System Identification and Control of Anatomically Accurate Biomechanical Human Limb Models

Willman, Jaxton
Saxena, Shreya
University of Florida May 20, 2022

Abstract*— Developing neuromusculoskeletal models including neural activation, muscle contraction, and skeletal dynamics for targeted sensorimotor control is critical for (a) understanding the control strategies implemented by the brain to drive movements, and (b) improving assistive technologies for motor diseases and disorders, including neurodegenerative diseases that affect movements. This work aims to develop a brain-inspired closed-loop controller for an anatomically and physiologically accurate arm model. We consider a 6-muscle, 2-joint human arm model created in the OpenSim software. We use OpenSim's computed muscle control tool to generate the corresponding muscle excitations from the states of the elbow and shoulder joints from a generated set of 5000 unique motion files. With the muscle excitations as the input and states of the joints as the output, we estimate a 10th order linear dynamical system to approximate the musculoskeletal dynamics, reaching a high accuracy for the elbow angle across the training data, but performing poorly on the 500 held-out movements for system validation. We design a proportional-integrative-derivative (PID) controller to each muscle input to drive the estimated linear dynamical system towards a reference trajectory. The best step response for the system had a rise time of the step response was found to be 0.009 seconds, the settling time was 0.047 seconds, and the overshoot of the step response was 18.7327%. We then model the neuromuscular control of a human arm by placing these tuned controllers directly in closed loop with the OpenSim musculoskeletal model to achieve desired movements. The tracking performance is oscillatory but stable and results in the anatomical arm model moving as desired. We compare the closed loop results of the OpenSim and linearized models. This framework can be used in the study of assistive neurotechnologies for combatting the effects of neurodegenerative diseases, for example, by designing compensators to reduce the effect of weaker neural signals to the muscles.

***Index Terms*— Computed Muscle Control, Control Theory, Forward Dynamics, Linear Dynamical System, MATLAB, Neuromuscular Control, OpenSim, PID Control**

I. Introduction

DEVELOPING neuromusculoskeletal models including neural activation, muscle contraction and skeletal dynamics for targeted sensorimotor control is critical for (a) understanding the control strategies implemented by the brain to drive movements, and (b) improving assistive technologies for motor diseases and disorders, including neurodegenerative diseases that affect movements. This work takes a controls theory perspective on sensorimotor control. From previous research, we understand that the computational role of the motor regions of the brain is to drive a complex, multi-dimensional arm to achieve a specific goal, for example, reaching from point A to point B. Thus, we want to understand the neural signals better by understanding how they control our limbs. This can also help us understand how to fix things when they break down, in the case of motor diseases and disorders. We have two important goals for our work, the first is to understand the control strategies implemented by the brain to drive movements, and our second goal is to improve assistive technologies for motor diseases and disorders, including neurodegenerative diseases that affect movements. To this extent, we are developing a brain-inspired closed-loop controller for an anatomically and physiologically accurate arm model.

In previous work, Saxena et al. looked at a neurally-inspired controller to drive a musculoskeletal model. This framework was used to design a compensator to reduce the effect of weaker neural signals to the muscles [1]. However, their arm model was not anatomically accurate. Saxena et al. used a single joint model under the class of equilibrium-point models for motor control from McIntyre and Bizzi (1993) thus the model was linear and not multidimensional [1], [2]. The OpenSim model allows this framework to be developed under biomechanically accurate conditions which will hopefully allow more meaningful conclusions to be drawn.

Previous research in this field details the internal representation of the sensorimotor loop within the CNS, as shown in this figure by Wolpert et al. The analysis of behavior has led us to an understanding that driving a limb model can be performed by a combination of inverse and forward control models [3]. However, the neural basis of these models is not clear [3]. As research focusing on motor control, our work explores a possible computational model of the sensorimotor loop.

Wellington Cassio Pinheiro et al. wanted to identify approximate linear dynamics of an OpenSim arm model. However, their work focused on a reduced version of the Arm26 model with only the biceps brachii long head muscle [4] thus it was not anatomically accurate. The authors used a recursive instrumental-variable method for system identification of the musculoskeletal dynamics and designed a stable system with PID controllers that allowed the OpenSim

model to track the reference with a biophysically accurate rise time and zero steady-state error [4]. This is promising as this work seeks to develop robust controllers for the full OpenSim arm model.

