A Neural Control Architecture for Joint-Drift-Free and Fault-Tolerant Redundant Robot Manipulators

NAN ZHONG¹, XUANZONG LI¹, ZIYI YAN², and ZHIJUN ZHANG², (Member, IEEE).

¹College of Engineering, South China Agricultural University, Guangzhou 510642, China.
²School of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, China.

Corresponding author: Zhijun Zhang (e-mail: drzhangzhijun@gmail.com).

This research was supported in part by the National Key Research and Development Plan 2017YFD070100103, the Key Science and Technology Plan of Guangdong Province 2017B010116003, the National Natural Science Foundation under Grants 61603142 and 61633010, the Guangdong Foundation for Distinguished Young Scholars under Grant 2017A030306009, the Guangdong Youth Talent Support Program of Scientific and Technological Innovation under Grant 2017TQ04X475, the Science and Technology Program of Guangzhou under Grant 201707010225, the Fundamental Research Funds for Central Universities under Grant xZdD2182410, the Scientific Research Starting Foundation of South China University of Technology, the National Key R and D Program of China under Grant 2017YFB1002505, the National Key Basic Research Program of China (973 Program) under Grant 2015CB351703, and the Guangdong Natural Science Foundation under Grant 2014A030312005.

ABSTRACT Fault tolerance is important for a redundant robot manipulator, which endows the robot with the capability of finishing the end-effector task even when one or some of joints’ motion fails. In this paper, a varying-parameter neural control architecture is designed to achieve fault tolerance for redundant robot manipulators. Specifically, a quadratic programming based fault-tolerant motion planning scheme is formulated. Secondly, a varying parameter recurrent neural network (VP-RNN) is proposed to resolve the standard quadratic programming problem, which can make the remaining healthy joints to remedy the whole system which is effected by faulty joints, and complete the expected end-effector path. Theoretical analysis based on Lyapunov stability theory proves that the proposed VP-RNN solver can globally converge to the optimal solution to the fault-tolerant motion planning scheme, and the joint motion failure problems are solved successfully. Computer simulations and physical experiments based on a six degrees-of-freedom (DOF) Kinova Jaco² robot, substantiate the effectiveness of the proposed varying-parameter neural control architecture for fault-tolerant motion planning scheme to redundant robot manipulators.

INDEX TERMS Neural networks, fault-tolerant, quadratic programming, redundant robot manipulators

I. INTRODUCTION

Redundant manipulator means that the degree of freedom of the manipulator is more than the necessary degree of freedom to complete the task. Actually, the redundant design idea is widely used in many fields. For instance, this idea has also been used in the area of network-based systems to improve the control/filtering performance as packet dropouts occur. Compared with the traditional non-redundant manipulator, the additional freedom can offer many advantages including obstacle avoidance [1], singularity avoidance [3], dexterity improvement [4], [5], plan selection [6], [7], fault tolerance [8], [9], etc. That means even when the redundant manipulators happened joint failure, redundance allows the manipulators use the extra freedom to make the manipulators continue to complete the task.

In recent decades, fault tolerant issue attracts more and more researchers and engineers in robot fields [14]–[16]. In fact, joint failures includes locked-joint and free-swinging joint failures. Many works on kinematic failure tolerance are based on pseudoinverse-type methods, by locking the failure joints to reduce degree of freedom so that the redundant manipulator can run as usual. Beside pseudoinverse-based method to solve the joint failures problem, other studies related to a robot’s failure tolerant including failure tolerance by trajectory planning, and layered failure tolerance control. However, the failure tolerance by trajectory planning needs complete trajectory information in advance, and the layered failure tolerance control just protects against application software, system software, hardware, and network failures. They are not suitable for real-time motion control which requires continuous trajectory correction based on sensory feedback.

In practical applications, joint motion failure problem
sometimes happens but we can’t repair immediately. For instance, long-term operations of robots in severe environment is prone to making the joint motion failure [12], [13]. This may greatly reduce the performance of the mechanical arms, which would even cause possible mechanical damage to the robot [2]. Therefore, it may be more feasible, timely and typical to complete the prescribed path tasks with the remaining healthy joints in fault failure research and engineering applications. In terms of fault tolerance research and engineering applications, the completion of prescribed tasks and maintaining healthy joint paths may be more practical and typical examples. When a joint failure occurs, if only few scheme parameters are adjusted, instead of reconsidering a new scheme, the prescribed path task can be completed well. Since traditional pseudoinverse-type methods can not solve the inequality problem and needs to compute the inverse of a matrix, quadratic programming (QP) based inverse kinematic schemes are preferred in recent years [21].

