2012 12th IEEE-RAS International Conference on Humanoid Robots
Nov.29-Dec.1, 2012. Business Innovation Center Osaka, Japan

Adaptive Biarticular Muscle Force Control for Humanoid Robot Arms

Haiwei Dong and Nikolaos Mavridis
Department of Computer Engineering, New York University Abu Dhabi
Abu Dhabi, United Arab Emirates
{haiwei.dong; nikolaos.mavridis}@nyu.edu

Abstract—**This paper presents an efficient method for adaptive control of humanoid robot arms with biarticular muscles, which exhibits multiple beneficial properties. In our approach, sliding control was chosen to get joint torque first and then the joint torque was distributed to muscle forces. The muscle force was computed based on a Jacobian matrix between joint torque space and muscle force space. In addition, internal forces were used to optimize the computed muscle forces – and thus, two important benefits arise: through our proposed method, not only we are making sure that each muscle force stays within its predefined force boundary; but we are also enabling the muscles to work in the middle of their working range, which is considered to be an anti-fatigue state. Yet more benefit derived, comes from the fact that in our method all the dynamic parameters are updated in real-time, and can thus one can account for perturbations and disturbances during operation. Compared with previous work in parameter adaptation, a composite method was proposed which utilizes the prediction error to accelerate parameter convergence speed. Our method was tested for the case of reaching motions. The results clearly illustrate the benefits of the method.**

Keywords-muscle cooperation; redundancy; internal force

I. INTRODUCTION

Traditionally robot arms are driven by motors, and usually, each motor drives a specific joint, and corresponds with an independent degree of freedom (DOF). Such a driven system has two main problems: First, if a motor is broken, the DOF corresponding to this motor will be completely lost. Second, the motors near the base link need to carry more load and thus need more power, leading it to the need for very heavy base motors. In contrast, when one compares the above state of affairs of traditional robot arms with the human movement system, one can see that there exist many advantages for the latter: First, as there are many muscles driving one joint, the total joint torque is distributed to every muscle. Thus, each muscle only has a relatively small load. Second, if one muscle is broken, the motion capability of the corresponding joint is not totally lost. As a result of the above observations, mimicking such aspects of the human body seems to be a promising research direction. Recently, there has been quite some work focusing on bionic arms [1-4]. In our paper, we propose a muscle control method for humanoid robot arms driven by biariticular muscles, which can also be easily extended to more complicated configurations. Our control method is designed to be adaptive, i.e. robust to the perturbation and disturbances from environment. In addition, the muscle forces that are produced have to satisfy preset boundary force limits. Finally, the method has to be efficient enough for practical application.

Towards our endeavor, we had to face two main issues, namely Redundancy and Adaptivity. Let us start we the first: There exist two types of redundancies. Type I redundancy is between end effector position and joints. The number of positional degrees of freedom (DOF) of the end effector is 3 (not taking pose into account), i.e., the end effector can move in 3D space. Nevertheless, the number of joints in the arm are always more than 3, indicating the fact that for the same end effector position, there exist many limb joint configuration possibilities. On the other hand, Type II redundancy, is between joint torque and muscle force. There is a greater number of muscles than that required to generate movement.

Regarding previous work, the Type I redundancy problem has been studied for more than fifty years. Considering different optimization criteria or restricted conditions, many methods have been proposed. Yoshikawa proposed a manipulatability measure, through minimization of which, the arm is kept away from singularities [5]. Maciejewski et al. used null-space vectors to aid obstacle avoidance [6]. On the other hand, for the Type II redundancy problem, there exists literature on building bionic robots that attempt to follow the movement principles of human. For example, Klug et al. developed a 3 DOF bionic robot arm which is controlled by a PD controller with feed-forward compensation [2]. The trajectory of the arm is optimized and adjusted for a time and energy-optimal motion [7]. Potkonjak et al. built a humanoid robot with antagonistic drives whose controller is designed by H_∞ loop shaping [3]. Tahara proposed a simple sensor-motor control scheme as internal force and simulated the overall stability [4].

Now, having discussed the issue of Redundancy, let us move on to the second issue: Adaptivity. Online parameter adaptivity has also been researched for a while in the robotics field. Many methods have been proposed, such as robust control, adaptive feedback control, neurofuzzy adaptive control, etc. However, these approaches have not

been applied towards the control of humanoid robot arms with biarticular muscles.

In this paper, we have chosen to utilize sliding control, in order to first derive joint torques, and then to distribute the joint torques to muscle forces. Specifically, the muscle forces were computed by utilizing a Jacobian matrix between joint torque space and muscle force space. Furthermore, we used internal forces to optimize the computed muscle forces. The optimization not only allowed us to keep the muscle forces within their predefined force boundary; but also enables the individual muscles to be mainly operating in the middle of their working range, which we consider to be an anti-fatigue state. Besides, all the dynamic parameters of the model are updated online. In comparison with previous work on parameter adaptation, we propose a method that uses prediction error to accelerate the convergence speed of parameters. The proposed method was tested for reaching movements. The results illustrate the benefits and effectiveness of the method.

II. METHOD

A. Arm Model

We built a 2-dimensional model of the arm in the horizontal plane (no gravity) based on the upper limb of a digital human. The model includes six muscles (shown as 1 to 6) and two degrees of freedom (shoulder flexion-extension and elbow flexion-extension). The range of the shoulder angle is from -20 to 100 degrees, and the range of the elbow is from 0 to 170 degrees. Four of the muscles are mono-articular, and two are bi-articular where 1 and 2 cross the shoulder joint; 3 and 4 cross the elbow joint; 5 and 6 cross both joints (Fig. 1).

