Robotica (2016) volume 34, pp. 1351–1366. © Cambridge University Press 2014
doi:10.1017/S0263574714002331

Gain tuning of position domain PID control using particle swarm optimization

V. Pano and P. R. Ouyang*

Department of Aerospace Engineering, Ryerson University, Toronto, Canada

(Accepted August 11, 2014. First published online: September 10, 2014)

SUMMARY
Particle swarm optimization (PSO) is a heuristic optimization algorithm and is commonly used for the tuning of PD/PID-type controllers. In this paper, PSO is applied for control gain tuning of a position domain PID controller in order to improve contour tracking performances of linear and nonlinear contours for a serial multi-DOF robotic manipulator. A new fitness function is proposed for gain tuning based on the statistics of the contour error, and pre-existed fitness functions are also applied for the optimization process, followed by some comparison studies. The PSO tuning technique demonstrated the same effectiveness in position domain controllers as in time domain controllers with the results being quite satisfying with low contour errors for both linear and nonlinear contours, and the proposed fitness function is proved to be on par with the pre-existed fitness functions.

KEYWORDS: Position domain control; PID; Particle swarm optimization; Control gain; Contour tracking; Contour error.

1. Introduction

PID control and its simple versions (PI and PD) are the most popular control schemes for controlling robotic manipulators. In fact, with more than 90% of industrial applications incorporating some forms of PID controllers, PID controllers dominate the real applications of feedback control.[1] The main advantages of PD/PID controller are its simple structure, linear nature, and ease of implementation. In spite of its simplicity, PID control is powerful enough to achieve acceptable tracking performance and deal with practical issues such as actuator saturation and integrator windup.

Furthermore, a lot of tuning techniques have been developed for PID controllers. Several methods, such as Ziegler–Nichols (ZN), D-partitioning, and pole placement, have been developed to tune control gains of PID controllers.[2] However, the majority of these techniques were developed for linear and time-invariant systems and require excessive knowledge of the controlled system's frequency response. For this reason, heuristic algorithms have been used for the tuning of PID controllers for nonlinear systems. From various heuristic techniques such as genetic algorithm (GA), evolutionary programming (EP), colony optimization (ACO), differential evolution (DE), and particle swarm optimization (PSO), the last one has been proven to be simple and efficient.[3]

Inspired by social behaviors of bird flocking and fish schooling, PSO was proposed as a population based meta-heuristic optimization technique for the solution of various nonlinear functions and optimization problems.[4] Over the years, PSO has been demonstrated to be quite popular in many optimization applications mainly due to its simplicity, robustness, and high dimensionality.[5] PSO was used for the optimum design of a PID controller in a linear brushless DC motion.[6] Similarly, variations of PSO were used for tuning purposes in different applications such as high performance drilling,[7] control of automatic gantry crane,[8] steam temperature control in boilers,[9] and control of autonomous underwater vehicles.[10] Furthermore, PSO has been used for control gain tuning of more advanced PID-type controllers such as fuzzy PID,[11] H_∞ PID,[12] and robust PID.[13]

In our previous research,[14–17] the concept of position domain control (PDC) was introduced in the form of PD and PID control. The main ideas behind PDC are the separation of a system into a master

* Corresponding author. E-mail: pouyang@ryerson.ca

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

motion and slave motions and the use of the master motion as a reference for the slave motions. To reach such a purpose, the dynamics of the slave motions is transformed from time domain to position domain via a one-to-one mapping[16] so that the features of the dynamic system in time domain are inherited by the PDC in position domain.

It should be mentioned that, in our previous studies, the selection of control gains for the position domain PID controller (PDC-PID) was based on a trial-and-error method that cannot obtain the best control performance. The main objective in this research is to find an optimization technique for gain tuning of PDC-PID to further improve contour tracking performances, and PSO is used for control gain tuning of PDC-PID responsible for contour tracking of the end-effector of a 3R planar serial robotic manipulator. The control gains are optimized for linear and nonlinear contours, and a new fitness function is proposed for gain tuning based on the statistics of contour errors. Also, some existed fitness functions are used as the performance criteria and comparison study is conducted.

The rest of this paper is organized as follows: Section 2 synopsizes the PDC-PID. Section 3 provides a summary of the PSO algorithm, and Section 4 shows the details of the optimization process for the PDC-PID. Section 5 provides some simulation results. Finally, Section 6 gives some conclusions.

2. Position Domain Control for Contour Tracking

The dynamics of a robotic manipulator can be mathematically expressed by the following equation:[18]

$$M(q)\ddot{q}(t) + C(q,\dot{q})\dot{q} + G(q) + F(t,q,\dot{q}) = \tau(t), \tag{1}$$

where $M(q)$ is the inertial matrix, $C(q,\dot{q})$ is the matrix of coriolis and centrifugal forces, $G(q)$ is the vector of gravitational terms, $F(t,q,\dot{q})$ is the vector of friction forces, $\tau(t)$ is the vector of joint torques, and $q(t)$, $\dot{q}(t)$, and $\ddot{q}(t)$ are the joint position, velocity, and acceleration vectors, respectively. Further information on the dynamic model can be found in the Appendix for a 3R planar serial robotic manipulator.

To facilitate the discussion of PDC, the robotic manipulator is viewed as a master motion and slave motions, and the dynamic model in Eq. (1) can be separated between the master motion and the slave motions by taking the following submatrix form:

$$\begin{bmatrix} m_{mm} & M_{ms} \\ M_{sm} & M_{ss} \end{bmatrix}\begin{bmatrix} \ddot{q}_m \\ \ddot{q}_s \end{bmatrix} + \begin{bmatrix} c_{mm} & C_{ms} \\ C_{sm} & C_{ss} \end{bmatrix}\begin{bmatrix} \dot{q}_m \\ \dot{q}_s \end{bmatrix} + \begin{bmatrix} G_m \\ G_s \end{bmatrix} + \begin{bmatrix} F_m \\ F_s \end{bmatrix} = \begin{bmatrix} \tau_m \\ \tau_s \end{bmatrix} \tag{2}$$

with subscripts m and s indicating the master and the slave motions, respectively.

In order for the master motion to be used as a reference by the slave motions instead of time, the slave section of Eq. (2) has to be expressed as functions of the master motion. This can be done as a one-to-one mapping as described in Section 2.1.

2.1. Relative derivative and position domain mapping

To transform the dynamics of the slave motions from time domain to position domain, a relationship has to be developed that relates time domain to position domain. This relationship is introduced by the relative derivative of the ith slave motion (q_i) with regard to the master motion (q_m):

$$q'_{si} = \frac{\mathrm{d}q_{si}}{\mathrm{d}q_m} = \frac{\dot{q}_{si}}{\dot{q}_m}. \tag{3}$$

Equation (3) indicates that q'_i, also called the relative position velocity, is the ratio of the slave motion speed over the master motion speed, and it describes the synchronized relationship between the master motion and slave motions.

