Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 3920 – 3931

doi:10.3906/elk-1608-143

Inverse kinematics of a 7-DOF redundant robot manipulator using the active set approach under joint physical limits

Modjtaba ROUHANI[1,*], **Sima EBRAHIMABADI**[2]
[1]Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[2]Department of Electrical Engineering, Islamic Azad University, Gonabad, Iran

Received: 12.08.2016 • **Accepted/Published Online:** 25.02.2017 • **Final Version:** 05.10.2017

Abstract: This paper presents a new approach for an online solution of the inverse kinematics problem based on nonlinear optimization for robots with joint physical constraints. The inverse kinematics problem is stated as a constrained nonlinear optimization problem and is solved using Kuhn–Tucker conditions analysis. The nonlinear multivariable optimization problem is locally converted into independent local linear constrained subproblems and each subproblem is analytically solved. For each joint, position limits and velocity limits are considered as physical constraints. The proposed structure benefits from a very low complexity design. The proposed method is fast and requires few calculations. The convergence of the proposed algorithm is proven based on the Lyapunov function. While keeping the algorithm stable, it can navigate the manipulator to a desired position under joint physical limits. The algorithm is simulated on a 7-DOF PA-10 manipulator. Results indicate good efficiency for the proposed structure in constrained path planning of joint space.

Key words: Inverse kinematics, redundant manipulator, active set, joint physical limits, Lyapunov analysis

1. Introduction

Serial manipulators are among the main tools employed in various industries, e.g., automobiles and aerospace. The main task of the manipulators is usually to move the end-effector along a predefined desired trajectory. Proper navigation toward the desired point requires necessary information about joint positions, enabling actuators to apply proper torques. There is usually no analytic method available to calculate joint angles as a function of end-effector positions and, if there are such methods, the solution is not unique. This is true especially in the case of redundant manipulators. Therefore, solving the inverse kinematics problem requires numerical algorithms. A large amount of research has previously been done and various numerical methods have been proposed to solve inverse kinematics of manipulators.

One of the most important and popular approaches is the Jacobian inverse-based method. The general basis of this method is to differentiate kinematics equations to obtain a relation between Cartesian velocity and joint space velocity through an inverse of the Jacobian matrix. An adaptation matrix for joint variables was proposed in [1–4] using the Lyapunov function, the Jacobian matrix, and its transpose. In addition to the Jacobian inverse approach, various approaches have been presented to solve inverse kinematics. In [5], a recursive approach based on manipulator geometry configuration is used. A recursive procedure is followed from the end-effector to the manipulator base and backwards. Yahya et al. [6] proposed a geometrical method that

*Correspondence: rouhani@um.ac.ir

assumes that all joint variables are constant except the first two. By using them as two independent variables in forward kinematics, the proper joint angles are determined. A combination of analytical and numerical solutions was used in [7]. First, one of the joint angles is assumed to be unknown and the rest to be known. Based on the robot position vector and using an iterative algorithm, the unknown variable is updated. However, this approach is time-consuming and requires numerous calculations. A number of research studies exist that focused on analytical solutions of inverse kinematics [8,9]. Considering neural network characteristics such as distributed and parallel computations, linear and nonlinear mapping, and environmental adaptation, they are widely used in the field of inverse kinematics [10–13]. In 2014, Toshani and Farrokhi [13] proposed a combination of an RBF neural network with quadratic programming to solve the inverse kinematics of a 7-DOF manipulator. The proposed approach aims to avoid joint position limits and obstacles available in the manipulator workspace and satisfies the required conditions to converge. In [14] and [15], recurrent neural networks were used for kinematics control of redundant manipulators with the obstacle avoidance capability of a 7-DOF manipulator. Such networks have simple structures and their convergences are guaranteed under proper conditions.

In this paper, joint angles are determined in order to minimize the tracking error of the end-effector using nonlinear programming. A gradient-like method is used to minimize tracking error. Joint physical position and velocity limits are separately imposed for each joint angle as linear constraints. Adaptation of the joint angles of the robot is done through the active-set method to satisfy those constraints. The solution to the Kuhn–Tucker conditions for the set of active constraints gives the appropriate joint angle deviations in each optimization step. A more detailed review of applications of nonlinear and quadratic programming in robotics can be found in the work of Escande et al. [16]. The paper itself uses the quadratic programming approach for equality and inequality constraints in a hierarchical way when some constraints have priorities over others. Toshani and Farrokhi [13] used the same approach while a neural network was learned for each robot joint to generate the desired angle deviations. However, inputs to neural networks were constant variables and learning was performed only in the output layer. As our approach shows here, instead of an indirect tuning of neural network weights, one can directly tune the desired angle deviation of each robot arm. This clearly simplifies the model and reduces the computational requirements for online applications. In addition, we considered both physical angle limits (up and down) and angular velocity limits (high and low) for each joint in our method.

The structure of the paper is as follows: in Section 2, basic ideas about the proposed method are stated and the required formulations are given. The overall algorithm for the joint angle adaptation procedure, based on the active-set method, is discussed in Section 3. Section 4 considers the convergence of the proposed method and proves that tracking error ultimately approaches zero, based on Lyapunov analysis. Simulation results are presented and analyzed in Section 5, and conclusions are given in Section 6.

