A Report on

OCULAR BIOMECHANIC MODEL

by

Φίλιπ Κωνσταντίνος Ειρηναίος
AM: 1003863

Submitted in partial fulfillment of the course requirement in
Modeling and Simulation

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

MSc. Biomedical Engineering
University of Patras

PATRAS 2018

Introduction

Rapid and accurate eye movements are crucial for coordinated direction of gaze. Studying human eye movement has significant implications for improving our understanding of the oculomotor system and treating visuomotor disorders. Over the years, biomechanic simulation has provided an analysis tool of different human movements, especially gait. This analysis can be extended to visual tasks by analyzing the mechanisms of ocular motility and providing a realistic ocular model that can be used to investigate muscle activation patterns during static fixations and the control and dynamics of various eye movements. This model can be used to calculate metabolic costs of eye movements and also simulate different eye disorders, such as different forms of strabismus. Furthermore, it can be easily integrated into other human body models to analyze the relation between vestibular system and eye movement by simulating their biomechanics in three dimensions and investigating the function and neural control of the extraocular muscles (EOMs).

Eye movements are produced through the activation of six extraocular eye muscles (EOMs) for each eye. Clinical trials gave a profound knowledge of how the EOMs rotate the globe, the resistive tension to this rotation and the length-tension relationship of the muscles. Various computational models of the extraocular muscle and orbital mechanics have been proposed, which provide insight and scientific bases for oculomotor biomechanics, control of eye movement and binocular misalignment. These models of oculomotor mechanics focus on the realism of muscle behavior and they were based on the viscoelastic properties and physiological data of the EOMs.

The first 3D biomechanical model was developed by Robinson (1975), who simplified the formulation by only considering the elasticity of the EOMs and ignoring the dynamics. The model incorporates anatomically realistic muscle paths and empirical EOM innervation-length-tension relationships. Two models were developed that improved Robinson's model: Simonsz's model (Simonsz and Spekreijse, 1996) and the SQUINT model (Miller and Robinson, 1984). One major limitation of these biomechanical models is that only static eye positions can be simulated. To study the neural control of rapid saccadic movement, models using anatomical and mechanical properties of EOMs have been developed. They take account that the actual EOM force is a complex nonlinear function of the muscle length, velocity, and innervation. Such models have the advantage of supporting 3D dynamic simulations and have been used to analyze neural controllers and the pulley hypothesis.

Here, we introduce a detailed 3D biomechanical model of the human extraocular eye muscles which can simulate the dynamics of ocular motility. The model is based on the biomechanical simulator OpenSim (Delp et. al. 2007). This is an open-source software that can provide the flexibility of in depth parameterization of the model. The architecture and dynamic muscle properties are based on physiological and kinematic measurements of the human eye. The model incorporates an eye-globe, orbital suspension tissues and six muscles with their connective tissues to test the passive pulley hypothesis. Furthermore, with this model we are able to assess the excitation and activation patterns for a variety of targets by applying a closed-loop fixation controller that drives the model to perform saccadic movements in a forward dynamic manner. The controller minimizes the error between the desired trajectory (reference input) and the predicted movement (calculated output).

1. Architecture of the Model

Construction of Eyeball

The orbital plant consists of the globe (eyeball), three pairs of extraocular muscles, and connective tissues. The size of an emmetropic human adult eye is approximately 24.2 mm (transverse, horizontal) × 23.7 mm (sagittal, vertical) × 22.0–24.8 mm (axial, anteroposterior) with no significant difference between sexes and age groups. In the transverse diameter, the eyeball size may vary from 21 mm to 27 mm. Thus, it can be approximated by a solid sphere with 12mm radius. The eyeball was constructed in Blender, an open-source software 3D creation Software. We used a spherical mesh with 32 segments and 12 rings, to construct the vitreous humor (body) as solid sphere and a conical plate to construct the cornea. The weight of an average human eye is 7.5grams and the moment of inertia can be calculated similarly as in the case of a spherical homogenous and isotropic object with radius 12mm ($I = \frac{2}{5}MR^2$ at the center of mass).

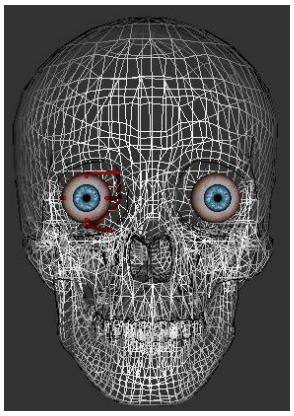


Figure 1. Eye model constructed in OpenSim.

