INTELLIGENT FRACTIONAL ORDER ITERATIVE LEARNING CONTROL USING FEEDBACK LINEARIZATION FOR A SINGLE-LINK ROBOT

IMAN GHASEMI, ABOLFAZL RANJBAR NOEI* AND JALIL SADATI

Faculty of Electrical and Computer Engineering,
Babol University of Technology, Babol, Iran.

**Corresponding author: a.ranjbar@nit.ac.ir*

(Received: 30th May 2015; Accepted: 6th Oct. 2016; Published on-line: 30th May 2017)

ABSTRACT: In this paper, iterative learning control (ILC) is proposed to be combined with an optimal fractional order derivative and optimal fractional and proportional-derivative (BBO-PD$^{\alpha}$-type ILC). During the procedure of the update law (of Arimoto's derivative iterative learning control), a first order derivative of the tracking error signal was used. As a novelty, fractional order derivative of the error signal is proposed in terms of s^{α} where $\alpha \in [0, 2]$ to update the learning control law. Two types of fractional order iterative learning control namely PD$^{\alpha}$-type ILC and D$^{\alpha}$-type ILC are used in a closed loop control system for different value of α. In this regard, a feedback linearization control technique is applied on a single link robot arm. Convergence of the nonlinear control system is also shown for the first time. In order to investigate and also to improve performance of the closed-loop feedback linearization control system, coefficients of both D$^{\alpha}$ and PD$^{\alpha}$ learning law i.e. proportional k_P, derivative k_D, and α are optimized using Biogeography-Based optimization algorithm (BBO). Simulation results signify performance of the proposed optimally tuned BBO-D$^{\alpha}$ and BBO-PD$^{\alpha}$-types ILC with respect to the conventional fractional order ILC.

ABSTRAK: Dalam kertas ini, kawalan pembelajaran lelaran (ILC) adalah dicadangkan untuk digabungkan dengan derivatif susunan pecahan optimum dan derivatif pecahan dan berkadar optimum (BBO-PD$^{\alpha}$-type ILC). Semasa prosedur undang-undang kemaskini (kawalan pembelajaran lelaran derivatif Arimoto ini), derivatif pengesanan isyarat ralat peringkat pertama telah digunakan. Sebagai sesuatu yang baru, derivatif susunan pecahan isyarat ralat dicadangkan mengikut s^{α} mana $\alpha \in [0, 2]$ untuk mengemaskini undang-undang kawalan pembelajaran. Dua jenis susunan pecahan kawalan pembelajaran lelaran iaitu PD$^{\alpha}$-jenis ILC dan D$^{\alpha}$-jenis ILC digunakan dalam sistem kawalan loop tertutup untuk nilai α yang berlainan. Dalam hal ini, satu teknik maklumbalas kawalan linear digunakan pada pautan tunggal lengan robot. Penumpuan sistem kawalan bukan linear juga ditunjukkan buat kali pertama. Bagi menyiasat dan juga untuk meningkatkan prestasi sistem maklumbalas kawalan pelinearan loop tertutup, pekali kedua-dua D$^{\alpha}$ dan PD$^{\alpha}$ undang-undang pembelajaran, iatu k_P, berkadar, derivatif k_D, dan α dioptimumkan menggunakan Biogeografi berasaskan algoritma pengoptimuman (BBO). Keputusan simulasi menunjukkan prestasi BBO-D$^{\alpha}$ yang diperkemaskan secara optimum dan prestasi BBO-PD$^{\alpha}$-jenis ILC yang dicadangkan dari segi susunan ILC pecahan konvensional.

KEYWORDS: *fractional order iterative learning control (FOILC);biogeography-based optimization; PD$^{\alpha}$-type ILC; D$^{\alpha}$-type ILC; input-state feedback linearization controller*

1. INTRODUCTION

Learning is a characteristic of living creatures, human beings among them. Several endeavors have been made to extend this learning ability to engineering systems in design and construction. Iterative learning control is such an important technique in the field of iterative learning systems, proposed by Arimoto and his colleagues in 1984 [1]. Intelligent iterative learning control falls into the intelligent control category. This involves new techniques to control iterative processes in certain amounts of time. In such control algorithms, the controller learns from its past experiences (iterations) to update itself to improve the performance of the closed loop system.

There are several practical examples in industry in which a system or plant must perform a particular duty successively within a certain time duration. For instance, a robot arm [2] must perform a special task (such as welding, cutting, painting, etc.) in a prescribed geometric path at any iteration. One way to control such a process is to modify a control input from the previous iteration to the next in order to decrease the error between the actual path and desired command [2, 3]. Typically, the update law of an iterative learning control can be given by:

$$u_{k+1}(t) = u_k(t) + \Delta u_k(t) \tag{1}$$

where $u_{k+1}(t)$ is the input of the process at iteration $(k+1)$ which $\Delta u_k(t)$ is a correction term. Several types of the iterative learning controls are developed in which different correction terms of $\Delta u_k(t)$ are used. A perfect description of linear and nonlinear iterative learning control law can be found in [4]. Convergence conditions of an adaptive iterative learning law is studied in [5]. A monotonic convergence control law is introduced in [6] to control linear discrete-time where a self-tuning iterative learning control for time variant systems is proposed in [7].

Iteration-based systems use the ability to learn and appropriate tuning of the input to perform a repetitive operation, although the iteration process is time and cost consuming. Accordingly, algorithms to reduce the necessary iterations within the learning process attracted more interest [8]. The first fractional order ILC is reported by [8] where a fractional order learning control law of type D^{α} is proposed. The convergence condition, as an important method in fractional order iterative learning control (FOILC), is also analyzed in the frequency domain.

Fractional calculus was primarily inspired by Leibniz in 1695 [9]. A fractional derivative is a proper tool to define properties such as memory and inherency of many materials and processes. As a result, broad research has been performed in recent years in the field of real-time modeling using fractional order differential equations. For example, power transmission lines can be modeled more accurately using fractional order models [10]. Occurrences of chaos in fractional order systems is investigated in a wide range of research [11]. Among other applications of fractional order systems, modeling of the oscillatory behavior of viscoelastic material [12], electrode-electrolyte polarization [13] and quantum evolution of complex systems [14] can be stated.

In recent years, various research was performed in the field of fractional calculus and its controllers. It is because such controllers are more flexible in comparison with integer order ones. A fractional controller increases the degree of freedom to choose the design parameters. This provides an opportunity for the designer to achieve smoother and more appropriate responses. On the other hand, this increases the performance of the controller in terms of the convergence time and also the steady state error [9].

The first fractional order controller was proposed by Oustaloup in 1988 [15] in the CRONE frame work. Later, this controller was extensively used and developed [16, 17]. Accordingly, initial fractional order controllers and application of fractional order calculus in control were introduced. In 1999, fractional PID controllers were proposed [18].

Biogeography-based optimization (BBO) was primarily proposed by Dan Simon in 2008 [19]. The BBO technique is a new global optimization algorithm based on the study of the geographical distribution of biological organisms. Mathematical models of biogeography deal with immigration of species from one island to another island, explaining the creation and extinction of species. Dan Simon further investigated the BBO algorithm in [20], using a new version of this algorithm and applying the probability theory. In parallel, Mehmet Ergezer and Dan Simon used biogeography-based optimization for some combined problems [21]. Other applications of biogeography-based optimization can be mentioned as reduction of movement estimation time in video [22], dynamic deployment of wireless sensor networks [23], solving the problem of economic load dispatch [24] and optimal dispatch in the power systems [25].

Although iterative learning control is successfully applied in nonlinear systems [8], the convergence analysis is only presented for linear systems. To overcome this shortcoming in the current manuscript, convergence analysis is performed when a nonlinear system is linearized through a feedback linearization approach. Iterative learning updating law of D^{α} type is simulated for $0 \leq \alpha \leq$. However, the result will prove to be improved here when choosing $1 \leq \alpha \leq 2$. Furthermore, a new optimal type of fractional PD^{α} of the iterative learning controller (FOILC) will be proposed for the first time. Coefficients of D^{α} and PD^{α}-types ILC namely, k_P, k_d, and α, are determined using a BBO algorithm. The rest of the paper is organized as follows:

In section **Error! Reference source not found.**, a brief description of fractional calculus is stated. Section 0 introduces integer and fractional order iterative learning control. Motion dynamic and structure of a single-link robot arm is presented in section 0. This will be used as a case study to assess the quality of the proposed fractional type ILC algorithm. In section 5, input-state feedback linearization method is applied on the robot arm. The relevant convergence analysis of the fractional order learning law is given in section 0. A criterion for optimal determination of FOILC coefficients using a BBO algorithm is presented in section 0. Performance of the proposed method is studied during the implementation on a robot arm in section 8. Finally, a conclusion is presented in section 9.