Likewise, the relative position acceleration of the slave motions can be defined as the second relative derivative of Eq. (3):

$$q''_{si} = \frac{\mathrm{d}q'_{si}}{\mathrm{d}q_m}. \tag{4}$$

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Reorganizing Eqs. (3) and (4), the one-to-one transformation from time domain to position domain can be expressed as:

$$\begin{cases} \dot{q}_{si} = \dot{q}_m q'_{si} \\ \ddot{q}_{si} = q''_{si}(\dot{q}_m)^2 + \ddot{q}_m q'_{si} \end{cases}. \tag{5}$$

The above equations indicate the relationship between absolute and relative motions, and relate the absolute velocity and acceleration in time domain with relative position velocity and acceleration in position domain. These two equations in (5) are used to transform the dynamic model from time domain to position domain being discussed in the following subsections.

2.2. Dynamic model in position domain

Substituting Eq. (5) into the slave section of Eq. (2), the dynamic model of the slave motions in position domain is derived as functions of the master motion:

$$\dot{q}_m^2 M_{ss} q''_s(q_m) + (\ddot{q}_m M_{ss} + \dot{q}_m C_{ss}) q'_s(q_m) + \ddot{q}_m M_{sm} + \dot{q}_m M_{sm} + G_s + F_s = \tau_s(q_m). \tag{6}$$

2.3. Position domain PID control law

In a similar manner as the dynamic model, a PDC-PID can be expressed in position domain with tracking error becoming a function of the master motion:

$$\tau_{si}(q_m) = K_{Pi} e_{si}(q_m) + K_{Di} e'_{si}(q_m) + K_{Ii} \int_0^{q_m} e_{si}(s)\,\mathrm{d}s, \tag{7}$$

where K_{Pi}, K_{Di}, and K_{Ii} are the control gains for the ith slave motion. The tracking errors of slave motions are defined as follows:

$$\begin{cases} e_{si}(q_m) = q_{\mathrm{sid}}(q_m) - q_{si}(q_m) \\ e'_{si}(q_m) = q'_{\mathrm{sid}}(q_m) - q'_{si}(q_m) \end{cases}.$$

It should be noted that the position domain controller described in Eq. (7) is used only for the control of all slave motions but not for the master motion. The master motion is controlled by a conventional time domain PID controller described as follows.

$$\tau_m(t) = K_{Pm} e_m(t) + K_{Dm} \dot{e}_m(t) + K_{Im} \int_0^t e_m(s)\,\mathrm{d}s \quad \text{with} \begin{cases} e_m(t) = q_{md}(t) - q_m(t) \\ \dot{e}_m(t) = \dot{q}_{md}(t) - \dot{q}_m(t) \end{cases}.$$

2.4. System stability theorem

Theorem: For a rigid robotic manipulator described in the position domain of Eq. (6), if the position domain synchronization control law in Eq. (7) is applied to control a contour tracking of the robotic manipulator, and the following conditions in Eq. (8) are satisfied, then the controlled robotic manipulator is globally asymptotical stable for contour tracking.

$$\begin{cases} K \succ 0 \\ K_{Ps} \succ 0, K_{Ds} \succ 0 \\ K + \dot{q}_m C_{ss}^T - \ddot{q}_m M_{ss} > 0 \\ \lambda_m(K_{Ps}) > \dot{q}_m^2 \lambda_M(M_{ss}) > 0 \\ \lambda_m(K_{Ps}) > \frac{1}{2}\lambda_M(K + \dot{q}_m C_{ss}^T - \ddot{q}_1 M_{ss}) \\ \lambda_m\left(K_{Ds} + \left(\ddot{q}_m - \dot{q}_m^2\right) M_{ss}\right) > \frac{1}{2}\lambda_M(K + \dot{q}_m C_{ss}^T - \ddot{q}_m M_{ss}) \end{cases}, \tag{8}$$

where K is a user defined matrix, K_{Ps} and K_{Ds} are the control gain matrices with diagonal elements. The proof about the theorem is very similar with the stability analysis in ref. [16].

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

2.5. Trajectory generation in position domain

To generate the trajectory of the master motion in time domain, a fifth-degree polynomial is used and defined as

$$r(t) = 10\left(\frac{t}{T}\right)^3 - 15\left(\frac{t}{T}\right)^4 + 6\left(\frac{t}{T}\right)^5, \tag{9}$$

where t is time and T is the total time required for the master motion. The linear and nonlinear trajectories are then defined parametrically as functions of $r(t)$.

It should be noted that the controller in Eq. (7) requires for the errors of the slave motions to be functions of the master motion, implying that the trajectories of the slave motions must be defined as functions of the master motion. This means that Eq. (9) cannot be solely used for the generation of the slave trajectories.

To solve this problem, a small procedure is followed in the simulation. First, the desired time domain trajectories for the slave motions are defined based on the inverse kinematics and the master motion. Then the position domain trajectories are calculated by interpolating and using the master motion as a reference, via lookup tables, the desired time domain trajectories with respect to the actual trajectory of the master motion. Consequently, the slave trajectory generation becomes highly dependent on the master motion.

3. Particle Swarm Optimization

3.1. Introduction of PSO

In the PSO, *pbest* represents the personal best position that a particle had achieved throughout the iterations, and *gbest* represents the best position achieved by the whole swarm. Both *pbest* and *gbest* do not change at every iteration but change only if the swarm or particles have achieved better positions in the current iteration than the stored *gbest* and *pbest* recorded from previous iterations. *gbest* and *pbest* are of high importance for the optimization since they affect the velocity of each particle and therefore, the convergence of the optimization.[3]

For every iteration, the velocity and position of each particle are calculated as follows:

$$v_i^{k+1} = \phi^k v_i^k + \alpha_1 \gamma_{1,i}\left(pbest_i - x_i^k\right) + \alpha_2 \gamma_{2,i}\left(gbest_i - x_i^k\right), \tag{10}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}, \tag{11}$$

where x_i^k and v_i^k are the position and velocity of the ith particle in the kth iteration, $\gamma_{1,i}$ and $\gamma_{2,i}$ are uniformly distributed parameters of the ith particle, and α_1 and α_2 are acceleration constants.

The acceleration parameters α_1 and α_2 in Eq. (10) are quite important for the convergence of the optimization. α_1 is the swarm's cognitive acceleration parameter that determines the "confidence" of each particle in itself, whereas α_2 is the social acceleration parameter of the swarm, determining the influence of the *gbest* position on the position of every particle.[19]

Furthermore, parameter ϕ represents the particle inertia that produces a certain momentum for the swarm and is defined as

$$\phi^k = \frac{\phi_b - \phi_a}{K} \cdot (k-1) + \phi_a, \tag{12}$$

where K is the maximum number of iterations, and ϕ_a and ϕ_b are the minimum and maximum inertia constants defined as[20]

$$\begin{cases} \phi_a = 1 - \varepsilon \\ \phi_b = \dfrac{\alpha_1 + \alpha_2}{2} - 1 + \varepsilon \end{cases}, \ \varepsilon < 1\ . \tag{13}$$

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Table I. Structural parameters of a serial robotic manipulator.

