ADAPTIVE ITERATIVE LEARNING CONTROL OF ROBOTIC SYSTEM BASED ON PARTICLE SWARM OPTIMIZATION

Živković LJ. Nikola[1], Lazarević P. Mihailo[2], Petrović M. Milica[2]

[1] Lola Institute

Kneza Višeslava 70a, 11030 Belgrade, Serbia

e-mail: nikola.zivkovic@li.rs

[2] Faculty of Mechanical Engineering,

The University of Belgrade, Kraljice Marije 16, 11120 Belgrade 35

e-mail: mlazarevic@mas.bg.ac.rs, mmpetrovic@mas.bg.ac.rs

Abstract:

In this paper, an adaptive iterative learning control algorithm for robotic manipulators is proposed. A simplified robot manipulator model with 3 degrees of freedom is used as control object for verification purposes. The mathematical model is obtained via Rodriguez approach for modeling differential equations of motion for multi-body systems. The model itself is a simple open-chain kinematic structure. The proposed control system design consists of two layers of controllers. In the inner loop, feedback linearization is applied to deal with the model nonlinearities. Post feedback linearization advanced iterative learning control (ILC) algorithm of sign-D (signum-Derivative) type is introduced as feed-forward compensation with classical PD (Proportional-Derivative) controller in feedback closed loop. A particle swarm optimization (PSO) algorithm is used to optimize ILC gain parameters while gains for PD controller are set by trial and error. Suitable cost function based on position error is chosen for PSO algorithm in order to ensure convergence. Numerical simulation is carried out in two cases – case with constant learning gains and case with PSO optimized learning gains. It is observed that the proposed control law converges to some steady-state error value in both cases.

Key words: robot dynamics, feedback linearization, iterative learning control, PSO optimization, control design.

1. Introduction

Iterative Learning Control (ILC) as a concept has been introduced by Uchiyama in [1] and Arimoto in [2]. Since then, ILC has gained a lot of interest in the scientific community especially in applications to robotics and batch processes. Iterative learning control is a memory type of control algorithm based on previous knowledge about the system. ILC can be also defined as an intelligent control methodology. Iterative learning control seeks to improve the transient performance of the system that operates over a fixed period of time repetitively. The key idea of ILC is learning through a predetermined hardware repetition. That means that some postulates need to be established for ILC beforehand. This is analogous to the human learning process about its environment through experience. A person's current actions in a certain environment are based on the experience of that same or similar environment from the past stored in memory and evaluation of current circumstances. ILC algorithm drives machine to 'learn' from stored data similarly [3]. The basic ILC algorithm uses input and error information from the previous iteration and some corrective terms to construct current system input.

Industrial or rehabilitative robotics, given their operations and tasks repetitive nature, are suitable for the implementation of ILC type of control schemes. In robotics, it is used for trajectory tracking performance improvement with or without some other type of control algorithm. Repetitive tasks in an industrial environment require repeating one operation over and over again during work hours so some sort of self-tuning control scheme is desirable [4]. The main advantage of iterative learning control is robustness towards uncertainty and disturbances. System dynamics of complex systems such as robot arms cannot be modelled without some degree of parameter uncertainty [5]. Various tasks also require disturbance rejection capabilities, like pick and place tasks where mass of picked object might vary. Robotic systems are non-linear systems and as such require as close as possible model dynamics knowledge in order to design control law for trajectory tracking. ILC comes as a promising solution for this problem because for error to converge to zero, the system's exact model is not necessarily needed to be known. Although the convergence of the error towards zero or some other acceptable steady-state value is ensured through the iterative process, it is also possible to have a large trajectory error in initial iterations before convergence occurs. This is unacceptable for purposes/applications where high precision is needed [6]. This happens mostly due to ILC being open-loop control law without real-time feedback information. One possible/promising solution for the aforementioned problem is to design the ILC algorithm to work partially offline and partially online or to add feedback-based controller besides the ILC one.

In this paper, such possibility is investigated. A fully open-loop ILC controller is applied in parallel with the classic feedback controller as a second level of control. For the first level, feedback linearizing control law is applied to cancel out known nonlinearities, leaving system uncertainties. The main idea of ILC control law is to deal with these uncertainties. Controllers are nowadays almost exclusively digital so computing time should be considered. The ILC algorithm proposed here is inspired by algorithm proposed in [7] and the basic ILC algorithm is proposed in [2]. To make it more robust to uncertainties, Particle Swarm Optimization (PSO) algorithm is introduced to optimize learning gain for the next trial. The experimental results show that PSO algorithm is a computationally inexpensive algorithm suitable for real-time implementations in robotics.

