Dynamic Neural Networks for Kinematic Redundancy Resolution of Parallel Stewart Platforms

Aquil Mirza Mohammed and Shuai Li, *Member, IEEE*

***Abstract*—Redundancy resolution is a critical problem in the control of parallel Stewart platform. The redundancy endows us with extra design degree to improve system performance. In this paper, the kinematic control problem of Stewart platforms is formulated to a constrained quadratic programming. The Karush–Kuhn–Tucker conditions of the problem is obtained by considering the problem in its dual space, and then a dynamic neural network is designed to solve the optimization problem recurrently. Theoretical analysis reveals the global convergence of the employed dynamic neural network to the optimal solution in terms of the defined criteria. Simulation results verify the effectiveness in the tracking control of the Stewart platform for dynamic motions.**

***Index Terms*—Constrained quadratic programming, kinematic redundancy, recurrent neural networks, Stewart platform.**

I. Introduction

KINEMATICALLY redundant manipulators are referred to those which have more degrees-of-freedom (DOFs) than required to determine the position and orientation. The redundancy of parallel manipulators can be utilized to overcome the obstacles, singularities [1], increasing workspace, improving dexterity and to optimize the performance and to achieve a smooth end-effector motion task [2]. As redundant robots have more DOF than required, there usually exist multiple solutions for kinematic control, which motivates us for the consideration of exploiting the extra DOFs to improve the control performance.

The inverse kinematics problem is one of the fundamental tasks in understanding the operability of parallel manipulators, i.e., to find the actuator inputs, provided the desired end-effector trajectories. Conventional design of the parallel mechanisms often encounter singularity problem. The design of redundancy in parallel mechanism often provides a effective remedy. Liu *et al.* [3] proposed a new 3-DOF symmetric spherical 3-universal, prismatic, and spherical joints/S parallel mechanism with three prismatic actuators, and studied the kinematics, statics, and workspace of the mechanism. In [3], a 2(spherical prismatic and prismatic joints+spherical prismatic, prismatic, and revolute joints+spherical prismatic, prismatic, and universal joints) serial-parallel manipulators was considered. Based on the analysis, they designed three new types of kinematically redundant parallel mechanisms, including a new redundant 7-DOF Stewart platform. In [4], the damped least square method was utilized to tackle singularity problem. However, it only modifies the end-effector path in terms of velocity. It was shown in [5] that it is important to analyze the inverse dynamic of parallel manipulators to find the joint friction and actuator dynamics.

Soft computing techniques, including neural networks [6]–[12], fuzzy logic [13], [14], and genetic algorithm [15], [16] have been extensive used for scientific computation, robotic control, and modeling. Due to the outstanding performance in parallel processing and recursive computation, dynamic neural networks, as a special type of neural networks, have long been employed as a powerful tool for the control of conventional serial robot arms. In [17], an adaptive recurrent neural network was employed for the control of a mobile manipulator with unknown dynamics. It was shown that the control strategy guarantees the asymptotical convergence to the desired motion. In [18], a unified quadratic-programming-based dynamical system approach was proposed to optimize the joint torque of serial manipulators. A minimum-energy redundancy resolution methodology was proposed in [19] for the recurrent neural networks control of serial manipulators. Periodic oscillations of the various neural networks were reported in this paper. In order to reduce the network complexity and further increase the computational efficiency, Xia and Jun [20] introduced a single layered dual neural network for the control of kinematically redundant manipulators. In [21] and [22], the recurrent neural network approach was extended to solve the cooperation of multiple manipulators organized in a distributed network.

Theoretical investigation of new types of neural models, and the applications of them to various real-world problems also experienced intensive research attention in past years. A simplified dual neural network architecture was proposed in [23] to reduce the structural complexity of conventional dual neural networks. In [24], dissipativity and stability properties were theoretically analyzed for a new type of memristor-based recurrent neural networks with time-varying delays. The synchronization of coupled neural networks in the presence of

Manuscript received November 11, 2014; revised March 24, 2015 and May 16, 2015; accepted June 26, 2015. Date of publication July 24, 2015; date of current version June 14, 2016. This work was supported in part by the NSFC under Grant 61401385 and in part by Hong Kong RGC ECS under Grant 25214015. This paper was recommended by Associate Editor Y. Xia. *(Corresponding author: Shuai Li.)*

The authors are with Department Computing, Hong Kong Polytechnic University, Hong Kong 07030 (e-mail: csmaquil@comp.polyu.edu.hk; shuaili@polyu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2451213

2168-2267 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

delayed feedback was discussed in [25]. A complete review on stability analysis of various continuous-time recurrent neural networks was presented in [26]. The generalization of real-valued recurrent neural networks to complex-valued neural dynamical approach was made in [27]. This approach significantly reduces redundant computation in a double real-valued space and features a low model complexity and storage capacity. In [28], the problem of blind data fusion was explored using a novel L_2 norm estimation method. Like most recent work on recurrent neural networks, this method also relies on an continuous dynamic model with a piecewise linear function as the activation function. A discrete-time recurrent neural network was proposed in [29] to solve constrained optimization problems with the presence of L_1 norm in the objective function, and was successfully applied to image restoration.

In spite of the great success of dynamic neural networks in both theory and application, there is rare report on using dynamic neural network to address the kinematic resolution problem of Stewart platform. In this paper, we make progress along this direction. The problem is formulated to a constrained quadratic programming in this paper. The Karush–Kuhn–Tucker (KKT) conditions of the problem is obtained by considering the problem in its dual space, and then we design a dynamic neural network to solve this problem recurrently. Due to the formulation of the problem from an optimization perspective, optimality in movement can be directly reached. Actually, for the Stewart platform, its forward kinematics are highly nonlinear and heavily coupled, which impose great challenges to the neural dynamic design to reach the same control goal. Additionally, due to the modeling of constraints as inequalities in the optimization problem, the obtained solution is direct in compliance with the physical constraints. Using convergence theories on projected neural networks, the global convergence of this dynamic neural network to the optimal solution is also proved rigorously. Simulation results verifies the effectiveness in the tracking control of the Stewart platform for dynamic motions. The contributions of this paper are threefold: first, to the best of our knowledge, this is the first attempt of using dynamic neural networks for kinematic redundancy resolution of parallel Stewart platforms. Second, in this paper, both minimization of the energy consumption and feasibility of physical constraints are taken into account. Accordingly, the obtained solution is both optimal and feasible. This is in contrast to other control methodology, which usually cannot guarantee the control action is within a feasible set. Third, different from feed-forward neural network, which usually suffer from local optimality, the dynamic neural approach employed in this paper is globally optimal.

The remainder of this paper is organized into eight sections. Section II describes the preliminaries of the Stewart platform. Section III provides the background information on the robotic kinematics of the constrained manipulators. Section IV describes the problem formulation as constrained optimization for the physically constrained manipulators. Section V provides a dynamic neural network model for the kinematic redundancy resolution problem of the parallel manipulator. Section VI presents the theoretical results on global optimality and dynamic convergence. Section VII illustrates the simulation results for position tracking and orientation tracking and their performance. Section VIII concludes this paper.

II. Preliminaries

The pose (position and orientation) of a rigid body in 3-D space is uniquely determined by a translation, represented by a 3-D vector, and a rotation, represented by a 3×3 rotational matrix in terms of three Euler angles. The rotational matrix $\Omega \in \mathbb{R}^{3\times 3}$ defined by the Euler angles $[\phi_x, \phi_y, \phi_z]^T \in \mathbb{R}^3$ has the following property for its time derivative:

$$\dot{\Omega}\Omega^T = \begin{bmatrix} 0 & -\dot{\phi}_z & \dot{\phi}_y \\ \dot{\phi}_z & 0 & -\dot{\phi}_x \\ -\dot{\phi}_y & \dot{\phi}_x & 0 \end{bmatrix}. \tag{1}$$

This property holds for all rotational matrices. Additionally, the rotational matrix is orthogonal, i.e., $\Omega\Omega^T = I$, $\Omega^{-1} = \Omega^T$ with I denoting a 3×3 identity matrix.

For two vectors $u = [u_1, u_2, u_3]^T \in \mathbb{R}^3$ and $v = [v_1, v_2, v_3]^T \in \mathbb{R}^3$, their cross-product, denoted as "$\times$," is defined as

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2v_3 - u_3v_2 \\ u_3v_1 - u_1v_3 \\ u_1v_2 - u_2v_1 \end{bmatrix}. \tag{2}$$

The triple product of three vectors in 3-D space is defined based on the cross product. For three vectors $u = [u_1, u_2, u_3]^T \in \mathbb{R}^3$, $v = [v_1, v_2, v_3]^T \in \mathbb{R}^3$, and $w = [w_1, w_2, w_3]^T \in \mathbb{R}^3$, their triple product is defined as $(u\times v)^T w$, and equals the following in value:

$$(u \times v)^T w = \det \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix}. \tag{3}$$

The triple product is invariant under circular shifting

$$(u \times v)^T w = (v \times w)^T u = (w \times u)^T v. \tag{4}$$

The matrix on the right-hand side in (1) is a 3×3 skew-symmetric matrix. For a general 3×3 skew-symmetric matrix, as can be simply verified, always holds for any $x, y, z \in \mathbb{R}$, $\alpha = [\alpha_1, \alpha_2, \alpha_3]^T \in \mathbb{R}^3$

$$\begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \alpha = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \times \alpha. \tag{5}$$

Due to the above relation, it is common to use $[x, y, z]^T_\times$ to represent a 3-D skew-symmetric matrix as

$$\begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \alpha = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_\times. \tag{6}$$

III. Robot Kinematics

The Stewart platform is a typical parallel mechanism and can be extended to different forms by modifying its mechanisms. It includes a mobile platform on the top as shown in

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

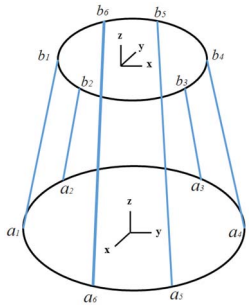


Fig. 1. Schematic of the Stewart platform.