Generally, a QP problem is equivalent to a linear variational inequalities (LVI) system which can be solved by numerical methods or neural networks [22]–[24]. In recent years, neural networks is becoming more popular due to their distributed structure and generalization property. In order to solve strictly convex quadratic programming (SCQP) problems, a gradient-based neural network (GNN) was proposed and investigated in [25]. But GNN is not suitable for solving the time-varying problems, since it’s difficult to track the variables in the equations. In order to solve this kind of time-varying problems more effectively, a zeroing neural network (ZNN) with global convergence performance was proposed [26]. It shows that ZNN is more effective in many time-varying problems than GNN in solving time-varying matrix inversion [25], time-varying quadratic-minimization(QM) and quadratic-programming (QP) problems [27]. However, with the increasing of computation scale, a faster convergence speed in practice is needed. In order to meet this demand, different from the typical fixed-parameter recurrent neural network (FP-RNN) [27], a novel varying-parameter recurrent neural network (VP-RNN) is designed and investigated in this paper.

The remainder of this paper is divided into four sections. In Section II, the problem formulation and some preliminaries are discussed. In Section III, the VP-RNN is proposed to resolve the quadratic programming problem in fault-tolerant motion planning scheme. In Section IV, computer simulations are conducted. Section V gives two physical experiments. In section VI, the full paper is summarized. Specifically, the main contributions of this paper are listed as follows.

- A novel joint-drift-free and fault-tolerant scheme is proposed and developed. Different from the traditional single criterion methods, the proposed control architecture can achieve simultaneously joint-drift-free and fault-tolerant criteria, and formulated as a standard QP problem.
- A varying-parameter redundant neural network (VP-RNN) is proposed and investigated to solve the QP problem. To the best of the authors’ knowledge, it is the first time to propose such a VP-RNN for resolving a joint-drift-free and fault-tolerant scheme with feedback of redundant robot manipulators. Theoretical analysis based on Lyapunov theorem proves that the proposed VP-RNN can globally converge to the theoretical solutions to the time-varying QP.

- Computer simulations illustrate the effectiveness of the proposed varying-parameter neural control architecture for fault-tolerant redundant robot manipulators.
- Practical robot experiments substantiate the accuracy, stability, and physical realizability of the varying-parameter neural control architecture for solving joint motion failure problem of redundant robot manipulators.

II. PRELIMINARIES AND PROBLEM FORMULATION

In this section, preliminaries on redundant robots and a QP-based fault tolerant scheme are firstly presented. A novel VP-RNN is then proposed to solve the QP-based fault tolerant scheme, and the convergence of the proposed VP-RNN have been proved.

A. PRELIMINARIES

For a robot manipulator, the forward kinematic problem is considered as follow:

\[ r(t) = f(\theta(t)). \]  \hspace{1cm} (2.1)

where the end-effector position-and-orientation vector can be expressed as \( r(t) \in \mathbb{R}^m \), and the joint-space vector can be expressed as \( \theta(t) \in \mathbb{R}^n \). When the joint angle vector \( \theta(t) \) is given, we can easily get the robot arm's end-effector position \( r(t) \). However, given \( r(t) \), it is quite difficult to get \( \theta(t) \) directly. A common approach is to obtain the joint rotation velocity \( \dot{\theta}(t) \) by differentiating Equation (2.1) with respect to time \( t \), and \( \theta(t) \) can be obtained by integral operations. Mathematically, the relation between the end-effector velocity \( \dot{r} \) and the joint velocity \( \dot{\theta} \) is yielded as

\[ J(\theta)\dot{\theta} = \dot{r} \] \hspace{1cm} (2.2)

where \( J(\theta) = \partial f(\theta) / \partial \theta \in \mathbb{R}^{m \times n} \) is the Jacobian matrix, and \( \dot{r} \) and \( \dot{\theta} \) denote end-effector position velocity vector and joint angle velocity vector, respectively.

If the degrees-of-freedom (DOF) of the redundant manipulator is more than the DOF required to perform an end-effector primary task, i.e., \( m < n \), Equation (2.1) and (2.2) are under-determined and thus admit an infinite number of solutions. In this paper, we only consider the end-effector position and the robot has 6 DOF, and thus \( m = 3 \) and \( n = 6 \).

B. QP FORMULATION FOR FAULT TOLERANCE

According to Ref. [28], a joint-drift-free scheme can be expressed as

\[ \min \quad \dot{\theta}^T W \dot{\theta} / 2 + q^T \dot{\theta} \] \hspace{1cm} (2.3)

\[ \text{s.t.} \quad J(\theta) \dot{\theta} = \dot{r} + K(r - f(\theta)) \] \hspace{1cm} (2.4)
where $W := \|\theta(t) - \theta(0)\|_2^2 I$ (with $\| \cdot \|_2$ denoting the two-norm of a vector, and $I$ denoting the identity matrix), $q := k(\theta(t) - \theta(0))$ (with $k > 0$ used to scale the convergence rate of $(\theta(t) - \theta(0))$, and $K \in \mathbb{R}^{m \times m}$ is a feedback-gain matrix, which is positive-definite and symmetric.