2. Nonlinear optimization approach to inverse kinematics

Joint angles of the robot should be adapted in such a way that Cartesian tracking error is minimized while physical limits are satisfied. For the manipulator to navigate properly, based on the end-effector tracking error, a cost function is considered as follows:

$$J = \frac{1}{2}\left\|\mathbf{p}-\mathbf{p}_d\right\|^2 = \frac{1}{2}((x-x_d)^2+(y-y_d)^2+(z-z_d)^2) \tag{1}$$

$$\begin{cases} \mathbf{p} = [\ x \quad y \quad z\ ]^T \\ \mathbf{p_d} = [\ x_d \quad y_d \quad z_d\ ]^T \end{cases} \tag{2}$$

Here $\mathbf{p}$ is the Cartesian coordinate of the end-effector and $\mathbf{p}_d$ is the desired coordinate.

The active-set method is a powerful tool in dealing with constrained optimization problems, and quadratic programming and Kuhn–Tucker conditions are used to evaluate the constraints in each step as active or inactive and accordingly adapt the joint angles. To formulate the process, for each joint angle, the optimization problem is defined as:

$$\begin{array}{ll} \min & \frac{1}{2} q_i \Delta\theta_i^2 + c_i \, \Delta\theta_i \\ s.t. & \mathbf{E}_i^T \Delta\theta \leq \mathbf{r}_i \quad , i = 1, 2, ..., 7 \end{array} \tag{3}$$

Here $\Delta\theta_i \in R^1$ is the angle adaptation or search direction and c_i is the gradient of the cost function with respect to joint angle ($c_i = \frac{\partial J}{\partial \theta_i} \in \mathbf{R}$). Furthermore, $E_i \in R^{1\times 2}$ and $r_i \in R^{2\times 1}$ are coefficients driven from position and velocity limits as follows in Section 4, and q_i is a positive constant chosen by the designer. q_i sets the tradeoff between convergence and speed. A larger q_i results in smaller steps ($\Delta\theta_i$) and therefore slows down the algorithm, while smaller values for q_i lead to a faster algorithm, but the algorithm may not converge.

The recursive equation for updating the ith joint angles is:

$$\theta_{i,new} = \theta_{i,old} + \Delta\theta_i \tag{4}$$

Considering optimization of the problem in Eq. (3), the Lagrangian function for each joint of the robot is defined as follows:

$$L_i = \frac{1}{2} q_i \Delta\theta_i^2 + c_i \Delta\theta_i + \lambda_i^T \left(E_i^T \Delta\theta_i - r_i + s_i \right), \qquad i = 1, \ldots, 7 \tag{5}$$

Here λ and $\mathbf{s}$ are the Lagrange multiplier and positive shortage variables, respectively. By differentiating the Lagrangian function with respect to vectors λ_i and $\Delta\theta_i$, we can get:

$$\begin{cases} \frac{\partial L_i}{\partial \theta_i} = q_i \Delta\theta_i + c_i + \mathbf{E}_i \lambda_i = 0 \\ \frac{\partial L_i}{\partial \lambda_i} = \mathbf{E}_i^T \Delta\theta_i - \mathbf{r}_i + \mathbf{s}_i = 0 \end{cases} \tag{6}$$

$$\mathbf{E}_i \in R^{1\times 2}, \;\; \Delta\theta_i \in R^{1\times 1}, \;\; \mathbf{s}_i \lambda_i \in R^{2\times 1}, \;\; s_{ji} \lambda_{ji} q \geq 0, \; j = 1, 2$$

Reformulating Eq. (6) in matrix form yields:

$$\begin{bmatrix} q_i & \mathbf{E}_i & \mathbf{0}_{1\times 2} \\ \mathbf{E}_i^{\mathbf{T}} & \mathbf{0}_{2\times 2} & \mathrm{I}_{2\times 2} \end{bmatrix} \begin{bmatrix} \Delta\theta_i \\ \lambda_i \\ \mathbf{s}_i \end{bmatrix} = \begin{bmatrix} -c_i \\ \mathbf{r}_i \end{bmatrix}, \quad i = 1, 2, \ldots, 7 \tag{7a}$$

Each joint angle has two joint constraints, position and velocity, which makes vectors λ_i and s_i have two dimensions. Therefore, Eq. (7a) can be rewritten in the following form:

$$\begin{bmatrix} q_i & [\, E_{1i} & E_{2i}] & \mathbf{0}_{1\times 2} \\ [E_{1i} & E_{1i}]^T & \mathbf{0}_{2\times 2} & \mathbf{I}_{2\times 2} \end{bmatrix} \begin{bmatrix} \Delta\theta_i \\ \lambda_{1i} \\ \lambda_{2i} \\ s_{1i} \\ s_{2i} \end{bmatrix} = \begin{bmatrix} -c_i \\ r_{1i} \\ r_{2i} \end{bmatrix}, \quad i = 1, 2, \ldots, 7 \tag{7b}$$

According to the Kuhn–Tucker conditions, sufficient conditions for an optimal solution are:

a) $s_{ji}\lambda_{ji} \geq 0,\ j = 1, 2$

b) $\lambda_i^T \mathbf{s}_i = 0$

Those two conditions are met depending on the constraints being active or inactive. It should be remembered that s_{1i} and s_{2i} are shortage variables related to constraints on joint position and velocity, respectively. The active-set method is based on determining which constraints are active and which are inactive. Evaluation of a constraint as active or inactive is determined by the value of its shortage variable. That is, for a joint angle, if its shortage variable is positive, the corresponding constraint is inactive, and if its shortage variable is zero, it is active. If a constraint is active, then the corresponding Lagrange multiplier has to be zero, as indicated by Kuhn–Tucker conditions. Eq. (**??**) is a system of three linear equations of five unknowns: $\Delta\theta_i$, $\lambda_{\mathbf{i}} \in \mathbf{R^{2\times 1}}$, and $\mathbf{s_i} \in \mathbf{R^{2\times 1}}$. Provided that it is known which constraint is active and which is inactive, two of these unknowns will be zero, and the equation can be easily solved for the three remaining unknowns. To find out which constraint is active and which is inactive, four possible situations must be checked, and the one with a solution is the optimal point.

