Upper Limb Muscle Dynamics

Hungtang Ko

School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, USA
hko40@gatech.edu

Visak C.V. Kumar

School of Interactive Computing
Georgia Institute of Technology
Atlanta, USA
visak3@gatech.edu

***Abstract*—Upper limb musculoskeletal injuries are associated with professional activities in sports and Chinese cooking, often times forcing them to retire early. In this report, we study the muscle dynamics in the upper limbs. We establish an efficient computational framework in Python that simulates the upper limb motion based on the activation levels of the muscles. We validate the model by comparing against the Hopping model in MATLAB Simulink environment and the two-linkage dynamics when the muscles are turned off. We show that different prescribed activation level can lead to a clockwise or counterclockwise motion in the hand depending on the delays among activation signals. We also use the reinforcement learning to demonstrate the effect of added exoskeleton on the muscle kinematics when the same motion needs to be generated.**

I. Introduction

There are many tasks that are demanding to the upper-limb muscle groups. In sports, for examples, it is a common struggle among pitchers and batters to deal with shoulder or elbow injuries. Another notable case is the wok tossing action commonly used in Chinese cuisine cooking. It is highly dynamical and requires a large amount of muscle work. As a result, the shoulder musculoskeletal injury rate is reported to be as as high as 60% among chefs [1]. It is of high interest to develop a muscle-level dynamical model for the upper-limb motion which will enable us to understand the activation levels profile in different tasks.

While Hill-type model provides a means to estimate the muscle force based on their kinematics and activation levels, it is difficult to estimate what the activation levels are when certain motion and force are exerted using traditional computational methods. In recent research, reinforcement Learning (RL) has shown great promise in learning complex motor skills. Yu et al. [2], Peng et al. [3] and Peng et al. [4] have shown that RL algorithms can not just be applied to learn human locomotion tasks but surpass all expectations. These impressive results have been in simulated environments where the action space is either torques or positions. In a more relevant setting to this work, interesting results have also been published with muscles as actuators in NIPS learning to run challenge [5], where a simulated human lower limb model (OpenSim) was used to learn walking and running motions. However, there has been little research in using reinforcement learning to understand how an exoskeleton interacts with human muscle behaviour. We want to explore this promising intersection between machine learning and wearable robotics research.

Combining the knowledge in muscle dynamics, multi-link physics and machine learning, we develop a Python model that maps the relationship between the activation levels of the four major upper-arm muscles and the joint-level movement. In the first step of the investigation, we validated the model against the single-muscle hopping model developed by the course instructors. In the second step, we validated that the model follows the two-linkage dynamics when the muscles are turned off. Then, we used the prescribed activation levels to study the free movement of the upper limbs without any constrains. Lastly, we used reinforcement learning algorithm to study the activation signals when the hand is following a circular trajectory. Finally, We investigated the effect of the exoskeleton property on the muscle behavior.

II. Model Description & Algorithm Outline

The forward model that describes the motion based on the activation level is provided by Hill et el. [6]. The force each muscle generate can be split into the active and passive component. The active component depends on the activation level of the muscle, the length of the muscle and the velocity of the muscle. Both the passive component of the muscle and the tendon can be modelled as nonlinear springs which are functions of length changes. The model [6] has 4 parameters for each muscle. They are maximum contraction force, optimal muscle fascicle length, maximum muscle velocity, tendon stiffness and slack length.

Most of the parameters for the upper limb muscles can be found in references. We obtain the maximum contraction force, optimal fascicle length and tendon slack length from Gordon et el. [7]. The major muscles that we chose to include in the model are anterior(AD) and posterior Deltoid(PD) that act on the shoulder and Biceps(BB) and Triceps(TB) that act on the elbow. The four muscles are chosen because they have the most force output capabilities in the plane that's perpendicular to the body. Both Biceps and Triceps have sub-muscles groups that have slightly different muscle length to tendon length ratio. In our simplified model, to represent the properties of Biceps and Triceps, we used the summation of the maximum force from different group and the optical muscle length and tendon length of the long muscle groups. We were unable

to find useful reference for the value of maximum muscle velocity and tendon stiffness. Therefore, we use the muscle length to re-scale the maximum muscle velocity from Soleus muscle in the Hopper model. The tendon stiffness that we used in the model are the same as the value used in the Hopper model. We obtain the bone parameters from the reference [7]. All the parameters we used in the model are listed in the table I and II.