In practical applications, joint motion failure problem sometimes happens, it may affect the performance of the manipulator. What is worse, the manipulator would fail to complete the desired end-effector task. Therefore, it is necessary to add fault tolerance criterion to the control scheme. For instance, if $m_a$ joints (the $i$th, $\ldots$, $j$th joints) fail, the corresponding joint-velocity values would be zero, i.e.,

$$\begin{align*}
0\dot{\theta}_1 + \ldots + 1\dot{\theta}_i + \ldots + 0\dot{\theta}_{n-1} + 0\dot{\theta}_n &= 0 \\
0\dot{\theta}_1 + 0\dot{\theta}_2 + \ldots + 1\dot{\theta}_j + \ldots + 0\dot{\theta}_n &= 0
\end{align*}
$$

The above equations can be reformulated as a matrix equation, i.e.,

$$A\dot{\theta} = 0 \quad (2.5)$$

with $A \in \mathbb{R}^{m \times n}$ being defined as

$$A = \begin{bmatrix}
0 & \ldots & 1 & \ldots & 0 & \ldots & 0 \\
0 & \ldots & 0 & \ldots & 1 & \ldots & 0
\end{bmatrix}.$$

Evidently, if Equation (2.5) is integrated into the traditional joint-drift-free scheme (2.3)-(2.4) as one of equality constraints, failure joints $\dot{\theta}_1, \ldots, \dot{\theta}_j$ would be forced into zero during the task execution [16]. That is to say, fault-tolerance would be achieved. A joint-drift-free and fault-tolerant scheme is formulated as

$$\begin{align*}
\min & \quad \dot{\theta}^T W \dot{\theta} / 2 + q^T \dot{\theta} \\
\text{s.t.} & \quad J(\theta)\dot{\theta} = b \\
& \quad A\dot{\theta} = 0
\end{align*} \quad (2.6)$$

The above equations (2.6)-(2.8) can be further rewritten as the following compact matrix equation formulation, i.e.,

$$\begin{align*}
\min & \quad \dot{\theta}^T W \dot{\theta} / 2 + q^T \dot{\theta} \\
\text{s.t.} & \quad C\dot{\theta} = d
\end{align*} \quad (2.9)$$

where coefficient matrix $C$ is defined as $C = [J; A]^T \in \mathbb{R}^{(m + m_a) \times n}$ and vector $d$ is defined as $d = [b; 0]^T \in \mathbb{R}^{m + m_a}$.

### III. VP-CDNN SOLVER

Lagrange multiplier method is a classical analytical method to solve conditional extremum, which can transform all constrained optimization model problems into unconstrained extremum problems [29]. For multidimensional optimization problems with one equality constraints (2.10), the original objective function is transformed into a new objective function in the following form

$$L(\dot{\theta}(t), \lambda(t), t) = \dot{\theta}^T(t) W(t) \dot{\theta}(t) / 2 + q^T(t) \dot{\theta}(t) + \lambda^T(t)(C(t) \dot{\theta}(t) - d(t)). \quad (3.1)$$

The undetermined coefficient $\lambda \in \mathbb{R}^m$ is called Lagrange multiplier. The equation (3.1) is shown by partial derivative method and let them be equal to 0, listed below

$$\begin{align*}
\frac{\partial L(\dot{\theta}(t), \lambda(t), t)}{\partial \dot{\theta}(t)} &= W(t) \dot{\theta}(t) + q(t) + C^T(t) \lambda(t) \\
\frac{\partial L(\dot{\theta}(t), \lambda(t), t)}{\partial \lambda(t)} &= C(t) \dot{\theta}(t) - d(t) = 0
\end{align*} \quad (3.2)$$

Equations (3.2) is then rewritten as the following matrix equation, i.e.,

$$B(t) \gamma(t) = g(t). \quad (3.3)$$

where

$$B(t) := \begin{bmatrix}
W(t) & C^T(t) \\
C(t) & 0_{m \times m}
\end{bmatrix} \in \mathbb{R}^{(m + n + m_a) \times (m + n + m_a)},$$

$$y(t) := \begin{bmatrix}
\dot{\theta}(t) \\
\lambda(t)
\end{bmatrix} \in \mathbb{R}^{n + m + m_a}, g(t) := \begin{bmatrix}
q(t) \\
d(t)
\end{bmatrix} \in \mathbb{R}^{n + m + m_a}.$$

Through the transformation above, solving QP (2.9)-(2.10) is transformed to finding the solution to matrix Equation (3.3). According to neural dynamics design method [30]-[33], a varying-parameter recurrent neural network can be obtained through the following three steps.

