output \( y(t) \) and corresponding control input \( u(t) \). Here we are not talking about one global inverse model but many local inverse models according to the query point. We are not going to make any strong assumption globally about \( \mu(.) \), however \( \mu(.) \) can be approximated by a polynomial around a local point or just a locally weighted average of the nearby points. The only drawback of this approach is that while selecting the desired input \( u_d(t) \) from the inverse model, we do not know the outcome of the system. But since we are applying the inverse model approach to the iterative learning controller, therefore any error remains after the application of the desired control input generated by the inverse model will be corrected by the iterative learning controller.

Once a query point vector \( q(t) = (x_d(t), y_d(t)) \) is defined, a linear local model can be constructed using the data points near the query point according to the
euclidean distance of the data points from the query point. The data points are represented as \((x(t), y(t))\), where \(x(t)\) is the system states vector and \(y(t)\) is the output of the system. These data points are weighted according to their distance from the query point. The data points that are near to the query point are given more weights as compared to the data points that lies far from the query point. The gaussian kernel, used as the weighting kernel, has infinite extent and defined as,

\[
K(d/h) = e^{-(d/h)^2} \quad (12)
\]

In this kernel, \(d\) is the euclidean distance defined as,

\[
d(x, q) = \sqrt{(x - q)^T(x - q)} \quad (13)
\]

The bandwidth \(h\) also plays an important role in local modelling. If \(h\) is too small, then it is possible that we will not have too many data points in the neighborhood for a good prediction. On the other hand if \(h\) is infinite then the local modelling becomes global modelling. In most of the cases, the volume of data points varies according to the density of the data points near the query point. Hence we will adopt the \(k\)-nearest neighbor bandwidth selection. In this technique, \(h\) is not a constant but varies according to the density of the data points near the query point. \(k\)-nearest neighbor search corresponds to the retrieval of the \(k\) data points that are most near to the query point \(q\).

**Definition 1 (k-nearest neighbor search[4])** Given a query point \(q\) and a query parameter \(k\), the \(k\)-nearest neighbor search method will return the smallest set \(NN_q(k) \subseteq DB\) that contains \(k\) data points from the database \(DB\) and for which the following condition holds;

\[
\forall p \in NN_q(k), \forall s \in DB - NN_q(k) : d(p, q) < d(s, q) \quad (14)
\]

For a certain query point, \(h\) will be the distance of the query point to the \(k\)-nearest data point to the query point, i.e.,

\[
h = \max\{d(p, q) \mid p \in NN_q(k)\} \quad (15)
\]

The locally weighted regression learning can be described as follow: Lets begin with a one variable system, the \(\mu(x)\) can be represented as a local model,

\[
\mu(x) = \beta_0 + \beta_1 x + ... \quad (16)
\]

The parameters \(\beta_0, \beta_1, \ldots\) can be found by minimizing the locally weighted sum of squared residuals,

\[
J_\beta = \frac{1}{2} |W(\mu - \hat{\mu})|^2 = \frac{1}{2} \sum_{i=1}^{m} (w_i(\mu_i - \hat{\mu}_i))^2 \quad (17)
\]

For two variables system, a multivariate local model can be represented as,

\[
\hat{\mu}(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 \ldots \quad (18)
\]

which can be written as,

\[
\hat{\mu}(x) = \beta^T A(x) \quad (19)
\]

where

\[
A(x) = [1 \ x_1 \ x_2 \ x_1^2 \ x_2^2 \ ... \ x_1 x_2 \ ... ]^T \quad (20)
\]

By minimizing the functional \(J_\beta\) as given in equation (17), we get \(\beta\) as,

\[
\beta = [(WX)^T WX]^{-1} (WX)^T W u \quad (21)
\]

\(X\) is \(k \times (n+1)\) matrix of the \(k\) data points that are searched by using the \(k\)-nearest neighbor method. \(u\) is \(k \times d\) matrix of the corresponding inputs and \(W\) is the diagonal \(k \times k\) matrix of weights. The \(i^{th}\) weight \(w_i\) for the \(i^{th}\) data point is given as,

\[
w_i = \sqrt{K(d/h)} \quad (22)
\]

where \(K(d/h)\) is the weighting kernel defined in equation (12). The local regression estimate \(\hat{\mu}(q)\) will be the first component of the parameter vector \(\beta\). Selection of the initial control input \(u_0(t)\) for a new desired trajectory can be done by decomposing the new desired trajectory into many query points. For each query point \(q\), the predicted value of the control input \(\hat{u}(q)\) using the regression parameter vector \(\beta(q)\) can be found by finding the set of \(k\) nearest neighbor data points \(NN_q(k)\) from the database and calculating the weight matrix \(W\) for these \(k\) data points as given in equation (22).

### 4 Numerical Illustrations

In this section, we will demonstrate the effectiveness of the proposed method. For this purpose, simulation of a nonlinear model of a single link manipulator has been done.

The dynamic equation of the system is

\[
I \ddot{\theta} + \frac{1}{2} m + M)g \sin \theta = \tau \quad (23)
\]

where \(\theta(t)\) is the angular position of manipulator, \(\tau(t)\) is the deriving torque, \(m, M\) and \(l\) are the mass of manipulator, tip load and length of manipulator respectively. \(I\) is the moment of inertia with respect to the joint and defined as

\[
I = M^2 + \frac{1}{3} m l^2 \quad (24)
\]

Let \(m = 1.0 \text{ kg}, M = 2.0 \text{ kg}, l = 0.5 \text{ m}\) and \(g = 9.8 \text{ m/s}^2\) be chosen as system parameters and the sampling period is 0.01 seconds.

Performance index \(J\) is defined as follows to evaluate the performance.

— 1213 —
A D type iterative learning controller, given in equation (2), is chosen for this numerical illustration. The learning gain \( K \) is set to be 0.2 for the smooth error convergence.

At first we have used the above controller to track a desired trajectory \( OT \) as shown in the Figure 1. All the data points \((x_i(t), y_i(t), u_i(t))\) are stored in the database \( DB \) to be used as an experience of the controller for the other trajectories.

In the next step, a new desired trajectory \( NT \) as shown in the Figure 2 is selected for the tracking purpose. Comparing the trajectories \( OT \) and \( NT \), it is obvious that the trajectory \( NT \) is not similar to the trajectory \( OT \) for which the database of experience is built. We have used the database that is built by using the data of the tracking task of the trajectory \( OT \) as an experience to predict the initial control input for the new desired trajectory \( NT \). To search the data points near the query point, we have used \( k \)-nearest neighbor method and \( k \) is selected to be 3. The prediction of the initial control input for the new desired trajectory is done by using the locally weighted regression method in which local model is a constant. We have performed the simulations with and without using the predicted control input as an initial control input. The performance index \( J \) is plotted in the Figure 3 for both the cases. It is very clear from the figure that the performance is improved drastically when we have used the predicted control input as an initial control input for the trajectory \( NT \).

5 Conclusions

Very simple structure and easy implementation made iterative learning control algorithms popular amongst the learning control algorithms, especially in robotics applications. We have proposed a new method of incorporation of the experience of iterative learning controllers about the tracking of the previous trajectories in the tracking task of some new trajectory. In this way it is possible to get better convergence without sacrificing the simplicity of the iterative learning control schemes. Prediction of the initial control input for a new trajectory using the experience of the previous trajectories is done by locally weighted regression method. The selection of the initial input is independent of the structure of the controller, therefore this method is very general for all class of the iterative learning control schemes.

References