Robotics and Computer–Integrated Manufacturing 48 (2017) 219–232

Contents lists available at ScienceDirect

Robotics and Computer–Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Optimal trajectory generation algorithm for serial and parallel manipulators

Serdar Kucuk

Kocaeli University, Technology Faculty, Department of Biomedical Engineering, Umuttepe Campus, 41380 Kocaeli, Turkey

ARTICLE INFO

Keywords:
Trajectory generation
Cubic spline
Jerk
PSO algorithm
Serial and parallel manipulators

ABSTRACT

In this paper, an **O**ptimal **T**rajectory **G**eneration **A**lgorithm (**OTGA**) is developed for generating minimum-time smooth motion trajectories for serial and parallel manipulators. OTGA is divided into two phases. The first phase encompasses derivation of minimum-time optimal trajectory using cubic spline due to its less vibration and overshoot characteristics. Although cubic splines are widely used in robotics, velocity and acceleration ripples in the first & last knots can worsen manipulator trajectory. The second phase includes changing cubic spline interpolation in the first and last knots of optimized trajectory with 7th order polynomial for having zero jerk at the beginning and end points of trajectory. Performing this modification eliminate undesired worsening in the trajectory and provide smoother start and stop of joint motions. **P**article **S**warm **O**ptimization (**PSO**) is chosen as optimization algorithm because of its easy implementation and successful optimization performance. OTGA has been tested in simulation for PUMA robot and results are compared with algorithms proposed by earlier authors. In addition, a discrete-time PID control scheme for PUMA robot is designed for comparing energy consumption of OTGA with algorithms developed by previous authors. Comparison results illustrated that OTGA is the better trajectory generation algorithm than the others.

1. Introduction

Trajectory generation is very important for robotic manipulators since it produces input to the control system of the robotic manipulators for executing the desired task with satisfactory performance [1]. In addition, generated trajectories must provide smooth kinematic motion to maintain high tracking accuracy and avoid exciting the natural modes of control system [2]. In industry, most of the robot trajectories are first constructed off-line and then end-effectors of robots are forced to track this path on-line. Off-line trajectory generation can be performed at two different approaches namely, hand level and joint level. Minimum time trajectory generation at hand level was performed by the authors of Luh and Lin [3]. In hand level approach, joint coordinates are converted into Cartesian coordinates by means of Jacobian transformation for each sampling period. Errors may occur in inverse Jacobian transformation therefore robot tracks the trajectory incorrectly. Afterwards reference input torque for each joint is calculated by means of robot dynamic equations which are highly nonlinear [4]. The major drawback of the hand level approach is that whole process is performed at one sampling period. This is always limited and decelerates the system performance [5]. In joint level approach, Cartesian coordinates are converted into joint coordinates by means of inverse kinematics. Robot manipulators are controlled at joint level which is less expensive in terms of computational complexity compared to hand level approach. In joint level approach, kinematic constraints are considered only during the trajectory generation [6]. The dynamic constraints that increase the computational effort are ignored. It is the major advantage of the joint level approach. Hence it is mostly preferred to reduce computational effort.

In joint level approach, a number of trajectory points (via points) is firstly defined in terms of reference end-effector pose in Cartesian space. Secondly these via points (called also as knots) are transformed into joint angles by using inverse kinematics. Finally these knots are interpolated using splines. Several kinds of splines such as polynomial, trigonometric [7], quantic [8,9], B-spline [10] and cubic spline [5] are proposed for trajectory generation in literature. Among these splines, cubic spline is mostly used for interpolating the robot trajectories since it provides continuity in velocity and acceleration at every knot. It should be noticed that cubic spline curves do not provide continuity in jerk. Although cubic spline curves does not provide continuity in jerk, they are only the third-degree spline that can provide jerk limitation [11]. Cubic splines can also produce lowest possible jerk peak [8]. This feature is especially important for reducing vibration. Thus a smooth curve trajectories can be obtained during motion. Minimizing jerk decreases joint position errors, causes less vibration, limits excessive wear on the robot and prevents large oscillations which can occur employing higher order polynomials [11–14].

Although cubic splines are commonly used in robotics due to its several advantageous mentioned above they have an important drawback. Joint velocity and acceleration ripples which take place in the

E-mail address: skucuk@kocaeli.edu.tr.

http://dx.doi.org/10.1016/j.rcim.2017.04.006
Received 31 August 2016; Received in revised form 13 April 2017; Accepted 13 April 2017

first and last knots can deteriorate the manipulator trajectory [15]. Several studies have been performed to generate smooth and time-optimal trajectories by overcoming this problem in the literature. Lin et al. [5] formulated and optimized cubic polynomial joint trajectories for using polyhedron search method. Piazzi and Visioli [6] studied optimization of global minimum-jerk trajectory planning of robot manipulators using interval analysis. Tandu and Bazaz [16] developed three-cubic methods to generate a joint trajectory by interpolating intermediate positions and velocities based on a combination of cubic splines. They used analytical optimization approach. Chettibi et al. [17] presented minimum cost trajectory planning algorithm for robotic manipulators using Sequential Quadratic Programming (SQP) method. Gasparetto and Zanotto [18] proposed a technique for time-jerk optimal planning of robot trajectories. They used sequential quadratic programming techniques in order to get the optimal trajectory. Kolter and Andrew [15] developed a method for optimizing task-space cubic spline trajectories using convex optimization techniques. Demeulenaere et al. [19] developed a general framework to synthesize optimal polynomial splines for rigid motion systems using convex programming framework. Aribowo and Terashima [11] performed a study about cubic spline trajectory planning and vibration suppression of semiconductor wafer transfer robot arm. They used SQP method for solving the constrained nonlinear trajectory planning optimization problem. The authors mentioned above have used classic numerical optimization algorithms based on successive linearization using the first and the second derivatives of objective functions in trajectory planning of robotic manipulator. The disadvantages of these derivative-based optimization algorithms arise from sensitivity to problem formulation and algorithm selection. They usually converge to a local minimum [20]. The PSO algorithm used also in this study have certain advantages over classical optimization techniques and other evolutionary algorithms: (i) It finds optimal or near optimal solutions to nonlinear and discontinuous problems at higher dimensions within the shorter computation time, (ii) it has memory that makes the knowledge of the better solution obtained from last iterations to save all the particles, (iii). The solution does not depend on the initial population. Therefore PSO algorithm is considered an important part of OTGA proposed in this study. Trajectory generation is performed into phases.

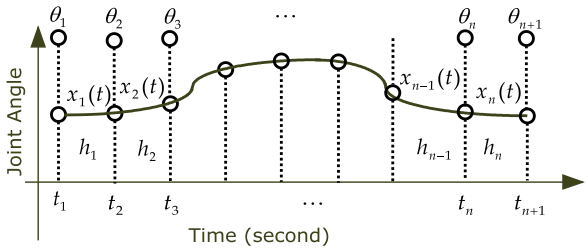


Fig. 1. Cubic spline trajectory with several piecewise cubic segments.

2. Formulation of cubic spline joint trajectory

In robotics, there are two types of motion used in general namely, pick & place motion and path-constraint motion. In pick and place motion, start and end-points are important. Robot manipulators perform a task freely between pick and place locations. However in constraint motion, robot manipulators follow a certain path that must be defined in advance [21]. Path constraint motion is of vital importance such as in welding, cutting, surgery and machining applications where continuous path motions are required.

Polynomial splines are often applied to perform path constraint motion in general. Among polynomial splines, cubic splines are often preferred since they provide continuous velocity and acceleration with lowest degree [5]. Cubic spline interpolation for robot manipulators uses several kinematically feasible Cartesian knots between start and end-points. These knots defined in Cartesian task space are then converted into joint space by using inverse kinematics. Subsequently each joint trajectory is planned as a combination of numerous piecewise cubic segments $(x_1(t), x_2(t),\ldots,x_{n-1}(t), x_n(t))$ that connects predefined knots as in Fig. 1. A joint trajectory with n piecewise cubic segments possess n+1 predetermined joint angle values $(\theta_1, \theta_2,\ldots,\theta_n, \theta_{n+1})$ [22].

The following system of linear equations can be derived by performing a sequence of mathematical operations as in [22]. The joint acceleration $(\ddot{\theta}_2, \ddot{\theta}_3,\ldots,\ddot{\theta}_{n-1}, \ddot{\theta}_n)$ at each knot can be found by solving following system of linear equations where h_i represents the time intervals between $t_i$and t_{i+1}, $(i = 1,2, \ldots, n + 1)$.

$$\begin{bmatrix} h_1^2+3h_1h_2+2h_2^2 & h_2^2 & 0 & 0 & 0 & 0 & 0 \\ 3(h_1+h_2) & 2h_3+3h_2 & h_3 & 0 & 0 & 0 & 0 \\ & h_3 & 2(h_3+h_4) & h_4 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} & 0 \\ 0 & 0 & 0 & 0 & -h_{n-2} & -3h_{n-1}-2h_{n-2} & -3(h_n+h_{n-1}) \\ 0 & 0 & 0 & 0 & 0 & -h_{n-1}^2 & -(h_n^2+2h_{n-1}^2+3h_{n-1}h_n) \end{bmatrix} \begin{bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\theta}_4 \\ \vdots \\ \ddot{\theta}_{n-2} \\ \ddot{\theta}_{n-1} \\ \ddot{\theta}_n \end{bmatrix}$$
$$= 6\left[\theta_3-\theta_1 \quad \frac{\theta_4-\theta_3}{h_3} \quad \frac{\theta_5-\theta_4}{h_4}-\frac{\theta_4-\theta_3}{h_3} \quad \ldots \quad \frac{\theta_{n-1}-\theta_{n-2}}{h_{n-2}}-\frac{\theta_{n-2}-\theta_{n-3}}{h_{n-3}} \quad \frac{\theta_{n-1}-\theta_{n-2}}{h_{n-2}} \quad \theta_{n+1}-\theta_{n-1}\right]^T \tag{1}$$

The first phase includes derivation of minimum-time optimal trajectory using PSO algorithm. Although cubic spline optimization is useful in minimizing the joint velocity, acceleration ripples and jerk, the first and last knots have still a sharp start and stop of motion that can deteriorate manipulator trajectory. In the second phase, the cubic spline interpolation in the first and last knots of optimized trajectory are changed with the interpolation of seventh order polynomial in order to get rid of this undesired deteriorations. Performing this change makes the joints have smoother start and stop of motion. Subsequently, OTGA has been tested in simulation for PUMA robot and the results are compared with the previous algorithms. Electrical energy consumptions of OTGA and the previous algorithms are also compared and given in a table. Finally comparison results are discussed.