Firstly, a vector-type error can be defined as

$$\varepsilon(t) = B(t)y(t) - g(t) \in \mathbb{R}^{n + m + m_a}. \quad (3.4)$$

In order to make the formula (3.4) converge to zero, we can design the following negative time derivative of error $\varepsilon(t)$, i.e.,

$$\frac{d\varepsilon(t)}{dt} = -(\gamma + \tau^e) \Phi(\varepsilon(t)), \forall t \geq 0, \quad (3.5)$$

where $\gamma + \tau^e$ is designed to adjust the rate of convergence; array $\Phi(\cdot)$ is an activation function column vector, which is composed of $(n + m)$ scalar-valued processing-unit $\phi(\cdot)$, and each $\phi(\cdot)$ should be a monotonically-increasing odd activation function. For instance, in this paper, a power-sigmoid activation function is used, i.e.,

$$\phi(\varepsilon_i(t)) = \begin{cases}
1 + \exp(-\sigma) \left[ 1 - \exp( - \sigma \varepsilon_i(t)) \right], & \text{if } 0 \leq \varepsilon_i(t) < 1, \\
1 - \exp(-\sigma) \left[ 1 + \exp( - \sigma \varepsilon_i(t)) \right], & \text{if } \varepsilon_i(t) \geq 1,
\end{cases} \quad (3.6)$$

where $\varepsilon_i(t)$ denotes the $i$-th element of error vector $\varepsilon(t)$.

Thirdly, substituting Equation (3.4) into Equation (3.5), the following implicit-dynamic equation is obtained, i.e.,

$$B(t) \dot{y}(t) = -B(t)y(t) - (\gamma + \tau^e) \Phi(B(t)y(t) - g(t)) + \dot{g}(t). \quad (3.6)$$

Because $(\gamma + \tau^e)$ is time-varying and implicit-dynamic equation (3.6) is base on differential equation theory, the above equation is termed varying-parameter recurrent neural network (VP-RNN).
Theorem 1: Consider the time-varying quadratic programming (2.9) and (2.10) are smoothly and strictly convex. If a monotonically-increasing odd activation-function array \( \phi(\cdot) \) is applied, then state variable \( y(t) = [\tilde{\theta}^T(t), \lambda^T(t)]^T \) of Equation (3.6), starting from any initial state \( y(0) \in R^{m+n+m+n}_{>0} \), can globally converges to the unique theoretical solution \( y^*(t) = [\tilde{\theta}^*^T(t), \lambda^*^T(t)]^T \), i.e., \( \lim_{t \to +\infty} [y(t) - y^*(t)] = 0 \). In addition, the first \( n \) elements of \( y^*(t) \) is the solution \( \tilde{\theta}^*(t) \) to the time-varying quadratic programming (2.9) and (2.10).

Proof: A Lyapunov function candidate is defined as
\[
v(t) = \|\varepsilon(t)\|_2^2/2 = \varepsilon^T(t)\varepsilon(t)/2 \geq 0,
\]
where \( \varepsilon(t) \) denotes error vector \( \varepsilon(t) = B(t)y(t) - g(t) \) and \( \| \cdot \|_2 \) denotes two norm of a vector.

Then the time derivative of Lyapunov function candidate \( v(t) \) is
\[
\dot{v}(t) = \frac{dv(t)}{dt} = \varepsilon^T(t)\frac{d\varepsilon(t)}{dt}.
\]
Substituting Equation (3.5) into Equation (3.8), we can get
\[
\dot{v}(t) = -(\gamma + \tau^T)\varepsilon^T(t)\Phi(\varepsilon(t))
= -(\gamma + \tau^T) \sum_{i=1}^{n+m+m+n} \varepsilon_i(t)\phi(\varepsilon_i(t)),
\]
where \( \varepsilon_i(t) \) denotes the \( i \)th element of error vector \( \varepsilon(t) \), and \( \phi(\varepsilon_i(t)) \) denotes the \( i \)th activation function element of vector \( \Phi(\varepsilon_i(t)) \).

Because \( \phi(\varepsilon_i(t)) \) is a monotonically-increasing odd activation function, we can get the following relationship, i.e.,
\[
\varepsilon_i(t)\phi(\varepsilon_i(t)) \begin{cases} > 0, & \text{if } \varepsilon_i(t) > 0 \text{ or } \varepsilon_i(t) < 0, \\ = 0, & \text{if } \varepsilon_i(t) = 0. \end{cases}
\]

The aforementioned relationship (3.10) guarantees that \( \dot{v}(t) \) is non-positive for all \( t \in [0, +\infty) \). According to Lyapunov theory, \( y(t) - y^*(t) \) globally converges to 0, i.e., \( x(t) - x^*(t) \) globally converges to 0. The proof is completed.

