Abstract

Internuclear Ophthalmoplegia (INO) is a condition commonly associated with multiple sclerosis, brain trauma, and viral infections affecting primarily older individuals, resulting in impaired eye motor control [1], [2]. INO is often characterized by impaired adduction in the ipsilateral eye and nystagmus in the abducting eye [1], [2]. Most cases of INO are treatable depending on the underlying cause, but treatment relies on accurate clinical diagnosis from physicians [1], [2]. While research exists on the effect of INO on saccadic eye movements, clinical diagnosis tools have yet to be developed to assist physicians in diagnosing INO. As such, this paper aims to build a model to determine neural input from eye position to predict the likelihood of INO in a patient. Simulations were performed with the model to compare healthy and INO abductions, delays in INO abductions, and delay-removed healthy and INO comparisons. The model was validated through patient data and successfully characterized the behaviour of INO patients. Further testing is recommended with additional patient data and EEG data, however, the model demonstrates strong clinical indications.

I. Introduction

Internuclear ophthalmoplegia (INO) is a neurological disorder that impairs horizontal eye movement due to a lesion in the medial longitudinal fasciculus (MLF), disrupting communication between the cranial nerves, leading to uncoordinated eye movement. The affected side's eye adduction becomes impaired and the opposite eye experiences jerky movements, called nystagmus, as it tries to compensate for the other eye [1], [2]. This particularly impacts saccadic eye movement, defined as rapid movements of the eye with sudden changes in the point of fixation [3]. This movement is controlled by neural "pulse-steps", where the pulse triggers rapid eye movement to reach a target quickly, while the overall step keeps the eye in the new position [4]. Graphs of saccades in agonist and antagonist eye muscles in INO patients can be found in Appendix A.

The process for the treatment of INO begins with a diagnosis of the condition. However, the detection of INO can be difficult, especially in milder cases, as most patients still demonstrate normal degrees of ocular function with varying degrees of impaired horizontal eye movement [5], [6]. Diagnosis is often done using infrared oculography, which tracks eye movement and blinking through a variety of tests [7]. From there, peak velocities of the adducting and abducting eye can be obtained, and the ratio compared, to determine the Vertical Disconjugacy Index (VDI). If a person's VDI is greater than 1, they are considered to have some level of INO. However, this diagnosis requires significant lab work, and peak velocities may be insufficient for evaluating INO, with other metrics such as adduction delay suggested as potential additional metrics [8].

Another indicator of INO is glissadic overshoot or undershoot, both issues with eye movement caused by errors in muscle activation pulses. Overshoot is where the eye looks past a target due to pulse height errors, and undershoot is where the eye takes longer than usual to settle on a target due to pulse width errors [9], [10]. Graphs demonstrating glissades in agonist and antagonist eye muscles in INO patients can be found in Appendix A.

Saccadic eye movement has been explored in literature through the development of state-space models linking muscle activation to eye position in healthy individuals vs. those with INO. This project builds off of one such model by McSpadden that focuses on horizontal saccades by isolating for the medial and lateral rectus muscles, making use of the Hill muscle model for estimates nonlinear properties such as muscle viscosity and tendon elasticity [11]. The McSpadden model produces accurate simulations of eye position, eye velocity, and muscle tension in relation to neuronal inputs when compared with existing data [12], but has not been quantitatively validated relative to empirical INO eye movement data.

Thus, our project objective is to modify McSpadden's model to be able to take in eye position and velocity data from individuals with and without INO, run simulations, and output information such as muscle velocity, muscle activation, and tendon force. Patterns can be identified from these simulations, such as signs of glissadic overshoot or undershoot to ultimately aid physicians in improving INO diagnosis accuracy while reducing lab work. In future endeavours, this model can be further developed to allow for the diagnosis of other neurological disorders with INO as a common symptom.

II. Methodology

A pre-existing model by McSpadden was utilized to generate the new model that generates eye dynamics that can provide insight into INO. McMcSpadden's model combined kinematics of eye globe rotation, dynamics on agonist and antagonist muscles, and formulations from the Hill muscle model to ultimately produce a state vector with 10 variables suitable for predicting saccadic eye movement given reasonable initial conditions, which are listed in Appendix G. However, it made several assumptions for missing values of terms such as the passive muscle damping factor, activation and deactivation delays, and parameters in the muscle-force-length, muscle-force-velocity, and tendon-force-length curve equations by using reasonable guesses, values in animal studies, or values that produced continuous curves [11].

Importantly, in McSpadden's original model, the neural input functions (i.e., muscle activations) are assumed to be known and are used as inputs to the system of ordinary differential equations (ODEs) [11]. In our approach, we aimed to work in the opposite direction, which was estimating the muscle activation profiles from available kinematic data. By rearranging the equations presented in McSpadden's paper, we found that muscle activation can be computed from tendon force, muscle length, and the first and second derivatives of muscle length (Eqs. (1) and (2)).

$$a(F_{t1}, l_m, \dot{l}_{m1}, \ddot{l}_{m1}) = \frac{F_{t1} - \frac{M}{980}\ddot{l}_{m1} - B_{pm}(\frac{180}{\pi r})\dot{l}_{m1} - F_{pe}(l_{m1})}{F_{\max}F_l(l_{m1})F_v(\dot{l}_{m1})} \tag{1}$$

$$a(F_{t2}, l_m, \dot{l}_{m2}, \ddot{l}_{m2}) = \frac{F_{t2} - \frac{M}{980}\ddot{l}_{m2} - B_{pm}(\frac{180}{\pi r})\dot{l}_{m2} - F_{pe}(l_{m2})}{F_{\max}F_l(l_{m2})F_v(\dot{l}_{m2})} \tag{2}$$

The functions for $F_{pe}()$, $F_l()$, and $F_v()$ can be found in [11]; we will be using the same constants McSpadden has used as well. Following McSpadden's format, we will be using subscripts 1 and 2 to describe agnoist and antagonist associations.

Furthermore, muscle length and its derivatives can be approximated from rotational eye position (θ) and its derivatives, under the assumption that tendon length is negligible. This assumption is consistent with sections of McSpadden's model where tendon length is also ignored for simplification purposes. The horizontal eye movement alters the length of the muscle due to globe rotation. The change in length is proportional to the arc length on the eye. Since the primary muscle length (l_{mp}) when $\theta = 0$ is known (Fig. 23), then we can create the equations for agonist and antagonist muscle lengths l_{m1} and l_{m2}:

$$l_{m1} = l_{mp} - \frac{\pi r}{180}\theta \tag{3}$$

$$l_{m2} = l_{mp} + \frac{\pi r}{180}\theta \tag{4}$$

Then, using chain rule to derive Eqs. (3) and (4) we can find $\dot{l}_m$ and $\ddot{l}_m$ equations to use.

To generate smooth and differentiable rotational position data and its derivatives, we used SPAPS (smoothing spline approximation) with a smoothing parameter of 0.001. This method was chosen over other smoothing techniques because of its ability to produce continuous and smooth curves while preserving the overall shape and trends of the data. This is essential for accurately approximating derivatives.

At this point we are only missing tendon force to be able to calculate muscle activation. We modified McSpadden's model so that we were only using agonist and antagonist tendon forces F_{t1} and F_{t2} as state variables x_1 and x_2:

$$\dot{\vec{x}}(t) = \begin{bmatrix} K_t(x_1)\left[-\dot{\theta}(t) - \left(\frac{180}{\pi r}\right)\dot{l}_{m1}\right] \\ K_t(x_2)\left[\dot{\theta}(t) - \left(\frac{180}{\pi r}\right)\dot{l}_{m2}\right] \end{bmatrix}. \tag{5}$$

These two ODEs are reliant on muscle length velocities and angular velocity, which we already have. For initial values, an observation was made from the simulation graphs in McSpadden's paper that when the eye is not moving, the halfway point between the two tendon forces are 20 gt. We have available the tendon force difference formula from McSpadden's model:

$$F_{t1} - F_{t2} = J_g\ddot{\theta} + B_g\dot{\theta} + K_g\theta \tag{6}$$

Thus using Eq. (6), we are able to calculate the initial tendon force values and run the model to generate vectors for F_{t1} and F_{t2}. However, as muscle length calculations assumed negligible tendon length these tendon forces were not accurate enough to use in muscle activation calculations. Assuming that the generated F_{t1} and F_{t2} are accurate in all but amplitude, we used Eq. (6) to scale them down to accurate levels. Then, we were able to plug all the values into Eqs. (1) and (2) and generate muscle activation data.

To validate the accuracy of our generated activation curves, we re-used them as input to McSpadden's original model and simulated the resulting eye position using `ode45`. For initial values of the state varibles needed to run the simulation, we used the first time point data from the vectors we generated while calculating We then compared the shape of the resulting kinematic output with the original θ data from the literature. Since we still used approximated muscle lengths and their derivatives in computing activation, a perfect match is not expected. Instead, we focus on qualitative comparison—matching the overall shape, timing, and dynamics of the response.

For this final simulation step, activation data points needed to be passed as continuous functions into ode45. Therefore, we again used SPAPS smoothing; but due to the high density of data points resulting from `ode45`'s internal time steps, downsampling was necessary before smoothing to avoid erratic spline behavior. Even after downsampling, the amount of data points meant that we needed a finer smoothing parameter of 0.000001 for the best fit. It is important to note that the resulting activation curves should be interpreted qualitatively. Given the simplifications in McSpadden's model and our modifications, artifacts such as negative activation may appear. Rather than focusing on actual values, it is recommended to focus on features such as pulse width, height, step size, and overall shape (especially the presence of abnormal slopes or atypical pulse dynamics) as indicators of neural activation patterns.

III. Results

A. Validation Approaches

The project aims to reverse McSpadden's original model, using eye position as input and activation as output. The model's accuracy was first validated by inputting eye positions from the paper and comparing the resulting activations to those reported in the published study. As shown in Fig. 3 and Fig. 4 from Appendix C, the Matlab simulation result closely matches the modelled simulation from McSpadden's publication.

To further validate our model on other datasets, we reintroduced the activation vector derived from our modified model into the original McSpadden framework to obtain validation eye position outputs. This validation method involves comparing the raw eye position data with the outputs generated by the model. As detailed in Appendix C, there is a close match in the shape of the graphs, although the exact magnitudes differ due to the assumptions made in our modelling approach. Given that the primary focus of the current project is on comparing the shape of these graphs, our model remains valid despite these discrepancies.

B. Evaluation of Iterations and Accuracy of Modeling

The project's end goal is to track eye movements and interpolate neural signals, requiring a reverse approach to validate key shapes. In previous model iterations, attempts were made to fine-tune parameters such as maximum activation amplitude, initial and final times of the activation phase, and the rates of muscle activation and deactivation. This was to ensure that the simulated eye positions aligned with the raw data, as illustrated in Fig. 15 from Appendix D. Assumptions made about parameters and constants, such as the initial conditions of activation pulses in McSpadden's publication, might not hold true with a different input dataset. Additionally, variations in data collection methods can introduce noise into eye-tracking data. On account of the inconsistencies induced by the parameter approach, the final modelling approach instead utilized SPAPS smoothing in Matlab to achieve a differentiable fit over the literature data, with an example shown in Fig. 16 from Appendix D. Initially, three interpolation methods were considered for processing the data: linear, PCHIP, and spline as displayed in Fig. 17 from Appendix D. However, these methods tended to amplify minor fluctuations in position data. To address the unrealistic behaviour in the derived velocity and acceleration profiles, a method is needed that incorporates a smoothing parameter to effectively fit noisy data and supports differentiation to any order. The linear method results in discontinuous derivatives, while the PCHIP method is only differentiable to the first order. SPAPS, a type of spline method, uses absolute error tolerance to control the smoothing level, which allows it to manage higher orders of continuous differentiation more effectively than standard cubic spline methods. The current modelling approach automatically optimizes state-space parameters.

C. Simulation Results and Analysis

(1) Comparing muscle and neural responses between healthy and INO populations during a 20-degree abducting saccade

Horizontal saccade eye-tracking data for both healthy individuals and subjects with bilateral INO are presented in Fig. 5 and Fig. 7 from Appendix C, obtained from the study by Matta et al. [13]. The healthy control demonstrates typical saccadic movements with a smooth eye movement trajectory, whereas the INO subject exhibits a clear overshoot before stabilizing. This simulation aims to compare state trajectories between the two groups. The response from the healthy participant is smoother than that from the INO participant. The output of the INO population is more oscillatory and displays more peaks in the pattern. The overshoot is also reflected in the muscle length and tendon forces graphs, suggesting that INO participants overcompensate with their muscles to achieve certain eye movements. The muscle velocity profiles are blunted and less distinct in INO subjects. Notably, the muscle activation plots show a prolonged pulse width in INO subjects, which indicates a slower change in activation and suggests that it takes longer for their eyes to settle.

(2) Comparing none to severe adduction delays in the INO population during a 12-degree saccade

In this simulation, eye-tracking data from three INO participants with varying degrees of adduction delay are analyzed. The raw position data is presented in the first row of Appendix E, from a study by Nij Bijvank et al. [14]. All participants were instructed to perform a left saccade, involving abduction of the left eye and adduction of the right eye. The data has been mirrored to show increasing position values for movement. This simulation explores different state-space outputs, particularly muscle activation, with different extents of delay within the INO population. Differences in muscle activation among the participants are shown in Appendix E, and the complete state trajectories can be found in Appendix C.

As seen in Fig. 18 and Fig. 19 found in Appendix E, the severity of adduction delay correlates with increased asynchronization between the adducting and abducting eye movements. A greater delay results in a wider pulse width in the adducting eye. Although a similar increase in pulse duration is observed in the abducting eye, it is less pronounced. This aligns with previous research where adducting saccades are significantly slower in INO patients compared to non-INO subjects, and that saccades of the abducting eye may also exhibit reduced speed [14]. This result shows both the asynchronization and delayed responses in eye movement associated with INO from the perspective of muscle activation.

(3) Comparing a healthy individual and an INO patient with no adduction delay

As detailed previously, VDI compares adduction and abduction movements; however, it may not capture all INO characteristics, especially in the absence of adduction delay. This simulation aims to compare groups to investigate whether other characteristics, such as muscle activation, correspond to known INO features when VDI alone is insufficient. The data used in this simulation come from Simulation Tasks #1 and #2.

Appendix F shows findings consistent with Simulation #1: muscle activation in the healthy participant is smoother compared to the INO participant, possibly due to overshooting/oscillation observed in the INO participant. However, the duration of activation pulses is similar between the two. This suggests that the adduction delay may be associated with errors in pulse width, and oscillation could be correlated to unstable muscle activation.