3. Particle swarm optimization

PSO algorithm introduced by Kennedy and Eberhart in 1995 [23] is a robust stochastic population-based technique for solving numerical optimization problems. The PSO algorithm (whose flowchart given in Fig. 2) have certain advantages over classical optimization techniques and other evolutionary algorithms: (i) it finds optimal or near optimal solutions to nonlinear and discontinuous problems at higher dimensions, (ii) it is computationally inexpensive, (iii) it has fewer parameters to adjust, (iv) it has memory that makes the knowledge of the better solution obtained from last iterations to save all the particles, (v) quickly converge the fitness function and (vi) the solution does not depend on the initial population. PSO algorithm has been successfully applied in many kinds of engineering problems [24–26]. In particle swarm optimization technique, every

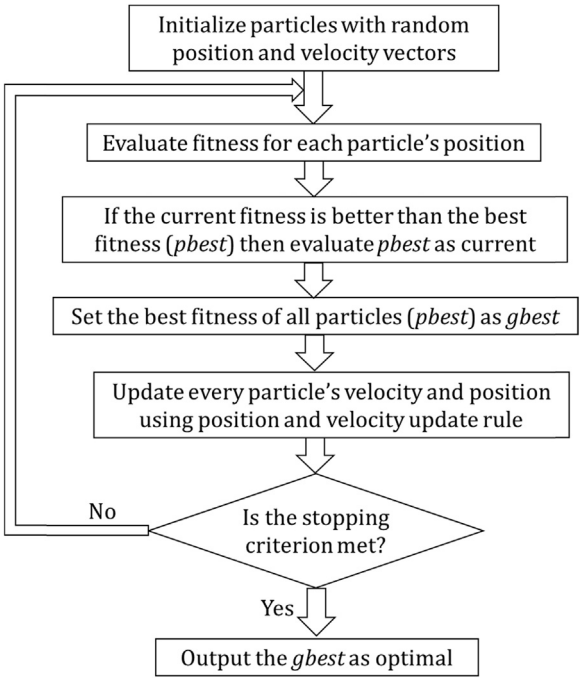


Fig. 2. Flowchart of PSO algorithm.

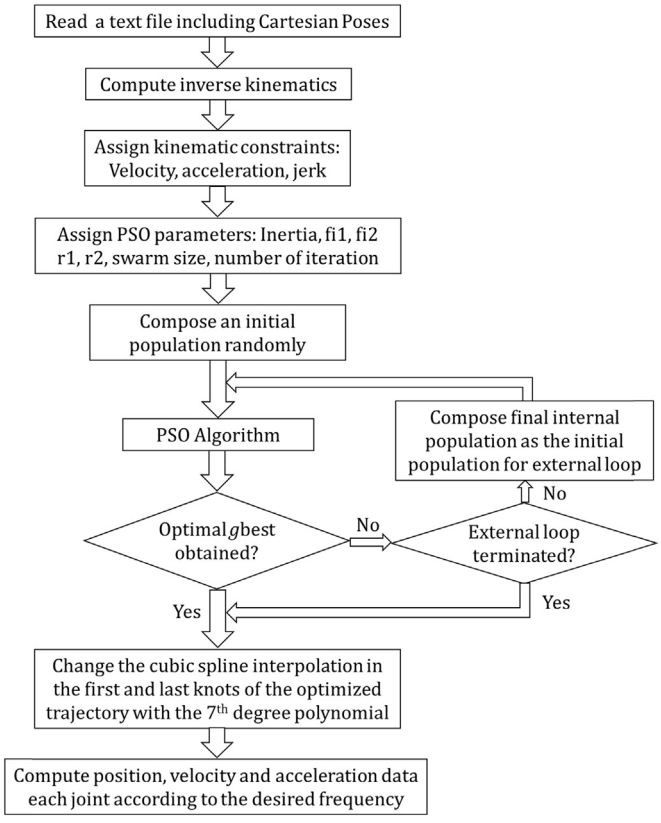


Fig. 3. Flowchart of OTGA.

individual of the swarm is considered as a particle in the search space which has own position and velocity. The swarm of particles fly in multidimensional search space based on the simple equations that look for the best solution. PSO algorithm is initialized with a group of random particles (solutions) and examines for the optimal solution by updating generations in every iteration. The detailed explanation of PSO algorithm can be found in [22].

Table 1
Text file including feasible Cartesian knots between start and end-points.

KN	p_x	p_y	p_z	α	β	γ
1	11.00	11.50	0.00	20.00	0.00	0.00
2	11.25	11.50	0.00	20.00	0.00	0.00
3	11.50	11.50	0.00	20.00	0.00	0.00
4	12.00	11.50	0.00	20.00	0.00	0.00
5	12.50	11.50	0.00	20.00	0.00	0.00
6	13.00	11.50	0.00	20.00	0.00	0.00
7	13.50	11.50	0.00	20.00	0.00	0.00
8	13.75	11.50	0.00	20.00	0.00	0.00
9	14.00	11.75	0.00	20.00	0.00	0.00
10	1	1	1	1	1	1

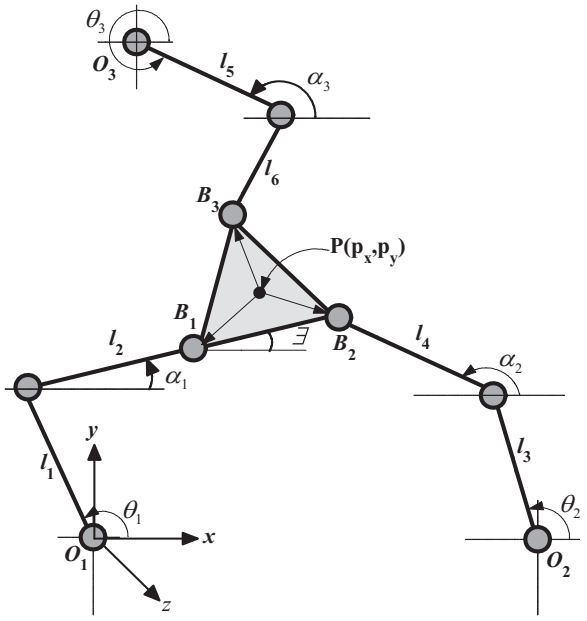


Fig. 4. The schematic sketch of 3-DOF RRR planar parallel manipulator.

4. Optimal trajectory generation algorithm

Flow chart of OTGA is illustrated on Fig. 3. The OTGA composes of the input data generation section, optimization section, trajectory modification section and data output section.

4.1. Input data generation

OTGA is designed to generate data for robot manipulators to perform smooth, time-optimal continuous path motions. In the first step, several kinematically feasible Cartesian knots between start and end-points are defined in advanced. OTGA reads Cartesian knots data from a text file as in Table 1. The text file include knot numbers (KN), position (p_x, p_y, p_z) and orientation (α, β, γ) of the Cartesian knots. When the algorithm reads "1", the data reading is terminated. These knots defined in Cartesian task space are then converted into joint space by using inverse kinematics. OTGA uses a function to perform the inverse kinematics of the Cartesian knots. This inverse kinematics function given as follows includes Cartesian pose, joint space and kinematics parameters.

function [theta1, theta2, …, theta$n$ − 1, theta$n$] = inversekin (px, py, pz, alfa, beta, gamma) (2)

In this function, the Cartesian poses (p_x, p_y, p_z, α, β, γ) are accepted as input parameters while the inverse kinematic solutions of the joints (theta1, theta2, …, theta$n$-*1*, theta$n$) are considered as output para-

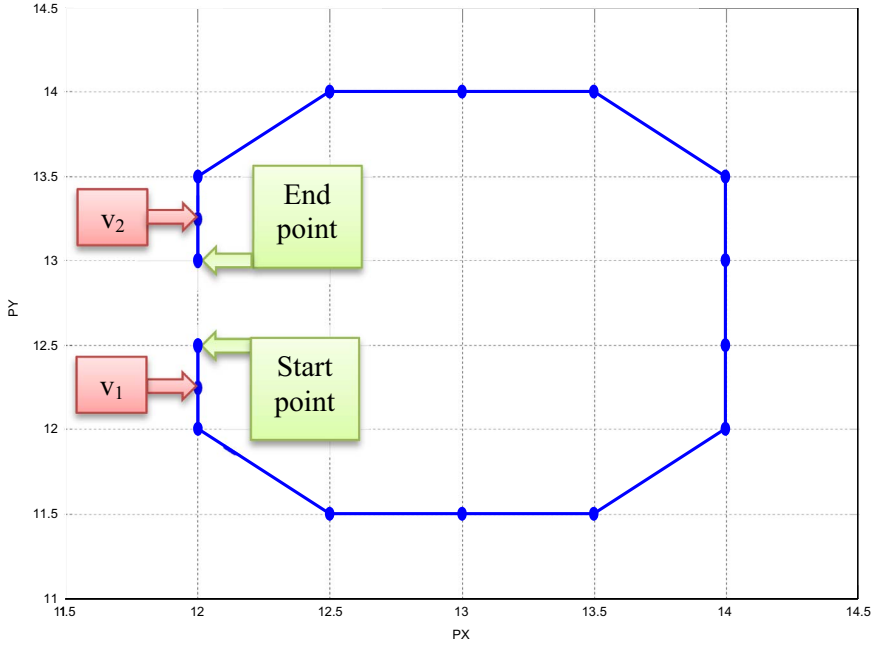


Fig. 5. A sample trajectory with knots and curve segments.

meters where n represents the number of joints. If the inverse kinematics is written as above format, optimal trajectory can be generated for every kinds of serial and parallel robot.