IV. COMPUTER SIMULATIONS

In this section, the proposed joint-drift-free and fault-tolerant scheme (2.6)-(2.8) is conducted on the robot model of a Kinova Jaco2, which has 6 DOF and works in three-dimensional space. For verification, two end-effector tasks are given, i.e., tracking a tower-path and a Eastern word Jia (means the first class in Chinese). In the computer simulations, the initial joint state is set as \( \theta(0) = [1.675; 2.843; -3.216; 4.187; -1.71; -2.65] \) (rad). Convergence parameter \( \gamma \) in VP-RNN solver is set as 8. The parameters of the activation function are \( \sigma = 4 \) and \( n = 3 \). The feedback-gain matrix \( K \) is set as \( 8J \) where \( J \) denotes an identity matrix. The task execution period \( T = 5 \) seconds.

A. TOWER-PATH TRACKING

Four tests are conducted in this subsection, i.e., single faulty joint without fault-tolerant (FT) scheme, single joint with fault-tolerant scheme, two joints with fault-tolerant scheme, and three joints with fault-tolerant scheme.

1) Fault Tolerance of Single Joint
First of all, Fig.1(a) shows the expected tower-path and the trajectory with single faulty joint generated by the FP-RNN solver. In the figure, the desired path and the end-effector trajectory of single faulty joint without FT scheme are denoted respectively by the (black) dashed line and the (pink) diamond. It can be easily judged that the actual trajectory generated by the no FT scheme deviates from the desired path when the 4th joint is faulty. The corresponding positioning errors, joint-angle transients and joint-velocity transients are shown in Figs. 2(a)-(c), respectively. Fig. 2(a) shows that the system exists a a great number of position errors which are unacceptable. Fig.2(b) shows that the 4th joint do not move and is fixed, and the corresponding joint-velocity transients was zero in Fig. 2(c).

Second, a single-joint fault-tolerant scheme (2.6)-(2.8) is conducted on the redundant robot manipulators, and the 4th joint is set faulty. To achieve fault-tolerance, the coefficient matrix \( A \) can be set as \( [0,0,0,1,0,0] \), and integrated into the fault-tolerant scheme, and the corresponding simulation results are illustrated in Fig. 2(d)-(f). As can be seen from Fig. 2(e) and (f), although the 4th joint is fixed, the end-effector can still complete the tower-path tracking task well with the maximal positioning error being less than \( 8 \times 10^{-8} \) m as shown in Fig. 2(d). The simulation results related to the faulty joint situations also show that the healthy joints can compensate the loss of the faulty joints to the end-effector to maintain the prescribed end-effector path-tracking. Table 1 shows the joint-drifts controlled by different schemes. From the third column of Table 1, we see that the maximum
joint-angular-drift is $-2.983 \times 10^{-9}$ rad, and the two norm $\|\theta(5) - \theta(0)\|_2$ is just $2.223 \times 10^{-9}$ rad. Compared with the second column of Table 1 and Table 2, it is obvious that fault-tolerant performance is better than that with no fault-tolerant scheme.
TABLE 1. Comparisons of Joint-Angular-Drifts (Rad) between the Fault-Tolerant Scheme Synthesized by VP-RNN and without Considering Faulty Joint Synthesized by FP-RNN when the End-Effecter Completes The Tower Path

<table>
<thead>
<tr>
<th>Joint-drift(rad)</th>
<th>Single joint without fault-tolerant scheme</th>
<th>Single joint with fault-tolerant scheme</th>
<th>Two joints with fault-tolerant scheme</th>
<th>Three faulty joints with fault-tolerant scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_1(5) - \theta_1(0)$</td>
<td>$-2.245 \times 10^{-7}$</td>
<td>$1.208 \times 10^{-6}$</td>
<td>$8.252 \times 10^{-7}$</td>
<td>$-3.642 \times 10^{-10}$</td>
</tr>
<tr>
<td>$\theta_2(5) - \theta_2(0)$</td>
<td>$1.135 \times 10^{-5}$</td>
<td>$-2.602 \times 10^{-11}$</td>
<td>$3.404 \times 10^{-10}$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\theta_4(5) - \theta_4(0)$</td>
<td>$2.724 \times 10^{-5}$</td>
<td>$-2.983 \times 10^{-9}$</td>
<td>$0$</td>
<td>$-2.508 \times 10^{-10}$</td>
</tr>
<tr>
<td>$\theta_5(5) - \theta_5(0)$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\theta_6(5) - \theta_6(0)$</td>
<td>$2.487 \times 10^{-5}$</td>
<td>$-1.153 \times 10^{-9}$</td>
<td>$2.080 \times 10^{-9}$</td>
<td>$1.21 \times 10^{-10}$</td>
</tr>
<tr>
<td>$</td>
<td></td>
<td></td>
<td>\theta(5) - \theta(0)</td>
<td></td>
</tr>
<tr>
<td>$2.317 \times 10^{-9}$</td>
<td></td>
<td>$2.218 \times 10^{-9}$</td>
<td>$0$</td>
<td>$4.562 \times 10^{-10}$</td>
</tr>
</tbody>
</table>