Link	Mass m_i (kg)	Length l_i (m)	Center r_i (m)	Inertia I_i (kg m^2)
1	1.00	0.50	0.25	0.10
2	1.00	0.50	0.25	0.10
3	0.50	0.30	0.15	0.05

It should be noted that necessary and sufficient conditions for the optimization's stability are given by the following two inequalities:[20]

$$\begin{cases} \alpha_1 + \alpha_2 < 4 \\ \dfrac{\alpha_1 + \alpha_2}{2} - 1 < \phi < 1 \end{cases}. \tag{14}$$

Equation (14) illustrates the importance of the two acceleration parameters. To evaluate the effect of these two parameters on the optimization, a performance analysis is conducted in Section 4.5.

3.2. Position domain controller gain optimization algorithm

As in similar optimization procedures, the particles of the swarm are nothing but sets of gains for the controller in Eq. (7) for the gain tuning of PDC-PID. The main objective of PDC is contour tracking performance improvement of robotic manipulators; hence, the optimization objective is the minimums of contour errors (e_c). Unlike a linear system, the performance of each particle cannot be calculated using the frequency response of the system. So, the motion of the robotic manipulator for each particle has to be simulated and the resulting contour errors are calculated.

A fitness function is needed in order to convert contour errors from a vector to a singular arithmetic value that can be used for optimization purposes. In this paper, three different fitness (criteria) functions are introduced and used. Two of them, the integral of absolute error (IAE) in Eq. (15) and the integral of the square of error (ISE) in Eq. (16), are existed fitness functions and are widely used in gain tuning via optimization.[3,21]

$$\text{IAE} = \int_0^\infty |e_c| \, \mathrm{d}t, \tag{15}$$

$$\text{ISE} = \int_0^\infty e_c^2 \mathrm{d}t. \tag{16}$$

The third one, called mean and standard deviation of absolute error (MSMAE) in Eq. (17), is proposed in this paper in order to investigate its practicality as an alternative to the existed fitness functions.

$$\text{MSMAE} = \overline{|e_c|} + \sigma\left(|e_c|\right) + \max\left(|e_c|\right), \tag{17}$$

where $\overline{|e_c|}$, $\sigma(|e_c|)$, and $\max(|e_c|)$ are the mean, standard derivation, and maximum value of the absolute contour error that will be defined in the next section.

Figure 1 shows an optimization flowchart that is used to determine the control gains of the PDC-PID.

4. Optimization Process

4.1. Dynamic model

For the purposes of simulations, a 3-DOF planar robotic manipulator, setting on a vertical plane, is used to track linear and nonlinear contours. The robotic manipulator consists of three revolute joints, and the structural properties[16,22] of the robot are listed in Table I.

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Table II. Friction parameters.

Parameter	Value
Coulomb friction (f_c)	3
Stribeck friction (f_t)	100
Static friction (f_s)	5
Viscous friction (b)	1.5
δ	2

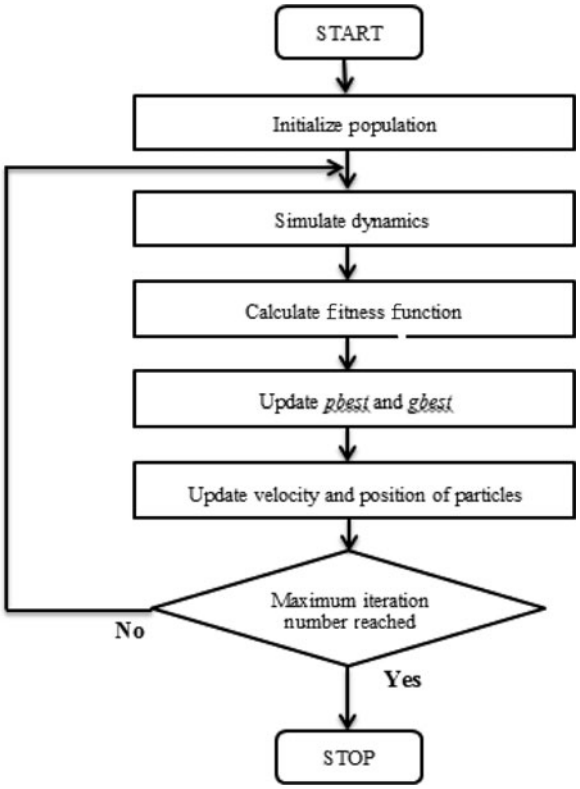


Fig. 1. Particle swarm optimization flowchart.

4.2. Friction model

The friction on the revolute joints is modeled based on the following equation:

$$\mathrm{F}(\dot{q}_i) = \left\{ f_c + (f_s - f_c)e^{-\left|\frac{\dot{q}_i}{f_t}\right|^{\delta}} \right\} \mathrm{sgn}(\dot{q}_i) + b\dot{q}_i, \tag{18}$$

where f_c is the Coulomb friction force, f_t is the magnitude of the Stribeck friction, b is the viscous friction parameter, f_s is the static friction coefficient, and δ is an empirically determined parameter.[23] All these parameters are listed in Table II.

4.3. Contours

To demonstrate the effectiveness of the PSO procedure, the PDC-PID is tuned for two different types of contours: linear and nonlinear. More specifically, a straight linear contour and a circular contour are traced by the end-effector of the robot. These two contours are quite different in nature, with the linear contour being short in both distance and time duration, while the circular one is longer in time duration and covers a large distance with a higher speed. The characteristics of both contours can be found in Table III:

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Table III. Simulation contours.

Contour type	Linear	Circular
Starting point (m)	$(0.5, 0.5, \frac{\pi}{3})$	$(0.6, 0.0, \frac{\pi}{3})$
Ending point (m)	$(0.7, 0.7, \frac{\pi}{3})$	$(0.6, 0.0, \frac{\pi}{3})$
Maximum joint speed (rad/s)	1.281	2.290
Motion duration (s)	1	8

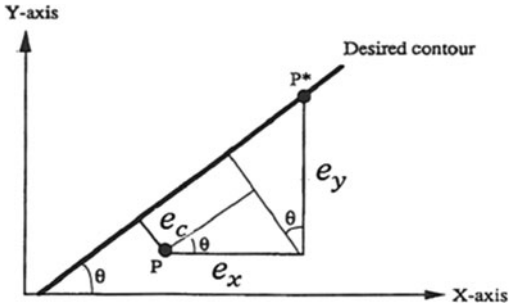


Fig. 2. Linear contour error.[24]

4.4. Contour error calculation

Due to the nature of the contour error (e_c), two different equations are used for its calculation. For linear contours, the contour error can be estimated by the following equation:

$$e_c = -e_x \sin\theta + e_y \cos\theta, \tag{19}$$

where e_x and e_y are the tracking errors for the end-effector in the X-axis and Y-axis, and θ is the angle, as shown in Fig. 2.

For circular contours, the contour error is defined as

$$e_c = \sqrt{(x_P - x_o)^2 + (y_P - y_o)^2} - R, \tag{20}$$

where (x_P, y_P) is the actual end-effector contour, and (x_o, y_o) and R are the coordinates of the center and the radius of the circle, respectively.[25] x_P and y_P can be calculated based on the forward kinematics of the robot from the actual joint positions.

