Proceedings of the ASME 2016 Dynamic Systems and Control Conference
DSCC2016
October 12-14, 2016, Minneapolis, Minnesota, USA

DSCC2016-9759

A TASK-LEVEL ITERATIVE LEARNING CONTROL ALGORITHM FOR ACCURATE TRACKING IN MANIPULATORS WITH MODELING ERRORS AND STRINGENT JOINT POSITION LIMITS

Pranav A. Bhounsule*, **Abhishek A. Bapat**
Department of Mechanical Engineering
University of Texas San Antonio
One UTSA Circle, San Antonio, TX 78249, USA.
Email: pranav.bhounsule@utsa.edu,
abhishek.bapat1@gmail.com

Katsu Yamane
Disney Research
4720 Forbes Avenue, Lower Level, Suite 110
Pittsburgh, PA 15213, USA.
Email: kyamane@disneyresearch.com

ABSTRACT

We present an iterative learning control algorithm for accurate task space tracking of kinematically redundant robots with stringent joint position limits and kinematic modeling errors. The iterative learning control update rule is in the task space and consists of adding a correction to the desired end-effector pose based on the tracking error. The new desired end-effector pose is then fed to an inverse kinematics solver that uses the redundancy of the robot to compute feasible joint positions. We discuss the stability, the rate of convergence and the sensitivity to learning gain for our algorithm using quasi-static motion examples. The efficacy of the algorithm is demonstrated on a simulated four link manipulator with joint position limits that learns the modeling error to draw the figure eight in 4 trials.

1 Introduction

Manipulator systems such as industrial robots are typically used to do repetitive tasks in end-effector space or task space. End-effector space control is typically done by computing a mapping from the task- to joint- space. However, finding this mapping in the presence of joint position limits is significantly challenging. The use of constraint optimization provides an easy and generalizable method to find inverse kinematics solutions in the presence of joint position limits [1, 2]. But the optimization is sensitive to the availability of a good kinematics model. When the robots parts are replaced (e.g., due to wear and tear), the model parameters change and the inverse kinematics is not capable of providing precise end-effector control. In such cases, the Iterative Learning Control (ILC) algorithm provides an relatively easy and quick way to re-tune the robot motion [3].

Iterative Learning Control is technique that iteratively refines feed-forward commands based on tracking errors in the previous trials [4]. The simplest learning update rule is

$$\mathbf{U}^{i+1} = \mathbf{U}^i + \gamma(\mathbf{Y}_{\text{ref}} - \mathbf{Y}^i), \tag{1}$$

where i is the iteration, γ is the user defined learning parameter, $\mathbf{U}^i$ is the control command at iteration i (e.g., torque, force, current), $\mathbf{Y}^i$ is the quantity that needs to be tracked (e.g., joint angle, end-effector position) and is measured at each iteration using external sensing, and $\mathbf{Y}_{\text{ref}}$ is the reference motion. The goal is to apply this learning rule repeatedly such that the the tracking error, $\mathbf{e}^i = (\mathbf{Y}_{\text{ref}} - \mathbf{Y}^i)$ reduces at each iteration, subsequently reaching a value close to zero.

The most common use of ILC is in the joint space [5, 6], where the tracking error, $\mathbf{e}^i$, are the positions and/or velocities and the control command, $\mathbf{U}^i$, is the torque or the voltage/current for DC motors or the valve position for hydraulic/pneumatic actuators. However, in this paper we are interested in improving the tracking performance in the task or the end-effector space of redundant manipulators. Specifically, the dimension of $\mathbf{U} >$ di-

*Address all correspondence to this author.