2. BRIEF DESCRIPTION OF FRACTIONAL ORDER CALCULUS

Fractional order calculus is an extension of the integer order differential. Operator ${}_{a}\mathrm{D}_{\mathrm{t}}^{\alpha}$ denotes the fractional order derivative-integral, which is defined as:

$$ {}_{c}D_{t}^{\alpha}(.) = \begin{cases} \dfrac{d^{\alpha}}{dt^{\alpha}}(.) & \Re(\alpha) > 0, \\ 1 & \Re(\alpha) = 0, \\ \displaystyle\int_{\alpha}^{t} (.)(d\tau)^{-\alpha} & \Re(\alpha) < 0. \end{cases} \tag{2} $$

where α indicates order of derivative-integral. For the derivative, α is a positive real value while for the integral, it is a negative real value. Parameters *t* and *c* denote the time and

initial time respectively. This extension leads to various definitions of fractional calculus. A common definition is Caputo's, which is described as follows:

Definition: Necessity of the initial condition such as $f(a)$, $f'(a)$ and… requires the use of a new definition of Riemann-Liouville fractional derivative which is called Caputo fractional derivative [26] as in the following:

$$ {}_c\mathrm{D}_\mathrm{t}^{\alpha} f(t) = \begin{cases} \dfrac{1}{\Gamma(\alpha-n)} \displaystyle\int_c^t \frac{f^{(n)}(T)}{(t-T)^{\alpha-n+1}} dT & , m-1<\alpha<m \\ \dfrac{d^m}{dt^m} f(t) & \alpha = m \end{cases} \tag{3} $$

where m is the first integer number lower than α. The Laplace transform of the Caputo fractional derivative is shown as:

$$ \int_0^{\infty} e^{-st}\, {}_0D_t^{\alpha} f(t)\, dt = s^{\alpha}F(s) - \sum_{k=0}^{n-1} {}_0D_t^{\alpha-k-1} f^{(k)}(t)|_{\,t=0}, \qquad n-1<\alpha \le n \in N \tag{4} $$

Unlike to the Laplace transform of the Riemann-Liouville fractional derivative, only integer order $F(s)$ is appeared in the Laplace transform of the Caputo fractional derivative. In case of a zero initial condition, the Laplace transform has the form;

$$ L\{\, {}_0D_t^{\alpha} f(t)\} = s^{\alpha}F(s) \tag{5} $$

3. INTEGER AND FRACTIONAL ORDER ITERATIVE LEARNING CONTROL

The basic idea of using iterative learning control is shown in Fig. 1. It is assumed that all signals are defined in the time duration of $t \in [0, t_f]$ where k denotes the number of the experiment(s) or iteration. It means that during k^{th} iteration, prior information of input signal up to $u_k(t)$, actual output $y_k(t)$, and error signal $e_k(t)$ are stored in the memory. This information is used to update the iterative learning control law in iteration $k+1$ in order to improve the control input. This is necessary to decrease the error between the actual value and that of the desired system output as well as to increase performance of the closed loop system. The new input must be designed such that the error is absolutely less than that of in the last iteration.

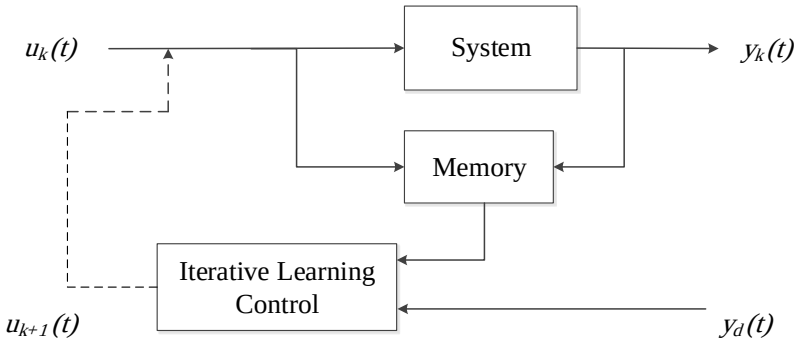

Fig. 1: Basic idea of ILC.

Conventional iterative learning control laws are proportional and derivative. For the derivative iterative learning control updating law, $\Delta u_k(t)$ contains only a derivative according to [27]:

$$\Delta u_k(t) = \Gamma \frac{d}{dt} e_k(t), \ t = 0,1, \ldots, T \ , \quad k = 0,1,2, \ldots \tag{6}$$

Replacing Eqn. (1) into (6) yields the following update law:

$$u_{k+1}(t) = u_k(t) + \Gamma \frac{d}{dt} e_k(t) \tag{7}$$

Likewise, a proportional iterative learning control updating law generates $\Delta u_k(t)$ and $u_{k+1}(t)$ which is as follows:

$$\Delta u_k(t) = \Gamma e_k(t) \tag{8}$$

$$u_{k+1}(t) = u_k(t) + \Gamma e_k(t) \qquad t = 0,1, \ldots, T \ , \quad k = 0,1,2, \ldots \ldots \tag{9}$$

In equations (6)–(10), $e_k(t)$ is the tracking error between the actual output $y_k(t)$ and the desired path $y_d(t)$ at k^{th} iteration. This can be obtained by:

$$e_k(t) \triangleq y_d(t) - y_k(t) \tag{10}$$

From equations (6) and (8), it can be seen that the way of achieving $\Delta u_k(t)$ defines the type of iterative learning control law. In the above equation Γ is the learning gain. Signal $u_k(t)$ is the control input at k^{th} iteration and k denotes the number of iteration. Parameter $t \in [0, t_f]$ indicates the time variable, which may be of a discrete or continuous variable. t_f is the known duration of each iteration. Input (u) and output (y) are not known in advance. Equations (11) and (12), introduce fractional order iterative learning control of D^α and PD^α types in the time respectively.

$$u_{k+1}(t) = u_k(t) + k_D \frac{d^\alpha}{dt^\alpha} e_k(t) \tag{11}$$

$$u_{k+1}(t) = u_k(t) + k_p e_k(t) + k_D \frac{d^\alpha}{dt^\alpha} e_k(t) \tag{12}$$

where $\alpha = 0$ defines a proportional iterative learning law. Similarly, when $\alpha = 1$ shows a derivative iterative learning law. Frequency domain of the fractional order iterative learning controls of (11) and (12) are shown as:

$$u_{k+1}(s) = u_k(s) + k_D s^\alpha E_k(s) \tag{13}$$

$$u_{k+1}(s) = u_k(s) + k_P E_k(s) + k_D s^\alpha E_k(s) \tag{14}$$

where $E_k(s)$ is the Laplace transform of the error $e_k(t)$ in (10). In equations (11) and (12), the proportional coefficient k_P and the derivative coefficient k_D are unknown constant learning gains that must be appropriately determined. In the current manuscript, the effect of choosing α in the interval $\alpha \in [0,2]$ on the convergence of the error in (10) will be investigated.

4. ROBOT ARM STRUCTURE AND THE MOTION DYNAMIC

Robot arms usually perform repetitive operations. It is then meaningful to use benefits of past experience(s) in iterative learning control approaches. This improves the response and increases the efficiency and accuracy. A single-link robot arm [28] as in Eqn. **Error! Reference source not found.** is used to study the dynamic and to simulate behavior of the proposed FOILC.

$$\ddot{\theta}(t) = \frac{1}{J}\big(u(t) - F(t)\big) + \frac{1}{J}\Big(\frac{1}{2}m + M\Big)gl\ \sin\theta(t) \tag{15}$$

where $\theta(t)$ is the position of the robot hand, $u(t)$ is the applied joint moment (as an input), $F(t)$ is the frictional moment, m and l are the mass and the length of the robot arm respectively. Furthermore *M* is the mass of blade tip, g is the gravity acceleration and finally, *J* is the joint momenta inertia. The joint inertia and frictional moment are described as follows:

$$J = Ml^2 + ml^2/3 \tag{16}$$

$$F(t) = \begin{cases} f^+ + B^+\dot{\theta}(t) \\ f^- + B^-\dot{\theta}(t) \end{cases} \tag{17}$$

where the column friction is assumed [28] as $\mathrm{f}^+ = 8.43\mathrm{Nm}$ and $\mathrm{f}^- = -8.26\mathrm{Nm}$ and the viscous frictional coefficient is $\mathrm{B}^+ = 4.94\frac{\mathrm{Nm}}{\mathrm{rad}}/\mathrm{sec}$ and $\mathrm{B}^- = 3.45\frac{\mathrm{Nm}}{\mathrm{rad}}/\mathrm{sec}$. Other simulation setting are: $\mathrm{m} = 2\ kg$, $l = 0.5\ m$ and $g = 9.8\ m/s^2$. Finally, the state vector of the nonlinear robot arm system is considered as follows:

$$x = [x_1 \quad x_2]^T = [\theta \quad \dot{\theta}]^T \tag{18}$$

Comparing the system dynamic equations in the following state space format:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{19}$$

results nonlinear function $f(x)$ and $g(x)$ as:

$$\begin{aligned} f(x) &= \begin{pmatrix} x_2 \\ -\frac{1}{J}(f + Bx_2) + \frac{1}{J}\big(\frac{1}{2}m + M\big)gl\ \sin x_1 \end{pmatrix}, \\ g(x) &= \begin{pmatrix} 0 \\ \frac{1}{J} \end{pmatrix},\ h(x) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \end{aligned} \tag{20}$$

The dynamic of the robot arm in Eqn. (15) will be controlled through a state feedback linearizing controller in the next section.

