Bi-criteria Velocity Minimization of Robot Manipulators
Using a Linear Variational Inequalities-Based Primal-Dual
Neural Network and PUMA560 Example

Yunong Zhang\textsuperscript{a,*}, Binghuang Cai\textsuperscript{a}, Lei Zhang\textsuperscript{b} and Kene Li\textsuperscript{a}
\textsuperscript{a} Department of Electronics and Communication Engineering, Sun Yat-Sen University, Guangzhou 510275, China
\textsuperscript{b} School of Software, Sun Yat-Sen University, Guangzhou 510275, China
Received 7 January 2008; accepted 20 May 2008

Abstract
In this paper, to diminish discontinuity points arising in the infinity-norm velocity minimization scheme, a bi-criteria velocity minimization scheme is presented based on a new neural network solver (i.e., a primal-dual neural network based on linear variational inequalities (LVI)). Such a kinematic control scheme of redundant manipulators can incorporate joint physical limits such as joint limits and joint velocity limits simultaneously. Moreover, the presented kinematic control scheme can be formulated as a quadratic programming (QP) problem. As a real-time QP solver, the LVI-based primal-dual neural network is established with a simple piecewise linear structure and higher computational efficiency. Computer simulations performed based on a PUMA560 manipulator are presented to illustrate the validity and advantages of such a bi-criteria neural control scheme for redundant robots.


Keywords
Bi-criteria velocity minimization, quadratic programming, linear variational inequalities-based primal-dual neural network, PUMA560 robot manipulator

1. Introduction
A manipulator is said to be redundant when more degrees of freedom (d.o.f.) are available to execute a given specific end-effector task [1]. This implies that redundancy can be established simply with respect to some particular tasks. For non-redundant manipulators, the joint motion is uniquely determined by a prescribed end-effector trajectory and, thus, there is no freedom left to handle joint physical

* To whom correspondence should be addressed. E-mail: ynzhang@ieee.org

limits and/or environmental constraints such as obstacles. In comparison, redundant manipulators have a wider operational space and extra d.o.f. to meet a number of functional constraints, because a large number of joint configurations could be feasible solutions. They have, thus, been developed for use in the carrying and cleanup of nuclear and hazardous materials, as well as in remote applications such as space or sea exploration.

A fundamental issue in controlling robot manipulators is the redundancy resolution problem. A redundancy resolution scheme is a method or an algorithm that selects one joint space solution (trajectory) from a large number of possible solutions, given the primary end-effector task of following a desired workspace trajectory in terms of position and orientation. The scheme usually accomplishes several secondary tasks, in addition to its primary end-effector tracking task. Furthermore, the redundancy resolution scheme could often exploit optimization techniques.

Over the past two decades, researchers have resolved the redundancy problem mainly at the level of joint velocities [2–7]. The conventional minimum velocity-norm (MVN) solution (including a pseudoinverse-type solution) has been widely investigated by the vast majority of researchers. As a two-norm (also termed a Euclidian-norm) scheme, it minimizes the sum of squared joint velocities, which may not necessarily minimize the magnitudes of individual joint velocities. It is used as the optimization criterion in many robotic applications more because of its mathematical tractability than physical desirability [7]. On the other hand, the infinity-norm velocity minimization (INVM) solution, also known as the minimum effort solution or the minimum amplitude solution, explicitly minimizes the largest component of the joint velocity vector in magnitude and is more consistent with joint physical limits. In addition, the INVM enables a better direct monitoring and control of the magnitudes of individual joint velocities (e.g., in safety-critical robotic systems) [8–12]. It is therefore more desirable in situations where primary concerns are on low individual magnitude of joint velocities, even distribution of workload and/or motion diversity analysis.

The minimum infinity-norm solution may, however, encounter a discontinuity problem. It is shown in Refs [3, 9] that a discontinuity point of the minimum infinity-norm solution exists, possibly because of the non-uniqueness at such a solution point and the separation of two successive solution sets. To remedy the discontinuity problem, a balancing scheme is presented in Ref. [9] which calculates the minimum infinity-norm and two-norm solutions separately, and then incorporates the two weighted solutions as the final solution. With $\dot{\theta}^*$, $\dot{\theta}^{(\infty)}$ and $\dot{\theta}^{(2)}$ denoting, respectively, the final solution, the minimum infinity-norm solution and the minimum two-norm solution, such a balancing scheme is:

$$\dot{\theta}^* = \alpha \dot{\theta}^{(2)} + (1 - \alpha) \dot{\theta}^{(\infty)}, \quad 0 \leq \alpha \leq 1.$$

The computation of the inverse kinematics solution is usually time consuming, especially for high-d.o.f. robotic systems, and for the usual cases with many subtask criteria and/or physical constraints (e.g., joint limits and joint velocity limits).
Compared to a single minimum-norm solution, the balancing scheme (1) may at least double the computational time, which is evidently less efficient and may hinder online sensor-based robotic applications. Parallel computational methods such as neural network approaches have, therefore, been proposed, being effective and efficient alternatives to real-time solutions of such a balanced inverse kinematics problem.