4.2. Cubic spline optimization

In this section, the smooth jerk limited trajectory is obtained by using PSO algorithm. Jerk can also be minimized instead of segments times. But in this case, decreasing joint jerk would increase the traveling time. In this study, firstly the feasible Cartesian knots between start and end-points of the trajectory are defined in advanced. Secondly OTGA finds the joint angle values for each Cartesian knot by using inverse kinematics. Afterwards joint velocities, accelerations and jerks are designed between certain values for each knot considering performance of the robot manipulator and trajectory requirements. Finally, segment times between each pair of adjacent angle values (θ_1, θ_2,…, θ_n, θ_{n+1}) are optimized such that total traveling time is minimized. Another words, while segment times change, joint angle values obtained from Cartesian knots by means of inverse kinematics remain constant. As a conclusion the optimization problem can be stated as follows.

$$min \sum_{i=1}^{n} h_i \tag{3}$$

Subject to

$$\dot{\theta}_i^{min} \le \dot{\theta}_i \le \dot{\theta}_i^{max} i=1, \quad 2, \quad \ldots, \quad m$$

$$\ddot{\theta}_i^{min} \le \ddot{\theta}_i \le \ddot{\theta}_i^{max} i=1, \quad 2, \quad \ldots, \quad m$$

$$\dddot{\theta}_i^{min} \le \dddot{\theta}_i \le \dddot{\theta}_i^{max} i=1, \quad 2, \quad \ldots, \quad m$$

$$h_i^{min} \le h_i \le h_i^{max} i=1, \quad 2, \quad \ldots, \quad n$$

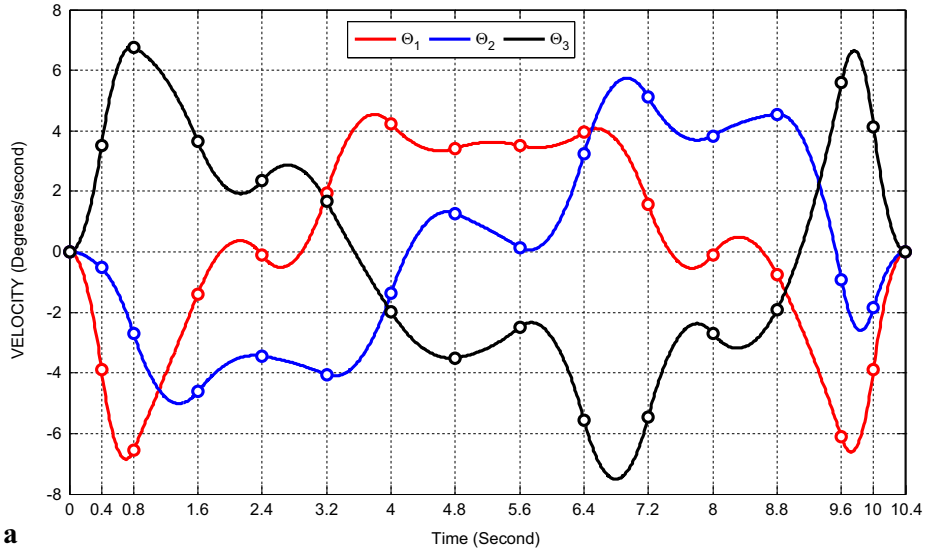
where m and n denote the number of robot joints and knots, respectively. The h_i represents segment times which are aimed to be optimized between certain specified values h_i^{min} and h_i^{max}. The first and last segment times h_1,and,h_n can be designed as longer than internal segment times in order to reduce the velocities and accelerations at the first and last knots. The variables $\dot{\theta}_{start}$, $\ddot{\theta}_{start}$ and $\dot{\theta}_{end}$, $\ddot{\theta}_{end}$ denote velocities and accelerations at start and end points, respectively. $\dot{\theta}_i$, $\ddot{\theta}_i$ and $\dddot{\theta}_i$ illustrate the joint velocity, acceleration and jerk values that have to be the ranges between minimum ($\dot{\theta}_i^{min}$, $\ddot{\theta}_i^{min}$ and $\dddot{\theta}_i^{min}$) and maximum ($\dot{\theta}_i^{max}$, $\ddot{\theta}_i^{max}$ and $\dddot{\theta}_i^{max}$) values, respectively.

4.3. Elimination of undesired deterioration in the first and last knots of the optimized trajectory

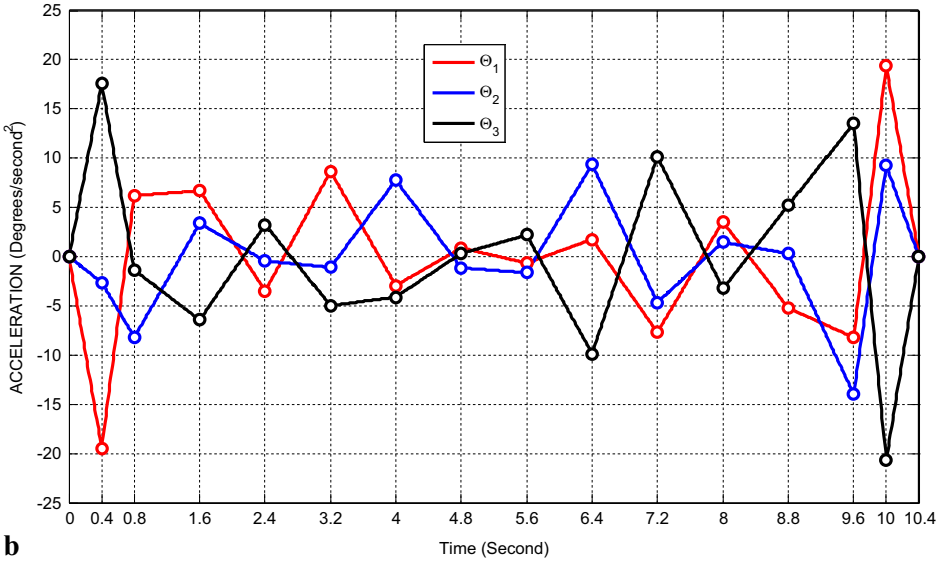
It is very important for robot manipulators to have maximized smoothness of the motion during the critical periods at start-up and rest. Due to its nature, cubic spline has some jerk values along the trajectory. However these jerk values are the lowest possible jerk values [5, 6 and 8]. Although cubic spline provides continuous position, velocity and acceleration at the start and end points it does not support the continuous jerk which may results in vibrations especially at start and rest point. This may reduce tracking accuracy. The most important thing is that the jerk values must be zero especially at the start and end of the trajectory [27]. Therefore, robot designers especially want to set the velocity, acceleration and jerk at the start and rest point to zero [27]. Considering this fact, cubic spline interpolation has been changed with 7th order polynomial for having zero jerk at the beginning and end points of trajectory. Among 5th, 7th and 9th-degree polynomials, Boryga and Graboś [28] stated that the minimum linear and angular jerk are obtained from the 7th-degree polynomial. The second phase includes changing the cubic spline interpolation in the first and last knots of the optimized trajectory (including virtual knots) with the 7th order polynomial joint trajectory interpolation in order to set the first three derivatives of joint positions at the start and rest point to zero. Thus the undesired worsening at the start and end points can be eliminated. The seventh order polynomial function can be described as

$$x(t) = a_7t^7 + a_6t^6 + a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \tag{4}$$

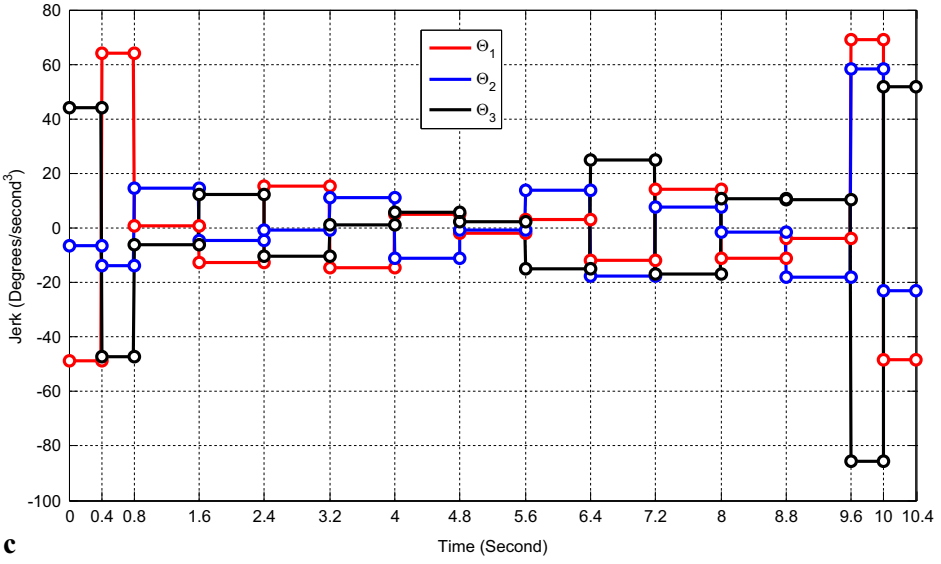
The desired velocity, acceleration and jerk can be obtained by taking the first, second and third derivative of the Eq. (4). Combining position, velocity, acceleration and jerk with eight constraints ($\theta(t_0) = \theta_0$), $\theta(t_f) = \theta_f$, $\dot{\theta}(t_0) = \dot{\theta}_0$, $\dot{\theta}(t_f) = \dot{\theta}_f$, $\ddot{\theta}(t_0) = \ddot{\theta}_0$, $\ddot{\theta}(t_f) = \ddot{\theta}_f$, $\dddot{\theta}(t_0) = \dddot{\theta}_0$ and $\dddot{\theta}(t_f) = \dddot{\theta}_f$) yields eight equations in eight unknowns (a_0, a_1, a_2, a_3, a_4, a_5, a_6 and a_7) where t_0, θ_0, $\dot{\theta}_0$, $\ddot{\theta}_0$ and $\dddot{\theta}_0$ illustrate the initial time, position, velocity, acceleration and jerk while t_f, θ_f, $\dot{\theta}_f$, $\ddot{\theta}_f$ and $\dddot{\theta}_f$ denote the final time, position, velocity, acceleration and jerk,