Fig. 1, a fixed base, and six independent driving legs connecting the aforementioned two parts. The two ends of each leg are fixed on the mobile platform and the fixed based, respectively, using universal joints. Each leg can be actuated to change its length for the adjustment of the distance between the two fixed points on the platform and the base. All together, the six legs collaborates to adjust the orientation and position of the mobile platform by changing their lengths.

A. Geometric Relation

For the Stewart platform, the global coordinate is fixed on the base and the platform coordinate is fixed on the mobile platform. $a_i \in \mathbb{R}^3$ for $i = 1, 2, \ldots, 6$ represents the position in global coordinates of the ith connection point on the base. $b_i' \in \mathbb{R}^3$ for $i = 1, 2, \ldots, 6$ represents the position in platform coordinates of the ith connection point on the platform. We use b_i to represents its position in the global coordinate, as shown in Fig. 1. $d_i = b_i - a_i$ for $i = 1, 2, \ldots, 6$ represents the vector corresponding to the ith leg, which points from the base to the platform. For a point $x' \in \mathbb{R}^3$ in the platform coordinate, its position $x \in \mathbb{R}^3$ in global coordinate is obtained after a rotational and translational transformation

$$x = p + Qx' \tag{7}$$

where $p = [x_p, y_p, z_p]^T \in \mathbb{R}^3$ is the global coordinate of the zero position in the platform coordinate, and it corresponds to the translational transformation, $Q \in \mathbb{R}^{3\times 3}$ is the rotational matrix, which is uniquely defined by the Euler angles $\theta = [\theta_x, \theta_y, \theta_z]^T \in \mathbb{R}^3$

$$\begin{aligned} Q &= Q_z Q_y Q_x \\ Q_x &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \\ Q_y &= \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \\ Q_z &= \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_y & \cos\theta_y & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{8}$$

Following (7), as to the ith connection point on the platform, i.e., the ones with $x = b_i$ in the global coordinates or the ones with $x' = b_i'$ in the platform coordinates, we have:

$$b_i = p + Qb_i'. \tag{9}$$

Therefore, the ith leg vector can be further expressed as

$$d_i = b_i - a_i = p + Qb_i' - a_i. \tag{10}$$

For the vector d_i, we define $r_i = \|d_i\|$ to represent its length. Accordingly, we have

$$r_i = \left\|p + Qb_i' - a_i\right\|. \tag{11}$$

Notice that both a_i and b_i are constants and are determined by the geometric structure. $p = [x_p, y_p, z_p]^T$ defines the translation of the platform, and Q as a function of the Euler angles $\theta = [\theta_x, \theta_y, \theta_z]^T$, defines the rotation of the platform. Overall, the right-hand side of (11) depends on the pose variables of the platform $\pi = [x_p, y_p, z_p, \theta_x, \theta_y, \theta_z]^T \in \mathbb{R}^6$ while the left-hand side of (11) is the length of the leg, which is controlled for actuation. In this sense, (11) for $i = 1, 2, \ldots, 6$ defines the kinematic relation between the actuation variables and the pose variables. For a six-dimensional reference pose, the desired leg length r_i can be directly obtained from (11). However, in real applications, the reference are usually not six-dimensional. For example, for surgical applications of Stewart platform, people may only care about the position of an end-effector on the platform, instead of its orientation. In this situation, the reference is 3-D and we have three additional DOF as redundancy. For such a situation, we usually have infinite number of feasible solutions of r_i for $i = 1, 2, \ldots, 6$ to reach the reference. Among the feasible solutions, we may be able to identify one, which outperforms others in terms of certain optimization criteria. This intuitive analysis motivate us to model it as an optimization problem and identify the optimal one for improved performance. However, due to the nonlinearity of (11), direct treatment of (11) is technically prohibitive. Instead of direct solution in position space, we turn to solve the problem in its velocity space to exploit the approximate linearity.

B. Velocity Space Resolution

For easy treatment, (11) is rewritten as

$$r_i^2 = \left(p + Qb_i' - a_i\right)^T \left(p + Qb_i' - a_i\right). \tag{12}$$

To obtain the velocity space relations, we first compute time derivative on both sides of (12), which yields

$$\begin{aligned} r_i \dot{r}_i &= \left(p + Qb_i' - a_i\right)^T \left(\dot{p} + \dot{Q}b_i' + Q\dot{b}_i' - \dot{a}_i\right) \\ &= \left(p + Qb_i' - a_i\right)^T \left(\dot{p} + \dot{Q}b_i'\right). \end{aligned} \tag{13}$$

Recall that both a_i and b_i' are constants and their time derivatives, $\dot{a}_i$ and $\dot{b}_i'$, equal to zero. For the rotational matrix Q, according to the preliminary equations (1) and (6), it has the following property for its time derivative:

$$\dot{Q}Q^T = \begin{bmatrix} 0 & -\dot{\theta}_z & \dot{\theta}_y \\ \dot{\theta}_z & 0 & -\dot{\theta}_x \\ -\dot{\theta}_y & \dot{\theta}_x & 0 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix}_\times = \dot{\theta}_\times. \tag{14}$$

Therefore, $\dot{Q}$ writes as follows:

$$\dot{Q} = \dot{\theta}_\times \left(Q^T\right)^{-1} = \dot{\theta}_\times Q. \tag{15}$$

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

Substituting (15) into (13) yields

$$\begin{aligned} r_i\dot{r}_i &= \left(p + Qb_i' - a_i\right)^T\left(\dot{p} + \dot{\theta}_\times Qb_i'\right) \\ &= d_i^T\left(\dot{p} + \dot{\theta}_\times Qb_i'\right) \\ &= d_i^T\dot{p} + \left(\left(\dot{\theta}\times\left(Qb_i'\right)\right)^T d_i \\ &= d_i^T\dot{p} + \left(\left(Qb_i'\right)\times d_i\right)^T\dot{\theta} \\ &= \left[\begin{array}{cc} d_i^T & \left(\left(Qb_i'\right)\times d_i\right)^T \end{array}\right]\left[\begin{array}{c} \dot{p} \\ \dot{\theta} \end{array}\right]. \end{aligned} \tag{16}$$

In the above equation, (10) and (4) are used for the derivation in the second line and the derivation in the second last line, respectively. Noticing that $r_i = \|d_i\| > 0$ could be guaranteed by the mechanical structure, we have the following result:

$$\begin{aligned} \dot{r}_i &= \frac{1}{r_i}\left[\begin{array}{cc} d_i^T & \left(\left(Qb_i'\right)\times d_i\right)^T \end{array}\right]\left[\begin{array}{c} \dot{p} \\ \dot{\theta} \end{array}\right] \\ &= \frac{1}{r_i}\left[\begin{array}{cc} d_i^T & \left(\left(Qb_i'\right)\times d_i\right)^T \end{array}\right]\dot{\pi}. \end{aligned} \tag{17}$$

For the six-dimensional vector $r = [r_1, r_2, \ldots, r_6]^T$, we have the compact matrix form as follows:

$$\dot{r} = A_1\dot{\pi} \tag{18}$$

where

$$\begin{aligned} A_1 &= \left[\begin{array}{cc} \frac{1}{r_1}d_1^T & \frac{1}{r_1}\left(\left(Qb_1'\right)\times d_1\right)^T \\ \frac{1}{r_2}d_2^T & \frac{1}{r_2}\left(\left(Qb_2'\right)\times d_2\right)^T \\ \cdots & \cdots \\ \frac{1}{r_6}d_6^T & \frac{1}{r_6}\left(\left(Qb_6'\right)\times d_6\right)^T \end{array}\right] \\ &= \left[\begin{array}{cc} \frac{1}{r_1}\left(p + Qb_1' - a_1\right)^T & \frac{1}{r_1}\left(\left(Qb_1'\right)\times\left(p - a_1\right)\right)^T \\ \frac{1}{r_2}\left(p + Qb_2' - a_2\right)^T & \frac{1}{r_2}\left(\left(Qb_2'\right)\times\left(p - a_2\right)\right)^T \\ \cdots & \cdots \\ \frac{1}{r_6}\left(p + Qb_6' - a_6\right)^T & \frac{1}{r_6}\left(\left(Qb_6'\right)\times\left(p - a_6\right)\right)^T \end{array}\right]. \end{aligned} \tag{19}$$

Equation (18) gives the kinematic relation of a six-DOF Stewart platform from the velocity of the pose variables to the speed of the legs.