II. Methods

OpenSim Arm Model

For our musculoskeletal dynamics, we consider a 6-muscle, 2-joint human arm model named Arm26 which was created in the OpenSim software. OpenSim, created by Scott Delp et al., is a wonderful software that users can develop models of musculoskeletal structures and create dynamic simulations of movement [5], [6]. An OpenSim model is built from bodies, joints, forces, markers, constraints, contact geometry, and controllers to provide a comprehensive simulation of the musculoskeletal dynamics. Additionally, there is a reserve actuator for each joint which adds torque about each joint that can make up for insufficient muscle strength during a simulation. The use of these reserve actuators is penalized during the simulation and requires a threshold to be hit before they add their torque. The inclusion of these reserve actuators makes the simulation less biophysically relatable, however these reserve actuators are needed as often the skeletal structures are incomplete. Our Arm26 model does not include the muscles, skin, organs, bones, and other bodies of the entire human body but rather stops at the shoulder joint. Our simulation lacks neck muscles, intercostal muscles, abdominal and oblique muscles and all the rigid bodies typically found in a complete human to help exert force on the arm. So, the reserve actuators can be seen to compensate for the lack of whole-body force dynamics. The Arm26 model includes the shoulder and elbow joints and the triceps brachii (long, lateral, and medial heads), biceps brachii (long and short heads), and the brachialis muscle. OpenSim tracks the joint angles, joint angular velocities, muscle excitations, and muscle fiber lengths.

Motion Files

We created a script in MATLAB to pseudo-randomly define motions (sets of desired kinematics) and write each motion to a file to run OpenSim's CMC tool on. Every motion was 1.1 seconds long with a sampling rate of 0.001 seconds. The first 0.1 seconds act as an initial condition to allow analysis of the muscle excitations from a baseline. For the initial condition, the elbow joint remained motionless at 30 degrees with zero velocity. For all motions, the shoulder joint was kept motionless for the entire duration. For the main 1 second of motion, the time range was divided into a random number of features (between 1 and 4) for the movement which were either zero motion or linear motion. OpenSim's CMC tool doesn't give an exact match of the idealized motion file provided to it, but rather a very close approximation that is feasible provided its scripted dynamics [7]. These motions were generated with biological plausibility in mind to mitigate this approximation. The velocity was never allowed to go above 400 degrees per second, the states were linear with time, and to avoid undifferentiable transitions, the entire signal was smoothed at the end with a moving window of 40 timesteps or 0.04 seconds. Before smoothing, OpenSim's CMC tool crashed when trying to compute the motions. Following this approach, we pseudo-randomly generate 5000 motion files for the training data set and 500 motion files for the held-out validation data set.

OpenSim Computed Muscle Control

OpenSim's CMC tool computes the muscle excitations needed to achieve a specified motion. In addition, it internally conducts a forward dynamics simulation which allows it to return the associated states as well. Our goal is to identify a linearized version of the OpenSim model with the muscle excitations as the input and the kinematics as the output. So, this one tool provides both the input and output of the OpenSim arm model in one operation which is convenient. The kinematics the forward dynamics simulation returns are not the exact ideal states, but the closest equivalent approximation of them required by the scripted musculoskeletal dynamics. Thus, we compared the ideal generated ideal motions and their approximated equivalents after running OpenSim's computed muscle control tool. Running the CMC tool on a single motion file required approximately 30 seconds of computational time for it to compute the 1.1 seconds of motion. We sought faster computational ability at our scale to allow us to debug and iterate so we distributed the workload across more than 60 AMD EPYC 75F3 Milan 3.0 GHz cores in the University of Florida's supercomputer HiPerGator which parallelized the process and offered performance increases [8].

System Identification

OpenSim's dynamics are nonlinear in nature and are not able to be accessed in any simple manner. We approximate a linear dynamical system of the OpenSim model. We use MATLAB's system identification toolbox to accomplish the identification of the dynamics of the OpenSim plant. We begin by creating an iddata file in MATLAB that specifies each motion as an experiment with the muscle excitations (as well as the shoulder and elbow joint reserve actuators) as the inputs to the system and the position and velocity of the elbow joint as the outputs of the system [9]. Thus, our approximated system will have 8 inputs and 2 outputs. We use MATLAB's implementation of N4SID to estimate a state-space model of the form (1), (2) discretized with a sampling rate, Ts, of 0.001 seconds [10].

$$\dot{x}(t) = Ax(t) + Bu(t) + Ke(t) \quad (1)$$

$$y(t) = Cx(t) + Du(t) + e(t) \quad (2)$$

We explore what order of system models our data best by calculating a state-space model for each order between 2 and 15. We observed how well each state-space model performed on the training and held-out validation data. MATLAB's compare function accomplished this system comparison easily and provided NRMSE values to gauge how well the systems performed. All calculations were carried out on UF's HiPerGator clusters to decrease computational time and parallelize the tasks to improve design iteration and script debugging.