2. Particle Swarm Optimization (PSO) algorithm description

PSO (Particle Swarm Optimization) algorithm is a stochastic optimization method for continuous nonlinear functions. It was discovered through simulation of bird social behavior. The algorithm itself is mathematically very simple and computationally inexpensive. Moreover, PSO is an iterative and population-based algorithm where the initial population is randomly chosen. Each particle's position in the search space is calculated based on previous position and randomized velocity which drives search towards the optimum position. Previous particle positions are stored in memory and compared to current positions in order to determine the best position for each particle and the best position among the entire population.

Since its discovery, the PSO algorithm received a multitude of changes, improvements and adaptations. Initial versions of PSO algorithm belong to the same basic concept. First is the GBEST model, or the original form of this algorithm, and the second is the LBEST model. GBEST model is based on evaluating each particle's best position, comparing it to the previous best position, and finding the global best position. On the other hand, the LBEST model represents/is the local version of the GBEST model. In LBEST each particle has knowledge of its own best position and two nearest neighboring particles instead of the entire population. Particle neighborhood can be minimal consisting of only two adjacent particles or can be larger. In contrast to the GBEST model, the LBEST model showed robustness to falling into local optima, although its convergence is slower comparing with the GBEST model. [8-9].

PSO algorithm can be mathematically described as follows. Let the population be the size of N particles. Each particle is associated with position vector $X_n = (X_{n,1}, X_{n,2}, \dots, X_{n,D})$, velocity vector $V_n = (V_{n,1}, V_{n,2}, \dots, V_{n,D})$, best position vector $Pbest_n = (P_{n,1}, P_{n,2}, \dots, P_{n,D})$ in D-dimensional space. Also, there is a group of particles with best position vector $Gbest = (G_1, G_2, \dots, G_D)$. Best positions are evaluated through suitable fitness function minimization. Particles individual best position is updated via the following expression:

$$Pbest_n^{k+1} = \begin{cases} X_n^{k+1}, \text{ if } f(X_n^{k+1}) < f(Pbest_n^k) \\ Pbest_n^k, \text{ otherwise} \end{cases} \tag{1}$$

Group's optimal position is best position of all individual positions for the current iteration. Velocity and position vectors are calculated via the following equations:

$$V_n^{k+1} = V_n^k + c_1 * rand * \left(Pbest_n^k - X_n^k\right) + c_2 * rand * \left(Gbest_n^k - X_n^k\right) \tag{2}$$

$$X_n^{k+1} = X_n^k + V_n^{k+1} \tag{3}$$

where k is iteration index, n is population index, c_1 and c_2 are acceleration constants and $rand$ is random uniform number in interval [0,1]. The PSO search mechanism based on the previous equations (2) and (3) can be depicted as it is shown in Fig. 1. The original formulation of the PSO showed some shortcomings particularly in terms of the algorithm's convergence characteristics. Therefore, various improved versions of the algorithm were proposed in the literature. Following modified version is used in this paper:

$$V_n^{k+1} = \omega * V_n^k + c_1 * rand * \left(Pbest_n^k - X_n^k\right) + c_2 * rand * \left(Gbest_n^k - X_n^k\right) \tag{4}$$

where ω is is called inertia weight. Introducing this parameter, performance of the PSO

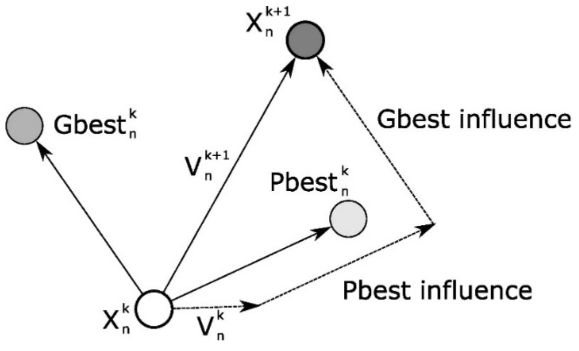

Fig. 1. PSO algorithm search scheme

algorithm in optimization problems is greatly improved, without increasing algorithms complexity. This modified version (4) is generally called the canonical PSO algorithm while (2) is called the original PSO algorithm [10].