Various kinds of recurrent neural networks have recently been developed and investigated for the optimization part of robot manipulator control, e.g., Refs [2–4, 11–22]. In particular, a neural computation approach [12] which incorporates a pseudoinverse network [19] and a linear programming network [22] is applied to the minimum infinity-norm kinematic control. As an improved neural network model of Ref. [12], a two-layered primal-dual neural network [11] is then presented to minimize online the infinity-norm of joint velocities. To reduce network complexity and to increase computational efficiency, a single-layered dual neural network [20] is proposed for kinematic control of redundant manipulators. In the aforementioned neural schemes, it is usually assumed in an implicit manner that there exist no joint limits or joint velocity limits when solving the inverse kinematics problem. However, joint physical limits always exist to some extent. If a solution exceeds the mechanical joint limits or velocity limits, the desired motion path may fail to execute, not to mention the possibility of mechanical damage [23].

The work presented in this paper aims at explicitly remedying the discontinuity problem of joint velocities via a bi-criteria neural weighting scheme. Compared with a single-criterion scheme, such a bi-criteria scheme is much more flexible in the sense that it can yield any combination of the minimum effort and minimum energy solutions as needed. In this paper, we first present a bi-criteria velocity minimization scheme for redundant manipulators, then formulate it as a quadratic programming (QP) problem, and finally exploit the primal-dual neural network based on linear variational inequalities (LVI) to solve online the QP problem. Computer simulations, based on the PUMA560 robot arm, illustrate the efficacy of such a bi-criteria neural redundancy resolution scheme.

2. Problem Formulation

The end-effector position-and-orientation vector \( r(t) \in \mathbb{R}^m \) in Cartesian space is related to the joint space vector \( \theta(t) \in \mathbb{R}^n \) through a forward kinematic equation:

\[
r(t) = f(\theta(t)).
\]

Unfortunately, (2) is usually difficult to solve due to the nonlinearity and redundancy of \( f(\cdot) \). By differentiating (2), the redundancy resolution problem could be resolved at the joint velocity level:

\[
J(\theta) \dot{\theta} = \dot{r},
\]

where \( \dot{r} \) and \( \dot{\theta} \) denote the \( m \)-dimensional end-effector velocity vector and the \( n \)-dimensional joint velocity vector, respectively. Jacobian matrix \( J(\theta) \) is defined as
\[ \partial f(\theta)/\partial \theta \in \mathbb{R}^{m \times n}. \] Note that, as the manipulator system is redundant (i.e., \( m < n \)), (2) and (3) are under-determined, admitting an infinite number of solutions. Based on the weighting idea of (1), we proposed a bi-criteria redundancy resolution scheme as follows [4]:

\[
\begin{align*}
\text{minimize} & \quad (\alpha \| \dot{\theta} \|_2^2 + (1 - \alpha) \| \dot{\theta} \|_\infty^2) / 2 \\
\text{subject to} & \quad J(\theta)\dot{\theta} = \dot{r} \\
& \quad \theta^- \leq \theta \leq \theta^+ \\
& \quad \dot{\theta}^- \leq \dot{\theta} \leq \dot{\theta}^+, 
\end{align*}
\] (4)

where:

- The weighting factor \( \alpha \in (0, 1) \).
- \( \| \cdot \|_2 \) and \( \| \cdot \|_\infty \) denote, respectively, the two-norm and infinity-norm of a vector.
- \( \theta^\pm \) and \( \dot{\theta}^\pm \) denote the upper and lower physical limits for joint vector and joint velocity vector, respectively.