IV. Discussion

A. Outcomes of The Study

The state space model developed on Matlab was able to successfully simulate and capture the key dynamics of saccadic eye movement in healthy individuals and those who struggle with INO. The simulations ran on healthy individuals for both abduction and adduction exhibited controlled and symmetrical movement. As seen in Fig. 5 and Fig. 6 located in Appendix C, for healthy individuals, the eye position curves had a smooth rise and the eye velocity peaked sharply before declining symmetrically. Both eye position and velocity graphs generated for healthy data indicate normal saccadic behaviour with no glissadic overshoot. This was established by comparing the graphs to literature describing the eye position and velocity profiles of those with glissadic overshoot, occurring often in those with INO. The healthy data eye position graphs plateau without dipping and the eye velocity graphs do not have any fluctuations, indicating that normal saccadic movement was achieved. Additionally, the muscle length and velocity profiles displayed well-coordinated, reciprocal actions between the antagonist and agonist muscles, which was concluded based on the inverse graphs between the agonist and antagonist muscle profiles. Tendon forces followed a smooth rise and fall pattern, indicating that the movement was well coordinated. Lastly, the neural activation patterns were complementary between the agonist and antagonist muscles, indicating that there were no pulse height or pulse width errors. Eye position, eye velocity, muscle length, muscle velocity, tendon forces and activation graphs for healthy individuals can be found in Fig. 5 and Fig. 6.

In contrast, the INO data displayed clear disruptions in the dynamics described, which can be seen in Figs. 7 to 14 found in Appendix C. The eye position and eye velocity profiles of both the left and right eyes from unhealthy data showed signs of glissadic overshoot, as the eye position graph contains a small dip before plateauing, or there are fluctuations in the eye velocity graph. These fluctuations and dips occur due to glissadic overshoot, which occurs when there is a mismatch between the pulse step and components of the motoneuronal controlled signal caused by errors in pulse width [4]. Furthermore, muscle velocities for both the agonist and antagonist muscles were not symmetrical anymore, tendon forces were unbalanced and did not plateau, and the neural activation profile showed signs of irregular peaks or prolonged duration. These graphs, generated from the unhealthy data, are indicative of uncoordinated neural signalling, providing evidence of INO.

B. Interpretation of Results

The primary goal of this project was to reconstruct an existing model that is capable of predicting the neural activation patterns behind saccadic eye movement based on eye position-time data. By applying Matlab's smoothing feature to eye position data and inputting this into the new model, the simulation was able to generate muscle and neural profiles that match the responses found in literature. In healthy cases, the absence of post-peak fluctuations in the velocity graph and smooth stabilization in the position graphs can be strongly correlated to normal saccadic behaviour. In INO cases, the simulation presented pre-plateau dips in position data and post-peak instabilities in velocity data, matching the indications of the known pathophysiology of INO. The

Matlab program was also able to simulate how different degrees of adduction delay can manifest in INO. With an increasing delay observed, taking a larger amount of time for neural activation to stabilize, the muscle activation pulses widen and neural synchronization decreases. This can be seen in Fig. 18, comparing the activation graphs for varying delay in INO individuals. This aligns with studies by Matta et al. and Nij Bijvank et al., who describe longer activation durations and increased disconjugate eye movement in INO patients [13]. Even in INO cases where there was not an obvious adduction delay, the simulation showed a clearly unstable neural activation profile compared to healthy controls, demonstrating the sensitivity of the model. To summarize, the model achieved the project objective of generating profiles for eye dynamics using eye position data as an input for the system, to provide insight into saccadic dysfunction for INO diagnosis.

As previously explained, internuclear ophthalmoplegia arises from lesions on the medial longitudinal fasciculus (MLF), which disrupts the signal coordination between the oculomotor and abducens nuclei. This leads to delayed or impaired adduction and nystagmus in the abducting eye. The model shows evidence supporting the physiological basis of INO. There is a clear delay or prolonged activation in the adducting muscle while the antagonist muscle does not deactivate in time. Furthermore, the neural drive is less efficient, resulting in unstable and inefficient eye movement. The state space model can provide quantitative information that can allow clinicians or researchers to better understand the severity and progression of INO, where measurements such as VDI might be inconclusive.

C. Limitations

There were a few limitations to the MATLAB model that was developed for this project that will be addressed in future iterations. One of the main limitations was the simplification of the muscle-tendon dynamics. The simplification neglects tendon length in the muscle length calculations, which can, in turn, affect the accuracy of the model. Additionally, the model relies on the smoothening of the position data rather than raw data, which can overlook minor instabilities that can provide further insight. The model created also does not show demographics of the data, which limits the customization of the simulations. Lastly, the absence of clinical validation limited the ability to determine the accuracy and applicability of the results.

V. Conclusion

In summary, this project achieved the intended goals of simulating the neural activations and dynamics of horizontal saccadic eye movement in those with and without INO. It successfully implemented portions of [11]'s state-space model relating to muscle activation, producing simulations of muscle and neural responses during 20-degree saccades, adduction delays during 12-degree saccades, and adduction delays vs. lack thereof in INO data vs. non-INO data. These simulations were validated using shape validity, comparison to existing literature, and comparison to the original state-space model to confirm that the model was sufficient in identifying patterns associated with INO such as glissadic overshoot / undershoot and adduction delays.

For future iterations of the model, incorporating tendon length is a priority as tendon elasticity and length should be considered to simulate muscle mechanics more accurately. Furthermore, using real clinical data from a larger pool of individuals would allow for more statistical comparisons and model testing based on individual physiology. This could also provide confidence statistics based on how patients were correctly identified as INO. Adding on, cross-validation with fMRI or EEG data could help verify that the neural activation profiles simulated correspond to the clinical data. Lastly, the final recommendation for improving this state space model is developing it into a tool that can be used by physicians for diagnosing INO in a more efficient and non-invasive manner, satisfying the initial goal of the project.

References

[1] P. Fiester, S. Baig, J. Patel, and D. Rao, “An anatomic, imaging, and clinical review of the medial longitudinal fasciculus,” *Journal of Clinical Imaging Science*, vol. 10, p. 83, 2020. doi: 10.25259/jcis_49_2020.

[2] J. D. Nguyen and H. Duong, “Anatomy, head and neck: Eye nerves,” *StatPearls*, 2023, Last updated July 25, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK549919/.

[3] “Types of eye movements and their functions,” in *Neuroscience*, D. Purves, G. J. Augustine, D. Fitzpatrick, *et al.*, Eds., 2nd. Sunderland, MA: Sinauer Associates, 2001. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK10991/.

[4] S. Ramat, R. J. Leigh, D. S. Zee, and L. M. Optican, “What clinical disorders tell us about the neural control of saccadic eye movements,” *Brain*, vol. 130, pp. 10–35, 1 2006. doi: 10.1093/brain/awl309.

[5] Y. Wu, M. Cafiero-Chin, and C. L. Marques, “Wall-eyed bilateral internuclear ophthalmoplegia: Review of pathogenesis, diagnosis, prognosis and management,” *Clinical and Experimental Optometry*, vol. 98, pp. 25–30, 1 2015. doi: 10.1111/cxo.12200.

[6] M. F. Rahmadiansyah and M. Hidayat, “Clinical insights into internuclear ophthalmoplegia: A case report,” *Bioscientia Medicina : Journal of Biomedicine and Translational Research*, vol. 7, pp. 3789–3794, 11 2023. doi: 10.37275/bsm.v7i11.891.

[7] K. Vodrahalli, M. Filipkowski, T. Chen, J. Zou, and Y. J. Liao, “Predicting visuo-motor diseases from eye tracking data,” *Pacific Symposium on Biocomputing*, vol. 27, pp. 242–253, 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10635679/.

[8] J. Ventre, A. Vighetto, G. Bailly, and C. Prablanc, “Saccade metrics in multiple sclerosis: Versional velocity disconjugacy as the best clue?” *Journal of the Neurological Sciences*, vol. 102, pp. 144–149, 2 1991. doi: 10.1016/0022-510x(91)90062-c.

[9] R. W. Baloh, “Internuclear ophthalmoplegia,” *Archives of Neurology*, vol. 35, p. 484, 8 1978. doi: 10.1001/archneur.1978.00500320004002.

[10] L. F. Dell’Osso, D. A. Robinson, and R. B. Daroff, “Optokinetic asymmetry in internuclear ophthalmoplegia,” *Archives of Neurology*, vol. 31, pp. 138–139, 2 1974. doi: 10.1001/archneur.1974.00490380086012.

[11] A. McSpadden, “A mathematical model of human saccadic eye movement,” M.S. thesis, Texas Tech University, 1998. [Online]. Available: https://ttu-ir.tdl.org/bitstreams/549eaf8e-458a-4187-8f35-aa0e774fe0cb/download.

[12] P. Bach-y-Rita and C. C. Collins, Eds., *The Control of Eye Movements*. New York: Academic Press, 1971, isbn: 978-0-12-071050-8. [Online]. Available: https://shop.elsevier.com/books/the-control-of-eye-movements/bach-y-rita/978-0-12-071050-8.

[13] M. Matta, R. J. Leigh, M. Pugliatti, I. Aiello, and A. Serra, “Using fast eye movements to study fatigue in multiple sclerosis,” *Neurology*, vol. 73, pp. 798–804, 10 2009. doi: 10.1212/wnl.0b013e3181b6bbf4.

[14] J. A. N. Bijvank, L. J. v. Rijn, L. J. Balk, H. S. Tan, B. M. Uitdehaag, and A. Petzold, “Diagnosing and quantifying a common deficit in multiple sclerosis,” *Neurology*, vol. 92, 20 2019. DOI: 10.1212/wnl.0000000000007499.

[15] N. A. Flierman, A. Ignashchenkova, M. Negrello, P. Thier, C. I. D. Zeeuw, and A. Badura, “Glissades are altered by lesions to the oculomotor vermis but not by saccadic adaptation,” *Frontiers in Behavioral Neuroscience*, vol. 13, 2019. DOI: 10.3389/fnbeh.2019.00194.

Appendices

A Background Information on Internuclear Ophthalmoplegia

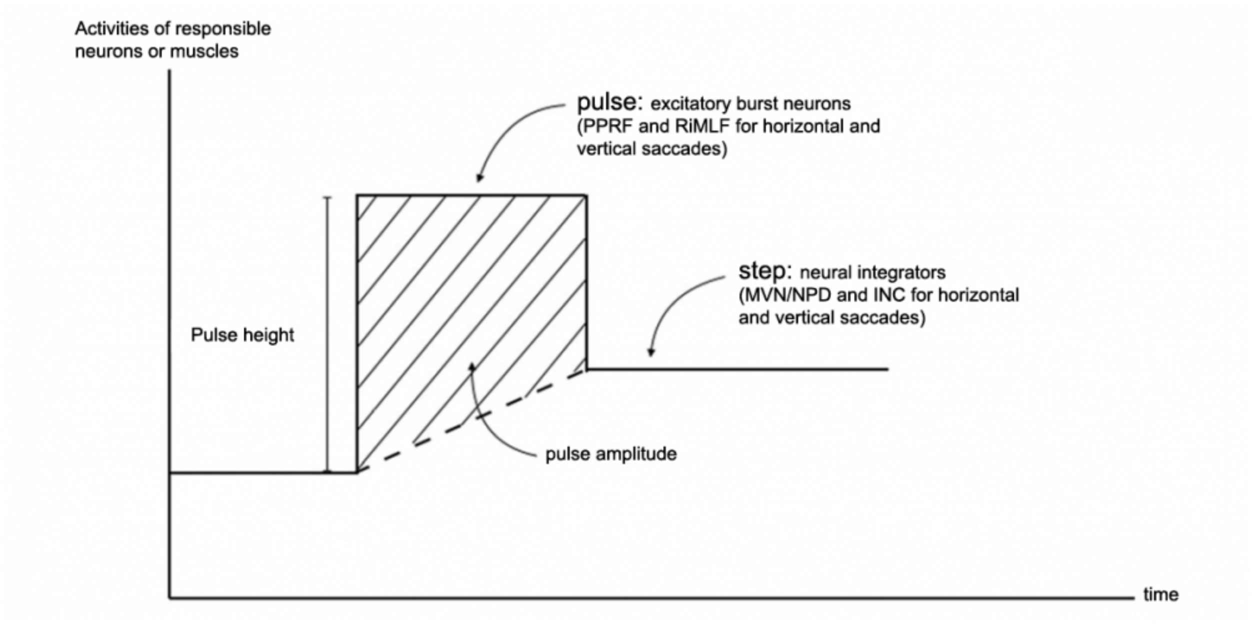

Activities of responsible neurons or muscles
pulse: excitatory burst neurons (PPRF and RiMLF for horizontal and vertical saccades)
step: neural integrators (MVN/NPD and INC for horizontal and vertical saccades)
Pulse height
pulse amplitude
time

Fig. 1: Pulse-step innervations and resulting saccades [4].

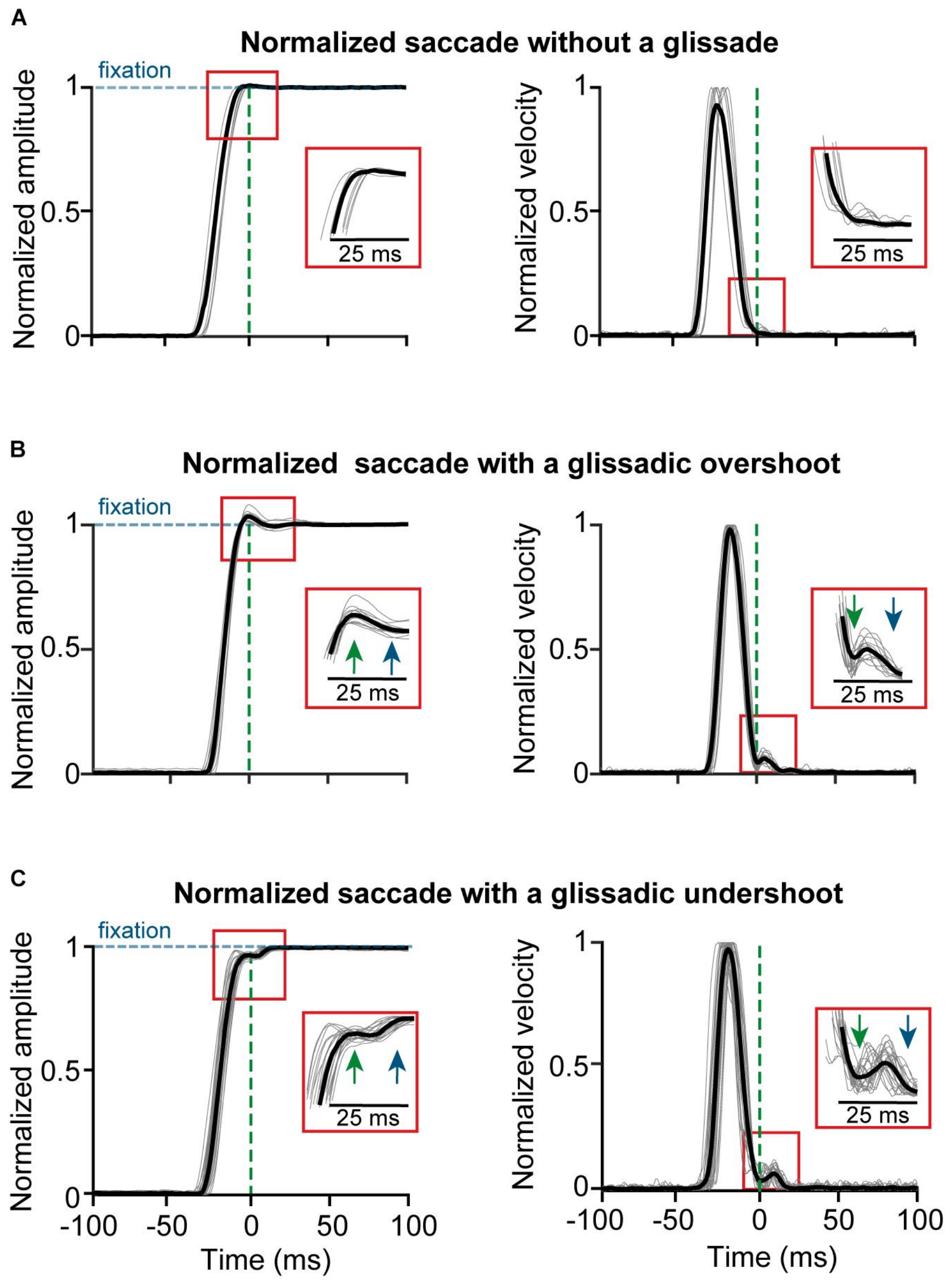


Fig. 2: Normal saccades vs. saccades with glissadic overshoot [15].