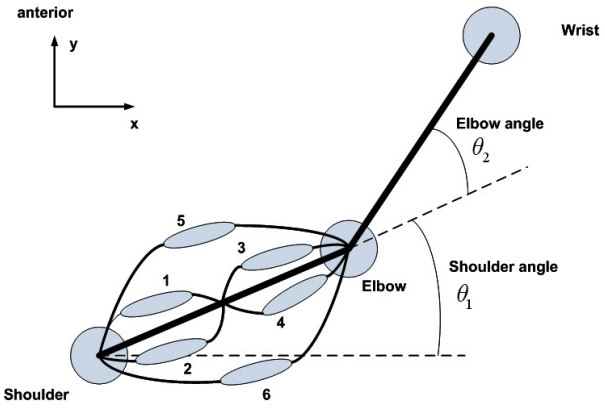


Figure 1. Humanoid robot arm model.

Considering the arm (including upper arm and lower arm) as a planar, two-link, articulated rigid object, the position of hand can be derived by a 2-vector q of two angles. The input is a 6-vector F_m of muscle forces. The dynamics of the rigid object is strongly nonlinear. Using the Lagrangian equations in classical dynamics, we get the dynamic equations of the ideal upper limb model

$$\begin{bmatrix} H_{11}(t) & H_{12}(t) \\ H_{21}(t) & H_{22}(t) \end{bmatrix}\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}+\begin{bmatrix} C_{11}(t) & C_{12}(t) \\ C_{21}(t) & C_{22}(t) \end{bmatrix}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} +\begin{bmatrix} G_1(t) \\ G_2(t) \end{bmatrix}=\begin{bmatrix} \tau_1(t) \\ \tau_2(t) \end{bmatrix} \quad (1)$$

or abbreviated as

$$H(t)\ddot{q}+C(t)\dot{q}+G(t)=\tau \quad (2)$$

with $q=\begin{bmatrix} q_1 & q_2 \end{bmatrix}^T=\begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}^T$ being the two joint angles. $\tau=\begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}^T=f(F_m)$ is joint torque which is a function of muscle force F_m

$$F_m=\begin{bmatrix} F_{m,1} & F_{m,2} & F_{m,3} & F_{m,4} & F_{m,5} & F_{m,6} \end{bmatrix}^T \quad (3)$$

$H(q,t)$ is an inertia matrix containing information with regard to the instantaneous mass distribution. $C(q,\dot{q},t)$ is centripetal and coriolis torques representing the moments of centrifugal forces. $G(q,t)$ represents gravitational torques changing with the posture configuration of the arm.

$$\begin{aligned} H_{11} &= J_1+J_2+m_2d_1^2+2m_2d_1c_2\cos(q_2) \\ H_{12} &= H_{21}=J_2+m_2d_1c_2\cos(q_2) \\ \mathrm{H_{22}} &= \mathrm{J_2} \\ C_{11} &= -2m_2d_1c_2\sin(q_2)\dot{q}_2 \\ C_{12} &= -m_2d_1c_2\sin(q_2)\dot{q}_2 \\ C_{21} &= m_2d_1c_2\sin(q_2)\dot{q}_1 \\ \mathrm{C_{22}} &= 0 \\ G_1 &= g(m_1c_1+m_2d_1)\cos(q_1)+gm_2c_2\cos(q_1+q_2) \\ G_2 &= gm_2c_2\cos(q_1+q_2) \end{aligned} \quad (4)$$

where g is the acceleration of gravity. c_i is the distance from the center of a joint i to the center of the gravity point of link i. d_i is the length of link i. $J_i=m_id_i^2+I_i$ where I_i is the moment of inertia about axis through the center of mass of link i. m_i is the mass of link i.n

B. Arm Model with Estimated Parameter

In our model, we assume that the arm model of Eq. (1) might be influenced by disturbances and perturbations from the environment. Hence, we use an estimated arm model to control, which is written as

$$\begin{bmatrix} \hat{H}_{11}(t) & \hat{H}_{12}(t) \\ \hat{H}_{21}(t) & \hat{H}_{22}(t) \end{bmatrix}\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}+\begin{bmatrix} \hat{C}_{11}(t) & \hat{C}_{12}(t) \\ \hat{C}_{21}(t) & \hat{C}_{22}(t) \end{bmatrix}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} +\begin{bmatrix} \hat{G}_1(t) \\ \hat{G}_2(t) \end{bmatrix}=\begin{bmatrix} \hat{\tau}_1(t) \\ \hat{\tau}_2(t) \end{bmatrix} \quad (5)$$

and which can also be abbreviated as

$$\hat{H}(t)\ddot{q}+\hat{C}(t)\dot{q}+\hat{G}(t)=\hat{\tau} \quad (6)$$

where $\hat{\cdot}$ means estimated value of $(\cdot)$. The connection between the ideal model (Eq.(1)) and estimated model (Eq. (5)) comes through the choice of $\tau=\hat{\tau}$. Below, we use the estimated model (5) to generate torques for the control of the

real system. In addition, the parameter adaptation updates the estimated parameters $\hat{H}$, $\hat{C}$ and $\hat{G}$ in real time.