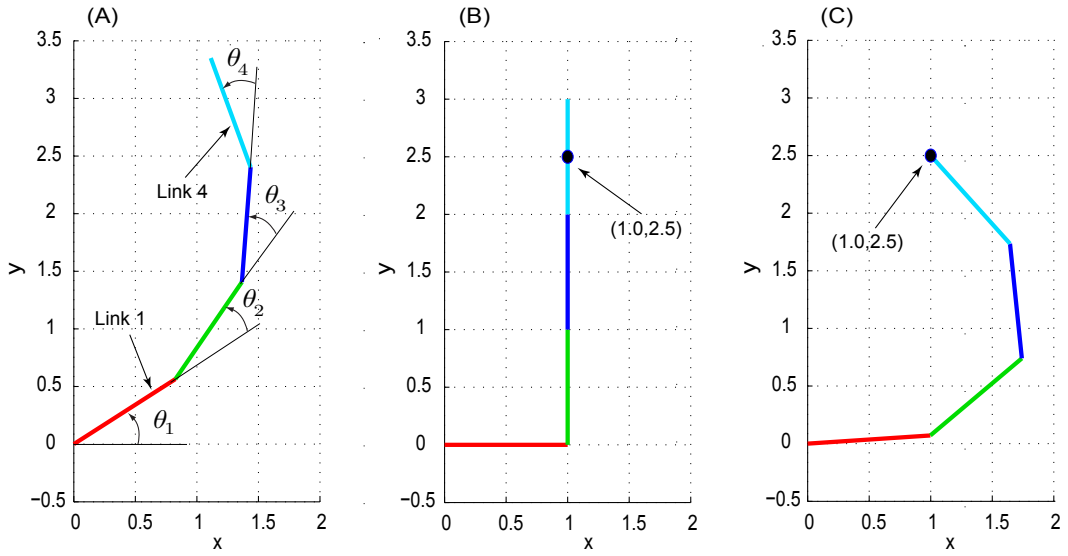

FIGURE 1. (A) THE FOUR LINK MANIPULATOR WITH ANGLES SHOWN (B) THE INITIAL POSITION OF THE MANIPULATOR IS SHOWN WITH THE END-EFFECTOR AT $\{1,3\}$. THE GOAL IS TO MOVE THE END-EFFECTOR TO THE POSITION $\{1,2.5\}$. ARIMOTO'S ILC UPDATE BASED ON THE JACOBIAN TRANSVERSE MOVES THE JOINT 1 DOWN BUT THAT VIOLATES THE JOINT LIMIT. (C) OUR ILC ALGORITHM WHICH IS BASED ON CONSTRAINT OPTIMIZATION IS ABLE TO FIND A MANIPULATOR POSITION THAT RESPECTS THE JOINT POSITION LIMITS.

mension of $\mathbf{Y}$, which means that the mapping from task space $\mathbf{Y}$ to $\mathbf{U}$ is non-unique.

In order to adapt ILC to for end-effector space control, Arimoto et al. [7] suggested the use of the transpose of the Jacobian ($\mathbf{J}$) of the map from joint- to task-space. We have modified their update rule for the *static* case and is given below

$$\theta^{i+1} = \theta^i + \gamma \mathbf{J}^T(\mathbf{Y}_{\text{ref}} - \mathbf{Y}^i), \tag{2}$$

where θ is the joint position and $\mathbf{Y}$ is the end-effector position.

We claim that Arimoto's update rule (Eqn. 2) fails to improve performance in some instances when the joint position is close to the limits. We illustrate this next using a four link manipulator example.

A four link manipulator is shown in Fig. 1 (a). Link 1 is hinged to the ground and the far end of link 4 is the end-effector position. Each link is of length 1 and the angles are $\theta_1, \ldots \theta_4$ as shown. The only joint limit on the manipulator is that on link 1, $0 \leq \theta_1 \leq \pi/2$. The x and y positions of the end-effector are given by

$$x = c_1 + c_{12} + c_{123} + c_{1234}, \tag{3}$$
$$y = s_1 + s_{12} + s_{123} + s_{1234}, \tag{4}$$

where $s_1 = \sin(\theta_1)$, $c_1 = \cos(\theta_1)$, $s_{12} = \sin(\theta_1 + \theta_2)$, $c_{12} = \cos(\theta_1 + \theta_2)$ and so on. The Jacobian is obtained by taking the partial derivative of the position of the end-effector with respect to joint angles, $\theta_1, \theta_2, \theta_3, \theta_4$. After taking the transpose we get $\mathbf{J}^T$ to be

$$\begin{pmatrix} -s_{1234} - s_{123} - s_{12} - s_1 & c_{1234} + c_{123} + c_{12} + c_1 \\ -s_{1234} - s_{123} - s_{12} & c_{1234} + c_{123} + c_{12} \\ -s_{1234} - s_{123} & c_{1234} + c_{123} \\ -s_{1234} & c_{1234} \end{pmatrix}$$

Assume that the manipulator is in the position shown in Fig. 1 (b). The angles are $\theta_1 = 0$, $\theta_2 = \pi/2$, $\theta_3 = \theta_4 = 0$. Note that θ_1 is at its joint limit ($0 \leq \theta_1 \leq \pi/2$). The end-effector position in this case is at $\mathbf{Y} = \{x, y\} = \{1, 3\}$. We want the end-effector to move to the position $\mathbf{Y}_{\text{ref}} = \{1, 2.5\}$. Assuming $\gamma = 1$, we can compute the Arimoto's update from Eqn. 2

$$\theta^{i+1} = \begin{pmatrix} 0 \\ \pi/2 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -3 & 1 \\ -3 & 0 \\ -2 & 0 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ -0.5 \end{pmatrix} = \begin{pmatrix} -0.5 \\ \pi/2 \\ 0 \\ 0 \end{pmatrix}. \tag{5}$$

But the value of $\theta_1 = -0.5$ violates the joint limit (Note: $0 \leq \theta_1 \leq \pi/2$). Thus Arimoto's algorithm has failed when the manipulator's angles are at the joint limit.

In this paper, we present a new iterative learning scheme that is capable of improving task-level performance using imperfect kinematics models and with stringent joint limits. At iteration i, we can get the measured end-effector position, $\mathbf{Y}^i$, from the given joint position, θ^i, and using the true kinematics model, $\mathbf{F}$ to get

$$\mathbf{Y}^i = \mathbf{F}(\theta^i). \tag{6}$$

Then, our update scheme is given by

$$\mathbf{Y}^{i+1}_{\text{des}} = \mathbf{Y}^i_{\text{des}} + \gamma(\mathbf{Y}_{\text{ref}} - \mathbf{Y}^i), \tag{7}$$
$$\theta^{i+1} = \hat{\mathbf{F}}^{-1}(\mathbf{Y}^{i+1}_{\text{des}}), \tag{8}$$

where $\mathbf{Y}_{\text{des}}$ are the desired end-effector position and $\hat{\mathbf{F}}^{-1}$ is the inverse of the approximate kinematic model $\hat{\mathbf{F}}$. Eqn. 7 is the update rule and the Eqn. 8 maps the end-effector desired position, $\mathbf{Y}_{\text{des}}$, to the joint space, θ^{i+1}. In our algorithm, a constraint optimization ensures that the inverse, $\hat{\mathbf{F}}^{-1}$, is within the joint position limits. The cost function and constraints for the optimization are as follows