B Implementation of McSpadden Paper Model

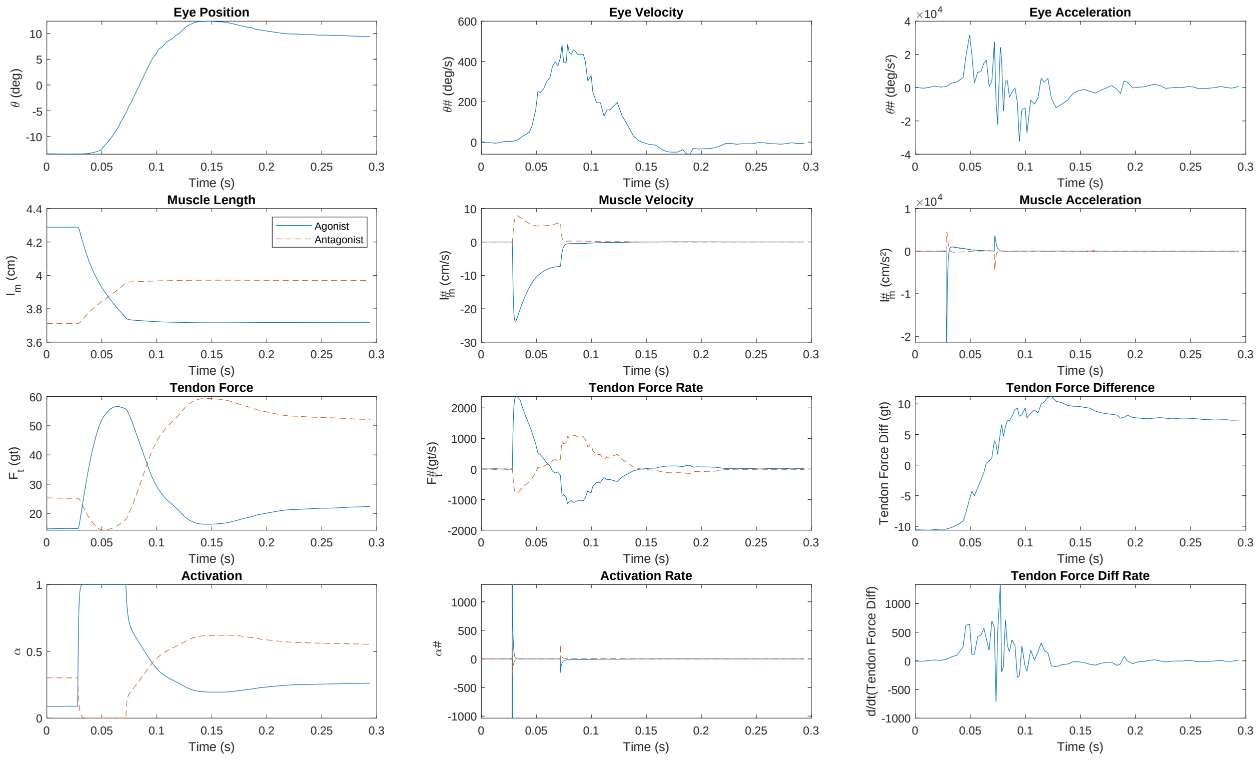


Fig. 3: Matlab implementation of McSpadden's model.

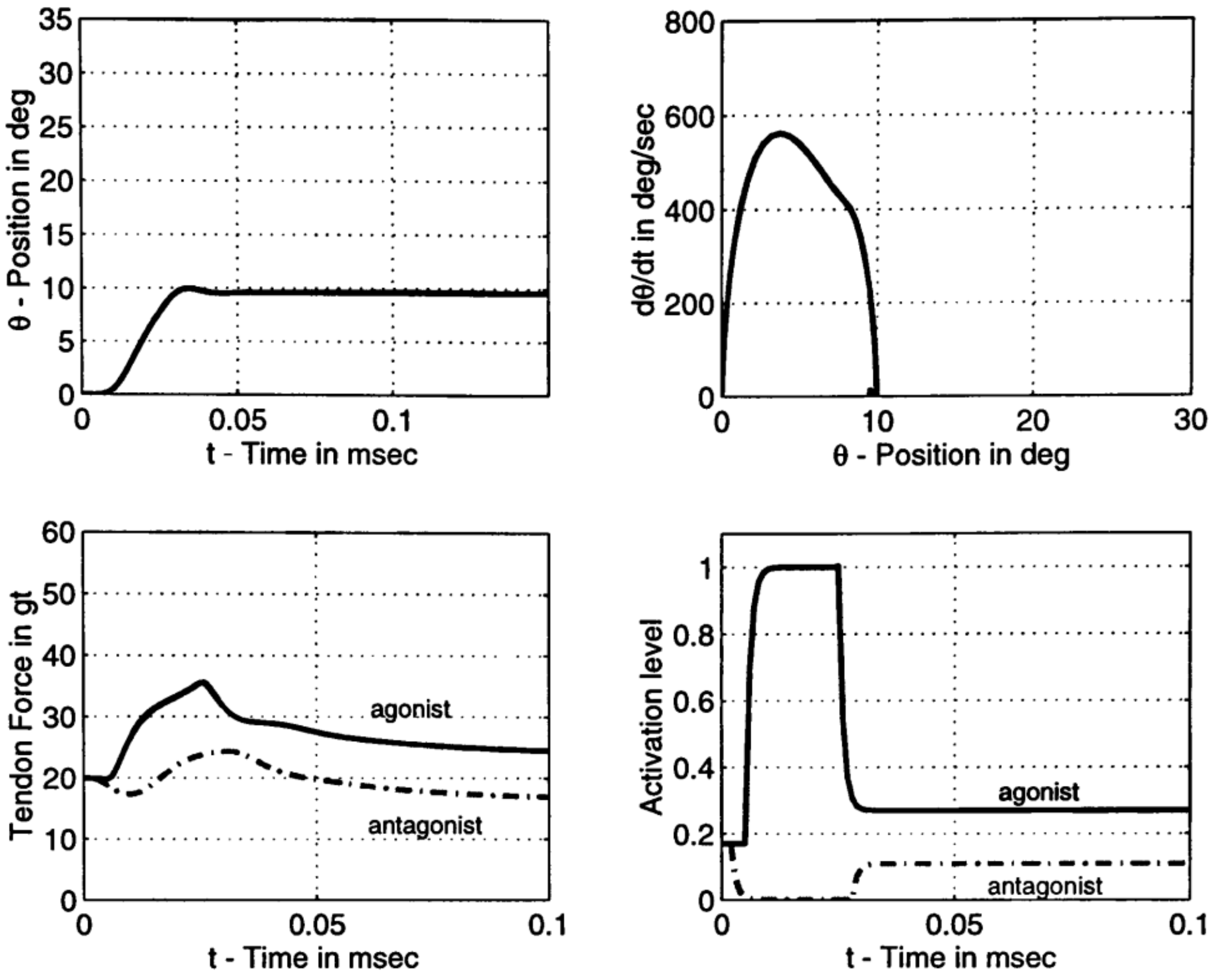


Fig. 4: Model simulation from McSpadden's publication.

C Validation Graphs and Simulation Results

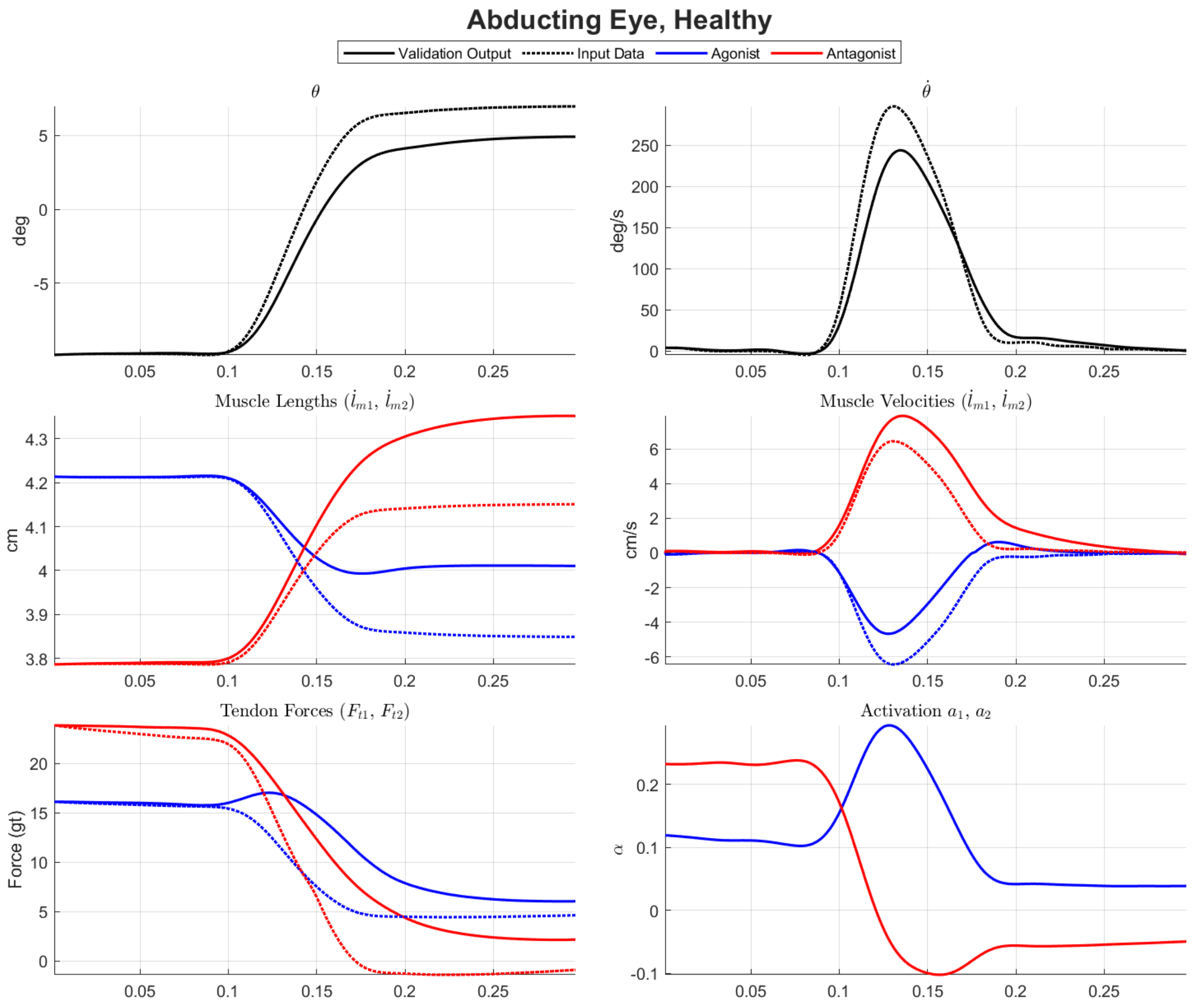


Fig. 5: Abducting eye dynamics for healthy individuals.

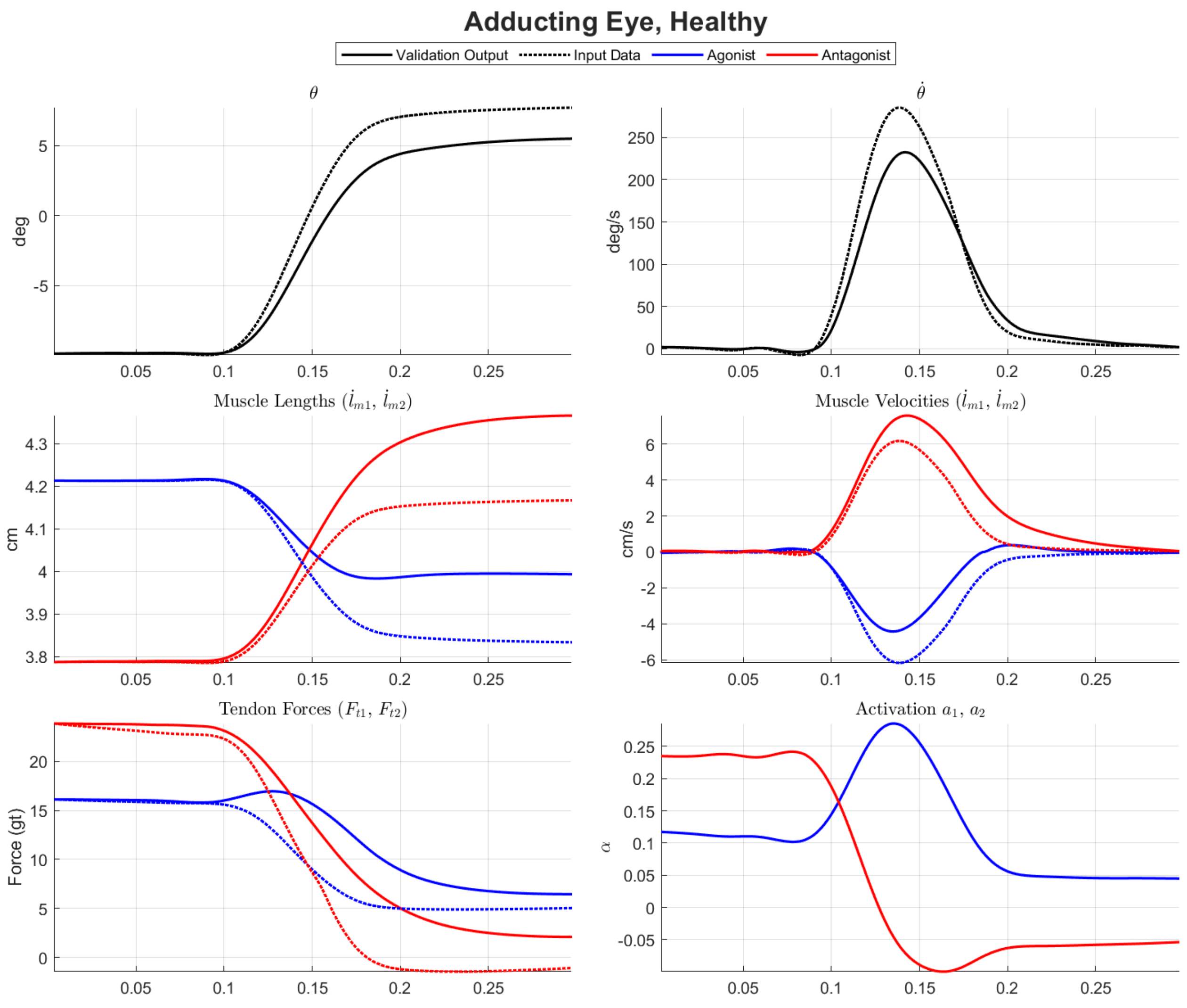


Fig. 6: Adducting eye dynamics for healthy individuals.