a



b

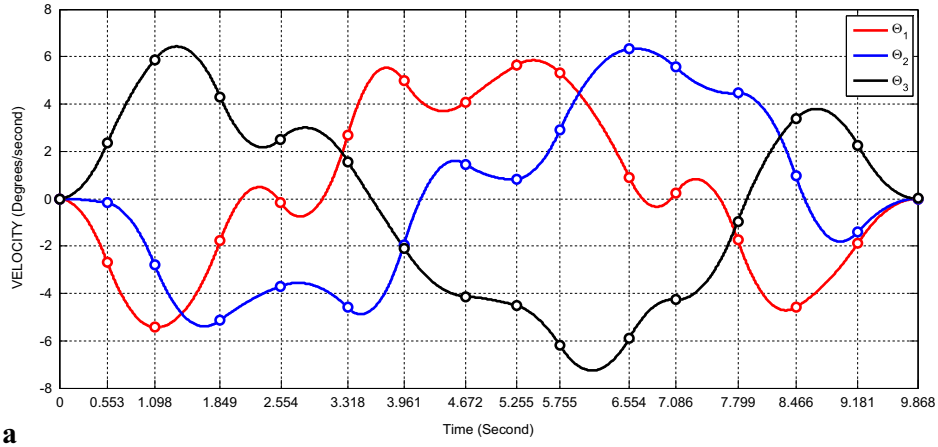


c

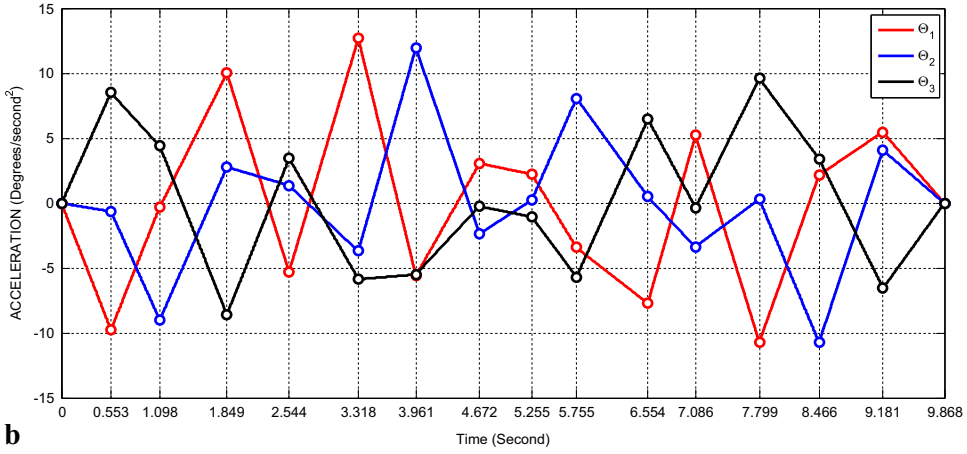
Fig. 6. a. Joint velocity profiles of 3-RRR planar parallel manipulator. b. Joint acceleration profiles of 3-RRR planar parallel manipulator. c. Joint jerk profiles of 3-RRR planar parallel manipulator.

Table 2
The joint displacements of three joints at the knots.

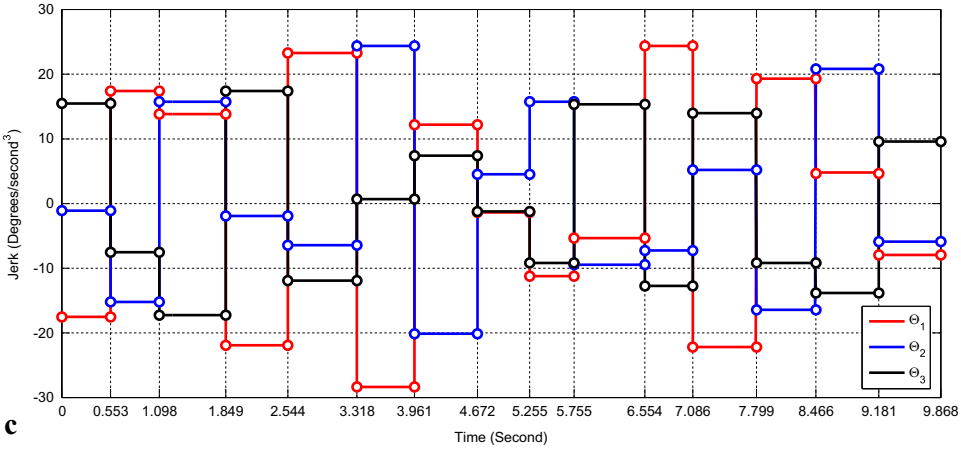
Joint	Knots (degrees)															
	1	VS 2	3	4	5	6	7	8	9	10	11	12	13	14	VS 15	16
1	−7.27	−7.71	−10.2	−13.4	−13.5	−13.3	−10.2	−7.41	−4.56	−1.7	0.99	0.98	1.11	−1.46	−3.89	−4.35
2	109.3	109.4	108.7	105.2	102.2	99.2	96.6	97.0	97.6	98.3	102.4	105.7	109.1	111.3	110.5	110.2
3	181.6	182.0	184.4	188.8	190.7	192.8	192.6	190.2	187.7	185.1	179.6	177.1	174.8	175.8	178.2	178.8



a



b



c

Fig. 7. a. Joint velocity profiles of 3-RRR planar parallel manipulator. b. Joint acceleration profile of 3-RRR planar parallel manipulator. c. Joint jerk profile profiles of 3-RRR planar parallel manipulator.

Table 3
The joints space knots designed for test trajectory of PUMA robot manipulator.

Knots	Joints					
	1	2	3	4	5	6
1	10	15	45	5	10	6
Virtual						
2	60	25	180	20	30	40
3	75	30	200	60	−40	80
4	130	−45	120	110	−60	70
5	110	−55	15	20	10	−10
6	100	−70	−10	60	50	10
7	−10	−10	100	−100	−40	30
Virtual						
8	−50	10	50	−30	10	20

respectively. These eight equations can be combined into a single matrix equation as follows

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 & t_0^6 & t_0^7 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 & t_f^6 & t_f^7 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 & 6t_0^5 & 7t_0^6 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 & 6t_f^5 & 7t_f^6 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 & 30t_0^4 & 42t_0^5 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 & 30t_f^4 & 42t_f^5 \\ 0 & 0 & 0 & 6 & 24t_0 & 60t_0^2 & 120t_0^3 & 210t_0^4 \\ 0 & 0 & 0 & 6 & 24t_f & 60t_f^2 & 120t_f^3 & 210t_f^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_f \\ \dot{\theta}_0 \\ \dot{\theta}_f \\ \ddot{\theta}_0 \\ \ddot{\theta}_f \\ \dddot{\theta}_0 \\ \dddot{\theta}_f \end{bmatrix} \quad (5)$$

The coefficients (a_0, a_1,...,a_7) of the polynomial are determined by solving this matrix equality.

5. An illustrative example

3-DOF RRR planar parallel manipulator performs a trajectory in order to show the efficiency of OTGA. The schematic sketch of 3-DOF RRR planar parallel manipulator is shown in Fig. 4. It has a moving

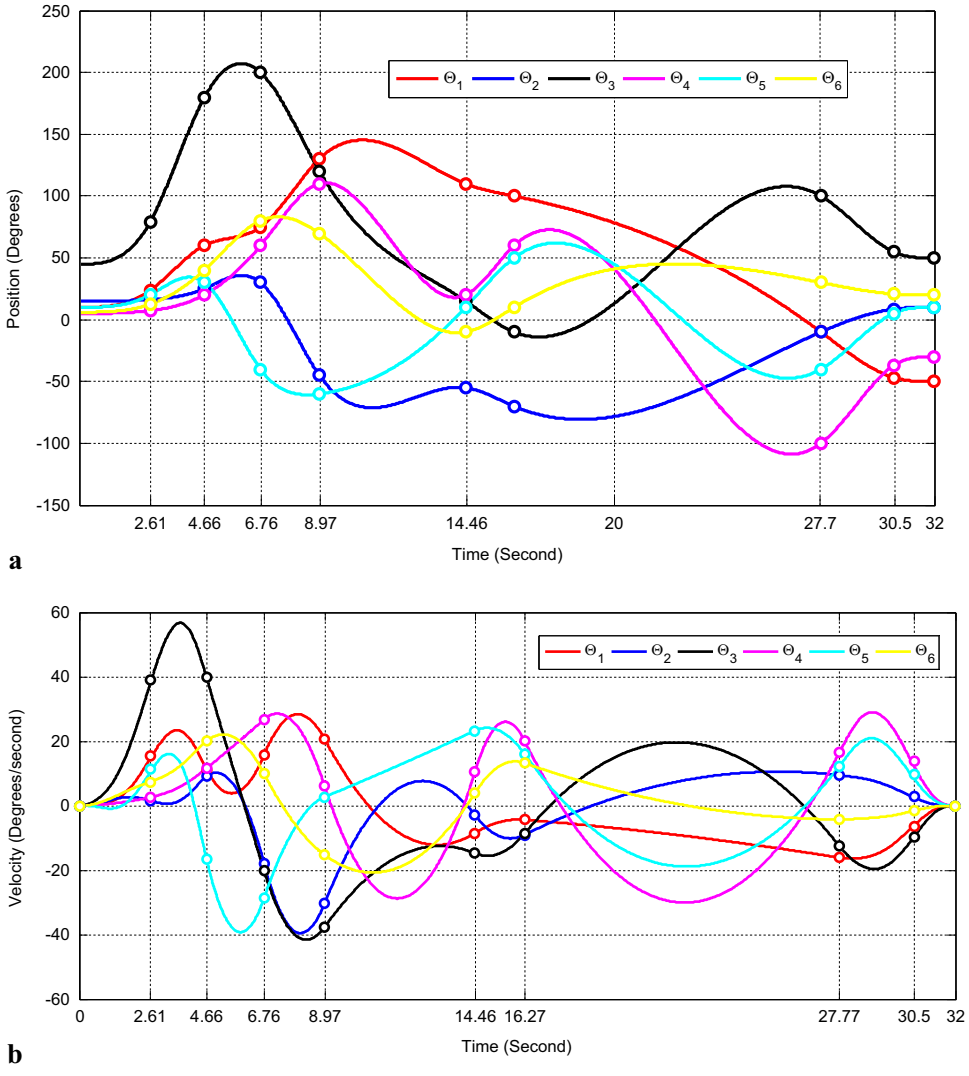
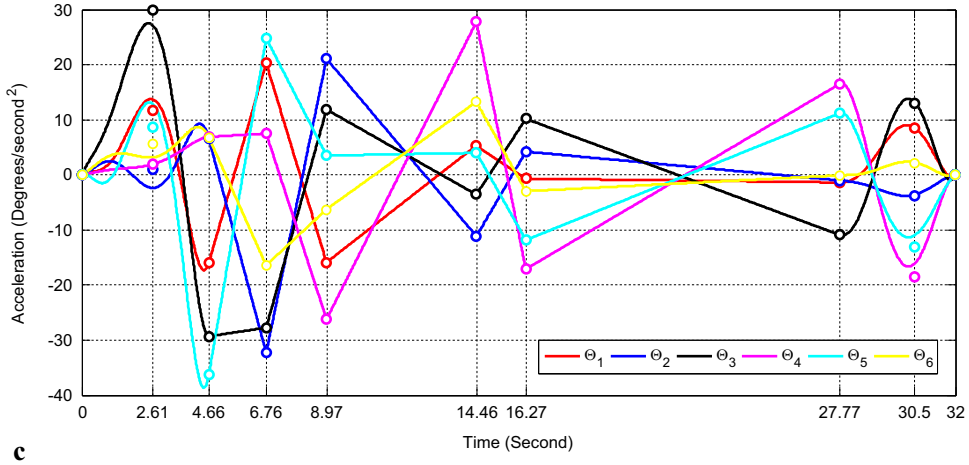


Fig. 8. a. Optimized joint positions. b. Optimized joint velocities. c. Optimized joint accelerations. d. Optimized joint jerks.