Controller Tuning

Our goal is to design a proportional-integrative-derivative (PID) controller to each muscle input to drive the estimated linear dynamical system towards a reference input. Fig. 1 shows the desired closed loop control structure. With MATLAB's control systems toolbox, we created 8 tunable discretized PID controllers C and placed them in closed loop with our chosen approximated linear dynamical system as the plant G with a reference signal, r, and output signal, y, as the angular position and velocity of the elbow joint. A discretized tunable decoupling gain matrix D was added to the control structure to track with minimal crosstalk [11]. The controllers and decoupling gain matrix were tuned with MATLAB's looptune function which tunes the feedback loop to ensure: (a) the gain crossover for each loop falls in the frequency interval ωc, (b) integral action at frequencies below ωc, and (c) adequate stability margins and gain roll-off at frequencies above ωc [12]. The looptune command requires a crossover frequency interval ωc. The range can be from greater than 0 to the Nyquist frequency. Each system order was tested with a crossover frequency range of [0.01, 500] to determine if a stable system could be identified. The lower bound of the crossover frequency interval, ωc_{min}, was adjusted until the best system response was found for each stable system identified.

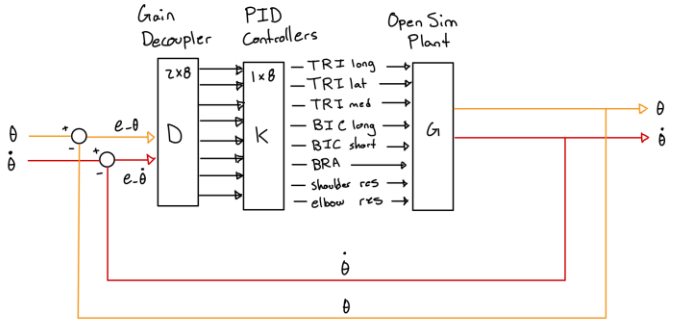

Fig. 1: Closed loop control structure. The position and velocity of the elbow joint are the inputs and outputs of the system. A PID controller for each muscle drives the input to make the system track the provided reference trajectory. A 2x8 gain decoupler minimizes crosstalk of the input signals when the plant is tracking the reference.

Control Loop Implementation

We modeled the neuromuscular control of a human arm by placing these tuned controllers directly in closed loop with the OpenSim musculoskeletal model to achieve desired movements. The muscle excitations, u(t), were calculated programmatically with (3)-(8) and the Kp, Ki, and Kd values of the tuned PID controllers [13]. OpenSim's forward dynamics tool allowed the states of the OpenSim arm model to be computed from the muscle excitations using OpenSim's musculoskeletal dynamics. The muscle excitations were computed for each time step, binned in batches, and then were fed to the forward dynamics tool to obtain the states in batches. We chose a batch size based on the settling time of the position step function to ensure little unstable behavior in the output. Finally, we compared the output of the OpenSim model with the output of our linearized approximation of the OpenSim model to determine how well the controllers performed. To make control of the system interactive and visually appealing, an application was developed using MATLAB's App Designer tool [14]. The GUI allows the user to select a desired reference signal for automatic control of the simulation or manually control the reference signal with sliders. Additionally, the GUI plots the reference signal and output states of the OpenSim model and linearized approximation for both the position and velocity.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{3}$$

$$e(t) = e[k] \tag{4}$$

$$e_i(t) = \int_0^t e(t)dt \tag{5}$$

$$e_i(t) = e_i[k] = e_i[k-1] + T_s e[k-1] \tag{6}$$

$$e_d(t) = \frac{d}{dt} e_f(t) \tag{7}$$

$$e_d(t) = e_d[k] = \frac{e_f[k-1] - e_f[k-1]}{T_s} \tag{8}$$

III. Results

Motion Files

We generated a set of 5000 motion files to get the muscle excitations and kinematics for the training data set. For the validation data set, we generated 500 motion files. The kinematics for a random selection of 500 of the generated motion files can be seen in Fig. 2. The initial condition of 30 degrees for the first 0.1 seconds can be seen before the kinematics deviate from there. The kinematics were smoothed with a moving window of 40 timesteps, or 0.04 seconds. The top image is the ideal motion, and the bottom image is the approximated motion from OpenSim's dynamics. OpenSim's computed muscle control, or CMC, tool approximates the ideal kinematics fed in as it can't always achieve it, thus it generates an approximation of the kinematics.

