# Application of a Recursive Minimum-Norm Learning Controller to Precision Motion Control of an Underactuated Mechanical System

Ai-Ping Hu
Dept. of Mechanical & Industrial Engineering
Southern Illinois University at Edwardsville
Edwardsville, IL 62026
ahu@siue.edu

Nader Sadegh
Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332
nader.sadegh@me.gatech.edu

***Abstract*— A recursive method is first presented for obtaining the minimum-norm solution of a linear system of equations. The formulation is then extended to the case of a nonlinear system of equations, the result of which is used as the basis of a repetitive learning controller that is applied experimentally to precision motion control of an underactuated mechanical system tracking a periodic trajectory.**

## I. Introduction

This paper describes the development of a repetitive learning controller based on a recursive formulation of the minimum-norm solution of a system of equations. The recursive method is first developed for a linear system of equations and then extended to the nonlinear case, which is used as the basis of the repetitive learning controller. The repetitive learning controller is applied to motion control of a testbed system consisting of a cart and down-hanging pendulum (i.e., a mechanical system resembling a crane) tracking a periodic trajectory.

## II. Recursive Minimum-Norm Solution of a Linear System of Equations

Consider the following system of linear equations:

$$\boldsymbol{y_k} = \boldsymbol{\Phi_k X}, \text{ for } k = 1, 2, \ldots, N, \tag{1}$$

where $\boldsymbol{y_k} \in \mathbb{R}^n$ and $\boldsymbol{\Phi_k} \in \mathbb{R}^{n\times m}$ are known quantities and $\boldsymbol{X} \in \mathbb{R}^m$ is the unknown vector being solved for. It is assumed that the matrix

$$\boldsymbol{A_k} = \begin{bmatrix} \boldsymbol{\Phi_1} \\ \boldsymbol{\Phi_2} \\ \vdots \\ \boldsymbol{\Phi_k} \end{bmatrix} \tag{2}$$

is full rank for each $k$ and that $m = nN$. It is well-known (see, e.g., [1]) that the minimum-norm solution of (1) at step $k$ is given by $\hat{\boldsymbol{X}}_{\boldsymbol{k}} = \boldsymbol{A_k^+ Y_k}$, where the so-called *pseudo-inverse* $\boldsymbol{A_k^+} = \boldsymbol{A_k^T}(\boldsymbol{A_k A_k^T})^{-1}$ and $\boldsymbol{Y_k} = [\boldsymbol{y_1}; \boldsymbol{y_2}; \cdots; \boldsymbol{y_k}]$.

To recursively compute $\hat{\boldsymbol{X}}_{\boldsymbol{k}} = \boldsymbol{A_k^+ Y_k}$, the following recursive formula for computing $\boldsymbol{A_k^+}$ is applied:

$$\boldsymbol{A_k^+} = \begin{bmatrix} \boldsymbol{A_{k-1}^+} - \boldsymbol{P_{k-1}\Phi_k^T D_{k-1}^{-1}\Phi_k A_{k-1}^+} & \boldsymbol{P_{k-1}\Phi_k^T D_{k-1}^{-1}} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{P_k} = \boldsymbol{I} - \boldsymbol{A_k^+ A_k}$ and $\boldsymbol{D_{k-1}} = \boldsymbol{\Phi_k P_{k-1} \Phi_k^T}$. $\boldsymbol{I}$ is the $k \times k$ identity matrix. Expressing $\boldsymbol{Y_k}$ as $[\boldsymbol{Y_{k-1}};\ \boldsymbol{y_k}]$, the following is then obtained:

$$\hat{\boldsymbol{X}}_{\boldsymbol{k}} = \hat{\boldsymbol{X}}_{\boldsymbol{k-1}} + \boldsymbol{P_{k-1}\Phi_k^T D_{k-1}^{-1}} \left( \boldsymbol{y_k} - \boldsymbol{\Phi_k} \hat{\boldsymbol{X}}_{\boldsymbol{k-1}} \right). \tag{4}$$

Using the recursive expression for $\boldsymbol{A_k^+}$ from (3) in $\boldsymbol{P_k} = \boldsymbol{I} - \boldsymbol{A_k^+}\,[\boldsymbol{A_{k-1}};\ \boldsymbol{\Phi_k}]$ yields

$$\boldsymbol{P_k} = \boldsymbol{P_{k-1}} - \boldsymbol{P_{k-1}\Phi_k^T D_{k-1}^{-1}\Phi_k P_{k-1}}, \tag{5}$$

with $\boldsymbol{P_0} = \boldsymbol{I}$. Observe that the above formulation renders the unique solution $\boldsymbol{X^*} \triangleq \boldsymbol{A_N^{-1} Y_N}$ at $k = N$. In situations where there is uncertainty about the process, it is desirable to use a priori information about $\boldsymbol{X}$ and update $\hat{\boldsymbol{X}}_{\boldsymbol{k}}$ at a slower rate. That is,

$$\hat{\boldsymbol{X}}_{\boldsymbol{k}} = \gamma \boldsymbol{A_k^+ Y_k} + (1-\gamma)\hat{\boldsymbol{X}}_{\boldsymbol{0}}, \tag{6}$$

where $\hat{\boldsymbol{X}}_{\boldsymbol{0}}$ is the initial estimate of $\boldsymbol{X}$ and $0 < \gamma \leq 1$ is a constant scalar, referred to as the update gain of $\hat{\boldsymbol{X}}_{\boldsymbol{k}}$. Defining the estimation error at step $k$ as

$$\boldsymbol{e_k} \triangleq \boldsymbol{y_k} - \boldsymbol{\Phi_k} \hat{\boldsymbol{X}}_{\boldsymbol{k-1}}, \tag{7}$$