Muscle Paths and Pulleys

The six EOMs, including four rectus muscles and two oblique muscles, are controlled by the cranial nerves to generate force, to rotate the globe to track a visual target, and to stabilize the image of the object. The four rectus EOMs originate from the annulus of Zinn; they course anteriorly, pass through Tenon's capsule, and insert on the sclera. The lateral rectus (LR) and medial rectus (MR) muscles form an antagonistic pair to produce horizontal eye movements. The superior rectus (SR) and inferior rectus (IR) muscles form the vertical antagonist pair, which mainly controls vertical eye movement and also affects rotation about the horizontal plane and the line of sight (secondary action) due to insertion positions and the path of the muscles. The superior oblique (SO) muscle passes through the cartilaginous trochlea attached to the orbital wall, which reflects the SO path by 51^{o}. The inferior oblique (IO) muscle originates from the orbital wall anteroinferior to the globe center and inserts on the sclera posterior to the globe equator. The primary actions of SO and IO cause rotation of the globe around the visual axis and vertical movement. A description of primary, secondary and tertiary actions of the EOMs is shown in the Table.1.

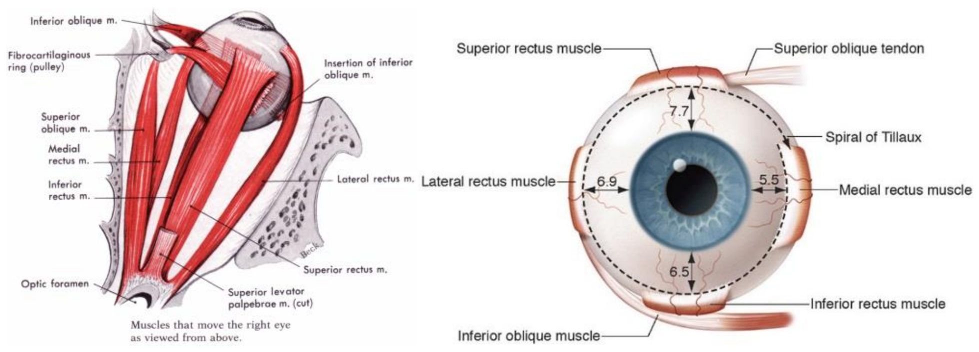


Figure 2. Anatomy of the human eye and extraocular muscles.

Table 1. Description of Primary, Secondary and Tertiary actions of EOMs.

Muscle	Primary action	Secondary action	Tertiary action
Lateral Rectus (LR)	Abduction	–	–
Medial Rectus (MR)	Adduction	–	–
Superior Rectus (SR)	Supraduction	Incycloduction	Adduction
Inferior Rectus (IR)	Infraduction	Excycloduction	Adduction
Superior Oblique (SO)	Incycloduction	Infraduction	Abduction
Inferior Oblique (IO)	Excycloduction	Supraduction	Abduction

Based on evidence from MRI and computerized 3D reconstruction, Miller (1989) first proposed that the rectus muscle connective tissues posterior to the globe equator, called "pulleys", couple rectus EOMs to the orbital wall and constrain the transverse shifts of muscle paths even in extreme eye positions. The pulleys restrict the transverse sideslips of the muscles, which lead to muscle path inflections in secondary and tertiary gazes. Pulleys are believed to have significant implications on the kinematics and neural control of the orbital plant. Also, it is supported by several experiments that the Listing's law describing torsional eye movements is implemented in the muscle pulleys and not via brain-level control.

To our knowledge, there are two theories behind pulley functionality. Passive pulley theory states that the pulleys have fixed to the orbit pulley points, whereas the active pulley theory states that the pulleys move posteriorly when the rectus muscles contract and anteriorly with relaxation. In reality, passive pulleys move, but only as a consequence of their compliant suspensions responding to the transverse forces produced by deflection of the EOMs sliding freely through their sleeves. The passive pulley hypothesis applies only to rectus EOMs and not to oblique muscles.

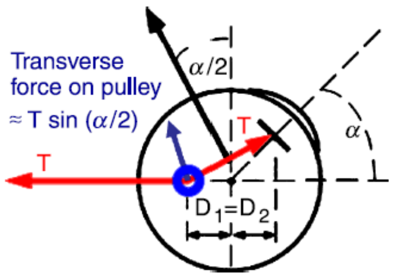


Figure 3. A muscle (red line) passing freely through a pulley (blue ring), exerts a transverse force (dark blue arrow) on the pulley that depends on its angle of deflection a/2, where a is the eye's angle of eccentricity.