2.1. Joint angle and velocity constraints are inactive

In this case, both Lagrange multipliers of the ith joint angle are zero ($\lambda_{1i} = 0,\ \lambda_{2i} = 0$) and Eq. (**??**) is reduced to:

$$\begin{cases} q_i \Delta\theta_i = -c_i \\ E_{1i}\Delta\theta_i + s_{1i} = r_{1i} \\ E_{2i}\Delta\theta_i + s_{2i} = r_{2i} \end{cases}$$

The first equation can be readily solved for $\Delta\theta_i$:

$$\Delta\theta_i = -\frac{1}{q_i} c_i\ , \quad i = 1, 2, ..., 7 \tag{8}$$

The last two equations give the shortage variables:

$$\begin{cases} s_{1i} = r_{1i} - E_{1i}\Delta\theta_i \\ s_{2i} = r_{2i} - E_{2i}\Delta\theta_i \end{cases} \tag{9}$$

If both shortage variables are nonnegative (i.e. $s_{1i} \geq 0,\ s_{2i} \geq 0$), the Kuhn–Tucker conditions are satisfied, and the solutions resulting from Eqs. (8) and (9) are valid. If any of shortage variables are negative, the solutions resulting from Eqs. (8) and (9) are not feasible. Depending on which shortage variable(s) is negative, one of the following three cases will be met.

2.2. Joint position constraint is inactive and joint velocity constraint is active

In this case, the Lagrangian multiplier of the first constraint and the shortage variable of the second constraint ith joint angle are zero ($\lambda_{1i} = 0,\ s_{2i} = 0$), and Eq. (**??**) is reduced to:

$$\begin{cases} q_i \Delta\theta_i + E_{2i}\lambda_{2i} = -c_i \\ E_{1i}\Delta\theta_i + s_{1i} = r_{1i} \\ E_{2i}\Delta\theta_i = r_{2i} \end{cases}$$

The last equation can be readily solved for $\Delta\theta_i$:

$$\Delta\theta_i = \frac{r_{2i}}{E_{2i}} \tag{10}$$

The first two equations give the two remaining unknowns:

$$\begin{cases} \lambda_{2i} = \frac{-c_i - q_i\Delta\theta_i}{E_{2i}} \\ s_{1i} = r_{1i} - E_{1i}\Delta\theta_i \end{cases} \tag{11}$$

If $s_{1i} \geq 0$ and , $\lambda_{2i} \geq 0$, the Kuhn–Tucker conditions are satisfied and the solution resulting from Eqs. (10) and (11) are valid.

2.3. Joint position constraint is active and joint velocity constraint is inactive

In this case, the Lagrangian multiplier of the first constraint and shortage variable of the second constraint ith joint angle are zero ($\lambda_{2i} = 0$, $s_{1i} = 0$), and Eq. (**??**) is reduced to:

$$\begin{cases} q_i\Delta\theta_i + E_{1i}\lambda_{1i} = -c_i \\ E_{1i}\Delta\theta_i = r_{1i} \\ E_{2i}\Delta\theta_i + s_{2i} = r_{2i} \end{cases}$$

The second equation can be readily solved for $\Delta\theta_i$:

$$\Delta\theta_i = \frac{r_{1i}}{E_{1i}} \tag{12}$$

The first two equations give the two remaining unknowns:

$$\begin{cases} \lambda_{1i} = \frac{-c_i - q_i\Delta\theta_i}{E_{1i}} \\ s_{2i} = r_{2i} - E_{2i}\Delta\theta_i \end{cases} \tag{13}$$

If $s_{2i} \geq 0$ and $\lambda_{1i} \geq 0$, the Kuhn–Tucker conditions are satisfied and the solutions resulting from Eqs. (12) and (13) are valid.

2.4. Joint position limit and velocity limit are active

In this case, both shortage variables are zero ($s_{1i}s_{2i} = 0$) and Eq. (**??**) is rewritten as:

$$\begin{cases} q_i\Delta\theta_i + E_{1i}\lambda_{1i} + E_{2i}\lambda_{2i} = -c_i \\ E_{1i}\Delta\theta_i = r_{1i} \\ E_{2i}\Delta\theta_i = r_{2i} \end{cases}$$

The ith joint angle deviation $\Delta\theta_i$ can be calculated from each of the last equations:

$$\Delta\theta_i = \frac{r_{1i}}{E_{1i}} = \frac{r_{2i}}{E_{2i}} \tag{14}$$

The Lagrange multipliers $\lambda_{1i} \geq 0$ and $\lambda_{2i} \geq 0$ cannot be uniquely determined by the first equation and we then have:

$$E_{1i}\lambda_{1i} + E_{2i}\lambda_{2i} = -q_i\Delta\theta_i - c_i \tag{15}$$

As long as Eq. (15) has positive solutions for the Lagrange multipliers $\lambda_{1i} \geq 0$ and $\lambda_{2i} \geq 0$, the Kuhn–Tucker conditions are met and the solution that results from Eq. (14) is valid.