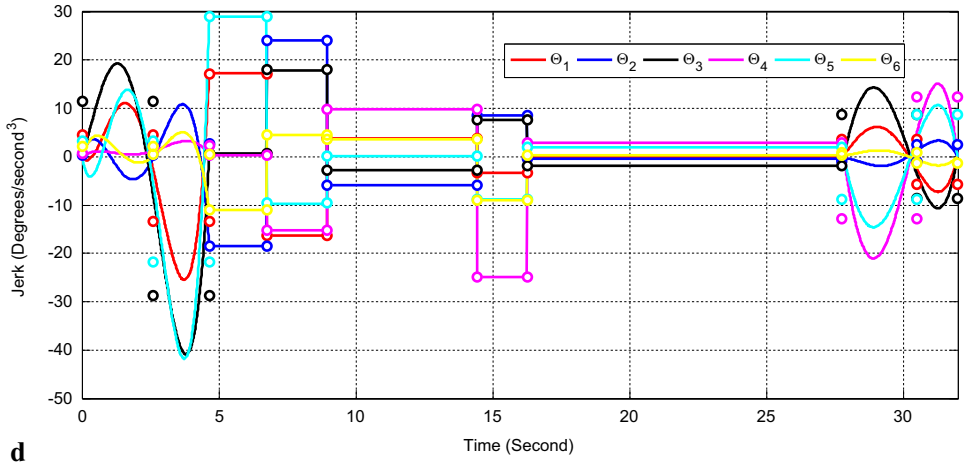


Fig. 8. (*continued*)

Table 4
Maximum joint velocities, accelerations and jerks obtained from optimization techniques.

Algorithm		$Joint_1$	$Joint_2$	$Joint_3$	$Joint_4$	$Joint_5$	$Joint_6$	Average
OTGA proposed in this paper	V_{max}	28.36	39.49	54.53	30.03	39.22	22.13	35.62
	A_{max}	20.29	32.27	29.95	27.78	36.18	16.44	27.15
	J_{max}	17.22	24.15	29.05	24.81	29.04	11.06	22.55
Gasparetto–Zanotto [8]	V_{max}	37.57	41.96	61.49	28.9	41.53	38.14	41.59
	A_{max}	39.07	43.65	65.60	15.87	33.88	39.94	39.66
	J_{max}	54.94	65.90	78.13	23.02	52.73	65.15	56.64
Simon–Isik [27]	V_{max}	39.84	47.67	57.54	28.67	44.29	43.00	43.50
	A_{max}	39.03	46.40	62.05	19.89	35.61	43.73	41.11
	J_{max}	54.82	59.28	80.84	25.99	49.04	61.47	55.24

Table 5
Mean joint velocity, acceleration and jerk values obtained from optimization techniques.

Algorithm		$Joint_1$	$Joint_2$	$Joint_3$	$Joint_4$	$Joint_5$	$Joint_6$	Average
OTGA proposed in this paper	V_{mean}	10.27	8.14	17.59	16.05	12.77	7.87	12.11
	A_{mean}	4.32	4.88	8.63	8.78	7.16	3.80	6.26
	J_{mean}	4.94	4.94	6.79	7.10	6.63	2.59	5.49
Gasparetto–Zanotto [8]	V_{mean}	16.94	20.7	27.45	15.38	15.92	19.47	19.31
	A_{mean}	19.08	18.15	30.60	6.50	14.92	21.84	18.51
	J_{mean}	26.26	20.69	41.26	6.39	18.18	30.35	23.85
Simon–Isik [27]	V_{mean}	16.87	22.11	26.41	15.38	16.76	20.24	19.62
	A_{mean}	19.48	19.99	30.07	6.53	15.02	23.52	19.10
	J_{mean}	27.51	23.97	41.27	9.10	18.07	33.77	25.61

platform linked to the ground by three independent kinematics chains [29]. The points O_1, O_2, O_3 and B_1, B_2, B_3 define the geometry of the base and the moving platform, respectively. The symbols θ_i and α_i illustrate the active and passive revolute joints, respectively where i=1, 2 and 3. The link lengths and the orientation of the moving platform are denoted by l_j and ϕ, respectively, j=1, 2, ⋯ ,6. P illustrates the point where the moving platform is positioned. A reference coordinate frame {xyz} is attached to the base for having forward and inverse kinematic equations easily as in [30].

A sample trajectory is designed for having 14 knots and 13 curve segments (h_1, h_2, ..., h_{12} and h_{13}). In addition to these 14 knots, two virtual points (v_1, v_2) between the first and last two knots as illustrated in Fig. 5 are added for satisfying the zero velocity and acceleration conditions. The 3-RRR planar parallel manipulator [29] starts the motion at point (p_x=12, p_y=12.5) and rests at point (p_x=12, p_y=13) for a period of 10.4 s. After performing manipulator between start and end-points under zero velocity and acceleration conditions, the velocity, acceleration and jerk profiles of 3-RRR planar parallel manipulator is obtained as in Fig. 6a, b and c, respectively.

As can be seen in Fig. 6b, although all the segments are having the same time bounds, the acceleration profiles of 3 joints are not equal. It is an expected result since angular displacements of the individual robot joints of 3-DOF RRR robot manipulator are not equal for each segment. Table 2 illustrates joint angle values at each knot where VS stands for virtual segment. For example, the angular displacements between the sixth and seventh knots for joint 1, 2 and 3 are not equal

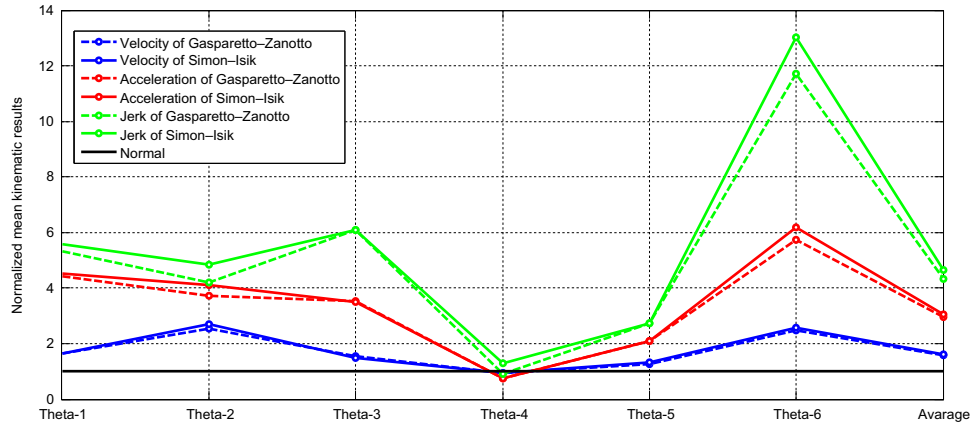


Fig. 9. Normalized mean kinematic results of the optimization technique.