TABLE I: Muscle parameters used in the model are the maximum muscle force Fm_{max}, optimal muscle length $Lm_{optimal}$ and tendon slack length Lt_{slack}. Parameter values obtained from [7]. Maximum muscle velocity Vm_{max} is calculated by $Vm_{max} = 8.18Lm_{optimal}$ m/s. Tendon stiffness for all the muscles is $1.8X10^5$ N/m

Muscle Tendon Unit	$Fm_{max}(N)$	$Lm_{optimal}(m)$	$Lt_{slack}(m)$
Biceps(BB)	1063.99	0.16	0.23
Triceps(TB)	2004.65	0.14	0.12
Anterior Deltoid(AD)	1147.99	0.11	0.10
Posterior Deltoid(PD)	265.99	0.14	0.04

TABLE II: Bone parameters used in the model are mass m, total length l, distance from the center of mass r and moment of inertia with respect to the mass center in the direction perpendicular to the bone I. The bone in the upper arm is Humerus and the bones in the lower arm are Ulna and Radius. The parameters we used for the lower limb is obtained from adding or averaging over the two bones. Parameter values obtained from [7]

Link	$m(kg)$	$l(m)$	$r(m)$	$I(10^6 \cdot kg \cdot m^2)$
Upper	1.79	0.30	0.13	1.32
Lower	1.09	0.25	0.10	0.56

We used the Pigeon et al [8] to model the muscle length and moment arm of the muscle as a function joint angle for the elbow joint, the paper reported detailed information about the polynomial approximations that were made to model muscle lengths and moment arms. The moment arms for anterior and posterior Deltoids were re-constructed from the figures reported in [9], they were also compared to values reported in [7] and they agree well. However, there is a lack of data about the muscle lengths as a function of the joint angles for the shoulder muscles, to tackle this problem, we model the change in MTU lengths as linear, this seems to be a reasonable approximation when compared to the MTU curves obtaind from literature for Biceps and Triceps muscles which are fairly linear as well. This is illustrated in figure 1.

In this model, only the movement in the plane vertical to the human body is considered. Therefore, the shoulder muscles that lift the arms to the side can be discarded. Following the literature, we use the convention that when the $\theta_1 = 0$ and $\theta_2 = 0$ the arm is straight and pointing vertically downwards and the positive direction of the two joint angles is the direction that lift the arm to the front.

III. Muscle Dynamics Validation

We compared the Python model we built against the MATLAB Simulink model where the hopping dynamics is simulated. In this simplified case, only one muscle is used. The force produced by the muscle is scaled through a constant ratio to push the center of mass vertically upward. The equation of

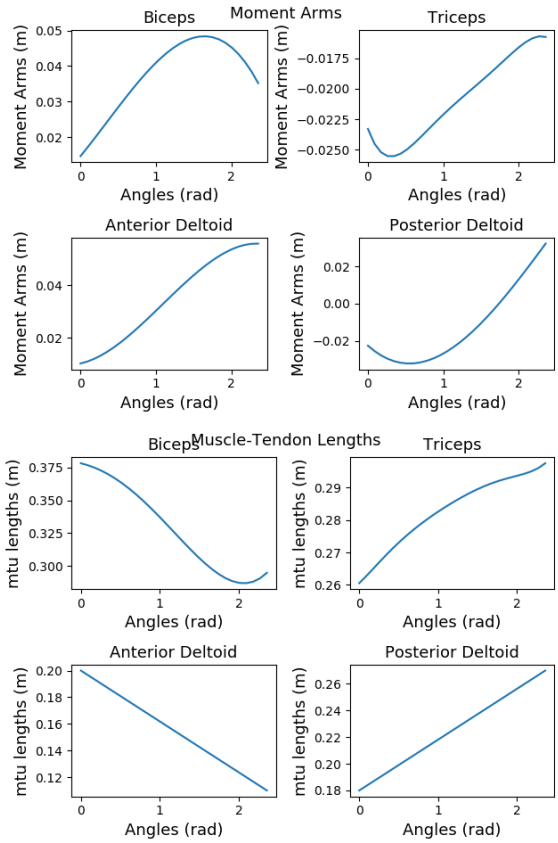


Fig. 1: Moment-arm and muscle lengths as a function of joint angles for each muscle.

