Manipulability Optimization of Redundant Manipulators Using Dynamic Neural Networks

Long Jin, Shuai Li, *Member, IEEE*, Hung Manh La, *Senior Member, IEEE*, and Xin Luo, *Member, IEEE*

***Abstract*—For solving the singularity problem arising in the control of manipulators, an efficient way is to maximize its manipulability. However, it is challenging to optimize manipulability effectively because it is a nonconvex function to the joint angles of a robotic arm. In addition, the involvement of an inversion operation in the expression of manipulability makes it even hard for timely optimization due to the intensively computational burden for matrix inversion. In this paper, we make progress on real-time manipulability optimization by establishing a dynamic neural network for recurrent calculation of manipulability-maximal control actions for redundant manipulators under physical constraints in an inverse-free manner. By expressing position tracking and matrix inversion as equality constraints, physical limits as inequality constraints, and velocity-level manipulability measure, which is affine to the joint velocities, as the objective function, the manipulability optimization scheme is further formulated as a constrained quadratic program. Then, a dynamic neural network with rigorously provable convergence is constructed to solve such a problem online. Computer simulations are conducted and show that, compared to the existing methods, the proposed scheme can raise the manipulability almost 40% on average, which substantiates the efficacy, accuracy, and superiority of the proposed manipulability optimization scheme.**

***Index Terms*—Dynamic neural network, kinematic control, manipulability optimization, redundancy resolution.**

I. Introduction

IN RECENT decades, robotics has attracted considerable attention in scientific research works and engineering applications. Much effort has been contributed to robotics, and different types of robots have thus been developed and investigated [1]–[8]. Among these robots, redundant robot manipulators, possessing more degrees of freedom (DOFs) in joint space than workspace and offering increased control flexibility for complicated tasks, have played a more and more important role in numerous fields of engineering applications [7]–[9]. For example, the problem of finite-time stabilization and control of redundant manipulators is investigated in [9], and a controller is designed to attenuate the effects of model nonlinearity, uncertainties, and external disturbances and improve the response characteristics of the system. The forward kinematics of a redundant manipulator provides a nonlinear mapping from its joint space to its operating region in Cartesian space. This nonlinear mapping makes it difficult to directly solve the redundancy resolution problem at the angle level. Instead, in most approaches, the problem is first converted into a problem at the velocity- or acceleration level, and solutions are then sought in the converted space. A popular method is to apply the pseudoinverse formulation for obtaining a general solution at the joint-velocity level. However, this strategy suffers from an intensive computational burden because of the need to perform the pseudoinversion of matrices continuously over time, in addition to the weakness that the joint physical limits are usually not considered [10], [11]. Recent progress shows the advantages of unifying the treatment of various constraints of manipulators' redundancy resolution into a quadratic program (QP). Such a QP formulation is general in the sense that it incorporates equality, inequality, and bound constraints, simultaneously. For example, reformulated as a QP, the equivalence between different-level redundancy-resolution of redundant manipulators is investigated in [12]. Then, the work is extended in [7] for obstacle avoidance of redundant robot manipulators with the aid of QP. In addition to the control of redundant manipulators, the QP is also exploited to the other types of robots, e.g., formation control of leader–follower mobile robots' systems in [1], [3], [13]–[18].

Neural networks, which feature capabilities of high-speed parallel distributed processing, and can be readily implemented by hardware, have been recognized as a powerful tool for real-time processing and successfully applied widely in various control systems [3], [5], [8], [12], [19]–[26]. Particularly, using neural networks for the control of robot manipulators have attracted much attention and various related schemes and methods have been proposed and investigated [3], [5], [8], [24], [25]. Cai and Zhang present two neural networks for the online solution of quadratic programming problem existing in the redundancy res-

Manuscript received October 2, 2016; revised December 1, 2016; accepted December 20, 2016. Date of publication February 24, 2017; date of current version May 10, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61401385, in part by the Hong Kong Research Grants Council Early Career Scheme under Grant 25214015, in part by the Departmental General Research Fund of Hong Kong Polytechnic University under Grant G.61.37.UA7L, in part by the Pioneer Hundred Talents Program of the Chinese Academy of Sciences, and in part by the International Joint Project funded jointly by the Royal Society of the U.K. and the National Natural Science Foundation of China under Grant 61611130209. *(Corresponding authors: S. Li and X. Luo.)*

L. Jin and S. Li are with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: longjin@polyu.edu.hk; shuaili@polyu.edu.hk).

H. M. La is with the Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557 USA (e-mail: hla@unr.edu).

X. Luo is with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: luoxin21@cigit.ac.cn).

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/TIE.2017.2674624

olution of manipulators in [12]. They further modify their work in [8] by proposing new noise-tolerant neural networks with application to the motion generation of redundant manipulators. It is worth pointing out here that those neural networks presented in [8], [12] do not consider the bound constraint and thus cannot avoid the joint physical constraints in the manipulator applications. For the direct consideration of inequalities in constrained quadratic programming problems, researchers have attempted to consider the problem in its dual space. Various dual neural networks with different architectures have been proposed [7], [27]–[30]. The dual neural networks-based methods are highly efficient for real-time processing and have been successfully used in various applications, including redundant manipulator control.

The performance index plays an important role in quadratic programming formulation-based manipulator control, which, to some extent, determines the application potential of redundant manipulators in different industry fields. Therefore, how to model an efficient performance index is an important issue for the redundancy resolution. It is worth pointing out that, when a manipulator is at a kinematic singularity configuration, its Jacobian matrix becomes ill-conditioned and rank-deficient [28]. In addition, getting close to a singularity point of the kinematic mapping is also undesirable and unacceptable due to the fact that, in such a state, when the end-effector moves in certain directions, joint velocities and accelerations can be arbitrarily large and this may damage the manipulator. Therefore, the research on maximizing the manipulability of manipulators has explicit significance and has been investigated widely. Yoshikawa establishes the first quantitative measure of the manipulability for redundant manipulators at the joint velocity level in [31]. The seminal study presented by Yoshikawa on this direction exploits the pseudoinverse-type formulation to avoid the singularity of redundant manipulators by maximizing the manipulability measure. A multiobjective optimization of a hybrid robotic machine tool is presented in [32], which involves different types of manipulability. Note that most of the aforementioned techniques are based on pseudoinverse-type formulations and do not consider physical limits of manipulators. Zhang *et al.* present a quadratic programming formulation-based manipulability-maximizing scheme, which can handle the physical limits. However, this scheme suffers from an intensive computational burden because of the need to perform the matrix inversion continuously over time in performance index [28]. Moreover, the tradeoff between manipulability and energy consumption is fixed in this scheme, which leads to less adaptation to environments. This paper makes progress on this frontier through the modeling of the manipulability optimization of redundant manipulators and the proposal of dynamic neural network to remedy the weaknesses of existing schemes. Note that, the scheme presented in [28], together with the other schemes mentioned, will be compared in details in Section VI of the paper to clarify the differences and advantages of the new scheme proposed.

The remainder of this paper is organized into six sections. Preliminaries are presented in Section II. Then, problem formulation is presented in Section III. Section IV formulates such a manipulability optimization into a quadratic programming problem. Section V presents a dynamic neural network for the online solution of such a QP with theoretical analyses on convergence conducted. Section VI provides illustrative simulation examples to substantiate the efficacy and superiority of the proposed scheme for manipulability optimization of redundant manipulators. Section VII concludes the paper with final remarks.

II. Preliminaries

In this section, we present useful definitions and convergence lemmas that play an important role for the theoretical derivation in this paper.

Definition 1 (Projection Operator): The projection operator for a set $S \subset \mathbb{R}^k$ and $x \in S$ is defined by

$$P_S(x) = \text{argmin}_{y \in S} \|y - x\|^2 \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm.