4.5. Choice of acceleration parameters

The control gains for the linear contour described in Table III were optimized under various values of two acceleration parameters. Figure 3 shows the absolute error per iteration of the PSO, which result from various values of the acceleration parameters.

From all these seven cases, the combination of $\alpha_1 = 1.5, \alpha_2 = 1.0$ produced the lowest (best) results while the combination of $\alpha_1 = 2.0, \alpha_2 = 3.0$ produced the highest (worst) results. The former can be justified by the parameter α_1 that amplified the influence of each particle's previous position in its velocity, while the latter resulted from the amplified influence of the *gbest* on the swarm's new velocities.

Despite their success, the above two combinations are not considered as the suitable ones for the optimization due to their low repeatability. Most of the acceleration combinations of Fig. 3 failed to produce dependably good results, with their main disadvantage being inconsistent convergence which occurred at extremely different iteration times. This can be partially attributed to the acceleration constants leading the optimization to local minima.[20]

In terms of convergence, the most consistent combination of acceleration parameters through simulation study for this research problem was $\alpha_1 = 0.5$ and $\alpha_2 = 1.0$, which was also used for the following optimizations.

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Table IV. Optimization parameters.

Master motion sampling frequency	100 Hz
Population size	10
Maximum allowed iterations	50
Feasible bounds of control gains	$0 - 10000$
α_1	0.5
α_2	1.0

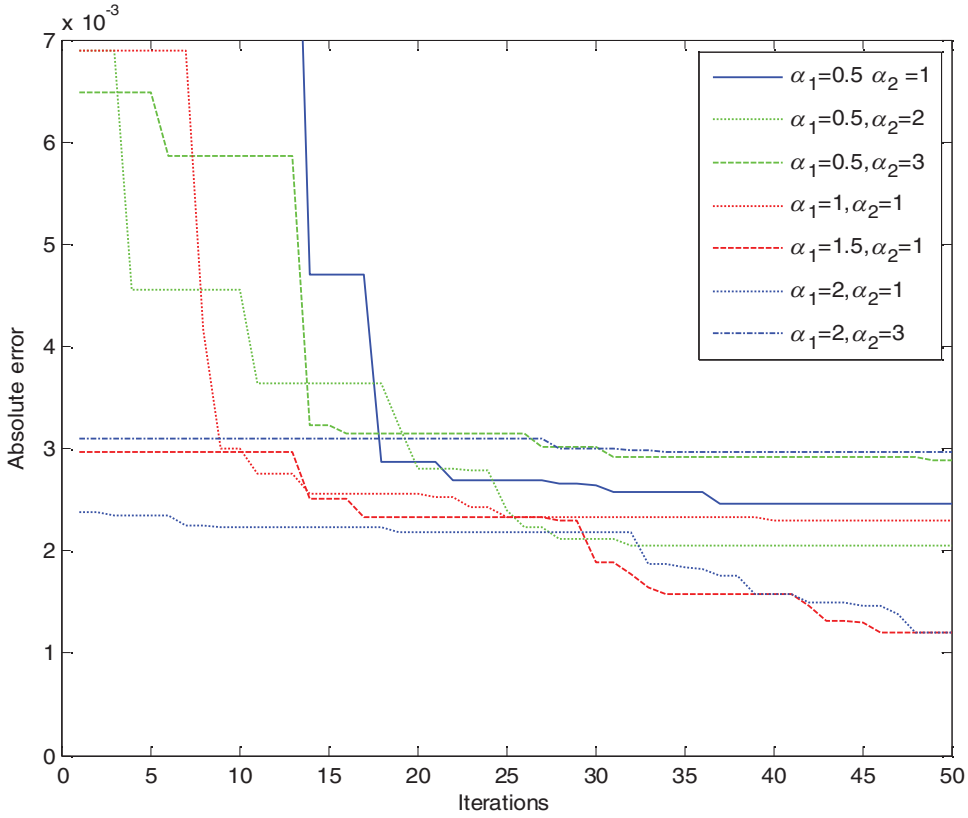


Fig. 3. *gbest* convergence under various acceleration parameters.

4.6. Optimization parameters

There are nine control gains (K_P, K_I, and K_D for three actuators) that need to be optimized in this research. Table IV summarizes the values of the parameters affecting the optimization. It should be noted that the sampling frequency for the master motion was kept low in order for the optimization to maintain logical completion time.

5. Results

5.1. PSO performance

Based on the optimization parameters listed in Table IV, linear and nonlinear contour tracking for the 3R robotic manipulator are simulated using three different fitness functions. Table V lists the optimized control gain values that resulted from the PSO process, where the control gains are expressed as the gains for the master motion (subscript m) and the slave motions (subscript s). For example, $K_P = [K_{Pm} K_{Ps}]$ is the proportional gain matrix. As expected, each fitness function and each contour case produced different values for the optimized control gains.

Figures 4 and 5 show the values of the *gbest* per iteration for the linear and circular contours respectively. From these two figures, one can see that IAE was the smoothest of all fitness functions,

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Table V. Optimization results.

	Linear contour		
Fitness function	$K_P = [K_{Pm} K_{Ps}]$	$K_D = [K_{Dm} K_{Ds}]$	$K_I = [K_{Im} K_{Is}]$
IAE	diag{756193277728}	diag{401117195765}	diag{202010552033}
ISE	diag{787991455312}	diag{172821589973}	diag{45251841944}
MSMAE	diag {625598443747}	diag {747122507287}	diag{840093594871}
Trial and error	diag{48327123120}	diag{38303835560}	diag{774015906230}
	Circular contour		
	$K_P = [K_{Pm} K_{Ps}]$	$K_D = [K_{Dm} K_{Ds}]$	$K_I = [K_{Im} K_{Is}]$
IAE	diag{983379956379}	diag{15858501059}	diag{999726038888}
ISE	diag{851380486524}	diag{6828254248}	diag{584488198226}
MSMAE	diag{845872987353}	diag{165325384309}	diag{999519917411}
Trial and error	diag{300030002500}	diag{200017501250}	diag{500500500}

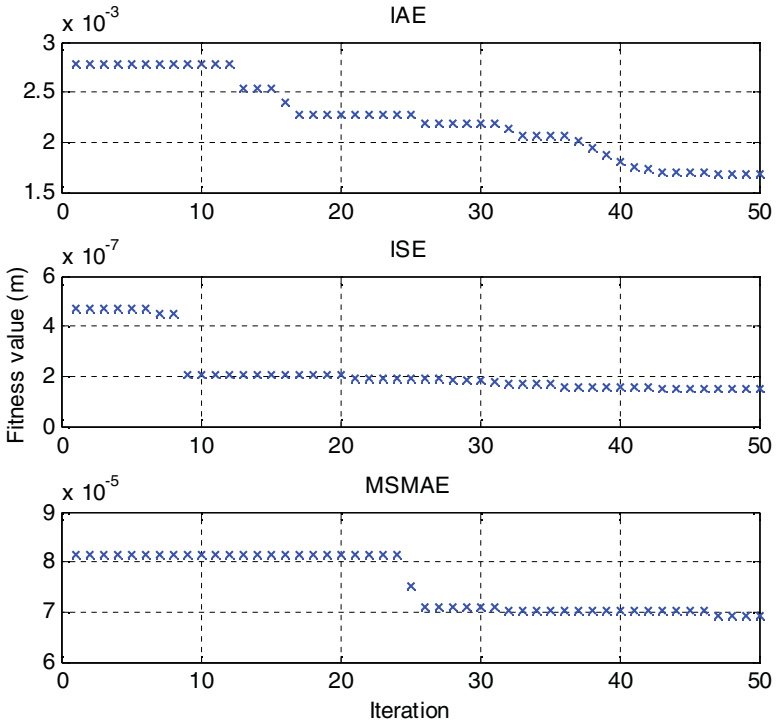


Fig. 4. Linear contour *gbest* values for different fitness functions.