IV. Problem Formulation as Constrained Optimization

Compared to (11) and (18) significantly simplifies the problem as $\dot{r}$ in (18) is now affine to the $\dot{\pi}$ while the relation between r_i and π (or p and Q) in (11) are nonlinear, or even nonconvex to the pose variables. Similar to our analysis before, in the case that the reference pose velocity is given in six dimensions, the solution of $\dot{r}$ can be solved directly from (18). However, in the situation that the reference pose velocity is described in lower dimensions that six, the extra redundancies are available to reach improved performance. The following equality model the reference velocity constraint in reduced dimensions:

$$\alpha = A_2\dot{\pi} \tag{20}$$

where the reference vector $\alpha \in \mathbb{R}^m$ with $0 < m < 6$ is pregiven, the matrix $A_2 \in \mathbb{R}^{m\times 6}$ is the transformation matrix and is also pregiven. As an example, if we would like to maintain the platform at a given height, i.e., $\dot{p}_z = 0$, we set $\alpha = 0$ with $m = 1$ and $A_2 = [0, 0, 1, 0, 0, 0]^T$ in (20). Due to the extra design freedom, the value of $\dot{\pi}$ usually cannot be uniquely solved from (20). We thus define the following criteria to optimize the solution:

$$\min_{(\dot{\pi},\tau)} \frac{1}{2}\dot{\pi}^T\Lambda_1\dot{\pi} + \frac{1}{2}\tau^T\Lambda_2\tau \tag{21}$$

where $\Lambda_1 \in \mathbb{R}^{6\times 6}$ and $\Lambda_2 \in \mathbb{R}^{6\times 6}$ are both symmetric constant matrices and are both positive definite, $\tau = \dot{r}$ is the controllable speed of the platform legs. In application, the term $\dot{\pi}^T\Lambda_1\dot{\pi}$ can be used to specify the kinematic energy (including translational kinetic energy and rotational kinetic energy) by choosing proper weighting matrix Λ_1 (say choosing Λ_1 as one formed from the mass of the platform and its moment of inertia for the kinematic energy case), and the term $\tau^T\Lambda_2\tau$ characters the input power consumed by the robotic system. This objective function also follows the convention of control theory [30]. The decision variable τ, which is controlled by the actuators, is subjected to physical constraints. In this paper, we model the physical constraints as linear inequalities in the following form:

$$B\tau \le b \tag{22}$$

where $B \in \mathbb{R}^{k\times 6}$ and $b \in \mathbb{R}^k$ with k being an integer. Note that constraints are not imposed to the variable $\dot{\pi}$ since its value usually is specified as a feasible one in the planning stage. In summary of (21) as the object function, (18) as the mapping relation, (20) to fulfil the reference tracking requirements, and (22) as physical constraints, we can formulate the kinematic control problem of the Stewart platform as the following constrained programming:

$$\begin{aligned} \min_{(\dot{\pi},\tau)} \quad & \frac{1}{2}\dot{\pi}^T\Lambda_1\dot{\pi} + \frac{1}{2}\tau^T\Lambda_2\tau && \text{(23a)} \\ \text{s.t.} \quad & \tau = A_1\dot{\pi} && \text{(23b)} \\ & \alpha = A_2\dot{\pi} && \text{(23c)} \\ & B\tau \le b. && \text{(23d)} \end{aligned}$$

Due to the presence of both equation and inequality constraints in the optimization problem (23), usually it cannot be solved analytically. Conventional approaches introduces extra penalty terms formed by the constraints to the objective function and solve the problem numerically using gradient descent along the new objective function. However, penalty-based approaches only reaches an approximate solution of the problem and thus are not suitable for error-sensitive applications. Instead of using this approximate approach, in the next section, we will propose a dynamic neural network solution, which asymptotically converges to the theoretical solution.

V. Dynamic Neural Network Model

In this section, we first consider the optimization problem (23) in its dual space and then present a neural network to

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

solve it dynamically. After that, we investigate the hardware realization of the proposed model.

A. Neural Network Design

According to the KKT conditions [31], the solution of problem (23) satisfies

$$\Lambda_1\dot{\pi} - A_1^T\lambda_1 - A_2^T\lambda_2 = 0 \tag{24a}$$
$$\Lambda_2\tau + \lambda_1 + B^T\mu = 0 \tag{24b}$$
$$\tau = A_1\dot{\pi} \tag{24c}$$
$$\alpha = A_2\dot{\pi} \tag{24d}$$
$$\begin{cases}\mu > 0 & \text{if } B\tau = b\\ \mu = 0 & \text{if } B\tau < b\end{cases} \tag{24e}$$

where $\lambda_1 \in \mathbb{R}^6$, $\lambda_2 \in \mathbb{R}^m$ (m is the number of rows in matrix A_2), and $\mu \in \mathbb{R}^6$ are dual variables to the equation constraint (23b), the equation constraint (23c) and the inequality constraint (23d), respectively. The expression (24e) can be simplified to

$$\mu = (\mu + B\tau - b)^+ \tag{25}$$

where the nonlinear mapping "$(\cdot)^+$" is a function which maps negative values to zero and non-negative values to themselves. From (24a), $\dot{\pi}$ can be solved as

$$\dot{\pi} = \Lambda_1^{-1}\left(A_1^T\lambda_1 + A_2^T\lambda_2\right). \tag{26}$$

Substituting $\dot{\pi}$ in (26) to (24c) and (24d) yields

$$\tau = A_1\Lambda_1^{-1}\left(A_1^T\lambda_1 + A_2^T\lambda_2\right) \tag{27}$$
$$\alpha = A_2\Lambda_1^{-1}\left(A_1^T\lambda_1 + A_2^T\lambda_2\right). \tag{28}$$

To eliminate τ, we first represent it in terms of λ_1 and μ according to (24b) as

$$\tau = -\Lambda_2^{-1}\left(\lambda_1 + B^T\mu\right) \tag{29}$$

then substitute (29) into (24c) and (27), which results in

$$-\Lambda_2^{-1}\left(\lambda_1 + B^T\mu\right) = A_1\Lambda_1^{-1}\left(A_1^T\lambda_1 + A_2^T\lambda_2\right) \tag{30}$$
$$\mu = \left(\mu - B\Lambda_2^{-1}\lambda_1 - B\Lambda_2^{-1}B^T\mu - b\right)^+. \tag{31}$$

We use the following dynamics for the solutions of λ_1, λ_2, and μ in (28), (30), and (31):

$$\begin{aligned}\epsilon\dot{\lambda}_1 &= -\Lambda_2^{-1}\left(\lambda_1 + B^T\mu\right) - A_1\Lambda_1^{-1}\left(A_1^T\lambda_1 + A_2^T\lambda_2\right)\\ \epsilon\dot{\lambda}_2 &= -A_2\Lambda^{-1}A_1^T\lambda_1 - A_2\Lambda^{-1}A_2^T\lambda_2 + \alpha\\ \epsilon\dot{\mu} &= -\mu + \left(\mu - B\Lambda_2^{-1}\lambda_1 - B\Lambda_2^{-1}B^T\mu - b\right)^+\end{aligned} \tag{32}$$

where $\epsilon > 0$ is a scaling factor. Overall, the proposed dynamic neural network has λ_1, λ_2, and μ as state variables and τ in (27) as the output, which is expressed as follows in summary.

State equations

$$\epsilon\dot{\lambda}_1 = -\Lambda_2^{-1}\lambda_1 - A_1\Lambda_1^{-1}A_1^T\lambda_1 - A_1\Lambda_1^{-1}A_2^T\lambda_2 - \Lambda_2^{-1}B^T\mu \tag{33a}$$
$$\epsilon\dot{\lambda}_2 = -A_2\Lambda^{-1}A_1^T\lambda_1 - A_2\Lambda^{-1}A_2^T\lambda_2 + \alpha \tag{33b}$$
$$\epsilon\dot{\mu} = -\mu + \left(-B\Lambda_2^{-1}\lambda_1 + \mu - B\Lambda_2^{-1}B^T\mu - b\right)^+. \tag{33c}$$

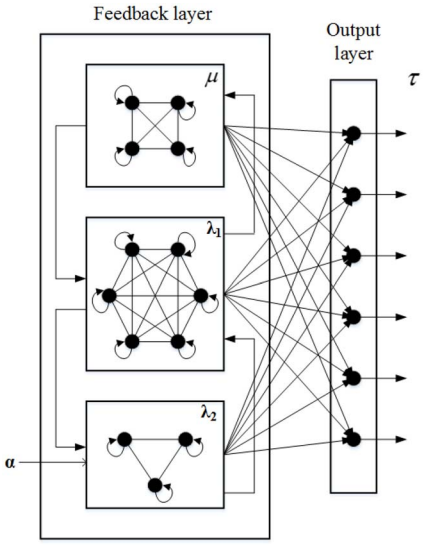


Fig. 2. Two layer neural network architecture.

Output equation

$$\tau = A_1\Lambda_1^{-1}A_1^T\lambda_1 + A_1\Lambda_1^{-1}A_2^T\lambda_2. \tag{33d}$$

About the proposed neural network model (33) for the kinematic redundancy resolution problem (23) of parallel manipulator, we have the following remark.

Remark 1: Fig. 2 shows the architecture of the proposed model, for the situation with $m = 3$, $k = 4$ (m is the dimension of α, k is the number of rows of matrix B). From this figure, it is clear that this neural network (33) is organized in a two-layer architecture. The first layer(feedback layer), which is composed of $6 + m + k$ neurons, is a nonlinear layer with dynamic feedback. This layer of neurons is associated with the state variables $\lambda_1 \in \mathbb{R}^6$, $\lambda_2 \in \mathbb{R}^m$, and $\mu \in \mathbb{R}^k$, and get input from the input variable α. It follows (33a), (33b), and (33c) for dynamic updates. The second layer (output layer) is a static layer with linear mapping as described by (33d). It consists of six neurons and maps the state variables to the output, which is the platform leg speed τ.

In comparison with various feed-forward neural network models, such backpropagation neural network, radial basis function neural network, extreme learning machine, the neural network investigated in this paper falls into the category of recurrent neural networks with dynamic feedbacks. Different from the Kennedy–Chua network [32], which represent constraints as a extra penalty, the neural network model of this paper consider the problem in its dual space and is able to reach optimal solution. The general purpose projected neural network [33] only has a dynamic layer, while the neural network employed in this paper is organized in a two-layer structure, where the first layer is a dynamic layer with self-feedback, and the second layer is a static one. Projected neural networks are widely used to solve kinematic control of redundant serial manipulators. The dynamic neural network employed in [34] to solve kinematic redundancy in robot motion control also has a two-layered architecture. However, in that model, the hidden layer nodes has an all-to-all connection, while the hidden nodes are

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISONS OF DIFFERENT METHODS FOR KINEMATIC CONTROL OF STEWART PLATFORMS

	Optimality	Online v.s. Off-line	Pre-training	Compliance to extra constraints	Precision
Feed-forward neural network approach [37]	No	Off-line	Yes	No	Low
Self organization based approach [38]	No	Off-line	Yes	No	Low
Sliding mode based approach [39]	No	Online	No	No	High
Fuzzy logic based approach [40]	No	Online	No	No	High
This paper	Yes	Online	No	Yes	High

partitioned into three cliques in the neural model used in this paper, where neurons in each clique are connected in an all-to-all manner but the connections between different cliques are local (e.g., the clique associated with the state variable μ is not connected with that associated with the state variable λ_2). The property of the local connectivity between hidden cliques also distinguish this model from other ones using dynamic neural networks for manipulator control [10], [19], [20].