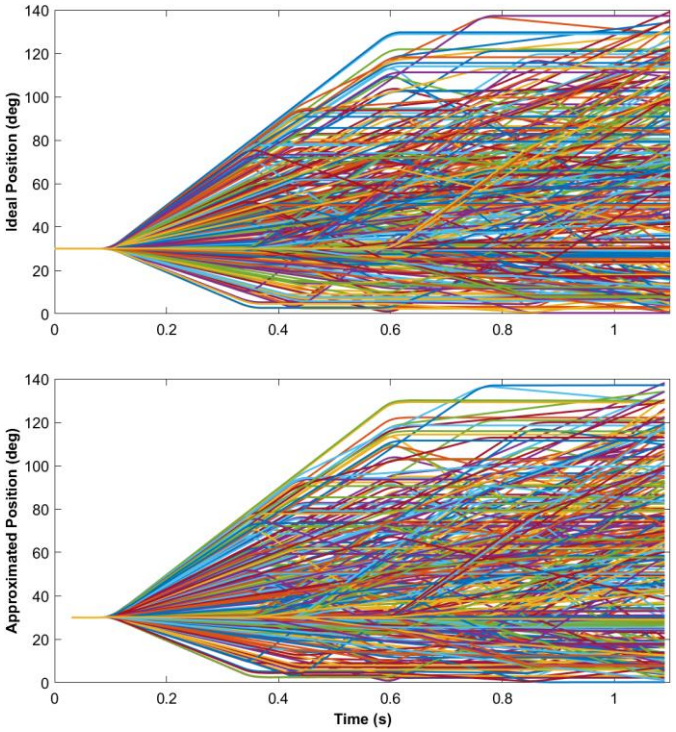

Fig. 2: A sample of 500 generated idealized motions and their approximated equivalents after running OpenSim's computed muscle control tool which runs a forward dynamics simulation internally to obtain these states from the computed muscle excitations.

System Identification

With the muscle excitations as the input and kinematics from the internal forward dynamics simulation as the output, we approximated a linear dynamical system of the OpenSim model with MATLAB's N4SID. We explored how the system order affected the ability of the system to represent the OpenSim model. We explored systems between order 2 and 15. We observed how well each system fit the training data set and the validation data set to gauge the system performance by analyzing both the NRMSE and MSE values of the systems as can be seen in Table I. We wanted to measure how well each system order performed on the training and validation data, so we simulated the response of the training and validation data and compared it to the ideal motion for each system order. The comparison of the elbow angular position and angular velocity for the training data can be seen in Fig. 2. The NRMSE values are in the 90% range for the most part. Fig. 3 displays the comparison against the held-out validation data. The NRMSE values are almost all negative and large. The system did not perform well on the held-out data. Some systems had larger oscillatory components.

Based on the NRMSE values, the system of order 10 was chosen as it represents the OpenSim model the best and reached a high training accuracy of 96.67% for the elbow joint angular position. However, the system did not perform well on the held-out validation data set of 500 movements. Table II displays the initial conditions of the 10th order system as estimated by the N4SID algorithm.

TABLE I
COMPARISON OF SYSTEM ORDERS FOR POSITION TRACKING

ORDER	TRAINING DATA NRMSE	VALIDATION DATA NRMSE
2	87.11%	-43.25%
3	69.57 %	-218.0%
4	85.18%	-18.74%
6	87.23%	-178.9%
7	90.15%	-232.1%
8	97.99%	-66.00%
9	90.17%	1.531%
10	**96.67%**	**-56.08%**
11	92.13%	-64.49%
12	95.63%	-28.28%
13	96.42%	-106.6%
14	94.26%	-32.85%
15	84.38%	-296.7%

TABLE II
10TH ORDER STATE SPACE SYSTEM INITIAL CONDITIONS

INPUT	X0 POSITION	X0 VELOCITY
1	0.1573	0.1573
2	-0.0622	-0.0622
3	-0.0535	-0.0528
4	-2.1954	-2.1953
5	2.6616	2.6628
6	-0.7143	-0.7103
7	0.4091	0.4110
8	-0.3411	-0.3422

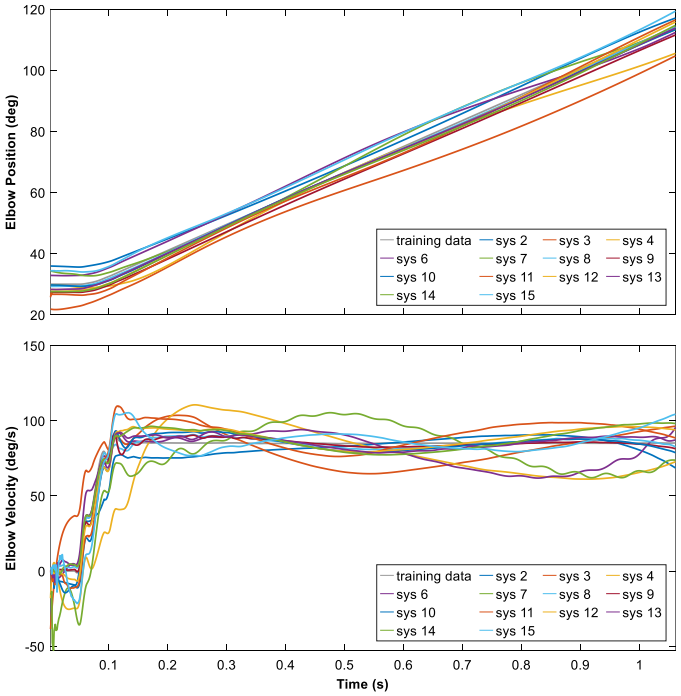

Fig. 3: Simulated response of the training data for each system order compared with the ideal response. This is for the first motion of the training data set. The system generally tracks the elbow angle well but has more oscillatory behavior when tracking the angular velocity of the elbow.