5. INPUT-STATE FEEDBACK LINEARIZATION CONTROLLER

A schematic diagram of an input-state feedback linearization technique is illustrated in Fig. 2 where a nonlinear controller will be realized in two steps. First, a state transformation will be used to convert nonlinear dynamic into a linear one in terms of input-state of the

plant as: $\dot{z} = Az + Bv$. Some other conventional (and even advanced) controllers may be used to make the control aim possible. It is primarily necessary to

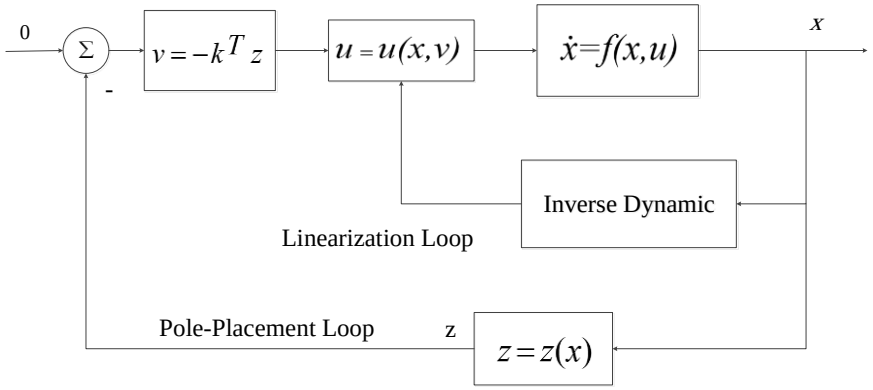

Fig. 2: Schematic diagram of the input-state linearization.

verify if the dynamic can be linearized in terms of input to state variables. This is possible if and only if both following conditions are simultaneously satisfied [29]:

1- Vectors of matrix $G(x) = [g(x), ad_f g(x), \ \ldots, ad_f^{n-1} g(x)]$ are of full rank.

2- Distribution of $span[g(x), ad_f g(x), \ldots, ad_f^{n-2}\ g(x)]$ is involutive.

In the above enumerated terms, $ad_f g(x)$ denotes the Lie bracket derivative which is defined as:

$$ad_f g(x) = \nabla g.f - \nabla f.g \tag{21}$$

The number n indicates the relative order (degree) of the system. This eventually means the required number of differentiations to get the input to appear, which is found to be $n = 2$ for the robot arm dynamic. The Jacobian matrix of $f(x)$ as shown by ∇f, is achieved as:

$$\nabla f = \frac{\partial f}{\partial x}, \quad \nabla g = \frac{\partial g}{\partial x} \tag{22}$$

The $n \times n$ Jacobian matrix consists elements of $(\nabla f)_{ij} = \partial f_i / \partial x_j$. In order to show that the robot arm system can be fully linearized, both enumerated conditions must be met. By evaluating vector fields, matrix $G(x)$ in the first condition can be achieved as:

$$ad_f g(x) = \nabla g\, f - \nabla f\, g = \begin{bmatrix} -\frac{1}{J} \\ \frac{1}{J^2} B \end{bmatrix} \tag{23}$$

$$G(x) = \left[g(x), ad_f g(x)\right] = \begin{bmatrix} 0 & -\frac{1}{J} \\ \frac{1}{J} & \frac{1}{J^2} B \end{bmatrix} \tag{24}$$

The rank of the $G(x)$ matrix is found 2 ($\mathrm{rank}\big(G(x)\big) = 2$). The span of $\Delta = \mathrm{span}\{g\}$ is also found involutive. Therefore a state $z = z(x)$ alteration, together with the control $u =$

$\sigma(x) + \beta(x)v$ transformation, linearize the dynamic in Eqn. (19) in the sense of the input-state linearization. In this regard, $z(x)$ must be determined such that:

$$\begin{cases} \nabla z_1 ad_f^i g = 0 & i = 0, \dots, n-2 \\ \nabla z_1 ad_f^{n-1} g \neq 0 \end{cases} \tag{25}$$

Equation (25) immediately yields:

$$\frac{\partial z_1}{\partial x_2} = 0, \qquad \frac{\partial z_1}{\partial x_1} \neq 0. \tag{26}$$

As a result, z_1 must only be a function of x_1 i.e.:

$$z_1 = x_1 \tag{27}$$

Other states can be successively achieved from $z_1$by:

$$z_2 = \nabla z_2 f = x_2 \tag{28}$$

Hence, the input linearizing the dynamic is found as seen in Eqn. (29), together with the relevant coefficients in Eqns. (30) and (31).

$$u = \sigma(x) + \beta(x)v \tag{29}$$

$$\sigma(x) = -\frac{L_f^n z_1}{L_g L_f^{n-1} z_1} = -\frac{L_f^2 z_1}{L_g L_f z_1} = f + Bx_2 - \left(\frac{1}{2}m + M\right) gl \sin x_1 \tag{30}$$

$$\beta(x) = -\frac{1}{L_g L_f^{n-1} z_1} = J \tag{31}$$

Ultimately input (u)-state (x_1, x_2) transformation linearizes the dynamic as follows:

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = v \\ y = z_1 \end{cases} \tag{32}$$

This eventually means:

$$\begin{cases} \dot{z} = Az + Bv \\ y = Cz \end{cases} \tag{33}$$

where matrices A, B, and C in Eqn. (33) are as:

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v \tag{34}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \tag{35}$$

In conclusion, the input-state transformation in Eqns. (26) to (31) converts the stability analysis of the nonlinear dynamic (15) using the main control input *u* to the stability analysis of the new dynamic (32) via the new input *v*. Since the new dynamic is linear and controllable, the following pole placement technique may be used to generate *v*.

$$v = -K_1 z_1 - K_2 z_2 \tag{36}$$

It is seen that the linear state feedback control is capable of arbitrarily placing poles by choosing proper feedback gains.

6. CONVERGENCE ANALYSIS OF THE PROPOSED FRACTIONAL ORDER CONTROLLER

In this section, convergence analysis of the proposed PD$^{\alpha}$-type ILC is presented. For this system, input is $u(t)$ where the output is $y(t)$. It is assumed that the linearized plant $\mathrm{G_C(s)}$, is internally BIBO stable. It is assumed that a unique input can be found such that the desired output trajectory $y_d(t)$ in $t \in [0, T]$ can be produced:

$$y_d(t) = G_c u_d(t) \tag{37}$$

Using (12) to (14), a PD$^{\alpha}$-type ILC update law can be found as in the following:

$$\begin{aligned} u_{k+1}(s) &= u_k(s) + k_P\left(y_d(s) - y_k(s)\right) + k_{\mathrm{D}}\mathrm{s}^{\alpha}\left(y_d(s) - y_k(s)\right) \\ &= u_k(s) + k_P\, G_C(s)\left(u_d(s) - u_k(s)\right) + k_D s^{\alpha}\, G_C(s)\left(u_d(s) - u_k(s)\right) \\ &= \left(1 - k_P G_C(s) - k_D s^{\alpha} G_C(s)\right)u_k(s) + (k_P + k_{\mathrm{D}}\mathrm{s}^{\alpha})\mathrm{G_C(s)}u_d(s) \end{aligned} \tag{38}$$

where ρ is defined as follows:

$$\rho = 1 - k_P \mathrm{G_C(s)} - k_{\mathrm{D}}\mathrm{s}^{\alpha}\mathrm{G_C(s)} \tag{39}$$