In this paper, we further the research by considering the following bi-criteria redundancy resolution scheme formulation:

\[
\begin{align*}
\text{minimize} & \quad (\alpha \| \dot{\theta} \|_2^2 + (1 - \alpha) \| \dot{\theta} \|_\infty^2) / 2 \\
\text{subject to} & \quad J(\theta)\dot{\theta} = \dot{r} \\
& \quad \theta^- \leq \theta \leq \theta^+ \\
& \quad \dot{\theta}^- \leq \dot{\theta} \leq \dot{\theta}^+, 
\end{align*}
\] (5)

where, different from (4):

- The weighting factor \( \alpha \in [0, 1] \) (which could be exactly 0 and 1, thus incorporating the INVM and MVN schemes as special cases).
- The infinity-norm part of performance index (5) is \( \| \dot{\theta} \|_\infty \) instead of \( \| \dot{\theta} \|_\infty^2 \) (of which the possible theoretical and engineering differences attracted thesis committee members’ attention [4]).

In this paper, we will show the following facts about the similarities and differences of schemes (4) and (5):

- Similar to scheme (4), the new redundancy resolution scheme (5)–(8) could also be reformulated as QP.
- However, the QP problem corresponding to scheme (4) is strictly convex (with \( 0 < \alpha < 1 \) and \( Q > 0 \)), which was solved by a dual neural network [4].
- In comparison, the QP problem corresponding to scheme (5)–(8) is only convex (which is not necessarily strictly convex) with \( 0 \leq \alpha \leq 1 \) and \( Q \geq 0 \). This QP could be solved by the proposed primal-dual neural network (instead of the dual neural network).
As practically observed, using schemes (4) or (5), we could have similar joint trajectory curves.

However, the aforementioned similar trajectory curves correspond to different values of $\alpha$ in schemes (4) and (5), as the physical meaning of $\alpha$ changes. For example, $\alpha = 0.6$ in scheme (4) and $\alpha = 0.9$ in scheme (5) generate similar joint trajectory curves (see Fig. A1 in the Appendix).

To show the above facts, we need to reformulate the redundancy resolution scheme as a QP problem through the following two subsections.

2.1. Conversion of Bound Constraints

The limited joint range (7) can be reformulated in terms of $\dot{\theta}$ by using the following variable bound constraint:

$$\kappa_p(\theta^- - \theta) \leq \dot{\theta} \leq \kappa_p(\theta^+ - \theta),$$

where the positive coefficient $\kappa_p$ is used to scale the feasible region of $\dot{\theta}$ and is usually selected as 20 in our PUMA560-based computer simulations. Joint limits (7) and joint velocity limits (8) can thus be combined into a unified variable bound constraint:

$$\eta^- \leq \dot{\theta} \leq \eta^+,$$

where the $i$th elements of $\eta^-$ and $\eta^+$ are defined, respectively, as:

$$\eta^-_i = \max\{\dot{\theta}^-_i, \kappa_p(\theta^-_i - \theta_i)\}, \quad \eta^+_i = \min\{\dot{\theta}^+_i, \kappa_p(\theta^+_i - \theta_i)\}.$$

2.2. Conversion of Performance Index

Now, let us convert the minimum infinity-norm part of performance index (5) into a QP (which includes the linear program as a special case). For $\dot{\theta} = [\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_n]^T$ with the superscript $T$ denoting the transpose operator of a matrix or vector, its infinity-norm $\|\dot{\theta}\|_{\infty}$ is defined as:

$$\|\dot{\theta}\|_{\infty} = \max\{|\dot{\theta}_1|, |\dot{\theta}_2|, \ldots, |\dot{\theta}_n|\} = \max_{1 \leq j \leq n} |e_j^T \dot{\theta}|,$$

where $|\cdot|$ denotes the absolute value of a scalar and $e_j \in R^n$ is the $j$th column of identity matrix $I$. By defining $s(t) = \|\dot{\theta}(t)\|_{\infty}$, we could have the following conversion theorem.

**Theorem 1.** The minimization of $(1 - \alpha)\|\dot{\theta}(t)\|_{\infty}/2$ is equivalent to:

$$\text{minimize} \quad (1 - \alpha)s(t)/2$$

$$\text{subject to} \quad |e_j^T \dot{\theta}| \leq s(t),$$

which can be written further as:

$$\text{minimize} \quad (1 - \alpha)s(t)/2$$

$$\text{subject to} \quad \begin{bmatrix} I & -1_v \\ -I & -1_v \end{bmatrix} \begin{bmatrix} \dot{\theta}(t) \\ s(t) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$
where \(1_v := [1, 1, \ldots, 1]^T\) is an appropriately dimensioned vector made of ones.