2) Fault tolerance of two faulty joints

In order to verify the fault tolerance of two faulty joints, the robot manipulator is applied to complete the tower-path tracking task. In the simulation, all the parameters of the fault-tolerant scheme and neural networks are set the same, i.e., $k=4$ and design parameter $\gamma=8$. Without loss of generality, we assumed that the 3th, 4th joints are faulty, i.e., matrix $A$ is set as $[0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1]$. The simulation results are shown in Fig. 2(g)-(i). Even the 3th, 4th joint are locked (As is shown in Fig. 2(h) and (i)), the maximal position error is less than $1.5 \times 10^{-7}$, which is show in Fig.2(g). The joint-angular-drifts are listed in the forth column of the Table 1, which shows that the joint-angular-drifts are small. Specifically, when we locked the 3th, 4th joints, the maximum joint-angular-drift is $2.080 \times 10^{-9}$ rad, and $|||\theta(5) - \theta(0)||_2$ synthesized by VP-RNN is just $2.218 \times 10^{-9}$ rad. Evidently, with the VP-RNN, the joint-angular-drift is tiny, even comparable to the case without locked-joints. In addition, from Table 1, the integral absolute values of errors (IAE) of two faulty joints is similar with no faulty joint. These results further verify the effectiveness and accuracy of the proposed fault-tolerant scheme (2.6)-(2.8) dealing with two joint failures.

3) Verification the Fault Tolerance with Three Faulty Joints

In order to further test the effectiveness of the proposed VP-RNN for solving multiple-joints failure problem, three faulty joints are considered in the fault tolerant scheme. The robot manipulator is simulated to complete the tower-path tracking task. Matrix $A$ is set as $[0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1]$, and other parameters are set the same values. The corresponding simulation results are illustrated in Fig. 2(j). From Fig. 2(k), (i), we see that the 2nd, the 4th, and the 6th joints are fixed, and the tiny position errors are shown in Fig. 2(j), which are less than $8 \times 10^{-8}$m in each dimension can illustrates the tower-path tracking task can be completed well. The more specific joint-angular-drifts of the final and initial states planned by fault-tolerant scheme synthesized by the VP-RNN are listed in Table 1. It can be seen from the fourth column of Table 1 that the norm of joint-drifts $|||\theta(5) - \theta(0)||_2$ is just $4.562 \times 10^{-10}$ rad, which is much tinier than that synthesized by FP-RNN. For comparative purposes, the integral absolute values of errors (IAE) of four situations are given in Table 2. The first situation is with no faulty joint corresponding to second column. The second situation is with the fault of the 4th joint considered corresponding to third column. The third situation is with the faults of the 3th, 4th joints considered corresponding to forth column. The last situation is with the faults of the 2nd, the 4th, and the 6th joints considered corresponding to fifth column. It shows that IAE in each dimension synthesized by fault tolerant scheme are smaller than the IAE planned by FP-RNN without faulty joint.

In summary, this tower-path trajectory simulation illustrates the validity, safety, accuracy and practicability of the proposed VP-RNN for solving joint locked failure scheme, and can solved joint locked problem.
FIGURE 4. Simulation results of the robot end-tracking a jia-path as synthesized by fault-tolerant scheme (7)-(9) with $k = 4$ synthesized by VP-RNN or FP-RNN when the robot manipulator tracks a jia-path. (a) Position error synthesized by FP-RNN without fault-tolerant scheme. (b)-(d) Position errors synthesized by VP-RNN with fault-tolerant scheme (7)-(9) with $k = 4$. (e)-(h) $\theta$-Profile. (i)-(l) $\dot{\theta}$-Profile.

B. JIA-PATH TRACKING

In order to prove the wide applicability of this method, we design another track for experiments in this section. All the parameters of the manipulator and neural networks are set the same as those in the tower-path tracking example.