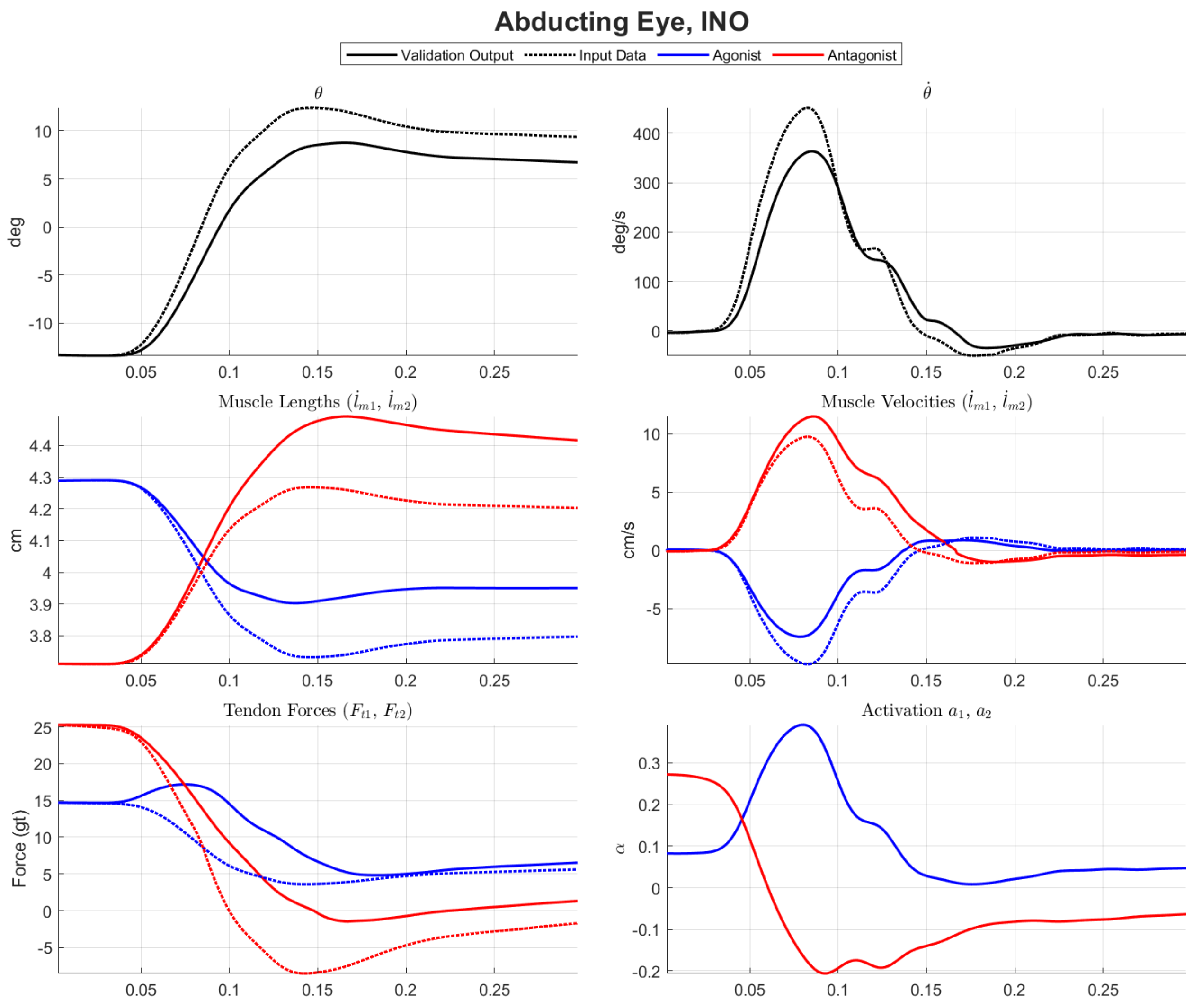


Fig. 7: Abducting eye dynamics for INO individuals.

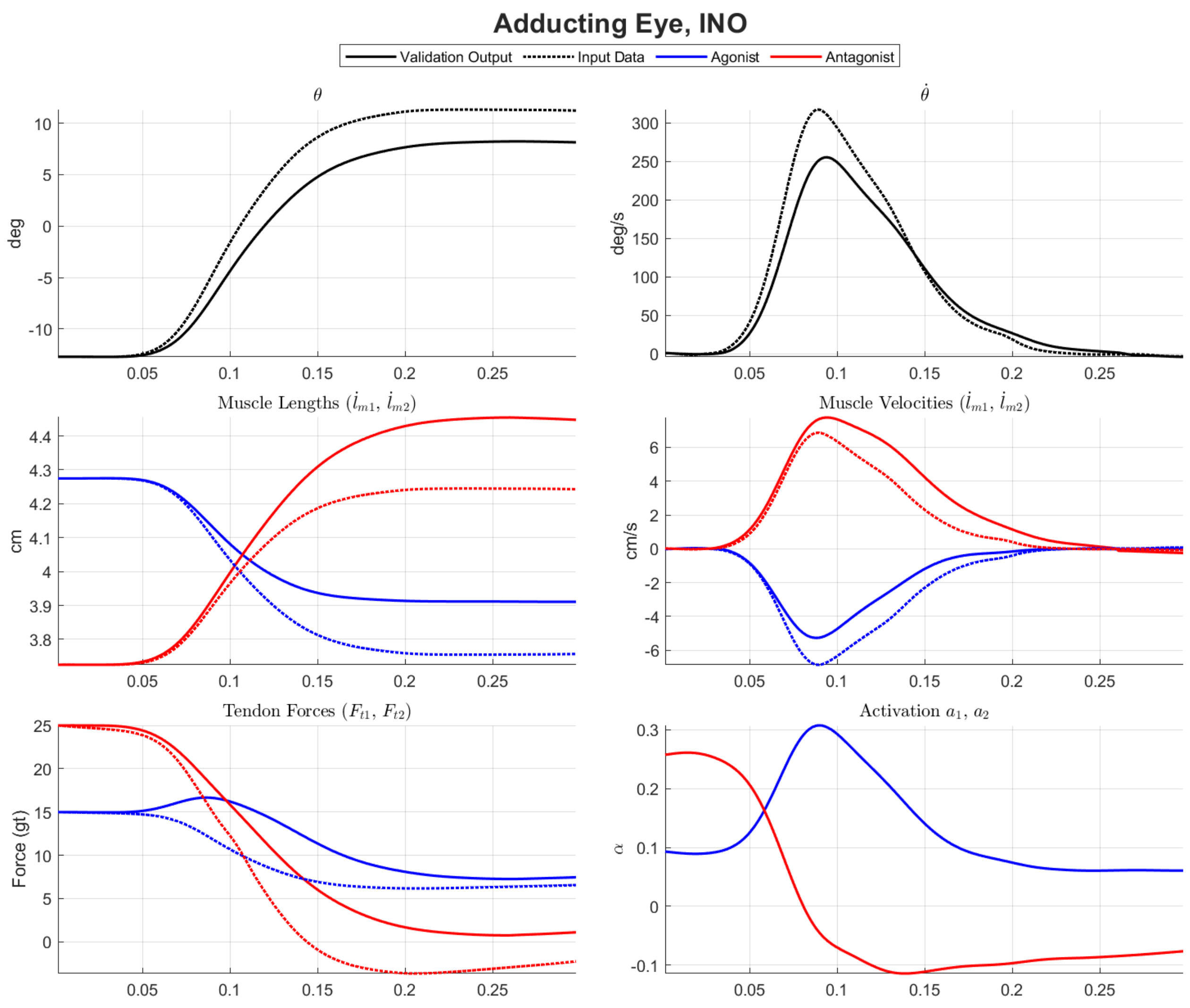


Fig. 8: Adducting eye dynamics for INO individuals.

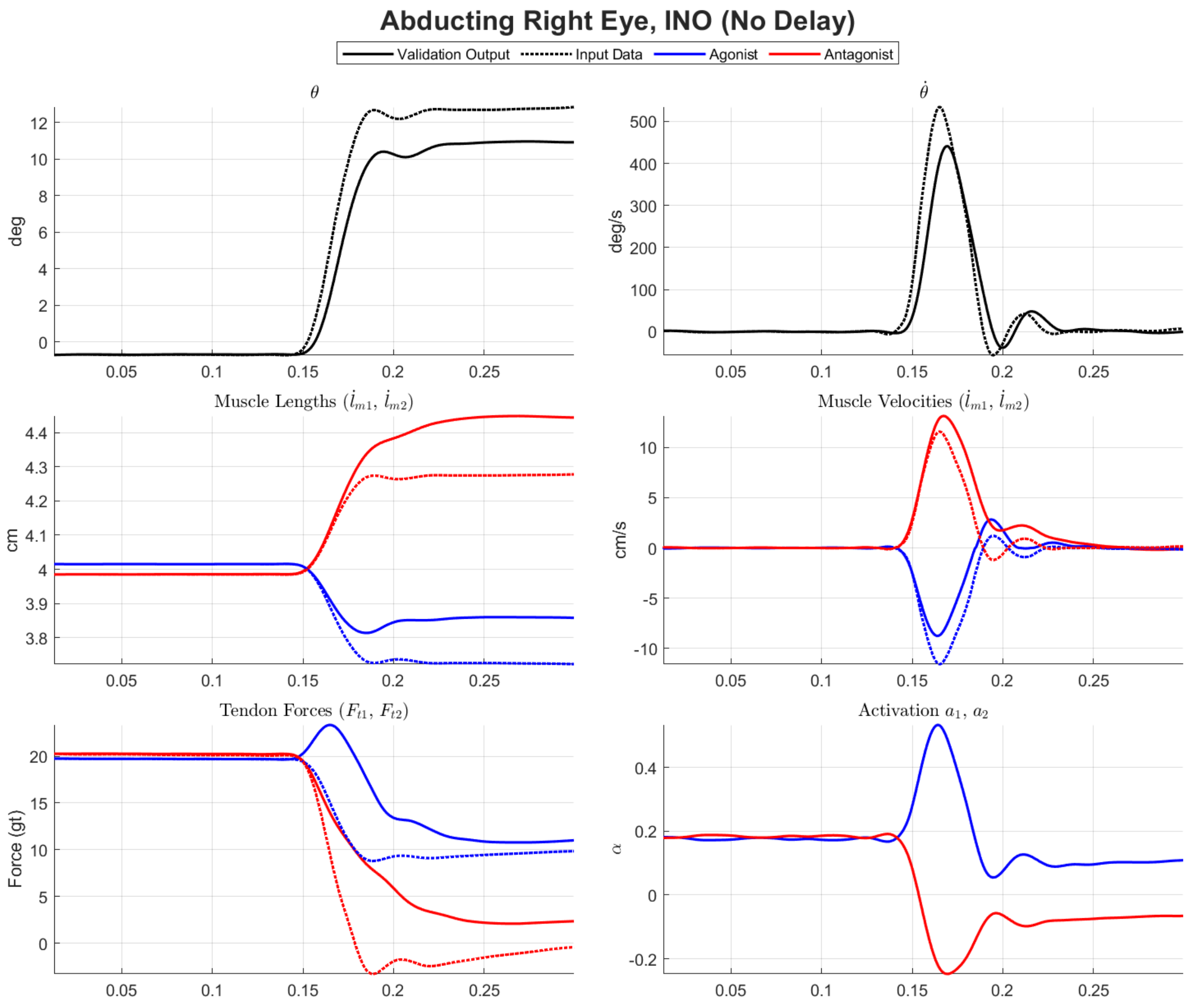


Fig. 9: Right eye abducting dynamics for INO individuals with no obvious delay.

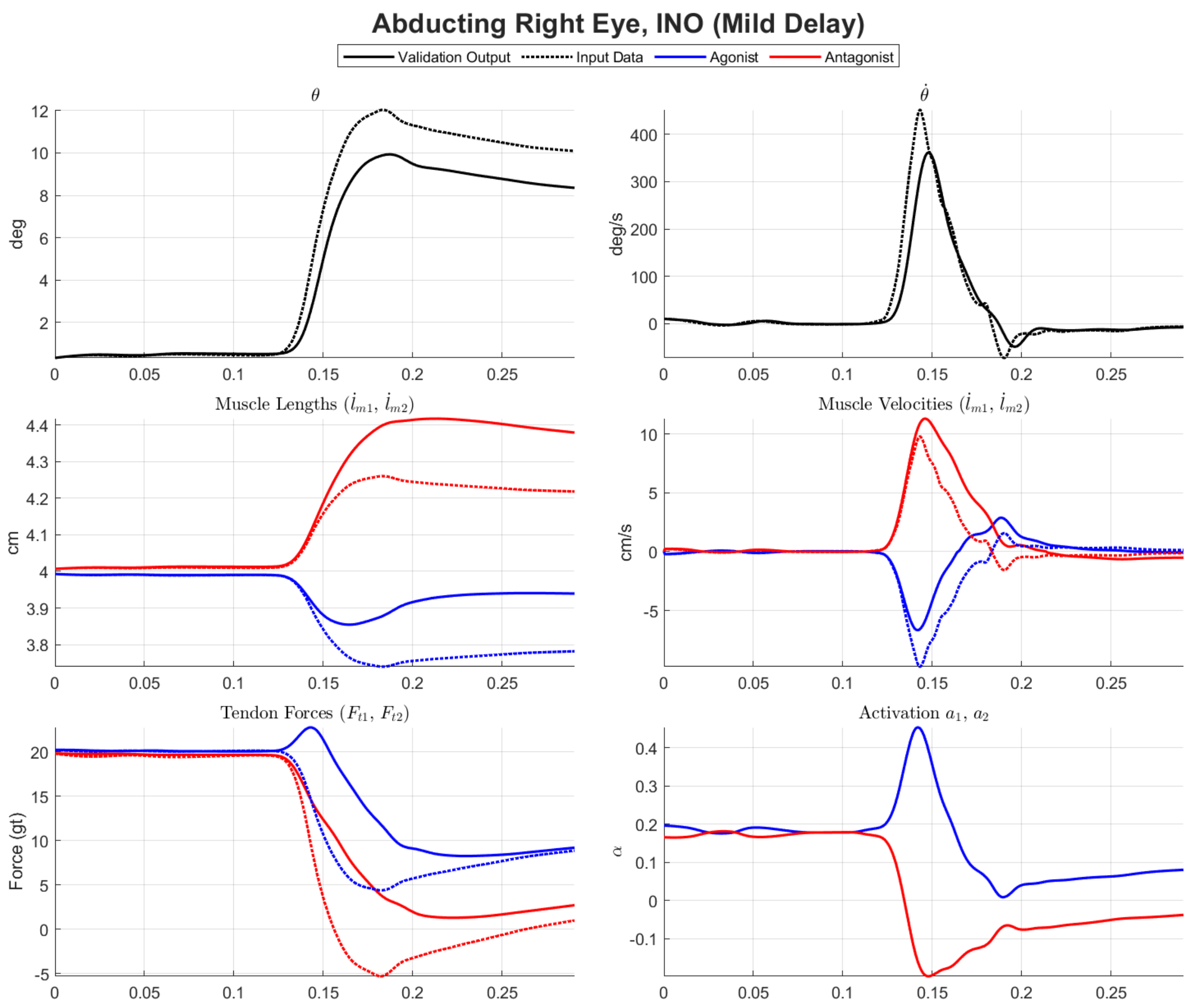


Fig. 10: Right eye abducting dynamics for INO individuals with mildly obvious delay.