For convenience during the following derivation, we define the actual and estimated arm parameter vector

$$P=\begin{bmatrix} P_H^T & P_C^T & P_G^T \end{bmatrix}^T,\ \hat{P}=\begin{bmatrix} \hat{P}_H^T & \hat{P}_C^T & \hat{P}_G^T \end{bmatrix}^T \tag{7}$$

where

$$P_H=\begin{bmatrix} H_{11}\\ H_{12}\\ H_{21}\\ H_{22}\end{bmatrix},\ P_C=\begin{bmatrix} C_{11}\\ C_{12}\\ C_{21}\\ C_{22}\end{bmatrix},\ P_G=\begin{bmatrix} G_1\\ G_2\end{bmatrix}$$
$$\hat{P}_H=\begin{bmatrix} \hat{H}_{11}\\ \hat{H}_{12}\\ \hat{H}_{21}\\ \hat{H}_{22}\end{bmatrix},\ P_C=\begin{bmatrix} \hat{C}_{11}\\ \hat{C}_{12}\\ \hat{C}_{21}\\ \hat{C}_{22}\end{bmatrix},\ P_G=\begin{bmatrix} \hat{G}_1\\ \hat{G}_2\end{bmatrix} \tag{8}$$

thus the estimation error vector can be defined as

$$\tilde{P}=\hat{P}-P=\begin{bmatrix} \tilde{P}_H^T & \tilde{P}_C^T & \tilde{P}_G^T \end{bmatrix}^T \tag{9}$$

C. Joint Torque Computation

Sliding control is used to control the posture of the arm [8]. A 2-vector q_d is the desired states. Define a sliding term s as

$$s=\dot{\tilde{q}}+\Lambda\tilde{q}=(\dot{q}-\dot{q}_d)+\Lambda(q-q_d) \tag{10}$$

where Λ is a positive diagonal matrix. Defining the reference velocity $\dot{q}_r$ and reference acceleration $\ddot{q}_r$ as

$$\begin{aligned}\dot{q}_r&=\dot{q}-s\\ \ddot{q}_r&=\ddot{q}-\dot{s}\end{aligned} \tag{11}$$

we then choose the control method as

$$\tau=\hat{H}(q)\ddot{q}_r+\hat{C}(q,\dot{q})\dot{q}_r+\hat{G}-K\mathrm{sgn(s)} \tag{12}$$

where K is a diagonal matrix and $\mathrm{sgn}(\cdot)$ is sign function. For the proof of sliding control we refer to [9].

D. Arm Parameter Adaptation

To accelerate the parameter update speed, we use two error sources to update estimated parameters. The first source is tracking error. We chose the parameter adaptation method based on tracking error as

$$\dot{\hat{P}}_{tra}=-\Gamma^{-1}\begin{bmatrix} s_1\ddot{q}_r^T & s_2\ddot{q}_r^T & s_1\dot{q}_r^T & s_2\dot{q}_r^T & s_1 & s_2\end{bmatrix}^T \tag{13}$$

where Γ is a diagonal matrix. The proof of this parameter adaptation method is in [10]. On the other side, the dynamic equation (Eq.(1)) can be written in the form

$$\begin{aligned}\tau&=S(\ddot{q},\dot{q},q)P\\ &=\begin{bmatrix} \ddot{q}_1 & 0 & \ddot{q}_1 & 0 & \dot{q}_1 & 0 & \dot{q}_1 & 0 & 1 & 0\\ 0 & \ddot{q}_2 & 0 & \ddot{q}_2 & 0 & \dot{q}_2 & 0 & \dot{q}_2 & 0 & 1\end{bmatrix}\begin{bmatrix} P_H\\ P_C\\ P_G\end{bmatrix}\end{aligned} \tag{14}$$

In this equation, τ is the "output" of the system. S is a signal matrix. P is a vector of real parameters. We can predict the value of the output τ based on the parameter estimation, i.e.

$$\hat{\tau}=S\hat{P} \tag{15}$$

Then the prediction error e can be defined as

$$e=\hat{\tau}-\tau=S\hat{P}-SP=S\tilde{P} \tag{16}$$

According to it, we can get the parameter adaptation method based on prediction error, i.e.

$$\begin{aligned}\dot{\hat{P}}_{pre}&=-\Xi\frac{\partial(e^Te)}{\partial\hat{P}}=-\Xi\frac{\partial\left(\left(S\hat{P}-SP\right)^T\left(S\hat{P}-SP\right)\right)}{\partial\hat{P}}\\ &=-2\Xi S^T\left(S\hat{P}-SP\right)=-2\Xi S^Te=-2\Xi S^T\left(\hat{\tau}-\tau\right)\end{aligned} \tag{17}$$

where Ξ is a diagonal coefficient matrix. If we assume that the parameters change much slower as compared to the parameter identification, from Eq. (17), we can get

$$\dot{\tilde{P}}=\dot{\hat{P}}-\dot{P}=-2\Xi S^TS\tilde{P} \tag{18}$$

Here we choose a Lyapunov function candidate

$$V(t)=\frac{1}{4}\tilde{P}^T\tilde{P} \tag{19}$$

then the derivative of $V(t)$ is

$$\dot{V}(t)=\frac{1}{2}\tilde{P}^T\dot{\tilde{P}}=\frac{1}{2}\tilde{P}^T\left(-2\Xi S^TS\tilde{P}\right)=-\Xi\left(S\tilde{P}\right)^T\left(S\tilde{P}\right)\le 0 \tag{20}$$

which means the parameter estimation converges to real values. Therefore, according to Eq. (13) and Eq. (17), the overall adaptation law is

$$\begin{aligned}\dot{\hat{P}}&=\dot{\hat{P}}_{tra}+\dot{\hat{P}}_{pre}\\ &=-\Gamma^{-1}\begin{bmatrix} s_1\ddot{q}_r^T & s_2\ddot{q}_r^T & s_1\dot{q}_r^T & s_2\dot{q}_r^T & s_1 & s_2\end{bmatrix}^T\\ &\quad -2\Xi S^T\left(\hat{\tau}-\tau\right)\end{aligned} \tag{21}$$