3. Formulation of the overall algorithm

Constraints imposed on the inverse kinematics problem are joint angle and velocity limits. Constraints imposed on joint position for each joint of the robot have the following form:

$$\begin{aligned}
&\theta_{i\min} \leq \theta_i \leq \theta_{i\max} \rightarrow \begin{cases} \theta_i - \theta_{i\max} \leq 0 \\ \theta_{i\min} - \theta_i \leq 0 \end{cases} ; \ \theta_{i\min} = -\theta_{i\max}; \\
&\rightarrow \begin{cases} \theta_i - \theta_{i\max} \leq 0 \\ -\theta_i - \theta_{i\max} \leq 0 \end{cases} \rightarrow |\theta_i| \leq \theta_{i\max} \rightarrow |\theta_i| - \theta_{i\max} \leq 0 \\
&\rightarrow h_{1i} = |\theta_i| - \theta_{i\max}; \ \ i = 1, 2, ..., 7
\end{aligned} \tag{16}$$

Here h_{1i} is the joint position constraint of the ith joint. Joint velocity constraints are analogously determined as follows:

$$\begin{aligned}
&\dot{\theta}_{i\min} \leq \dot{\theta}_i \leq \dot{\theta}_{i\max} \rightarrow \begin{cases} \dot{\theta}_i - \dot{\theta}_{i\max} \leq 0 \\ \dot{\theta}_{i\min} - \dot{\theta}_i \leq 0 \end{cases} ; \ \dot{\theta}_{i\min} = -\dot{\theta}_{i\max}; \\
&\rightarrow \begin{cases} \dot{\theta}_i - \dot{\theta}_{i\max} \leq 0 \\ -\dot{\theta}_i - \dot{\theta}_{i\max} \leq 0 \end{cases} \rightarrow \left|\dot{\theta}_i\right| \leq \dot{\theta}_{i\max} \rightarrow \left|\dot{\theta}_i\right| - \dot{\theta}_{i\max} \leq 0 \\
&\rightarrow h_{2i} = \left|\dot{\theta}_i\right| - \dot{\theta}_{i\max}; \ \ i = 1, 2, ..., 7
\end{aligned} \tag{17}$$

Therefore, $\mathbf{E}_i \in R^{1\times 2}$ and $\mathbf{r}_i \in R^{2\times 1}$ defined by Eq. (3) for the ith joint angle can be calculated as:

$$\begin{aligned}
&\mathbf{r}_i = \begin{bmatrix} b_{1i} \\ b_{2i} \end{bmatrix} = -\begin{bmatrix} h_{1i} \\ h_{2i} \end{bmatrix} = \begin{bmatrix} \theta_{i\max} - |\theta_i| \\ \dot{\theta}_{i\max} - \left|\dot{\theta}_i\right| \end{bmatrix} \geq 0 \ ; \\
&\mathbf{E}_i = \begin{bmatrix} \frac{\partial h_{1i}}{\partial \theta_i} & \frac{\partial h_{2i}}{\partial \theta_i} \end{bmatrix} = \begin{bmatrix} E_{1i} & E_{2i} \end{bmatrix}; \ i = 1, \ ..., \ 7,
\end{aligned} \tag{18}$$

where

$$\begin{aligned}
\frac{\partial h_{1i}}{\partial \theta_i} &= \frac{\partial \left(|\theta_i| - \theta_{i\max}\right)}{\partial \theta_i} = sign(\theta_i) \\
\frac{\partial h_{2i}}{\partial w_{ij}} &= \frac{\partial \left(\left|\dot{\theta}_i\right| - \dot{\theta}_{i\max}\right)}{\partial \dot{\theta}_i} = sign(\dot{\theta}_i);
\end{aligned} \tag{19}$$

and finally for c_i:

$$c_i = \frac{\partial J}{\partial \theta_i} = (x - x_d)\frac{\partial x}{\partial \theta_i} + (y - y_d)\frac{\partial y}{\partial \theta_i} + (z - z_d)\frac{\partial z}{\partial \theta_i} \tag{20}$$

The overall proposed algorithm is presented here:

Proposed Algorithm:
for $t = 1$ *to* t_{final}
  repeat
    calculate Cartesian error J
    for i=1 to 7
      calculate $\boldsymbol{E}_i, \boldsymbol{r}_i, c_i$
      calculate $\Delta\theta_i$ *by Eq. (8) and* s_{1i}, s_{2i} *using Eq. (9)*
      if $s_{1i} < 0$, $s_{2i} \geq 0$
        set $s_{1i} = 0$
        calculate $\Delta\theta_i$ *by Eq. (10) and* λ_{1i}, s_{2i} *using Eq. (11)*
      endif
      if $s_{1i} \geq 0$, $s_{2i} < 0$
        set $s_{2i} = 0$
        calculate $\Delta\theta_i$ *by Eq. (12) and* λ_{2i}, s_{1i} *using Eq. (13)*
      endif
      if $s_{1i} < 0$, $s_{2i} < 0$
        set $s_{2i}, s_{2i} = 0$
        set $\Delta\theta_i$ *to the minimum value determined by either Eq. (10) or Eq. (12)*
      endif
    endfor
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \eta\Delta\boldsymbol{\theta}$
  until Error$<\varepsilon$
endfor

The proposed algorithm decomposed the inverse kinematics problem of a 7-DOF robot to 7 joint level problems. For each joint, joint angle deviation $\Delta\theta_i$ is calculated by Kuhn–Tucker conditions. In the first case, both constraints are considered inactive and $\Delta\theta_i$ and $s_{1i}s_{2i}$ are calculated using Eqs. (8) and (9). If $s_{1i}s_{2i}$ are positive, the Kuhn–Tucker conditions are met. If not, either Eqs. (10) and (11) or Eqs. (12) and (13) determine the solution based on which s_{1i} or s_{2i} is negative. If both are negative, angle deviation must be set to the minimum value determined by Eq. (10) or (12) to satisfy the Kuhn–Tucker conditions. The procedure is iterated until end-effector error is minimized to a predefined level.