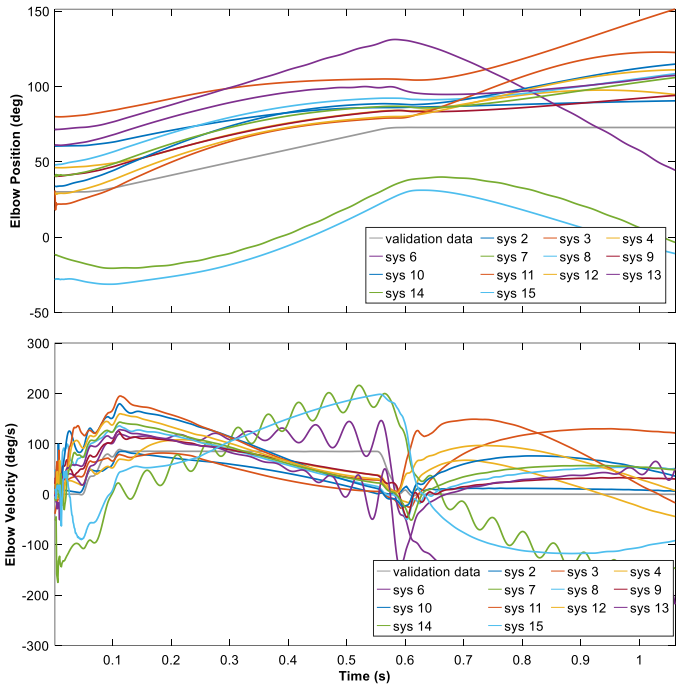

Fig. 4: Simulated response of the validation data for each system order compared with the ideal response. This is for the first motion of the validation data set. The system does not perform well on held-out data yet. It does not track the angle of the elbow well. And it adds low-frequency oscillatory behavior when tracking the angular velocity of the elbow.

Controller Tuning

Only two system orders were found to be stable: 4 and 10. The large number of poles and zeros on the unstable side of pole-zero map, which can be seen in Fig. 5 for the of the 10th order system, highlights why so many of the other systems were unstable. The controllers were continually tuned for different crossover frequencies until the best step response was found. Table III documents the crossover frequency interval testing for a system of order 10. The best step response was determined to be for a ωc of [50, 500], the rise time of the step response was found to be 0.009 seconds, the settling time was 0.047 seconds, and the overshoot of the step response was 18.7327%. The step response of the 10th order system can be seen in Fig. 6. For comparison, the best step response of the other stable system found, the system of order 4 was determined to be for a ωc of [13, 500], the rise time of the step response was found to be 0.019 seconds, the settling time was 0.123 seconds, and the overshoot of the step response was 13.4016%. The resulting proportional, integrative, and derivative gain values of the controller and the gain decoupler values are recorded in Table IV for the 8 inputs of the 10th order system. Fig. 7 displays the loop gains and sensitivity of the tuned system and display the guaranteed phase margins for both the inputs and outputs of the plant.

TABLE III
10TH ORDER SYSTEM CONTROLLER GAIN VALUES

WC_{MIN}	WC_{MAX}	RISE TIME (S)	SETTLING TIME (S)	OVERSHOOT (%)
0.01	500	4.024	5.360	3.1434
0.1	500	2.316	11.893	8.8182
1	500	0.026	0.101	9.8627
5	500	0.045	0.324	9.0540
9	500	0.016	0.096	19.7725
10	500	0.299	1.758	24.6243
11	500	0.016	0.099	22.9199
12	500	0.157	1.071	11.5211
13	500	0.016	0.095	21.6854
14	500	0.015	0.097	24.9932
15	500	0.015	0.096	24.2322
50	500	0.009	0.047	18.7327
100	500	0.011	0.057	23.6878

TABLE IV
10TH ORDER SYSTEM GAIN VALUES

	CONTROLLER			GAIN DECOUPLER	
INPUT	KP	KI	KD	POSITION	VELOCITY
1	-0.018	-0.311	0	4.432	-0.037
2	0.239	1.366	0	-2.241	0.002
3	0.000	-1.580	0	-5.639	1.504
4	-0.526	4.677	0	-131.243	0.769
5	0.007	0.650	0	2.809	0.026
6	-0.045	-1.982	0	-0.792	0.506
7	-0.099	-1.235	0	-10.032	-0.101
8	-1.120	0.553	0	-0.288	0.019