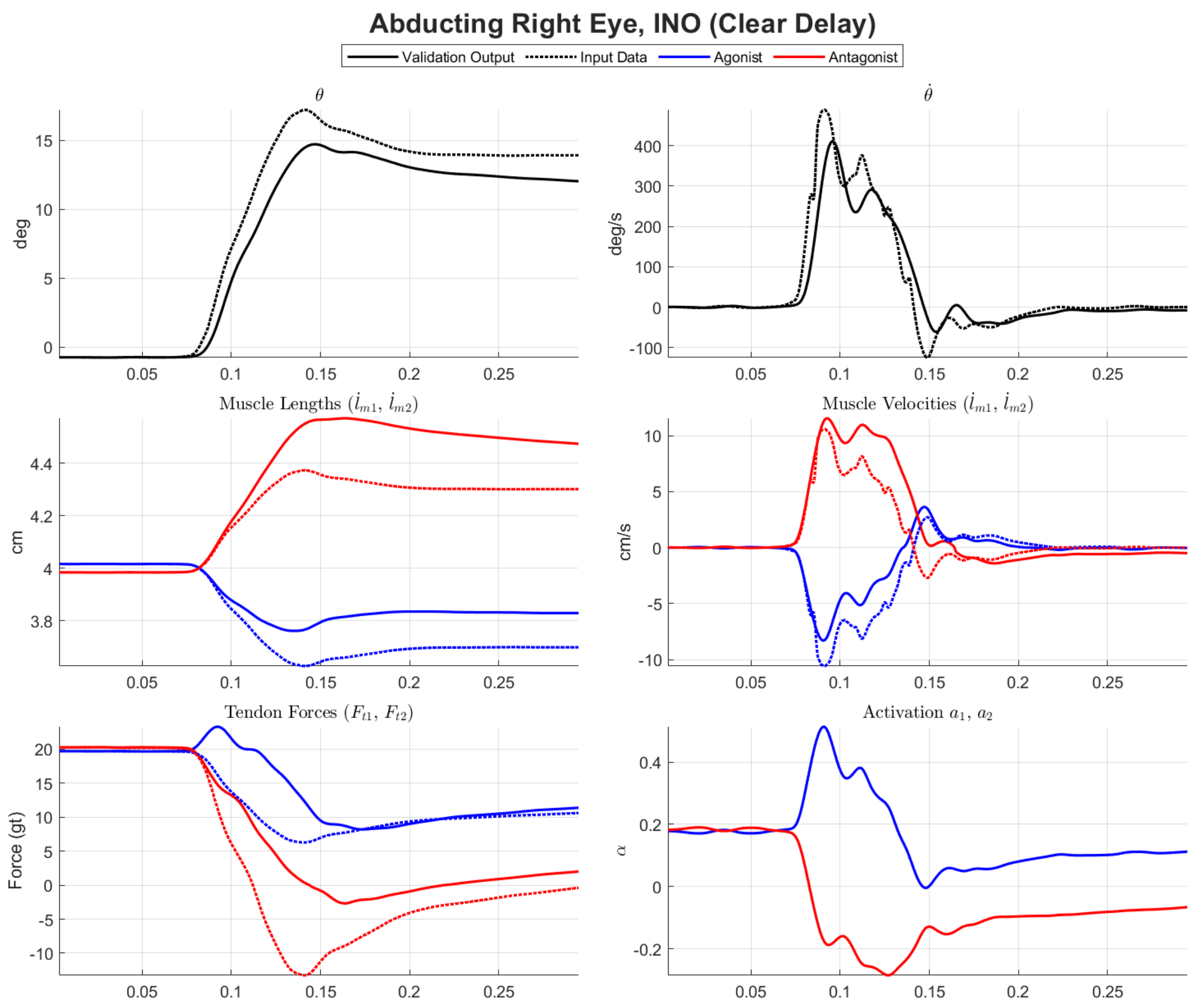


Fig. 11: Right eye abducting dynamics for INO individuals with obvious delay.

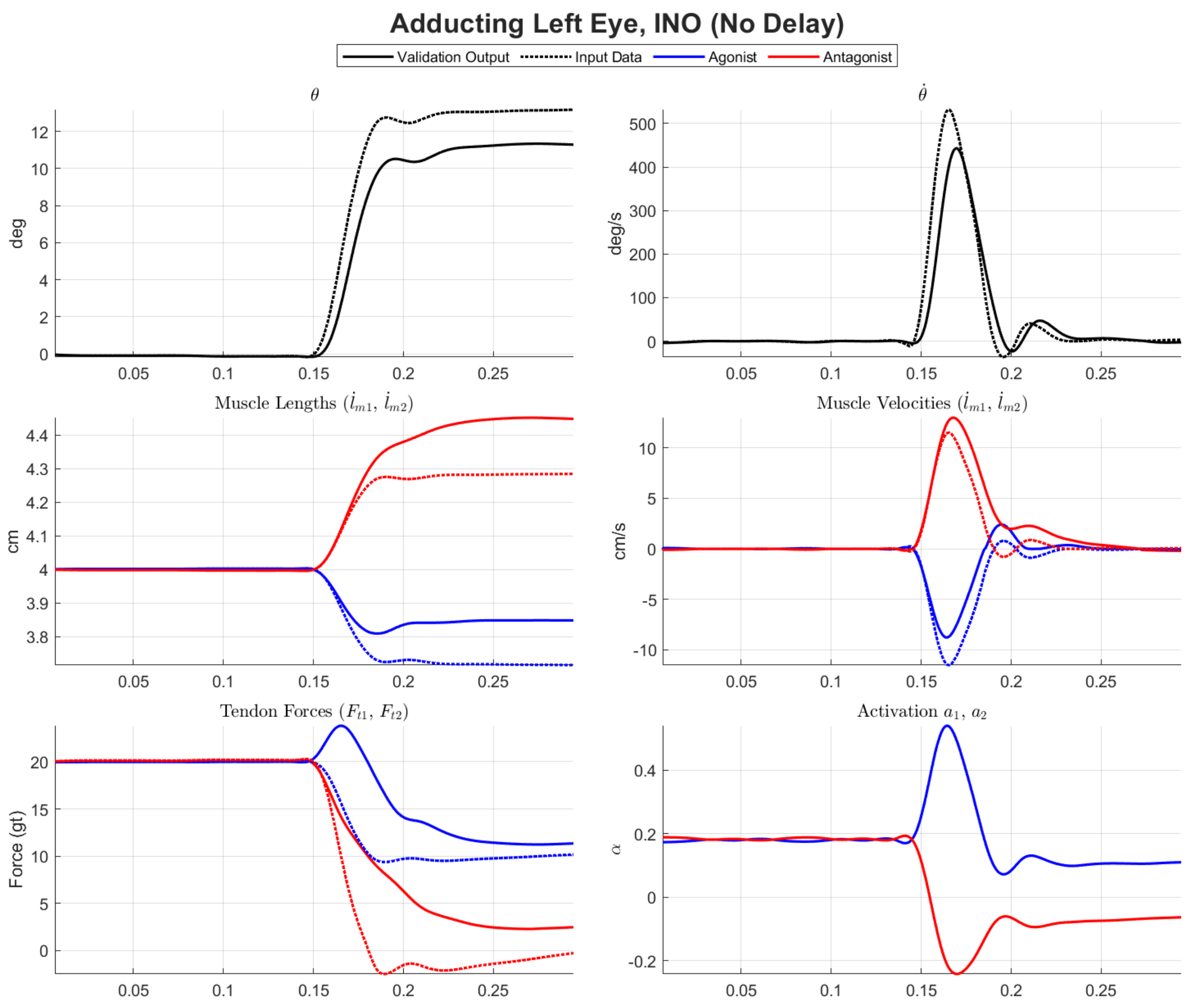


Fig. 12: Left eye adducting dynamics for INO individuals with no obvious delay.

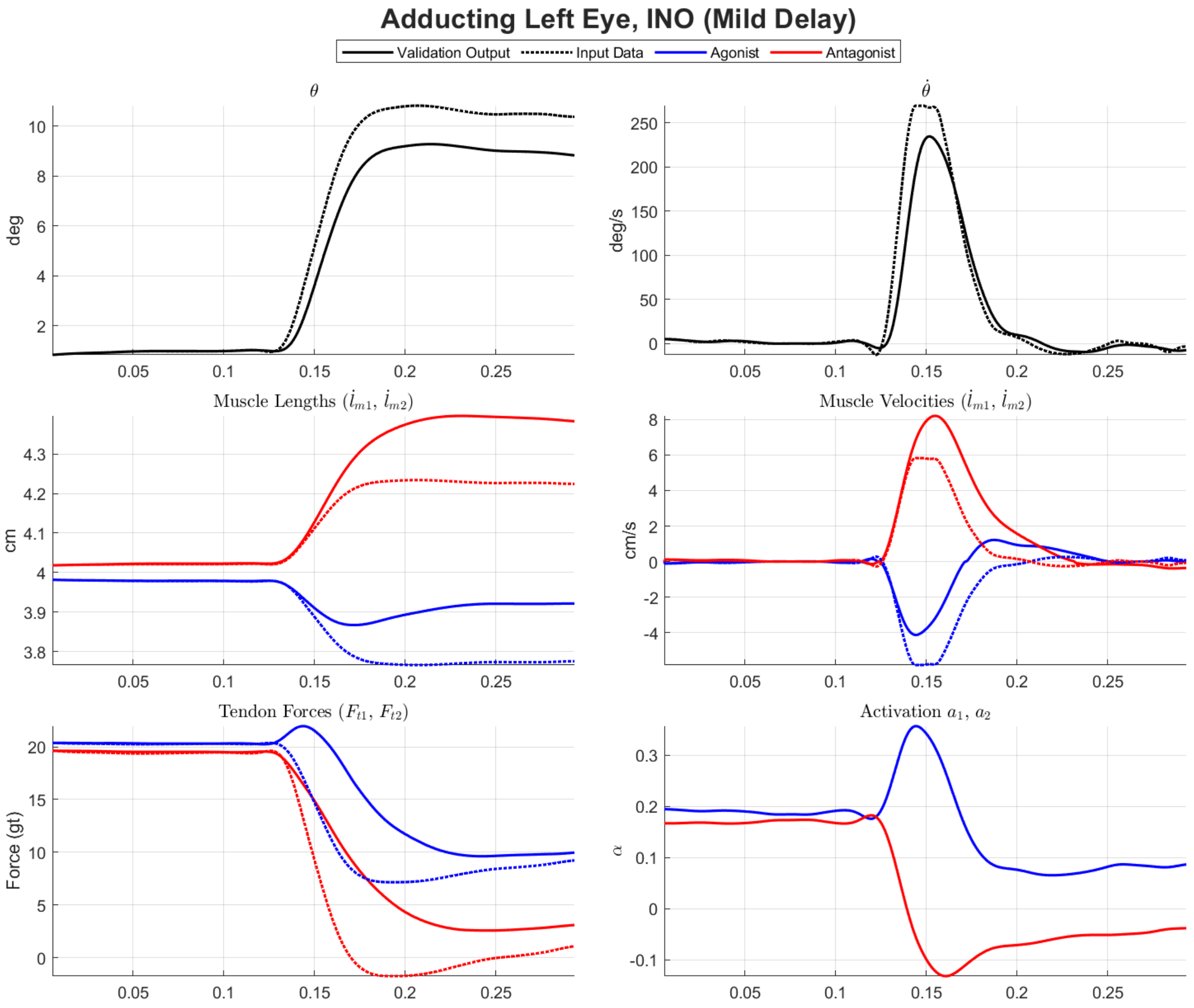


Fig. 13: Left eye adducting dynamics for INO individuals with mildly obvious delay.

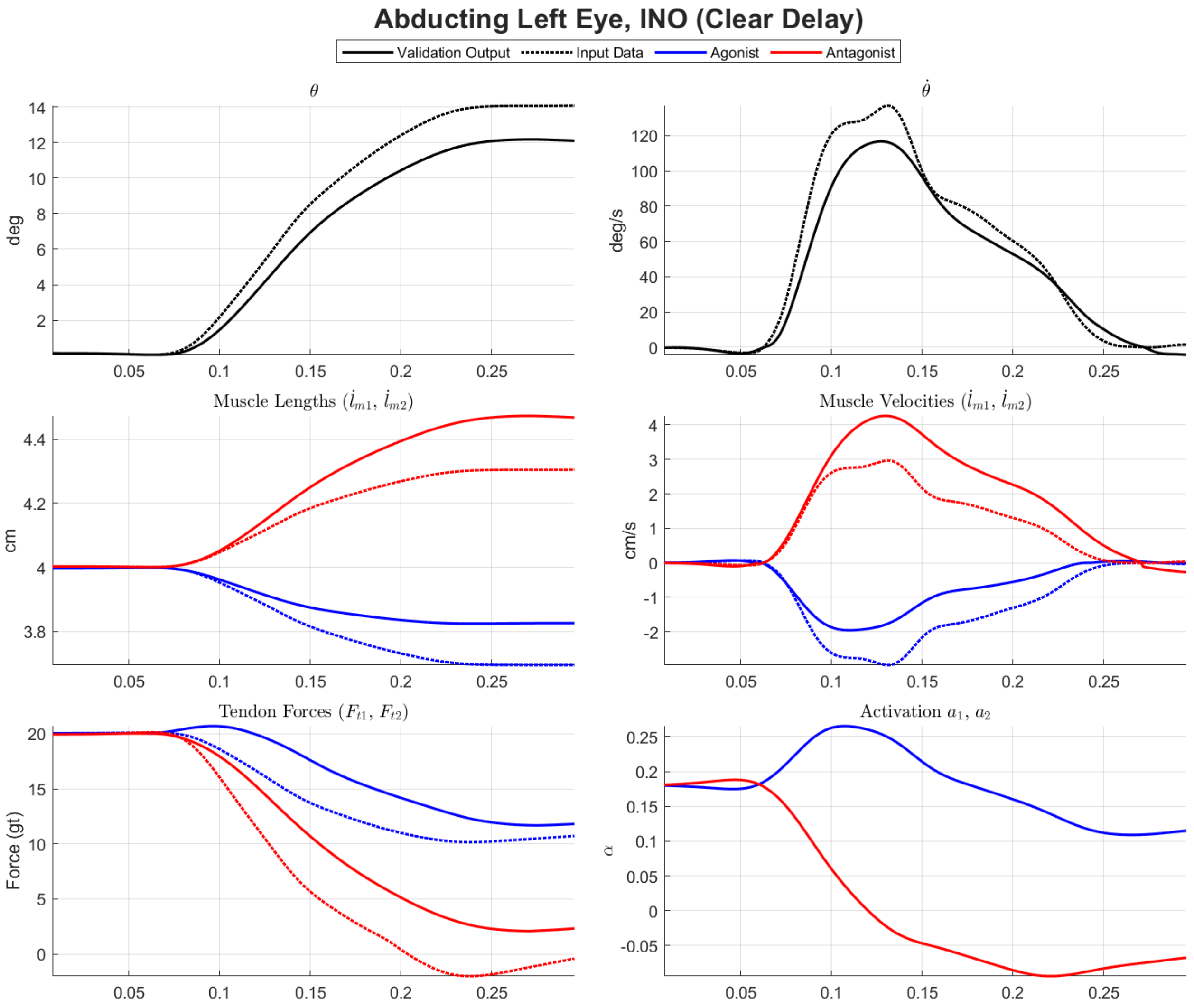


Fig. 14: Left eye adducting dynamics for INO individuals with obvious delay.