4. Convergence analysis

This section considers the convergence of the overall algorithm. To prove the convergence of the proposed algorithm, tracking error is considered as the Lyapunov function in Eq. (1). Differentiating J with respect to time, using Eq. (20), gives:

$$\dot{J} = \frac{dJ}{dt} = \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial t} = \sum_{i=1}^{7} c_i \frac{d\theta_i}{dt} \approx \frac{1}{\Delta t}\sum_{i=1}^{7} c_i \Delta\theta_i \tag{21}$$

To prove that tracking error will ultimately approach zero, it is sufficient to prove that $\dot{J} < 0$. To prove $\dot{J} < 0$, we show that $c_i\Delta\theta_i < 0$ for each joint angle $i = 1, \ldots, 7$.

In each instance, the joint angle variation of the ith joint $\Delta\theta_i$ is calculated with either Eq. (8), (10), or (12), based on which of the four cases discussed in Section 2 is active.

For case 1, where both joint angle and velocity constraints are inactive, $\Delta\theta_i$ is calculated with Eq. (8),

and we have:

$$c_i \Delta\theta_i = -\frac{1}{q_i} c_i^2 < 0 \tag{22}$$

This case is active if both shortage variables calculated with Eq. (9) are positive. If each of these quantities is not positive, $\Delta\theta_i$ has to be calculated with Eq. (10) or (12).

For the second case, when the joint position constraint is inactive and the joint velocity constraint is active, $\Delta\theta_i$ has to be calculated with Eq. (10) and the quantity of s_{2i} as calculated with Eq. (9) is negative; that is:

$$r_{2i} - E_{2i}\left(-\frac{c_i}{q_i}\right) = r_{2i} + \frac{1}{q_i} E_{2i} c_i < 0 \tag{23a}$$

Or equally:

$$E_{2i} c_i < -q_i r_{2i} \tag{23b}$$

Using Eq. (10) to calculate $\Delta\theta_i$ gives:

$$c_i \Delta\theta_i = c_i \frac{r_{2i}}{E_{2i}} = \frac{1}{E_{2i}^2} r_{2i} \left(E_{2i} c_i\right) < \frac{q_i}{E_{2i}^2}(-r_{2i}^2) < 0 \tag{24}$$

The corresponding analysis for the third case results in:

$$c_i \Delta\theta_i = c_i \frac{r_{1i}}{E_{1i}} = \frac{1}{E_{1i}^2} r_{1i} \left(E_{1i} c_i\right) < \frac{q_i}{E_{1i}^2}(-r_{1i}^2) < 0 \tag{25}$$

From Eqs. (22), (24), and (25), it is shown that $c_i \Delta\theta_i < 0$ in all cases.

5. Simulations results

To evaluate the efficiency of the proposed algorithm in determining joint angles, simulations are conducted on a 7-DOF PA-10 manipulator. The joint physical limits of this manipulator are presented in Table 1. The learning rate of joint angle adaptation is 0.5 and the algorithm termination criterion is set on a least squares error of 10^{-6}.

Table 1. Joint angle and velocity of the PA-10 robot.

Joint velocity limits (rad/s)	Joint angle limits (degrees)	Joint number
±1	±177	1
±1	±91	2
±2	±174	3
±2	±137	4
±2π	±255	5
±2π	±165	6
±2π	±360	7

The first desired path is considered to be a straight line starting from point (–0.2, 0.2, 0) and ending at point (0.4, 0.6, 0.5). Final configurations of the manipulator in Cartesian space under unconstrained and constrained conditions are shown in Figures 1a and 1b, respectively.