It recursively yields:

$$\begin{aligned} u_1(s) &= u_0(s)\rho + (k_P + k_{\mathrm{D}}\mathrm{s}^{\alpha})\mathrm{G_C(s)}u_d(s) \\ u_2(s) &= u_0(s)\rho^2 + \tfrac{1-\rho^2}{1-\rho}(k_P + k_{\mathrm{D}}\mathrm{s}^{\alpha})\ \mathrm{G_C(s)}u_d(s) \\ u_3(s) &= u_0(s)\rho^3 + \tfrac{1-\rho^3}{1-\rho}(k_P + k_D s^{\alpha})\ G_C(s)u_d(s) \\ &\vdots \\ u_{k+1}(s) &= u_0(s)\rho^{k+1} + \tfrac{1-\rho^{k+1}}{1-\rho}(k_P + k_D s^{\alpha})\ G_C(s)u_d(s) \end{aligned} \tag{40}$$

In Eqn. (40), if $|\rho| < 1$ then:

$$\lim_{k\to\infty}\left(u_{k+1}(s)\right) = \lim_{k\to\infty}\left(u_0(s)\rho^{k+1} + \left(1 - \rho^{k+1}\right)u_d(s)\right) = u_d(s) \tag{41}$$

Thus, the convergence condition of the proposed learning law may be stated as follows:

$$|1 - k_P G_C(j\omega) - k_D(j\omega)^{\alpha} G_C(j\omega)| < 1, \quad \forall\omega \tag{42}$$

If α, k_P and k_{D} are chosen such that the condition in Eqn. (42) is met, the proposed D$^{\alpha}$-type ILC is convergent and the error asymptotically approaches zero. i.e:

$$\lim_{k\to\infty}\left(y_d(s) - y_k(s)\right) = 0 \tag{43}$$

7. OPTIMIZATION OF FOLIC PERFORMANCE USING THE BBO ALGORITHM

It is observed in the previous section that if α, k_P, and k_D are properly chosen such that (42) is satisfied, then the proposed D^{α}-type ILC is convergent. In this section, a biogeography-based optimization algorithm [19] will be used to tune those gain parameters by defining a criterion. A BBO algorithm is an evolutionary population-based technique that is inspired by animal and bird migrations to islands. This method has some common properties with other biology-based algorithms such as genetic and suspended particle swarms.

Appropriate islands are shown by a habitat suitability index (HSI) to spot the merit for the life of a biological species is. Factors such as the rainfall, vegetative diversity, land area, temperature, or ground determine properties of HSI. The suitability index variables (SIVs) are other factors to be considered as habitat independent variables while the HSIs are considered as habitat-dependent variables.

Many species live in habitats with high HSI, thus the species emigration rate to the adjacent habitat is high whilst the immigration rate is low. Habitats with low HSI have few species that define a high immigration rate and the lower emigration rate. An example of species abundance in a habitat is depicted in Fig. 3.

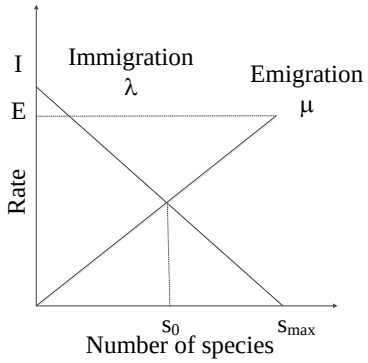

Fig. 3: Species model of a single habitat based on [19].

The emigration rate (μ) and immigration rate (λ) are functions of the numbers of species in the habitats. The maximum rate of emigration E, occurs when the habitat hosts a maximum number of species that it can support. Whereas the maximum immigration rate to habitat I occurs when there are zero species in the habitat. The species balance number is the point where the emigration and immigration rates are equal (denoted by S_0). Immigration and emigration rates are functions of the number of species living in a habitat. They can be respectively evaluated by λ_S and μ_S, which are as follows:

$$\lambda_S = I\left\{(1 - \frac{S}{S_{max}})\right\} \tag{44}$$

$$\mu_S = E(\frac{S}{S_{max}}) \tag{45}$$

In a specific case for E=I, immigration and emigration rates are as follows:

$$\mu_S + \lambda_S = E \tag{46}$$

where maximum allowable rate of immigration is E, S is the number of S^{th} personal species. The maximum number of species in habitat is shown by $S_{\max}$ in which the immigration rate is zero and the maximum allowable emigration rate (E) occurs. BBO concept is based on migration and mutation which are dealt in the following.

7.1 Migration Strategy

The migration action in the biogeography algorithm is similar to the recombination operator in the genetic and evolutionary algorithm. This is used to modify non-elite responses. Migration can be described as a mapping of $H_j(SIV)$ to $H_i(SIV)(H_j(SIV) \rightarrow H_i(SIV))$ [30]. It is assumed that there are N habitats where H_i is the one with the immigration rate λ_i. H_j is the next habitat with the emigration rate μ_j. An extended migration operator of the standard BBO operator is blended migration which is as follows [30]:

$$H_i(SIV) \leftarrow \gamma H_i(SIV) + (1-\gamma)H_j(SIV) \tag{47}$$

In Eqn. (47), γ is a real number between 0 and 1 that can be chosen either at random or by the user.

7.2 The Mutation Operator

Sudden events lead to deviation of the species number from their mean (balance) value and also sudden changes of habitat HSI. In BBO, this is shown by SIV mutation. BBO algorithm may not lead to an optimal point or may diverge from an optimal point. However, after migration, the mutation operator has to be applied on the achievement so far to prevent diversion (or getting stuck at a point). The main aim of the mutation is to create diversity in the solution set or to increase the habitat among the population [19, 31]. The probability of the species number P_S, denotes that the habitat involves exactly S species. P_S is updated from t to $(t + \Delta t)$ using λ_s and μ_s according to:

$$P_s(t+\Delta t) = P_s(t) + \left(1 - \lambda_s \Delta t - \mu_s \Delta t\right) + P_{s-1}\lambda_{s-1}\Delta t + P_{s+1}\mu_{s+1}\Delta t \tag{48}$$

where Eqn. (48) is used to express the mutation rate. Assume that a habitat with S species is determined to be mutated; change the chosen variable (SIV) based on the existence probability P_S. The mutation rate $m(s)$ is calculated as follows using the probability of the number of species in habitat P_S:

$$m(s) = m_{max}\left(1 - \frac{P_s}{P_{max}}\right) \tag{49}$$

where m_{max} is determined by the designer and P_{max} is the maximum probable number of species. Number P_S shows the existence of the probability of species S in the habitat.

7.3 The Procedure and the Cost Function

To achieve an optimal set of FOILC coefficients, the prescribed BBO optimization algorithm (in section 3) is used. Each SIVs (suitability index variables) of PD^{α}-type ILC controller consists of three parameters α, k_P, and k_D. Similarly, each SIVs of D^{α}-type ILC controller consists of two parameters: α and k_D. To determine the HSI or the cost function for each SIV, perform the following steps:

- First, the iterative learning control algorithm begins with an initial value of SIV.
- At the end of the iteration in the ILC algorithm, evaluate the error in terms of the integral of time multiplied by the square value of the error (ITSE). The last term denotes the cost function, which is defined as follows:

$$Fitness\ Function = \int_0^{t_{\text{sim}}} t.e^2(t)\,.\,\mathrm{dt} = \int_0^{t_{\text{sim}}} t.(y_d(t) - y_k(t))^2\,.\,\mathrm{dt} \quad (50)$$

- The procedure continues until the stop criterion is met. The criteria will be given either in terms of the number of the iterations or the relative discrepancy of successive cost functions.

In Eqn. (50), t_{sim} is the previously described duration of the simulation time. In brief, the extended BBO algorithm diagram is depicted in Fig. 4.

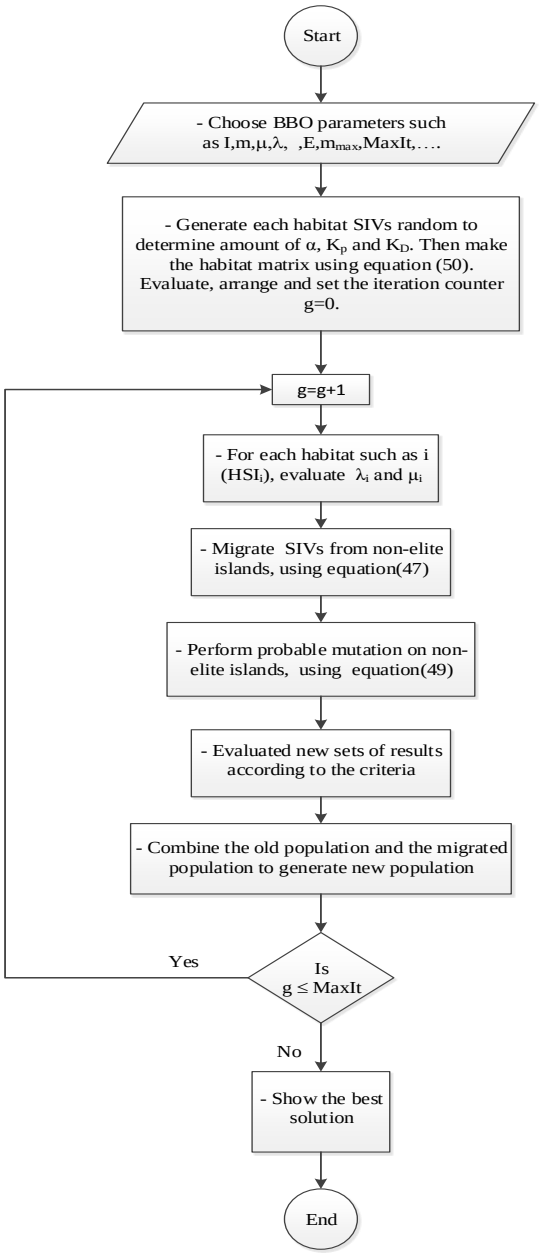

Fig. 4: The prescribed BBO flowchart.