Definition 2 (Normal Cone): For a convex set $S \subset \mathbb{R}^k$ and $x \in S$, the normal cone of S at x, denoted by $N_S(x)$ is defined as

$$N_S(x) = \{\varphi \in \mathbb{R}^k, (y - x)^{\mathrm{T}} \varphi \leq 0, \forall y \in S\}. \tag{2}$$

Definition 3 (Monotone Mapping): A mapping $F(\cdot)$ is called monotone if for each pair of points x, y, there is

$$(x - y)^{\mathrm{T}}(F(x) - F(y)) \geq 0. \tag{3}$$

The above definition generalizes the definition of monotonicity from univariable mappings to multivariable mappings. Note that it is possible to verify the monotonicity of a univariable mapping by checking whether its derivative is greater than zero. Similarly, this property can also be extended to the general multi-variable mapping $F(\cdot)$. For a continuously differentiable mapping $F(\cdot)$, it is a monotone mapping if

$$\nabla F + \nabla^{\mathrm{T}} F \succeq 0 \tag{4}$$

where "$\succeq 0$" means the left-hand side of this operator is a positive semi-definite matrix.

For the projection operator and the normal cone, their relationship is summarized in the following lemma.

Lemma 1 (Projection and Normal Cone [33]): For a compact convex set $S \in \mathbb{R}^k$ and $x \in S$, $z \in N_S(x)$ is equivalent to

$$P_S(x + z) = x. \tag{5}$$

For a general dynamic systems, the so-called LaSalle's invariance principle applies, as stated below.

Lemma 2 (LaSalle's Invariance Principle [34]): Let S be a compact set that is positively invariant with respect to the dynamics of a system $\dot{x} = g(x)$. Let $V = V(x) \in \mathbb{R}$ be a continuously differentiable function, such that $\dot{V} \leq 0$ in S. Let E be the set of all points in S, where $\dot{V} = 0$. Let M be the largest invariant set in E. Then, every solution starting in S approaches M as $t \to \infty$.

For a time-varying function, Barbalat's lemma holds.

Lemma 3 (Barbalat's Lemma [34]): If $f(t) \in \mathbb{R}^k$ has a finite limit as $t \to \infty$ and if $\dot{f}$ is uniformly continuous (or $\ddot{f}$ is bounded), then $\lim_{t \to \infty} \dot{f}(t) = 0$.

For a class of well-studied dual neural networks, one sufficient condition for its convergence has been established as follows.

Lemma 4 (Convergence for a Class of Neural Networks [35]): Assume that $F(x)$ is monotone and continuously

differentiable. The following dynamic system

$$\epsilon\dot{x} = -x + P_S(x - \varrho F(x)) \tag{6}$$

where $\epsilon > 0$ and $\varrho > 0$ are both positive constants, S is a closed convex set, $P_S(\cdot)$ denotes the projection operator as defined by (1). Then, the trajectory corresponding to (6) globally converges to its equilibrium point.

III. Problem Formulation

In this section, definitions on manipulator kinematics and manipulability are presented for problem formulation.

A. Manipulator Kinematics

For a k-DOF redundant manipulator with the joint angle $\theta(t) = [\theta_1(t), \theta_2(t), ..., \theta_k(t)]^{\mathrm{T}} \in \mathbb{R}^k$, its Cartesian coordinate $r \in \mathbb{R}^m$ in the workspace can be described as a nonlinear mapping

$$r(t) = f(\theta(t)) \tag{7}$$

where the mapping $f(\cdot)$ carries mechanical and geometrical information of a manipulator. By definition, the joint space dimension k of a redundant manipulator is greater than the workspace dimension m, i.e., $k > m$. Computing time derivations on both sides of (7), we have

$$\dot{r}(t) = J(\theta(t))\dot{\theta}(t) \tag{8}$$

where $J(\theta(t)) = \partial f/\partial\theta \in \mathbb{R}^{m\times k}$ is called the Jacobian matrix of $f(\theta(t))$, and usually is abbreviated as J. The end-effector $r(t)$ of the redundant manipulator is expected to track the desired path $r_d(t)$, i.e., $r(t) \to r_d(t)$ with $\dot{r}(t) \to \dot{r}_d(t)$. It is worth pointing out here that, in the rest of this paper, the argument t is frequently omitted for presentation convenience, e.g., by writing $r(t)$ as r.

B. Manipulability

The manipulability of a robot manipulator is a function of the Jacobian matrix as defined below [36]:

$$\mu = \sqrt{\det(JJ^{\mathrm{T}})} = \sqrt{\mu_1\mu_2...\mu_m} \tag{9}$$

where $\mu_i \geq 0$ for $i = 1, 2, ..., m$ is the ith largest eigenvalue of JJ^{T} (note $JJ^{\mathrm{T}} \succeq 0$ and thus its eigenvalues are all nonnegative). This measure gives an overall scalar description on the gain from joint velocity θ to $\dot{r}$ and measures the amount of singularity. When the robot arm is singular, i.e., $\mathrm{rank}J < m$, μ reaches its least value 0. To increase the manipulability and avoid singularity, a large value of μ is preferable in operation.

C. Optimization Problem Formulation

Therefore, based on the above analysis, the redundancy resolution of a redundant manipulator with manipulability optimality considered is formulated as

$$\min_{\theta} -\mu, \tag{10a}$$

$$r_d = f(\theta). \tag{10b}$$

So far the redundancy resolution has been formulated as a constrained optimization with the manipulator kinematics as an equation constraint and the manipulability as a cost to maximize. In this optimization problem μ, as a function of $J = J(\theta)$, usually is nonconvex relative to θ. Additionally, the equation constraint is also usually nonlinear. Thus, the solution of θ in (10) becomes a challenging problem.

IV. Reformulation as a Constrained QP

Here, by incorporating physical constraints and redefining the objective function, we reformulate such a redundancy resolution problem with optimal manipulability considered into a constrained quadratic programming problem.

A. Equation Constraint: Speed-Level Resolution

Equation (7) describes the mapping from the joint space to the workspace in the position level and has a strong nonlinearity. As shown in (8), the mapping in the velocity level can be significantly simplified. Defining $v = \dot{r}$ as the end-effector velocity in the workspace and $w = \dot{\theta}$ as the joint angular velocity in the joint space and abbreviating $J(\theta(t))$ as J, (8) can be rewritten as

$$v = Jw. \tag{11}$$

This equation describes the kinematic mapping from the joint space to the workspace at the velocity level. In comparison with the position-level mapping (7) expressed as a general nonlinear function, the velocity-level mapping is much simpler as it is in an affine form. Simply enforcing $\dot{r}_d = Jw$ usually is able to obtain w satisfying the speed requirement $v = \dot{r}_d$, but may suffer from drifting in position due to the loss of explicit information on r_d. Instead, we restrict the motion of the redundant manipulator in the following way to reach a desired position:

$$Jw = -k_0(f(\theta) - r_d) + \dot{r}_d \tag{12}$$

where $k_0 > 0$ is the position error feedback coefficient. Note that $Jw - \dot{r}_d = \mathrm{d}(f(\theta) - r_d)/\mathrm{d}t$. Accordingly, the above equation can be written as $\mathrm{d}(f(\theta) - r_d)/\mathrm{d}t = -k_0(f(\theta) - r_d)$ and is able to drive $f(\theta)$ to r_d over time. Following the above reasoning, we can define an equation constraint as follows to represent the speed requirement:

$$Jw = v_d \text{ with } v_d = -k_0(f(\theta) - r_d) + \dot{r}_d. \tag{13}$$

B. Redefinition of the Objective Function

As to $\mu^2/2$, we examine its value in time derivative as

$$\begin{aligned}\frac{\mathrm{d}(\mu^2/2)}{\mathrm{d}t} &= \frac{\mathrm{d}[\det(JJ^{\mathrm{T}})/2]}{\mathrm{d}t} \\ &= \frac{\det(JJ^{\mathrm{T}})\mathrm{tr}((JJ^{\mathrm{T}})^{-1}(J\dot{J}^{\mathrm{T}} + \dot{J}J^{\mathrm{T}}))}{2}\end{aligned} \tag{14}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. In addition, we have $\mathrm{tr}((JJ^{\mathrm{T}})^{-1}(J\dot{J}^{\mathrm{T}})) = \mathrm{tr}((JJ^{\mathrm{T}})^{-1}J\dot{J}^{\mathrm{T}}) = \mathrm{tr}(\dot{J}J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1})$ and $\mathrm{tr}((JJ^{\mathrm{T}})^{-1}(\dot{J}J^{\mathrm{T}})) = \mathrm{tr}((\dot{J}J^{\mathrm{T}})(JJ^{\mathrm{T}})^{-1})$. Thus, (14) can be

simplified to

$$\frac{\mathrm{d}(\mu^2/2)}{\mathrm{d}t} = \frac{\mathrm{d}(\det(JJ^{\mathrm{T}})/2)}{\mathrm{d}t} = \det(JJ^{\mathrm{T}})\mathrm{tr}(\dot{J}J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}). \quad (15)$$