and the proposed MSMAE coming second. For the linear contour, ISE started converging earlier than the other two fitness functions; however, the other two fitness functions exceeded it in contour tracking performance. For the circular contour, IAE was constantly smooth and converged faster than the other two.

Figures 6 and 7 show simulation results as produced with gains resulting after 10, 20, 30, and 40 iterations of the PSO algorithm. One can see that for the linear case (Fig. 6), IAE produced initially better results, but eventually MSMAE and ISE started being more efficient as the iteration numbered increased. The circular contour optimization in Fig. 7 did not feature the diversity of the linear case, with the results maintaining generally similar magnitudes through the iterations. Nonetheless it should be noted that IAE and MSMAE managed to decrease their maximum values as the iteration numbers increased, which of course led to them producing lower results than ISE.

The simulations of the final results show that all the fitness functions produced good contour tracking results for the linear and circular contours as indicated by Figs. 8 and 9 respectively. From

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

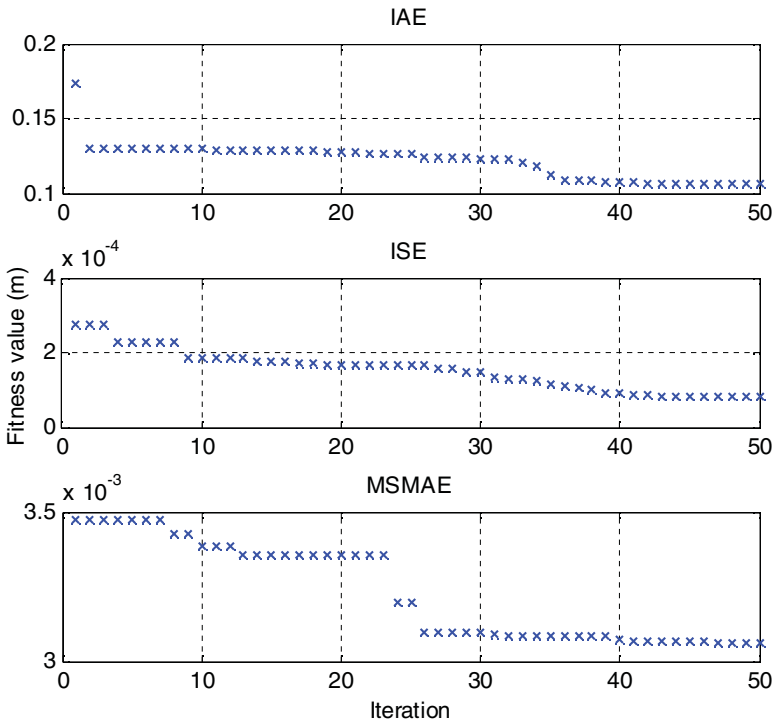


Fig. 5. Circular contour *gbest* values for different fitness functions.

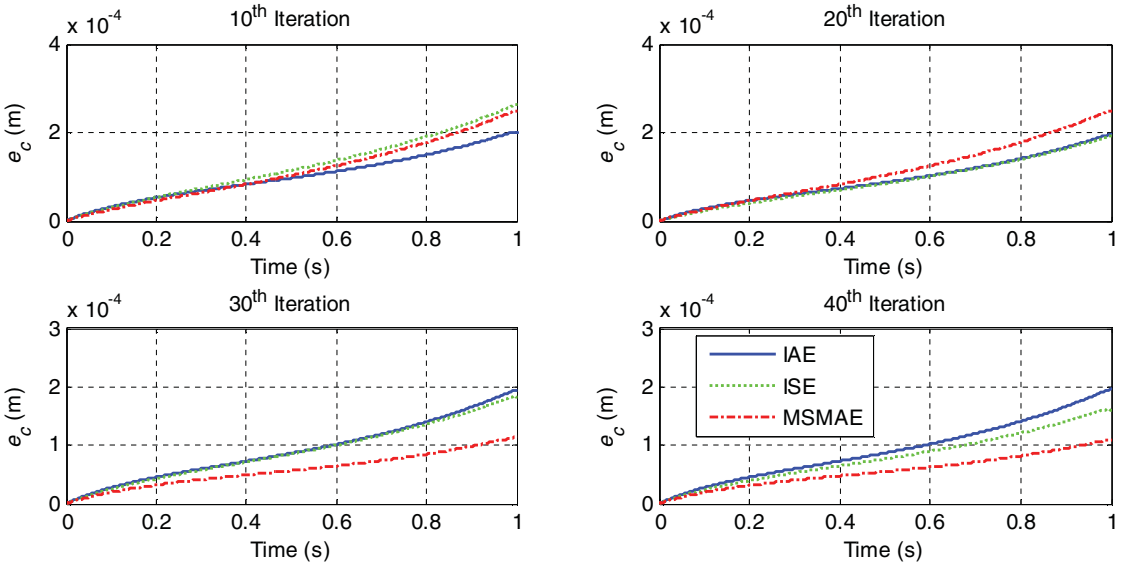


Fig. 6. Linear contour error for gain results of various iterations.

these two figures, one can see that the MSMAE is the best fitness function for linear contour tracking, and IAE and MSMAE are better than ISE for circular contour tracking.