VI. THEORETICAL RESULTS

In this section, we present theoretical results on the proposed neural networks for solving the redundancy resolution problem of parallel manipulators.

A. Optimality

In this part, we show the equilibrium point of the dynamic neural networks (33) ensures that the corresponding output τ given by (33d) is identical to the optimal solution of the problem (23). On this point, we have the following theorem.

Theorem 1: Suppose $(\lambda_1^*, \lambda_2^*, \mu^*)$ is the equilibrium point of the dynamic neural network (33). Then the corresponding output τ^* obtained from the output equation (33d) is optimal to the constrained programming problem (23).

Proof: Letting the right-hand sides of the state equations (33a), (33b), and (33c) equal to zero, we find the following conditions about the equilibrium point $(\lambda_1^*, \lambda_2^*, \mu^*)$:

$$-\Lambda_2^{-1}\lambda_1^* - A_1\Lambda_1^{-1}A_1^T\lambda_1^* - A_1\Lambda_1^{-1}A_2^T\lambda_2^* - \Lambda_2^{-1}B^T\mu^* = 0 \tag{34}$$

$$-A_2\Lambda^{-1}A_1^T\lambda_1^* - A_2\Lambda^{-1}A_2^T\lambda_2^* + \alpha = 0 \tag{35}$$

$$-\mu^* + \left(-B\Lambda_2^{-1}\lambda_1^* + \mu^* - B\Lambda_2^{-1}B^T\mu^* - b\right)^+ = 0 \tag{36}$$

and the corresponding output is

$$\tau^* = A_1\Lambda_1^{-1}A_1^T\lambda_1^* + A_1\Lambda_1^{-1}A_2^T\lambda_2^*. \tag{37}$$

Define an auxiliary value

$$\dot{\pi}^* = \Lambda^{-1}A_1^T\lambda_1^* + \Lambda^{-1}A_2^T\lambda_2^*. \tag{38}$$

To show that τ^* is optimal to (23), we only need to show that $(\lambda_1^*, \lambda_2^*, \mu^*, \tau^*, \dot{\pi}^*)$ satisfy the KKT condition (24) of the optimization problem (23), or equivalently the equation set composed of (26)–(28), (30), and (31) according to the analysis in Section V-A, we can conclude that the equation set composed of (26)–(28), (30), and (31) is equivalent to the KKT condition (24). Comparing the equation set composed of (34)–(38), and the one composed of (26)–(28), (30), and (31), we find that they are identical and therefore are equivalent. The above procedure implies that the solution $(\lambda_1^*, \lambda_2^*, \mu^*, \tau^*, \dot{\pi}^*)$ is optimal to (23). Therefore, we conclude that τ^* is optimal to the problem (23). ■

B. Stability

In this part, we present theoretical results on the stability of the proposed dynamic neural network model. In Section VI-A, we have concluded that the equilibrium point of the neural network (33) is optimal solution of (23). Generally speaking, a dynamic system may not converge to its equilibrium points. It may happen for a dynamic system to evolve toward divergence, oscillation, or even chaos. It is necessary for the proposed neural network to converge for effective computation purpose. Before presenting the convergence results, we first present a lemma about a general projected dynamic system as

$$\dot{u} = -u + P_\Omega(u - F(u)) \tag{39}$$

where $\mu \in \mathbb{R}^l$, Ω is a closed convex set of $\mathbb{R}^l$, $P_\Omega(\cdot)$ is the projection operator onto the set Ω.

Lemma 1 [33]: If $\nabla F(u)$ is symmetric and positive semidefinite in $\mathbb{R}^l$, then the dynamic system (39) is stable in the sense of Lyapnov and is globally convergent to its equilibrium.

The above lemma gives a general convergence results on dynamic systems with the presence of projection operators. In our system, the operator $(\cdot)^+$ is also a projection operator, which projects input values to non-negative ones. With Lemma 1, it is provable for the following stability results on the proposed model (33).

Theorem 2: The dynamic neural network (33) is stable in the sense of Lyapunov and is globally convergent to the optimal solution of (23).

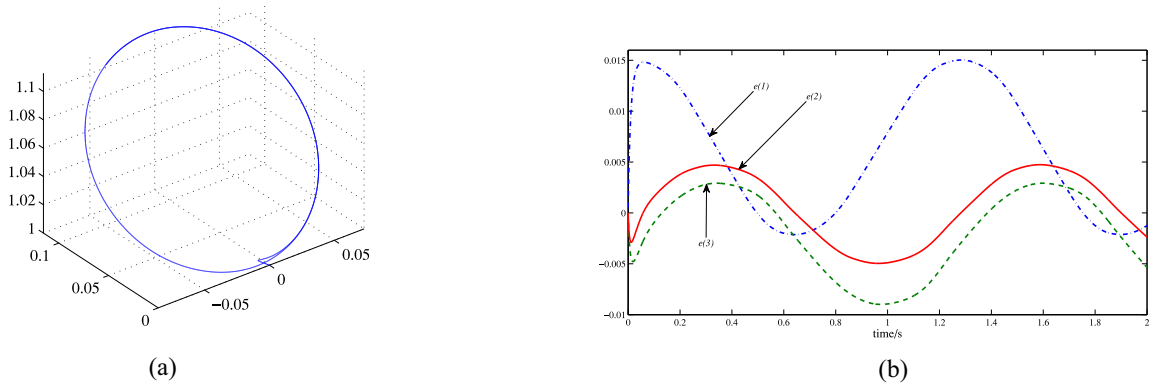
Proof: As we have proved that the output τ associated with the equilibrium points is optimal to problem (23) in Theorem 1, to draw the conclusion in this theorem we only need to show the convergence of (33) to its equilibrium points. To leverage the results presented in Lemma (1), we first convert (33) into a similar form as (39). Define a vector function $F = [F_1^T, F_2^T, F_3^T]^T$, with F_1–F_3 defined as follows:

$$\begin{aligned} F_1 &= \Lambda_2^{-1}\lambda_1 + A_1\Lambda_1^{-1}A_1^T\lambda_1 + A_1\Lambda_1^{-1}A_2^T\lambda_2 + \Lambda_2^{-1}B^T\mu \\ F_2 &= A_2\Lambda_1^{-1}A_1^T\lambda_1 + A_2\Lambda^{-1}A_2^T\lambda_2 - \alpha \\ F_3 &= B\Lambda_2^{-1}\lambda_1 + B\Lambda_2^{-1}B^T\mu + b \end{aligned} \tag{40}$$

and define a set Ω as

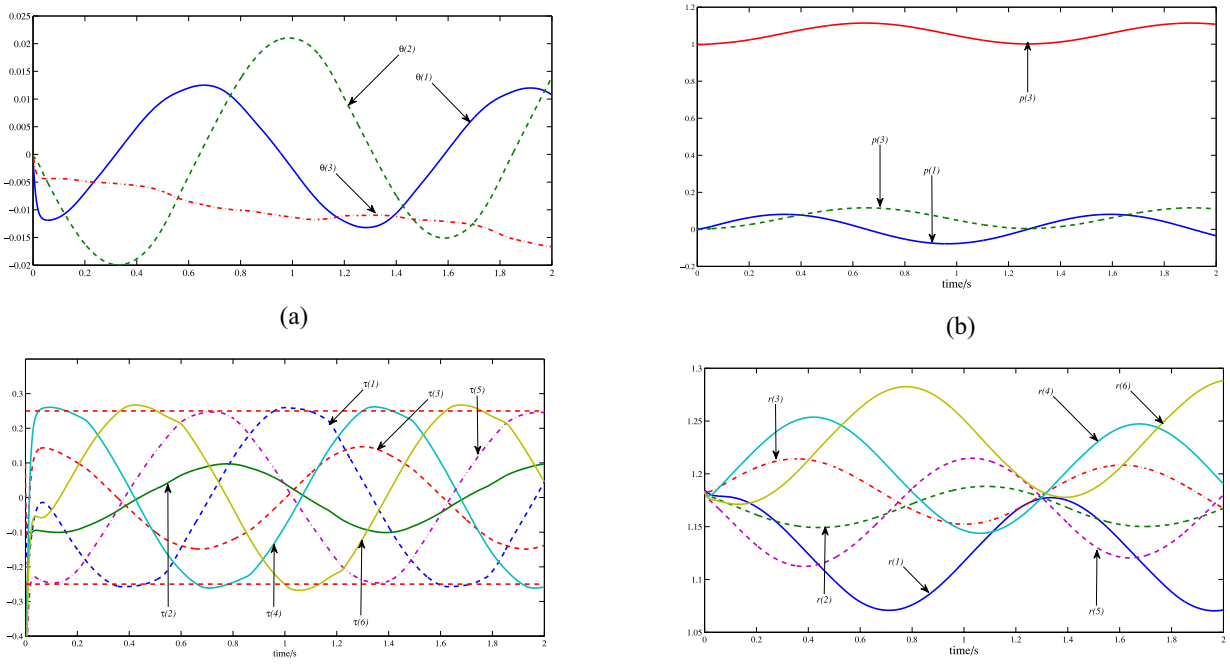
$$\Omega = \left\{(\lambda_1, \lambda_2, \mu), \lambda_1 \in \mathbb{R}^6, \lambda_2 \in \mathbb{R}^m, \mu \in \mathbb{R}^k, \mu \geq 0\right\} \tag{41}$$

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 3. Tracking of a circular motion. (a) Trajectory of the end-effector. (b) Time history of the position tracking error.