Note that

$$\dot{\mu} = \frac{1}{\mu}\frac{\mathrm{d}(\mu^2/2)}{\mathrm{d}t} = \sqrt{\det(JJ^{\mathrm{T}})}\mathrm{tr}\,(\dot{J}J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}). \quad (16)$$

Overall, the quantity $\dot{\mu}$ describes the change of μ with time. By maximizing its value, the system is enforced to evolve along the direction to increase μ greedily for the most under constraints. To compute $\dot{J}$, we first define $H_i \in \mathbb{R}^{m\times k}$ for $i = 1, 2, ..., k$ as follows:

$$H_i = \frac{\partial J}{\partial \theta_i} \quad (17)$$

with which, $\dot{J}$ is expressed as

$$\dot{J} = \sum_{i=1}^{k}\frac{\partial J}{\partial \theta_i}\dot{\theta}_i = \sum_{i=1}^{k} H_i w_i. \quad (18)$$

Incorporating this expression, (16) becomes

$$\dot{\mu} = \sqrt{\det(JJ^{\mathrm{T}})}\mathrm{tr}\left(\left(\sum_{i=1}^{k} H_i w_i\right)J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}\right) = \sqrt{\det(JJ^{\mathrm{T}})}\sum_{i=1}^{k} w_i \mathrm{tr}\,(H_i J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}). \quad (19)$$

For the convenience of later treatment of matrix $(JJ^{\mathrm{T}})^{-1}$, we now convert the expression of $\mathrm{tr}\,(H_i J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1})$ into a form using $\mathrm{vec}((JJ^{\mathrm{T}})^{-1})$, which is the vectorization of the matrix $(JJ^{\mathrm{T}})^{-1}$ formed by stacking the columns of $(JJ^{\mathrm{T}})^{-1}$ into a single column vector. We have

$$\mathrm{tr}\,(H_i J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}) = \mathrm{tr}\,((JH_i^{\mathrm{T}})^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}) = \mathrm{vec}^{\mathrm{T}}(JH_i^{\mathrm{T}})\mathrm{vec}((JJ^{\mathrm{T}})^{-1}). \quad (20)$$

With this, (19) further becomes

$$\dot{\mu} = \sqrt{\det(JJ^{\mathrm{T}})}\sum_{i=1}^{k} w_i \mathrm{vec}^{\mathrm{T}}(JH_i^{\mathrm{T}})\mathrm{vec}((JJ^{\mathrm{T}})^{-1}). \quad (21)$$

Note that

$$\sum_{i=1}^{k} w_i \mathrm{vec}^{\mathrm{T}}(JH_i^{\mathrm{T}}) = [\,w_1,\, w_2,\, ...,\, w_k\,]\begin{bmatrix}\mathrm{vec}^{\mathrm{T}}(JH_1^{\mathrm{T}})\\ \mathrm{vec}^{\mathrm{T}}(JH_2^{\mathrm{T}})\\ ...\\ \mathrm{vec}^{\mathrm{T}}(JH_k^{\mathrm{T}})\end{bmatrix}. \quad (22)$$

To simplify the notation, we define a new operator "$\diamondsuit$" to capture the operation in (22) on matrices $H_1, H_2, ..., H_k$, and J as below:

$$J\diamondsuit\{H_1, H_2, ..., H_k\} = \begin{bmatrix}\mathrm{vec}^{\mathrm{T}}(JH_1^{\mathrm{T}})\\ \mathrm{vec}^{\mathrm{T}}(JH_2^{\mathrm{T}})\\ ...\\ \mathrm{vec}^{\mathrm{T}}(JH_k^{\mathrm{T}})\end{bmatrix} = \begin{bmatrix}\mathrm{vec}^{\mathrm{T}}(H_1^{\mathrm{T}})(I_m\otimes J^{\mathrm{T}})\\ \mathrm{vec}^{\mathrm{T}}(H_2^{\mathrm{T}})(I_m\otimes J^{\mathrm{T}})\\ ...\\ \mathrm{vec}^{\mathrm{T}}(H_k^{\mathrm{T}})(I_m\otimes J^{\mathrm{T}})\end{bmatrix} = \begin{bmatrix}\mathrm{vec}^{\mathrm{T}}(H_1^{\mathrm{T}})\\ \mathrm{vec}^{\mathrm{T}}(H_2^{\mathrm{T}})\\ ...\\ \mathrm{vec}^{\mathrm{T}}(H_k^{\mathrm{T}})\end{bmatrix}(I_m\otimes J^{\mathrm{T}}) \quad (23)$$

where the second equality is obtained as follows: Recall the equation $(X_2^{\mathrm{T}}\otimes X_1)\mathrm{vec}(X_3) = \mathrm{vec}(X_1X_3X_2)$ holds for matrices X_1, X_2, and X_3 of proper sizes. Setting $X_1 = J$, $X_2 = I_m$, and $X_3 = H_i^{\mathrm{T}}$ for $i = 1, 2, ..., k$ in this equation with I_m being the $m \times m$ identity matrix, we thus have $\mathrm{vec}(JH_i^{\mathrm{T}}) = (I_m\otimes J)\mathrm{vec}(H_i^{\mathrm{T}})$. In addition, recall the transposition rule over the Kronecker product as $(X_1\otimes X_2)^{\mathrm{T}} = X_1^{\mathrm{T}}\otimes X_2^{\mathrm{T}}$, with which we conclude $\mathrm{vec}^{\mathrm{T}}(JH_i^{\mathrm{T}}) = \mathrm{vec}^{\mathrm{T}}(H_i^{\mathrm{T}})(I_m\otimes J^{\mathrm{T}})$.

With the notation defined in (23) and the relation (22), (21) is converted to

$$\dot{\mu} = \sqrt{\det(JJ^{\mathrm{T}})}w^{\mathrm{T}}(J\diamondsuit\{H_1, H_2, ..., H_k\})\mathrm{vec}((JJ^{\mathrm{T}})^{-1}). \quad (24)$$

The above expression involves the matrix inversion $(JJ^{\mathrm{T}})^{-1}$, which is usually time-consuming for online computation. In order to avoid this operation, we treat this quantity as a variable and embed it into the solution procedure. To proceed, we re-examine the definition of matrix inverse, which gives the following for $(JJ^{\mathrm{T}})^{-1}$:

$$JJ^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1} = I_m. \quad (25)$$

Setting $X_1 = JJ^{\mathrm{T}}$, $X_2 = I_m$, $X_3 = A$, equality (25) is converted to

$$\mathrm{vec}(I_m) = \mathrm{vec}(JJ^{\mathrm{T}}A) = (I_m\otimes JJ^{\mathrm{T}})\mathrm{vec}((JJ^{\mathrm{T}})^{-1}). \quad (26)$$

C. Set Constraint

Due to physical constraints, the angular velocity of a manipulator cannot exceed certain limits, expressed as

$$\underline{w} \le w \le \bar{w} \quad (27)$$

where $\underline{w} \in \mathbb{R}^k$ and $\bar{w} \in \mathbb{R}^k$ are the lower and the upper bounds of w, respectively. The bound expression (27) can also be expressed as a convex set constraint:

$$w = \dot{\theta} \in \Omega \quad (28)$$

where the convex set Ω is defined as

$$\Omega = \{w \in \mathbb{R}^k, \underline{w} \le w \le \bar{w}\}. \quad (29)$$

For a redundant manipulator with kinematics (11) and a given desired workspace velocity v_d, the goal of kinematic redundancy resolution is to find a real-time control law $u = h(w, v_d)$, such

that the workspace velocity error $v - v_d$ converges to zero, and the joint angular velocity converges to the convex constraint set Ω as in (28).