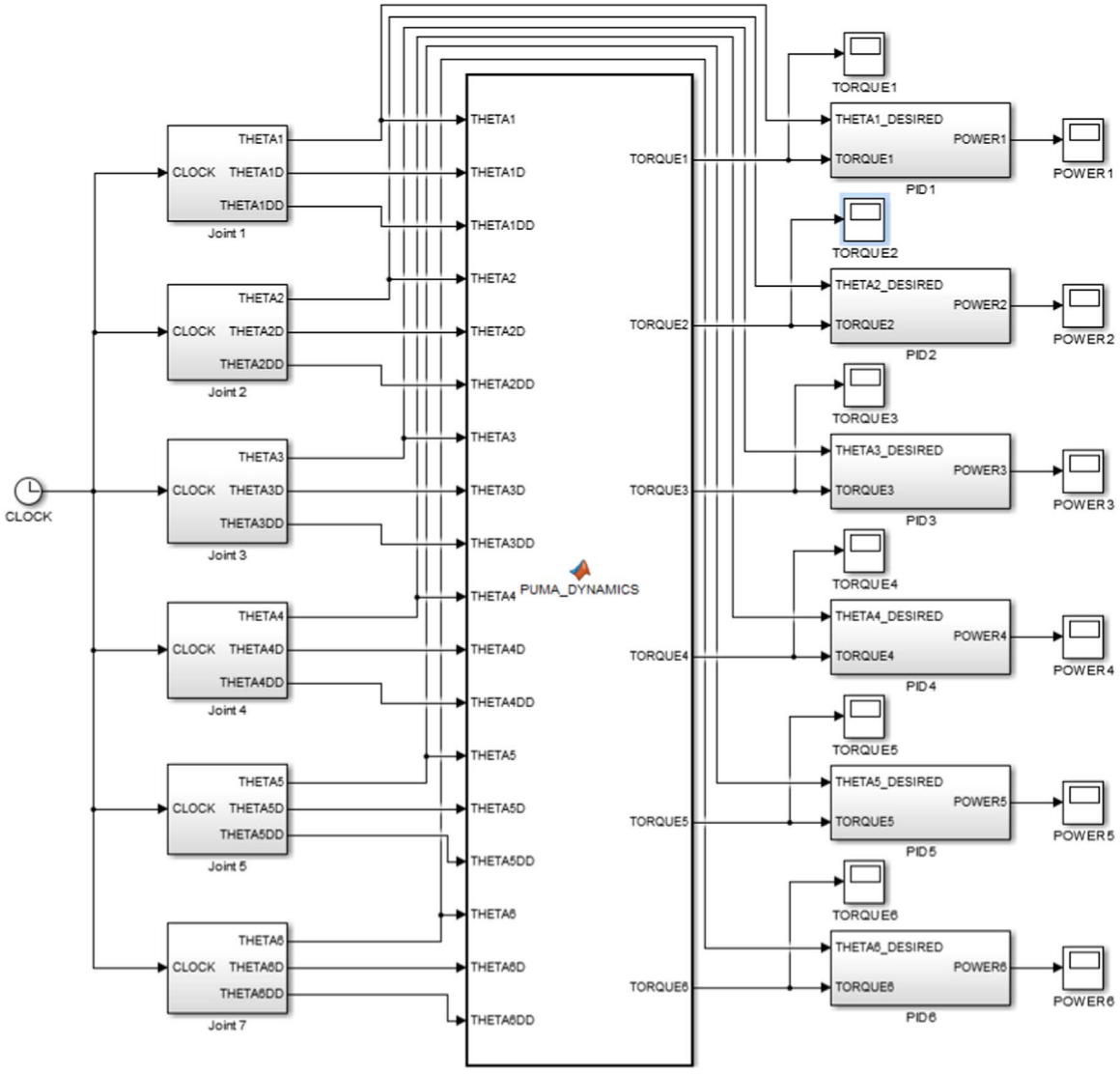


Fig. 10. A discrete-time PID control scheme designed for PUMA robot manipulator.

and measured as 3.1, 2.6 and 0.2 degrees, respectively.

As can be seen in Fig. 6b joint accelerations have some ripples in the first and the last knots and therefore manipulator has poor trajectory that is the main drawback of cubic splines [15]. An optimization has been performed in order to improve the results. Segment times between each pair of adjacent knots are used as design variables. The parameters of optimization algorithm (PSO) are chosen as follows: Population size is selected as 10 particles. Number of iterations is chosen as 250. The inertia weight is set to 0.76. Distributed random variables are used between 0 and 1. The positive acceleration constants are chosen as 2.05. The maximum and minimum values of the velocity, acceleration and jerk are designed between −30 and 30

Table 6
Some sample values of θ_i(Pos i)$\dot{\theta}_i$(Vel i)*and*$\ddot{\theta}_i$(Acc i) in radian.

Time	Pos 1	Vel 1	Acc 1	Pos 2	Vel 2	Acc 2	Pos 3	Vel 3	Acc 3
0.0000	0.1745	0.0000	0.0000	0.2617	0.0000	0.0000	0.7853	0.0000	0.0000
0.0100	0.1745	0.0000	0.0008	0.2618	0.0000	0.0001	0.7853	0.0000	0.0020
0.0200	0.1745	0.0000	0.0016	0.2618	0.0000	0.0001	0.7853	0.0000	0.0040
0.0300	0.1745	0.0000	0.0024	0.2618	0.0000	0.0002	0.7853	0.0001	0.0060
0.0400	0.1745	0.0001	0.0031	0.2618	0.0000	0.0003	0.7853	0.0002	0.0080
0.0500	0.1745	0.0001	0.0039	0.2618	0.0000	0.0003	0.7853	0.0002	0.0100
0.0600	0.1745	0.0001	0.0047	0.2618	0.0000	0.0004	0.7853	0.0004	0.0120
0.0700	0.1745	0.0002	0.0055	0.2618	0.0000	0.0005	0.7853	0.0005	0.0140
0.0800	0.1745	0.0003	0.0063	0.2618	0.0000	0.0006	0.7853	0.0006	0.0160
0.0900	0.1745	0.0003	0.0071	0.2618	0.0000	0.0006	0.7853	0.0008	0.0180
0.1000	0.1745	0.0004	0.0079	0.2618	0.0000	0.0007	0.7853	0.0010	0.0200
...	...	...	...	...	...	...	...	...	...
31.910	−0.8725	0.0004	0.0089	0.1745	0.0002	−0.0040	0.8725	−0.0006	0.0135
31.920	−0.8725	0.0003	0.0079	0.1745	0.0001	−0.0036	0.8725	−0.0005	0.0120
31.930	−0.8725	0.0002	0.0069	0.1745	0.0001	−0.0031	0.8725	−0.0004	0.0105
31.940	−0.8725	0.0002	0.0059	0.1745	0.0001	−0.0027	0.8725	−0.0003	0.0090
31.950	−0.8725	0.0001	0.0049	0.1745	0.0001	−0.0022	0.8725	−0.0002	0.0075
31.960	−0.8725	0.0001	0.0040	0.1745	0.0000	−0.0018	0.8725	−0.0001	0.0060
31.970	−0.8725	0.0000	0.0030	0.1745	0.0000	−0.0013	0.8725	−0.0001	0.0045
31.980	−0.8725	0.0000	0.0020	0.1745	0.0000	−0.0009	0.8725	−0.0000	0.0030
31.990	−0.8725	0.0000	0.0010	0.1745	0.0000	−0.0004	0.8725	−0.0000	0.0015
32.000	−0.8725	0.0000	0.0000	0.1745	0.0000	0.0000	0.8725	−0.0000	0.0000

degrees/s, degrees/second^2 and degrees/second^3 respectively. The maximum and minimum values of the segment times are designed between 0.5 and 0.8 s. The joint velocities, accelerations and jerks have been optimized as illustrated in Fig. 7a, b and c. After optimizing the total traveling time is decreased from 10.4 s to 9.868 s. The segment times are optimized as h_1 =0.553, h_2=0.545, h_3=0.751, h_4=0.695, h_5=0.774, h_6=0.643, h_7=0.711, h_8=0.583, h_9=0.500, h_{10}=0.799, h_{11}=0.532, h_{12}=0.713, h_{13}=0.667, h_{14}=0.715, h_{15}=0.687. The execution time of modified trajectory for initial and final segments are obtained as h_1 =0.553 and h_{15}=0.687, respectively. If the joint velocity, acceleration and jerk profiles on Fig. 6 (before optimization) and Fig. 7 (after optimization) are examined, the joint velocity, acceleration and jerk values of initial and final trajectory segments are notably decreased. For example, joint jerk values of the first joint for initial and final segments decreased from the interval [50–60] degrees/s^3 (Fig. 6d) to [10–20] degrees/s^3 (Fig. 7c), respectively.

Although the amplitudes of the joint velocities, accelerations and jerk and motion time decreased, the joint jerk is greater than zero at start-up and rest. In order to set the joint jerk to zero at the start-up and rest point, the cubic spline interpolation in the first and last knots of the optimized trajectory including virtual knots are changed with the 7th order polynomial joint trajectory interpolation. Instead of giving joint velocity, acceleration and jerk profiles of this modification, it is preferred to compare OTGA with two important trajectory algorithm developed earlier. This may illustrates efficiency of OTGA evidently.

OTGA proposed in this paper has been tested in simulation for PUMA robot manipulator which has been a common test platform for trajectory algorithms. The results are compared with the algorithms proposed by two important earlier studies [8,27]. The inverse kinematics has multiple solutions. The nature of trajectory differs on the combination of joint variables at the knot points. There are 8 different poses for PUMA robot manipulator for each knot point. If the knot points increase, the combinations of joint variables at the knot points will increase. Although PUMA robot manipulator has several inverse kinematics solutions, authors [8,27] have used the directly joint space values which have been obtained from one of the suitable inverse kinematics solutions of PUMA robot manipulators. They have not mentioned about the Cartesian Space values and the specific inverse kinematics solution of PUMA robot manipulators in their studies. Therefore the same joint space values as in [8,27] are also used in this study for making meaningful comparisons.

The joints space knots designed for test trajectory of PUMA robot manipulator is given in Table 3. The test trajectory composes of 8 knots and two virtual knots. The maximum and minimum values of the segment times are designed as the interval of [10–12] seconds for 7th segment and [1.5–6] seconds for the remaining segments.

The total execution time of the trajectory has been evaluated as 32 s in order to have meaningful comparisons between the results obtained from previous studies and proposed algorithms in this paper. After optimization, the joint positions, velocities, accelerations and jerks have been obtained as in Fig. 8a, b, c and d, respectively. The segment times are optimized as h_1=2.61, h_2=2.05, h_3=2.1, h_4=2.21, h_5=5.49, h_6=1.81, h_7=11.5, h_8=2.73, h_9=1.5. Note that 9 segment times have been obtained since the h_2 and h_8 compose of virtual segments. It can be noticed from Fig. 8d that the joint jerks have been obtained as zero both at start-up and rest.

Table 4 illustrates the maximum joint velocities, accelerations, jerks and their average values obtained from OTGA, Gasparetto and Zanotto [8] and Simon and Isik [27]. As can be seen from Table 4, most of the joint velocity, acceleration and jerk values have been obtained much lower than these of [8,27]. The entire average values of the maximum joint velocities, accelerations and jerks have been obtained lower than these of [8,27]. Especially, the average values of the jerks have been obtained about 2.5 times lower than these of [8,27]. It can be easily said that the results obtained from OTGA are much better than those presented in the earlier studies [8,27] in terms of jerk values.

Table 5 illustrates the mean kinematic values of the joint velocities, accelerations, jerks and their average values obtained from OTGA, [8,27]. Mean values are especially important since it illustrates the real performance of the system and reliability of the results obtained from a group of serial data. As can be observed from Table 5, almost all of the mean joint velocity, acceleration and jerk values have been obtained much lower than these of [8,27]. In order to compare the mean kinematic results precisely, a normalization has been performed between data given by Table 5.