Table VI shows that, for the linear case, ISE produced 14.3 % lower mean error results than IAE, but MSMAE produced 32.9 % lower contour error than ISE. For the circular contour, ISE outperformed the IAE and MSMAE fitness functions by 17.4 % and7.9 %, respectively. All fitness functions produced better contour tracking performances than the trial-and-error technique, with ISE resulting in 72.0 % and 79.8 % lower than the trial-and-error technique contour error for linear and circular contours, respectively. The standard deviations of the contour error followed the same trends

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

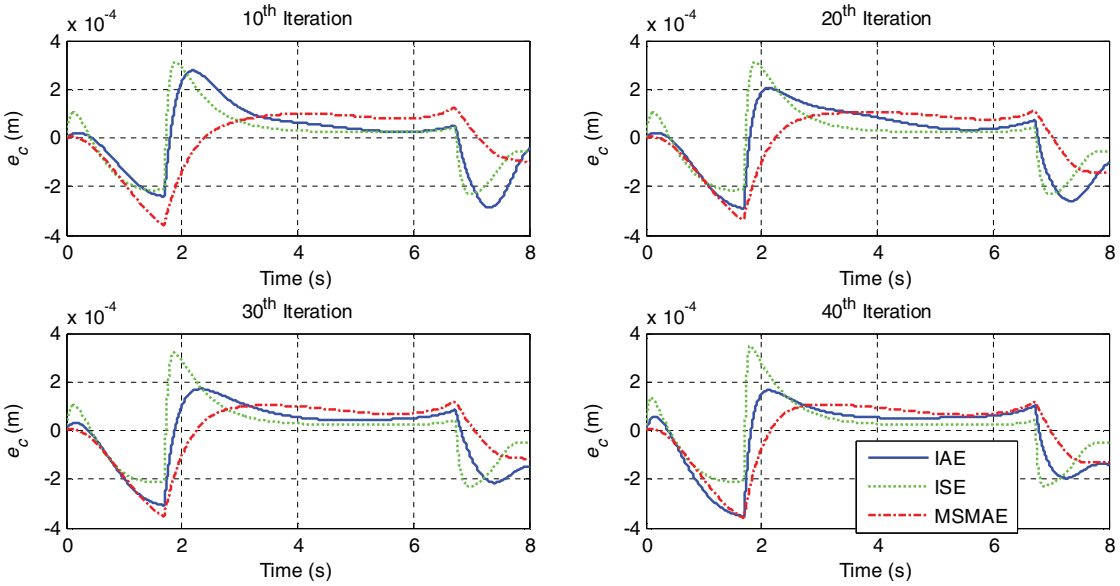


Fig. 7. Circular contour error for gain results of various iterations.

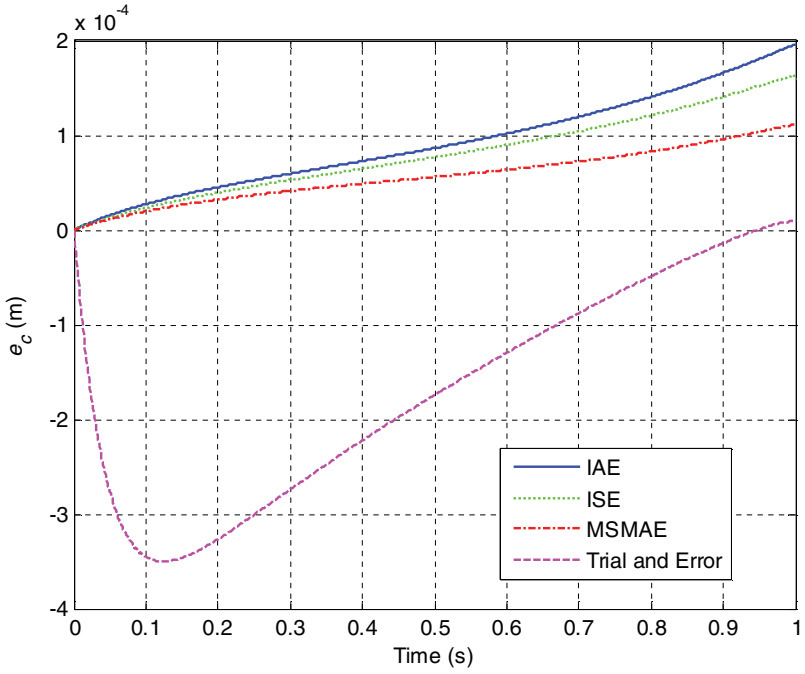


Fig. 8. Final contour tracking results—linear contour.

as the mean of the fitness functions. Conversely, the maximum values of the contour errors stayed close in value for each contour.

5.2. Comparison of PSO and DE

In addition, the DE method[26,27] is also applied for the gain tuning of position domain PID controller and is compared with the SPO method. *It should be noted that for each PID tuning process in the simulation, the sampling frequency is set 100 Hz for the master motion in order to reduce the total computation time for the optimization, as the computation time is not proportional but exponential growth with the sampling frequency for PID simulation.*

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Table VI. Resulting contour error (all results in meters).

Fitness function	Mean	S.D.	Max.
	Linear contour		
IAE	9.1710^{-5}	5.0210^{-5}	2.0010^{-4}
ISE	7.9510^{-5}	4.2310^{-5}	1.6010^{-4}
MSMAE	5.7010^{-5}	2.7510^{-5}	1.1010^{-4}
Trial and error	1.6910^{-4}	1.1510^{-4}	3.5010^{-4}
	Circular contour		
IAE	1.1010^{-4}	8.0310^{-5}	3.5010^{-4}
ISE	9.2410^{-5}	8.1810^{-5}	3.5010^{-4}
MSMAE	1.0010^{-4}	7.2910^{-5}	3.6010^{-4}
Trial and error	2.1510^{-4}	1.5710^{-4}	7.8810^{-4}

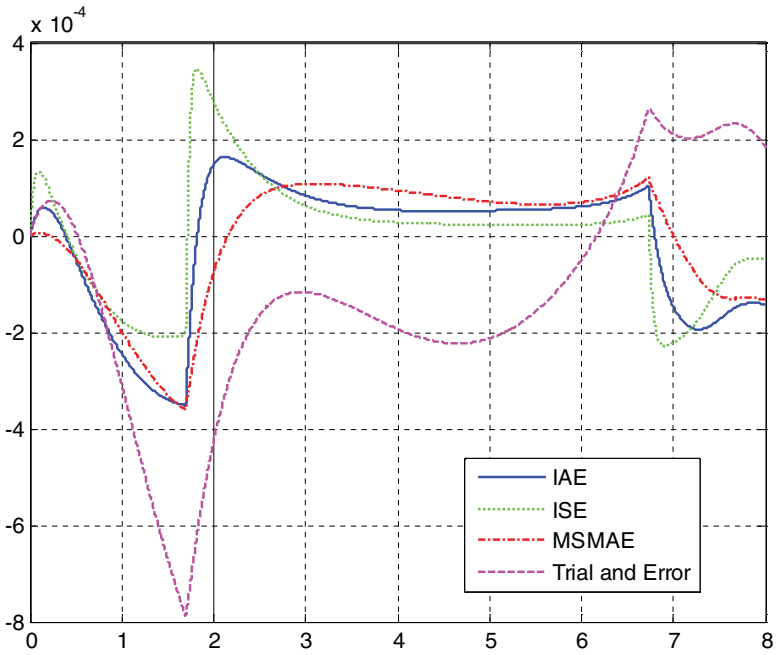


Fig. 9. Final contour tracking results—circular contour.

Table VII shows the comparison results between these two optimization methods. From this table, one conclusion is that both methods obtained very good performances based on the MSMAE fitness function. The DE obtained slightly better performances than the PSO for linear contour motion, and much better performance for circular contour motion. Such results can be explained as follows: Nonlinear contours tracking is more complicated and harder to select control gains than linear contours. Looking at them as two different optimization problems, the nonlinear case is therefore harder to be solved and probably features more local minimums that can trap the optimization than the linear case. PSO tends to follow the best solution in order to create the next iteration, while DE randomly selects three members of the population in order to create the next generation for each member of the population. This makes DE better than PSO in avoiding local minima, which means that DE is better and efficient in solving nonlinear problems.