**Proof.** Can be generalized from Refs [3, 4, 11].

2.3. QP Reformulation

By defining the augmented variable vector \(x = [\dot{\theta}^T, s]^T \in \mathbb{R}^{n+1}\), the bi-criteria redundancy resolution scheme (5)–(7) could be expressed as the following QP:

\[
\begin{align*}
\text{minimize} & \quad x^T Q x/2 + p^T x \\
\text{subject to} & \quad C x = d \\
& \quad Ax \leq b \\
& \quad x^- \leq x \leq x^+, \tag{13}
\end{align*}
\]

where the coefficient matrices and vectors are defined as:

\[
Q = \begin{bmatrix}
\alpha I & 0 \\
0 & 0
\end{bmatrix} \in \mathbb{R}^{(n+1) \times (n+1)}, \quad p = \begin{bmatrix}
0 \\
(1 - \alpha)/2
\end{bmatrix} \in \mathbb{R}^{n+1},
\]

\[
C = \begin{bmatrix}
J(\theta) & 0
\end{bmatrix} \in \mathbb{R}^{m \times (n+1)}, \quad d = \dot{r}(t) \in \mathbb{R}^m,
\]

\[
A = \begin{bmatrix}
I & -1_v \\
-I & -1_v
\end{bmatrix} \in \mathbb{R}^{2n \times (n+1)}, \quad b = 0 \in \mathbb{R}^{2n},
\]

\[
x^- = \begin{bmatrix}
\eta^- \\
0
\end{bmatrix} \in \mathbb{R}^{n+1}, \quad x^+ = \begin{bmatrix}
\eta^+ \\
\sigma\varpi
\end{bmatrix} \in \mathbb{R}^{n+1},
\]

with constant \(\sigma \gg 0\) being sufficiently large to replace \(+\infty\) numerically.

**Remark.** It is worth mentioning that, different from other researchers’ work and our previous work [4], \(\alpha\) here can exactly be 0 or 1. This is because we will use the LVI-based primal-dual neural network to solve the above QP problem, instead of using the dual neural network. The LVI-based primal-dual neural network does not entail any matrix inversion and thus it can handle any value of \(\alpha\), in addition to reducing computational time considerably. When \(\alpha = 0\), the bi-criteria solution \(\dot{\theta}^*\) becomes the minimum infinity-norm solution \(\dot{\theta}^{(\infty)}\). When \(\alpha = 1\), the bi-criteria solution \(\dot{\theta}^*\) reduces the minimum two-norm solution \(\dot{\theta}^{(2)}\). In addition, the presented bi-criteria scheme could generate any solution between the INVM and MVN when \(0 < \alpha < 1\). In this sense, the presented bi-criteria resolution scheme is much more flexible than a single-criterion resolution scheme. Moreover, the value of \(\alpha\) could be set close to 0 if small joint velocity amplitude is of primary concern, whereas \(\alpha\) could be set close to 1 if small velocity energy is of main concern. With many thanks to reviewers for inspiring suggestions, a future research direction may lie in the mathematical optimization of such a weighting factor \(\alpha\).
3. Neural Network Solver

In this section, by using the following design method, we develop a LVI-based primal-dual neural network, which solves the QP (10)–(13) as well as redundancy resolution scheme (5)–(8). First, we can convert QP (10)–(13) to a set of LVI. That is, to find a primal-dual equilibrium vector \( y^* \in \Omega := \{ y \mid y^- \leq y \leq y^+ \} \subset R^{3n+m+1} \) such that \( \forall y \in \Omega \):

\[
(y - y^*)^T (Wy^* + q) \geq 0,
\]

where the primal-dual decision vector \( y \) and its lower/upper bounds are defined, respectively, as:

\[
y = \begin{bmatrix} x \\ u \\ v \end{bmatrix}, \quad y^- = \begin{bmatrix} x^- \\ -1, \omega \end{bmatrix}, \quad y^+ = \begin{bmatrix} x^+ \\ 1, \omega \end{bmatrix},
\]

with \( u \) and \( v \) being dual decision vectors defined, respectively, for equality constraint (11) and inequality constraint (12). In addition, the augmented coefficients are defined as:

\[
W = \begin{bmatrix} Q & -C^T \\ C & 0 \\ -A & 0 \end{bmatrix}, \quad q = \begin{bmatrix} p \\ -d \\ b \end{bmatrix}.
\]

In summary, we could have the following important results.

**Theorem 2.** With the existence of at least one optimal solution \( x^* \), the QP problem (10)–(13) could be converted to the LVI problem (14).

**Proof.** Can be generalized from Refs [17, 24] by taking into account dual decision vector \( v \geq 0 \), which is introduced corresponding to inequality constraint (12).