$$g(\theta) = \sum_{i=1}^{n_{\text{dof}}} (\theta_i - \theta_i^{\text{rest}})^2, \tag{9}$$
$$\begin{aligned} h_1(\theta) &= x_{\text{des}} - x_{\text{ref}} = 0, \\ &= c_1 + c_{12} + c_{123} + c_{1234} - x_{\text{ref}} = 0, \end{aligned} \tag{10}$$
$$\begin{aligned} h_2(\theta) &= y_{\text{des}} - y_{\text{ref}} = 0, \\ &= s_1 + s_{12} + s_{123} + s_{1234} - y_{\text{ref}} = 0, \end{aligned} \tag{11}$$
$$h_3(\theta) = \theta - \theta_{\text{min}} \geq 0, \tag{12}$$
$$h_4(\theta) = \theta - \theta_{\text{max}} \leq 0. \tag{13}$$

The first equation is the cost. The next two equations are the constraints on the end-effector position and the final two are the position limits. We use the nonlinear optimization software SNOPT (Sparse Nonlinear OPTimizer, [8]), a robust implementation of sequential quadratic programming with constraints. The constant value for the rest position, θ_i^{rest}, biases the solution towards it. We heuristically chose a value of 10^{-3} but not equal to zero to avoid manipulator singularity. In general, we are interested in redundant systems, so there are multiple solutions to the inverse kinematics problem. However, by introducing a cost function as we have done here (that is, $||\theta_i - \theta_i^{\text{rest}}||$), we are able to find a unique solution. A related idea is the norm optimal iterative learning control where the squared sum of the error and control is minimized [9].

When we apply the above inverse kinematics scheme, we get the results shown in Fig. 1 (c) with $\theta_1 = 0.0710$, $\theta_2 = 0.6620$, $\theta_3 = 0.9342$, and $\theta_4 = 0.6039$. Note that the value of θ_1 is within the joint limits. This way we are able to find a solution that is within the joint position limits.

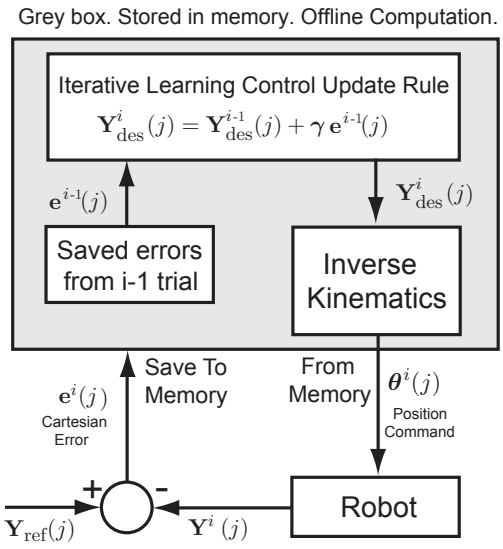

FIGURE 2. BLOCK DIAGRAM OF OUR TASK-SPACE ITERATIVE LEARNING CONTROL ALGORITHM.

We have successfully used the above method to do end-effector control of a 26 degree of freedom humanoid robot [10]. But that work was limited to the application of the algorithm. In this paper, we discuss the stability, the rate of convergence, and the sensitivity to learning gain for our algorithm using simple static motion examples.

The rest of the paper is organized as follows. We describe our algorithm in detail in Sec. 2 and the performance metrics of the algorithm. Next, we illustrate the stability property and rate of convergence of the algorithm for various learning gains for a two link manipulator doing a pointing task. Finally, we show how a four link manipulator with stringent joint position limits iteratively learns to draw the figure eight using the algorithm.

2 Iterative Learning Control Algorithm

2.1 Algorithm

Let i represent the trial number and j the time index that goes from 1 to n_j (end time). Let the reference motion in task space be defined by $\mathbf{Y}_{\text{ref}}(j)$. The input to the inverse kinematics solver are desired poses in end-effector space, which we denote by $\mathbf{Y}^i_{\text{des}}(j)$. Let the measured position in task space be defined by $\mathbf{Y}^i(j)$. Let the error between actual end-effector position and reference position be denoted by $\mathbf{e}^i(j) = \mathbf{Y}^i_{\text{ref}}(j) - \mathbf{Y}^i(j)$. The ILC algorithm is depicted in Fig. 2 and described below:

1. Set the error $\mathbf{e^0}(j) = 0$ and initialize the desired position in the task space $\mathbf{Y}^0_{\text{des}}(j) = \mathbf{Y}_{\text{ref}}(j)$.
2. For subsequent trials, $i = 1, 2, 3...$, do:

    1. Command modification in end-effector space:

$$\mathbf{Y}^i_{\text{des}}(j) = \mathbf{Y}^{i-1}_{\text{des}}(j) + \boldsymbol{\gamma}^i \mathbf{e}^{i-1}(j), \tag{14}$$

    where $\boldsymbol{\gamma}^i$ is a manually tuned learning parameter.
    2. Command modification in joint-space: Find a joint space update that is within the joint position limits using constraint optimization.

$$\theta^{i+1} = \hat{\mathbf{F}}^{-1}(\mathbf{Y}^{i+1}_{\text{des}})$$

    3. Command execution on robot: The new joint position, $\theta^i(j)$, is executed on the robot. Measure the end-effector position, $\mathbf{Y}^i(j)$ and save the tracking errors in end-effector space $\mathbf{e}^i(j)$, for $j = 1, 2, ..., n_j$).
3. Stop when the error metric e^i_{norm} is below a threshold value set by the control designer. The learnt joint position is then $\theta^i(j)$ $(j = 1, 2, ..., n_j)$.