D Modelling Iterations

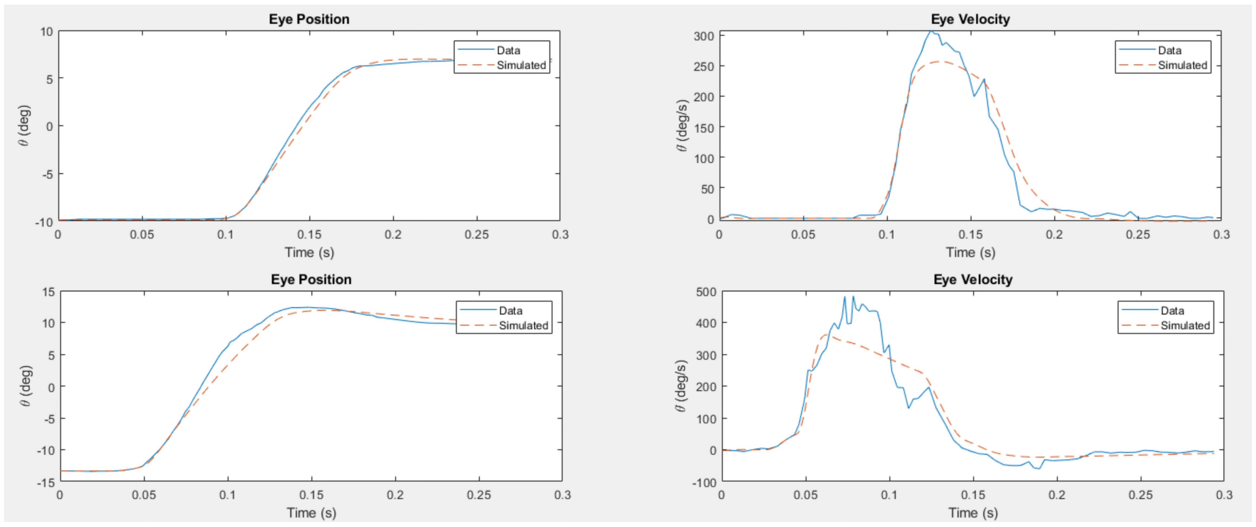


Fig. 15: Comparison of raw and simulated eye positions using parameter matching techniques.

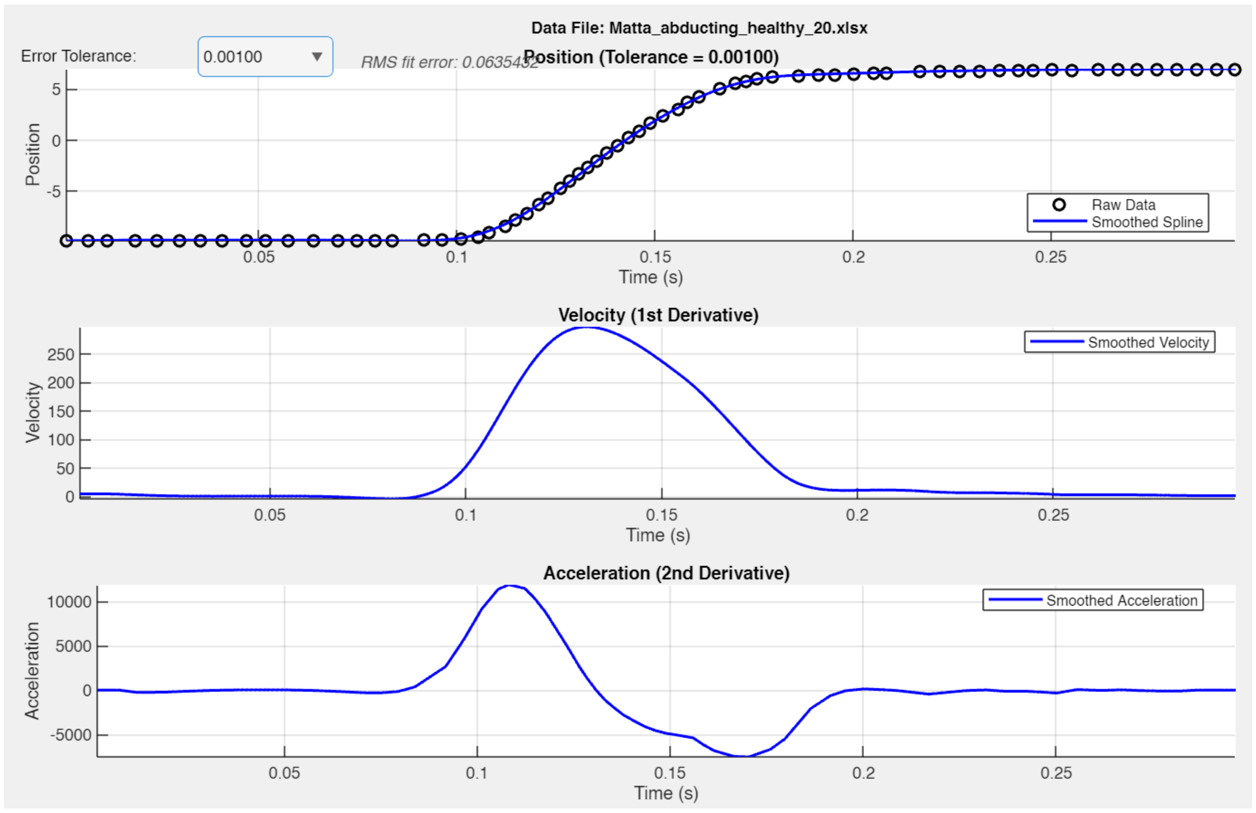


Fig. 16: SPAPS smoothing applied to raw and smoothed eye position data.

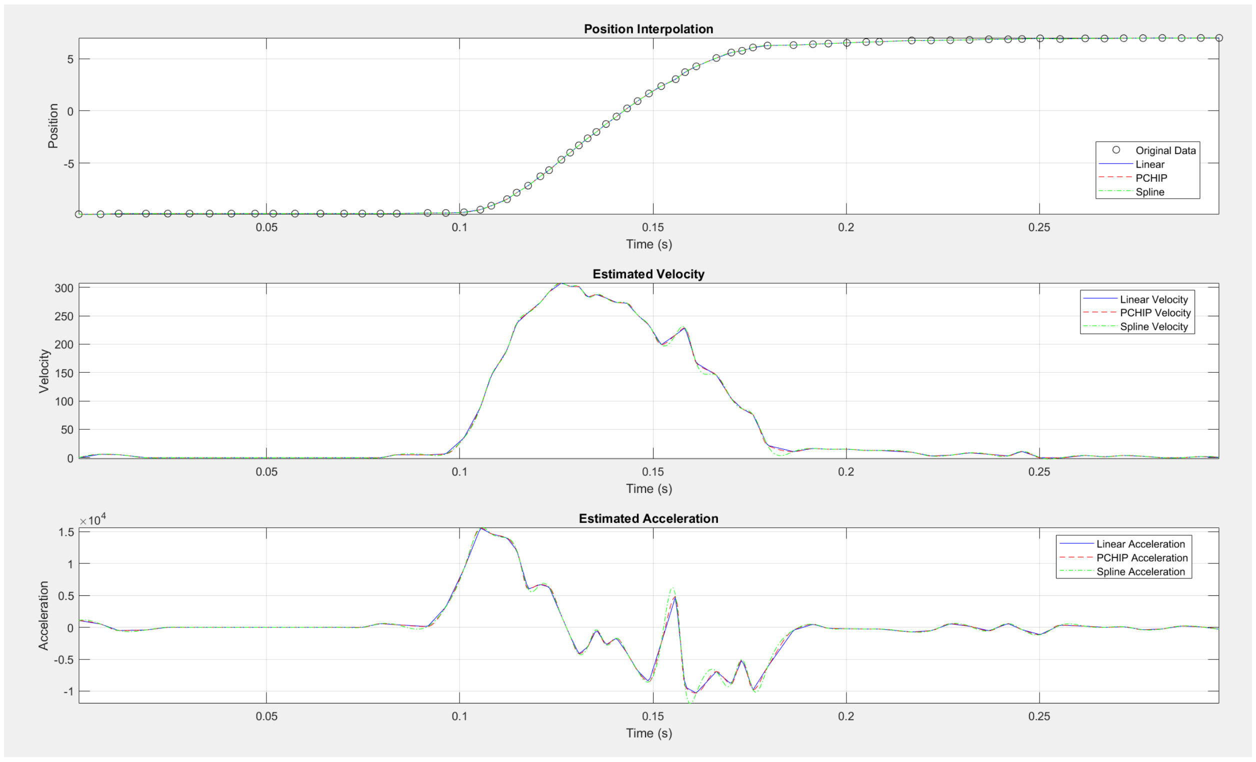


Fig. 17: Evaluating various Matlab smoothing algorithms from the Curve Fitting Toolbox.

E INO Varying Delay Comparisons

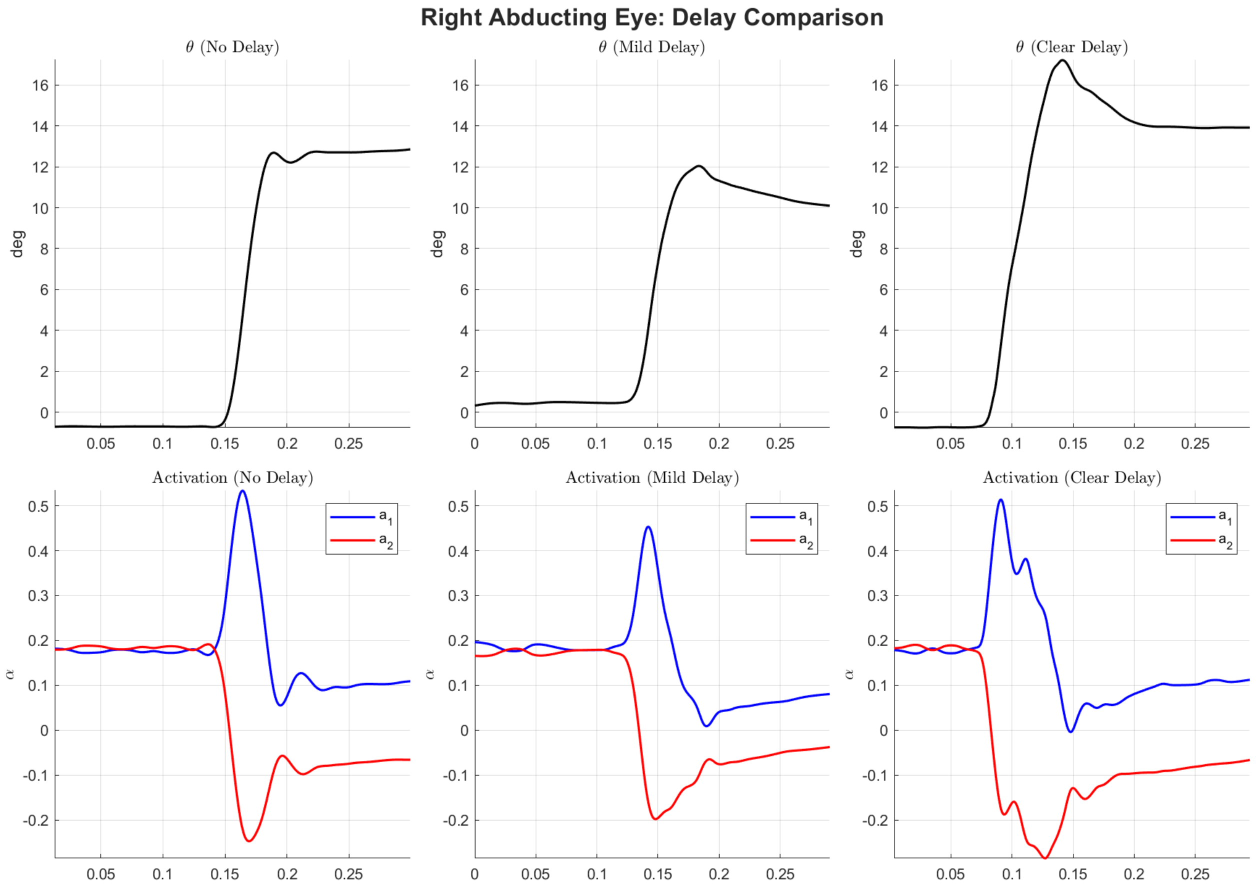


Fig. 18: Right abducting eye position and activation graphs for INO individuals with no obvious delay, mildly obvious delay and clear delay.