it then follows that at $k = N$,

$$\boldsymbol{e_N} = (1-\gamma)\boldsymbol{e_0}. \tag{8}$$

The recursive version of $\hat{\boldsymbol{X}}_{\boldsymbol{k}}$ in (6) is given by:

$$\hat{\boldsymbol{X}}_{\boldsymbol{k}} = \hat{\boldsymbol{X}}_{\boldsymbol{k-1}} + \boldsymbol{P_{k-1}\Phi_k^T D_{k-1}^{-1}} \bar{\boldsymbol{e}}_{\boldsymbol{k}} \tag{9}$$

$$\bar{\boldsymbol{e}}_{\boldsymbol{k}} = \gamma \boldsymbol{e_k} + (1-\gamma)\boldsymbol{\Phi_k} \left( \hat{\boldsymbol{X}}_{\boldsymbol{0}} - \hat{\boldsymbol{X}}_{\boldsymbol{k-1}} \right). \tag{10}$$

The above formulation may be extended to $k$ beyond $N$ in order to drive the error $\boldsymbol{e_k}$ asymptotically to zero.

0-7803-8335-4/04/$17.00 ©2004 AACC

## III. Extension to Nonlinear Systems of Equations

Now consider the following system of nonlinear equations:

$$\boldsymbol{y_k} = \boldsymbol{f_k}(\boldsymbol{X}), \text{ for } k = 1, 2, \ldots, N, \tag{11}$$

where $\boldsymbol{y_k} \in \mathbb{R}^n$ and $\boldsymbol{f_k}\colon \mathbb{R}^m \to \mathbb{R}^n$ are known quantities, $\boldsymbol{X} \in \mathbb{R}^m$ is the unknown vector, and $m = nN$. The system of equations in (11) may be expressed as $\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{X})$, with $\boldsymbol{Y} = [\boldsymbol{y_1};\ \boldsymbol{y_2};\ \cdots;\ \boldsymbol{y_N}]$ and $\boldsymbol{F}(\boldsymbol{X}) = [\boldsymbol{f_1}(\boldsymbol{X});\ \boldsymbol{f_2}(\boldsymbol{X});\ \cdots;\ \boldsymbol{f_N}(\boldsymbol{X})]$. Similar to the linear case, the following equations may then be used to iteratively solve for the unknown vector $\boldsymbol{X}$ that satisfies the system of nonlinear equations (11):

$$\hat{\boldsymbol{X}}_{\boldsymbol{k+1}} = \hat{\boldsymbol{X}}_{\boldsymbol{k}} + \gamma \boldsymbol{A_N^{-1}}(\hat{\boldsymbol{X}}_{\boldsymbol{k}})\boldsymbol{E_k} \tag{12}$$

$$\boldsymbol{E_k} = \boldsymbol{Y} - \boldsymbol{F}(\hat{\boldsymbol{X}}_{\boldsymbol{k}}), \tag{13}$$

where

$$\boldsymbol{A_N} = \begin{bmatrix} \boldsymbol{\partial f_1(z)/\partial z}\big|_{\boldsymbol{z}=\hat{X}_k} \\ \boldsymbol{\partial f_2(z)/\partial z}\big|_{\boldsymbol{z}=\hat{X}_k} \\ \vdots \\ \boldsymbol{\partial f_N(z)/\partial z}\big|_{\boldsymbol{z}=\hat{X}_k} \end{bmatrix} \tag{14}$$

is the matrix of Jacobians. It is proved as **Theorem 1** in [2] that there exists $0 < \bar{\gamma} \leq 1$ such that if $\gamma < \bar{\gamma}$ then $\|\boldsymbol{E_k}\|$ converges to zero exponentially. ($\|\ \|$ denotes the 2-norm.)

A recursive minimum-norm algorithm based on a modified version of the equations (12) and (13) forms the basis of the repetitive learning controller implemented in this paper.

## IV. Repetitive Learning Controller

The nominal form of the discrete-time repetitive learning controller developed in this section will match that of equations (12) and (13). This particular form is governed by the fact that the reference trajectory to be tracked, $y_d(t)$, is (restricted to be) periodic in time, $t$. We designate the period of $y_d(t)$ by $\tau$, which is assumed to be an integer multiple of the discrete-time sample period, $T$, i.e., $\tau = NT$, where $N$ is an integer. Let the output of the learning controller at sample index $k$ be denoted by $r(kT)$. The complete control input that $r(kT)$ is part of is developed fully in [2], which addresses tracking control of $n$th-order SISO nonlinear, non-minimum phase dynamical systems of the form: $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\,u$, with output defined to be the scalar function $y = h(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of state variables, $\dot{}$ denotes differentiation with respect to time, $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^n$ and $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^n$ are smooth vector fields, and $u \in \mathbb{R}$ is the control input. The controller sought to achieve exponential output tracking of $y_d(t)$ in a neighborhood of the region $\mathcal{C}$, a connected subset of the set of equilibrium states defined by $\mathcal{C}_0 \triangleq \{\boldsymbol{x_0}\colon \exists\ u_0 \text{ such that } \boldsymbol{f}(\boldsymbol{x_0}) + \boldsymbol{g}(\boldsymbol{x_0})u_0 = \boldsymbol{0}\}$. The subset $\mathcal{C}$ is comprised of states that satisfy a linear controllability assumption. (Refer to [2] for details.) The primary purpose of the present paper is to describe how the learning control update law results in a recursive minimum-norm solution, and then to apply it to an experimental testbed.