We have chosen the passive pulley model for our extraocular model in order to keep it simple and provide faster simulation speed. Table.2 shows the positions of muscle pulleys, as well as the origin and insertions points of the EOMs onto the eye, with respect the center of the globe. These data were acquired from [2] which were based on physiological measurements, but in our case, they were slightly modified to be located outside of the globe radius in order to not penetrate the eye globe. Since no position was documented for the origin of the SO, a point close to the origins of the rectus muscles was chosen to match the fiber length in the primary position of the SO muscle.

Table 2. Origin, Pulley and Insertion Points with respect the center of the globe. All measures are given in meters.

	Origins			Pulleys			Insertions		
	Ox	Oy	Oz	Px	Py	Pz	Ix	Iy	Iz
LR	-0.034	0.0006	-0.013	-0.0102	0.0003	0.012	0.0065	0	0.0101
MR	-0.030	0.0006	-0.017	-0.0053	0.00014	-0.0146	0.0088	0	-0.0096
SR	-0.0317	0.0036	-0.016	-0.0092	0.012	-0.002	0.0076	0.0104	0
IR	-0.0317	-0.0024	-0.016	-0.0042	-0.0128	-0.0042	0.00805	-0.0102	0
SO	0.0082	0.0122	-0.0152	-0.030834	0.001145	-0.01644	-0.0044	0.011	0.0029
IO	0.0113	-0.0154	-0.0111	-0.00718	-0.0135	0	-0.008	0	0.009

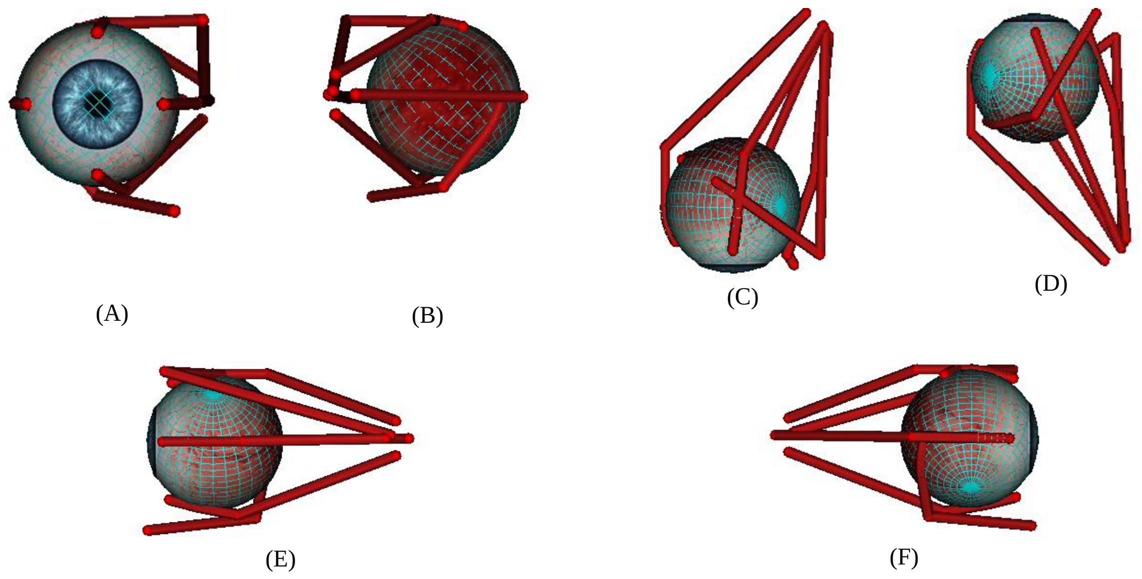
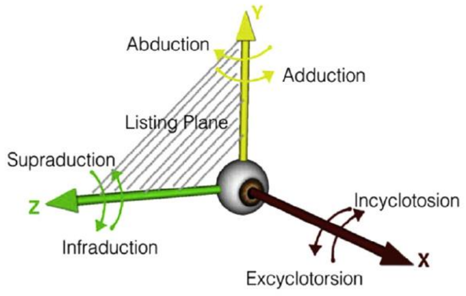


Figure 4. Constructed Eye Model. (A) Frontal View (B) Posterior View. (C) Top View, (D) Bottom View. (E) Lateral View. (F) Medial View