Firstly, the scheme (2.9)-(2.10) without fault-tolerant considered and with single faulty joint with fault-tolerant considered synthesized by FP-RNN and VP-RNN, respectively
are performed on the robot manipulator for tracking a jia-path. The simulation results are shown in Fig. 3 and Table III and IV. As can be seen from Fig. 3(a) and (b), the end-effector trajectory with fault-tolerant scheme coincides well with the desired path, and it performs much better than the case without fault-tolerant scheme. However, the errors of the scheme without considering fault-tolerant synthesized by FP-RNN are large which is shown in Fig. 4(a). Contrastively, Fig. 4(d) illustrates the result of the scheme with fault-tolerant considered synthesized by VP-RNN. It shows that the maximal error is just less than $6 \times 10^{-7}$ m. Which is very tiny and can be ignored in practical applications. Table IV lists the tiny joint-angular-drifts of the final and initial states synthesized by VP-RNN for solving fault-tolerant scheme. This example demonstrates the effectiveness of the proposed scheme.

Secondly, in order to further illustrate the advantages of the proposed fault-tolerant scheme synthesized by VP-RNN, the robot manipulator is simulated to complete the jia-path tracking task with two joints locked. As is seen from Fig. 4(d)-(f), the angles of the 3th, 4th joints remain unchanged from the beginning to the end. In addition, the maximal error $\epsilon(y)_{\max} = 1 \times 10^{-7}$, which is small and acceptable in practice. From the forth column of Table III, we can see that the joint-angular-drifts are small too.

Thirdly, for further studies on multiple joint failures, the robot manipulator is simulated to complete the jia-path tracking task with three joints locked. That is, the 2nd, 4th, 6th joint is faulty from the beginning to finish that we can see from Fig. 4(j)-(l). From Fig. 4(j), the position errors of the end-effector planned by fault-tolerant scheme (2.9)-(2.10) are still very small. The corresponding numerical computation results are listed in Table IV, i.e., the IAE planned by fault-tolerant scheme by VP-RNN(3rd to 5th column of Table IV) are much smaller than those without faulty joint by FP-RNN(2nd column of Table IV).

In summary, the above two trajectory simulation illustrate the effectiveness, safety, accuracy and availability of the proposed VP-RNN for solving joint locked failure scheme, which can remedy the joint locked failure very well.

### V. PHYSICAL EXPERIMENT VERIFICATION

In this section, a fault-tolerant experiment of tracking a tower-path and jia-path is implemented on the six-DOF Kinova Jaco$^2$ robot shown in Fig. 5. For illustration, the initial joint state is set as $\theta(0) = [1.675;2.843;3.216;4.187;1.71;2.65]$ (rad) for physical experiment (all the parameter is same as the simulations). In addition, the 3st joint is set to be faulty all the time, the fault-resultant configuration matrix $A$ is set as $[0,0,1,0,0,0]$. The corresponding physical experiment results are shown in Figs. 6 and 7, respectively.

Specifically speaking, Figs. 6 and 7 show the actual task executions of the Kinova Jaco$^2$ robot manipulator synthesized by the fault-tolerant scheme (2.9)-(2.10) with the 6th joint assumed be faulty. In Figs. 6 and 7, the snapshots at different time instants of the task execution show that the Kinova Jaco$^2$ robot manipulator completes the tower-path and jia-path tasks smoothly and stably.

These two physical experiments verify the effectiveness, accuracy and physical reliability of the proposed fault-tolerant redundancy-resolution scheme (2.9)-(2.10) on the motion planning and control of redundant robots.

### VI. CONCLUSION

In order to solve the locked faulty joint problem, a varying-parameter neural control architecture is designed to achieve fault tolerance for redundant robot manipulators. By redefining the fault-tolerant scheme as a QP problem, the manipulator can run normally. In addition, a varying-parameter redundant neural network (VP-RNN) has been presented and used to solve the redefinition QP problem. Compared with the traditional fixed-parameter redundant neural net-

---

**Table 3.** Comparisons of joint-angular-drifts (rad) between the fault-tolerant scheme by VP-RNN and without faulty joint by FP-RNN when end-effector completing the jia-path