The error metric at iteration i is given by

$$e^i_{\text{norm}} = \frac{1}{n_j} \sum_{j=1}^{n_j} \sum_{k=1}^{n_{\text{eff}}} (\mathbf{e}^i_k(j))^2 \tag{15}$$

where $\mathbf{e}^i_k(j)$ is the tracking error in the pose element k, at iteration i and at time j, n_j is the total data points in the trial, n_{eff} is the number of pose elements considered. For example, for a two link manipulator when we are interested in controlling the x- and y-position of the end-effector, $n_{\text{eff}} = 2$.

Note that we assume the motion is static and hence there is no dependence on the velocities. Also, we assume that there is high gain position servo that can achieve the given desired joint position, $\theta^i(j)$, on the robot.

2.2 Performance of the algorithm

We derive formulas for the convergence and stability of the algorithm.

2.2.1 System equations: The ILC update equation is:

$$\mathbf{Y}^i_{\text{des}}(j) = \mathbf{Y}^{i-1}_{\text{des}}(j) + \boldsymbol{\gamma}^i e^{i-1}(j). \tag{16}$$

In this paper, we assume that the parametric uncertainty is in the length parameters. We note that the end-effector position is linearly dependent on the length, and hence to the parametric uncertainty. Thus, the measured end-effector pose, $\mathbf{Y}^i$, is going to be linearly proportional to the desired end-effector position, $\mathbf{Y}^i_{\text{des}}$,

$$\mathbf{Y}^i(j) = \mathbf{G}^i_j \mathbf{Y}^i_{\text{des}}(j) \tag{17}$$

where $\mathbf{G}^i_j$ is a diagonal matrix of size 6×6 if position and orientations are both considered in the formulation. Note that $\mathbf{G}$ is a function of iteration number, i, time sample, j, and is a linear function of the parametric uncertainty in the link lengths.

2.2.2 Convergence analysis: To do convergence analysis, we need to relate the errors between successive trials. This is done as follows.

$$\begin{aligned}
\mathbf{e}^i(j) &= \mathbf{Y}_{\text{ref}}(j) - \mathbf{Y}^i(j) \\
&= \mathbf{Y}_{\text{ref}}(j) - \mathbf{G}^i_j \mathbf{Y}^i_{\text{des}}(j) \\
&= \mathbf{Y}_{\text{ref}}(j) - \mathbf{G}^i_j \mathbf{Y}^{i-1}_{\text{des}}(j) - \mathbf{G}^i_j \boldsymbol{\gamma}^i \mathbf{e}^{i-1}(j) \\
&= \mathbf{Y}_{\text{ref}}(j) - \mathbf{Y}^{i-1}(j) - \mathbf{G}^i_j \boldsymbol{\gamma}^i \mathbf{e}^{i-1}(j) \\
&= \mathbf{e}^{i-1}(j) - \mathbf{G}^i_j \boldsymbol{\gamma}^i \mathbf{e}^{i-1}(j) \\
&= (\mathbf{I} - \mathbf{G}^i_j \boldsymbol{\gamma}^i) \mathbf{e}^{i-1}(j)
\end{aligned} \tag{18}$$

The condition for convergence is that the $|\mathbf{e}^{i-1}(j)| < |\mathbf{e}^i(j)|$. This condition is met when the magnitude of all eigenvalues of $(\mathbf{I} - \mathbf{G}^i_j \boldsymbol{\gamma}^i)$ are less than 1.

2.2.3 Stability analysis: We simplify the ILC update equation as follows:

$$\begin{aligned}
\mathbf{Y}^i_{\text{des}}(j) &= \mathbf{Y}^{i-1}_{\text{des}}(j) + \boldsymbol{\gamma}^i (\mathbf{Y}_{\text{ref}}(j) - \mathbf{Y}^{i-1}(j)) \\
&= \mathbf{Y}^{i-1}_{\text{des}}(j) + \boldsymbol{\gamma}^i \mathbf{Y}_{\text{ref}}(j) - \boldsymbol{\gamma}^i \mathbf{G}^i_j \mathbf{Y}^{i-1}_{\text{des}}(j) \\
&= (\mathbf{I} - \boldsymbol{\gamma}^i \mathbf{G}^i_j) \mathbf{Y}^{i-1}_{\text{des}}(j) + \boldsymbol{\gamma}^i \mathbf{Y}_{\text{ref}}(j)
\end{aligned} \tag{19}$$

The condition for stable learning is that the control command, $\mathbf{Y}^i_{\text{des}}(j)$ should be bounded. This happens when the magnitude of all eigenvalues of $(\mathbf{I} - \boldsymbol{\gamma}^i \mathbf{G}^i_j)$ are less than 1.