Second, it is known (from Refs [17, 24] and references therein) that the LVI problem (14) is equivalent to the piecewise linear equation, \( P_\Omega(y - (Wy + q)) - y = 0 \), where \( P_\Omega(\cdot) \) is a piecewise linear projection operator from space \( R^{3n+m+1} \) onto set \( \Omega \). The \( i \)th processing element of \( P_\Omega(y) \) is defined as \( \forall i \in \{1, 2, 3, \ldots, (3n + m), (3n + m + 1)\} \) with \( N := (3n + m + 1) \):

\[
\rho_i(y_i) := \begin{cases} y_i^- & \text{if } y_i < y_i^- \\ y_i & \text{if } y_i^- \leq y_i \leq y_i^+ \\ y_i^+ & \text{if } y_i > y_i^+ \end{cases}.
\]

Third, from our neural network design experience [2–4, 13, 14], it follows naturally that the LVI-based primal-dual neural network, being the real-time solver for LVI (14) as well as QP (10)–(13), can take the following dynamic equation [17, 24]:

\[
\dot{y} = y(I + W^T)(P_\Omega(y - (Wy + q)) - y),
\]
Figure 1. The LVI-based primal-dual neural network structure.

where design parameter $\gamma > 0$ is used to scale the network convergence. It is worth noting that $\gamma$, being the reciprocal of a capacitance parameter in the hardware implementation, should be set as large as the hardware permits (e.g., in analog circuits or VLSI [25]) or selected appropriately (e.g., between $10^3$ and $10^8$) for simulative and/or experimental purposes. In the ensuing simulations, for example, $\gamma$ is set to be $10^5$.

For a better understanding on the neural network structure, we show Fig. 1 and explain it as follows. Expressed in the $i$th neuron form (with $i = 1, 2, \ldots, N$), the above LVI-based primal-dual neural network (15) can be further written as:

$$y_i = \int \sum_{k=1}^{N} g_{ik} \left( \rho_k \left( \sum_{j=1}^{N} h_{kj} y_j - q_k \right) - y_k \right) \, dt,$$

where $h_{kj}$ denotes the $kj$th entry of scaling matrix $H := I - W$ and $g_{ik}$ denotes the $ik$th entry of combined scaling matrix $G := \gamma(I + W^T)$. The structure of LVI-based primal-dual neural network (15) is given in Fig. 1, which consists of three layers with each layer having $N$ neurons. In the first layer, the piecewise linear activation function $\rho_k(\cdot)$ (being the $k$th processing element of activation function array $\mathcal{P}_{\Omega}(y)$) is employed in the $k$th neuron and $-q_k$ is the bias for the $k$th neuron, $k = 1, 2, \ldots, N$. In addition, the piecewise linear activation function $\rho_k(\cdot)$ could be realized by using operational amplifiers known as limiters [13, 25]. The second layer is constructed by using linear neurons, which sum up correspondingly the first-layer neuron output and the negative feedback of the last-layer neuron output. In the third layer, as we can see from Fig. 1, integrating neurons are employed to complete the neural network structure and generate the network outputs $y_i, i =$.
When such a LVI-based primal-dual neural network (15) starts to work, the solution to QP (10)–(13) could be established online. The network outputs are then input to a configuration controller to control the manipulator. By using standard feedback control algorithms (such as a PID controller) [1, 2, 5], the manipulator should be able to track the desired configuration trajectories.

Furthermore, we could have the following theoretical results on global exponential convergence of the neural network (15).

**Theorem 3.** With the existence of at least one optimal solution $x^*$ to QP (10)–(13), starting from any initial state, the state vector $y(t)$ of the LVI-based primal-dual neural network (15) converges to an equilibrium point $y^*$, of which the first $n + 1$ elements constitute the optimal solution $x^*$. Moreover, if there exists a constant $\varrho > 0$ such that $\|y - P_2(y - (Wy + q))\|_2^2 \geq \varrho \|y - y^*\|_2^2$, then exponential convergence can be achieved for neural network (15) with the convergence rate proportional to $\gamma \varrho$.

**Proof.** Can be generalized from Refs [17, 24] and the references therein by using Lyapunov function candidate $\|y - y^*\|_2^2$ and projection-related inequalities.

### 4. Simulation Studies

The Unimation PUMA560 manipulator has 6 d.o.f. If we consider both the position and orientation of the end-effector, PUMA560 is not a redundant manipulator. However, if we consider only the positioning of the end-effector, the PUMA560 robot arm functionally becomes a redundant manipulator with the associated Jacobian matrix $J(\theta) \in \mathbb{R}^{3 \times 6}$. Moreover, it is worth pointing out that, in the ensuing simulations, the proposed scheme and PUMA560 manipulator are both simulated by using MATLAB 7.1 performed on a personal digital computer with a Pentium V 3.20 GHz CPU, 1 GB and Windows XP Professional SP2 operating system.