We note here, for use below, that it was shown in [2] that $r(kT)$ may be expressed as a linear combination:

$$r(kT) = \bar{r}(kT) + a_1\bar{r}((k-1)T) + \ldots + a_n\bar{r}((k-n)T). \tag{15}$$

In addition, it is also shown in [2] that the nonlinear dynamical system's output, denoted $y(kT)$ in discrete-time, may be expressed as

$$y(kT) = h(\bar{r}(kT), \bar{r}((k-1)T), \ldots, \bar{r}((k-n)T)). \tag{16}$$

The action of the learning controller may be described as progressively refining, through practice, the value of $r(kT)$ at a given point in discrete-time with the goal of achieving perfect tracking of $y_d(t)$ by the output $y(kT)$ of the nonlinear dynamical system. Having the desired reference trajectory be periodic means that the system being controlled is mandated to repeatedly perform the same task. The learning controller takes advantage of this repetition by essentially trying to solve the same problem over and over again, with its present iteration an improvement upon past attempts (this is what is meant here by "practice").

The periodicity of the reference trajectory naturally facilitates the use of a *circular buffer*. This circular buffer will be an array of length $N$ that contains the values of $\bar{r}((k-j)T)$. The first element of the circular buffer (to which we assign the index 0) will correspond to the first discretized point of (one period of) the desired trajectory and the last element of the circular buffer (with index $N$–1) will correspond to the final discretized point of one period of the desired trajectory. $\bar{r}((k-j)T)$ is stored as the element in the circular buffer with index ($(k-j)$ modulus $N$). In other words, the elements of the circular buffer, which we denote as $\bar{\boldsymbol{R}} = \begin{bmatrix}\bar{R}_0 & \bar{R}_1 & \cdots & \bar{R}_{N-1}\end{bmatrix}^T$, are (over-)written to sequentially and, when the end of the circular buffer is reached (index $N-1$, corresponding to the end of a period), the index is re-set to 0, signifying the start of a new period. $\bar{\boldsymbol{R}}$ is referred to as the *learning array*. At any given discrete-time $kT$, the learning control output $r(kT)$ will be comprised of a linear combination of $n$+1 elements of the learning array, where we assume that $N > n$. The expression $y(kT) = h(\bar{r}(kT), \bar{r}((k-1)T), \ldots, \bar{r}((k-n)T))$ may thus be equivalently written as $y(kT) = h(\bar{\boldsymbol{R}})$.

A given period of the desired trajectory is to be divided into $N$ segments, each of which is associated with a distinct left-hand endpoint. We designate the vector holding the values of $y_d(t)$ at these endpoints as $\boldsymbol{Y_d} \triangleq \begin{bmatrix}y_d(0) & y_d(T) \cdots y_d((N-1)T)\end{bmatrix}^T$. We will be interested in keeping track of what is taking place at particular indices $0, 1, \ldots, N$–1 within a given period. We thus introduce two new indices, $i$ and $j$, which will be used as a double index appearing as subscripts separated by a comma. $j$ will take on integer values from 0 to $N-1$, and serves to denote at which time step we are at within period number $i$. For example, the output at the $j$th step within period number $i$

is denoted by $h_{i,j}(\bar{\boldsymbol{R}})$. The relationship among $i$, $j$, and our discrete-time index $k$ is given by $k = (i-1)N + j$, where we note that the period number count starts from $i = 1$.

With the above bookkeeping notation, the objective of our learning law may be explicitly stated as: iteratively refining the learning array $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$ (i.e., the learning array $\bar{\boldsymbol{R}}$ corresponding to period number $i$) such that $\boldsymbol{F}(\bar{\boldsymbol{R}}_{\boldsymbol{i}}) \triangleq \boldsymbol{Y_d} - \left[h_{i,0}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\ h_{i,1}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\ \cdots\ h_{i,N-1}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\right]^T \rightarrow \boldsymbol{0}$ as the period number $i \rightarrow \infty$. For later use, we define $\boldsymbol{H}(\bar{\boldsymbol{R}}_{\boldsymbol{i}}) \triangleq \left[h_{i,0}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\ h_{i,1}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\ \cdots\ h_{i,N-1}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\right]^T$.

Determining the correct $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$ such that $\boldsymbol{F}(\bar{\boldsymbol{R}}_{\boldsymbol{i}}) = \boldsymbol{0}$ constitutes a root-finding problem for a system of nonlinear equations. The *Newton-Raphson root-finding method for systems of nonlinear equations* (see, e.g., Press et al. [4]) is an algorithm that seeks to determine the required $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$ iteratively. The learning law will be a modified version of this particular algorithm.

The Newton-Raphson root-finding method for systems of nonlinear equations may be correctly thought of as an extension into higher-dimensions of the familiar Newton-Raphson root-finding method for a nonlinear function of a single variable; its development proceeds in a way that is completely analogous to the scalar case. We start with the Taylor's series expansion of $\boldsymbol{F}(\bar{\boldsymbol{R}}) \in \mathbb{R}^N$ in a neighborhood of (i.e., a small deviation $\boldsymbol{\delta\bar{R}_i}$ from) $\bar{\boldsymbol{R}}_{\boldsymbol{i}} \in \mathbb{R}^N$, as given by $\boldsymbol{F}(\bar{\boldsymbol{R}}_{\boldsymbol{i}} + \boldsymbol{\delta\bar{R}_i}) = \boldsymbol{F}(\bar{\boldsymbol{R}}_{\boldsymbol{i}}) + \boldsymbol{A}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\,\boldsymbol{\delta\bar{R}_i} + \sum_{j=1}^{N} \boldsymbol{\delta\bar{R}_i}^T \boldsymbol{M_j}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\,\boldsymbol{\delta\bar{R}_i} + \ldots$, where $\boldsymbol{A}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})$ represents the Jacobian of $\boldsymbol{F}(\bar{\boldsymbol{R}})$ with respect to $\bar{\boldsymbol{R}}$, evaluated at $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$, and $\boldsymbol{M_j}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})$ represents the Hessian of the $j$th component of $\boldsymbol{F}(\bar{\boldsymbol{R}})$ with respect to $\bar{\boldsymbol{R}}$, evaluated at $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$ (it is assumed that the components of $\boldsymbol{F}(\bar{\boldsymbol{R}})$ are at least twice differentiable with respect to $\bar{\boldsymbol{R}}$). Upon neglecting the terms nonlinear in $\boldsymbol{\delta\bar{R}_i}$, the following iterative formula (starting with an initial guess $\bar{\boldsymbol{R}}_{\boldsymbol{0}}$) yields gradually improved estimates of the root of $\boldsymbol{F}(\bar{\boldsymbol{R}}) = \boldsymbol{0}$:

$$\begin{aligned} \bar{\boldsymbol{R}}_{\boldsymbol{i+1}} &= \bar{\boldsymbol{R}}_{\boldsymbol{i}} + \boldsymbol{\delta\bar{R}_i} && (17)\\ &= \bar{\boldsymbol{R}}_{\boldsymbol{i}} - \boldsymbol{A}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})^{-1}\,\boldsymbol{F}(\bar{\boldsymbol{R}}_{\boldsymbol{i}}), && (18)\end{aligned}$$

where the various Jacobian matrices $\boldsymbol{A}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})$ are assumed to be non-singular.

The learning law is based on the following modification of the Newton-Raphson method of equation (18):

$$\begin{aligned} \bar{\boldsymbol{R}}_{\boldsymbol{i+1}} &= \bar{\boldsymbol{R}}_{\boldsymbol{i}} + \gamma\boldsymbol{\delta\bar{R}_i} && (19)\\ &= \bar{\boldsymbol{R}}_{\boldsymbol{i}} - \gamma\boldsymbol{A}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})^{-1}\,\boldsymbol{F}(\bar{\boldsymbol{R}}_{\boldsymbol{i}}) && (20)\\ &= \bar{\boldsymbol{R}}_{\boldsymbol{i}} + \gamma\boldsymbol{C}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})^{-1} \cdot \left(\boldsymbol{Y_d} - \boldsymbol{H}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})\right), && (21)\end{aligned}$$

where $0 < \gamma \leq 1$ is a constant we will refer to as the *learning gain* and $\boldsymbol{C}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})$ is defined to be the Jacobian of $\boldsymbol{H}(\bar{\boldsymbol{R}})$ with respect to $\bar{\boldsymbol{R}}$, evaluated at $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$ (this replacement accounts for the change in sign in equation (21) from (20), since $\boldsymbol{C}(\bar{\boldsymbol{R}}_{\boldsymbol{i}}) = -\boldsymbol{A}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})$). From (21), we point out that the current learning array is determined by its value one full period ago, corrected by a term based on the tracking error incurred over the period *just* completed. For $\gamma = 1$ we simply have the Newton-Raphson method. For values of $\gamma$ less than 1, however, the effect is that $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$ is updated more slowly.

The learning controller works by gradually updating the learning array, $\bar{\boldsymbol{R}}_{\boldsymbol{i}}$, via the application of a correction term $\boldsymbol{\delta\bar{R}_i}$ (or, more precisely: $\boldsymbol{\delta\bar{R}_{i,N-1}}$). Equation (21) is of the exact same form as the recursive minimum-norm solution of equations (12) and (13). Observe that in (12) and (13), the "learning array" is updated only once per "period." In the context of the recursive algorithm, this update takes place after $N$ iterative steps. For each of the $N-1$ interim steps, corresponding to $j = 0, 1, \ldots, N-2$, a minimum-norm "best fit" correction term is computed, $\boldsymbol{\delta\bar{R}_{i,j}}$. Each of these terms is temporarily stored in memory, to be expressly used in the very next iteration step. We next propose to modify the learning law by making use of $\boldsymbol{\delta\bar{R}_{i,j}}$, for $j = 0, 1, \ldots, N-2$, by utilizing these intermediate correction terms to update the learning array at *every* iteration step. (And, of course, $\boldsymbol{\delta\bar{R}_{i,N-1}}$ is still used as well, in what will be for the final update per period.) The reasoning behind the change is that by updating the learning array more frequently (and in a proper manner), the learning controller may be made more responsive.

Since we will now have a learning array associated with every discrete-time step $k$, we invoke our double subscript indexing notation for use with the learning array: at time $kT$, let the learning array be denoted by $\bar{\boldsymbol{R}}_{\boldsymbol{i,j}}$. Our proposed modified learning law may then be readily expressed as

$$\bar{\boldsymbol{R}}_{\boldsymbol{i,j}} = \bar{\boldsymbol{R}}_{\boldsymbol{i,-1}} + \gamma\,\boldsymbol{C_j^+}(\hat{R}_{i,j-1})\left(\boldsymbol{Y_{d_j}} - \boldsymbol{H_j}(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}})\right), \tag{22}$$

where $i = 1, 2, 3, \ldots$ and $j = 0, 1, \ldots, N-1$. The learning array is initialized to be $\bar{\boldsymbol{R}}_{\boldsymbol{1,-1}} = \boldsymbol{0}$ and, for $i \geq 2$, we adopt the notational convention that $\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}} = \bar{\boldsymbol{R}}_{\boldsymbol{i-1,N-1}}$. We use $\hat{R}_{i,j-1}$ to denote the set of learning array vectors $\left\{\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}, \bar{\boldsymbol{R}}_{\boldsymbol{i,0}}, \ldots, \bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}\right\}$. The pseudo-inverse $\boldsymbol{C_j^+}(\hat{R}_{i,j-1})$ is then defined as