motion can be simulated under the same parameter setting in the Simulink model as well as in the Python model. For the sake of the simplicity, we made some minor changes to the virtual hopper feedback code that we directly got from course materials in week 8 from Canvas. First, we took out the feedback part in the stimulation block. By doing this, the activation signal is independent of the muscle state. It is crucial because one of the objectives of the project is to inform the activation signal. It has to be set to be an independent variable. In the validation against the Hopper Model, the activation signal is a prescribed square function with 10% duty cycle at 2.5 Hz. The second change that we made was taking away the aerial part of the model. The force loading environment in the upper limb is task-dependent. For the purpose of this investigation, we only compared against the hopping model without specific consideration for the aerial stages. This was done by removing the switch in the Load Dynamics block assuming the person is always in stance. Fig 2 shows that our Python model can reproduce the result from the hopper model when only one muscle is active.

IV. Two Link Dynamics Validation

The muscle environment of the upper limb follows the physics of the two linkage system. Following the reference [10], the dynamic system can be described as:

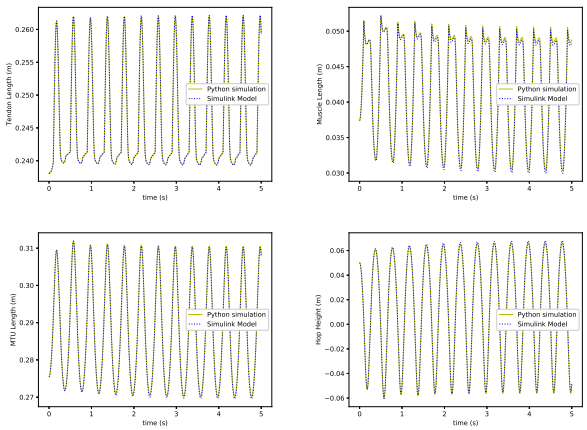


Fig. 2: Simulink model(blue dash) and custom Python code(yellow) shows similar result in calculating Tendon length, muscle length, muscle-tendon unit length and the hop height during single-muscle hopping.

$$M \cdot \ddot{\vec{q}} + C \cdot \dot{\vec{q}} + \vec{\tau}_G = \vec{\tau} \tag{1}$$

In the equation, $\vec{q} = [\theta_1, \theta_2]^T$ is the state vector, dots overhead indicate time derivative. M and C are two-by-two inertial and damping matrices, respectively. $\vec{\tau}_G$ is the torque the gravity acts on the two joints. $\vec{\tau} = [\tau_1, \tau_2]$ is the vector that describe the external torque on the arm. This can include the torque generated by the muscle and external robotic device. Any external loading shall also be included in this term.

To express the inertia matrix in a compact form, we define:

$$\alpha = I_1 + I_2 + m_1 r_1^2 + m_2(l_1^2 + r_2^2) \tag{2}$$

$$\beta = m_2 l_1 r_2 \tag{3}$$

$$\delta = I_2 + m_2 r_2^2 \tag{4}$$

,where I_i, m_i, l_i, r_i are the moment of inertia, mass and the limb length and the distance from the center of mass to the joint, with subscript $i = 1, 2$ representing shoulder and elbow respectively. Then, we can have:

$$M = \begin{bmatrix} \alpha + 2\beta c_2 & \delta + \beta c_2 \\ \delta + \beta c_2 & \delta \end{bmatrix} \tag{5}$$

$$C = \begin{bmatrix} -\beta s_2 \dot{\theta}_2 & -\beta s_2(\dot{\theta}_1 + \dot{\theta}_2) \\ \beta s_2 \dot{\theta}_1 & 0 \end{bmatrix} \tag{6}$$

$$\vec{\tau}_G = \begin{bmatrix} (m_1 r_1 + m_2 l_1) g \sin\theta_1 + m_2 g r_2 \sin(\theta_1 + \theta_2) \\ m_2 g r_2 \sin(\theta_1 + \theta_2) \end{bmatrix} \tag{7}$$

In the equations above, g is the magnitude of the gravitational acceleration. In the case of free movement i.e. movement without external device or loading, $\tau = \tau_{muscle}$, which can be expressed as:

$$\tau_{muscle} \begin{bmatrix} MA_{AD}(\theta_1) F_{AD} - MA_{PD}(\theta_1) F_{PD} \\ MA_{BB}(\theta_2) F_{BB} - MA_{TB}(\theta_2) F_{TB} \end{bmatrix} \tag{8}$$

,where MA is the moment arm of each muscle, which is a function of the joint angles as described in figure 1.