(a) (b) (c) (d)

Fig. 4. Time evolution of the Stewart platform state variables in the case of circular motion tracking. (a) Orientation of the platform θ. (b) Position of the end-effector p. (c) Control action τ. (d) Leg length r.

where $\mu \geq 0$ is defined in the piece-wise sense. We also define a new variable

$$x = \begin{bmatrix} \lambda_1^T & \lambda_2^T & \mu^T \end{bmatrix}^T. \tag{42}$$

With the above definitions of F, Ω, and x, the proposed neural network (33) can be converted as

$$\epsilon \dot{x} = -x + P_\Omega(x - F(x)). \tag{43}$$

Define a new time scale $\phi = t/\epsilon$. With the new time scale, $\epsilon \dot{x} = \epsilon(dx/dt) = dx/d\phi$, and the neural dynamic (43) converts to

$$\frac{dx}{d\phi} = -x + P_\Omega(x - F(x)) \tag{44}$$

which are in the nominal form of the projected dynamic systems. To prove the convergence, we show the symmetricity of ∇F in (40) and its positive semi-definiteness

$$\nabla F = \begin{bmatrix} \Lambda_2^{-1} + A_1\Lambda_1^{-1}A_1^T & A_1\Lambda_1^{-1}A_2^T & \Lambda_2^{-1}B^T \\ A_2\Lambda_1^{-1}A_1^T & A_2\Lambda_1^{-1}A_2^T & 0 \\ B\Lambda_2^{-1} & 0 & B\Lambda_2^{-1}B^T \end{bmatrix}. \tag{45}$$

Clearly, ∇F is symmetric. As to the positive semi-definiteness, we decompose ∇F in (45) into the following form:

$$\begin{aligned} \nabla F &= \begin{bmatrix} A_1\Lambda_1^{-1}A_1^T & A_1\Lambda_1^{-1}A_2^T & 0 \\ A_2\Lambda_1^{-1}A_1^T & A_2\Lambda_1^{-1}A_2^T & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &\quad + \begin{bmatrix} \Lambda_2^{-1} & 0 & \Lambda_2^{-1}B^T \\ 0 & 0 & 0 \\ B\Lambda_2^{-1} & 0 & B\Lambda_2^{-1}B^T \end{bmatrix} \\ &= \begin{bmatrix} A_1\Lambda_1^{-\frac{1}{2}} \\ A_2\Lambda_1^{-\frac{1}{2}} \\ 0 \end{bmatrix} \begin{bmatrix} A_1\Lambda_1^{-\frac{1}{2}} \\ A_2\Lambda_1^{-\frac{1}{2}} \\ 0 \end{bmatrix}^T + \begin{bmatrix} \Lambda_2^{-\frac{1}{2}} \\ 0 \\ B_2\Lambda_2^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \Lambda_2^{-\frac{1}{2}} \\ 0 \\ B_2\Lambda_2^{-\frac{1}{2}} \end{bmatrix}. \end{aligned} \tag{46}$$

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

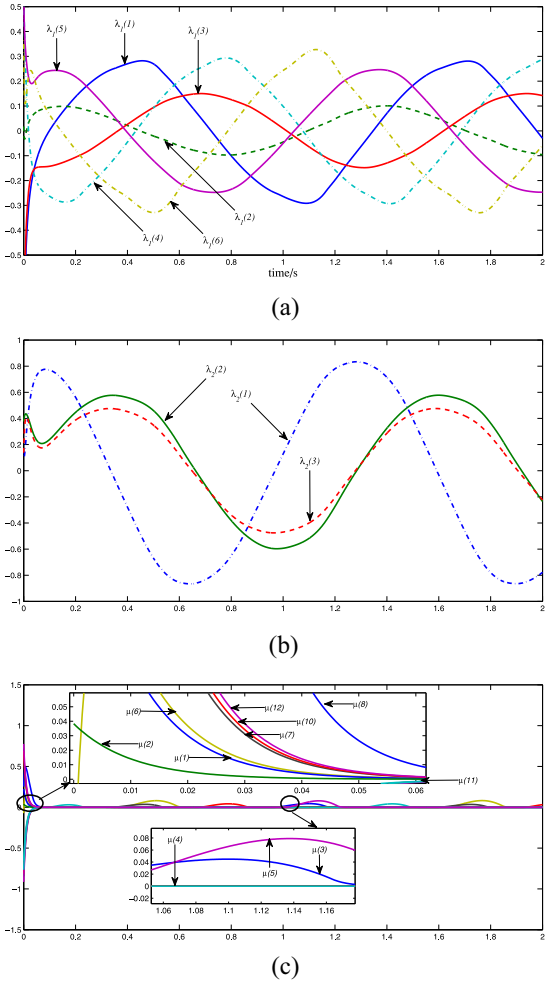


Fig. 5. Time evolution of the neural network state variables in the case of circular motion tracking. Neural network state variables (a) λ_1, (b) λ_2, and (c) μ.

The above expression implies that ∇F is indeed positive semi-definite. In summary, as ∇F defined in this proof is symmetric and positive semi-definite, the dynamic system (44), i.e., the proposed neural network (33) is stable and is convergent to the optimal solution of (24) according to Lemma 1. ■

C. Comparison With Other Control Schemes

In this part, we compare the proposed method with some existing control schemes for parallel robotic platform.

In [35], feed-forward neural network-based approaches, including multilayer perception neural network and radial basis functions neural network, were employed to learn the nonlinear mapping of the robot in its forward kinematics. Thanks to the nonlinear approximation power of feed-forward neural networks, the static mapping error can be minimized by feeding enough training data to the model. However, to determine the values on the weights of neural networks, an off-line training procedure is necessary before the use of the method. Additionally, due to the lack of consideration for speed limits of each actuator, such constraint may not be satisfied in real application. Moreover, due to the existence of remaining training error, the control precision largely depends on the size of training set, and the similarity between the training data and the real ones. Assal [36] proposed to use self organization mapping to approximate the nonlinearity from joint space to workspace. Due to the required training procedure, and the strong dependence of the performance on training data, this method also inevitably bears similar limitations as [35], such as no compliance to speed limitations, nonoptimality in kinematics. Differently, Chen and Fu [37] considered the kinematic model into the design and used sliding mode method for the tracking control of the Stewart platform. In comparison with methods presented in [35] and [37], this method is inherently an online control scheme, does not require any pretraining, and enjoys a high control precision. However, without considering speed constraints, it still lacks compliance to such a physical restriction in speed. In [38], a fuzzy controller was designed for the online intelligent control of Stewart platforms. This controller is able to reach high-precision tracking of signals in the workspace, but may suffer from similar limitations as the sliding mode-based approach [37]. In contrast, since the method presented in this paper directly consider the problem as an optimization problem, it reaches the optimal solution accordingly. Further, since the speed constraints are taken into account as inequality constraints in the problem formulation, compliance to constraints are thus achieved. The comparisons of different methods for kinematic control of Stewart platforms are summarized in Table I, from which it is clear to observe the advantage of the dynamic neural network-based approach for the control of Stewart platforms.

VII. Numerical Investigation

To validate the effectiveness of the proposed approach, in this section, we apply the neural network model to the redundancy resolution of a physically constrained Stewart platform.

A. Simulation Setups

In the simulation, we consider a Stewart platform with the leg connectors on the mobile platform locating around a circle with radius 1.0 m at $b_1' = [0.7386, 0.1302, 0]$, $b_2' = [0.7386, -0.1302, 0]$, $b_3' = [-0.4821, 0.5745, 0]$, $b_4' = [-0.2565, 0.7048, 0]$, $b_5' = [-0.2565, -0.7048, 0]$, and $b_6' = [-0.4821, -0.5745, 0]$ in the platform coordinate, and the leg connectors locating around a circle with radius 0.75 m at $a_1 = [0.3750, 0.6495, 0]$, $a_2 = [0.3750, -0.6495, 0]$, $a_3 = [-0.7500, 0.0000, 0]$, $a_4 = [0.3750, 0.6495, 0]$, $a_5 = [0.3750, -0.6495, 0]$, $a_6 = [-0.7500, 0.0000, 0]$ on the fixed base. For simplicity, the end-effector is put at the origin of the platform coordinate (this can always be achieved by defining the platform coordinate with its origin at the end-effector position). In the situation for position tracking in 3-D space, the total redundancy is 3 as the input dimension is 6 while the output dimension is 3.

In the simulation, we use the tracking error, defined as the difference between the desired position and the real position at time t, to measure the tracking performance in the circular

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

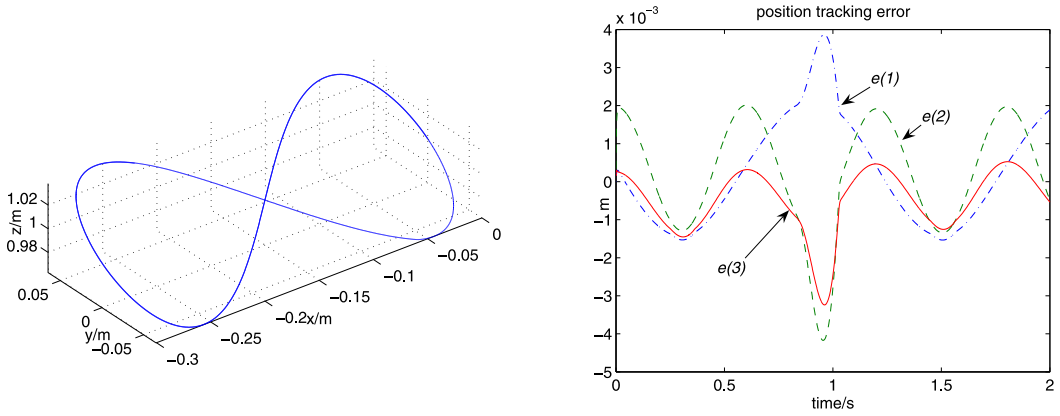


Fig. 6. Tracking of an infinity-sign motion. (a) Trajectory of the end-effector. (b) Time history of the position tracking error.