D. Reformulation and Convexification

Define a new variable $\psi \in \mathbb{R}^{m^2}$, as an estimation of $\text{vec}((JJ^{\mathrm{T}})^{-1})$, to replace the expression of $\text{vec}((JJ^{\mathrm{T}})^{-1})$ in (24) and (26). Then, the manipulability optimization in speed level is formulated as

$$\min_{w,\psi} \; -\dot{\mu} = \; -\sqrt{\det(JJ^{\mathrm{T}})}w^{\mathrm{T}}(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi \tag{30a}$$

$$Jw = \; v_d \tag{30b}$$

$$(I_m \otimes JJ^{\mathrm{T}})\psi = \; \text{vec}(I_m) \tag{30c}$$

$$w \in \; \Omega. \tag{30d}$$

It is noteworthy that the term $\sqrt{\det(JJ^{\mathrm{T}})}$ is nonnegative and independent of the decision variables w and ψ. Therefore, it is equivalent to change the objective function to $w^{\mathrm{T}}(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi$. In addition, note that the objective function in (30) is bilinear but is nonconvex relative to the decision variables. We incorporate one extra term $\|w\|^2$ to regulate the kinematic energy consumption. Additionally, we augment the objective function with the equality constraints (30b) and (30c) to convexify the objective function as

$$\begin{aligned} &-c_3 w^{\mathrm{T}}(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi + c_0\|Jw - v_d\|^2/2 \\ &\quad + c_1\|(I_m \otimes JJ^{\mathrm{T}})\psi - \text{vec}(I_m)\|^2/2 + c_2\|w\|^2/2 \end{aligned} \tag{31}$$

where $c_0 > 0$, $c_1 > 0$, $c_2 > 0$, and $c_3 > 0$ are constants. Note that this new objective function is quadratic relative to w and ψ with $w^{\mathrm{T}}(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi$ being a cross term. Now, the reformulated optimization problem can be summarized as

$$\begin{aligned} \min_{w,\psi} \quad & -c_3 w^{\mathrm{T}}(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi + c_0\|Jw - v_d\|^2/2 \\ & + c_1\|(I_m \otimes JJ^{\mathrm{T}})\psi - \text{vec}(I_m)\|^2/2 + c_2\|w\|^2/2 \end{aligned} \tag{32a}$$

$$Jw = v_d \tag{32b}$$

$$(I_m \otimes JJ^{\mathrm{T}})\psi = \text{vec}(I_m) \tag{32c}$$

$$w \in \Omega. \tag{32d}$$

So far, the redundancy resolution for manipulability optimization has been formulated as a constrained quadratic programming. However, the solution to the optimization problem (30) cannot be obtained directly.

V. Neural Networks for Redundancy Resolution

In this section, we consider to develop neural dynamics for real-time solution of the optimal manipulability redundancy resolution problem (32). We first convert the problem into the solution of a nonlinear equation set and then establish neural dynamics for solving this nonlinear equation set.

A. Conversion to a Nonlinear Equation Set

To solve (32), we first convert it into a set of nonlinear equations. Define a Lagrange function as follows:

$$\begin{aligned} &L(w \in \Omega, \psi, \lambda, \sigma) = -c_3 w^{\mathrm{T}}(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi \\ &+ c_0\|Jw - v_d\|^2/2 + c_1\|(I_m \otimes JJ^{\mathrm{T}})\psi - \text{vec}(I_m)\|^2/2 \\ &+ c_2\|w\|^2/2 + \lambda^{\mathrm{T}}(Jw - v_d) \\ &+ \sigma^{\mathrm{T}}\big((I_m \otimes JJ^{\mathrm{T}})\psi - \text{vec}(I_m)\big). \end{aligned} \tag{33}$$

According to the Karush–Kuhn–Tucker condition [33], the solution of (32) has to satisfy the following:

$$-\frac{\partial L}{\partial w} \in N_\Omega(w), \; \frac{\partial L}{\partial \psi} = 0 \tag{34}$$

where $N_\Omega(w)$ denotes the normal cone of set Ω at w. Equation (34) includes normal cone operation $N_\Omega(w)$ in its expression. It can also be equivalently written in the following form with the aid of projection operators:

$$w = P_\Omega\left(w - \frac{\partial L}{\partial w}\right) \tag{35}$$

i.e.,

$$\begin{aligned} w = P_\Omega[&c_3(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi - c_0 J^{\mathrm{T}}(Jw - v_d) \\ &+ (1 - c_2)w - J^{\mathrm{T}}\lambda]. \end{aligned} \tag{36}$$

Together with $\partial L/\partial\psi = 0$ in (34), and the equation constraints in (32), the nonlinear equations for the optimization problem (32) can be summarized as

$$\begin{aligned} 0 = \; & -w + P_\Omega[c_3(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi \\ & - c_0 J^{\mathrm{T}}(Jw - v_d) + (1 - c_2)w - J^{\mathrm{T}}\lambda], \end{aligned} \tag{37a}$$

$$\begin{aligned} 0 = \; & -c_3(J\diamondsuit\{H_1, H_2, ..., H_k\})^{\mathrm{T}}w + c_1\big((I_m \otimes JJ^{\mathrm{T}}) \\ & \cdot[(I_m \otimes JJ^{\mathrm{T}})\psi - \text{vec}(I_m)]\big) + (I_m \otimes JJ^{\mathrm{T}})\sigma, \end{aligned} \tag{37b}$$

$$0 = \; Jw - v_d, \tag{37c}$$

$$0 = \; \text{vec}(I_m) - (I_m \otimes JJ^{\mathrm{T}})\psi. \tag{37d}$$

The above derivation can be summarized in the following lemma.

Lemma 5: The solution to the problem of manipulator redundancy resolution with maximal manipulability, i.e., the constrained optimization problem (32), is identical to the solution to the nonlinear equation set (37).

B. Neural Dynamics for Real-Time Redundancy Resolution

So far, we have converted the manipulability optimization problem into a nonlinear equation one. However, solving (37), due to its inherent nonlinearity, is a time-consuming task and usually is hard to implemented in real time. Additionally, notice that the nonlinear equation set (37) is in nature a time-varying system and its solution thus also varies with time. How to find the solution of (37) efficiently and keep track of it with time is a challenging problem. In this part, we present a dynamic neural network, which is able to address the problem recurrently.

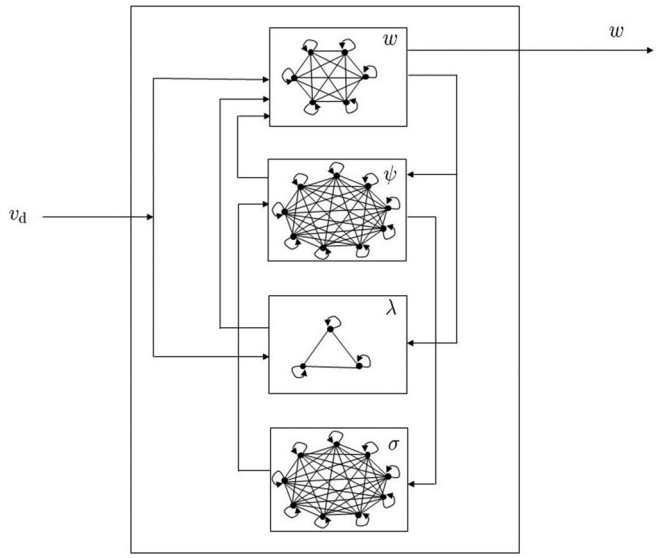

Fig. 1. Neural network architecture.

For the nonlinear equation set (37), we construct a dynamic neural network, as described by the following ordinary differential equations with the equilibrium point identical to (37), to solve the redundancy resolution problem:

$$\epsilon\dot{w} = -w + P_\Omega[c_3(J\diamondsuit\{H_1, H_2, ..., H_k\})\psi - c_0 J^{\mathrm{T}}(Jw - v_d) + (1 - c_2)w - J^{\mathrm{T}}\lambda], \quad (38a)$$

$$\epsilon\dot{\psi} = -c_3(J\diamondsuit\{H_1, H_2, ..., H_k\})^{\mathrm{T}}w + (I_m \otimes JJ^{\mathrm{T}})\sigma + c_1\left((I_m \otimes JJ^{\mathrm{T}})[(I_m \otimes JJ^{\mathrm{T}})\psi - \mathrm{vec}(I_m)]\right), \quad (38b)$$

$$\epsilon\dot{\lambda} = Jw - v_d, \quad (38c)$$

$$\epsilon\dot{\sigma} = \mathrm{vec}(I_m) - (I_m \otimes JJ^{\mathrm{T}})\psi \quad (38d)$$

where scaling factor $\epsilon > 0$.