#### 4.1. Circular Path Following

In this subsection, the LVI-based primal-dual neural network (15) and bi-criteria velocity minimization scheme (5)–(8) are simulated for the kinematic control of the PUMA560 robot arm. The PUMA560 end-effector is expected to track a circular path with the radius being 10 cm and the revolute angle about the $X$-axis being $\pi/6$ rad. Design parameter $\gamma = 10^5$ is chosen. The task duration $T = 10$ s and initial joint vector $\theta(0) = [0, 0, 0, 0, 0, 0]^T$ rad. We have observed the following facts.

- Synthesized by the bi-criteria neural weighting scheme (with $\alpha = 0.3$), Fig. 2 illustrates the simulated motion trajectories of the PUMA560 manipulator in the three-dimensional workspace. The motion trajectory of the PUMA560 end-effector is very close to the desired circular one (with maximal positioning error less than $2 \times 10^{-5}$ m, as shown in Figs 3b and 4b).
Figure 2. The end-effector of the PUMA560 manipulator tracks a circle.

Figure 3. Profiles of joint velocity variables of the PUMA560 robot arm synthesized by the bi-criteria neural weighting scheme with different values of $\alpha$, where the end-effector tracks a circular path. (a) $\alpha = 0$ (i.e., INVM scheme). (b) $\alpha = 0.3$. (c) $\alpha = 0.6$. (d) $\alpha = 1$ (i.e., MVN scheme).
Figure 4. Positioning errors of the PUMA560 end-effector on the X, Y and Z-axes when tracking a circular path with different values of $\alpha$ (e.g., $\alpha = 0, 0.3, 0.6$ and 1). (a) $\alpha = 0$ (i.e., INVM scheme). (b) $\alpha = 0.3$. (c) $\alpha = 0.6$. (d) $\alpha = 1$ (i.e., MVN scheme).

- Figure 3a–3d illustrates the transient behaviors of joint velocities when $\alpha = 0$, 0.3, 0.6 and 1. As described in Section 2, when $\alpha = 0$, the bi-criteria weighting scheme reduces to the INVM scheme and, when $\alpha = 1$, the bi-criteria weighting scheme reduces to the MVN scheme. As shown in Fig. 3b and 3c, when $\alpha = 0.3$ and $\alpha = 0.6$, the bi-criteria weighting scheme generates solutions between the INVM and MVN schemes. Thus, in this sense, the proposed neural weighting scheme is very flexible.

- During circular path following, the discontinuity points of the pure INVM solution may occur. For example, in Fig. 3a, $\dot{\theta}_4$ has discontinuous points, which may not be accepted in practice. In contrast, if we use the bi-criteria neural weighting scheme with $\alpha > 0$, the discontinuity problem could be much remedied or even diminished. This is shown in Fig. 3b–3d. Moreover, as shown in Fig. 3, all joint variables have been kept within their mechanical limits.
Table 1.
Simulation time (s) for different α (circular case)

<table>
<thead>
<tr>
<th>α</th>
<th>0</th>
<th>0.3</th>
<th>0.6</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>201.266</td>
<td>30.251</td>
<td>28.6423</td>
<td>305.343</td>
</tr>
<tr>
<td>3</td>
<td>192.075</td>
<td>30.1074</td>
<td>28.7897</td>
<td>314.692</td>
</tr>
<tr>
<td>4</td>
<td>197.248</td>
<td>30.3101</td>
<td>28.5312</td>
<td>313.846</td>
</tr>
<tr>
<td>5</td>
<td>202.853</td>
<td>30.6268</td>
<td>28.8256</td>
<td>311.594</td>
</tr>
<tr>
<td>6</td>
<td>202.067</td>
<td>30.7662</td>
<td>29.1555</td>
<td>311.909</td>
</tr>
<tr>
<td>7</td>
<td>200.573</td>
<td>30.5022</td>
<td>28.9875</td>
<td>312.291</td>
</tr>
<tr>
<td>8</td>
<td>203.294</td>
<td>30.8619</td>
<td>29.002</td>
<td>312.521</td>
</tr>
<tr>
<td>9</td>
<td>205.471</td>
<td>32.7419</td>
<td>30.500</td>
<td>312.358</td>
</tr>
<tr>
<td>10</td>
<td>206.081</td>
<td>29.9807</td>
<td>28.6625</td>
<td>312.625</td>
</tr>
</tbody>
</table>

- Figure 4 shows the end-effector positioning error in Cartesian space. We can see from Fig. 4 that all the maximal Cartesian positioning errors synthesized by the proposed bi-criteria neural weighting scheme are very small (less than $2 \times 10^{-5}$ m).