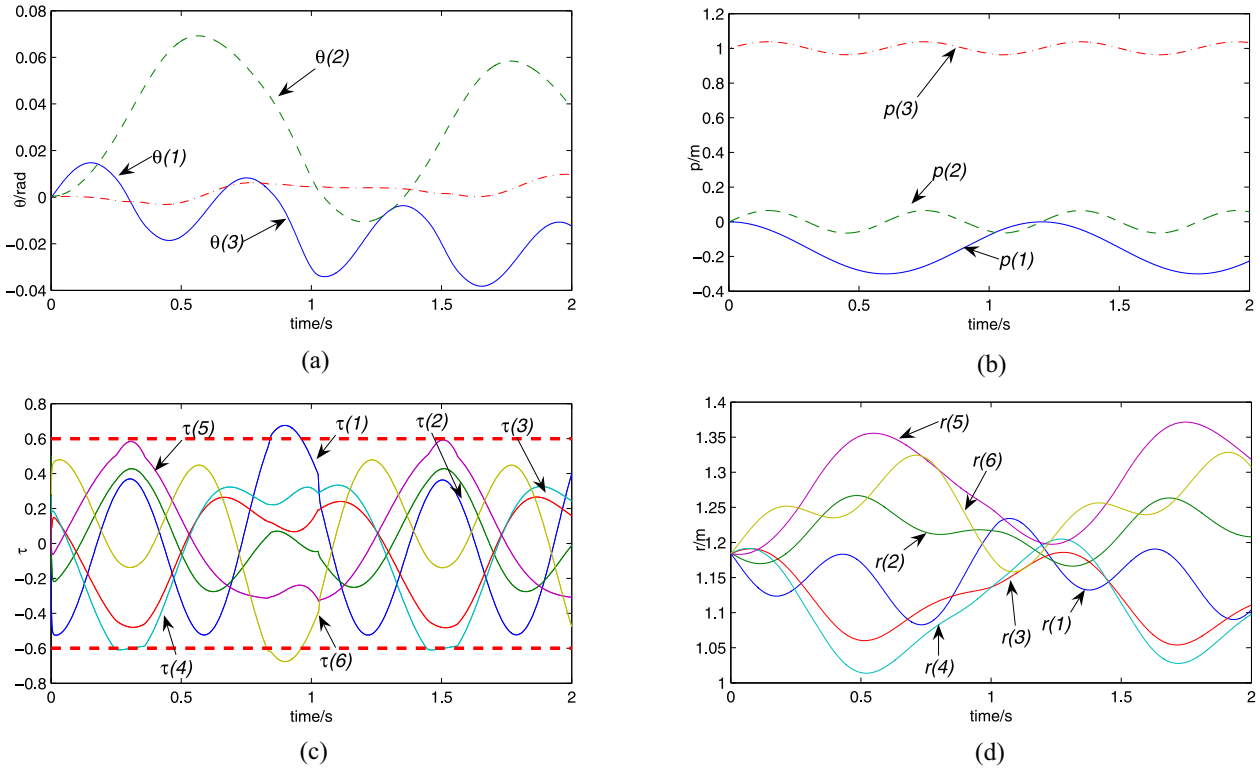


Fig. 7. Time evolution of the Stewart platform state variables in the case of infinity-sign motion tracking. (a) Orientation of the platform θ. (b) Position of the end-effector p. (c) Control action τ. (d) Leg length r.

motion tracking, infinity-sign curve tracking, and the square motion tracking situation.

B. Circular Trajectory

In this part, we consider the tracking of a smooth circular path using the proposed method. The desired motion of the end-effector is to follow a circular trajectory at the speed of 2 m/s. The desired circle trajectory is centered at $[-0.04, 0.06, 1.05]$ with an radius of 0.08 m, and has a revolute angle around the x-axis for 45 degrees. In the simulation, we simply choose Λ_1 and Λ_2 as an identity matrix. The value of the matrix A_1 is computed in real time according to (19). The matrix A_2 is chosen as $A_2 = [I_{3\times3}, 0_{3\times3}]$ such that the position tracking requirements can be achieved. In practice, the actuation speed of each leg is limited within a range due to the physical constraints of the actuators. To capture this property, we impose the constraint that the speed tau is no greater than $\eta > 0$ in its absolute value, i.e., $|\tau_i| \leq \eta$ for $i = 1, 2, \ldots, 6$, which is equivalent to $-\eta \leq \tau_i \leq \eta$. Organizing to a matrix form yields an inequality in the form of (23d) with $B = [I_{6\times6}, I_{6\times6}]^{\mathrm{T}}$, $b = \eta \mathbf{1}_{12}$ with $\mathbf{1}_{12}$ representing a twelve-dimensional vector with all entries equal one. In the simulation, we set the speed bound $\eta = 0.25$ m/s. The scaling factor ϵ is chosen as $\epsilon = 0.01$.

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

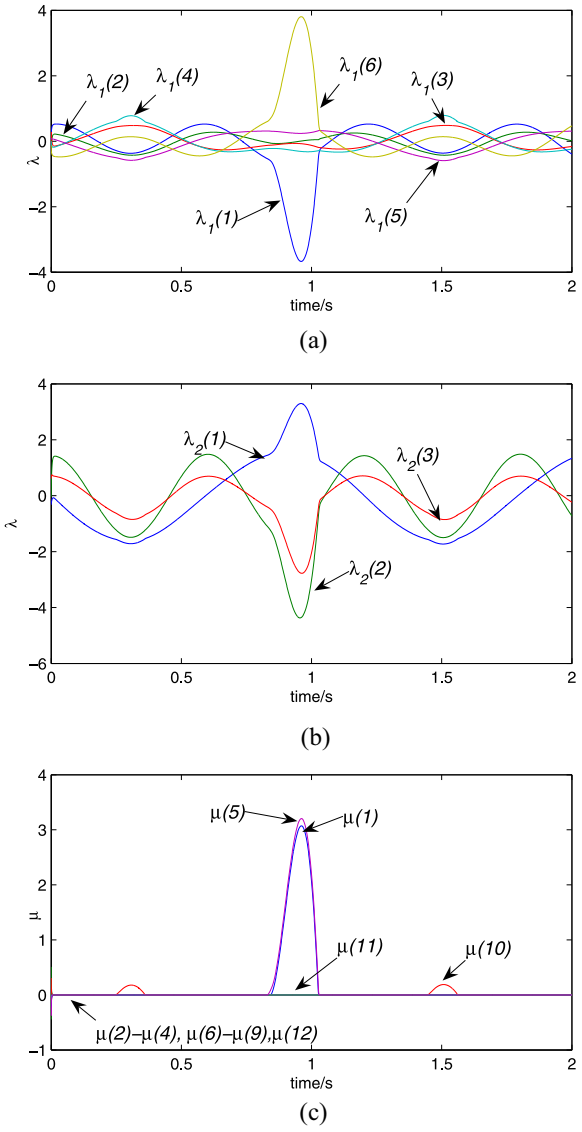


Fig. 8. Time evolution of the neural network state variables in the case of infinity-sign motion tracking. Neural network state variables (a) λ_1, (b) λ_2, and (c) μ.

The tracking results are shown in Fig. 3 by running the simulation for 2 s. As shown in Fig. 3(a), the end-effector successful tracks the circular path with a small tracking error [as shown in Fig. 3(b), where $e(1)$, $e(2)$, and $e(3)$ denote the components of the position tracking error e, respectively, along the x, y, and z axes of the base frame, the errors are less than 0.015 m in amplitude]. The circular-path following experiments demonstrated the capability of the proposed dual neural network for online resolving of kinematic redundancy of physically constrained manipulators. Considering the input motion for the six legs, Fig. 4 shows the time profile of the Stewart platform state variables, i.e., the three Euler orientations of the mobile platform [Fig. 4(a)], the end-effector position [Fig. 4(b)], the speed of each leg [Fig. 4(c)] and the length of each leg [Fig. 4(d)]. The attached moving frame to the upper platform is exactly in the middle and the $p(1)$ and $p(2)$ coordinates started from zero and the $p(3)$ oscillates between almost 0.005 and -0.006 m. The $p(2)$ coordinate started from approximately 0.125 meters which is the altitude of the upper platform before the motion of the actuators. It was observed that the harmonic response is repeated every 1.25 s or 2π after that the cycle is repeated again. From Fig. 4(a), it is observed that there is a small drift from the neutral position. Although the presence of this drifting in orientation, the desired motion, which is given in terms of end-effector positions, are still achived as shown in Fig. 3. This in turn validates the robustness of the proposed scheme in the presence of orientation drifting. The red dashed line in Fig. 4(c) depicts the bound ± 0.25 ($\eta = 0.25$), which is the bound for the action speed. It can be observed that τ converges to the region $[-0.25, 0.25]$ very fast and stays approximately inside this region through the runtime, except some short period [e.g., $\tau(6)$ at around time $t = 1.7$ s] due to the dynamics of the desired motion. The neural network state variable are plotted in Fig. 5. From this figure, we can clearly observer the dynamic evolution of the neural activities. It is noteworthy that the neural activities do not converge to a constant value. Instead, they varies with time as they are utilized to compensate and regulate the dynamic motion of the robot.