8. SIMULATION

In this section, a brief description of the desired trajectory, together with the feedback linearization technique, will be given. Thereafter, the proposed fractional order iterative learning control updating law of D$^{\alpha}$-type ILC and PD$^{\alpha}$-type ILC is used to verify the capability of the proposed controller. The fractional controller is then used to manipulate the single-link robot arm. Thereafter, the BBO algorithm is used to optimize coefficients of the fractional order ILC controller.

The desired output trajectory of the robot arm is depicted in Fig. 5 [8], which is expressed as follows:

$$\theta_d(t) = \theta_b + (\theta_b - \theta_f)(15r^4 - 6r^5 - 10r^3) \tag{51}$$

where:

$$r = \frac{t}{t_f - t_0}, \text{ and } \quad t_{sim} = t_f - t_0. \tag{52}$$

During simulation, the angular position of the robot arm is assumed $\theta_b = 0$ and $\theta_f = 90$ where $t_0 = 0$ and $t_f = 1$. Finally, the state feedback gain vector $K = [2, 3]$ locates desired poles of the closed loop system in $[-1, -2]$.

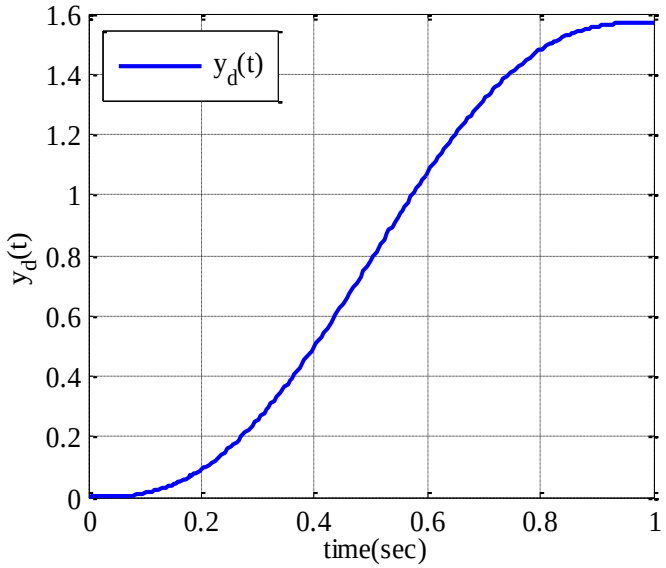

Fig. 5: Desired output trajectory $\theta_d(t) = y_d(t)$.

The proposed procedure is performed according to the following steps:

- First, the nonlinear dynamic of the robot hand in Eqn. (15) is linearized as schematically shown in Fig. 2.
- The closed-loop poles are placed at the desired location, using the state feedback scheme through the feedback gain vector $K = [K_1\ K_2]$. This essentially stabilizes the complete system.
- The proposed ILC (13), as shown in Fig. 1, will be used to generate the control effort via an updating law in Eqn. (13) or Eqn. (14) where applicable.

- Ultimately, the BBO algorithm will be used to tune parameters of the fractional ILC.

8.1 Fractional-Type ILC

The initial condition of the ILC at any iteration is set to zero. Moreover, the angular velocity is assumed accessible. For $\alpha = 1$ in (13), the best choice for Γ is J that is determined by (16) [32]. In the meantime, 'N integer' is used to implement the fractional order controllers in MATLAB®. Fractional order derivative S^{α}, $\alpha \in R$ is calculated using the 'nid' function in the "CRONE" approximation.

The gain k in the 'nid' function is evaluated as the leaning gain Γ or k_D, which is equal to $k = J$. The bandwidth is [0.01 100] rad/s, the number of zeros and poles are assumed to be "n = 5", the applied method is "expansion = cf" and the approximation style is set on "decomposition = all". The above setting leads to achieve the error e_m ($e_m = \sup_{t\in[0,1]}\left(\theta_d(t) - \theta(t)\right)^2$) to be depicted as in Fig. 6, for 40 iterations. The D^{α}-type ILC updating law is used generate the process input at $k+1^{th}$ iteration.

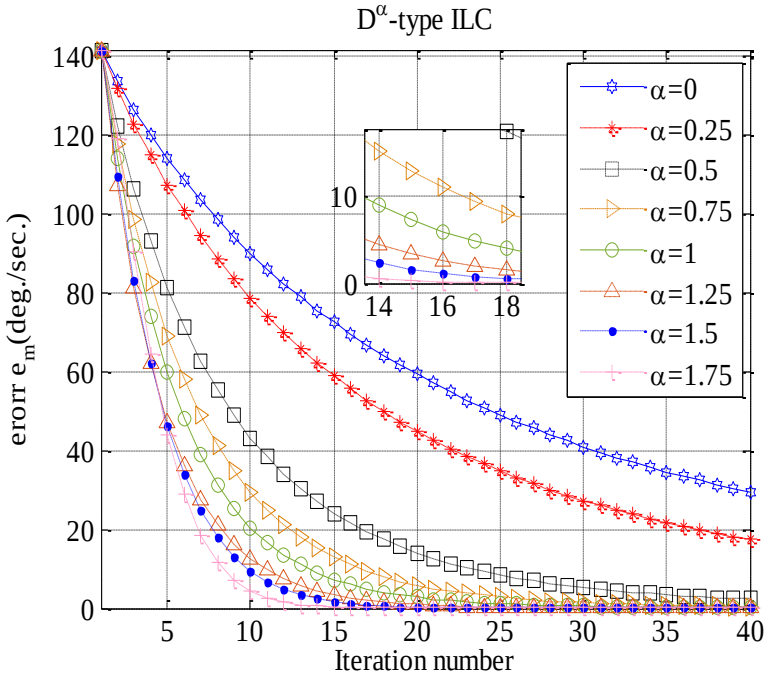

Fig. 6: Maximum square error (MSE) of the tracking versus the iteration number, Comparison of D^{α}-type ILC with different α.

In Fig. 6, a D^{α}-type ILC updating law is used to investigate performance of the integer and fractional iterative learning control. The simulation results are depicted for the square of the tracking error versus the iteration number for different value of α. When $\alpha = 0$, the state variables are only used to update the ILC. In this case, the update law is of a P-type ILC. In contradiction, for $\alpha = 1$, the derivative of state variables are used, which involves angular acceleration. This means the updating law is of a D-type ILC. Figure 7 shows the required control signal when the ILC has experienced 40 iterations. These control efforts cause the output of D^{α}-type ILC as seen in Fig. 8 for different values of α, as illustrated.

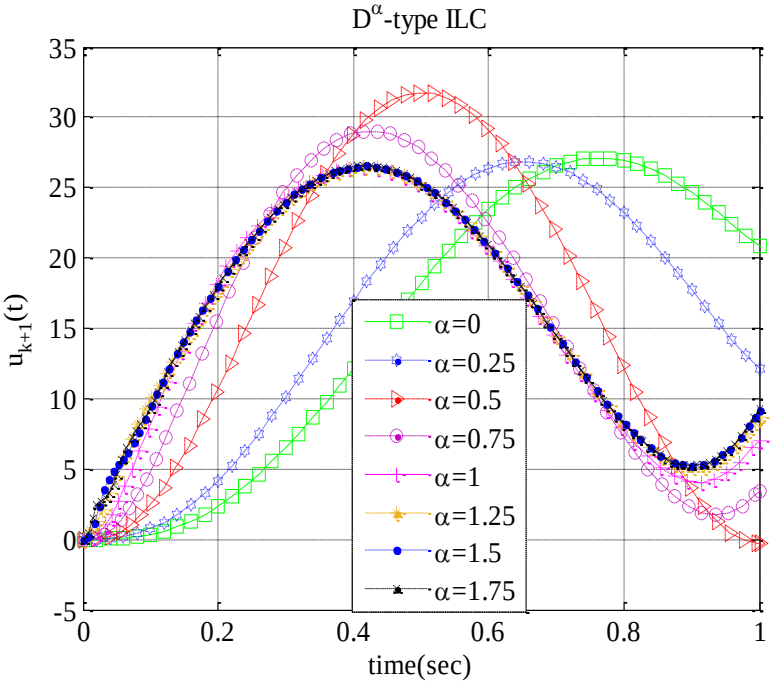

Fig. 7: The control effort vs. different values of α after 40 iterations using the updating law of D$^{\alpha}$-type ILC.