$$\boldsymbol{C_j^+}(\hat{R}_{i,j-1}) \triangleq \boldsymbol{C_j^T}(\hat{R}_{i,j-1})\left(\boldsymbol{C_j}(\hat{R}_{i,j-1})\,\boldsymbol{C_j^T}(\hat{R}_{i,j-1})\right)^{-1}, \tag{23}$$

where $\boldsymbol{C_j}(\hat{R}_{i,j-1})$ is the $(j+1) \times N$ matrix, which is assumed to be full rank, given by

$$\boldsymbol{C_j}(\hat{R}_{i,j-1}) = \begin{bmatrix} \left.\dfrac{\partial h_{i,0}(\bar{\boldsymbol{R}})}{\partial \bar{\boldsymbol{R}}}\right|_{\bar{\boldsymbol{R}} = \bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}} \\ \left.\dfrac{\partial h_{i,1}(\bar{\boldsymbol{R}})}{\partial \bar{\boldsymbol{R}}}\right|_{\bar{\boldsymbol{R}} = \bar{\boldsymbol{R}}_{\boldsymbol{i,0}}} \\ \vdots \\ \left.\dfrac{\partial h_{i,j}(\bar{\boldsymbol{R}})}{\partial \bar{\boldsymbol{R}}}\right|_{\bar{\boldsymbol{R}} = \bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}} \end{bmatrix}, \tag{24}$$

i.e., $\boldsymbol{C_j}(\hat{R}_{i,j-1})$ consists of the first $j$+1 rows of a Jacobian matrix whose rows are each evaluated at different points

(i.e., evaluated at different learning arrays). Define

$$\begin{aligned}
\boldsymbol{\Phi_{i,0}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}) &\triangleq \left.\frac{\partial h_{i,0}(\bar{\boldsymbol{R}})}{\partial \bar{\boldsymbol{R}}}\right|_{\bar{\boldsymbol{R}}=\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}},\\
\boldsymbol{\Phi_{i,1}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,0}}) &\triangleq \left.\frac{\partial h_{i,1}(\bar{\boldsymbol{R}})}{\partial \bar{\boldsymbol{R}}}\right|_{\bar{\boldsymbol{R}}=\bar{\boldsymbol{R}}_{\boldsymbol{i,0}}},\\
&\vdots\\
\boldsymbol{\Phi_{i,N-2}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,N-3}}) &\triangleq \left.\frac{\partial h_{i,N-2}(\bar{\boldsymbol{R}})}{\partial \bar{\boldsymbol{R}}}\right|_{\bar{\boldsymbol{R}}=\bar{\boldsymbol{R}}_{\boldsymbol{i,N-3}}}, \text{ and}\\
\boldsymbol{\Phi_{i,N-1}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,N-2}}) &\triangleq \left.\frac{\partial h_{i,N-1}(\bar{\boldsymbol{R}})}{\partial \bar{\boldsymbol{R}}}\right|_{\bar{\boldsymbol{R}}=\bar{\boldsymbol{R}}_{\boldsymbol{i,N-2}}}
\end{aligned} \tag{25}$$

to be these rows.

The new learning law in equation (22) may be compared to the former one in equations (12) and (13). The difference between the two is that (12) and (13) proceed by "batch" updates of the learning array while (22) performs "step-by-step" updates. However, observe that for both, each element in the vector and partial vector $\boldsymbol{H}(\bar{\boldsymbol{R}}_{\boldsymbol{i}})$ and $\boldsymbol{H_j}(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}})$, respectively, is to be evaluated at the learning array determined at the end of the previous period. One might expect, in order to be as current as possible, that we would instead specify that the (partial) vector of output approximations for the step-by-step learning law be evaluated at the set $\hat{R}_{i,j-1}$, i.e., thereby yielding $\boldsymbol{H_j}(\hat{R}_{i,j-1})$. The reason for our choice will become clear once the recursive algorithm for (22) is formulated in the next section. It is noted here that use of the learning law (22) will lead to $\left\|\boldsymbol{Y_d} - \boldsymbol{H}(\bar{\boldsymbol{R}}_{\boldsymbol{i,N-1}})\right\|$ converging to zero exponentially.

*Theorem 1:* Consider the modified learning law given by equation (22) for solving the system of nonlinear equations $\boldsymbol{Y_d} = \boldsymbol{H}(\bar{\boldsymbol{R}})$. Assume that for $j = 0, 1, \ldots, N-1$ the matrix $\boldsymbol{C_j}(\hat{R}_{i,j-1})$ defined by equation (24) is full rank for $\hat{R}_{i,j-1} \subset \mathcal{R}$, which means that the corresponding pseudo-inverse $\boldsymbol{C_j^+}(\hat{R}_{i,j-1})$ (defined in equation (23)) exists for $\hat{R}_{i,j-1} \subset \mathcal{R}$, where $\mathcal{R}$ is some region in $\mathbb{R}^N$ that corresponds to permissable states of the considered nonlinear dynamical system, i.e., $\hat{R}_{i,j-1}$ is such that $\boldsymbol{x}$ is maintained to lie within a close neighborhood of the equilibrium manifold $\mathcal{C}$. In addition, assume the following is satisfied, for $j = 0, 1, \ldots, N-1$:

$$\sup_{\hat{R}_{i,j-1}\subset\mathcal{R}} \left\|\boldsymbol{C_j^+}(\hat{R}_{i,j-1})\right\| = \sigma_1, \tag{26}$$