E. Jacobian Matrix between Muslce and Joint Space

The coordinate system of the robot arm is shown in Fig. 2 where we define $a_{ij}\left(1\le i\le 6, 1\le j\le 2\right)$ as the distance between the muscle endpoint and center of its adjacent joint. Define $l_k\left(1\le k\le 6\right)$ as the k-th muscle length.

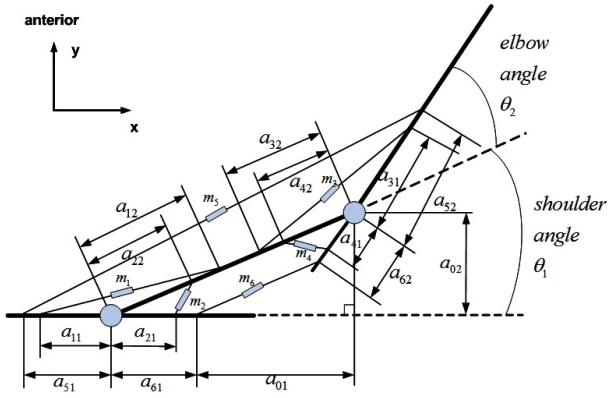


Figure 2. Humanoid robot arm model.

According to the Sine Law and the Cosine Law, we can get

$$
\begin{aligned}
l_1^2 &= a_{11}^2 + a_{12}^2 - 2a_{11}a_{12}\cos(\pi-\theta_1)\\
l_2^2 &= a_{21}^2 + a_{22}^2 - 2a_{21}a_{22}\cos(\theta_1)\\
l_3^2 &= a_{31}^2 + a_{32}^2 - 2a_{31}a_{32}\cos(\pi-\theta_2)\\
l_4^2 &= a_{41}^2 + a_{42}^2 - 2a_{41}a_{42}\cos(\theta_2)
\end{aligned}
\tag{22}
$$

We constructed a right triangle to calculate l_5 and l_6

$$
\begin{aligned}
l_5^2 &= (a_{51}+a_{01})^2 + (a_{52}+a_{02})^2 - 2(a_{51}+a_{01})(a_{52}+a_{02})\cos(\pi-\theta_1-\theta_2)\\
l_5^2 &= (a_{01}-a_{61})^2 + (a_{02}-a_{62})^2 - 2(a_{01}-a_{61})(a_{02}-a_{62})\cos(\pi-\theta_1-\theta_2)
\end{aligned}
\tag{23}
$$

where a_{01} and a_{02} can be obtained by the Sine Law

$$
a_{01} = \frac{d_1\sin(\theta_2)}{\sin(\theta_1+\theta_2)},\quad a_{02} = \frac{d_1\sin(\theta_1)}{\sin(\theta_1+\theta_2)} \tag{24}
$$

After simplification, we finally get

$$
\begin{aligned}
l_1 &= \left(a_{11}^2 + a_{12}^2 + 2a_{11}a_{12}\cos(\theta_1)\right)^{1/2}\\
l_2 &= \left(a_{21}^2 + a_{22}^2 - 2a_{21}a_{22}\cos(\theta_1)\right)^{1/2}\\
l_3 &= \left(a_{31}^2 + a_{32}^2 + 2a_{31}a_{32}\cos(\theta_2)\right)^{1/2}\\
l_4 &= \left(a_{41}^2 + a_{42}^2 - 2a_{41}a_{42}\cos(\theta_2)\right)^{1/2}\\
l_5 &= \left((a_{51}+a_{01})^2 + (a_{52}+a_{02})^2 + 2(a_{51}+a_{01})(a_{52}+a_{02})\cos(\theta_1+\theta_2)\right)^{1/2}\\
l_6 &= \left((a_{01}-a_{61})^2 + (a_{02}-a_{62})^2 + 2(a_{01}-a_{61})(a_{02}-a_{62})\cos(\theta_1+\theta_2)\right)^{1/2}
\end{aligned}
\tag{25}
$$

Assuming the kinematics between muscle length and joint angle is given by

$$
L = Q(\Theta) \tag{26}
$$

where

$$
\begin{aligned}
L &= \begin{bmatrix} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \end{bmatrix}^T\\
\Theta &= \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}^T
\end{aligned}
\tag{27}
$$

then the derivative of Eq. (26) is

$$
\dot{L} = J_m\dot{\Theta} \tag{28}
$$

where J_m is a Jacobian matrix between the joint space and the muscle space. It has a format as

$$
J_m = \begin{bmatrix}
J_{m,1,1} & J_{m,1,2}\\
J_{m,2,1} & J_{m,2,2}\\
J_{m,3,1} & J_{m,3,2}\\
J_{m,4,1} & J_{m,4,2}\\
J_{m,5,1} & J_{m,5,2}\\
J_{m,6,1} & J_{m,6,2}
\end{bmatrix} \tag{29}
$$