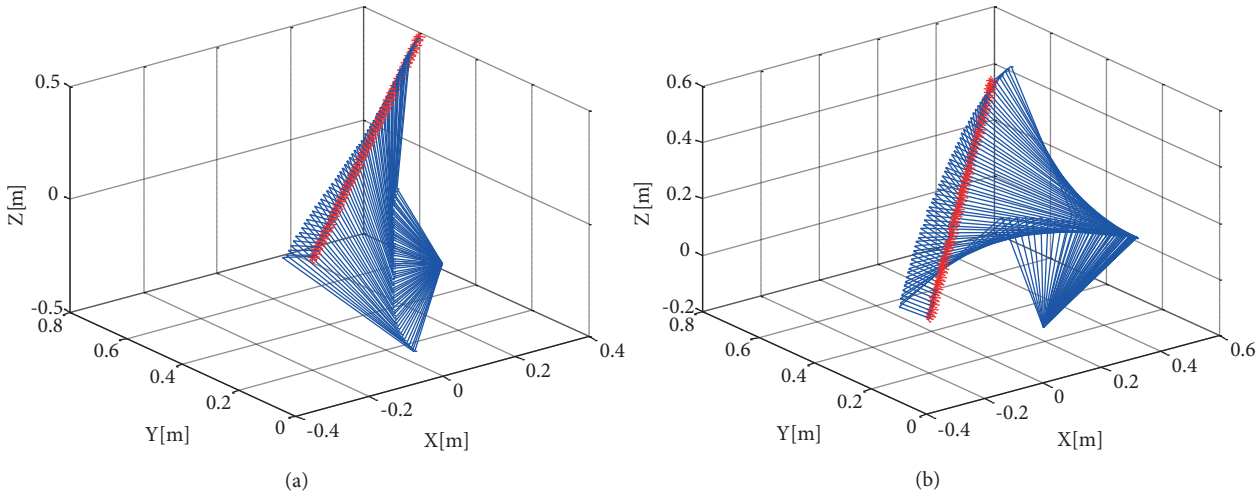

Figure 1. Final configurations of the robot while tracking the desired straight trajectory: a) unconstrained; b) constrained.

Final joint angles are shown under unconstrained and constrained conditions in Figures 2a and 2b, respectively. As seen in Figure 2a, the second joint angle deviates from its safe range at most points and from the fourth joint angle at some points, while by implementing the proposed algorithm, all manipulator joints remain in their safe ranges. Figures 3a and 3b show final angular velocity of the joints under unconstrained and constrained conditions, respectively. As can be seen in the figure, angular velocities of joints 2 and 4 deviate from their safe ranges, while the proposed algorithm efficiently keeps them in their safe ranges, properly implementing the redundancy resolution.

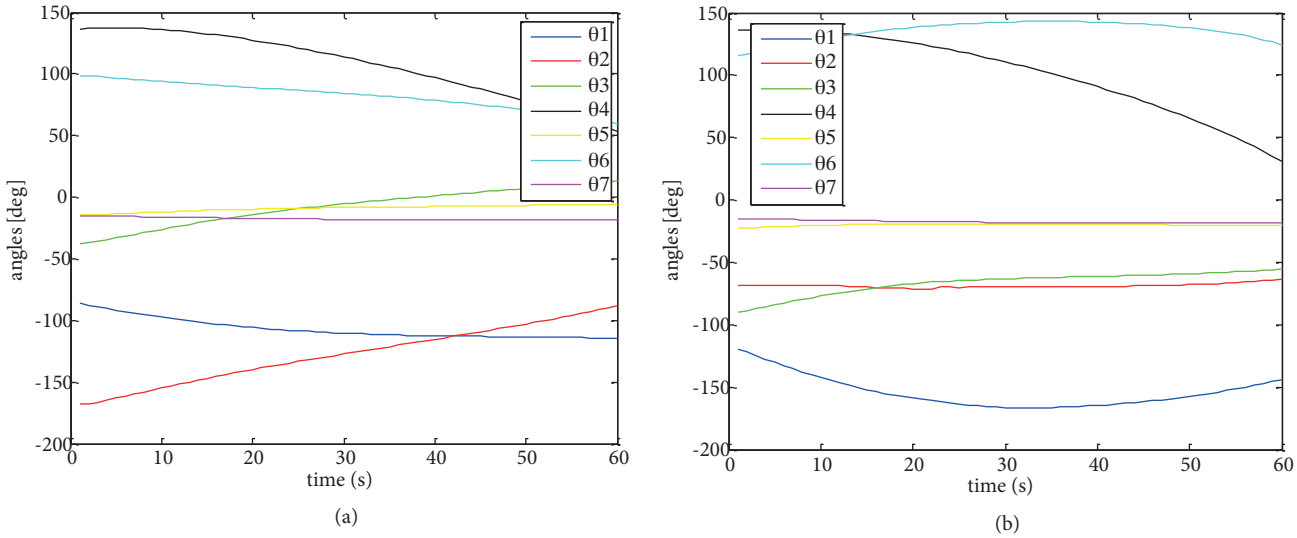

Figure 2. Final joint angles of the robot: a) unconstrained; b) constrained.

Results show that the proposed algorithm in this paper can generate a constrained trajectory planning in joint space of the manipulator so that high accuracy path tracking is carried out. The second trajectory is considered a circular path for which the radius is equal to 0.4, the center is located at point (0.2, 0.2), and the height is equal to 0.7. Figures 4a and 4b show the final configurations of the robot in unconstrained and