The architecture of the proposed dynamic neural network (39) is shown in Fig. 1 for the situation with $m = 3$, $k = 6$ (m is the dimension of workspace, k is the number of joint angle of a PUMA560 redundant robot manipulator investigated in the ensuing simulations). From this figure, it is clear that the neural network is organized in a one-layer architecture, which is composed of $k + m + 2m^2$ neurons, and is a nonlinear layer with dynamic feedback. This layer of neurons is associated with the state variables $w \in \mathbb{R}^k$, $\psi \in \mathbb{R}^{m^2}$, $\lambda \in \mathbb{R}^m$, and $\sigma \in \mathbb{R}^{m^2}$, and gets input from v_d. It follows (39 a)–(39 d) for dynamic updates and maps the state variables to the output, which is the joint velocity w of the robot manipulator.

C. Convergence Analysis

This part proves the stability and convergence on the proposed neural network (39) via the following theorem.

Theorem 1: The dynamical neural network (39) is stable and is globally convergent to the optimal solution of (32).

Proof: Note that, in this proof, $J\diamondsuit H$ stands for $J\diamondsuit\{H_1, H_2, ..., H_k\}$ due to space limitation. By letting $\Upsilon = [w, \psi, \lambda, \sigma]^{\mathrm{T}}$, dynamical neural network (39) can be formulated as ■

$$\epsilon\dot{\Upsilon} = -\Upsilon + P_\Theta[\Upsilon - \varrho F(\Upsilon)] \quad (39)$$

where $\varrho = 1$, $P_\Theta(\cdot) = [P_\Omega, P_\Lambda]^{\mathrm{T}} \in \mathbb{R}^{2m^2+m+k}$ with $P_\Lambda \in \mathbb{R}^{2m^2+m}$ and with the upper and lower limits of set Λ being $\pm\infty$. Defining $F_1(\Upsilon) = -c_3(J\diamondsuit H)\psi + c_0 J^{\mathrm{T}}(Jw - v_d) + c_2 w + J^{\mathrm{T}}\lambda$ and $F_2(\Upsilon) = c_3(J\diamondsuit H)^{\mathrm{T}}w - c_1\left((I_m \otimes JJ^{\mathrm{T}}) \cdot [(I_m \otimes JJ^{\mathrm{T}})\psi - \mathrm{vec}(I_m)]\right) - (I_m \otimes JJ^{\mathrm{T}})\sigma$,

$$F(\Upsilon) = \begin{bmatrix} F_1(\Upsilon) \\ F_2(\Upsilon) \\ v_d - Jw \\ (I_m \otimes JJ^{\mathrm{T}})\psi - \mathrm{vec}(I_m) \end{bmatrix} \in \mathbb{R}^{k+2m^2+m}$$

and

$$\nabla F(\Upsilon) = \begin{bmatrix} \frac{\partial F_1(\Upsilon)}{\partial w} & \frac{\partial F_1(\Upsilon)}{\partial \psi} & \frac{\partial F_1(\Upsilon)}{\partial \lambda} & \frac{\partial F_1(\Upsilon)}{\partial \sigma} \\ \frac{\partial F_2(\Upsilon)}{\partial w} & \frac{\partial F_2(\Upsilon)}{\partial \psi} & \frac{\partial F_2(\Upsilon)}{\partial \lambda} & \frac{\partial F_2(\Upsilon)}{\partial \sigma} \\ -J & 0 & 0 & 0 \\ 0 & (I_m \otimes JJ^{\mathrm{T}}) & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} c_0 J^{\mathrm{T}}J + c_2 I & -c_3 J\diamondsuit H & J^{\mathrm{T}} & 0 \\ c_3(J\diamondsuit H)^{\mathrm{T}} & c_1(I_m \otimes JJ^{\mathrm{T}})^2 & 0 & -I_m \otimes JJ^{\mathrm{T}} \\ -J & 0 & 0 & 0 \\ 0 & I_m \otimes JJ^{\mathrm{T}} & 0 & 0 \end{bmatrix}.$$

Therefore, $F(\Upsilon)$ is continuously differentiable in view of the existence of $\nabla F(\Upsilon)$. In addition, it can be readily deduced that $I_m \otimes JJ^{\mathrm{T}} = (I_m \otimes JJ^{\mathrm{T}})^{\mathrm{T}}$. In view of the facts that $c_0 > 0$, $c_1 > 0$, and $c_2 > 0$, we further have

$$\nabla F(\Upsilon) + \nabla^{\mathrm{T}} F(\Upsilon) = \begin{bmatrix} 2c_0 J^{\mathrm{T}}J + 2c_2 I & 0 & 0 & 0 \\ 0 & 2c_1(I_m \otimes JJ^{\mathrm{T}})^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \succeq 0.$$

Based on Definition 3, it can be concluded that matrix $F(\Upsilon)$ is a semi-definite matrix and then we have that the corresponding mapping $F(\Upsilon)$ is monotone.

From Lemma 4, it is thus summarized and generalized from the above analyses that the proposed dynamical neural network (39) is stable and is globally convergent to the optimal solution of (32). The proof is thus completed. ■

VI. Illustrative Examples

In this section, computer simulations are conducted on a PUMA 560 manipulator with the parameters summarized in Table I to demonstrate the effectiveness of the proposed manipulability optimization scheme (32) as well as its dynamical

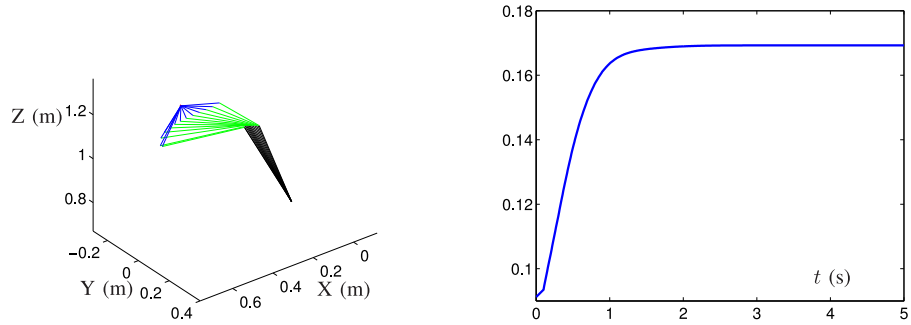

Fig. 2. Simulation results on motion trajectories and manipulability measures for the manipulability optimization of PUMA 560 manipulator via self motion with its end-effector fixed at [0.55, 0, 1.3] m in the workspace. (a) Motion trajectories. (b) Manipulability measures.

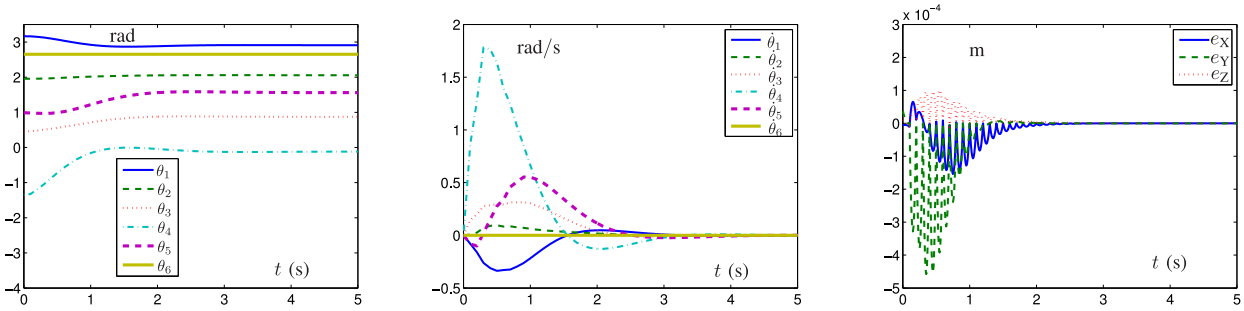

Fig. 3. Simulation results on profiles of joint-angle, joint-velocity, and position-error for the manipulability optimization of PUMA 560 manipulator via self motion with its end-effector fixed at [0.55, 0, 1.3] m in the workspace. (a) Joint-angle profiles. (b) Joint-velocity profiles. (c) Position-error profiles.