Notably, the system can have its own natural frequency even without the muscle forces. It is critical to estimate the natural frequency of the system. When the muscles are active, by examining how far the simulated joint trajectory frequency is away from the natural frequency, we can know how strong the muscle is influencing the dynamic system.

For simplicity, instead of calculating the actual natural frequency of the two degrees of freedom system which requires matrix decomposition, we calculate the natural frequency of the joint when the other joint is fixed. Therefore, the natural frequency(NF) of the shoulder and the elbow can be expressed as followed:

$$NF_{shoulder} = \frac{1}{2\pi} \sqrt{\frac{\partial \vec{\tau}_{G1}}{\partial \theta_1} / M_{11}} \tag{9}$$

$$NF_{elbow} = \frac{1}{2\pi} \sqrt{\frac{\partial \vec{\tau}_{G2}}{\partial \theta_2} / M_{22}} \tag{10}$$

The natural frequencies of the two joints are a function of both joint angles. For example, when the elbow angle different, the center of mass of the arm change in position which results in different shoulder natural frequencies. To compare our model with the theoretical result, we simulated two cases that have the same initial condition when $\theta_1 = 0$ and $\theta_2 = \pi/2$. In the first case, the elbow is fixed and in the second, the shoulder is fixed. We purposefully relaxed the physical constrain that the shoulder angle shouldn't be negative. The simulated joint kinematics is shown in fig. 3. The calculated frequencies match with the theoretical values at 0.9 Hz and 1.28 Hz, respectively for the shoulder(left) and for the elbow(right).

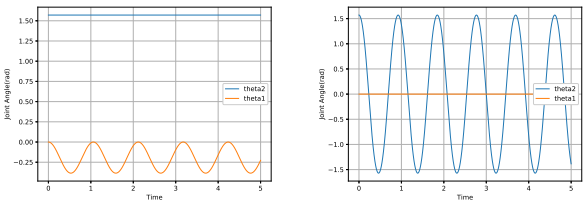


Fig. 3: Joint trajectories when all the muscles are inactive. Both cases have the same initial condition but in case 1 (left) the elbow joint is fixed while in case 2 (right) the shoulder joint is fixed. Both joint signals oscillate close to the theoretical natural frequency.

V. Free Upper-limb Movement

We performed the numerical simulation with the full model that incorporated both the muscle dynamics and the two-linkage physics(eq. 1). In all the cases in this section, the physical model is only subject to gravity and the muscle force. We prescribed the activation signal to calculate the joint movement as a simulation output. In practice, we predefined the activation level signals to be periodic square waves. The input parameters we vary to change the activation levels are the peak level, the background level, duty cycle and the time delays. The activation frequencies in all cases in this section are 3 Hz. We also used the same initial condition for all the cases here where $\theta_1 = 0$ and $\theta_2 = \pi/2$. The background activation level is only used for the Biceps because it was necessary to prevent lower arm from falling and hitting the shoulder joint limit.

In case 1, only elbow muscles are active. Fig. 4 shows the result of the simulation. As mentioned previously, it was necessary to have background activation at the Biceps to keep the lower arm from falling. With the pair of elbow antagonist muscles being activated completely out of phase, the elbow joint moment is well controlled and moves at the activation frequency at 3 Hz. However, the dominant frequency at the shoulder joint is at a lower value which correspond to the natural frequency.