- Moreover, because of using the LVI-based primal-dual neural network (which does not entail matrix inversion), we can perform the computer simulation at a very fast speed. For example, when the bi-criteria factor $\alpha = 0.3$, the computer simulation time is approximately between 29 and 33 s. When the factor $\alpha = 0.6$, the computer simulation time is between 28 and 31 s. In contrast, when using the INVM (or MVN) scheme alone, the simulation time is approximately 200 s (or 310 s). Table 1 compares the computer simulation times with respect to different $\alpha$, where we tested 10 times for each situation.

4.2. Straight-line Path Following

In this subsection, for a comparison, the end-effector task is to track a desired straight-line Cartesian path of length 1.125 m. The motion duration is required to be 10 s. Parameters $\kappa_p$ and $\gamma$ are, respectively, 20 and $10^5$. Computer simulation results are given in Figs 5–7 and Table 2.

- The three-dimensional motion trajectories of the PUMA560 manipulator synthesized by our bi-criteria minimization is illustrated in Fig. 5 (corresponding to $\alpha = 0.3$). The arrow in Fig. 5 shows the motion direction when the end-effector performs such a straight-line path. The motion trajectory of the PUMA560 end-effector is very close to the desired straight-line segment (with maximal positioning error less than $4 \times 10^{-5}$ m, as shown in Fig. 7b).

- Figure 6 shows the transient behavior of joint velocities when different values of $\alpha$ are used. From Fig. 6a–6d, we can see again that the bi-criteria weighting scheme could reduce to the INVM scheme (i.e., $\alpha = 0$) and the MVN scheme
Figure 5. End-effector of the PUMA560 manipulator tracks a straight-line segment.

Figure 6. Profiles of joint velocity variables of the PUMA560 robot arm synthesized by the bi-criteria neural weighting scheme with different values of $\alpha$, when its end-effector tracks a straight-line path.
(a) $\alpha = 0$ (i.e., INVM scheme). (b) $\alpha = 0.3$. (c) $\alpha = 0.6$. (d) $\alpha = 1$ (i.e., MVN scheme).
Figure 7. Positioning errors of the PUMA560 end-effector on the $X$, $Y$- and $Z$-axes while tracking a straight-line path when $\alpha = 0, 0.3, 0.6$ and 1, respectively. (a) $\alpha = 0$ (i.e., INVM scheme). (b) $\alpha = 0.3$. (c) $\alpha = 0.6$. (d) $\alpha = 1$ (i.e., MVN scheme).

Table 2.
Simulation time (s) for different $\alpha$ (straight-line case)

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0$</td>
<td>642.430</td>
<td>636.846</td>
<td>636.935</td>
<td>634.612</td>
<td>635.133</td>
<td>634.792</td>
<td>634.568</td>
<td>635.857</td>
<td>635.049</td>
<td>638.573</td>
</tr>
<tr>
<td>$\alpha = 0.6$</td>
<td>15.3070</td>
<td>15.0885</td>
<td>15.3631</td>
<td>15.1590</td>
<td>15.2474</td>
<td>14.9554</td>
<td>15.3646</td>
<td>15.1383</td>
<td>15.4310</td>
<td>15.2696</td>
</tr>
<tr>
<td>$\alpha = 1$</td>
<td>454.067</td>
<td>458.028</td>
<td>456.302</td>
<td>456.918</td>
<td>455.584</td>
<td>456.949</td>
<td>457.349</td>
<td>456.689</td>
<td>455.507</td>
<td>454.988</td>
</tr>
</tbody>
</table>
(i.e., $\alpha = 1$) if necessary. Thus, in this sense, the proposed neural weighting scheme is very flexible.

- Again, in terms of the discontinuity phenomenon, we can discuss the interesting observations as follows. During the straight-line path-following task, Fig. 6a shows the discontinuity deficiency which occurs in the pure infinity-norm solution (i.e., INVM with $\alpha = 0$). That is, $\dot{\theta}_2$ through $\dot{\theta}_5$ all have discontinuous points. However, by weighting with the minimum two-norm solution (with $\alpha > 0$), such a discontinuity problem is much remedied/diminished, as shown in Fig. 6b–6d. Similar to the circular path following example, this also demonstrates the uniqueness property and remedy effectiveness of the proposed bi-criteria inverse kinematics solution (when $\alpha > 0$). In addition, Fig. 6 shows that all joint variables have remained within their limited physical ranges.