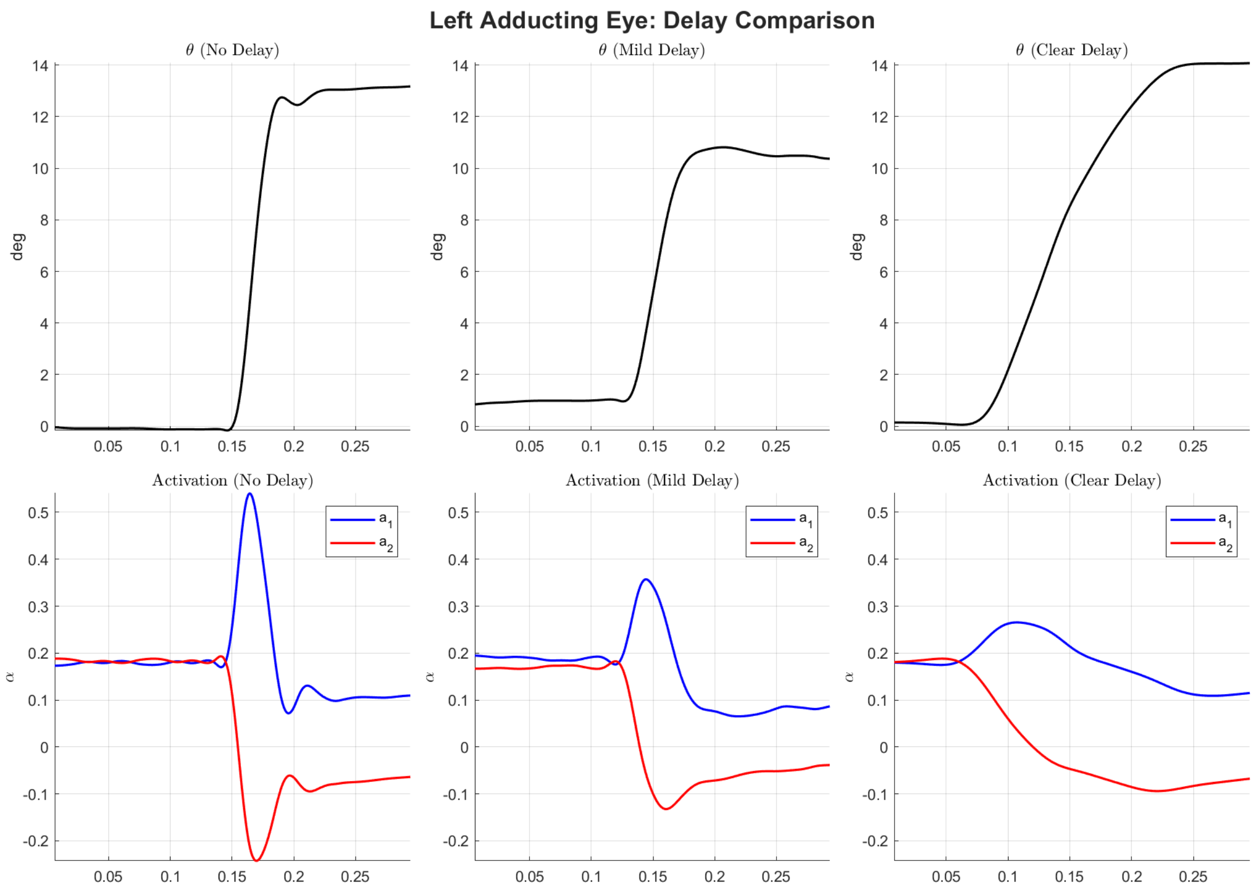


Fig. 19: Left adducting eye position and activation graphs for INO individuals with no obvious delay, mildly obvious delay and clear delay.

F Healthy Versus No Delay

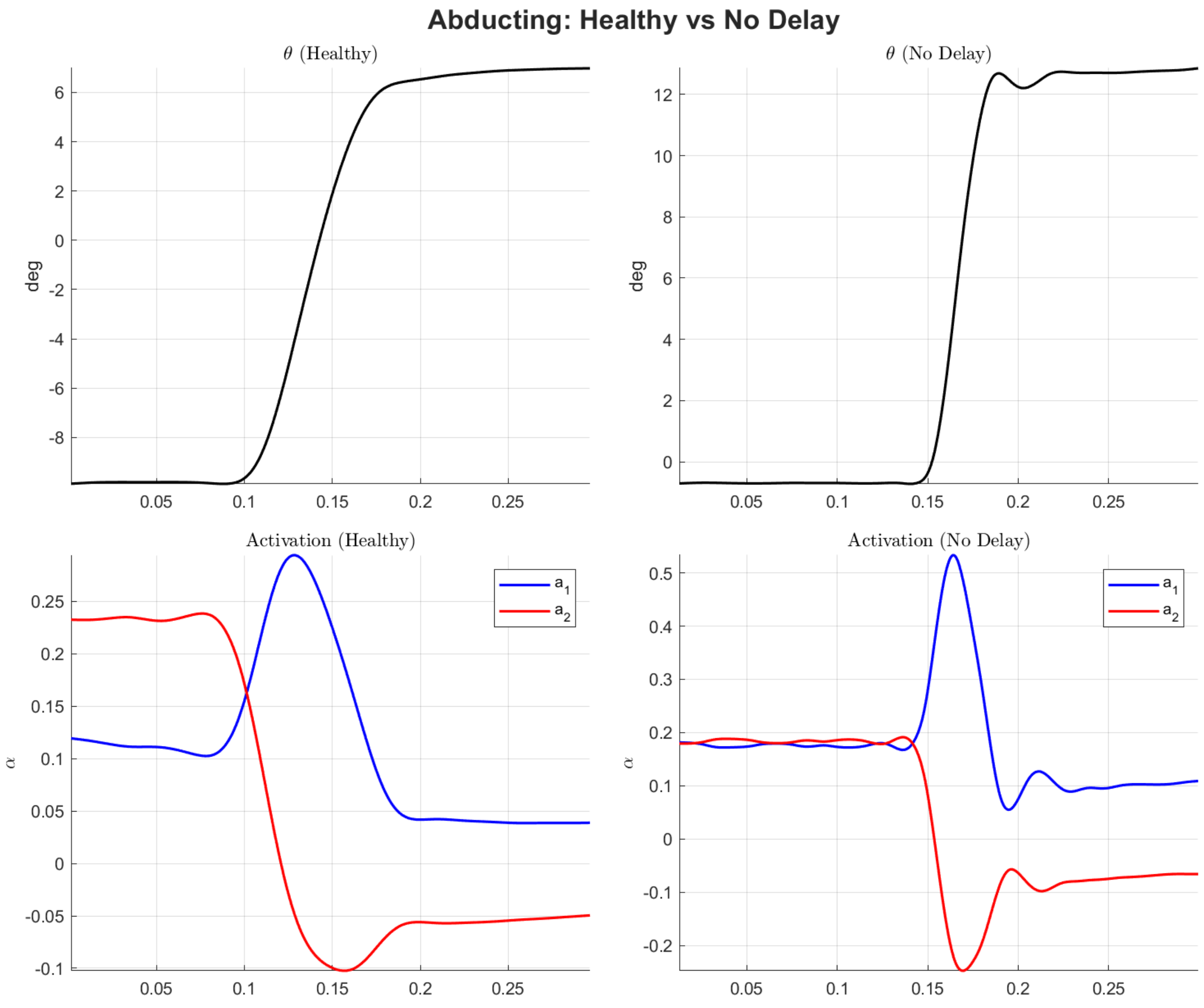


Fig. 20: Abducting eye position and activation graphs for healthy vs. INO individuals with no obvious delay.

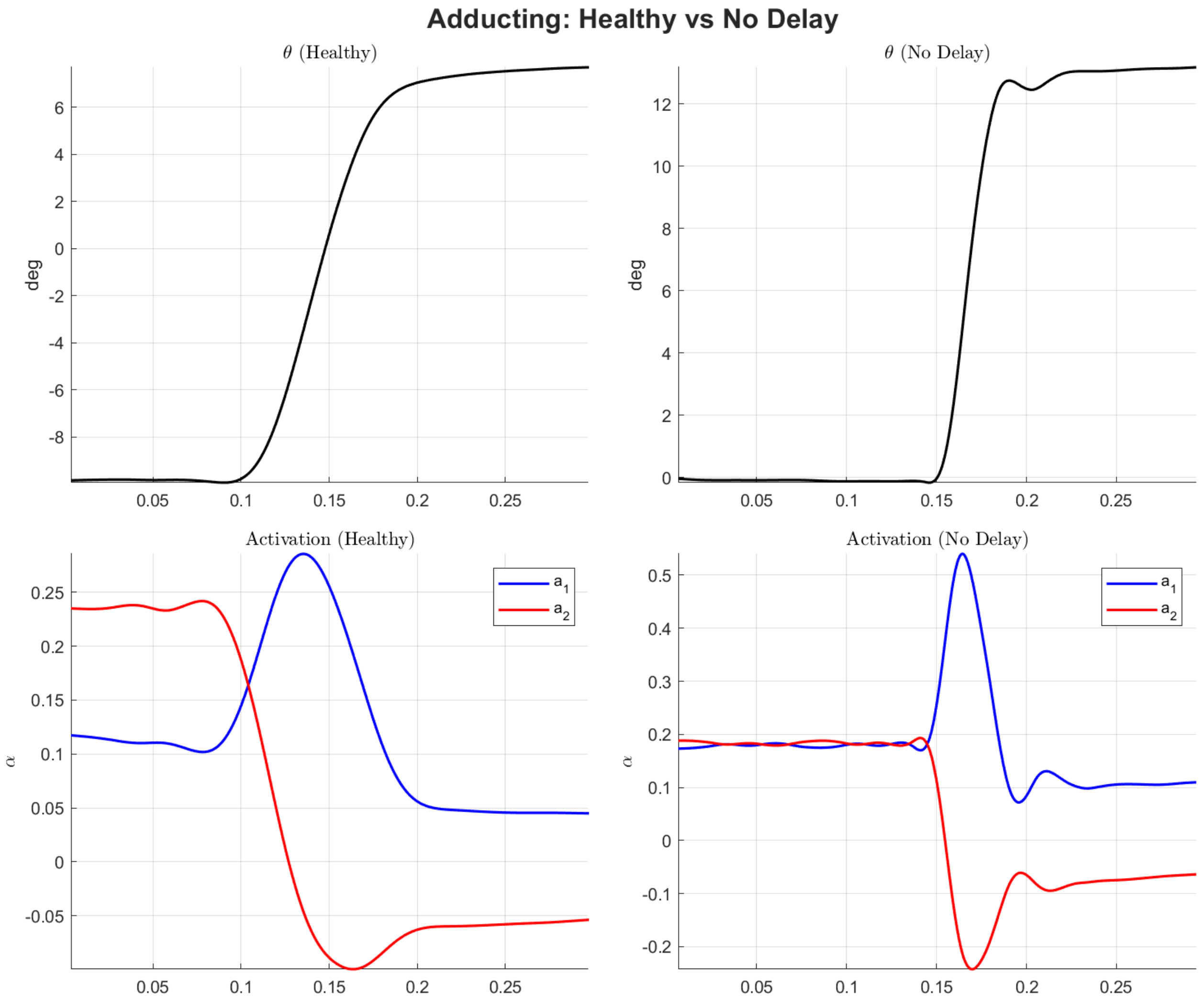


Fig. 21: Adducting eye position and activation graphs for healthy vs. INO individuals with no obvious delay.

G McSpadden's Complete Model

Saccadic eye movement is modelled by a system of first order ordinary differential equations (ODEs) describing eye rotation, muscle dynamics, tendon forces and neural activitations [11]. Forces are reported in the unit of *grams tension*, gt, where 1 gt is the force corresponding to a 1 gram mass subject to gravitational acceleration:

$$1 \text{ gt} = 1 \text{ g} \times 980 \,\frac{\text{cm}}{\text{s}^2}$$

The state vector is defined by

$$\vec{x}^\top(t) = \begin{bmatrix} \theta & \dot{\theta} & l_{m1} & \dot{l}_{m1} & l_{m2} & \dot{l}_{m2} & F_{t1} & F_{t2} & a_1 & a_2 \end{bmatrix}$$

where

$x_1 \; = \theta$ (eye position in degrees),

$x_2 \; = \dot{\theta}$ (eye velocity in degrees/sec),

$x_3 \; = l_{m1}$ (agonist muscle length in cm),

$x_4 \; = \dot{l}_{m1}$ (agonist muscle velocity in cm/s),

$x_5 \; = l_{m2}$ (antagonist muscle length in cm),

$x_6 \; = \dot{l}_{m2}$ (antagonist muscle velocity in cm/s),

$x_7 \; = F_{t1}$ (tendon force of agonist muscle in gt),

$x_8 \; = F_{t2}$ (tendon force of antagonist muscle in gt),

$x_9 \; = a_1$ (agonist muscle activation level from 0 to 1),

$x_{10} = a_2$ (agonist muscle activation level from 0 to 1).

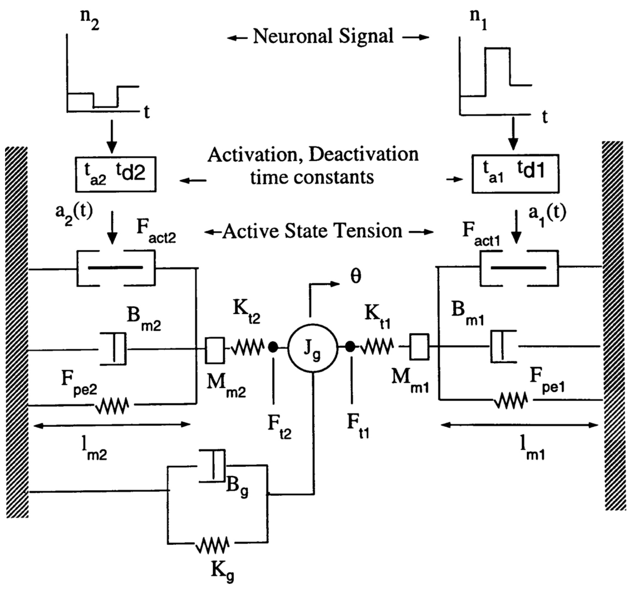


Fig. 22: The eye plant model, taken as screenshot from [11]