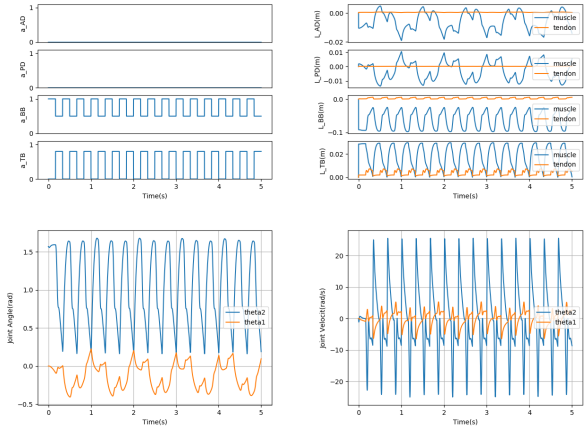


Fig. 4: Activation level(top left), change in tendon and muscle length(top right), joint angles(bottom left) and joint velocities(bottom right) when the two elbow muscles are active and out of phase.

In case 2, except for the constant activation of Biceps, only shoulder muscles are active. Fig. 5 shows the result of the simulation. With the shoulder muscles being fired at the prescribed frequency, in this case, the motion of the shoulder joint is well controlled at 3 Hz. The natural frequency of the system is barely noticeable anymore.

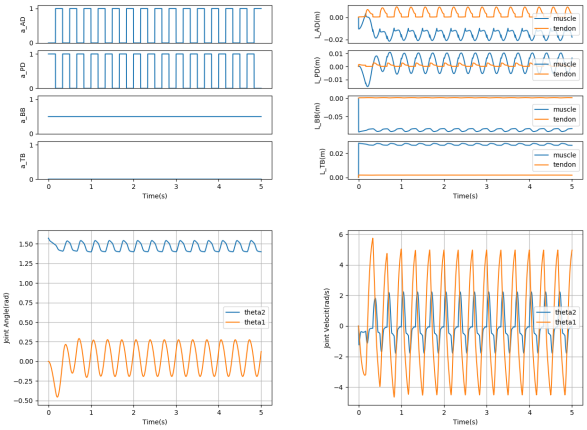


Fig. 5: Activation level(top left), change in tendon and muscle length(top right), joint angles(bottom left) and joint velocities(bottom right) when the two shoulder muscles are active and out of phase. Biceps is also activated at a constant level to stabilize the elbow joint.

Finally in this section, we demonstrate two cases when all four muscles are active(fig. 6 and 7). With different signal delays among the four muscles, we are able to generate the motion when the hand rotate counterclockwise(fig. 6) and clockwise(fig. 7). However, the complex non-linear dynamics of the physical system and the muscle force generation makes the trajectories irregular. This implies that a much more complicated activation levels are needed even for generating simple motions such as tracking a circle. In practice, we are more interested in the inverse problem where the joint level kinematics is known but the muscle states are unknown. It is apparent that incorporating reinforcement learning is necessary to make this model useful for meaningful applications.

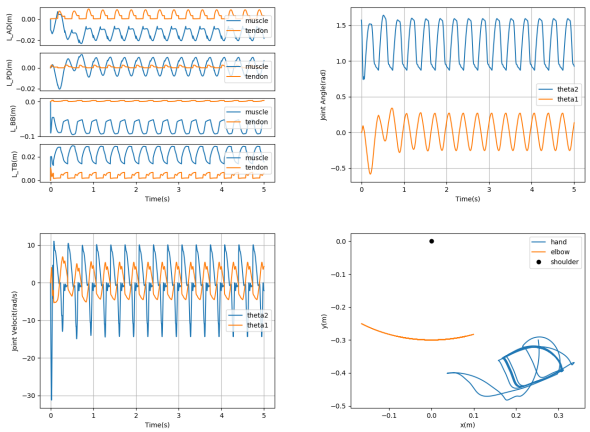


Fig. 6: Activation level(top left), change in tendon and muscle length(top right), joint angles(bottom left) and joint trajectories(bottom right). All the muscles are active. The activation levels of choice enable the hand to move counterclockwise.

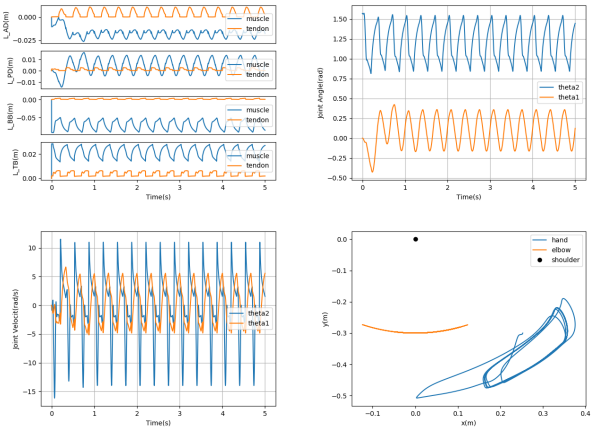


Fig. 7: Activation level(top left), change in tendon and muscle length(top right), joint angles(bottom left) and joint trajectories(bottom right). All the muscles are active. The activation levels of choice enable the hand to move clockwise.