where $0 < \sigma_1 < \infty$. Furthermore, there exists $0 < \sigma_2 < \infty$ such that the Hessian of the $(j+1)$st component of $\boldsymbol{H}(\bar{\boldsymbol{R}})$ with respect to $\bar{\boldsymbol{R}}$ evaluated at a given $\bar{\boldsymbol{R}} = \boldsymbol{\xi}$, denoted $\boldsymbol{K_j}(\boldsymbol{\xi})$, satisfies the following conditions, for $j = 0, 1, \ldots, N-1$:

$$\sup_{\boldsymbol{\xi_{ja}}\in B_\ell(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}})} \|\boldsymbol{K_j}(\boldsymbol{\xi_{ja}})\| = \sigma_2 \tag{27}$$

and

$$\sup_{\boldsymbol{\xi_{jb}}\in B_\ell(\bar{\boldsymbol{R}}_{\boldsymbol{i,N-1}})} \|\boldsymbol{K_j}(\boldsymbol{\xi_{jb}})\| = \sigma_2, \tag{28}$$

where, e.g., $B_\ell(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}})$ denotes an open "ball" of radius $\ell > 0$ centered at $\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}$. Then, for any initial error $\boldsymbol{Y_d} - \boldsymbol{H}(\bar{\boldsymbol{R}}_{\boldsymbol{1,-1}})$, there exists $0 < \bar{\gamma} \leq 1$ such that if $\gamma < \bar{\gamma}$ then $\left\|\boldsymbol{Y_d} - \boldsymbol{H}(\bar{\boldsymbol{R}}_{\boldsymbol{i,N-1}})\right\|$ converges to zero exponentially.

Proof Refer to Hu [3].

## V. Recursive Algorithm for Repetitive Learning Control Update Law

The formulation of the recursive algorithm for the step-by-step update learning law of equation (22) is able to make expedient use of the development in section 2 for the linear case. There, to solve for

$$\hat{\boldsymbol{X}}_{\boldsymbol{k}} = \boldsymbol{A_k^+}\boldsymbol{Y_k}, \tag{29}$$

the recursive equations (9) and (10) were determined. The learning law (22) may be re-written such that it is identical in form to equation (29): upon defining

$$\boldsymbol{\Delta\bar{R}_{i,j}} \triangleq \bar{\boldsymbol{R}}_{\boldsymbol{i,j}} - \bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}, \tag{30}$$

equation (22) may be equivalently expressed as

$$\frac{\boldsymbol{\Delta\bar{R}_{i,j}}}{\gamma} = \boldsymbol{C_j^+}(\hat{R}_{i,j-1})\left(\boldsymbol{Y_{d_j}} - \boldsymbol{H_j}(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}})\right). \tag{31}$$

Comparing (29) and (31), we see that $\hat{\boldsymbol{X}}_{\boldsymbol{k}}$, $\boldsymbol{A_k^+}$, and $\boldsymbol{Y_k}$ are analogous to, respectively, $\boldsymbol{\Delta\bar{R}_{i,j}}/\gamma$, $\boldsymbol{C_j^+}(\hat{R}_{i,j-1})$, and $\left(\boldsymbol{Y_{d_j}} - \boldsymbol{H_j}(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}})\right)$. Accordingly, then, the analogues of recursive equations (9) and (10) are

$$\begin{aligned}
\boldsymbol{\Delta\bar{R}_{i,j}} &= \boldsymbol{\Delta\bar{R}_{i,j-1}} + \boldsymbol{P_{i,j-1}}(\hat{R}_{i,j-2})\,\boldsymbol{\Phi_{i,j}^T}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}) \times\\
&\quad \boldsymbol{D_{i,j-1}^{-1}}(\hat{R}_{i,j-1}) \cdot (\gamma\,\bar{\varepsilon}_{i,j})
\end{aligned} \tag{32}$$

and

$$\begin{aligned}
\gamma\,\bar{\varepsilon}_{i,j} &= \gamma\,(y_d(jt) - \hat{y}(kT)) +\\
&\quad (\gamma - 1)\,\boldsymbol{\Phi_{i,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}})\,\boldsymbol{\Delta\bar{R}_{i,j-1}},
\end{aligned} \tag{33}$$

where $\hat{y}(kT) \triangleq h_{i,j}(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}) + \boldsymbol{\Phi_{i,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}})\,\boldsymbol{\Delta\bar{R}_{i,j-1}}$. The term $\hat{y}(kT)$ is equal to the sum of the first two terms in a Taylor's series expansion of $h_{i,j}(\bar{\boldsymbol{R}})$ about $\bar{\boldsymbol{R}} = \bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}$. To arrive at this truncated Taylor's series is the reason why, in the learning law of equation (22), we decided to evaluate the $h_{i,j}$ at $\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}}$ instead of, e.g., at some more-recently updated learning array. Assuming that the state of the nonlinear dynamical system remains in a close neighborhood of the linearly controllable equilibrium manifold $\mathcal{C}$, the term $\hat{y}(kT)$ is a good approximation of the actual output $y(kT)$.

In addition (again, analogous to the linear case):

$$\begin{aligned}
\boldsymbol{P_{i,j}}(\hat{R}_{i,j-1}) &= \boldsymbol{P_{i,j-1}}(\hat{R}_{i,j-2}) - \boldsymbol{P_{i,j-1}}(\hat{R}_{i,j-2}) \times\\
&\quad \boldsymbol{\Phi_{i,j}^T}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}})\,\boldsymbol{D_{i,j-1}^{-1}}(\hat{R}_{i,j-1}) \times\\
&\quad \boldsymbol{\Phi_{i,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}})\,\boldsymbol{P_{i,j-1}}(\hat{R}_{i,j-2}),
\end{aligned} \tag{34}$$

for $j = 0, 1, \ldots, N-1$. This matrix is initialized to be $\boldsymbol{P_{i,-1}}(\hat{R}_{i,-2}) = \boldsymbol{I}$, where the argument $\hat{R}_{i,-2}$ has no

meaning. And:

$$\boldsymbol{D}^{-1}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-1}) = \left(\boldsymbol{\Phi}_{\boldsymbol{i,\,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}})\,\boldsymbol{P}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-2})\,\boldsymbol{\Phi}^{T}_{\boldsymbol{i,\,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}})\right)^{-1}, \quad (35)$$

for $j = 0, 1, \ldots, N-1$.