2.2.4 Implementation of the algorithm: Assuming the learning gain $\boldsymbol{\gamma}$ is a diagonal matrix of appropriate size, we note that the learning is stable and converges if all the eigenvalues of $(\mathbf{I} - \boldsymbol{\gamma}^i \mathbf{G}^i_j)$ are less than 1 (see Eqn. 18 and 19). Thus the learning gain, $\boldsymbol{\gamma}^i$, is chosen at each iteration to meet the above condition. A convenient way to do this is as follows: after implementing the control on the system we first evaluate $\mathbf{G}^i_j$ using

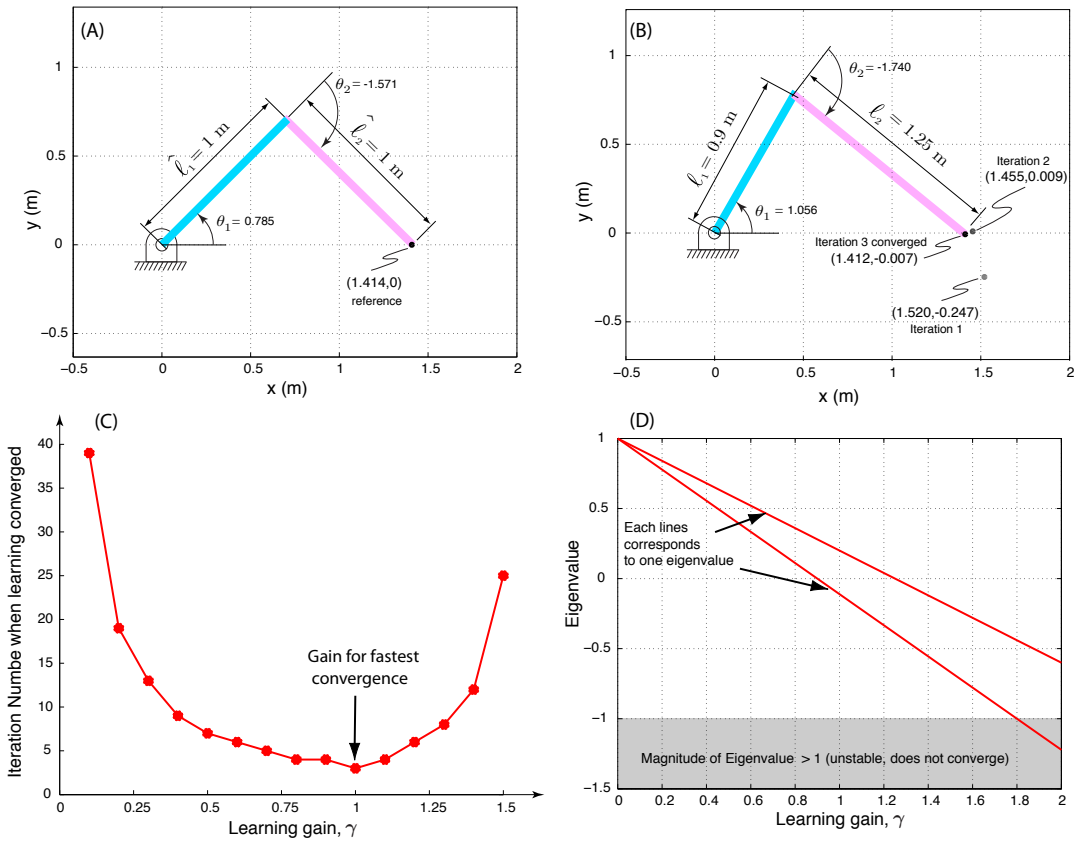

FIGURE 3. (A) OUR INACCURATE MODEL CONSISTS OF TWO LINKS OF LENGTH 1M EACH. (B) HOWEVER, THE ACTUAL SYSTEM HAS LINKS OF LENGTH 0.9 M AND 1.25 M AS SHOWN. OUR GOAL IS TO MAKE THE END-EFFECTOR GO TO THE POSITION $\mathbf{Y}_{\text{REF}} = 1.414$ AND $\mathbf{Y}_{\text{REF}} = 0$. WE USE THE INACCURATE MODEL FOR INVERSE KINEMATICS AND LEARN USING THE ACTUAL SYSTEM. USING A LEARNING GAIN $\gamma = 1$ AND USING THE MODEL IN (A), THE LEARNING CONVERGES IN THREE ITERATIONS. THE ANGLES θ_1 AND θ_2 IN (B) CORRESPOND TO THE CONVERGED TRIAL. (C) NUMBER OF ITERATIONS TO CONVERGENCE VS LEARNING GAIN FOR THE EXAMPLE SHOWN IN (B). (D) THE EIGENVALUES VS LEARNING GAIN. WHEN BOTH EIGENVALUES ARE LESS THAN 1, WE GET A STABLE LEARNING. THE LEARNING IS UNSTABLE IF EITHER EIGENVALUE (SHOWN BY RED LINES) IS GREATER THAN 1.

Eqn. 17, then we choose γ^i based on the maximum value of $\mathbf{G}_j^i$, that is, $\gamma^i = \frac{1}{\max G_j^i}$, and use this value for the ILC update.

3 Example 1: Two link manipulator doing a pointing task

We illustrate the performance of the algorithm with respect to rate of convergence and stability on a two link manipulator doing a pointing task. There are no joint position limits in this problem but the four link manipulator presented later has joint position limits.

3.1 The control problem

Figure 3 (a) shows our model of the manipulator. Link 1 is hinged to the ground while the link 2 is attached to link 1. The two links of the manipulator have length $\hat{\ell}_1 = 1$ m and $\hat{\ell}_2 = 1$ m. Fig. 3 (b) shows the actual manipulator. Note that the two links of the manipulators have different link lengths $\ell_1 = 0.9$ m and $\ell_2 = 1.25$ m. Thus our model which assumes a link length of 1m is incorrect.