After the optimized PID control gains are obtained from PSO and DE, the contour tracking simulations are re-conducted again by increasing the sampling frequency from 100 to 1000 Hz. Table VIII shows the new comparison results based on PSO and DE for linear contour and circular contour, respectively. Compared with the results shown in Table VII, it can be seen that the contour tracking performances are improved for both PSO and DE algorithms, and the performances based on PSO

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

Table VII. Comparison results between PSO and DE based on MSMAE (all results in meters).

Optimization method	Mean	S.D.	Max.
Linear contour			
PSO	1.3810^{-4}	9.8710^{-5}	3.7210^{-4}
DE	1.1810^{-4}	8.6810^{-5}	3.4310^{-4}
Circular contour			
PSO	5.8210^{-4}	3.1610^{-4}	1.3910^{-3}
DE	4.4010^{-4}	3.0210^{-4}	1.1610^{-3}

Table VIII. Comparison results between PSO and DE with high sampling frequency (all results in meters).

Optimization method	Mean	S.D.	Max.
Linear contour			
PSO	5.7010^{-5}	2.7510^{-5}	1.1010^{-4}
DE	6.5610^{-5}	3.0710^{-5}	1.2410^{-4}
Circular contour			
PSO	1.0010^{-4}	7.2910^{-5}	3.6010^{-4}
DE	8.5910^{-5}	4.7910^{-5}	2.6210^{-4}

have significant improvement compared with the results obtained in low sampling frequency. For the linear contour tracking case, it is shown that the PSO is better than the DE for all three tracking indexes. The exact reason for such results is not clear and that will become one of the future research work.

For the simulation studies, it was found that the computation time of 50 iterations for DE algorithm is longer than that for PSO algorithm for both linear and nonlinear contour tracking problems. For the linear contour case, the computation time is 123.7 min for PSO and 140 min for DE; for the circular contour case, the computation time are 152.3 and 194.6 min for PSO and DE, respectively.

One possible reason for such results may contribute to the different strategies for the generation of offspring. In DE algorithm, the mutation is conducted by randomly choosing three individuals and calculating the fitness function. Therefore, each individual in the mutant population is evaluated and compared with the parent individuals. For the optimization problems discussed in this paper, the population size is 10 and the iteration number is 50, that means 500 PID control simulations are needed based on the DE algorithm. On the other hand, for the PSO algorithm, the new position of a particle is determined by its previous position and velocity. If the velocity becomes zero that is highly possible when the swarm's best known position $\mathbf{g}$ is approaching (converging to) the optimum of the problem, then the particle stays at the same position and there is no need to calculate the fitness function for that particle. The more the particles closer to the optimum of the problem, the less the fitness function calculation needed. Therefore, it is expected that the total number of PID control simulations based on the PSO algorithm is less than 500 (10×50).

From the above discussions, it can be seen that the PSO algorithm is very good for position domain PID control tuning in terms of computation time compared with DE algorithm. PSO also obtained compatible contour tracking performances (even better performance for linear contour) under a high sampling frequency using the optimized control gains obtained from a low sampling frequency. It provides a good alternative for the position domain PID optimization using low sampling frequencies, as the computation time is exponential growth with the sampling frequency.

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

6. Conclusion

In this paper, the most common PSO algorithm is used to tune the control gains of a position domain PID controller for contour tracking of a serial robotic manipulator. The purpose of the optimization was to find control gains that can reduce the system's contour error of the end-effector for linear and nonlinear motions. Two common existed fitness functions, IAE and ISE, were used for the optimization. A new fitness function called MSMAE was introduced, which was based on the resulting contour error statistics of the controlled system. To assess the validity of the optimization, the controller was optimized for two different contours, a linear motion and a circular motion.

PSO was proven quite efficient for gain tuning of the position domain PID controller, resulting in consistently low contour errors for both contours with the IAE and MSMAE fitness functions providing better results for both cases. The proposed fitness function MSMAE led results quite close to IAE and better than the results of ISE, making it a valid fitness function for this type of optimization problems. Further comparison study for different optimization algorithms under different operation conditions would be a future work.

Acknowledgment

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant awarded to the second author.

References

1. K. J. Astrom and T. Hagglund, "The future of PID control," *Control Eng. Pract.* **9**(11), 1163–1175 (2001).
2. P. Cominos and N. Munro, "PID controllers: Recent tuning methods and design to specification," *IEEE Proc., Control Theory Appl.* **149**(1), 46–53 (2002).
3. B. Nagaraj and P. Vijayakumar, "A comparative study of PID controller tuning using GA, EP, PSO and ACO," *J. Autom. Mobile Robot. Intell. Syst.* **5**(2), 42–48 (2011).
4. J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Procceedings of IEEE International Conference on Neural Networks*, vol. 4, Perth, Australia (Nov. 27–Dec. 1, 1995) pp. 1942–1948.
5. M. R. Airashidi and M. E. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *IEEE Trans. Evol. Comput.* **13**(4), 913–918 (2009).
6. M. Nasri, H. Nezamabadi-pour and M. Maghfoori, "A PSO-based optimum design of PID controller for a linear brushless DC motor," *Word Acad. Sci. Eng. Technol.* **26**(40), 211–215 (2007).
7. S. M. GirirajKumar, D. Jayaraj and A. R. Kishan, "PSO-based tuning of a PID controller for a high performance drilling machine," *Int. J. Comput. Appl.* **1**(19), 12–18 (2010).
8. M. Solihin, M. Kamal and A. Legowo, "Optimal PID Controller Tuning of Automatic Gantry Crane Using PSO Algorithm," *Procceeding of the 15th International Symposium on Mechatronics and its Applications*, Amman, Jordan (May 27–29, 2008) pp. 1–5.
9. C. A. Kumar, "Multi-objective PID controller based on adaptive weighted PSO with applications to steam temperature control in boilers," *Int. J. Eng. Sci. Technol.* **2**(7), 3179–3184 (2010).
10. C. Lv and Y. Pang, "Improved PD Controller for AUV Based on MPSO," *Proceedings of the IEEE International Asia Conference on Informatics in Control, Automation and Robotics*, Bangkok, Thailand (Feb. 1–2, 2009) pp. 24–28.
11. H. T. Dorrah, A. M. El-Garhy and M. E. El-Shimy, "Design of PSO-Based Optimial Fuzzy PID Controllers for the Two-Coupled Distillation Column Process," *Proceedings of the 14th International Middle East Power Systems Conference*, Cairo, Egypt (Dec. 19–21, 2010) pp. 181–188.
12. M. Zamani, N. Sadati and M. K. Ghartemani, "Design of an H PID controller using particle swarm optimization," *Int. J. Control Autom. Syst.* **7**(2), 273–280 (2009).
13. L. Lin and F. Wang, "Robust PID Controller Design Using Particle Swarm Optimization," *Proceedings of the 7nth Asian Control Conference, IEEE*, Hongkong (Aug. 27–29, 2009) pp. 1673–1678.
14. P. R. Ouyang, J. Huang, W. J. Zhang and T. Dam, "Contour tracking control in position domain," *Mechatronics* **22**(7), 934–944 (2012).
15. P. R. Ouyang, V. Pano and T. Dam, "PID position domain control for contour tracking," *Int. J. Syst. Sci.* DOI: http://dx.doi.org/10.1080/00207721.2013.775385 (2013).
16. P. R. Ouyang, V. Pano and J. Acob, "Position domain contour control for multi-DOF robotic system," *Mechatronics* **23**(18), 1061–1071 (2013).
17. P. R. Ouyang, T. Dam and V. Pano, "Cross-coupled PID control in position domain for contour tracking," *Roboticia* DOI: http://dx.doi.org/10.1017/S0263574714000769 (2014).
18. J. Craig, *Introduction to Robotis: Mechanics and Control* (Peason Prentice Hall, Upper Saddle River, New Jersey, 2005).
19. R. E. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," *Comput. Struct.* **85**(19–20), 1579–1588 (2007).