Making use of the fact that $\boldsymbol{\Delta\bar{R}_{i,j}} = \boldsymbol{\bar{R}_{i,j}} - \boldsymbol{\bar{R}_{i,-1}}$ and by defining $\bar{e}_{i,j} \triangleq \gamma\,\bar{\varepsilon}_{i,j}$, the recursive algorithm takes the form of the following nested `for` loop.

```
for i = 1,2,3,...,
```

$$\boldsymbol{P}_{\boldsymbol{i,\,-1}}(\hat{R}_{i,-2}) \;=\; \boldsymbol{I} \quad (36)$$

```
    for j = 0,1,...,N-1,
```

$$e_{i,j} \;=\; y_d(jt) - \hat{y}(kT) \quad (37)$$

$$\bar{e}_{i,j} \;=\; \gamma\, e_{i,j} + (1-\gamma)\,\boldsymbol{\Phi}_{\boldsymbol{i,\,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}) \times \left(\bar{\boldsymbol{R}}_{\boldsymbol{i,-1}} - \bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}\right) \quad (38)$$

$$\bar{\boldsymbol{R}}_{\boldsymbol{i,j}} \;=\; \bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}} + \boldsymbol{P}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-2})\,\boldsymbol{\Phi}^{T}_{\boldsymbol{i,\,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}) \times \boldsymbol{D}^{-1}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-1})\,\bar{e}_{i,j} \quad (39)$$

$$\boldsymbol{P}_{\boldsymbol{i,\,j}}(\hat{R}_{i,j-1}) \;=\; \boldsymbol{P}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-2}) - \boldsymbol{P}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-2})\,\boldsymbol{\Phi}^{T}_{\boldsymbol{i,\,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}) \times \boldsymbol{D}^{-1}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-1})\boldsymbol{\Phi}_{\boldsymbol{i,\,j}}(\bar{\boldsymbol{R}}_{\boldsymbol{i,j-1}}) \times \boldsymbol{P}_{\boldsymbol{i,\,j-1}}(\hat{R}_{i,j-2}) \quad (40)$$

```
    end
```

$$\bar{\boldsymbol{R}}_{\boldsymbol{i+1,-1}} = \bar{\boldsymbol{R}}_{\boldsymbol{i,N-1}} \quad (41)$$

```
end
```

In practice, the term $\hat{y}(kT)$ appearing in (37) would be replaced by the actual (measured) output $y(kT)$.

## VI. Experimental Application to Cart and Down-Hanging Pendulum System

The recursive algorithm of the step-by-step learning update law (the nested `'for'` loop of equations (36) through (41)) has been implemented experimentally as a component of the tracking controller described in [2] on a cart and down-hanging pendulum system, illustrated in Figure 1.

This system consists of a uniform rigid bar (the pendulum) of mass $m$ hinged to a (rigid) cart of mass $M$ at a pivot point, $O$. The length of the pendulum (measured from point $O$) is $L$. Let $C$ denote a point that lies at the pendulum's center of mass; it is located at a distance $L/2$, along the pendulum, from point $O$. And let $I$ denote the moment of inertia of the pendulum about point $C$, i.e., $I = mL^2/12$. The cart is constrained by pre-loaded, recirculating ball bearings to travel back and forth along the horizontal track (guide rails) of a permanent-magnet DC linear motor that is used to apply a horizontal force to the cart. Let the variable $x$ denote the position of the cart on the linear motor track and let $\theta$ denote the angle of the pendulum with respect to vertical, i.e., with respect to the direction of gravity, $g$. The positive sense of $x$ and $\theta$ are as shown in Figure 1.

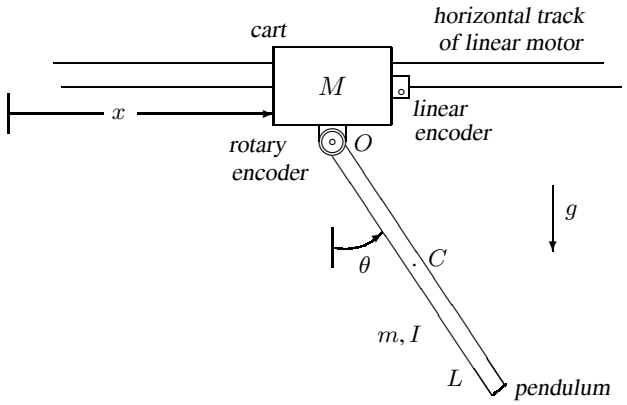


Fig. 1. Schematic diagram of cart and down-hanging pendulum experimental hardware.

The control objective is to get the horizontal position of the pendulum tip, $x + L\sin\theta$, to track a periodic reference trajectory.

The permanent-magnet DC linear motor used to actuate the system is manufactured by *Anorad Corporation*, as is the three-phase, brushless DC amplifier that precedes it. Use of the linear motor and the amplifier, as well as the ball bearings and rail guides that provide (nominally) constant-offset translation of the cart with respect to the linear motor, gives rise to the presence of nonlinearities that are difficult to model precisely: friction, cogging, and torque ripple.