- Figure 7 shows the end-effector Cartesian positioning errors. We can see from Fig. 7 that the maximal Cartesian positioning error obtained by our proposed bi-criteria neural weighting scheme is less than $7 \times 10^{-5}$ m.

- Similar to the circle path following example, we can perform the computer simulation very quickly because there is no matrix inversion entailed in our scheme. When the bi-criteria factor $\alpha = 0.3$ or 0.6, the computer simulation time is approximately between 14 and 16 s. In contrast, when using the pure INVM (or MVN) scheme, the simulation time is approximately 635 s (or 455 s). Table 2 compares the computer simulation time with respect to different values of $\alpha$, where we tested 10 times for each situation.

Hence, compared with single-criterion velocity minimization schemes, the bi-criteria neural weighting scheme is much more flexible, efficient and effective.

5. Conclusions

A bi-criteria neural optimization scheme is presented in this paper that could remedy effectively the discontinuity problem of the minimum infinity-norm solutions. The redundancy is resolved at the joint velocity level, and subject to general joint physical limits and joint velocity limits. A LVI-based primal-dual neural network approach is developed as well for the online redundancy resolution of robot manipulators. Such a LVI-based primal-dual neural network has a simple piecewise linear architecture and guaranteed fast convergence. Computer simulation results based on the PUMA560 robot arm have demonstrated the performance, characteristics and efficacy of the bi-criteria neural weighting scheme on robot manipulator redundancy resolution. Future work may lie in the application, hardware implementation and experiments of the proposed bi-criteria redundancy resolution scheme onto actual redundant manipulators (such as the physically structured PUMA560 and PA10 manipulators).
Acknowledgements

This work is funded by National Science Foundation of China under Grant 60643004 and by the Science and Technology Office of Sun Yat-Sen University. Before joining Sun Yat-Sen University in 2006, the corresponding author, Y. Z., had been with the National University of Ireland, University of Strathclyde, National University of Singapore and Chinese University of Hong Kong, since 1999. He has continued the line of this research, supported by those research fellowships and assistantships. His web page is now available at http://www.ee.sysu.edu.cn/teacher/detail.asp?sn=129.

References


**Appendix**

To show the relationship between bi-criteria neural weighting schemes (4) and (5), the computer simulation is performed additionally by using scheme (4) and by forcing the PUMA560 manipulator to track the same straight-line path as the one in Section 4.2. The design parameters are also the same as before. Figure A1 shows the profiles of joint velocity variables of the PUMA560 robot arm under different conditions.

![Figure A1](image-url)  
*Figure A1. Profiles of joint velocity variables of the PUMA560 robot arm synthesized by bi-criteria neural weighting schemes (4) and (5) when its end-effector tracks a straight-line path. (a) Scheme (4) with $\alpha = 0.6$. (b) Scheme (5) with $\alpha = 0.9$.***
schemes. The observation is that the velocity curves are similar when $\alpha = 0.6$ and $\alpha = 0.9$ are employed, respectively, in schemes (4) and (5).

About the Authors

Yunong Zhang is a Professor at the School of Information Science & Technology, Sun Yat-Sen University (SYSU), China. He received his BS, MS and PhD degrees, respectively, from Huazhong University of Science and Technology, South China University of Technology and Chinese University of Hong Kong, in 1996, 1999 and 2003. Before joining SYSU, in 2006, he had been with the National University of Ireland, University of Strathclyde and National University of Singapore, since 2003. His main research interests include robotics, neural networks and Gaussian processes. He is a member of the IEEE.

Binghuang Cai received the BS degree in Electronics Information Engineering, in 2004, and the MS degree in Signal and Information Processing, in 2007, both from Shantou University, China. He is currently a PhD student (major in Communication and Information Systems) at the School of Information Science & Technology, Sun Yat-Sen University, China. His current research interests include robotics, neural networks and intelligent information processing.

Lei Zhang is currently an Undergraduate Student (major in Software Engineering) at the Software School, Sun Yat-Sen University, China. His research interests include robotics, neural networks, software and programming.

Kene Li received the BS degree in Polymer Materials, in 2002, from South China University of Technology, China, and the MS degree in Signal and Information Processing, in 2007, from Guangdong University of Technology, China. He is currently pursuing the PhD degree at the School of Information Science & Technology, Sun Yat-Sen University, China. His research interests include robotics and neural networks.