TABLE I
PUMA 560 D-H PARAMETERS

Link	a (m)	α (rad)	d (m)
1	0	$\pi/2$	0.67
2	0.4318	0	0
3	0.4318	$-\pi/2$	0.15005
4	0	$\pi/2$	0
5	0	$-\pi/2$	0
6	0	0	0.2

neural network solver (39). Being an articulated manipulator with six independently controlled joints, the PUMA 560 can reach any position in its workspace at any orientation via its end effector. In the following simulations, we consider only the three-dimension position control of the end effector, and thus, the PUMA 560 can be deemed as a redundant manipulator for this particular task.

A. Manipulability Optimization Via Self Motion

In this section, we perform simulations via self motion of PUMA 560 manipulator, i.e., by using different schemes to drive the manipulator from one state with low manipulability to another state with high manipulability and without moving the end-effector in Cartesian space. With parameters being set as $\epsilon = 0.01$, $c_0 = c_1 = 1$, $c_2 = c_3 = 0.01$, $k_0 = 5$, $\Omega = [-2, 2]^6$ and the rest ones being initially set as 0 (e.g., $\dot{w}$), a typical simulation run from a random initialization is generated as shown in Figs. 2 and 3. Specifically, as illustrated in Fig. 2(a), the manipulator adjusts its state with the end-effector being unmoved. As shown in Fig. 2(b), the manipulability measure $\mu = \sqrt{\det(JJ^{\mathrm{T}})}$ of the PUMA 560 manipulator increases to 0.17, which means that the proposed manipulability optimization scheme (32) is effective. The corresponding simulation results on the detailed profiles of joint-angle, joint-velocity, and position-error for the manipulability optimization of the PUMA 560 via self motion are illustrated in Fig. 3. It can be seen from Fig. 3 that, after a short-time adjustment, the joint angle θ of the PUMA 560 converges to a constant value in Fig. 3(a), and correspondingly the joint velocity $\dot{\theta}$ approaches to zero as shown in Fig. 3(b). The control error $e = r - r_d$, where r_d represents the reference position in workspace, approaches to zero with time as shown in Fig. 3(c). Overall, as shown in Figs. 2 and 3, the manipulability optimization of PUMA 560 manipulator via self-motion synthesized by the proposed manipulability optimization scheme (32) as well as its dynamical neural network solver (39) is illustrated preliminarily.

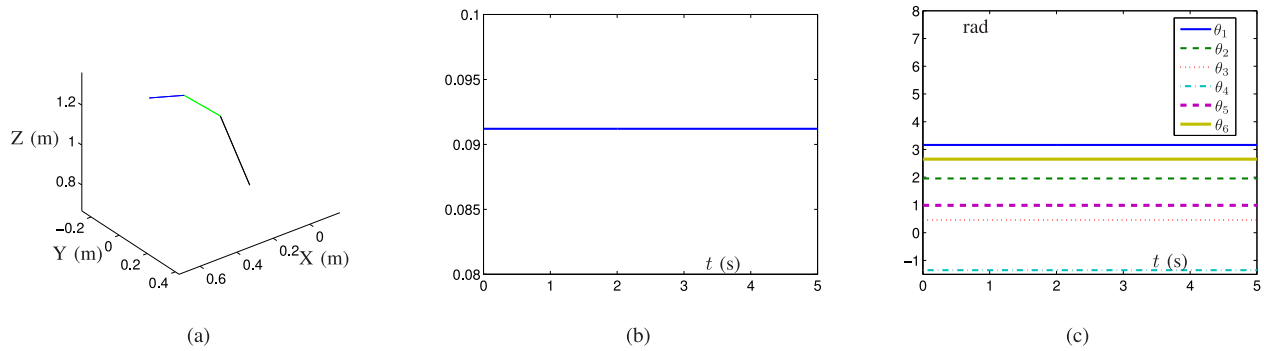

Fig. 4. Simulation results on motion trajectories, manipulability measures, and joint-angle profies of PUMA 560 synthesized by the scheme presented in [12] with its end-effector fixed at [0.55, 0, 1.3] m in the workspace. (a) Motion trajectories. (b) Manipulability measures. (c) Joint-angle profiles.

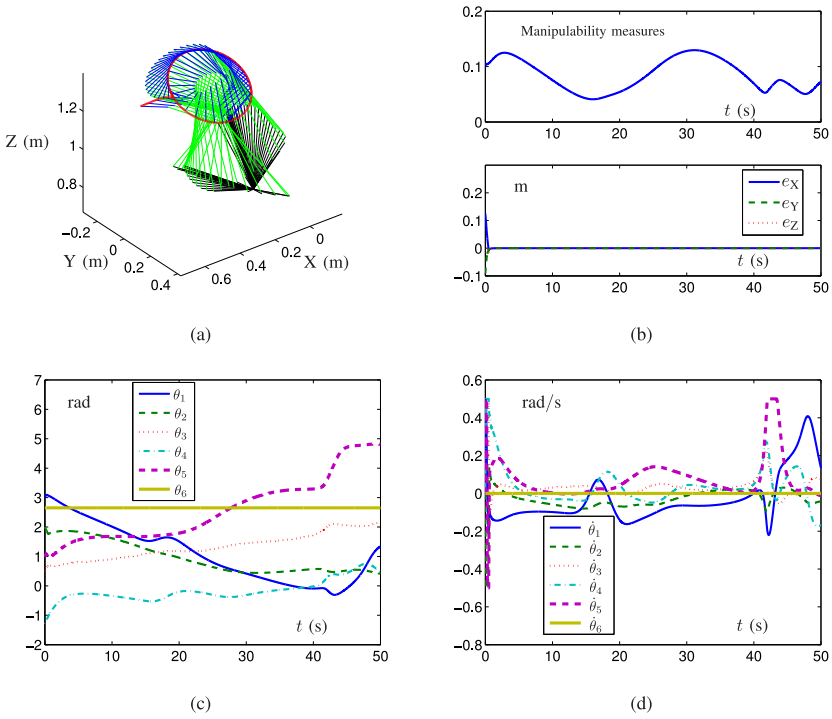

Fig. 5. Simulation results of PUMA 560 for tracking a circular path in the workspace synthesized by the proposed scheme (32). (a) Motion trajectories. (b) Manipulability measures and position-error profiles. (c) Joint-angle profiles. (d) Joint-velocity profiles.

In addition, as a comparison, simulations results of PUMA 560 manipulator synthesized by the scheme presented in [12] are illustrated in Fig. 4. With all the parameters being same as those in the simulations shown in Figs. 2 and 3, the PUMA 560 manipulator does not move and its manipulability keeps at the value of 0.091. In summary, these simulation results demonstrate the effectiveness of the proposed manipulability optimization scheme (32) in the increase of the manipulability as well as the singularity avoidance.

B. Manipulability Optimization in Circular Path Tracking

In this part, computer simulations synthesized by the proposed manipulability optimization scheme (32) are conducted to track a circular path. Specifically, the reference position of the PUMA 560's end-effector moves at an angular speed at 0.2 rad/s along a circle centered at $[0.25, 0, 1.3]$ m with radius 0.2 m and the revolute angle for 30° around the X-axis. In addition, the parameters are set as $\epsilon = 0.01$, $c_0 = c_1 = 1$, $c_2 = c_3 = 0.01$, $k_0 = 5$, $\Omega = [-0.5, 0.5]^6$ with the rest ones being initially set as 0 (e.g., $\dot{w}$). A typical simulation run, as shown in Fig. 5, can be generated using the proposed manipulability optimization scheme (32) with starting from a random initial configuration. As shown in Fig. 5(a), the end-effector of PUMA 560 manipulator tracks the circular path successfully in the three-dimension work-plane with its initial position being not on the desired path. The manipulability measures and position-error profiles are illustrated in Fig. 5(b), from which we can observe that the