3. Control object model

A robot manipulator with 3 degrees of freedom is used as a control object model. The mathematical model is obtained via Rodrigues approach [11-12]. A robot manipulator with open-chain kinematic structure is considered.

The robot manipulator model is represented as a sequence of three rigid bodies – links, interconnected with joints. All three joints are kinematic pairs of the fifth class, which means that each joint allows movement only along one degree of freedom. All three joints are rotational.

Links are adopted as homogeneous truncated cones with lengths l_1, l_2, l_3 respectively. Generalized coordinates associated with each joint are adopted as $q^i, i = 1,2,3$. Inertial reference frame $Oxyz$ origin is set on the axis of rotation of the first joint (see Fig. 2.). Each link has a local reference frame $C\xi_i\eta_i\zeta_i$ attached to it, with origin in the center of inertia. Reference configuration for robot manipulator is achieved when corresponding axes $\xi_i\eta_i\zeta_i$ are parallel with inertial reference frame $Oxyz$ axes at the initial time. Robot manipulator must be in reference configuration for Rodriguez approach to be applied.

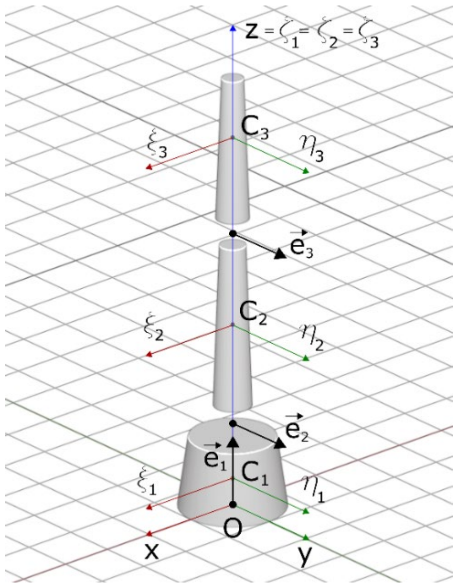

Fig. 2. Structure of the robot model

Parameters ξ_i, $\bar{\xi}_i = 1 - \xi_i$ are defined, which denote whether the joint is prismatic or cylindrical. Rotational axes of the model are defined by the unit vectors $\vec{e}_i$. The geometry of the links is defined by the position vectors $\vec{\rho}_i$ and $\vec{\rho}_{ii}$ expressed in local coordinate frames $C_i\xi_i\eta_i\zeta_i$. Differential equations of motion for the robot manipulator can be now obtained by applying Lagrange equations of the second kind in the covariant form as follows:

$$\sum_{\alpha=1}^{n} a_{\gamma\alpha}(q)\ddot{q}_\alpha + \sum_{\alpha=1}^{n}\sum_{\beta=1}^{n} \Gamma_{\alpha\beta,\gamma}(q)\dot{q}_\alpha\dot{q}_\beta = Q_\gamma,\ \ \gamma = 1,2,..n \tag{5}$$

where the coefficients $a_{\gamma\alpha}$ are the covariant coordinates of the basic metric tensor and $\Gamma_{\alpha\beta,\gamma}(q)$ are Christoffel symbols of the first kind. The coefficients $a_{\alpha\beta}$ of metric tensor are defined as:

$$a_{\alpha\beta} = \sum_{i=1}^{n} m_i\big(\vec{T}_{\alpha(i)}\big)\{\vec{T}_{\beta(i)}\} + \big(\vec{\Omega}_{\alpha(i)}\big)[J_{Ci}]\{\vec{\Omega}_{\beta(i)}\} \tag{6}$$

where translational $\vec{T}_{\alpha(i)}$ and rotational $\vec{\Omega}_{\alpha(i)}$ quasi-base vectors are defined as:

$$\vec{T}_{\alpha(i)} = \begin{cases} \bar{\xi}_\alpha\vec{e}_\alpha \times \vec{R}_{\alpha(i)} + \xi_\alpha\vec{e}_\alpha, \forall\alpha \le i \\ 0, \forall\alpha > i \end{cases}, \vec{\Omega}_{\alpha(i)} = \begin{cases} \bar{\xi}_\alpha\vec{e}_\alpha,\ \forall\alpha \le i \\ 0,\ \forall\alpha > i \end{cases} \tag{7}$$