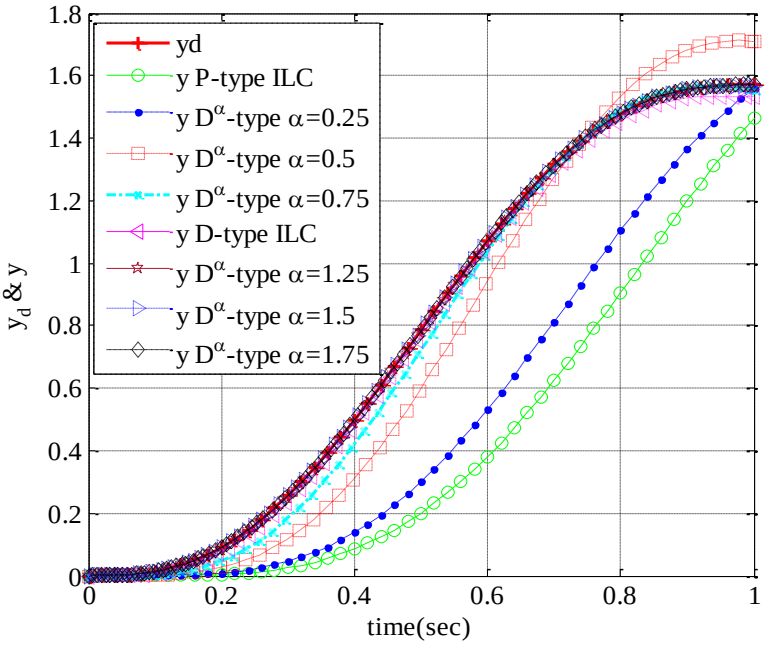

Fig. 8: Desired and the output response of the D$^{\alpha}$-type ILC update law, for different values of α and k=40.

From Fig. 8, it can be seen that after 40 iterations, the fractional type of $\alpha = \{1, 1.25, 1.5, 1.75\}$ provides perfect convergence. In contrast, lower α i.e. $\alpha = \{0, 0.25, 0.5, 0.75\}$ means that the number of tries must be increased. If the precision is reduced, meaning the stop criterion of the ILC is chosen as $e_{\mathrm{m}} < 0.01^{\circ}$, the ILC converges after 16 iterations as seen in Fig. 10.

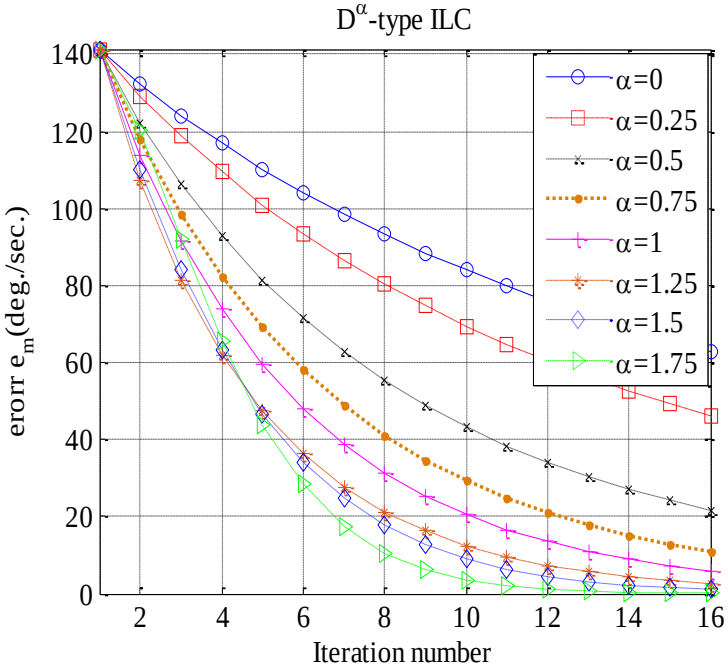

Fig. 9: Maximum absolute angular tracking errors versus the iteration number, Comparison of D$^{\alpha}$-type ILC with different α, with respect to iteration number 16.

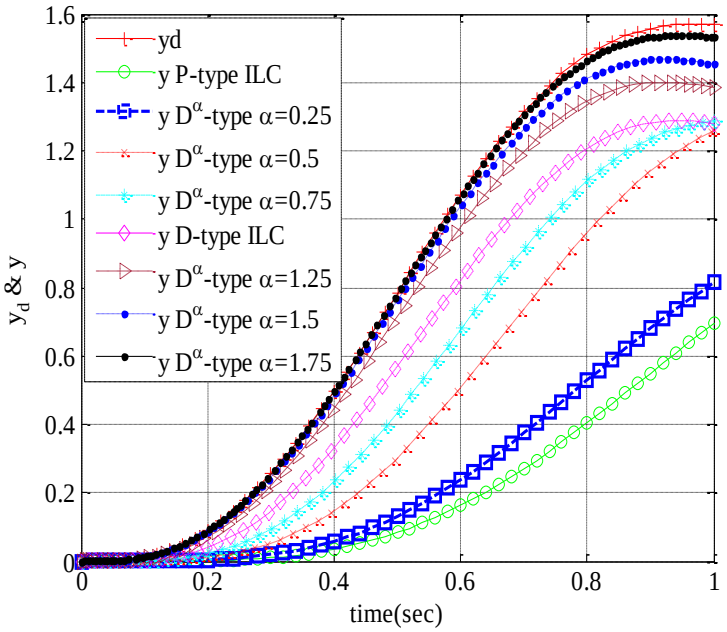

Fig. 10: The control effort for different values of α when $e_m < 0.01^{\circ}$, using updating law D$^{\alpha}$-type ILC and k=16.

From Fig. 10, it can be seen that $\alpha = 1.75$ provides rapid convergence with respect to other α's. To illustrate the performance of the PD$^{\alpha}$-type ILC updating law, Eqn. (**Error! Reference source not found.** is used to obtain the process input at the k+1st iteration. During simulation, the value of the learning gains are chosen as $k_P = J/2$ and $k_d = J$, whereas the other settings are kept the same as with the D$^{\alpha}$-type ILC law. The trend of the

error, in terms of maximum absolute tracking error, is depicted in Fig. 11 for different values of α.

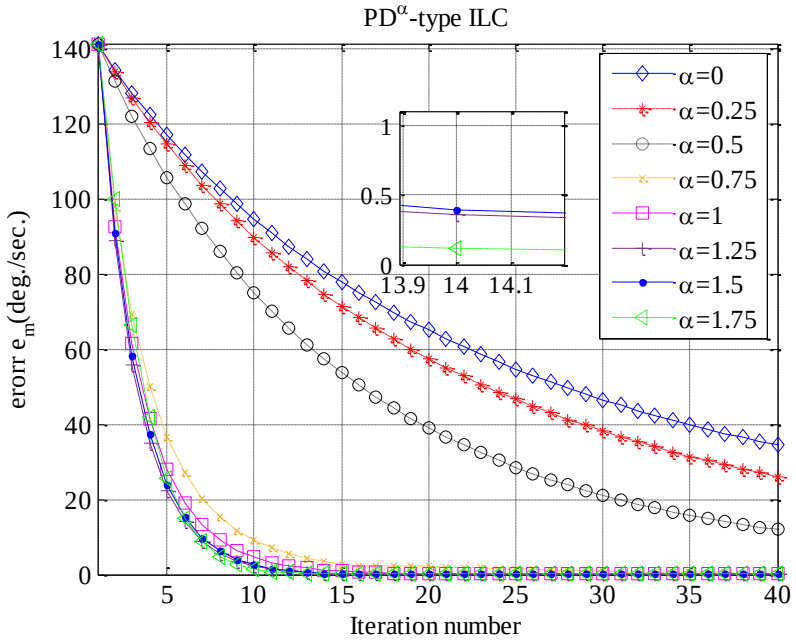

Fig. 11: Maximum absolute angular tracking errors versus the iteration number, comparison of PD$^{\alpha}$-type ILC with different α.