Fig. 9 shows the mean kinematic results for the optimization technique proposed in this paper, normalized with these of the studies [8,27], as function of robot joints and average kinematic results which includes the average velocity, acceleration and jerk values. As can be seen from Fig. 9, OTGA provides minimum 295% and maximum 465.8% better mean kinematic results compared to the studies [8,27]. OTGA also provides minimum 124% & maximum 271% better velocity, minimum 208% & maximum 618% better acceleration, and minimum 128% & maximum 1303% better jerk results compared to

the studies [8,27] excepting fourth joint. OTGA is good at especially minimizing the joint jerk values that decrease joint position errors, limit excessive wear on the manipulator and prevent large oscillations [11–14]. Therefore it can be said that OTGA proposed in this paper is highly efficient for trajectory generation of robotic manipulators.

A discrete-time PID control scheme of PUMA robot manipulator given by Fig. 10 is also designed for comparing the energy consumption of the three algorithms. The electrical energy consumption of an active actuator can be defined as [31]

$$E = \int_0^T P_T dt \tag{6}$$

where P_T denotes the instantaneous electrical power at the instantaneous time t. P_T is the summation of resistive power "P_R", inductive power "P_L", and power producing the electromotor force "P_E", and it can be described as follow

$$P_T = R_a I_a^2 + L_a I_a \frac{dI_a}{dt} + V_e I_a \tag{7}$$

where $R_a, I_a, \quad V_e and L_a$ are the armature resistance, actuator current,

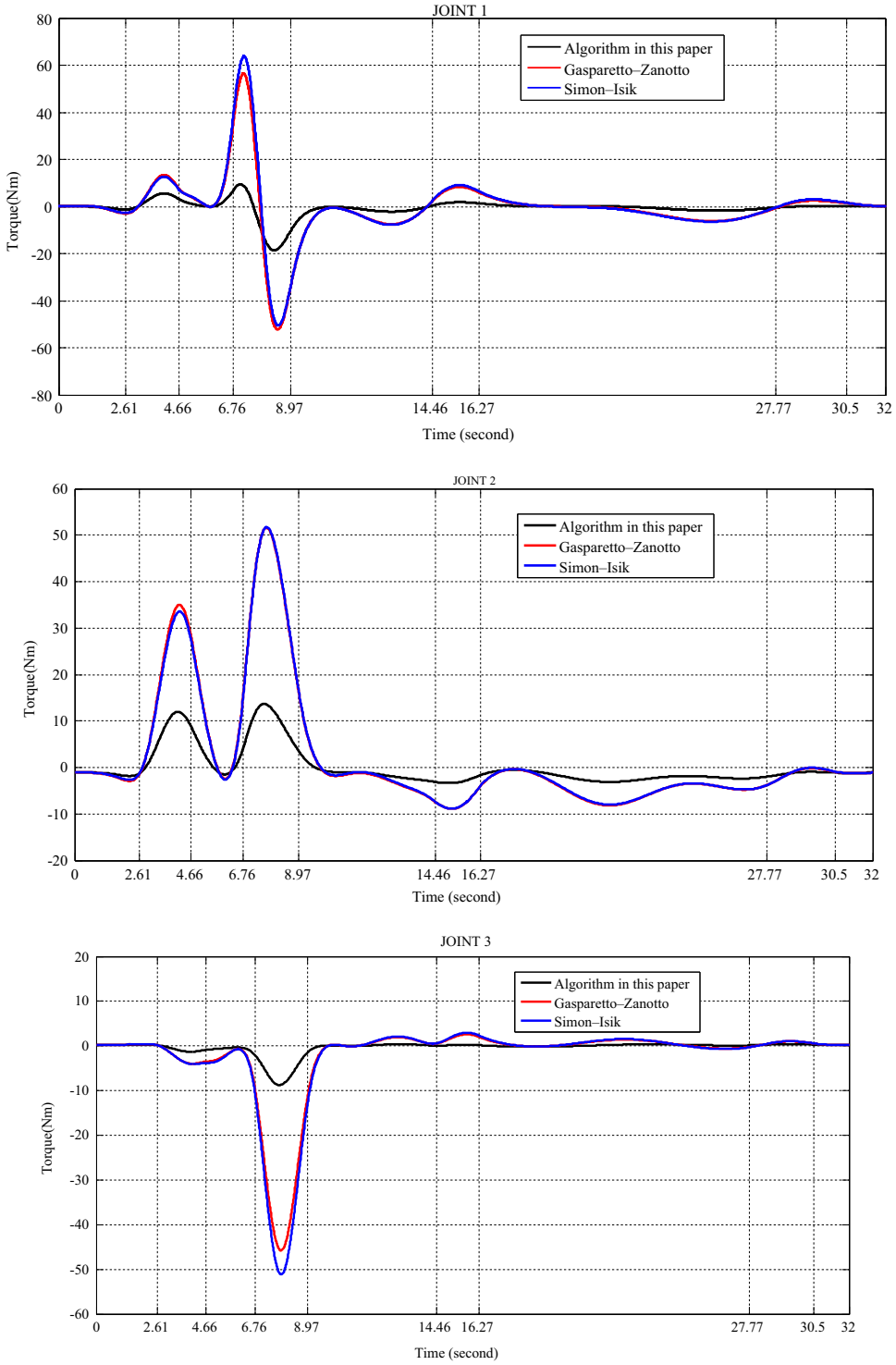


Fig. 11. Joint torques of PUMA robot obtained from three trajectory generation algorithms.

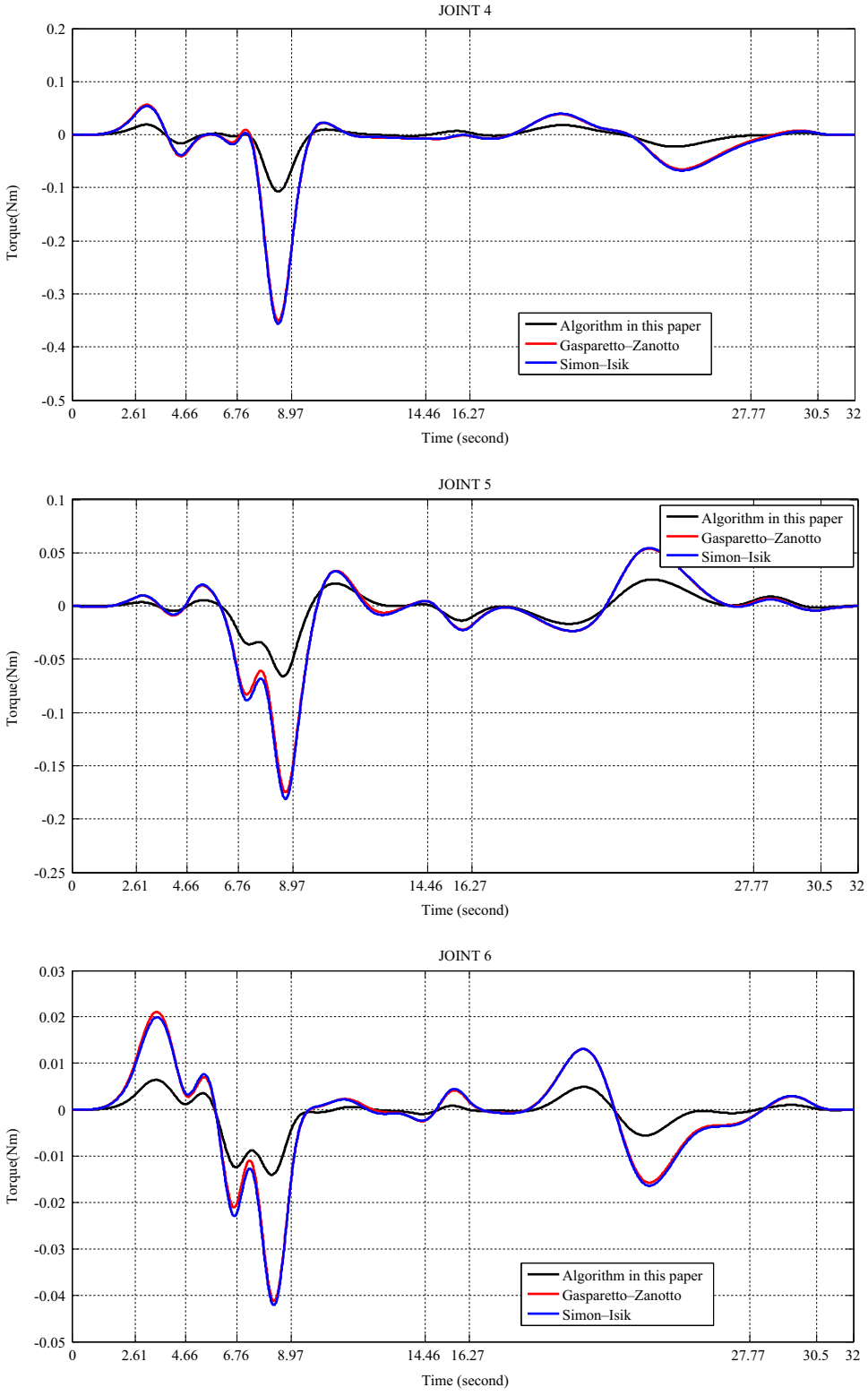


Fig. 11. (*continued*)

electromotor potential and armature inductance, respectively. The electrical energy consumed by the six actuators can be computed as

$$E_T = \sum_{j=1}^{6} E_j \quad \text{j} = 1,\ 2 \quad \ldots,\ 5\text{and}6. \qquad (8)$$

In Fig. 10, theta *i*, theta*i*d and theta*i*dd stand for the generated position (θ_i), velocity ($\dot{\theta}_i$) and acceleration ($\ddot{\theta}_i$) where i = 1, 2 …, 5and6. The theta*i* values are also the desired position values applied to the PID control block whose details can be found reference [29]. The theta*i*, theta*i*d and theta*i*dd are generated at 100 Hz frequency by using OTGA proposed in this paper. Some sample values of θ_i(Pos i)$\dot{\theta}_i$(Vel i)*and*$\ddot{\theta}_i$(Acc i) in radian are illustrated in Table 6 where *i*=1, 2 and 3.

Forward dynamics block includes the dynamic model of the PUMA robot manipulator. The dynamics equations are obtained by using

Table 7
Power consumption of the six actuator of PUMA robot manipulator for three algorithms.