VI. Reinforcement Learning

A Reinforcement Learning (RL) problem is formulated as a Markov Decision Process (MDP) which has the following components $\{s, a, P(s'|s,a), r(s,a), \rho\}$ where s is the current state, a is the action taken at the state, $P(s'|s,a)$ is the transition probability of the next state given the current state

and action, $r(s, a)$ is the reward for taking a certain action, and ρ is the initial state distribution from which all trajectories start. The goal of any RL algorithm is to maximize the sum of rewards along a trajectory, as described in equation 11. Since can choose the reward function, RL provides an ideal platform to test different reward functions and look at the corresponding behaviour that emerges for any task. For example, in human walking task, to test the hypothesis that symmetry and low-energy is the reason that humans walk the way we do, these factors could be enforced in the reward function and the resulting solution of the optimization problem should give us more insight [2] whether they really do influence the motion.

$$J = \mathbb{E}_{\mathbf{s}_{0:t_f}, \mathbf{a}_{0:t_f}}[\sum_{t=0}^{t_f} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)], \tag{11}$$

A. Proximal Policy Optimization

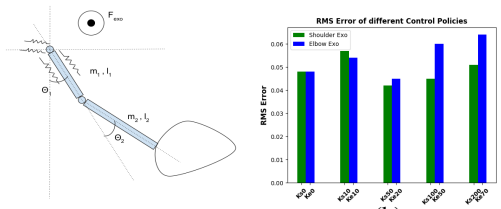
We used a policy gradient algorithm called Proximal Policy Optimization(PPO) [11]. This algorithm falls under the class of on-policy reinforcement learning methods, and the guiding principle of the algorithm is to take the maximum gradient step that will improve a parameterized policy, while penalizing Kullback-leibler divergence between the new and old policy in the objective function. This algorithm can optimize for highly non-linear function approximators like neural networks with many parameters. In our framework, the states S is defined as $\{\theta_1, \dot{\theta_1}, \theta_2, \dot{\theta_2}\}$ of the two link planar arm described in figure **??**. The actions A is the set of activations $\{a_{ad}, a_{bb}, a_{tb}, a_{pd}\}$ for Anterior Deltoids, Biceps Brachi, Triceps Brachi and Posterior Deltoid muscles respectively. The policy is represented as a neural network with two hidden layers with 128 neurons each and $tanh$ activations. The output of the network are the activations for each muscle and can take 10 equally spaced values between $0 - 1$. The reward function is defined in equation 12. The first term enforces actions that keep the end-effector as close as possible to the desired trajectory (x_t, y_t) while the second term penalizes excessive activations of the muscles. The input from the exoskeleton is modelled as a simple proportional controller, described in equation 13. The $\theta_{target}(t)$ is obtained from inverse kinematics of the arm from the given end-effector trajectory and δt is the delay of the exoskeleton controller.

$$r(s, a) = e^{-5((x-x_t)^2+(y-y_t)^2)} + w[\sum_{i=0}^{4} a_i^2] \tag{12}$$

$$\tau_{exo}(t + \delta t) = K_{exo}(\theta(t) - \theta_{target}(t)) \tag{13}$$

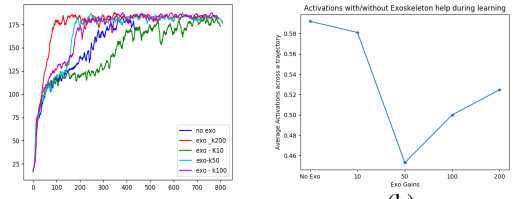
B. Experiments

In our experiments, we found that changing the activation of the muscles at each time-step makes the learning problem harder because of the non-linear muscle dynamics that has its own transient phase leading to delayed effects of actions. To make this problem simpler, we keep the activations constant for 0.050 seconds every time the neural network is queried for an action. The goal of the experiments was to collect as much information about the nature of interaction between the muscle activations that are learned from RL and the exoskeleton