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

20. S. Ebbesen, P. Kiwitz and L. Guzzella, "A Generic Particle Swarm Optimization Matlab Function," *Proceedings of the IEEE American Control Conference*, Montreal, Canada (Jun. 27–29, 2012) pp. 1519–1524.
21. Y. Zhang, F. Qiao, J. Lu, L. Wang and Q. Wu, "Performance Criteria Research on PSO-PID Control Systems," *International Conference on Intelligent Computing and Cognitive Informatics*, Kuala Lampur, Malaysia (Jun. 22–23, 2010) pp. 316–320.
22. K. M. Lynch, N. Shiroma and K. Tanie, "Colllision-free trajectory planning for a 3-DOF robot with a passive joint," *Int. J. Robot. Res.* **19**(12), 1171–1184 (2000).
23. B. Armstrong-Helouvry, P. Dupont and C. Canudas De Wit, "A survey of models, analysis tools and compensation methods for the control of machines with friction," *Automatica* **30**(7), 1083–1138 (1994).
24. S. S. Yeh and P. L. Hsu, "A New Approach to Biaxial Cross-Coupled Control," *Proceedings of the 2000 IEEE International Conference on Control Applications*, Anchorage, USA (Sep. 25–27, 2000) pp. 168–173.
25. M. Y. Cheng and C.C. Lee, "Motion controller design for contour-following tasks based on real-time contour error estimation," *IEEE Trans. Ind. Electron.* **54**(3), 1686–1695 (2007).
26. R. Storn and K. Price, "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.* **11**(4), 341–359 (1997).
27. S. Panda, "Robust coordinated design of multiple and multi-type damping controller using differential evolution algorithm," *Int. J. Electr. Power Energy Syst.* **33**(4), 1018–1030 (2011).

Appendix

For the 3R serial robotic manipulator simulated in this paper, the dynamics of Eq. (1) includes three active joints. The inertial matrix $M(q)$, the matrix of coriolis and centrifugal forces $C(q, \dot{q})$, and the vector of gravitational terms $G(q)$ can be expressed by the following two 3×3 matrices and one 3×1 vector.

A) Inertial matrix $M(q)$:

$$M_{11} = I_1 + I_2 + I_3 + m_1 r_1^2 + m_2\left(l_1^2 + r_2^2 + 2l_1 r_2 \cos(q_2)\right) + m_3(l_1^2 + l_2^2 + r_3^2 + 2l_1 l_2 \cos(q_2) + 2l_2 r_3 \cos(q_3) + 2l_1 r_3 \cos(q_2 + q_3)) \quad \text{(A 1)}$$

$$M_{12} = M_{21} = I_2 + I_3 + m_2\left(r_2^2 + l_1 r_2 \cos(q_2)\right) + m_3(l_2^2 + r_3^2 + 2l_1 l_2 \cos(q_2) + 2l_2 r_3 \cos(q_3) + 2l_1 r_3 \cos(q_2 + q_3)) \quad \text{(A 2)}$$

$$M_{13} = M_{31} = I_3 + m_3(r_3^2 + 2l_3 \cos(q_3) + 2l_1 r_3 \cos(q_2 + q_3)) \quad \text{(A 3)}$$

$$M_{22} = I_2 + I_3 + m_2 r_2^2 + m_3(l_2^2 + r_3^2 + 2l_2 r_3 \cos(q_3)) \quad \text{(A 4)}$$

$$M_{23} = M_{32} = I_3 + m_3(l_2^2 + r_3^2 + 2l_2 r_3 \cos(q_3)) \quad \text{(A 5)}$$

$$M_{33} = I_3 + m_3 r_3^2 \quad \text{(A 6)}$$

B) Coriolis-centrifugal forces $C(q, \dot{q})$:

$$C_{11} = \begin{bmatrix} 0 \\ m_2 l_1 r_2 \sin(q_2) - m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3)) \\ -m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \quad \text{(B 1)}$$

$$C_{12} = \begin{bmatrix} m_2 l_1 r_2 \sin(q_2) - m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3)) \\ -m_2 l_1 r_2 \sin(q_2) - m_3(l_1 l_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3)) \\ m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \quad \text{(B 2)}$$

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press

$$C_{13} = \begin{bmatrix} -m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3) \\ m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3) \\ -m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{B 3}$$

$$C_{21} = \begin{bmatrix} m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) + m_2 l_1 r_2 \sin(q_2) \\ 0 \\ m_3 l_2 r_3 \sin(q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{B 4}$$

$$C_{22} = \begin{bmatrix} 0 \\ 0 \\ -m_3 l_2 r_3 \sin(q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{B 5}$$

$$C_{23} = \begin{bmatrix} m_3 l_2 r_3 \sin(q_3) \\ -m_3 l_2 r_3 \sin(q_3) \\ -m_3 l_2 r_3 \sin(q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{B 6}$$

$$C_{31} = \begin{bmatrix} m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) \\ m_3 l_2 r_3 \sin(q_3) \\ 0 \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{B 7}$$

$$C_{32} = \begin{bmatrix} m_3 l_2 r_3 \sin(q_3) \\ m_3 l_2 r_3 \sin(q_3) \\ 0 \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{B 8}$$

$$C_{33} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{B 9}$$

C) Gravity vector $G(q)$:

$$G_{11} = g\{m_1 r_1 \cos(q_1) + m_2(l_1 \cos(q_1) + r_2 \cos(q_1 + q_2)) + m_3(l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) + r_3 \cos(q_1 + q_2 + q_2))\} \tag{C 1}$$

$$G_{12} = g\{m_2 r_2 \cos(q_1 + q_2) + m_3(l_2 \cos(q_1 + q_2) + r_3 \cos(q_1 + q_2 + q_2))\} \tag{C 2}$$

$$G_{13} = g m_3 r_3 \cos(q_1 + q_2 + q_2) \tag{C 3}$$

https://doi.org/10.1017/S0263574714002331 Published online by Cambridge University Press