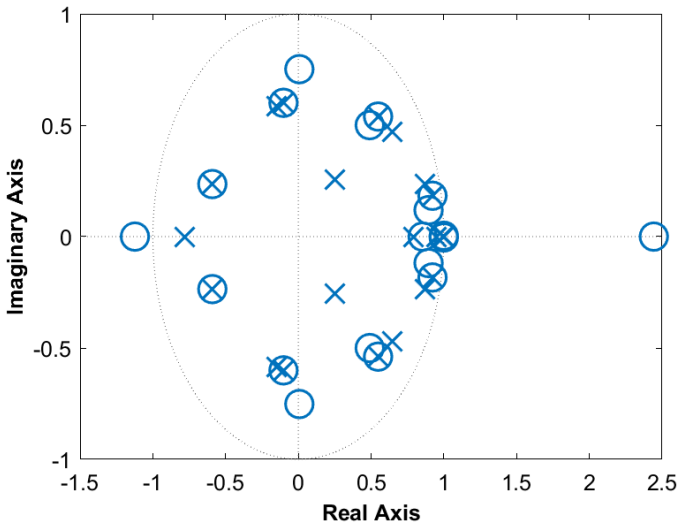

Fig. 5: Pole-zero map of the 10th order system. Most of the poles and zeros lie on the unstable side of the real axis.

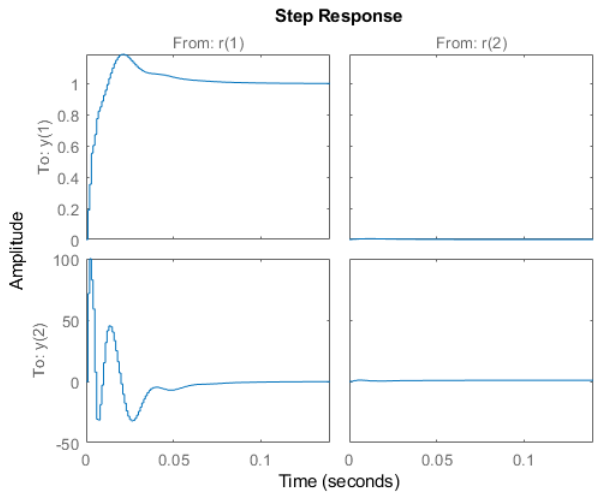

Fig. 6: Time-domain step response of the 10th order system tuned so that the open-loop gain crosses 0 dB within the range of between 50 rad/s and 500 rad/s.

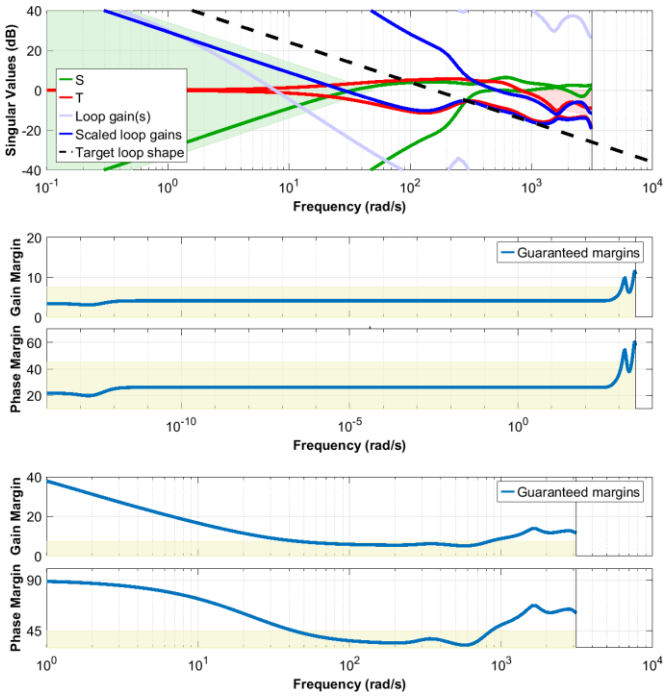

Fig. 7: Frequency-domain response of the 10th order system tuned so that the open-loop gain crosses 0 dB within the range of between 50 and 500 rad/s. The first graph displays the loop gains, the sensitivity, and the sensitivity complement of the closed loop. The second and third images show the guaranteed phase margins across all frequencies for the plant inputs and outputs.

Control Loop

The GUI of the application developed for ease of use of the control loop can be seen in Fig. 8. This allows the user to easily select trajectories or manually control the reference signal. The tracking performance of the control loop was tested for trajectory 2 in the automatic selector. The effect of batch size was tested by comparing batch sizes of 10, 20, and 200 timesteps which translates to 0.01, 0.02, and 0.2 seconds respectively. Fig 9-11 shows the tracking performance of the control loop for the different batch sizes. The batch size of 200 had more oscillatory movement as it could adjust less often. When testing with a batch size of 10, the velocity peaks were less than the batch size of 20. Step functions were tested by inputting a constant reference signal other than 30 which is the initial state for the system. Magnitudes of 60 and 130 degrees were tested with batch sizes of 20 timesteps. Fig. 12 shows the 60-degree step function and Fig. 13 shows the 130-degree step function.