TABLE II
COMPARISONS AMONG DIFFERENT SCHEMES FOR MANIPULABILITY OPTIMIZATION OF REDUNDANT MANIPULATORS

	Initial position	Singularity avoidance	Matrix inversion	Acceleration versus velocity	Physical limits avoidance	Average manipulability	Computational time with respecting to different radiuses r_a ‡ r_a = 0.2 m	r_a = 0.43 m	r_a = 0.45 m
This paper	Any	Yes	No	Velocity	Yes	0.0847	2.7 s	12.4 s	15.2 s
Scheme in [10]	Any	No	Yes	Velocity	No	0.0592	1.0 s	NA§	NA§
Scheme in [30]	Restrictive†	No	No	Acceleration	Yes	0.0606	1.4 s	15.6 s	18.4 s
Scheme in [12]	Restrictive†	No	No	Acceleration	Yes	0.0619	1.1 s	14.9 s	18.7 s
Scheme in [28]	Any	Yes	Yes	Velocity	Yes	0.0778	1.9 s	1029 s	NA§
Scheme (19) in [29]	Any	No	No	Velocity	Yes	0.0606	1.6 s	14.7 s	19.1 s
Scheme (41) in [29]	Any	No	No	Velocity	Yes	0.0605	1.7 s	13.6 s	17.9 s

† For the schemes in [12], [30], the initial position of the end-effector is required to be on the desired trajectory for tracking.
‡ $r_a = 0.2$ m, $r_a = 0.43$ m, $r_a = 0.45$ m, correspond to the situations of well-conditioned Jacobian matrix, almost singular Jacobian matrix and singular Jacobian matrix, respectively.
§ "NA" means that a crashed phenomenon arising in the solving process with the aid of the associated scheme

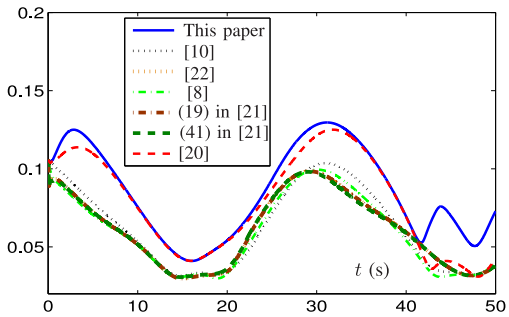

Fig. 6. Manipulability measures of PUMA 560 by different schemes.

manipulator is always away from singularity and the position error converges to zero after a short-time transient. In addition, the joint angle θ varies with time in Fig. 5(c). As shown in Fig. 5(d), the joint velocity $\dot{\theta}$ is kept within the limits. These results verify the effectiveness of our method.

C. Comparisons

In this part, we compare the proposed manipulability optimization scheme (32) with existing solutions for tracking control of redundant manipulators presented in [10], [12], [28]–[30]. Scheme presented in [12] extends results in [30] from an equivalent relationship viewpoint on the velocity-level redundancy resolution and acceleration-level resolution. In this paper, we focus on the velocity-level resolution and the proposed manipulability optimization scheme (32) applies to it. As summarized in Table II, the proposed one in this paper is able to deal with the manipulability optimization problem in an inverse-free manner, while existing ones [10], [12], [28]–[30] cannot handle such a knotty problem. In addition, due to the lack of direct position feedback in [12], [30], the initialization of the manipulator's end-effector is strictly restricted to the desired position on the desired path. Moreover, as indicated in Table II as well as Fig. 6, another major difference lies that the proposed scheme (32) always maximizes the manipulability of the manipulators, while others show less satisfying performance.

It also can be observed from the table that, for the situation of well-conditioned Jacobian matrix, the total computational time related to the proposed scheme (32) is 2.7 s, which is longer than that of the rest ones listed in the table. This is because that extra effort is paid for the optimization of manipulability. The total computational time of scheme (32) is much shorter than the task duration (i.e., 50 s), which means that such a scheme can be employed to the online solution of motion generation of redundant robot manipulators. In addition, it can be found in the table that, for the situation of almost singular Jacobian matrix, the scheme presented in [28] consumes 1029 s to fulfill the task, much longer than the task duration (i.e., 50 s), which means that such a scheme cannot be employed to the online solution of motion generation of redundant robot manipulators. By contrast, the total computational time related to scheme (32) is 12.4 s, which is much shorter due to the avoidance of time-consuming matrix inversion when the matrix is almost singular. It is also noteworthy that as the benefit of manipulability optimization, the proposed scheme can quickly solve the control action while others without optimizing manipulability have to update control actions very frequently when their Jacobian reaches singular, resulting the fact that the proposed scheme consumes less time than those without manipulability optimization [12], [29], [30]. Moreover, it can be observed from the table that, for the situation of singular Jacobian matrix, the scheme presented in [28] cannot fulfill such a given path-tracking task, which is mainly because of the real-time matrix-inversion operation involved in the scheme. Therefore, this scheme fails when encountering a singular Jacobian matrix. By contrast, as shown in the table, even without the ability to maximize the manipulability during the task execution process, these QP-based schemes presented in [12], [29], [30] could generate an approximated solution with no feasible solution at the cost of large position errors. As a QP-based control law designed for manipulability optimization of redundant robot manipulators in an inverse-free manner, the proposed manipulability optimization scheme (32) could handle such a knotty problem in an acceptable way. That is, our scheme could always maximize the manipulability in the situation of nonsingular Jacobian matrix, and generate an approximated solution for commanding the manipulator motion in the situation

of singular Jacobian matrix with the superior computational efficiency and lesser position errors compared with other existing schemes.

Besides, it is worth comparing here the manipulability optimization scheme (32) and the manipulability-maximizing scheme presented in [28], both of which are exploited to maximize the manipulability during the motion generation of redundant robot manipulators. To lay a basis for discussion, the manipulability optimization scheme presented in [28] is rewritten as

$$\min_{w} \quad \frac{1}{2}w^{\mathrm{T}}w - (\det(JJ^{\mathrm{T}})\mathrm{tr}((JJ^{\mathrm{T}})^{-1}\frac{\partial(JJ^{\mathrm{T}})}{\partial\theta}))^{\mathrm{T}}w \quad (40a)$$

$$Jw = v_d \quad (40b)$$

$$w \in \Omega. \quad (40c)$$

The reasons for the different performance of these schemes can be explained intuitively as follows. The manipulability optimization scheme (40) has more computational load due to the matrix inversion involved, i.e., the matrix JJ^{T} has to be inverted in real time. On the contrary, the proposed manipulability optimization scheme (32) is able to deal with the manipulability optimization problem in an inverse-free manner. On the one hand, for the well-conditioned Jacobian matrix, such a matrix inversion operation can be readily done and scheme (40) seems to be effective in this situation. On the other hand, it is unnecessary to implement a scheme for maximizing the manipulability with well-conditioned Jacobian matrix since the given task is far away from singularity. It is worth noting that, as matrix inversion operation existing in scheme (40), it would take much longer time than scheme (32) to compute the inverse of an almost singular Jacobian matrix, which may violate real-time requirements on the motion generation of redundant robot manipulators. What is worse, for the situation of singular Jacobian matrix, the manipulability optimization scheme (40) cannot work since there does not exist the inverse of a singular matrix. Note that, as long as one eigenvalue of the matrix JJ^{T} is zero, scheme (40) would fail to compete the task. However, the proposed manipulability optimization scheme (32) can handle such a knotty problem.

VII. Conclusion

In this paper, we established a dynamic neural network for recurrent calculation of manipulability-maximal control actions for redundant manipulators under physical constraints in an inverse-free manner. By expressing position tracking and matrix inversion as equality constraints, physical limits as inequality constraints, and velocity-level manipulability measure, which is affine to the joint velocities, as the objective function, the manipulability optimization scheme was further formulated as a constrained QP. Then, a dynamic neural network with rigorously provable convergence was constructed to solve such a problem online. Computer simulations show that compared to the existing methods, the proposed scheme can raise the manipulability by 40% on average, which substantiates the efficacy, accuracy, and superiority of the proposed scheme.

References

[1] S. Li, R. Kong, and Y. Guo, "Cooperative distributed source seeking by multiple robots: Algorithms and experiments," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 6, pp. 1810–1820, Dec. 2014.

[2] M. Li, W. Li, L. Niu, H. Zhou, G. Chen, and F. Duan, "An event-related potential-based adaptive model for telepresence control of humanoid robot motion in an environment with cluster obstacles," *IEEE Trans. Ind. Electron.*, vol. 64, no. 2, pp. 1696–1705, Feb. 2017. doi: 10.1109/TIE.2016.2538740.