where

$$
\begin{aligned}
J_{m,1,1} &= \frac{-a_{11}a_{12}\sin(\theta_1)}{\sqrt{a_{11}^2 + 2\cos(\theta_1)a_{11}a_{12} + a_{12}^2}}\\
J_{m,2,1} &= \frac{-a_{21}a_{22}\sin(\theta_1)}{\sqrt{a_{21}^2 + 2\cos(\theta_1)a_{21}a_{22} + a_{22}^2}}\\
J_{m,3,2} &= \frac{-a_{31}a_{32}\sin(\theta_2)}{\sqrt{a_{31}^2 + 2\cos(\theta_2)a_{31}a_{32} + a_{32}^2}}\\
J_{m,4,2} &= \frac{a_{41}a_{42}\sin(\theta_2)}{\sqrt{a_{41}^2 + 2\cos(\theta_2)a_{41}a_{42} + a_{42}^2}}\\
J_{m,1,2} &= J_{m,2,2} = J_{m,3,1} = J_{m,4,1} = 0
\end{aligned}
\tag{30}
$$

$J_{m,5,1}$, $J_{m,5,2}$, $J_{m,6,1}$ and $J_{m,6,2}$ are long equations and we do not provide them here. The derivation of the above Jacobian matrix can be done by Matlab Symbolic Toolbox. The program for this derivation is shown in Appendix.

F. Muscle Force Distribution

Hence, the relationship between muscle forces and joint torques can be derived by the principle of virtual work as

$$
\tau = f(F_m) = J_m^T F_m \tag{31}
$$

Hence, the inverse relation between the joint torques and the muscle forces can be expressed as

$$
\tau_{inv} = f^{-1}(\tau) = \left(J_m^T\right)^+ \tau \tag{32}
$$

where

$$
\left(J_m^T\right)^+ = J_m\left(J_m^T J_m\right)^{-1} \tag{33}
$$

is pseudo-inverse matrix of J_m^T. The above muscle force distribution solution satisfies

$$
\min\|F_m\| \quad s.t.\ J_m^T F_m = \tau \tag{34}
$$

which means the pseudoinverse is an optimization solution to obtain minimum muscle distribution force. However, the above solution does not consider physical constraints, such as the fact that the maximum output force of muscles is limited, that muscles can only contract, etc. To involve these constraints, we define F_{in} as a voluntary vector having the same dimension with F_m which expresses the internal forces generated by redundant muscles. Then we can define the internal force in F_m space, i.e.

$$
g(F_{in}) = \left(I - \left(J_m^T\right)^+ J_m^T\right)F_{in} \tag{35}
$$

where I is an identity matrix having the same dimension with muscle space. According to Moore-Penrose pseudoinverse, $g(F_{in})$ is orthogonal with the pseudo-inverse solution. Thus, we can choose any vector as F_{in}. Below, we give a gradient direction for F_{in} to make F_m satisfies boundary constraints.

Here, we assume that each muscle force is limited in the interval from $F_{m,i,min}$ to $F_{m,i,max}$ for $1 \le i \le 6$. Our objective is to choose a gradient direction to make each element of $F_{m,i}\ (1 \le i \le 6)$ equal or greater than $F_{m,i,min}$, and equal or less than $F_{m,i,max}$. Considering muscle fatigue, one reasonable way to achieve minimal fatigue is to make each output force of the muscles to be around at the middle magnitude between $F_{m,i,min}$ and $F_{m,i,max}$. The physical meaning of this

method is to distribute load to all muscles into their proper load interval, so that they can continue working for a longer time. Based on these considerations, we choose a function h as

$$h(F_m)=\sum_{j=1}^{6}\left(\frac{\tau_{inv,i}-F_{m,i,mid}}{F_{m,i,mid}-F_{m,i,max}}\right)^2 \tag{36}$$

where

$$\begin{aligned} &0\le F_{m,i,min}\le\tau_{inv,i}\le F_{m,i,max}\\ &F_{m,i,mid}=\frac{F_{m,i,min}+F_{m,i,max}}{2}\\ &i=1,2,\cdots 6 \end{aligned} \tag{37}$$

then we chose F_{in} as the gradient of the function h, i.e.

$$F_{in}=K_{in}\frac{\partial h(\tau_{inv})}{\partial\tau_{inv}}=K_{in}\nabla_h=K_{in}\cdot\begin{bmatrix} 2\cdot\dfrac{\tau_{inv,1}-F_{m,1,mid}}{F_{m,1,mid}-F_{m,1,\max}}\\ 2\cdot\dfrac{\tau_{inv,2}-F_{m,2,mid}}{F_{m,2,mid}-F_{m,2,\max}}\\ 2\cdot\dfrac{\tau_{inv,3}-F_{m,3,mid}}{F_{m,3,mid}-F_{m,3,\max}}\\ 2\cdot\dfrac{\tau_{inv,4}-F_{m,4,mid}}{F_{m,4,mid}-F_{m,4,\max}}\\ 2\cdot\dfrac{\tau_{inv,5}-F_{m,5,mid}}{F_{m,5,mid}-F_{m,5,\max}}\\ 2\cdot\dfrac{\tau_{inv,6}-F_{m,6,mid}}{F_{m,6,mid}-F_{m,6,\max}} \end{bmatrix} \tag{38}$$

where K_{in} is a scalar matrix. It is very easy to prove that the direction of F_{in} points to $F_{m,i,mid}$. According to the computation in Eq. (32) and Eq. (38), the muscle force is calculated as

$$F_m=\tau_{inv}+g(F_{in}) \tag{39}$$

III. RESULTS

The performance of the proposed muscle force control method was tested though a simulation of reaching movements. The desired movement is bending the upper arm and lower arm from 0 rad to $\pi/2$ rad and then stretching them back to 0 rad. The total simulation time is 10s.