We illustrate our learning algorithm with a simple pointing task. The goal is to point the end of link 2 in Fig. 3 (b) to $x_{\text{ref}} = 1.414$ m and $y_{\text{ref}} = 0$. Our goal is to use the nominal model shown in Fig. 3 (a) but learn the desired position for the inverse kinematics from the actual model shown in Fig. 3 (b).

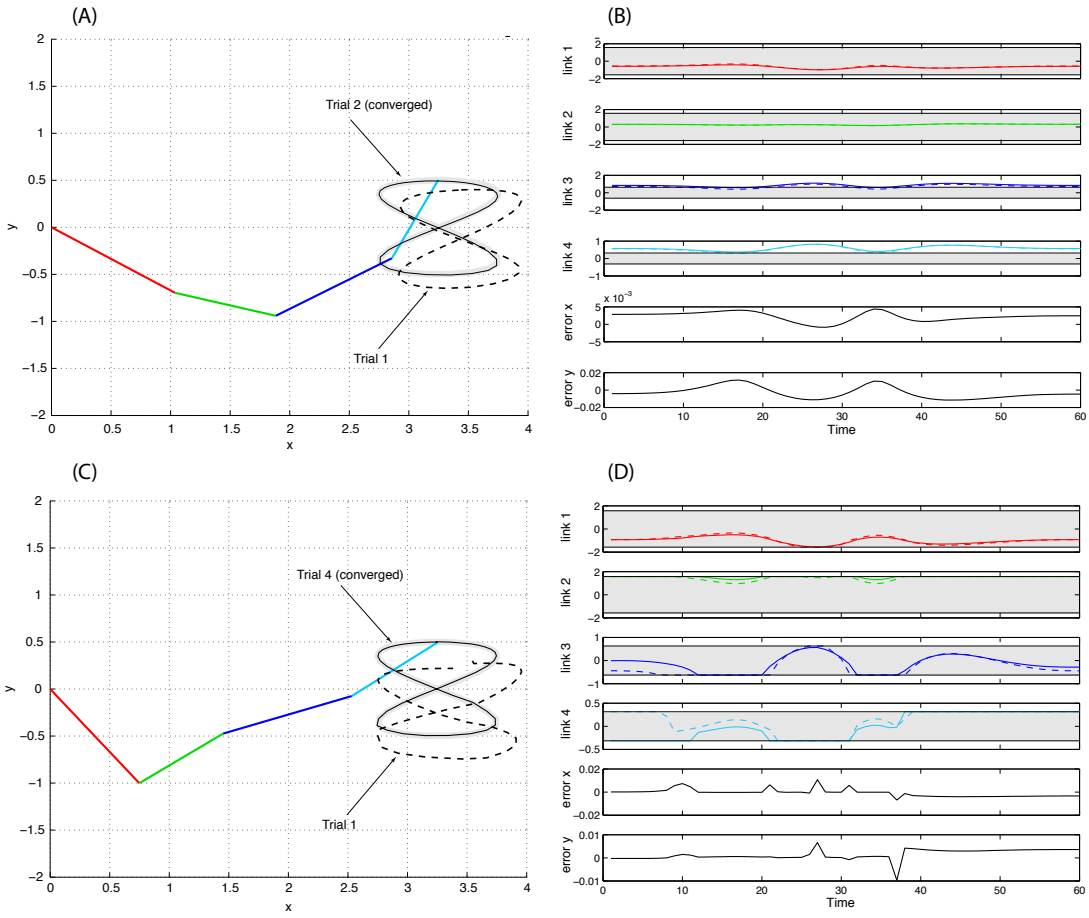

FIGURE 4. TOP ROW: LEARNING WITH JOINT LIMIT NOT ENFORCED. (A) FOUR LINK MANIPULATOR LEARNS MODELING ERROR IN JUST TWO TRIALS WHEN JOINT POSITION LIMITS ARE NOT ENFORCED. (B) JOINT COMMANDS VS .TIME FOR THE FIRST TRIAL (DASHED) AND CONVERGED TRIAL (SOLID LINE). THE GREY BANDS SHOW THE JOINT POSITION LIMITS THAT ARE NOT ENFORCED IN THIS EXAMPLE. BOTTOM ROW: LEARNING WITH JOINT LIMIT ENFORCED. (C) FOUR LINK MANIPULATOR LEARNS MODELING ERROR IN 4 TRIALS WHEN JOINT POSITION LIMITS ARE ENFORCED. (D) JOINT COMMANDS VS. TIME FOR THE FIRST TRIAL (DASHED) AND CONVERGED TRIAL (SOLID LINE). THE GREY BANDS SHOW THE JOINT POSITION LIMITS THAT ARE ENFORCED IN THIS EXAMPLE.

3.2 Forward and inverse kinematics model

The forward kinematics model is

$$x_{\rm des} = \hat{\ell}_1 c_1 + \hat{\ell}_2 c_{12}, \qquad y_{\rm des} = \hat{\ell}_1 s_1 + \hat{\ell}_2 s_{12}. \tag{20}$$

The inverse model can be computed by solving for θ_1 and θ_2 from Eqn. 20.

$$\theta_2 = \cos^{-1}\left(\frac{x_{\rm des}^2 + y_{\rm des}^2 - \hat{\ell}_1^2 - \hat{\ell}_2^2}{2\hat{\ell}_1\hat{\ell}_2}\right), \tag{21}$$

$$\theta_1 = \cos^{-1}\left(\frac{x_{\rm des}(\hat{\ell}_1 + \hat{\ell}_2 c_2) + y_{\rm des}\hat{\ell}_2 s_2}{\hat{\ell}_1^2 - \hat{\ell}_2^2 + 2\hat{\ell}_1\hat{\ell}_2 c_2}\right). \tag{22}$$

Note that we have assumed θ_1 and θ_2 to be positive in the inverse kinematics solution. If we allow negative values as well, the inverse becomes multi-valued. The true system has the same form as above but with lengths given by ℓ_1 and ℓ_2.