[3] H. Xiao, Z. Li, and C. L. P. Chen, "Formation control of leader-follower mobile robots' systems using model predictive control based on neural-dynamic optimization," *IEEE Trans. Ind. Electron.*, vol. 63, no. 9, pp. 5752–5762, Sep. 2016.

[4] 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.

[5] Z. Li, Z. Huang, W. He, and C. Su, "Adaptive impedance control for an upper limb robotic exoskeleton using biological signals," *IEEE Trans. Ind. Electron.*, vol. 64, no. 2, pp. 1664–1674, Feb. 2017. doi: 10.1109/TIE.2016.2538741.

[6] Z. Zhao, Z. Li, R. Cui, and Y. Kang, "Brain-machine interfacing-based teleoperation of multiple coordinated mobile robots," *IEEE Trans. Ind. Electron.*, to be published. doi: 10.1109/TIE.2016.2606089.

[7] D. Guo and Y. Zhang, "Acceleration-level inequality-based MAN scheme for obstacle avoidance of redundant robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6903–6914, Dec. 2014.

[8] L. Jin, Y. Zhang, S. Li, and Y. Zhang, "Modified ZNN for time-varying quadratic programming with inherent tolerance to noises and its application to kinematic redundancy resolution of robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 63, no. 11, pp. 6978–6988, Nov. 2016.

[9] N. Nikdel, M. Badamchizadeh, V. Azimirad, and M. A. Nazari, "Fractional-order adaptive backstepping control of robotic manipulators in the presence of model uncertainties and external disturbances," *IEEE Trans. Ind. Electron.*, vol. 63, no. 10, pp. 6249–6256, Oct. 2016.

[10] D. Guo and Y. Zhang, "Li-function activated ZNN with finite-time convergence applied to redundant-manipulator kinematic control via time-varying Jacobian matrix pseudoinversion," *Appl. Soft Comput.*, vol. 24, pp. 158–168, Nov. 2014.

[11] L. Jin and Y. Zhang, "Discrete-time Zhang neural network of $O(\tau^3)$ pattern for time-varying matrix pseudoinversion with application to manipulator motion generation," *Neurocomputing*, vol. 142, pp. 165–173, Oct. 2014.

[12] B. Cai and Y. Zhang, "Different-level redundancy-resolution and its equivalent relationship analysis for robot manipulators using gradient-descent and Zhang et al's neural-dynamic methods," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3146–3155, Aug. 2012.

[13] S. Li, J. He, U. Rafique, and Y. Li, "Distributed recurrent neural networks for cooperative control of manipulators: A game-theoretic perspective," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 2, pp. 415–426, Feb. 2017. doi: 10.1109/TNNLS.2016.2516565.

[14] S. Li and U. Rafique, "A novel recurrent neural network for manipulator control with improved noise tolerance," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2017.2672989.

[15] Y. Zhang and S. Li, "Predictive suboptimal consensus of multi-agent systems with nonlinear dynamics," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2017.2668440.

[16] L. Jin and S. Li, "Distributed task allocation of multiple robots: A control perspective," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2016.2627579.

[17] L. Jin, Y. Zhang, S. Li, and Y. Zhang, "Noise-tolerant ZNN models for solving time-varying zero-finding problems: A control-theoretic approach," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 992–997, Feb. 2017.

[18] L. Jin, Y. Zhang, and S. Li, "Integration-enhanced Zhang neural network for real-time varying matrix inversion in the presence of various kinds of noises," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2615–2627, Dec. 2016.

[19] H. Wang, X. Liu, and K. Liu, "Robust adaptive neural tracking control for a class of stochastic nonlinear interconnected systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 510–523, Mar. 2016.

[20] L. Xiao and Y. Zhang, "Dynamic design, numerical solution and effective verification of acceleration-level obstacle-avoidance scheme for robot manipulators," *Int. J. Syst. Sci.*, vol. 47, no. 4, pp. 932–945, 2016.

[21] H. Wang, K. Liu, X. Liu, and B. Chen, "Neural-based adaptive output-feedback control for a class of nonstrict-feedback stochastic nonlinear systems," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1977–1987, Sep. 2015.
[22] Z. Zhang, Z. Li, Y. Zhang, Y. Luo, and Y. Li, "Neural-dynamic-method-based dual-arm CMG scheme with time-varying constraints applied to humanoid robots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3251–3262, Mar. 2015.
[23] C. Luo, S. X. Yang, X. Li, and M. Q.-H. Meng, "Neural dynamics driven complete area coverage navigation through cooperation of multiple mobile robots," *IEEE Trans. Ind. Electron.*, vol. 64, no. 1, pp. 750–760, Jan. 2017. doi: 10.1109/TIE.2016.2609838.
[24] J. Na, Q. Chen, X. Ren, and Y. Guo, "Adaptive prescribed performance motion control of servo mechanisms with friction compensation," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 486–494, Jan. 2014.
[25] S. Li, M. Zhou, X. Luo, and Z. You, "Distributed winner-take-all in dynamic networks," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 577–589, Feb. 2017. doi: 10.1109/TAC.2016.2578645.
[26] S. Li, B. Liu, and Y. Li, "Selective positive-negative feedback produces the winner-take-all competition in recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 301–309, Feb. 2013.
[27] Y. Xia, G. Feng, and J. Wang, "A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations," *Neural Netw.*, vol. 17, no. 7, pp. 1003–1015, Sep. 2004.
[28] Y. Zhang, X. Yan, D. Chen, D. Guo, and Y. Li, "QP-based refined manipulability-maximizing scheme for coordinated motion planning and control of physically constrained wheeled mobile redundant manipulators," *Nonlinear Dyn.*, vol. 85, no. 1, pp. 245–261, Jul. 2016.
[29] S. Li, Y. Zhang, and L. Jin, "Kinematic control of redundant manipulators using neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2016.2574363.
[30] 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. 14, no. 1, pp. 426–667, Feb. 2005.
[31] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robot. Res.*, vol. 4, no. 2, pp. 3–9, Jun. 1985.
[32] Z. Gao and D. Zhang, "Performance analysis, mapping, and multiobjective optimization of a hybrid robotic machine tool," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 423–433, Jan. 2015.
[33] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[34] H. Khalil, *Nonlinear Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996.
[35] X. Gao, "Exponential stability of globally projected dynamic systems," *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 426–431, Mar. 2003.
[36] M. Spong and M. Vidyasagar, *Robot Dynamics and Control*. Hoboken, NJ, USA: Wiley, 2008.

Long Jin received the B.E. and Ph.D. degrees in electrical engineering from Sun Yat-sen University, Guangzhou, China, in 2011 and in 2016, respectively.

He is currently a Postdoctoral Fellow in the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. His main research interests include robotics, neural networks, and intelligent information processing.

Shuai Li (M'14) received the B.E. degree in precision mechanical engineering from 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. degree in electrical and computer engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2014.

He is currently a Research Assistant Professor in the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. His current research interests include dynamic neural networks, robotic networks, and other dynamic problems defined on a graph.

Hung Manh La (M'09–SM'14) received the B.S. and M.S. degrees in electrical engineering from Thai Nguyen University of Technology, Thai Nguyen, Vietnam, in 2001 and 2003, respectively, and the Ph.D. degree in electrical and computer engineering from Oklahoma State University, Stillwater, OK, USA, in 2011.

He is the Director of the Advanced Robotics and Automation (ARA) Laboratory, and an Assistant Professor in the Department of Computer Science and Engineering, University of Nevada, Reno, NV, USA. From 2011 to 2014, he was a Postdoctoral Research Fellow and then a Research Faculty Member in the Center for Advanced Infrastructure and Transportation, Rutgers University, New Brunswick, NJ, USA.

Dr. La is an Associate Editor of the IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS, and a Guest Editor of the *International Journal of Robust and Nonlinear Control*.

Xin Luo (M'14) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011.

He joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China in 2016, and is currently a Professor of computer science and engineering. His research interests include big data analysis and intelligent control. He has published 60+ papers (including 10+ IEEE TRANSACTIONS papers) in his related areas.