C. Infinity-Sign Trajectory

In this part, we consider the tracking of an infinity-sign curve using the presented method. Specifically, we consider the tracking at a constant speed 1 m/s. The desired trajectory is obtained by rotating a planar infinity-sign curve analytically expressed as $x^4 = a^2(x^2 - y^2)$ with $a = 0.15$ m around the x-axis for 45 degrees. The center of this curve locates at $[-0.04, 0.06, 1.05]$. The parameters, Λ_1, Λ_2, A_1, A_2, η, B, b, ϵ are chosen the same value as in the circular trajectory tracking situation. Simulation results are obtained by running it for 2 s. As shown in Fig. 6, the end-effector successful tracks the desired path with a small tracking error. Fig. 7 shows the time profile of the Stewart platform state variables, including the 3 Euler orientations of the mobile platform [Fig. 7(a)], the end-effector position [Fig. 7(b)], the speed of each leg [Fig. 7(c)] and the length of each leg [Fig. 7(d)]. The neural network state variable are plotted in Fig. 8. Due to the dynamic changing of the neural states, the nonlinearity in the Stewart platform is successfully compensated and an accurate tracking performance is thus achieved.

D. Square Trajectory

In this section, we investigate the square trajectory tracking using the proposed approach. Different from the case of smooth circular motion tracking, the desired square path is nonsmooth at the four corners and poses challenges to the controller on its real-time performance. In the simulation, the desired motion of the end effector is to follow a square trajectory, which is centered at $[0.15, 0.075, 0.74]$ with the edge length of 0.08 m, at the desired speed 1.0 m/s. The square has a revolute angle around the x-axis for 60 degrees. We choose the parameters Λ_1 and Λ_2, A_2 and B the same values as in

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

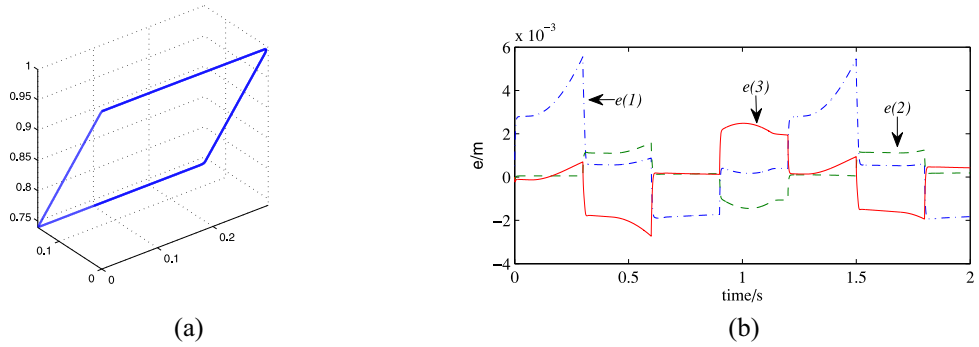


Fig. 9. Tracking of a square motion. (a) Trajectory of the end-effector. (b) Time history of the position tracking error.

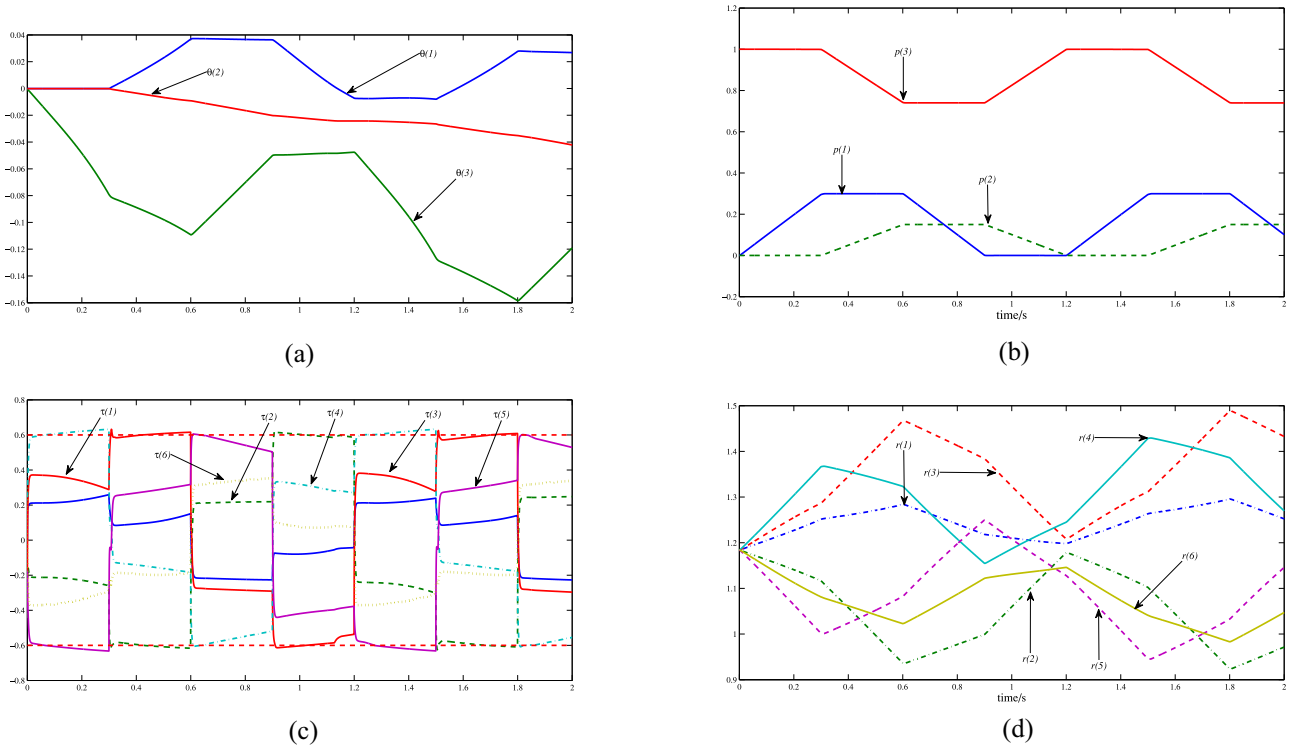


Fig. 10. Time evolution of the Stewart platform state variables in the case of square motion tracking. (a) Orientation of the platform θ. (b) Position of the end-effector p. (c) Control action τ. (d) Leg length r.

the situation for circular motion for simplicity. The parameter A_1 is computed in real time according to (19). The speed limit bound η is chosen as $\eta = 0.6$ m/s, and b is accordingly chosen as $b = 0.6\mathbf{1}_{12}$ with $\mathbf{1}_{12}$. The scaling factor ϵ is chosen as $\epsilon = 0.001$. The tracking results are shown in Fig. 9 by running the simulation for 2 s. As shown in Fig. 9(a), the end-effector successful tracks the square motion with a small tracking error [as shown in Fig. 9(b), where $e(1)$, $e(2)$, and $e(3)$ denote the components of the position tracking error e, respectively, along the x, y, and z axes of the base frame, the errors are less than 0.006 m in amplitude]. Remarkably that the error curves have jerks at time $t = 0.3, 0.6, 0.9, 1.2, 1.5, 1.8$ s, which is a result of the velocity switching from one direction to another one around the corner of the square. Despect of the existence of the jerks, due to the nonlinear feedback mechanism in the neural network, the errors reduces swiftly to a very low value [much lower than 0.001 m as shown in Fig. 3(b) after the jerk]. The above observation in turn validates the effectiveness of the proposed neural scheme in dealing with nonsmooth tracking problem. The real-time values on the Stewart platform states are shown in Fig. 10. Similar to the situation for circle tracking, the presence of drifting in orientations, as observed in Fig. 10(a), does not affect the tracking performance. The end-effector position evolves without drifting [Fig. 10(b)], and remains a very small error from its desired trajectory [Fig. 3(b)]. Fig. 3(c) shows the time profile of the control action, which is the speed of each leg. It is clear that the control action is approximately regulated inside the range $[-0.6, 0.6]$, which in turn validates the effect of the proposed solution in fulfilling the inequality regulation. The dynamic evolution of the neural activities are shown in Fig. 11.

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

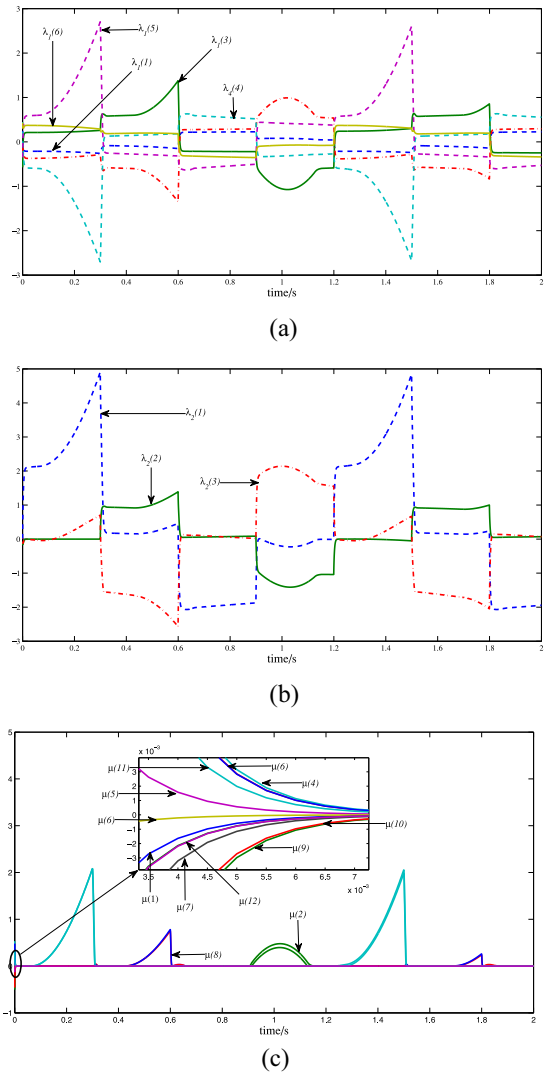


Fig. 11. Time evolution of the neural network state variables in the case of square motion tracking. Neural network state variables (a) λ_1, (b) λ_2, and (c) μ.