Upon defining $l \triangleq L/2$, $u$ to be the force applied by the linear motor to the cart, and $\boldsymbol{x} \triangleq [x_1\ x_2\ x_3\ x_4]^T = \left[x\ \theta\ \dot{x}\ \dot{\theta}\right]^T$, the system may be modeled by a nonlinear dynamical system with the following vector fields: $\boldsymbol{f}(\boldsymbol{x}) = \left[x_3\ x_4\ \frac{w_1(x_2,x_4)}{w(x_2)}\ \frac{w_2(x_2,x_4)}{-w(x_2)}\right]^T$ and $\boldsymbol{g}(\boldsymbol{x}) = \left[0\ 0\ \frac{I+l^2m}{w(x_2)}\ \frac{l\,m\cos x_2}{-w(x_2)}\right]^T$, where $w(x_2) = (I + l^2m)(m+M) - l^2m^2(\cos x_2)^2$, $w_1(x_2,x_4) = l\,m((I + l^2m)x_4^2 + g\,l\,m\cos x_2)\sin x_2$, and $w_2(x_2,x_4) = l\,m(g(m+M)\sin x_2 + l\,m\,x_4^2\cos x_2\sin x_2)$. For this system to be in a state of equilibrium, it is required that $\dot{\boldsymbol{x}} = \boldsymbol{0}$, or, that $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\,u = \boldsymbol{0}$. This requires that the velocities $x_3 = 0$ and $x_4 = 0$, and that $u = 0$ and $mgl\sin x_2 = 0$, i.e., no force is applied and the pendulum hangs straight down (vertically: $x_2 = 0$); we do not consider unstable equilibriums, e.g., corresponding to $x_2 = \pi$ rad. The cart position $x_1$ may take on any value at equilibrium. It is straightforward to show that the system is linearly controllable in a neighborhood of these equilibrium states. In addition, the system is non-minimum phase with respect to the specified output $y = x_1 + L\sin x_2$.

For this system, the set of linearly controllable equilibrium states is given by: $\mathcal{C} = \{\boldsymbol{x} : x_1 \in (-\infty, \infty), x_2 = 0, x_3 = 0, x_4 = 0,\ \text{and}\ \exists\ u = 0\}$. $\mathcal{C}$ may be expressed as a 1-dimensional manifold. In fact, $\mathcal{C}$ is already in this form, parameterized by $x_1$.

The numerical values of the physical system parameters are: $M = 4\,\mathrm{kg}$, $m = 0.4321\,\mathrm{kg}$, $l = 0.4493\,\mathrm{m}$, $I = 0.0291\,\mathrm{kg\,m^2}$, and $g = 9.81\,\mathrm{m/s^2}$. For the experiments, the desired trajectory $y_d(t)$ is the offset cosine $A - A\cos(2\pi t)$, with $A = 0.0275$ m, containing zero-velocity segments of duration $0.2$ s alternating with the offset cosine portions (the resulting periodic trajectory resembles a rounded-corner square wave).

The learning gain used for the experiments is $\gamma = 0.065$, chosen because this value preserves stability and results in an appreciable and settled diminishing of tracking error, $e = y_d - y$, over the $150$ seconds of the experimental runs. Figure 2 shows the learning control output, $r$, as a function of time for a typical experimental run.

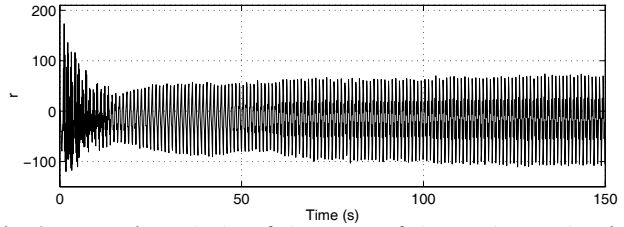


Fig. 2. Experimental plot of the output of the step-by-step learning controller, $r$, as a function of time.

After an initial transient, the response settles into periodicity. The resulting tracking error, $e$, is shown in Figure 3. The error is reduced by several orders of magnitude to achieve precise tracking of the periodic reference trajectory.

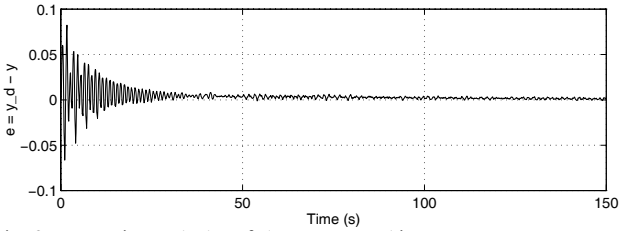


Fig. 3. Experimental plot of the output tracking error, $e = y_d - y$, as a function of time when the step-by-step learning controller is applied (units are meters).

The efficacy of the developed learning controller is strongly dependent on the state of the system to be controlled's proximity to its equilibrium manifold, $\mathcal{C}$. If the state remains in a "close enough" neighborhood of $\mathcal{C}$, then the learning controller guarantees very precise tracking of a periodic trajectory by the system's output (with respect to which the system may be non-minimum phase).

## REFERENCES


[1] G. Strang, *Introduction to Linear Algebra* (Wellesley-Cambridge Press, Welleseley, MA, 1993).

[2] A.-P. Hu and N. Sadegh, "Nonlinear non-minimum phase output tracking via output redefinition and learning control" *Proceedings of the American Control Conference* (2001) pp. 4264–4269

[3] A.-P. Hu, "Nonlinear non-minimum phase output tracking via output redefinition and learning control" *PhD thesis* (Georgia Institute of Technology, 2000).

[4] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes* (Cambridge University Press, New York, NY, 1986).