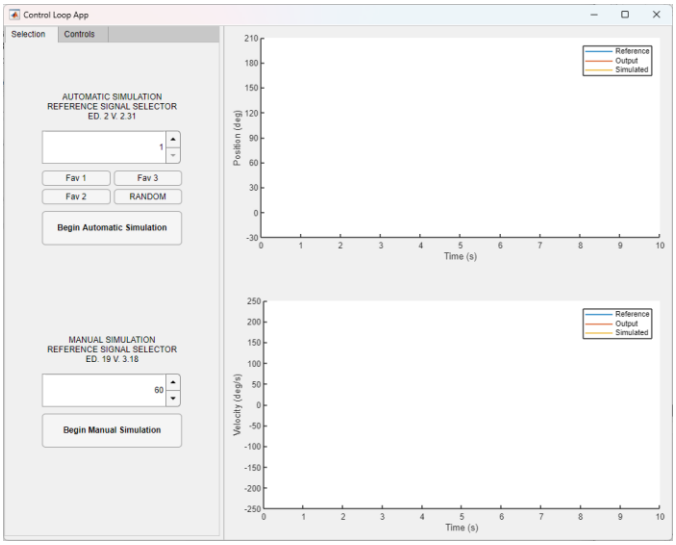

Fig. 8: Application start screen. An automatic or manual simulation of the Arm26 model can be selected.

[INSERT BIG TRACKING FIGURE HERE AT TOP OF PAGE FROM MARGIN TO MARGIN]

IV. Discussion

We endeavor to understand the sensorimotor control loop by analyzing how the neuromusculoskeletal dynamics may function. We developed a model including neural activation, muscle contraction and skeletal dynamics for targeted sensorimotor control. We approximated the musculoskeletal control of an anatomically accurate human arm model in OpenSim as a 10th order linear dynamical system. This system performed extremely well on the set of 5000 motions of training data but is not yet performing well on held-out data as can be seen by the distribution in Fig. 4 and the NRMSE values in Table I. The system is simply not capturing the dynamics of the actual OpenSim arm model well. We know that the OpenSim dynamics are nonlinear in nature, so our linear dynamical system approximation inherently loses the ability to capture the nonlinear dynamics. We speculate that the system was not trained on a large enough data set. Even though the sample of 500 motions which can be seen in Fig. 2 appears to explore the parametrized space well, perhaps the motions are far more complex than what was explored in the training data set. These motions neglected the shoulder and thus heavily focused on exciting the bicep brachii long head muscle which means the

system may not have been able to identify all the dynamics of the other muscles. Additionally, the training signals were 10 times shorter than the signals we desired to work with. Having longer training signals could elicit more complex behavior from the OpenSim model. In future work we wish to retrain our system on a larger data set with more complex movement signals and longer signals.

We identified controllers that were biophysically plausible to drive the neural signals of the muscles. From Table III, the rise time of the 10^{th} order system was 0.009 seconds which was 52.63% faster than the 4^{th} order system. The settling time of the 10^{th} order system was 0.047 seconds which was 61.79% faster than the 4^{th} order system. In exchange for the fast rise and settling times, the overshoot of the 10^{th} order system was 18.7327% which was 39.78% greater than the 4^{th} order system. The rise time and settling time were excellent and blazing fast for neuromuscular control. To get an overshoot less than 10% would require the rise time and settling time to be approximately 5 times greater. Furthermore, an overshoot less than 5% elicits a rise time and settling time above 4 seconds. Ultimately, this overshoot coupled with the blazing fast rise and settling times is a good tradeoff for neuromuscular control. The muscles can ideally respond with new signals before overshoot occurs. This allows the muscles to be incredibly responsive. This is biophysically relatable as the rise time and settling time is indicative of human muscle control.

The large number of poles and zeros on the unstable side of pole-zero map, which can be seen in Fig. 5 for the of the 10th order system, highlights why so many of the other systems were unstable. Only a few systems were able to become stable with feedback: 4 and 10.

As can be seen in Table IV, the gains of the PID controllers displayed relative relationships where the biceps brachii long head muscle contributed the most to the tracking, followed by the brachialis muscle, followed by the triceps muscle. The integral terms had the most weight which could be due to the incremental motor unit recruitment of muscles developed [15]. The gains of the gain decoupler displayed relative relationships that once again favored the biceps brachii long head muscle as it multiplied the position by -131, then next up is the elbow joint reserve actuator at -10, and then the triceps muscles follow at less than -5 gain. The gains for the shoulder reserve actuator barely contributes to tracking which is good. And the gains for velocity tracking are near 0.