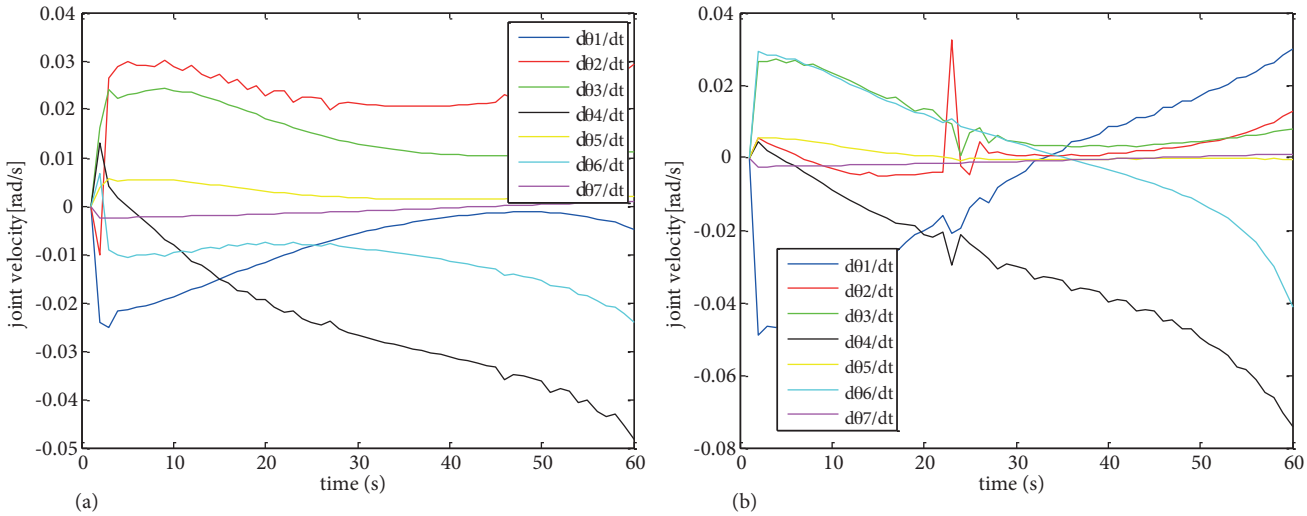

Figure 3. Final joint velocities of the robot: a) unconstrained; b) constrained.

constrained conditions, respectively. In addition, using an unconstrained algorithm, the four first angles of the robot have exceeded their velocity limitations (Figure 5a), while in the constrained process, all of the joints are within their physical ranges. Therefore, we can conclude that our proposed algorithm is able to solve inverse kinematics while considering physical joint limits.

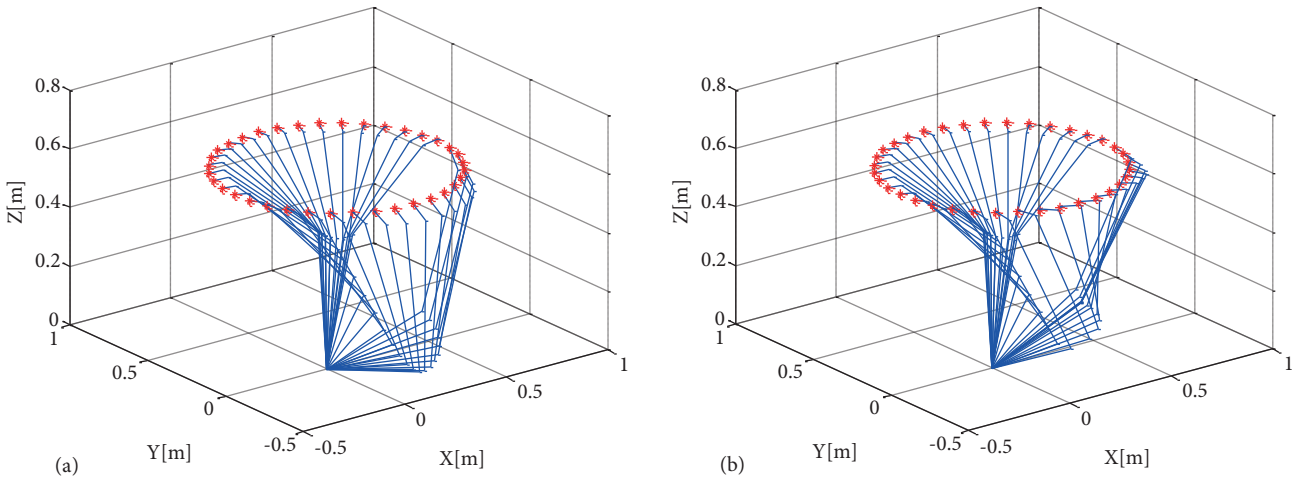

Figure 4. Final configurations of the robot while tracking the desired circular trajectory: a) unconstrained; b) constrained.

6. Conclusion

This paper proposed a numerical method to determine the joint angular position of a 7-DOF redundant manipulator subject to joint physical limits. The algorithm uses the active-set method for constrained optimization to solve inverse kinematics. For each joint, two constraints are considered: the position limit and the velocity limit. By decomposing the inverse kinematics problem to 7 joint level problems, appropriate angle deviation for each joint is determined by solving the Kuhn–Tucker conditions. The proposed algorithm is simple and requires few calculations.