TABLE 1. DATA FOR TWO LINK MANIPULATOR DOING POINTING TASK. THE ACTUAL LENGTH IS $\ell_1 = 0.9$ AND $\ell_2 = 1.2$ BUT THE INACCURATE MODEL HAS $\hat{\ell} = 0.9$ AND $\hat{\ell} = 1.2$. WE WANT THE END-EFFECTOR TO END ON THE POSITION $X_{\text{REF}} = 1.414$ AND $Y_{\text{REF}} = 0$. THE LEARNING GAIN, $\gamma = 1$, FOR THIS EXPERIMENT. WE STOP THE LEARNING WHEN THE ERROR NORM (LAST COLUMN) IS LESS THAN 0.01. SEE SECTION 3 FOR DETAILS.

i	x^i_{des}	y^i_{des}	θ^i_1	θ^i_2	x^i	y^i	e^i_{norm}
1	1.414	0.000	0.785	-1.571	1.520	-0.247	0.354
2	1.308	0.247	1.029	-1.685	1.455	0.009	0.050
3	1.268	0.238	1.056	-1.740	1.412	-0.007	0.009

3.3 ILC initialization and evaluation

The model is initialized using the inverse kinematics solution using the approximate model $\hat{f}$. The initial angles from the inverse kinematics solutions above gives, $\theta_1 = \pi/4 = 0.785$ and $\theta_2 = -\pi/2 = -1.571$. This puts the end of link 2 in the position (1.520,-0.247) as shown in Fig. 3 (b) (Iteration 1).

The goal of the iterative learning is to learn the desired position on the approximate model to get to the correct reference position. We show the performance of learning with the gain $\gamma = 1$ in Fig. 3 (b). The learning converges in three trials. Table 1 illustrates the learning process. The desired positions are changed in every trial based on measurement error. In this case, we stopped the learning when the error norm reached a value of 0.01 or lower.

3.4 How learning gain γ affects convergence

The learning gain γ is a design parameter that needs to be chosen. To understand the effect of learning gain on convergence, we did the learning for a gain in the range $(0,2]$. We stop the learning if it takes more than 40 iterations. Fig. 3 (c) shows the iterations for convergence vs learning gain. The convergence is in 3 trials at a gain of 1.

3.5 Stability

Figure 3 (d) shows a plot of the two eigenvalues versus gain. For $0 < \gamma < 1.8$, the biggest eigenvalue is smaller than 1 and the ILC scheme is stable. For $\gamma > 1.8$ the magnitude of the biggest eigenvalue is greater than 1 and ILC scheme becomes unstable. Further, we noticed that the convergence for $0 < \gamma < 0.9$ (both eigenvalues are real and positive) is monotonic and for $0.9 < \gamma < 1.8$ (at least one eigenvalue is real and negative) the convergence was oscillatory.

4 Example 2: Four Link Manipulator drawing the Lissojous figure

We consider a four link manipulator in 2-D space drawing the Lissojous curve. Again, we will have a model with incorrect values for the link lengths. We will consider the case with and without joint position limits to demonstrate how our algorithm works.

4.1 The control problem

Figure 4 (a) and (c) shows our model of the manipulator. Link 1 is hinged to the ground while the link 2 is attached to link 1, and so on. The four links of the manipulator have lengths $\hat{\ell}_1 = 1.25$ m, $\hat{\ell}_2 = 0.88$ m, $\hat{\ell}_3 = 1.15$m, and $\hat{\ell}_4 = 0.92$ m.

However, our model assumes that all links are of equal length $\ell_1 = \ell_2 = \ell_3 = \ell_4 = 1$ m. The joint position limits are as follows:

$$\begin{aligned} -\pi/2 \le \theta_1 \le \pi/2, \quad & -\pi/2 \le \theta_2 \le \pi/2, \\ -\pi/5 \le \theta_3 \le \pi/5, \quad & -\pi/10 \le \theta_4 \le \pi/10 \end{aligned} \tag{23}$$

Our goal is to draw the Lissojous figure [7] which is given by the following equation.

$$\begin{aligned} \alpha &= 2\pi\Big\{6\Big(\frac{j}{60}\Big)^5 - 15\Big(\frac{j}{60}\Big)^4 + 10\Big(\frac{j}{60}\Big)^3\Big\}; 0 \le j \le 60; \\ x_{\text{ref}} &= \sin(2\alpha);\ y_{\text{ref}} = \sin\Big(\alpha + \frac{\pi}{2}\Big); \end{aligned} \tag{24}$$

4.2 Forward and inverse kinematics model

The forward kinematics model is given by

$$\begin{aligned} x_{\text{des}} &= \hat{\ell}_1 c_1 + \hat{\ell}_2 c_{12} + \hat{\ell}_3 c_{123} + \hat{\ell}_4 c_{1234} \\ y_{\text{des}} &= \hat{\ell}_1 s_1 + \hat{\ell}_2 s_{12} + \hat{\ell}_3 s_{123} + \hat{\ell}_4 s_{1234} \end{aligned} \tag{25}$$

The true system has the same form as above but with lengths given by ℓ_1, ℓ_2, ℓ_3, and ℓ_4. The inverse kinematics solution methodology was given in Sec. 1.