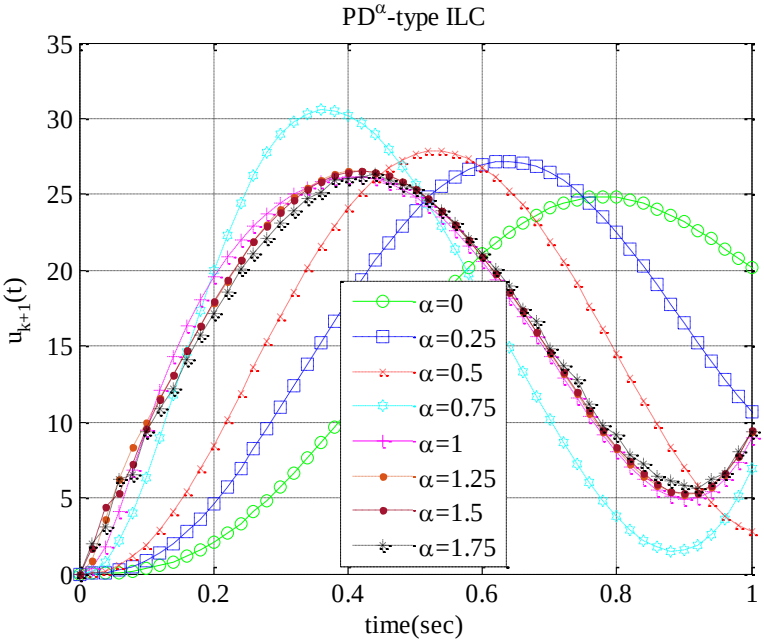

Fig. 12: The control effort for different values of α, by using updating law PD$^{\alpha}$-type ILC and k=40.

Time response of actual output y to desired output y_d using PD$^{\alpha}$-type ILC updating law is depicted in Fig. 11 during 40 iterations. It can be seen that for some value of α, the PD$^{\alpha}$-type ILC updating law can provide the convergence. The obtained control input signal, using PD$^{\alpha}$-type ILC updating law for different value of α, is shown in Fig. 12 for 40 iterations.

From Fig. 13, it can be seen that the PD$^{\alpha}$-type ILC updating law (for α=1.75 and considering the stop criterion $e_m < 0.01^{\circ}$) convergence is achieved with fewer iterations e.g. 14 iterations.

8.2 The BBO Tuned Fractional Types ILC

In this regard, FOILC coefficients are adjusted in collaboration with the previously described BBO algorithm. Parameters of the BBO are set as follows:

The maximum iteration is set to 10, the population size to 50, the elite island as 5, the species dimension (SIV) is 3, the maximum mutation rate is m_{max}= 0.05, and in (44) γ=0.9 and finally the maximum rate of Emigration and Immigration are set to E=1 and I=1 respectively.

Figure 14 shows a uniform convergence of the BBO algorithm when the fitness is gradually decreased. This BBO setting ends with coefficients of the FOILC, as in Table. 1. The time behavior of the error of both PD$^{\alpha}$ and D$^{\alpha}$-type ILC is illustrated in Fig. 15. The required control effort of using both types of fractional order learning law is depicted in Fig. 16. The results after 10 iterations of the BBO algorithm for BBO-D$^{\alpha}$-type ILC and BBO-PD$^{\alpha}$-type ILC are illustrated in Fig. 17.

Table. 2: Parameter values obtained from the BBO algorithm for performance comparison between PD$^{\alpha}$-type ILC and D$^{\alpha}$-type ILC.

FOILC	k_p	k_D	α	J_{ITSE}
BBO – D$^{\alpha}$ type	-	3.3712	1.5886	2.0172×10^{-9}
BBO - PD$^{\alpha}$ type	3.179	1.808	1.7302	1.384×10^{-10}

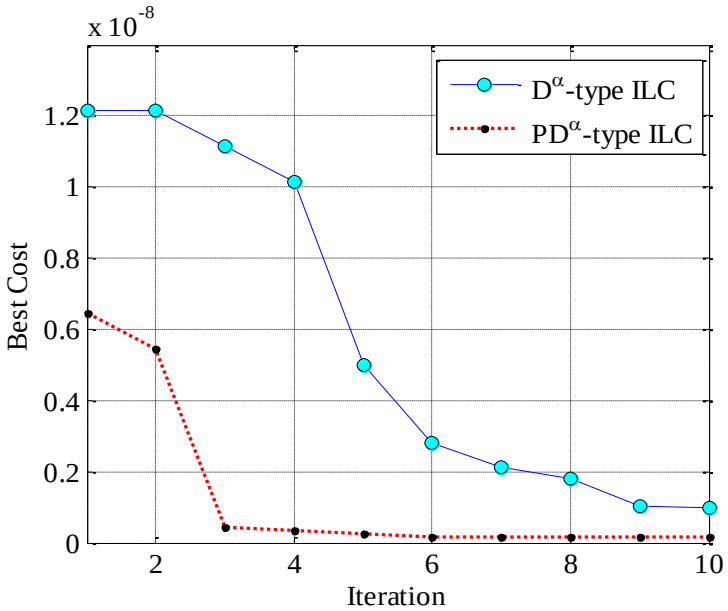

Fig. 14: Maximum absolute angular tracking errors versus the iteration number, for D$^{\alpha}$-type ILC and PD$^{\alpha}$-type ILC using the BBO algorithm.

In comparison with Fig. 15, it can be seen that the convergence speed is increased when FOILC coefficients are tuned by the BBO algorithm. Meanwhile, Fig. 9 confirms that the maximum convergence speed for D^{α}-type ILC occurs in 16 iterations. Whereas using the BBO adjusted D^{α}-type ILC much improves the convergence rate by a factor of 3 times i.e. 5 iterations, as seen in Fig. 14. A similar result is achieved for PD^{α}-type ILC when the maximum convergence speed occurs at iteration 14 (Fig. 11), whereas the BBO designed PD^{α}-type ILC occurs in 4 iterations (Fig. 14). This confirms a huge improvement of the convergence speed when the BBO is used to optimally tune parameters of the ILC.

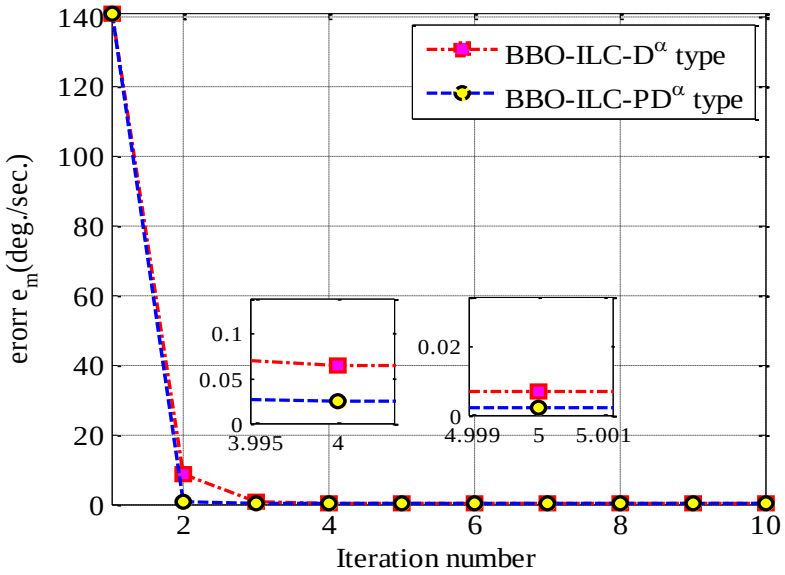

Fig. 15: Maximum absolute angular tracking errors versus the iteration number, for D^{α}-type ILC and PD^{α}-type ILC using the BBO algorithm.

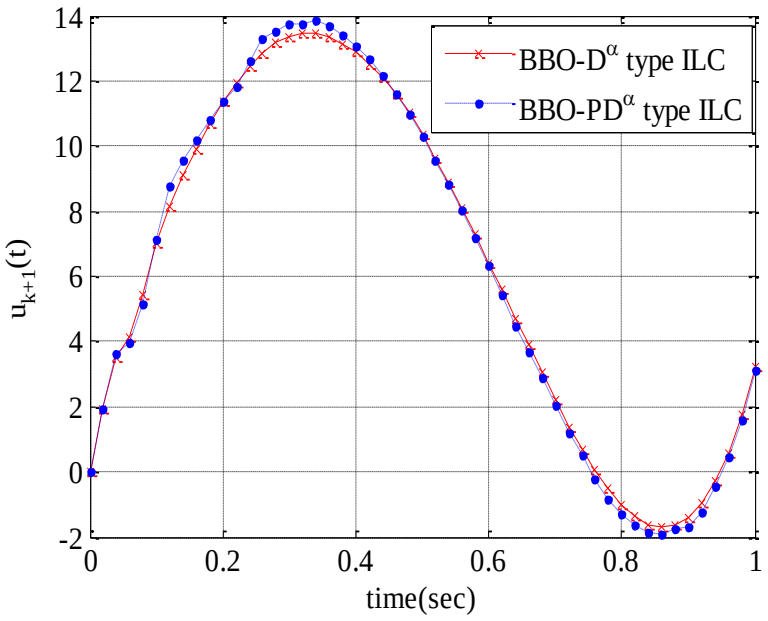

Fig. 16: The control signal for different values of α, by using updating laws BBO-D^{α}-type ILC and BBO-PD^{α}-type ILC for k=10.