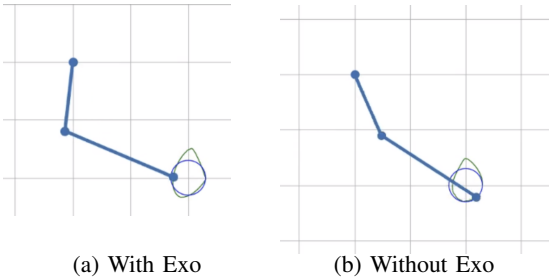
(a) (b)

Fig. 8: Figure 8(a) illustrates the Two link arm we used in our experiments and Figure 8(b) are the RMS errors of the end effector positions when exoskeletons of different gains are used. This shows that we are able to complete the task for any exoskeleton gain.



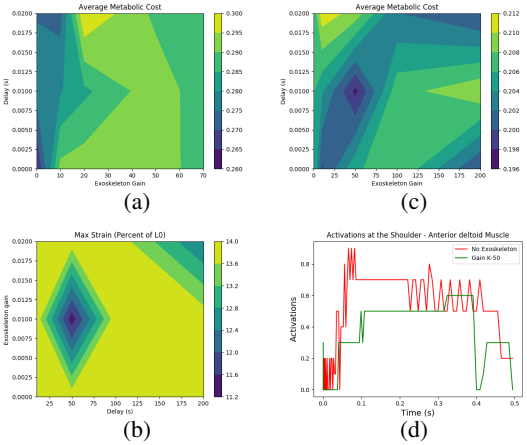
(a) (b)

Fig. 9: Figure 9(a) shows how the learning curve is affected by using the exoskeleton. Figure 9(b), is the average activation along a motion for different exoskeleton gains.



(a) With Exo (b) Without Exo

Fig. 10: Control Policies in action



(a) (c)

(b) (d)

Fig. 11: 11(a) and 11(c) illustrates the metabolic cost (W/kg) averaged across all muscles and the entire trajectory for elbow and shoulder exoskeletons. Figure 11(b) shows the max strain, and figure 11(d) shows the activations, output of the neural network policy, of the anterior Deltoid when exoskeleton of gain 0 and 50 are used.

input. We vary two important parameters in the exoskeleton controller, first is the exoskeleton gain, and second is the exoskeleton delay. 5 different exoskeleton gains were chosen between 0 and a maximum value at which the exoskeleton alone can produce enough torque to track a joint trajectory. The delay values were chosen based on the reasonable values that we can expect from a real hardware.

VII. Results Discussion

In our results we look for answers to the following questions

1) Is the Two-link muscle actuated upper arm dynamics model that we built reliable?
2) Can we learn control policies with the model for fast dynamic end-effector trajectory tracking?
3) How does the exoskeleton affect the muscle activations that we learn? In other words, what kind of activations emerge from the learning algorithm as a result of using exoskeleton.
4) How does exoskeleton affect performance parameters like average power consumption and muscle strain?
5) In this task, is a shoulder exoskeleton beneficial or an elbow exoskeleton?

To verify the platform we built, we took a two step approach: First, we verified that our hopper model corresponds well to the simulink model. Second, test our two link planar arm model by subjecting it to a variety of input signals and recording the behaviour. In section III we showed that our hopper model comparable to simulink version. In section IV and V we showed that our arm model also behaves as expected to different input signals and shows no signs of instability.

To answer our second question, our results are illustrated in figure 10, which shows the motion generated by the trained control policies. Furthermore, we compare the resulting RMS error of the end-effector trajectory are shown in figure 8(b) and the errors are comparable for all exoskeleton gains implying the algorithm is able to learn policies with or without the exoskeleton. This is important because, to compare the effects of the exoskeleton on the performance, we need the control policies to perform the same task, so that we have a solid basis for comparing the two results.