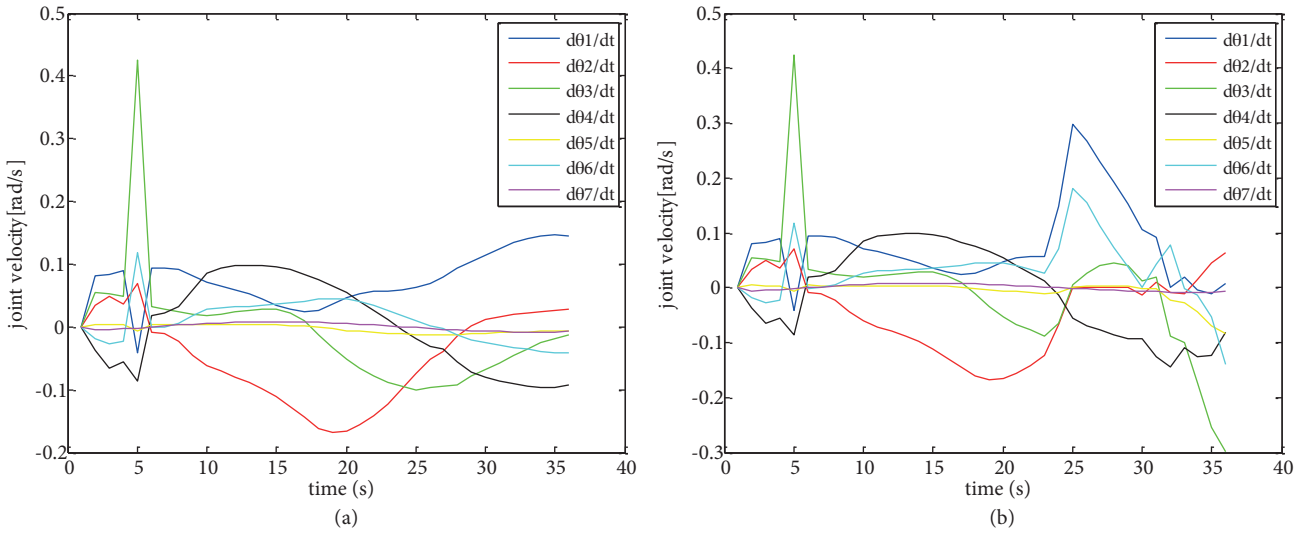

Figure 5. Final joint velocities of the robot: a) unconstrained; b) constrained.

Qualitative comparison of the proposed method to the Jacobian pseudoinverse [2], dual networks [14,17,18], and neural networks [13] is given in Table 2. The proposed method has a lower computational time than the method proposed by Toshani and Farrokhi [13] as both methods use nonlinear programming; however, the proposed method does not require neural weights to be updated. In the same way, the proposed method has the lowest number of parameters while the method of Toshani and Farrokhi has the advantage of independency in terms of robot structure.

Table 2. Comparison of the different methods.

Method/feature	Jacobian pseudoinverse [2]	Dual networks [14,17,18]	Neural networks [13]	Proposed method
Computational time	Medium	High	Low	Lowest
Stability analysis	No	Yes	Yes	Yes
Number of adjustable parameters	High	Low	Low	Lowest
Offline/online	Online	Online	Online	Online
Dependency to robot structure	Yes	Yes	No	Yes
Level of problem	Velocity	Velocity	Position	Velocity & position
Complexity	Low	Medium	Medium	Low

The proposed method is applied to a PA-10 robot and the tracking error of the end-effector shows the high accuracy of the proposed approach. In other words, the proposed algorithm can efficiently determine desired angular positions in the joint space of the robot while satisfying joint physical limits by properly setting constraints as active or inactive.

References

[1] Perdereau V. Real-time control of redundant robotic manipulators for mobile obstacle avoidance. Robot Auton Syst 2002; 41: 41-59.

[2] Wang J, Lee Y, Zhao, X. Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm. Advanced Robotic Systems 2010; 7: 1-10.

[3] Tchon K. Optimal extended Jacobian inverse kinematics algorithms for robotic manipulators. IEEE T Robot 2008; 24: 1440-1445.

[4] Ratajczak J. Design of inverse kinematics algorithms: extended Jacobian approximation of the dynamically consistent Jacobian inverse. Archives of Control Sciences 2015; 25: 35-50.

[5] Aristidou A, Lasenby J. FABRIK: A fast, iterative solver for the inverse kinematics problem. Graph Models 2011; 73: 243-260.

[6] Yahya S, Moghavvemi M, Mohamed H. Geometrical approach of planar hyper-redundant manipulators: Inverse kinematics, path planning and workspace. Simul Model Pract Th 2011; 19: 406-422.

[7] Kucuk S, Bingul Z. Inverse kinematics solutions for industrial robot manipulators with offset wrists. Appl Math Model 2014; 38: 1983-1999.

[8] Wei Y, Jian S, He S, Wang Z. General approach for inverse kinematics of nR robots. Mech Mach Theory 2014; 75: 97-106.

[9] Colom A, Torras C. Closed-loop inverse kinematics for redundant robots: comparative assessment and two enhancements. IEEE-ASME T Mech 2015; 20: 944-955.

[10] Bhattacharjee T, Bhattacharjee A. A study of neural network based inverse kinematics solution for a planar three joint robot with obstacle avoidance. Assam University Journal of Science and Technology 2010; 5: 1-7.

[11] Chiddarwar SS, Ramesh Babu N. Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach. Eng Appl Artif Intel 2010; 23: 1083-1092.

[12] Kumar K, Patel N, Behera L. Visual motor control of a 7 DOF robot manipulator using function decomposition and sub-clustering in configuration space. Neural Process Lett 2008; 28: 17-33.

[13] Toshani H, Farrokhi M. Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a Lyapunov-based approach. Robot Auton Syst s 2014; 62: 766-781.

[14] Zhang Y, Wang J. Obstacle avoidance for kinematically redundant manipulators using a dual neural network. IEEE T Syst Man Cyb B 2004; 34: 752-759.

[15] Zhang Y, Lv X, Li Z, Yang Z, Chen K. Repetitive motion planning of PA10 robot arm subject to joint physical limits and using LVI-based primal–dual neural network. Mechatronics 2008; 18: 475-485.

[16] Escande A, Mansard N, Wieber PB. Hierarchical quadratic programming: fast online humanoid-robot motion generation. Int J Robot Res 2014; 33: 1006-1028.

[17] Xia Y, Wang J. A dual neural network for kinematic control of redundant robot manipulators. IEEE T Syst Man Cyb 2001; 31: 147-154.

[18] Zhang Y, Wang J, Xu Y. A dual neural network for bi-criteria kinematic control of redundant manipulators. IEEE T Robotic Autom 2002; 18: 923-931.