as well as $\vec{R}_{\alpha(i)} = \sum_{k=\alpha}^{i}\big(\vec{\rho}_{kk} + \xi_k\vec{e}_k q^k\big) + \vec{\rho}_i$. Christoffel symbols are defined as:

$$\Gamma_{\alpha\beta,\gamma} = \frac{1}{2}\left(\frac{\partial a_{\beta\gamma}}{\partial q^\alpha} + \frac{\partial a_{\gamma\alpha}}{\partial q^\beta} - \frac{\partial a_{\alpha\beta}}{\partial q^\gamma}\right), \alpha, \beta, \gamma = 1, \dots, n \tag{8}$$

Generalized forces in our case can be written as:

$$Q_\gamma = Q_\gamma^a + Q_\gamma^g \tag{9}$$

where Q_γ^a, Q_γ^g are generalized control and gravitational forces, respectively.

Differential equations of motion of robot manipulator now can be presented in compact matrix form:

$$a(q)\ddot{q} + (K(q,\dot{q}) - Q^g) = a(q)\ddot{q} + c(q,\dot{q}) = Q^u \tag{10}$$

where $a(q)$ is the inertia matrix, $K(q,\dot{q})$ is matrix that includes centrifugal and Coriolis effects, Q^g is vector of gravitational forces and Q^u is vector of generalized control force.

4. Control Design

The robotic multi-body system is MIMO (Multi-Input Multi-Output) system and it is an inherently nonlinear time-varying system. To cope with nonlinearities, proposed control law is designed in two layers – feedback linearizing loop and PSO enhanced ILC algorithm with a classical PD feedback loop.

4.1 Feedback linearization

Feedback linearization is a model-based control algorithm applied to partially or completely mitigate nonlinearities. Feedback linearization in general is exact linearization [13]. Here, a real robotic manipulator model can be treated as a two-part model, with a nominal part and an uncertain part. Feedback linearizing control law is applied to cancel the nominal part of the model, and leave an uncertain part for the next stage PSO-ILC-PD controller to deal with it. The uncertain part is added to simulate real-world uncertainty, which arises due to the idealization and simplification of the model and simply because some of the parameters are unknown. Uncertainty of the given model can be represented, at the simplest, as an additive uncertainty [14]. Changing parameter is the mass of the object. With that in mind, inertia matrix, centrifugal matrix, and gravity vector can be written as a sum of nominal and uncertain part [15]. Given the equations of motion,

$$(a_N(q) + \Delta a)\ddot{q} + (K_N(q,\dot{q}) + \Delta K) - (Q_N^g + \Delta Q^g) = Q^u \tag{11}$$

input vector Q^u can be calculated as:

$$Q^u = a_N(q)u + K_N(q,\dot{q}) - \ Q_N^g \tag{12}$$

Inserting equation (12) in equation (11), new linear decoupled system is obtained:

$$\ddot{q}(t) = a^{-1}(q)a_N(q)u(t) + \eta(q,\dot{q}) \tag{13}$$

where $\eta(q,\dot{q}) = -a^{-1}(q)(\Delta K(q,\dot{q}) - \Delta Q^g)$. Now u can be chosen as new control law, in our case ILC/PD type.

Block diagram of first layer of control system is shown in next figure (Fig. 3.)

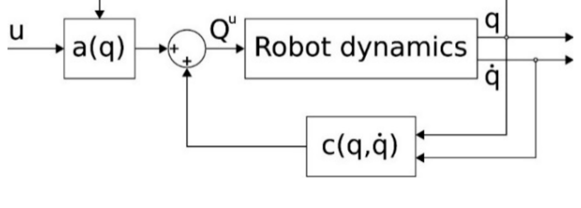

Fig. 3. Block diagram of feedback linearization control law

4.2 State space representation

After applied feedback linearization, equations of motion now can be represented in the state-space form:

$$\dot{x}(t) = Ax(t) + Bu(t) + D\eta(q,\dot{q}), \tag{14}$$

$$y(t) = Cx(t). \tag{15}$$

where

$$A = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ 0_{n\times n} & 0_{n\times n} \end{bmatrix}, B_a = \begin{bmatrix} 0_{n\times n} \\ a^{-1}a_N \end{bmatrix}, D = \begin{bmatrix} 0_{n\times n} \\ I_{n\times n} \end{bmatrix} \tag{16}$$

$$C = [I_{n\times n} \quad 0_{n\times n}]. \tag{17}$$

Before applying proposed ILC algorithm as input u, the following assumptions will be made:

I. Desired trajectories $q_d(t)$ are continuously differentiable on $[0, T]$,

II. Initial conditions for all iterations are $x_k(0) = x_d(0), k = 1,2,\dots,n$

III. Influence of changing the masses of links is negligible on matrix $a(q)$, so it follows that $a^{-1}a_N \approx I$, and $B_a = B = \begin{bmatrix} 0_{n\times n} \\ I_{n\times n} \end{bmatrix}$,

IV. System (14) and (15) is causal [15].

4.3 PSO-ILC-PD controller

The proposed linear control law should ensure trajectory tracking and robustness to uncertainties. After closing the linearizing loop, an open-closed loop ILC algorithm is introduced which consists of feed-forward sign-D type control law and feedback PD type control law as represented in the block diagram (Fig. 4.). In [7] sign-P type ILC control law with self-adapting steps is exploited for linear time-invariant (LTI) systems control suitable for implementation on microcontrollers due to absence of derivative calculation. It is observed in [7] and [16] that error convergence is ensured but learning gain matrix M is tiresome and time-consuming to tune by hand and error convergence in presence of uncertainties is achieved after 100 or more iterations. ILC algorithm is enhanced with PSO algorithm after every iteration making self-adaptation proposed in [7], optimal and somewhat continuous instead of the limited number of fixed gain parameters M. PSO algorithm optimizes learning gain matrix M thus ensuring error convergence along iteration axis and controller robustness. This also removes a large portion of time spent tuning M by trial and error. The simplicity of the PSO algorithm makes it a good choice for implementation on a digital Programmable Logic Controllers (PLC) in comparison to other optimization methods. Classical PD feedback stabilizes systems response along the time axis.

For the PSO algorithm to function properly, a suitable cost function should be chosen. Good cost function will drive the search process towards a global minimum. Cost function for norm optimal ILC (NOILC) is chosen [17]:

$$J^{k+1}(u^{k+1}) = \left\|u^{k+1} - u^k\right\|^2 + \left\|e^{k+1}\right\|^2 \tag{18}$$

This cost function penalizes input difference between iterations and ensures that position error is always small.

Control law can be partitioned to feed-forward $u^i_{ILC}(t)$ consisting of ILC controller and feedback $u^i_{PD}(t)$ part consisting of classical PD type control law. Now, the second layer of our system controller can be written as:

$$u^{k+1}(t) = u^{k}_{ILC}(t) + u^{k}_{PD}(t) = u^{k-1}_{ILC} + Msgn(\dot{e}^{k-1}) + K_P e^k + K_D \dot{e}^k \tag{19}$$

where k denotes ILC iteration index (not PSO iteration index), $e^k = y_d - y^k, \dot{e}^k = \dot{y}_d - \dot{y}^k$ are position and velocity errors respectively, M, Kp, and Kv are positive-definite diagonal gain matrices. Signum function is defined as:

$$sgn(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{20}$$

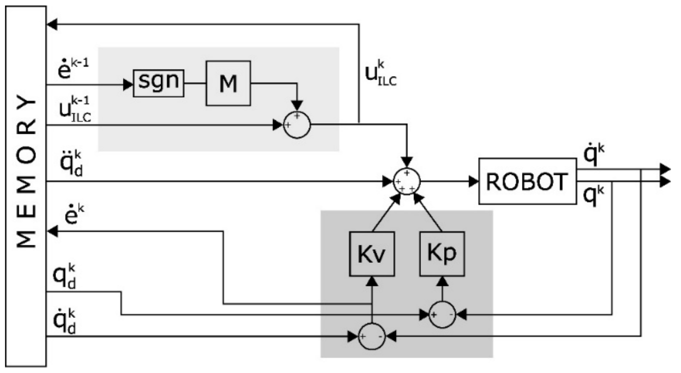

Fig. 4. Block diagram of the system

5. Numerical simulation

Proposed control law verification is carried out through numerical simulation in Matlab using ode45(.) function for solving ordinary differential equations. The robot manipulator is tasked with desired trajectory tracking, $q_d(t) \in \mathbb{R}^n$, and the maximum absolute error for each joint is set as $|e_{bound}| = 0.008\ [rad]$. Actual joint trajectories are observed over time interval $t = [0, T]$ where $T = 5s$. Desired trajectories are given as fifth-order polynomials in joint space coordinates:

$$q_d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{21}$$

with constraints showed in Table 1:

k	1	2	3
$q_{dk}(0)[rad]$	0	0	0
$q_{dk}(T)[rad]$	1.5708	0.7854	0.5236
$\dot{q}_{dk}(0)[rad/s]$	0	0	0
$\dot{q}_{dk}(T)[rad/s]$	0	0	0
$\ddot{q}_{dk}(0)[rad/s^2]$	0	0	0
$\ddot{q}_{dk}(T)[rad/s^2]$	0	0	0

Table 1. Constraints for desired trajectories, velocities and accelerations of joints

Uncertainty parameter in our case is robot link mass change $\Delta m_i = 0.15 m_i, i = 1,2,3$. Matrices Kp and Kv are set to be $Kp = diag(10)$ and $Kv = diag(6.3)$. PSO algorithm task is to optimize learning gain diagonal matrix *M*, so as a variable it is adopted 3-dimensional vector which is a vector of diagonal elements of the same matrix. That means PSO algorithm will search in 3-dimensional space. Parameters of the PSO algorithm are set as follows:

Unknown variable X_n	Learning gain matrix M diagonal
Variable X_n limits	$X_1 \in [0,1], X_2 \in [0,0.5], X_3 \in [0,0.3]$
Maximum number of iterations	20

Population size	30
Inertia weight	0.9
Personal acceleration coefficient	1.9
Social acceleration coefficient	2

Table 2. PSO algorithm parameters

Guidelines for PSO parameters choice can be found in [18]. In our case, variable limits play important role in error convergence. It is observed that better results are achieved when learning gain diagonal matrix M elements are in intervals from zero to one. The more interval limits grow the more error is getting bigger.

Results of simulation show that the proposed PSO-ILC-PD control law drives error value below proposed boundary value $|e_{bound}| = 0.008\ [rad]$. Convergence occurs around the 15th iteration for all three joints (Fig. 5.). In Fig. 6. it can be seen that actual trajectories after 50th iteration achieves good tracking of the reference trajectories. Position errors are depicted in Fig. 7.