A. Arm Model Parameter Setting

The parameters of the robot arm are based on the real data of a human upper limb. The setting of length, mass, mass center position and inertia coefficients are shown in Table 1. The anthropological data comes from [11]. Without loss of generality, the muscle configuration coefficients (in Eq. (25)) are set as $a_{ij}=0.1m\ (1\le i\le 6, 1\le j\le 2)$.

TABLE I. ANTHROPOLOGICAL PARAMETER VALUE

Segment	Upper arm	Lower arm
Length (m)	0.282	0.269
Mass (kg)	1.980	1.180
MCS Pos (m)	0.163	0.123
I_{11} (kg.m2)	0.013	0.007
I_{22} (kg.m2)	0.004	0.001
I_{33} (kg.m2)	0.011	0.006

MCS Pos means position of the mass center.

B. Computational Coefficient Setting

There are three groups of parameters that need to be set: the parameters for sliding control, the parameters for parameter adaptation, and the parameters for muscle force computation. These parameters are set as follows.

The control parameters are set by (Eq. (12)) as

$$K=20\cdot Diag\left(\begin{bmatrix}1 & 1\end{bmatrix}\right) \tag{40}$$

the adaptation parameters (Eq. (21)) are set as

$$\begin{aligned} \Gamma^{-1}&=0.0015\cdot Diag\left(\begin{bmatrix}1&1&1&1&1&1&1&1&1&1\end{bmatrix}\right)\\ 2\Xi&=0.001\cdot Diag\left(\begin{bmatrix}1&1&1&1&1&1&1&1&1&1\end{bmatrix}\right) \end{aligned} \tag{41}$$

and the muscle force computation parameters (Eq. (38)) are set as

$$\begin{aligned} K_{in}&=200\cdot Diag\left(\begin{bmatrix}1&3&1&1&1&2\end{bmatrix}\right)\\ F_{m,i,min}&=0,\quad F_{m,i,max}=1000\quad (1\le i\le 6) \end{aligned} \tag{42}$$

where $Diag(\cdot)$ is a diagonal matrix with diagonal elements being as $(\cdot)$.

C. Control Performance

According to the bend-stretch movement, two sinusoidal waves are set as reference signals for q_1 and q_2. The frequency of the two waves is set as 2π. Initial states of q_1 and q_2 are set as zero. Based on the joint torque coming from sliding control, we computed 6 muscle forces as shown in Fig. 3. All the muscle forces are in the range of $[F_{m,i,min}, F_{m,i,max}]$ for $(1\le i\le 6)$. These muscle forces are optimized to be around $F_{m,i,mid}$.

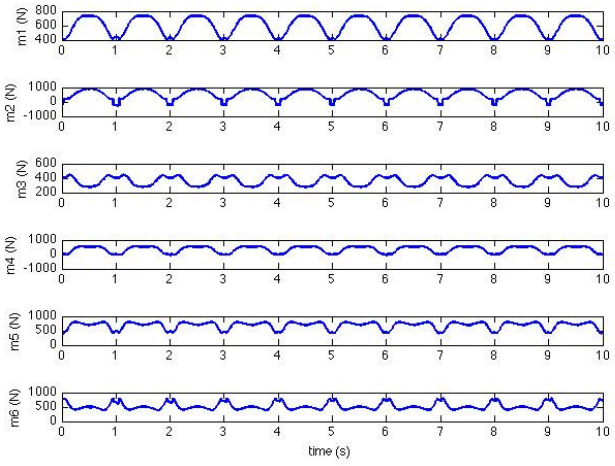
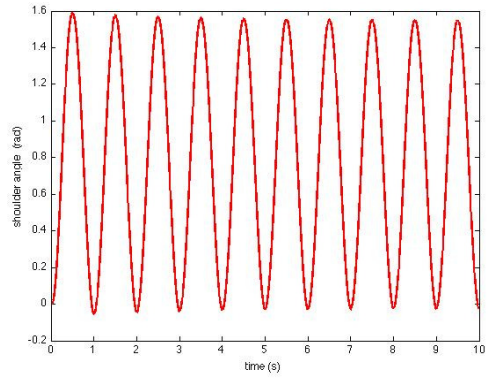
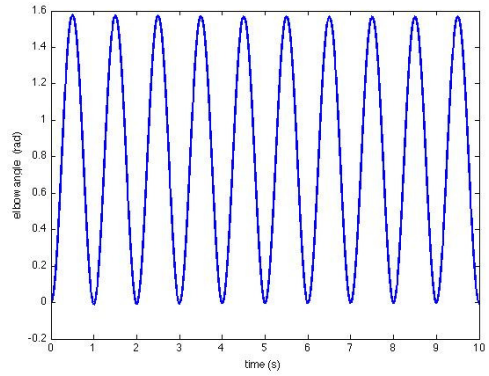


Figure 3. Muscle force.