4.3 Evaluation of the ILC algorithm

We tried two cases. First, without joint position limits and second with joint position limits as given in Eqn. 23.

Joint position limits not enforced: Figure 4 (a) and (b) shows results without joint position limits enforced. We used a learning gain of $\gamma = 1$ (we discuss this more in the next section). In particular, Fig. 4 (a) shows the performance for trial 1 and converged trial 2. Figure 4 (b) shows the joint position for the first and converged trial. The grey area shows the joint position limits for reference but not enforced here.

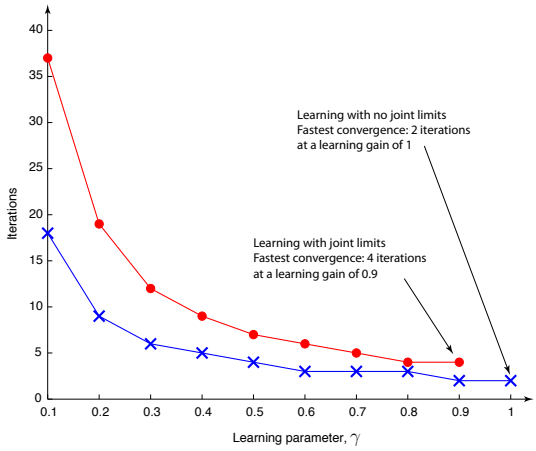

FIGURE 5. RATE OF LEARNING WITH (RED DOTS) AND WITHOUT (BLUE CROSSES) JOINT POSITION LIMITS FOR THE FOUR LINK MANIPULATOR. THE SOLUTION DIVERGES FOR $\gamma > 1$.

Joint position limits enforced: Figure 4 (c) and (d) shows results with joint position limits enforced. We used a learning gain of $\gamma = 0.9$ (we discuss this more in the next section). In particular, Fig. 4 (a) shows the performance for trial 1 and converged trial 4. Figure 4 (c) shows the joint position for the first and converged trial, trial 4. The grey area shows the joint position limits for reference. The optimization ensures that the joint position limits are enforced. We stopped the algorithm when the error was below 0.05. We also tried Arimoto's algorithm (discussed in Sec. 1) but it was not able to converge for any γ in the range $(0,2]$. Note that the final converged error in x- and y- position show kinks at the instance when the joint limits are hit.

4.4 How learning gain γ affects convergence

Figure 5 shows the convergence as a function of learning gain γ with and without joint position limits enforced. The fastest convergence for case with no joint position limits is for $\gamma = 1$ and convergence is in 2 trials. For the case with enforced joint position limits, the fastest convergence is at $\gamma = 0.9$ and convergence is in 4 trials.

5 Conclusion and future work

We presented an iterative learning control algorithm for accurate tracking in task space for redundant manipulators. The learning algorithm adjusts the desired end-effector position in an iterative fashion. The new desired position is used to update joint position using a non-linear inverse that respects joint position limits. We show the efficacy of the algorithm on a four-link manipulator with stringent joint position limits drawing the figure eight.

Future work will explore the iterative learning algorithm for dynamic (fast) motion for redundant manipulators.

REFERENCES

[1] Baerlocher, P., and Boulic, R., 2004. "An inverse kinematics architecture enforcing an arbitrary number of strict priority levels". *The visual computer,* ***20***(6), pp. 402–417.

[2] Wang, L.-C. T., and Chen, C. C., 1991. "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators". *Robotics and Automation, IEEE Transactions on,* ***7***(4), pp. 489–499.

[3] Tayebi, A., 2004. "Adaptive iterative learning control for robot manipulators". *Automatica,* ***40***(7), pp. 1195–1203.

[4] Bristow, D., Tharayil, M., Alleyne, A. G., et al., 2006. "A survey of iterative learning control". *Control Systems, IEEE,* ***26***(3), pp. 96–114.

[5] Guglielmo, K., and Sadegh, N., 1996. "Theory and implementation of a repetitive robot controller with cartesian trajectory description". *Journal of Dynamic Systems, Measurement, and Control,* ***118***(1), pp. 15–21.

[6] Kuc, T.-y., Nam, K., and Lee, J. S., 1991. "An iterative learning control of robot manipulators". *Robotics and Automation, IEEE Transactions on,* ***7***(6), pp. 835–842.

[7] Arimoto, S., Sekimoto, M., and Kawamura, S., 2007. "Iterative learning of specified motions in task-space for redundant multi-joint hand-arm robots". In Robotics and Automation, 2007 IEEE International Conference on, IEEE, pp. 2867–2873.

[8] Gill, P., Murray, W., and Saunders, M., 2002. "SNOPT: An SQP algorithm for large-scale constrained optimization". *SIAM Journal on Optimization,* ***12***(4), pp. 979–1006.

[9] Owens, D. H., 2016. "Norm optimal iterative learning control". In *Iterative Learning Control*. Springer, pp. 233–276.

[10] Bhounsule, P., and Yamane, K., 2015. "Iterative learning of inverse kinematics with applications to humanoid robots". In International Conference on Humanoid Robots, Seoul, South Korea.