Looking at the frequency-domain response of the closed loop control system in Fig. 7, the phase margins are rather low. The phase margins are important measures of stability. The system displays borderline stability which means the system may quickly become unstable due to process changes. The system should not experience any process changes as the OpenSim code is deterministic. Repeated trials will give the same result. So, the small margins of stability are fine. The sensitivity graph in Fig. 7 displays that the system is most sensitive to low frequency changes. At about 100 rad/s, the system is not as sensitive to changes. This mimics the behavior of the input and output as the changes are generally of lower frequency instead of much higher frequency movements.

V. Conclusion

A 10^{th} order linear dynamical system was identified as an approximation of the nonlinear complex dynamics of the anatomically correct human arm made in the OpenSim software. A PID controller was designed to each muscle input to drive the estimated linear dynamical system towards a reference input. The step response of the system was blazing fast allowing for the plant to track the reference with good quality. The tracking was a bit oscillatory in the shoulder joint, but the arm model was able to move to the desired positions with time. Thus, it is possible to model the nonlinear musculoskeletal dynamics of the human arm with linear time-invariant systems and produce tracking. Further work will be refining the tracking and developing controllers to augment the muscle signals in the case of neural decay.

VI. Acknowledgement

The authors would like to thank the University Scholars Program and the University of Florida for providing funding. A big thank you to my research advisor, Dr. Shreya Saxena, for her insightful contributions, guidance, and reviews throughout the year and a half that it took to get to this point. Your kindness has helped me to grow as a researcher! Thank you to Muhammad Noman Almani in the Saxena Lab for Neural Control for his support in control theory and for reviewing my paper. Thank you to my parents for their love and support!

VII. Appendix

The following GitHub link contains the repository with all my code. May this be helpful to anyone following my path and wanting to integrate OpenSim software into MATLAB scripting. The commit history includes documentation of updates to the code and my thoughts as I moved through the process. Feel free to email me at my email: jaxtonwillman@gmail.com if you have any inquiries about the code or process. There is little documentation for the process so I hope this code can serve as a guide.
https://github.com/saxenalab-neuro/controlled-movement

References

[1] S. Saxena, S. Sarma and M. Dahleh, "Performance Limitations in Sensorimotor Control: Trade-Offs Between Neural Computation and Accuracy in Tracking Fast Movements", Neural Computation, vol. 32, no. 5, pp. 865-886, 2020. Available: 10.1162/neco_a_01272.

[2] J. McIntyre and E. Bizzi, "Servo Hypotheses for the Biological Control of Movement", Journal of Motor Behavior, vol. 25, no. 3, pp. 193-202, 1993. Available: 10.1080/00222895.1993.9942049.

[3] D. Wolpert and Z. Ghahramani, "Computational principles of movement neuroscience", Nature Neuroscience, vol. 3, no. 11, pp. 1212-1217, 2000. Available: 10.1038/81497.

[4] W. Pinheiro, M. de Castro and L. Menegaldo, "Design of MATLAB/OpenSim Elbow Flexion Angular Setpoint Controller", XXVI Brazilian Congress on Biomedical Engineering, pp. 167-174, 2019. Available 10.1007/978-981-13-2119-1_26.

[5] S. Delp et al., "OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement", IEEE Transactions on Biomedical Engineering, vol. 54, no. 11, pp. 1940-1950, 2007. Available: 10.1109/tbme.2007.901024.
[6] A. Seth et al., "OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement", PLOS Computational Biology, vol. 14, no. 7, p. e1006223, 2018. Available: 10.1371/journal.pcbi.1006223.
[7] "Getting Started with CMC", OpenSim Documentation, 2022. [Online]. Available: https://simtk-confluence.stanford.edu:8443/display/OpenSim/Getting+Started+with+CMC.
[8] "HiPerGator", UF Research Computing, 2022. [Online]. Available: https://www.rc.ufl.edu/about/hipergator/.
[9] "iddata", MathWorks, 2022. [Online]. Available: https://www.mathworks.com/help/ident/ref/iddata.html.
[10] "n4sid", MathWorks, 2022. [Online]. Available: https://www.mathworks.com/help/ident/ref/n4sid.html.
[11] "Tune MIMO Control System for Specified Bandwidth", MathWorks, 2022. [Online]. Available: https://www.mathworks.com/help/control/ug/tune-a-mimo-control-system-for-a-specified-bandwidth.html.
[12] "looptune", MathWorks, 2022. [Online]. Available: https://www.mathworks.com/help/control/ref/lti.looptune.html.
[13] P. P, "PID Controllers", Peter's Pages, 2022. [Online]. Available: https://tttapa.github.io/Pages/Arduino/Control-Theory/Motor-Fader/PID-Controllers.html.
[14] "MATLAB App Designer", MathWorks, 2022. [Online]. Available: https://www.mathworks.com/products/matlab/app-designer.html.
[15] D. Thelen, "Adjustment of Muscle Mechanics Model Parameters to Simulate Dynamic Contractions in Older Adults", Journal of Biomechanical Engineering, vol. 125, no. 1, pp. 70-77, 2003. Available: 10.1115/1.1531112.