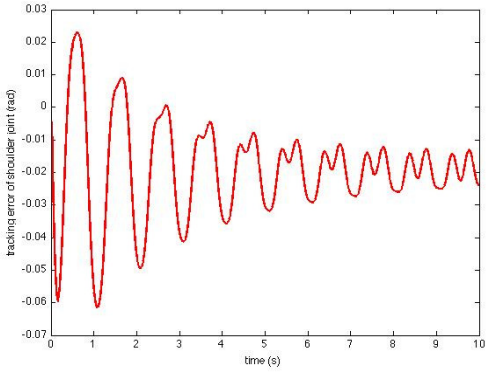
These computed muscle forces are used to control the humanoid robot arm model. The shoulder angle and elbow angle are shown in Fig. 4 (a) and (b), respectively. Compared with the desired trajectory, the tracking error of the shoulder joint and elbow joint are shown in Fig. 4 (c) and (d), respectively. It is clear the tracking performance is good. Additionally, both the two tracking errors decrease gradually. The reason is that the parameter update makes the estimated model parameters to approach the real ones gradually.



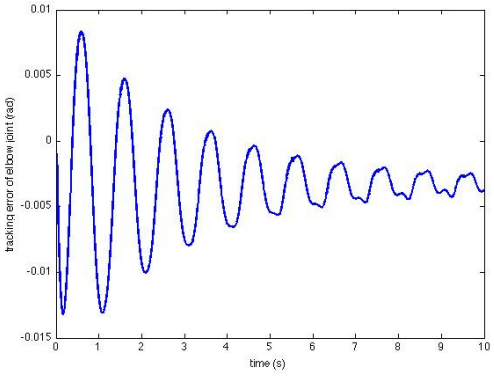
(a)



(b)



(c)

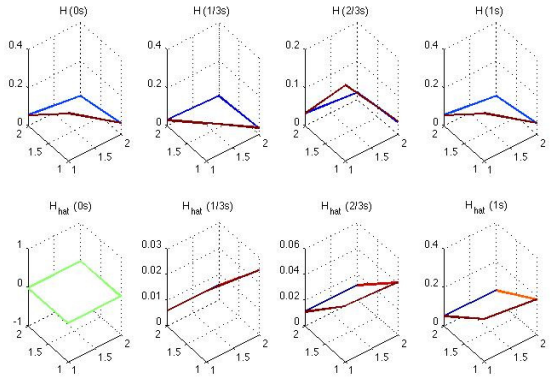


(d)

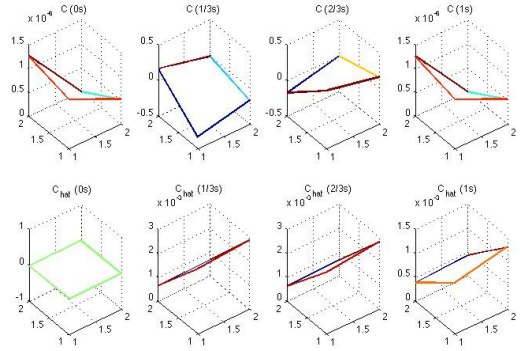
Figure 4. Arm control performance. (a) Shoulder angle. (b) Elbow angle. (c) Tracking error of the shoulder joint. (d) Tracking error of the elbow joint.

D. *Parameter Adaptation*

In order to test the functionality of the designed parameter adaptation method, we set the initial estimated model parameters $\hat{H}$, $\hat{C}$ and $\hat{G}$ to correspond to a zero matrix (or zero vector) at the beginning. After that, the parameter adaptation method adjusts the parameters based on the tracking error and the prediction error. Fig. 5 shows the parameters H, $\hat{H}$ and C, $\hat{C}$ in the time interval $[0,1]$ s. We took four snapshots of these parameters at the moment 0s, 1/3s, 2/3s, 1s, respectively. It is noted that, the estimated parameters do not coincide with real parameters. That is because the dynamic features of the model have been only partially explored through our simulated movement. The more complicated the movement that is chosen is, it is expected that the more consistency between the estimated parameters and the real parameters is exhibited.



(a)



(b)

Figure 5. Arm parameter update. (a) Snapshots of H and $\hat{H}$. (b) Snapshots of C and $\hat{C}$.

E. *Animation*

A humanoid arm was visualized by utilizing Simulink (SimMechanics Toolbox). The arm model consists of three parts: torso, right numerus, and right ulna radius hand. The three parts are created by 12 bones, 2 bones, and 58 bones, respectively. The polygon files of these bones come from SIMM. To make Simulink be able to import these polygons, we converted the format of the polygon files from .vtp files to .stl files. Fig. 6 shows three phases (i.e., start phase, middle phase, and end phase) of the arm gesture change, during one bend-stretch movement circle.