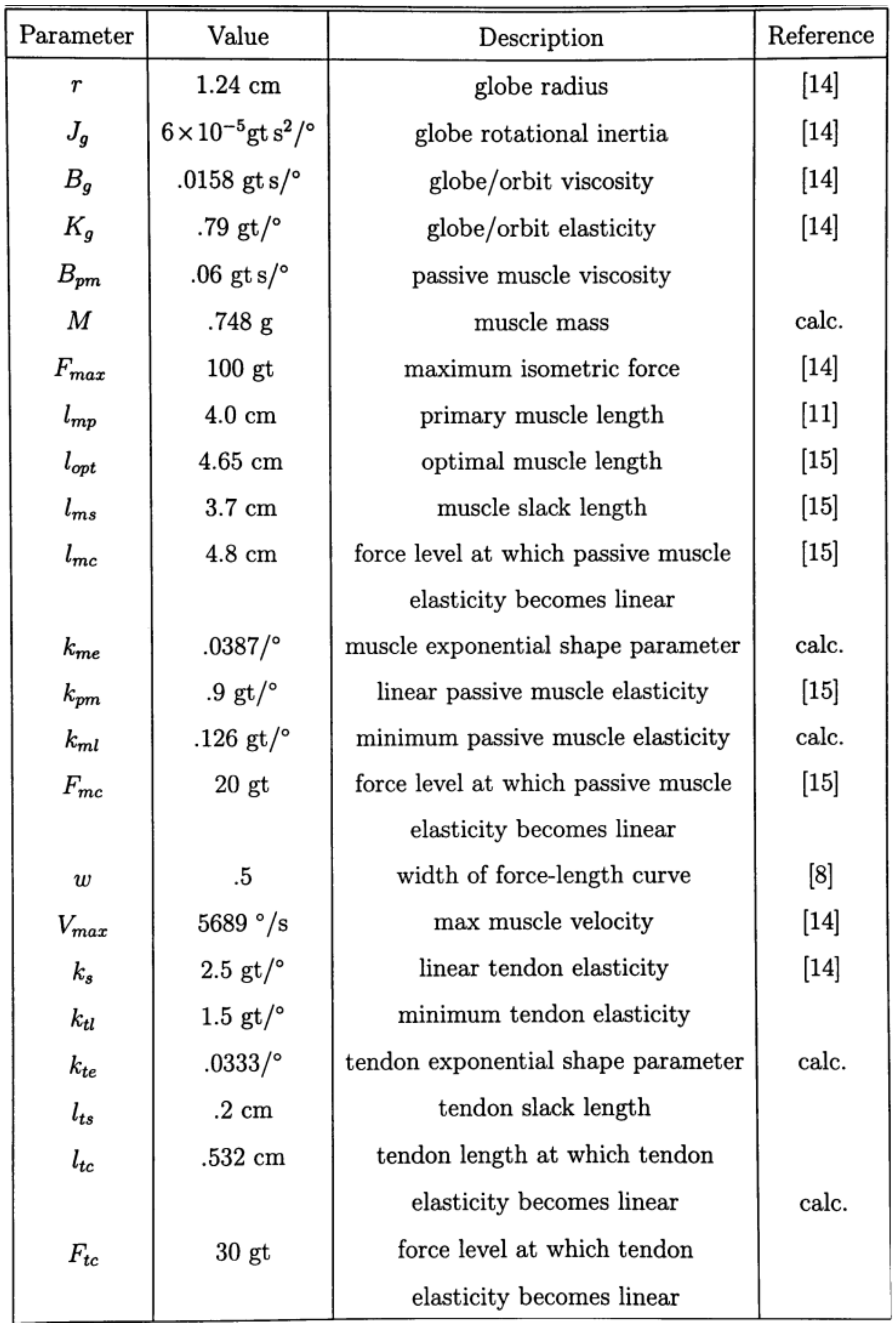
The complete system dynamics are governed by a vector of state equations

$$\dot{\vec{x}}(t) = \vec{f}(\vec{x})$$

where

$$\vec{f}(\vec{x}) = \begin{bmatrix} x_2 \\ \frac{1}{J_G}\left(x_7 - x_8 - B_g x_2 - K_g x_1\right) \\ x_4 \\ \frac{980}{M}\left(x_7 - F_{act}(x_3, x_4, x_9) - F_{pe}(x_3) - B_{pm}\left(\frac{180}{\pi r}\right) x_4\right) \\ x_6 \\ \frac{980}{M}\left(x_8 - F_{act}(x_5, x_6, x_{10}) - F_{pe}(x_5) - B_{pm}\left(\frac{180}{\pi r}\right) x_6\right) \\ K_t(x_7)\left[-x_2 - \left(\frac{180}{\pi r}\right) x_4\right] \\ K_t(x_8)\left[x_2 - \left(\frac{180}{\pi r}\right) x_6\right] \\ \frac{1}{\tau_1(t)}\left[n_1(t) - x_9\right] \\ \frac{1}{\tau_2(t)}\left[n_2(t) - x_{10}\right] \end{bmatrix}.$$

The functions for $F_{act}()$, $F_{pe}()$, and $K_t()$ can be found in [11]; $\tau_1(t)$, $\tau_2(t)$, $n_1(t)$ and $n_2(t)$ functions differ for each type of saccade, but the basic format is provided in [11]. The parameter values used for McSpadden's model and in turn, our model, are listed in Fig. 23.



Parameter	Value	Description	Reference
r	1.24 cm	globe radius	[14]
J_g	6×10^{-5}gt s^2/°	globe rotational inertia	[14]
B_g	.0158 gt s/°	globe/orbit viscosity	[14]
K_g	.79 gt/°	globe/orbit elasticity	[14]
B_{pm}	.06 gt s/°	passive muscle viscosity	
M	.748 g	muscle mass	calc.
F_{max}	100 gt	maximum isometric force	[14]
l_{mp}	4.0 cm	primary muscle length	[11]
l_{opt}	4.65 cm	optimal muscle length	[15]
l_{ms}	3.7 cm	muscle slack length	[15]
l_{mc}	4.8 cm	force level at which passive muscle elasticity becomes linear	[15]
k_{me}	.0387/°	muscle exponential shape parameter	calc.
k_{pm}	.9 gt/°	linear passive muscle elasticity	[15]
k_{ml}	.126 gt/°	minimum passive muscle elasticity	calc.
F_{mc}	20 gt	force level at which passive muscle elasticity becomes linear	[15]
w	.5	width of force-length curve	[8]
V_{max}	5689 °/s	max muscle velocity	[14]
k_s	2.5 gt/°	linear tendon elasticity	[14]
k_{tl}	1.5 gt/°	minimum tendon elasticity	
k_{te}	.0333/°	tendon exponential shape parameter	calc.
l_{ts}	.2 cm	tendon slack length	
l_{tc}	.532 cm	tendon length at which tendon elasticity becomes linear	calc.
F_{tc}	30 gt	force level at which tendon elasticity becomes linear	

Fig. 23: Parameter Values for the Complete Model, taken as screenshot from [11]

H MATLAB Code

The following sections are some of the MATLAB code files that were used to simulate the model in our report.

I. InputData.m

This is the class that defines the `InputData` object which stores all important datapoints and functions loaded in via the constructor.

```
classdef InputData

properties
    file_name
    raw_time_unit {mustBeNumeric}
    raw_data {mustBeNumeric}
    time {mustBeNumeric}
    position {mustBeNumeric}
    t_range {mustBeNumeric}
    fn  % struct containing pos/vel/acc/jerk functions
    display_name string
    tODE1 {mustBeNumeric}
    xODE1 {mustBeNumeric}
    tODE2 {mustBeNumeric}
    xODE2 {mustBeNumeric}
end

methods

function obj = InputData(file_name_, display_name_, raw_time_unit_)
% InputData Constructor
% Inputs:
%   file_name_ - string: path to file
%   display_name_ - string: human-readable label
%   raw_time_unit_ - numeric: time unit multiplier (e.g., 0.001 for ms to s)

% Set default values
if nargin < 3 || isempty(raw_time_unit_)
    raw_time_unit_ = 0.001;  % default: ms to s
end
if nargin < 2 || isempty(display_name_)
    display_name_ = string(file_name_);
end

if ~isfile(file_name_)
    error('File "%s" not found.', file_name_);
end

% Store properties
obj.file_name      = file_name_;
obj.display_name   = display_name_;
obj.raw_time_unit  = raw_time_unit_;
```

```
% Read and validate data
obj.raw_data = readmatrix(file_name_);
if size(obj.raw_data, 2) ~= 2
    error('Input file must have exactly two columns: [time, position].');
end

[obj.time, idx_unique] = unique(obj.raw_data(:,1), 'stable');
obj.position = obj.raw_data(idx_unique, 2);
obj.time = obj.time * raw_time_unit_;

if obj.position(end) < 0
    obj.position = -obj.position;
end

obj.t_range = [obj.time(1), obj.time(end)];

% Constants (imported from constant struct)
C = eyeModelConstants();

% Generate smoothed spline functions for theta and muscle length
thetaLengthFns = getThetaLengthSplines(obj, C);

obj.fn.theta  = thetaLengthFns.theta;
obj.fn.theta_d  = thetaLengthFns.theta_d;
obj.fn.theta_dd  = thetaLengthFns.theta_dd;

obj.fn.l_m1 = thetaLengthFns.l_m1;
obj.fn.l_m2 = thetaLengthFns.l_m2;
obj.fn.l_m1d = thetaLengthFns.l_m1d;
obj.fn.l_m2d = thetaLengthFns.l_m2d;
obj.fn.l_m1dd = thetaLengthFns.l_m1dd;
obj.fn.l_m2dd = thetaLengthFns.l_m2dd;

% Run stepOneModel to compute F_t points
[obj.tODE1, obj.xODE1] = computeStepOneModel(obj.t_range, obj.fn, C);

% Generate smoothed spline functions for tendon force and activation
forceActFns = getForceActSplines(obj.tODE1, obj.xODE1, obj.fn, C);

obj.fn.F_t1 = forceActFns.F_t1;
obj.fn.F_t2 = forceActFns.F_t2;
obj.fn.F_t1d = forceActFns.F_t1d;
obj.fn.F_t2d = forceActFns.F_t2d;

obj.fn.a_1 = forceActFns.a_1;
obj.fn.a_2 = forceActFns.a_2;
obj.fn.a_1d = forceActFns.a_1d;
obj.fn.a_2d = forceActFns.a_2d;

% Run stepTwoModel for validation
[obj.tODE2, obj.xODE2] = computeStepTwoModel(obj.t_range, obj.fn, C);
end

end

end
```

II. eyeModelConstants.m

This is the class that stores the constants used for McSpadden's model and our modified version.

```
classdef InputData

properties
    file_name
    raw_time_unit {mustBeNumeric}
    raw_data {mustBeNumeric}
    time {mustBeNumeric}
    position {mustBeNumeric}
    t_range {mustBeNumeric}
    fn  % struct containing pos/vel/acc/jerk functions
    display_name string
    tODE1 {mustBeNumeric}
    xODE1 {mustBeNumeric}
    tODE2 {mustBeNumeric}
    xODE2 {mustBeNumeric}
end

methods

function obj = InputData(file_name_, display_name_, raw_time_unit_)
% InputData Constructor
% Inputs:
%   file_name_ - string: path to file
%   display_name_ - string: human-readable label
%   raw_time_unit_ - numeric: time unit multiplier (e.g., 0.001 for ms to s)

% Set default values
if nargin < 3 || isempty(raw_time_unit_)
    raw_time_unit_ = 0.001;  % default: ms to s
end
if nargin < 2 || isempty(display_name_)
    display_name_ = string(file_name_);
end
```

```
if ~isfile(file_name_)
    error('File "%s" not found.', file_name_);
end

% Store properties
obj.file_name      = file_name_;
obj.display_name   = display_name_;
obj.raw_time_unit  = raw_time_unit_;

% Read and validate data
obj.raw_data = readmatrix(file_name_);
if size(obj.raw_data, 2) ~= 2
    error('Input file must have exactly two columns: [time, position].');
end

[obj.time, idx_unique] = unique(obj.raw_data(:,1), 'stable');
obj.position = obj.raw_data(idx_unique, 2);
obj.time = obj.time * raw_time_unit_;

if obj.position(end) < 0
    obj.position = -obj.position;
end

obj.t_range = [obj.time(1), obj.time(end)];

% Constants (imported from constant struct)
C = eyeModelConstants();

% Generate smoothed spline functions for theta and muscle length
thetaLengthFns = getThetaLengthSplines(obj, C);

obj.fn.theta  = thetaLengthFns.theta;
obj.fn.theta_d  = thetaLengthFns.theta_d;
obj.fn.theta_dd  = thetaLengthFns.theta_dd;

obj.fn.l_m1 = thetaLengthFns.l_m1;
obj.fn.l_m2 = thetaLengthFns.l_m2;
obj.fn.l_m1d = thetaLengthFns.l_m1d;
obj.fn.l_m2d = thetaLengthFns.l_m2d;
obj.fn.l_m1dd = thetaLengthFns.l_m1dd;
obj.fn.l_m2dd = thetaLengthFns.l_m2dd;

% Run stepOneModel to compute F_t points
[obj.tODE1, obj.xODE1] = computeStepOneModel(obj.t_range, obj.fn, C);

% Generate smoothed spline functions for tendon force and activation
forceActFns = getForceActSplines(obj.tODE1, obj.xODE1, obj.fn, C);
```

```
obj.fn.F_t1 = forceActFns.F_t1;
obj.fn.F_t2 = forceActFns.F_t2;
obj.fn.F_t1d = forceActFns.F_t1d;
obj.fn.F_t2d = forceActFns.F_t2d;

obj.fn.a_1 = forceActFns.a_1;
obj.fn.a_2 = forceActFns.a_2;
obj.fn.a_1d = forceActFns.a_1d;
obj.fn.a_2d = forceActFns.a_2d;

% Run stepTwoModel for validation
[obj.tODE2, obj.xODE2] = computeStepTwoModel(obj.t_range, obj.fn, C);
end

end

end
```

III. load_data_eye.m

This is the file ran to load all the raw data and make into `InputData` class objects.

```
% load_eye_data.m
% Loads all eye movement datasets and stores them in a structured format

% === Define base path to raw data folder ===
data_path = fullfile(pwd, 'raw_data');

% === Load data files ===
data_abduct_healthy     = InputData(fullfile(data_path, ...
    'Matta_abducting_healthy_20.xlsx'), "Abducting Eye, Healthy");
fprintf('Loaded data_abduct_healthy\n');
data_adduct_healthy     = InputData(fullfile(data_path, ...
    'Matta_adducting_healthy_20.xlsx'), "Adducting Eye, Healthy");
fprintf('Loaded data_adduct_healthy\n');

data_abduct_ino         = InputData(fullfile(data_path, ...
    'Matta_abducting_INO_20.xlsx'), "Abducting Eye, INO");
fprintf('Loaded data_abduct_ino\n');
data_adduct_ino         = InputData(fullfile(data_path, ...
    'Matta_adducting_INO_20.xlsx'), "Adducting Eye, INO");
fprintf('Loaded data_adduct_ino\n');

data_abduct_right_none  = InputData(fullfile(data_path, ...
    'No_Delay_Left_Eye.csv'), "Abducting Right Eye, INO (No Delay)");
fprintf('Loaded data_abduct_right_none\n');
data_adduct_left_none   = InputData(fullfile(data_path, ...
    'No_Delay_Right_Eye.csv'), "Adducting Left Eye, INO (No Delay)");
fprintf('Loaded data_adduct_left_none\n');

data_abduct_right_mild  = InputData(fullfile(data_path, ...
    'Mild_Delay_Left_Eye.csv'), "Abducting Right Eye, INO (Mild Delay)");
fprintf('Loaded data_abduct_right_mild\n');
data_adduct_left_mild   = InputData(fullfile(data_path, ...
    'Mild_Delay_Right_Eye.csv'), "Adducting Left Eye, INO (Mild Delay)");
fprintf('Loaded data_adduct_left_mild\n');

data_abduct_right_clear = InputData(fullfile(data_path, ...
    'Clear_Delay_Left_Eye.csv'), "Abducting Right Eye, INO (Clear Delay)");
fprintf('Loaded data_abduct_right_clear\n');
data_adduct_left_clear  = InputData(fullfile(data_path, ...
    'Clear_Delay_Right_Eye.csv'), "Abducting Left Eye, INO (Clear Delay)");
fprintf('Loaded data_adduct_left_clear\n');

% === Preserve insertion order using a manual list ===
ordered_vars = {
    'data_abduct_healthy'
    'data_adduct_healthy'
    'data_abduct_ino'
    'data_adduct_ino'
    'data_abduct_right_none'
    'data_abduct_right_mild'
    'data_abduct_right_clear'
    'data_adduct_left_none'
    'data_adduct_left_mild'
    'data_adduct_left_clear'
};

all_data = struct();
for i = 1:length(ordered_vars)
    all_data.(ordered_vars{i}) = eval(ordered_vars{i});
end

fprintf('Loading complete!\n');
```