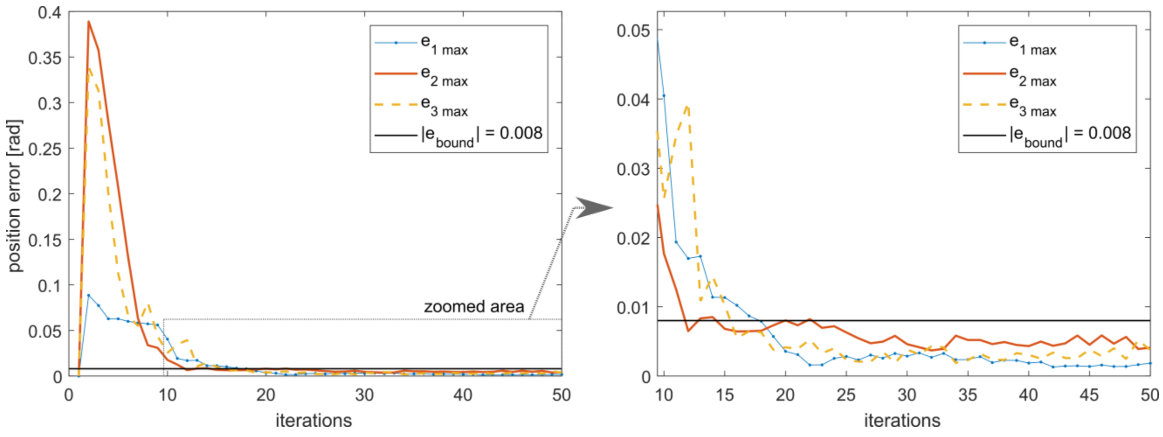

Fig. 5. Maximum position error evolution over iterations

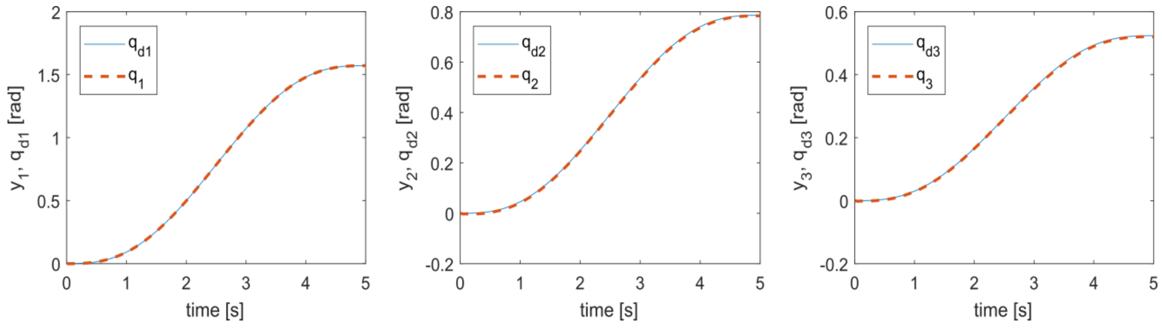

Fig. 6. Desired vs. actual trajectories after 50th iteration

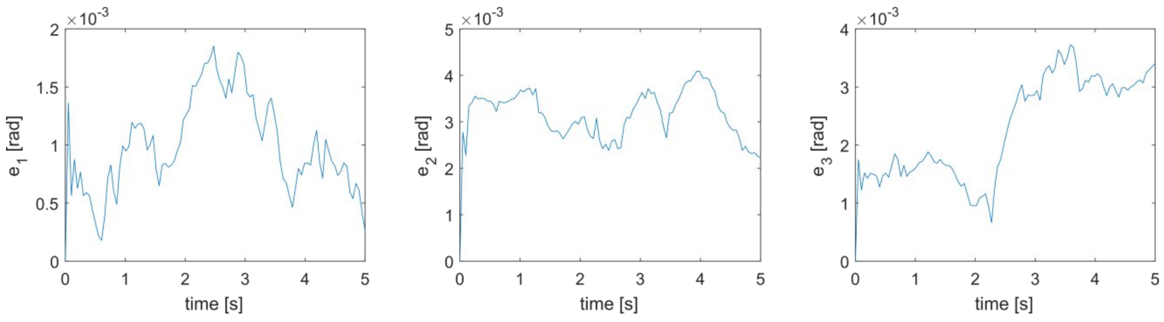

Fig. 7. Position errors over time after 50th iteration

A comparison between PSO-ILC-PD and ILC-PD without PSO is presented. System is tested with constant learning gain matrix $M = diag(0.05, 0.095, 0.06)$, for each iteration. Results show that error convergence in this case occurs slower than in the case with PSO (see Fig. 8). Due to the system being coupled, it is difficult to tune matrix M to get faster error convergence.