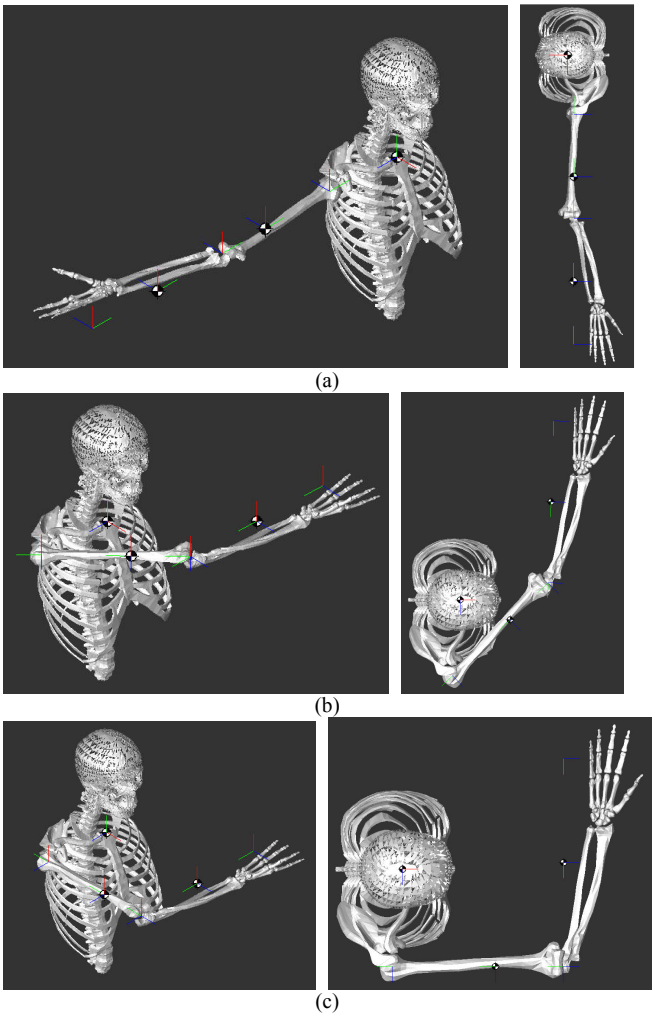


Figure 6. Arm movement snapshots. (a) Start phase. (b) Middle phase. (c) End phase.

IV. Conclusion

In this paper, after discussing the benefits of muscle-like systems for robots arms as compared to traditional one-motor-per-joint approaches, we proposed an adaptive biarticular muscle force control method, which exhibits a number of beneficial properties. Through our method, the derived muscle forces stay within prefixed bounds. Additionally, muscle forces are optimized to be in the middle of their output force range, corresponding to minimal fatigue. Our proposed method is not only easily expandable to other configurations, but it can also be combined with many other methods that output joint torque. Therefore, this paper provides a flexible solution for controlling a muscle-like-driven system. Furthermore, our muscle force distribution method provides a general solution for redundancy problem, and has proven its effectiveness and benefits through our derived simulation results.

References

[1] S. Oh and Y. Hori, "Development of two-degree-of-freedom control for robot manipulator with biarticular muscle torques," in American Control Conference, 2009, pp. 325-330.

[2] S. Klug, B. Mohl, O. V. Stryk, and O. Barth, "Design and application of a 3 DOF bionic robot arm," in 3rd International Symposium on Adaptive Motion in Animals and Machines, 2005, pp. 1-6.

[3] V. Potkonjak, K. M. Jovanovic, P. Milosavljevic, N. Bascarevic, and O. Holland, "The puller-follower control concept in the multi-jointed robot body with antagonistically coupled compliant drives," in IASTED International Conference on Robotics, 2011, pp. 375-381.

[4] K. Tahara, Z. Luo, S. Arimoto, and H. Kino, "Sensor-motor control mechanism for reaching movements of a redundant musculo-skeletal arm," Journal of Robotic Systems, vol. 22, pp. 639-651, 2005.

[5] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in Robotics Research: The First International Symposium, ed: MIT Press, 1984, pp. 735-748.

[6] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," International Journal of Robotics Research, vol. 4, pp. 109-117, 1985.

[7] A. Heim and O. V. Stryk, "Trajectory optimization of industrial robots with application to computer-aided robotics and robot controllers," Optimization, vol. 47, pp. 407-420, 2000.

[8] J. E. Slotine and W. Li, Applied Nonlinar Control: Prentice Hall, 1991.

[9] J. E. Slotine and W. Li, "Adaptive manipulator control: A case study," IEEE Transactions on Automatic Control, vol. 33, pp. 995-1003, 1988.

[10] H. Dong, Z. Luo, and A. Nagano, "Adaptive attitude control for redundant time-varying complex model of human body in the nursing activity," Journal of Robotics and Mechatronics, vol. 22, pp. 418-429, 2010.

[11] A. Nagano, S. Yoshioka, T. Komura, R. Himeno, and S. Fukashiro, "A three-dimensional linked segment model of the whole human body," International Journal of Sport and Health Science, vol. 3, pp. 311-325, 2005.

Appendix
Matlab Code for Deriving the Jacobian Matrix

```
% Usage: need Matlab Symbolic Toolbox
syms a11 a12 a21 a22 a31 a32 a41 a42 a51 a52 a61 a62
syms d1 l1 l2 l3 l4 l5 l6 theta1 theta2
l1=sqrt(a11^2+a12^2+2*a11*a12*cos(theta1));
l2=sqrt(a21^2+a22^2-2*a21*a22*cos(theta1));
l3=sqrt(a31^2+a32^2+2*a31*a32*cos(theta2));
l4=sqrt(a41^2+a42^2-2*a41*a42*cos(theta2));
l5=sqrt((a51+d1*sin(theta2)/sin(theta1+theta2))^2+(a52+d1*sin(theta1)…
/sin(theta1+theta2))^2+2*(a51+d1*sin(theta2)/sin(theta1+theta2))…
*(a52+d1*sin(theta1)/sin(theta1+theta2))*cos(theta1+theta2));
l6=sqrt((d1*sin(theta2)/sin(theta1+theta2)-a61)^2+(d1*sin(theta1)/…
sin(theta1+theta2)-a62)^2+2*(d1*sin(theta2)/sin(theta1+theta2)-a61)…
*(d1*sin(theta1)/sin(theta1+theta2)-a62)*cos(theta1+theta2));
% Jm is the Jacobian matrix
Jm=jacobian([l1; l2; l3; l4; l5; l6], [theta1 theta2]);
```