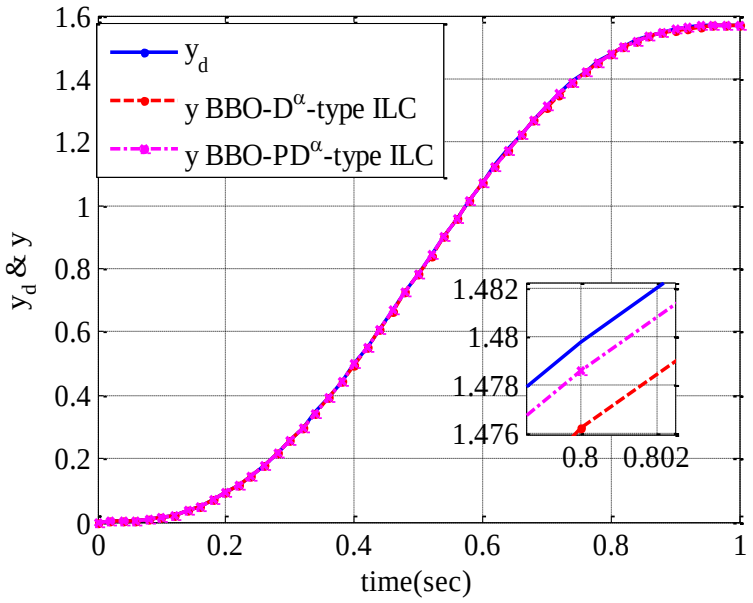

Fig. 17: The desired output trajectory of the robot arm of BBO-D$^{\alpha}$-type ILC and BBO-PD$^{\alpha}$-type ILC updating laws, for k=10 and different values of α.

9. CONCLUSION

In this paper, by combining fractional order calculus and an ILC controller, the performance of two types of updating laws, called PD$^{\alpha}$-type ILC and D$^{\alpha}$-type ILC, are investigated. Necessary conditions of convergence for the proposed PD$^{\alpha}$-type ILC for the nonlinear robot arm are presented for the first time. A feedback linearization approach is used together with a state feedback to arbitrarily assign the poles of the linearized system.

A simulation is carried out on a feedback-linearized robot arm. A BBO optimization algorithm is proposed to adjust coefficients of the FOILC. The simulation result illustrates that optimal tuning of FOILC coefficients increases the convergence speed. The simulation results confirm the improvement of convergence speed for both types of the proposed PD$^{\alpha}$ and D$^{\alpha}$ types ILC learning laws.

REFERENCES

[1] S. Arimoto, S. Kawamura, and F. Miyazaki (1984) Bettering operation of robots by learning. Journal of Robotic systems, vol. 1, pp. 123-140.

[2] A. Tayebi (2004) Adaptive iterative learning control for robot manipulators. Automatica, vol. 40, pp. 1195-1203.

[3] J. Xiao-gang and Y. Zhi-ye (2010) Adaptive iterative learning control for robot manipulators. Intelligent Computing and Intelligent Systems (ICIS), IEEE International Conference. pp. 139-142.

[4] J.-X. Xu and Y. Tan (2003) Linear and nonlinear iterative learning control. vol. 291: Springer Berlin.

[5] D. Owens and G. Munde (2000) Error convergence in an adaptive iterative learning controller. International Journal of Control, vol. 73, pp. 851-857.

[6] K. L. Moore, Y. Chen, and V. Bahl (2005) Monotonically convergent iterative learning control for linear discrete-time systems. Automatica, vol. 41, pp. 1529-1537.

[7] A. Madady (2008) A self-tuning iterative learning controller for time variant systems. Asian Journal of Control, vol. 10, pp. 666-677.

[8] Y. Q. Chen and K. L. Moore (2001) On D α-type iterative learning control. Decision and Control. Proceedings of the 40th IEEE Conference, pp. 4451-4456.

[9] D. del Castillo Negrete (2005) Fractional calculus: basic theory and applications (Part I). Foro-Red-Mat: Revista electrónica de contenido matemático, vol. 16, p. 1.

[10] G. Chen and E. G. Friedman (2005) An RLC interconnect model based on Fourier analysis. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 24, pp. 170-183.

[11] M. R. Faeghi and H. Delavari (2012) Chaos in fractional-order Genesio–Tesi system and its synchronization. Communications in Nonlinear Science and Numerical Simulation, vol. 17, pp. 731-741.

[12] R. L. Bagley and R. Calico (1991) Fractional order state equations for the control of viscoelastically damped structures. Journal of Guidance, Control, and Dynamics, vol. 14, pp. 304-311.

[13] M. Ichise, Y. Nagayanagi, and T. Kojima (1971) An analog simulation of non-integer order transfer functions for analysis of electrode processes. Journal of Electroanalytical Chemistry and Interfacial Electrochemistry, vol. 33, pp. 253-265.

[14] D. Kusnezov, A. Bulgac, and G. Do Dang (1999) Quantum Levy processes and fractional kinetics," Physical review letters, vol. 82, p. 1136.

[15] A. Oustaloup, (1988) From fractality to non-integer derivation: A fundamental idea for a new process control strategy. Analysis and optimization of systems, ed: Springer, pp. 53-64.

[16] A. Oustaloup, X. Moreau, and M. Nouillant (1996) The CRONE suspension. Control Engineering Practice, vol. 4, pp. 1101-1108.

[17] A. Oustaloup, J. Sabatier, and P. Lanusse (1999) From fractal robustness to CRONE control. Fractional Calculus & Applied Analysis, vol. 2, pp. 1-30.

[18] I. Polubny (1999) Fractional-order systems and PIλDμ controller. IEEE Transactions on Automatic Control, vol. 44, pp. 208-214.

[19] D. Simon (2008) Biogeography-based optimization. Evolutionary Computation, IEEE Transactions on, vol. 12, pp. 702-713.

[20] D. Simon (2011) A probabilistic analysis of a simplified biogeography-based optimization algorithm. Evolutionary computation, vol. 19, pp. 167-188.

[21] M. Ergezer and D. Simon (2011) Oppositional biogeography-based optimization for combinatorial problems. Evolutionary Computation (CEC). IEEE Congress, pp. 1496-1503.

[22] N. Jerath (2012) Implementation of Biogeography Based Optimization on Image Restoration. IJCAIT, vol. 1, pp. 76-80.

[23] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang (2012) Dynamic deployment of wireless sensor networks by biogeography based optimization algorithm. Journal of Sensor and Actuator Networks, vol. 1, pp. 86-96.

[24] A. Bhattacharya and P. K. Chattopadhyay (2010) Biogeography-based optimization for different economic load dispatch problems. Power Systems, IEEE Transactions on, vol. 25, pp. 1064-1077.

[25] A. Bhattacharya and P. Chattopadhyay (2011) Application of biogeography-based optimisation to solve different optimal power flow problems. IET generation, transmission & distribution, vol. 5, pp. 70-80.

[26] K. B. Oldham (1974) The fractional calculus: Elsevier.

[27] J.-X. Xu, T. Heng Lee, and H.-W. Zhang (2004) Analysis and comparison of iterative learning control schemes. Engineering Applications of Artificial Intelligence, vol. 17, pp. 675-686.

[28] P. I. Corke and B. Armstrong-Hélouvry (1995) A meta-study of PUMA 560 dynamics: A critical appraisal of literature data. Robotica, vol. 13, pp. 253-258.

[29] J.-J. E. Slotine and W. Li (1991) Applied nonlinear control. vol. 199: Prentice-Hall Englewood Cliffs, NJ.

[30] H. Ma and D. Simon (2011) Blended biogeography-based optimization for constrained optimization. Engineering Applications of Artificial Intelligence, vol. 24, pp. 517-525.

[31] M. Mittal (2013) Comparison between BBO and Genetic Algorithm. International Journal of Science, Engineering and Technology Research, vol. 2, pp. pp: 284-293.

[32] Y. Chen and C. Wen (1999) Iterative learning control: convergence, robustness and applications. Springer-Verlag.