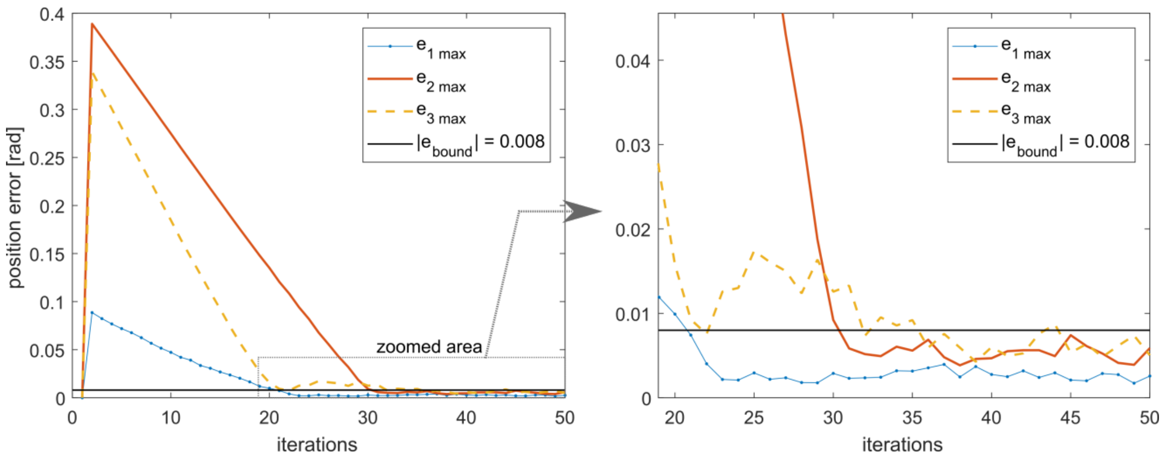

Fig. 8. Maximum position error evolution over iterations without PSO

6. Conclusions

In this paper PSO-ILC-PD type of controller is investigated. Control law behavior is tested, through simulation, on a robotic system with additive bounded uncertainties. The system with PSO optimized learning gain shows faster error convergence than the system without PSO. Both systems converge below proposed error boundary $|e_{bound}| = 0.008\ [rad]$. Proposed ILC control law in combination with Particle Swarm Optimization is suitable to be implemented on Programmable Logic Controllers (PLC) due to their computational simplicity.

Acknowledgment

The presented research was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia by contract no. 451-03-9/2021-14/200105 from 05.02.2021 and contract no. 451-03-9/2021-14/200066.

References

[1] Uchiyama M., *Formulation of high speed motion pattern of mechanical arm by trial (in Japanese)*. Transactions of the Society for Instrumentation and Control Engineers, 14, 706-712, 1978.

[2] Arimoto S., Kawamura S., and Miyazaki F., *Bettering operation of robots by learning*, Journal of Robot System, 1, 123-140, 1984.
[3] Ahn H.S., Moore K.L., Chen Y., *Iterative Learning Control: Robustness and Monotonic Convergence for Interval Systems*. London: Springer-Verlag, 2007.
[4] Ouyang P.R., and Pipatpaibul P., *ITERATIVE LEARNING CONTROL: A COMPARISON STUDY*, Proceedings of the ASME 2010 International Mechanical Engineering Congress & Exposition, 2010.
[5] Ahn H.S., Chen Y., Moore K.L., *Iterative Learning Control: Brief Survey and Categorization*, IEEE Transactions on Systems, Man, and Cybernetics—PART C: Applications and Reviews, Vol. 37, No. 6, November 2007.
[6] Jang T-J., Choi C.H., and Ahn H.S., *Iterative Learning Control in Feedback Systems*, Automatica. Vol. 31, No. 2, pp. 243-248, 1995.
[7] Wang Y., *Iterative Learning Control Algorithm with Self-adaptive Steps,* Proceedings of the 10th World Congress on Intelligent Control and Automation*,* Jul. 2012.
[8] Kennedy J., and Eberhart R.*, "Particle swarm optimization,* Proceedings of ICNN'95 - International Conference on Neural Networks, vol.4, pp. 1942-1948, 1995.
[9] Kennedy J., and Eberhart R.*, A new optimizer using particle swarm* theory, MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995.
[10] Wang D., Dapei Tan D., Lei Liu L., *Particle swarm optimization algorithm: an overview*, *Soft Computing,* vol. 22, pp. 387-408, 2018.
[11] Čović V., Lazarević M., *Robot mechanics*, Faculty of Mechanical Engineering, Belgrade, 2007.
[12] Lazarević M., *Mechanics of Human Locomotor System*, FME Transactions, vol. 34, no. 2, pp. 105–114, 2006.
[13] Slotine J., *Applied nonlinear control*, Englewood Cliffs N.J.: Prentice Hall, 1991.
[14] Astrom K., Murray R., *Feedback Systems: An Introduction for Scientists and Engineers*, Princeton University Press, 2008
[15] Lazarević M., Mandić P., Ostojić S., *Further results on advanced robust iterative learning control and modeling of robotic systems*, Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci., p. 0954406220965996, Nov. 2020, doi: 10.1177/0954406220965996.
[16] Lazarević M., Živković N., *The advanced iterative learning control algorithm for rehabilitation exoskeletons*, Scientific Technical Review, Vol. 70, No 3, pp. 29-34, 2020.
[17] Owens D.H., Hatonen J., *Iterative learning control — An optimization paradigm*, Annual Reviews in Control, Vol. 29, 1, pp. 57-70, 2005.
[18] Engelbrecht A., *Computational Intelligence: An Introduction*, Wiley, 2007.