Figure 9(b) provides an interesting insight into how the exoskeleton influences the activations of the muscle, the plot tells us that the average activations of the shoulder Anterior Deltoid muscle during the execution of motion is least when an exoskeleton of gain 50 is used. As muscle activations alone are not sufficient to paint the whole picture, we looked at how parameters such as exoskeleton delay and gain affects the average metabolic cost and the max strain of the muscles. Figure 11(c) and 11(a) show the metabolic cost averaged across all muscles and the entire motion, the values were computed using the equations reported in Alexander et al. [12]. These results indicate that, for this task, it is better to use a shoulder exoskeleton than an elbow exoskeleton because the lowest power consumption is lesser ($.26$ W/kg for elbow Vs 0.19 W/kg for shoulder). In addition to this, as illustrated in figure 11(b) it is also interesting to see that the max strain when shoulder exoskeleton is used is around 14% (well below injury risk, which is expected because this motion amounts to just moving our arms fast without any load), and the least max strain is in fact at the optimum point of the metabolic cost. Another interesting plot is figure 11(d), which compares the activations of the anterior Deltoid muscles at exoskeleton gains of 0 and 50 with delay of $10ms$ (optimum), and the activations are also comparatively lesser. These findings points towards the possibility that at the certain exoskeleton gain (exoskeleton gain, mathematically, amounts to adding some stiffness to the dynamics), the system is leveraging the exoskeleton input and its natural muscle properties to complete the task.

We have presented some interesting results, which by themselves are not conclusive, but point towards the possibility that following this line of research might prove to be a fruitful endeavour. In future work, we want to highlight these following modifications to the current platform that can provide more insightful results. First, is defining a better reward function for the RL algorithm to work with, for example, we could include some notion of expected exoskeleton behaviour which would result in activations that are closer to when a real human is using the device. Second, is using a better exoskeleton controller itself. Also, there is no reason to limit ourselves to a few upper arm motions, there are many motion categories where this platform for could be used for. For example, to analyze upper limb motions that are common in sports and exercise can be extremely beneficial.

VIII. Acknowledgements

We thank the instructors for helpful discussions during this project and a wonderful class!

IX. Supplmental Material

The Python model developed in this project can be found at: https://github.com/HungtangK/MuscleDynamics.

References

[1] L.-w. Liu, A.-h. Wang, S.-l. Hwang, Y.-h. Lee, A.-h. Wang, S.-l. Hwang, and Y.-h. Lee, "Prevalence and risk factors of subjective musculoskeletal symptoms among cooks in Taiwan," *Journal of the Chinese Institute of Industrial Engineers*, vol. 0669, 2011.

[2] W. Yu, G. Turk, and C. K. Liu, "Learning Symmetric and Low-energy Locomotion," *ACM Transactions on Graphics*, vol. 37, no. 4, p. 12, 2018.

[3] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–14, 2018.

[4] X. Peng, G. Berseth, and B. Columbia, "DeepLoco : Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning," vol. 36, no. 4, 2017.

[5] Ł. Kidziński, S. P. Mohanty, C. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, S. Kolesnikov, S. Plis, Z. Chen, Z. Zhang, J. Chen, J. Shi, Z. Zheng, C. Yuan, Z. Lin, H. Michalewski, P. Miłoś, B. Osiński, A. Melnik, M. Schilling, H. Ritter, S. Carroll, J. Hicks, S. Levine, M. Salathé, and S. Delp, "Learning to Run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments," pp. 1–27, 2018.

[6] L. O. F. Muscle, "The heat of shortening and the dynamic constants of muscle," pp. 136–195, 1938.

[7] D. Song, N. Lan, G. E. Loeb, and J. Gordon, "Model-based sensorimotor integration for multi-joint control: Development of a virtual arm model," *Annals of Biomedical Engineering*, vol. 36, no. 6, pp. 1033–1048, 2008.

[8] O. F. Human and O. F. J. Angles, "TECHNICAL NOTE," vol. 29, no. 10, 1996.

[9] K. R. S. Holzbaur, W. M. Murray, and S. L. Delp, "A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control," *Annals of Biomedical Engineering*, vol. 33, no. 6, pp. 829–840, 2005.
[10] R. M. Murray, *A mathematical introduction to robotic manipulation*. CRC press, 2017.
[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," pp. 1–12, 2017.
[12] L. Ls, "Optimum Muscle Design for Oscillatory Movements," no. August 1996, pp. 253–259, 1997.