<table>
<thead>
<tr>
<th>Joint-drift(rad)</th>
<th>Single faulty joint without fault-tolerant scheme</th>
<th>Single faulty joint with fault-tolerant scheme</th>
<th>Two faulty joints with fault-tolerant scheme</th>
<th>Three faulty joints with fault-tolerant scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_1(5) - \theta_1(0)$</td>
<td>$2.361 \times 10^{-4}$</td>
<td>$5.737 \times 10^{-4}$</td>
<td>$-1.074 \times 10^{-8}$</td>
<td>$1.639 \times 10^{-7}$</td>
</tr>
<tr>
<td>$\theta_2(5) - \theta_2(0)$</td>
<td>$1.021 \times 10^{-4}$</td>
<td>$-2.292 \times 10^{-10}$</td>
<td>$-2.022 \times 10^{-8}$</td>
<td>$3.140 \times 10^{-9}$</td>
</tr>
<tr>
<td>$\theta_3(5) - \theta_3(0)$</td>
<td>$3.632 \times 10^{-4}$</td>
<td>$1.495 \times 10^{-8}$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\theta_4(5) - \theta_4(0)$</td>
<td>$0$</td>
<td>$0$</td>
<td>$-5.239 \times 10^{-8}$</td>
<td>$3.905 \times 10^{-9}$</td>
</tr>
<tr>
<td>$\theta_5(5) - \theta_5(0)$</td>
<td>$1.465 \times 10^{-4}$</td>
<td>$2.431 \times 10^{-9}$</td>
<td>$7.165 \times 10^{-8}$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\theta(5) - \theta(0)$</td>
<td>$1.174 \times 10^{-4}$</td>
<td>$-2.707 \times 10^{-9}$</td>
<td>$9.167 \times 10^{-8}$</td>
<td>$5.118 \times 10^{-9}$</td>
</tr>
<tr>
<td>$</td>
<td>\theta(5) - \theta(0)</td>
<td></td>
<td>_2$</td>
<td>$4.830 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

**Table 4.** Integral absolute value of errors (iae) by the fault-tolerant scheme by VP-RNN and without faulty joint by FP-RNN when end-effector completing the jia-path

<table>
<thead>
<tr>
<th>IAE</th>
<th>Single joint without fault-tolerant scheme</th>
<th>Single joint with fault-tolerant scheme</th>
<th>Two joints with fault-tolerant scheme</th>
<th>Three joints with fault-tolerant scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\int_0^t</td>
<td>\epsilon_1</td>
<td>dt (m s)$</td>
<td>$1.299 \times 10^{-2}$</td>
<td>$1.478 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\int_0^t</td>
<td>\epsilon_2</td>
<td>dt (m s)$</td>
<td>$2.253 \times 10^{-2}$</td>
<td>$1.480 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\int_0^t</td>
<td>\epsilon_3</td>
<td>dt (m s)$</td>
<td>$0.779 \times 10^{-2}$</td>
<td>$6.656 \times 10^{-9}$</td>
</tr>
<tr>
<td>$\int_0^t</td>
<td>\epsilon_4</td>
<td>dt (m s)$</td>
<td>$1.299 \times 10^{-2}$</td>
<td>$1.478 \times 10^{-6}$</td>
</tr>
</tbody>
</table>
work (FP-RNN), VP-RNN solver is much better than FP-RNN solver for fault-tolerant scheme. Both simulates and experiments are based on the six-DOF Kinova Jaco robot in the situations with faulty joints. Comparative simulations and physical experiments have substantiated that the proposed fault-tolerant scheme was more feasibility, more accurate, safer and more efficient compared with no fault-tolerant scheme when solving the joint locked problems to redundant robot manipulators.

REFERENCES


Y. Zhang and Z. Zhang, Repetitive Motion Planning and Control of Redundant Robot Manipulators. Springer Berlin Heidelberg, 2013.


Nan Zhong received the Ph.D. degree in Agricultural Mechanization Engineering from South China Agricultural University, China, in 2006. She was a visiting professor at the Institute of Applied Mathematics in the University of British Columbia, Canada, in 2009–2010, and a Senior researcher at Institute of Agricultural Engineering in University of Hohenheim, Germany, in 2015-2016. Since 2006, she has been a professor at the College of Engineering in South China Agricultural University, Guangzhou, China. Her current research interests include robotics, neural networks, Pattern recognition and intelligent systems.

Zhijun Zhang (M’12) received the Ph.D. degree in communication and information system from Sun Yat-sen University, Guangzhou, China, in 2012. He was a Post-Doctoral Research Fellow with the Institute for Media Innovation in Nanyang Technological University, Singapore in 2013-2015. Since 2015, he has been an Associate Professor with the School of Automation Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include robotics, neural networks, and human-robot interaction.

Xuanzong Li received the B.S. degree in software engineering from Guangzhou College of Commerce, Guangzhou, China, in 2017. He has been working toward the M.S. degree at the College of Engineering, South China Agricultural University, Guangzhou, China since 2017. His current research interests include robotics, machine vision, neural networks, and deep learning.

Ziyi Yan received the B.S. degree in automation from Central South University, Changsha, China, in 2016. He has been working toward the M.S. degree at the School of Automation Science and Engineering, South China University of Technology, Guangzhou, China since 2016. His current research interests include robotics, neural networks, and machine learning.