Algorithms	Power consumption (Watt)						
	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Total
OTGA proposed in this paper	0.9153	0.5435	0.6808	0.2061	0.4444	0.2082	2.9983
Gasparetto–Zanotto [8]	5.0273	5.0374	3.7450	0.1542	0.3247	0.1346	14.4232
Simon–Isik [27]	5.5631	4.9327	4.4179	0.1542	0.3247	0.1346	15.5272

Lagrange-Euler method [32,33]. The joint torques of PUMA robot manipulator are computed as in Fig. 11 for Gasparetto–Zanotto [8], Simon–Isik [27] and the algorithm proposed in this paper. When the joint torques for Gasparetto–Zanotto [8] and Simon–Isik [27] algorithms are computed, the joint velocity and acceleration values of these algorithms are obtained by using the normalized mean kinematic results on Fig. 9. In addition the same kinematic and dynamic parameters (link lengths and mases) are used for having proper comparison.

The same trajectories given by Fig. 8 are also used in torque computations. The PUMA robot manipulator given by [13] is considered in the calculations. Note that the length a_3 is accepted zero in this study. In this case the link lengths of PUMA robot manipulator is accepted as a_2=0.20 m, d_3=0.05 m, d_4=0.15. Since Euler wrist intersects at a common point, there is not any link length used for last three joints of PUMA robot manipulator. However, small link masses are used for Euler wrist (m_4, m_5 and m_6,). The link masses are chosen as m_1=0.35, m_2=0.31, m_3=0.21, m_4=0.1, m_5=0.1, m_6=0.1.

As can be seen Fig. 11, joint torques obtained from OTGA are lower than these of [8,27]. The lower joint velocity and acceleration values produced by OTGA are the reason of having less joint torques.

Power consumption of PUMA robot for three trajectory generation algorithms are shown in Table 7. The same motor and PID parameters are used for having proper comparison. According to Table 7, the algorithm proposed in this paper consumes 79.212% less power than Gasparetto–Zanotto [8] and 80.69% less power than Simon–Isik [27] algorithms to perform the same trajectory tracking task. The lower joint velocity and acceleration values produced by the proposed algorithm cause this less power consumption that is very important for mass production in factories.

6. Conclusion

Minimum time, minimum torque and minimum jerk are general desired criteria for smooth motion trajectory planning of robotic manipulators in industry. Considering this fact, an optimal trajectory generation algorithm (OTGA) that generates minimum-time smooth motion trajectories for robotic manipulators is developed by using PSO algorithm in this paper. OTGA is developed into two stages, namely, obtaining minimum-time optimal trajectory using cubic spline and changing the cubic spline interpolation in the first and last knots of the optimized trajectory with the 7th order polynomial interpolation. Performing these two stages, designers will have minimum-time smooth motion trajectories with zero jerk at the beginning and end points of the trajectory. OTGA has also been tested in simulation for PUMA robot manipulator and the results are compared with two important algorithms proposed by the previous authors [8,27]. Comparison results can be summarized as follows:

i. The optimal trajectory generation algorithm (OTGA) successfully generates minimum-time smooth motion trajectories for robotic manipulators.
ii. Particle swarm optimization algorithm used for cubic spline optimization effectively reduced the joint velocity, acceleration and jerks of the manipulator trajectory.
iii. OTGA provides better mean kinematic results compared to previous the studies
iv. OTGA consumes less power than the algorithms in the literature to perform the same trajectory tracking task.
v. Almost all of the joint velocity, acceleration and jerk values are obtained much lower than previous studies.
vi. All of the average values of the maximum joint velocities, accelerations and jerks have been obtained lower than previous studies.
vii. The average values of the jerks have been obtained about 2.5 times lower than previous studies. The proposed algorithm is good at especially minimizing the joint jerk values.
viii. OTGA proposed in this paper provides minimum 295% and maximum 465.8% better mean kinematic results compared to previous the studies
ix. OTGA provides minimum 124% & maximum 271% better velocity, minimum 208% & maximum 618% better acceleration, and minimum 128% & maximum 1303% better jerk results than previous studies excepting fourth joint
x. OTGA consumes 79.212% less power than Gasparetto–Zanotto [8] and 80.69% less power than Simon–Isik [27] algorithms to perform the same trajectory tracking task.

References

[1] S.S. Perumaal, N. Jawahar, Automated trajectory planner of industrial robot for pick-and-place Task, Int. J. Adv. Robot. Syst. 10 (2013) 1–17.
[2] K. Erkorkmaz, Y. Altintas, High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation, Int. J. Mach. Tools Manuf. 41 (2001) 1323–1345.
[3] J.Y.S. Luh, C.S. Lin, Optimum path planning for mechanical manipulators, ASME Trans. J. Dyn. S p t. Meas. Control 103 (1981) 142–151.
[4] J.Y.S. Luh, M.W. Walker, R.P.C. Paul, On-line computational scheme for mechanical manipulators, ASME Trans. J. Dyn. Syst. Meas. Control 102 (1980) 69–76.
[5] C.S. Lin, P.R. Chang, J.Y.S. Luh, Formulation and optimization of cubic polynomial joint trajectories for industrial robots, IEEE Trans. Autom. Control 28 (12) (1983) 1066–1073.
[6] A. Piazzi, A. Visioli, Global minimum-jerk trajectory planning of robot manipulators, IEEE Trans. Ind. Electron. 47 (1) (2000).
[7] A. Visioli, Trajectory planning of robot manipulators by using algebraic and trigonometric splines, Robotica 18 (2000) 611–631.
[8] A. Gasparetto, V. Zanotto, A new method for smooth trajectory planning of robot manipulators, Mech. Mach. Theory 42 (4) (2007) 455–471.
[9] S. Macfarlane, E. Croft, Jerk-bounded manipulator trajectory planning design for real-time applications, IEEE Trans. Robot. Autom. 19 (2003) 42–52.
[10] E. Dyllong, A. Visioli, Planning and real-time modifications of a trajectory using spline techniques, Robotica 21 (2003) 475–482.
[11] W. Aribowo, K. Terashima, Cubic spline trajectory planning and vibration suppression of semiconductor wafer transfer robot arm, Int. J. Autom. Technol. 8 (2) (2014).
[12] K.J. Kyriakopoulos, G.N. Saridis, Minimum jerk path generation, in: Proc. IEEE Int. Conf. on Robot. Autom., Philadelphia, 364–369, 1988.
[13] J.J. Craig, Introduction to Robotics: Mechanics and Control, third ed., Pearson Prentice-Hall, New Delhi, 2005.
[14] D. Constantinescu, E.A. Croft, Smooth and time optimal trajectory planning for industrial manipulators along specified paths, J. Robot. Syst. 17 (5) (2000) 233–249.
[15] J.Z. Kolter, Y.Ng Andrew, Task-Space Trajectories via Cubic Spline Optimization, in: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1675–1682, 12-17 May, Kobe, 2009.
[16] B. Tondu, S.A. Bazaz, The three-cubic method: an optimal online robot joint trajectory generator under velocity, acceleration, and wandering constraints, Int. J. Robot. Res. 18 (9) (1999) 893–901.
[17] T. Chettibi, H.E. Lehtihet, M. Haddad, S. Hanchi, Minimum cost trajectory planning for industrial robot, Eur. J. Mech. – A/Solids 23 (4) (2004) 703–715.
[18] A. Gasparetto, V. Zanotto, A technique for time-jerk optimal planning of robot trajectories, Robot. Comput.-Integr. Manuf. 24 (3) (2008) 415–426.
[19] B. Demeulenaere, G. Pipeleers, J. De Caigny, J. Swevers, J. De Schutter, L. Vandenberghe, Optimal splines for rigid motion systems: a convex programming framework, J. Mech. Des. 131 (10) (2009) pp. 101004-101004-11.
[20] B. Sharma, S. Sehgal, A. Nain, Particle swarm optimization and genetic algorithm based optimal power flow solutions, Int. J. Appl. Innov. Eng. Manag. 2 (7) (2013).
[21] A. Olabi, R. Bearee, E. Nyiri, O. Gibaru, Enhanced Trajectory Planning For Machining With Industrial Six-Axis Robots, in: Proceedings of the IEEE

International Conference on Industrial Technology, pp. 500–506, Vi a del Mar, 14–17 march 2010, 2010.
[22] Serdar KUCUK, Maximal dexterous trajectory generation and cubic spline optimization for fully planar parallel manipulators, Comput. Electr. Eng. 56 (2016) 634–647.
[23] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks. NJ: Piscataway, pp.1942–1948, 1995.
[24] M. Toz, S. Kucuk, Dexterous workspace optimization of an asymmetric six-degree of freedom Stewart–Gough platform type manipulator, Robot. Auton. Syst. 61 (2013) 1516–1528.
[25] M. Toz, S. Kucuk, Dimensional optimization of 6-DOF 3-CCC type asymmetric parallel manipulator, Adv. Robot. 28 (9) (2014) 625–637.
[26] S. Kiranyaz, J. Pulkkinen, M. Gabbouj, Multi-dimensional particle swarm optimization in dynamic environments, Expert Syst. Appl. 38 (2011) 2212–2223.
[27] D. Simon, C. Isik, Optimal trigonometric robot trajectories, Robotica 9 (1991) 379–386.
[28] M. Boryga, A. Graboś, Planning of manipulator motion trajectory with higher-degree polynomials use, Mech. Mach. Theory 44 (2009) 1400–1419.
[29] S. Kucuk, Energy minimization for 3-RRR fully planar parallel manipulator using particle swarm optimization, Mech. Mach. Theory 62 (2013) 129–149.
[30] S. Kucuk, Simulation and design tool for performance analysis of planar parallel manipulators, Trans. Soc. Model. Simul. Int. 88 (5) (2012) 542–556.
[31] R. Ur-Rehman, S. Caro, D. Chablat, P. Wenger, Multiobjective path placement optimization of parallel kinematics machine based energy consumption, shaking forces and maximum actuator torques, Mech. Mach. Theory 45 (8) (2010) 1125–1141.
[32] R.P. Paul, Robot Manipulators: Mathematics, Programming, and Control, MIT Press, Cambridge Mass, 1981.
[33] M. Toz, S. Kucuk, Dynamics simulation toolbox for industrial robot manipulators, Comput. Appl. Eng. Educ. 18 (2) (2010) 319–330.