VIII. Conclusion

In this paper, a dynamic neural network is designed to solve the redundancy resolution problem of Stewart platforms. The physical constraints and optimization criteria are rigorously modeled as a constrained quadratic programming problem. To solve this problem in real time, a recurrent neural network is proposed to reach the equality constraints, inequality constraints, and optimality criteria simultaneously. Rigorous theoretical proof are supplied to verify the convergence of the proposed model. Simulation results validate the effectiveness of the proposed solution.

References

[1] S. Liu, W. Li, Y. Du, and L. Fang, "Forward kinematics of the Stewart platform using hybrid immune genetic algorithm," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Luoyang, China, Jun. 2006, pp. 2330–2335.

[2] O. Masory and W. Jian, "Workspace evaluation of Stewart platforms," *Adv. Robot.*, vol. 9, no. 4, pp. 443–461, 1994.

[3] Q. Liu, Q. He, and Z. Shi, "Extreme support vector machine classifier," in *Advances in Knowledge Discovery and Data Mining* (LCNS 5012), T. Washio, E. Suzuki, K. Ting, and A. Inokuchi, Eds. Berlin, Germany: Springer, 2008, pp. 222–233.

[4] Y. Nakamura and H. Hanafusam, "Inverse kinematics solutions with singularity robustness for robot manipulator control," *J. Dyn. Syst. Meas. Control*, vol. 108, no. 3, pp. 163–171, 1986.

[5] G. Thomas and M. John, "Inverse dynamic analysis of parallel manipulators with full mobility," *Mech. Mach. Theory*, vol. 38, no. 6, pp. 549–562, 2003.

[6] A. Yorukoglu and E. Altug, "Estimation of unbalanced loads in washing machines using fuzzy neural networks," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 3, pp. 1182–1190, Jun. 2013.

[7] N. Nikdel, P. Nikdel, M. A. Badamchizadeh, and I. Hassanzadeh, "Using neural network model predictive control for controlling shape memory alloy-based manipulator," *IEEE Trans. Ind. Electron.*, vol. 61, no. 3, pp. 1394–1401, Mar. 2014.

[8] Y. Zhang, B. Mu, and H. Zheng, "Link between and comparison and combination of Zhang neural network and quasi-Newton BFGS method for time-varying quadratic minimization," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 490–503, Apr. 2013.

[9] Y. Xia, T. Chen, and J. Shan, "A novel iterative method for computing generalized inverse," *Neural Comput.*, vol. 26, no. 2, pp. 449–465, Feb. 2014.

[10] L. Xiao and Y. Zhang, "A new performance index for the repetitive motion of mobile manipulators," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 280–292, Feb. 2014.

[11] S. Li, Z. You, H. Guo, X. Luo, and Z. Zhao, "Inverse-free extreme learning machine with optimal information updating," *IEEE Trans. Cybern.*, DOI: 10.1109/TCYB.2015.2434841.

[12] L. Jin and Y. Zhang, "G2-type SRMPC scheme for synchronous manipulation of two redundant robot arms," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 153–164, Feb. 2015.

[13] M. Jose, V. Santiba, R. Soto, and M. A. Llama, "Fuzzy self-tuning PID semiglobal regulator for robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 59, no. 6, pp. 2709–2717, Jun. 2012.

[14] S. Nasri, F. Piltan, S. Haghighi, I. Nazari, and S. Siamak, "Artificial control of puma robot manipulator: A-review of fuzzy inference engine and application to classical controller," *Int. J. Robot. Autom.*, vol. 2, no. 5, pp. 406–429, 2011.

[15] R. Köker, "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization," *Inf. Sci.*, vol. 222, pp. 528–543, Feb. 2013.

[16] H. Vicente, H. Ayala, and D. S. Coelho, "Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator," *Expert Syst. Appl.*, vol. 39, no. 10, pp. 8968–8974, 2012.

[17] Z. P. Wang, T. Zhou, Y. Mao, and Q. J. Chen, "Adaptive recurrent neural network control of uncertain constrained nonholonomic mobile manipulators," *Int. J. Syst. Sci.*, vol. 45, no. 2, pp. 133–144, 2014.

[18] Y. Zhang, S. S. Ge, and T. Lee, "A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 5, pp. 2126–2132, Oct. 2004.

[19] Y. Zhang, W. Jun, and Y. Xia, "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 658–667, May 2003.

[20] Y. Xia and W. Jun, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 1, pp. 147–154, Feb. 2001.

[21] S. Li, S. Chen, B. Liu, Y. Li, and Y. Liang, "Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks," *Neurocomputing*, vol. 91, pp. 1–10, Aug. 2012.

[22] S. Li, H. Cui, and Y. Li, "Decentralized control of collaborative redundant manipulators with partial command coverage via locally connected recurrent neural networks," *Neural Comput. Appl.*, vol. 23, nos. 3–4, pp. 1051–1060, 2013.

[23] S. Liu and J. Wang, "A simplified dual neural network for quadratic programming with its KWTA application," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1500–1510, Nov. 2006.

[24] Z. Guo, J. Wang, and Z. Yan, "Global exponential dissipativity and stabilization of memristor-based recurrent neural networks with time-varying delays," *Neural Netw.*, vol. 48, pp. 158–172, Dec. 2013.

[25] C. Zheng and J. Cao, "Robust synchronization of coupled neural networks with mixed delays and uncertain parameters by intermittent pinning control," *Neurocomputing*, vol. 141, pp. 153–159, Oct. 2014.

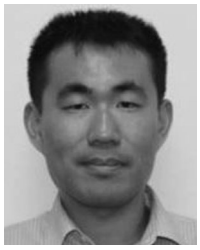
[26] H. Zhang, Z. Wang, and D. Liu, "A comprehensive review of stability analysis of continuous-time recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1229–1262, Jul. 2014.

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.

[27] S. Zhang, Y. Xia, and W. Zheng, "A complex-valued neural dynamical optimization approach and its stability analysis," *Neural Netw.*, vol. 61, pp. 59–67, Jan. 2015.
[28] Y. Xia and H. Leung, "A fast learning algorithm for blind data fusion using a novel L_2-norm estimation," *IEEE Sensors J.*, vol. 14, no. 3, pp. 666–672, Mar. 2014.
[29] Y. Xia, C. Sun, and W. Zheng, "Discrete-time neural network for fast solving large linear L_1 estimation problems and its application to image restoration," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 5, pp. 812–820, May 2012.
[30] E. D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, vol. 6. New York, NY, USA: Springer, 1998.
[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, Mar. 2004.
[32] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, no. 5, pp. 554–562, May 1988.
[33] Y. S. Xia, "Further results on global convergence and stability of globally projected dynamical systems," *J. Optim. Theory Appl.*, vol. 122, no. 3, pp. 627–649, 2004.
[34] Y. Xia, G. Feng, and J. Wang, "A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 1, pp. 54–64, Feb. 2005.
[35] D. Zhang and J. Lei, "Kinematic analysis of a novel 3-DOF actuation redundant parallel manipulator using artificial intelligence approach," *Robot. Comput.-Integr. Manuf.*, vol. 27, no. 1, pp. 157–163, 2011.
[36] S. Assal, "Self-organizing approach for learning the forward kinematic multiple solutions of parallel manipulators," *Robotica*, vol. 30, no. 6, pp. 951–961, 2012.
[37] S. Chen and L. Fu, "Output feedback sliding mode control for a Stewart platform with a nonlinear observer-based forward kinematics solution," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 1, pp. 176–185, Jan. 2013.
[38] G. Liu, Z. Qu, X. Liu, and J. Han, "Tracking performance improvements of an electrohydraulic Gough–Stewart platform using a fuzzy incremental controller," *Ind. Robot Int. J.*, vol. 41, no. 2, pp. 225–235, 2014.

Aquil Mirza Mohammed was born in Hyderabad, India. He received the B.E. degree in information technology from Osmania University, Hyderabad, in 2007, the M.S. degree in research of very-large-scale integration and embedded systems from the International Institute of Information Technology, Hyderabad, in 2009, and the M.S. degree in electrical engineering from the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia.

He is currently a Doctorate Research Scholar with Hong Kong Polytechnic University, Hong Kong. His current research interests includes routing in wireless sensor networks, performance evaluation of neural networks, Ashton Raggatt McDougall development for multiple-input and multiple-output systems, particle swarm optimization, and pattern recognition.



Shuai Li (M'14) received the B.E. degree in precision mechanical engineering from the Hefei University of Technology, Hefei, China, in 2005, the M.E. degree in automatic control engineering from the University of Science and Technology of China, Hefei, in 2008, and the Ph.D. in electrical and computer engineering from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2014.

He is a Research Assistant Professor with Hong Kong Polytechnic University, Hong Kong. His current research interests include distributed estimation, optimization and control of networked systems, kinematic and dynamic analysis of robotic systems, and neural networks.

Dr. Li is currently on the editorial board of the *Neural Computing and Applications* and the *International Journal of Distributed Sensor Networks*.

Authorized licensed use limited to: Nanchang University. Downloaded on December 16,2023 at 09:43:08 UTC from IEEE Xplore. Restrictions apply.