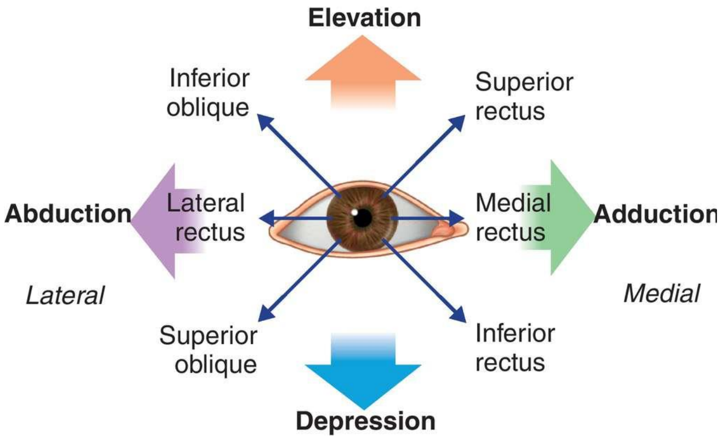
Coordinate System

The eye performs rotational movements around one of three axes passing through the center of rotation and perpendicular to one-another. Movement of the eye nasally is called adduction and temporally, abduction. Supraduction or elevation and infraduction or depression correspond to the movement of the eye up or down in the vertical direction. Torsional eye movements rotate the eye around the visual axis. A nasalward rotation is called incyclotorsion or intorsion and a temporalward rotation is called excyclotorsion or extorsion. Transitional movements of the globe are assumed negligible.

The eye-globe is modelled as a ball-socket joint with a range of rotation angles from -40^o to 40^o in all three axes. The vertical and the horizontal axes are assumed to lie in the Listing's plane, which is defined as a plane fixed in the orbit that passes through the center of rotation and the equator of the globe when the eye is in the primary position and perpendicular to the fixation line. The primary position can be defined as the position when the head is erect and looking straight forward. The positive and negative values of the rotation angles are based on the OpenSim notation. Thus, adduction movements (rotation around Y-axis) have negative angle values, and accordingly abduction movements have positive. Rotation upwards in the vertical direction (rotation of Z-axis) leads to positive values and rotation around the visual axis (X-axis) has negative angle values for incyclotorsion.



(A)



(B)

Figure 5. (A) Coordinate system in OpenSim. (B) EOMs actions

If the functional goal of an eye movement is to stabilize binocular gaze and keep images in a particular depth plane stable on the foveae, such as during pursuit of a small moving target, it suffices to control the two degrees of freedom of gaze direction (of each eye), leaving ocular torsion unspecified. Among the infinite possible 3-D orientations which the eyeball could adopt, it is now well-established that the torsional orientation of the eye is uniquely specified by gaze direction and vergence, a strategy which is appropriate for foveal and stereo vision. Accordingly, both smooth pursuit and saccadic eye movements obey robust kinematic constraints known as Listing's law.

Listing's law states that, when the head is fixed, there is an eye position called primary position, such that the eye assumes only those orientations that can be reached from primary position by a single rotation about an axis in a plane called Listing's plane. This plane is orthogonal to the line of sight when the eye is in primary position. In other words, one can visualize any given eye movement as caused by rotation about an axis that lies in the Listing's plane. Listing's law holds during fixation, saccades, and smooth pursuit.

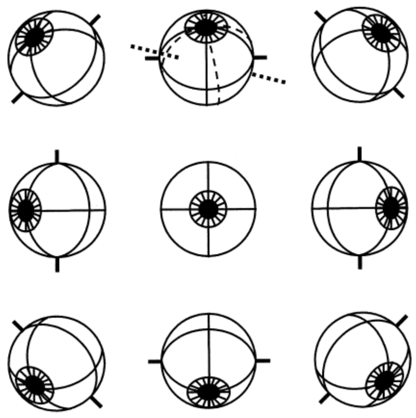


Figure 6. The nine orientations according to Listing’s law, attained by rotating from primary position about axes lying in Listing’s plane.

Whenever gaze changes direction during saccades or smooth pursuit, the axis about which the eye actually rotates tilts in an eye-position-dependent manner out of Listing's plane with the effect of keeping ocular orientation aligned with Listing's plane. This important kinematic effect has been called the velocity criterion of Listing's law and is usually referred to as the `half-angle rule'. If the eye starts its rotation from an eccentric eye position, then in this situation, the orientation of the eye is still determined by rotation about axes that lie in a plane (irrespective of the direction of movement), but this plane is no longer orthogonal to the line of sight; instead it is tilted in the same direction as the line of sight but only half as much. This relationship between rotational axes and gaze angle is called Listing’s half-angle rule. To summarize, Listing’s law can be expressed in terms of any initial eye position, not just primary position.

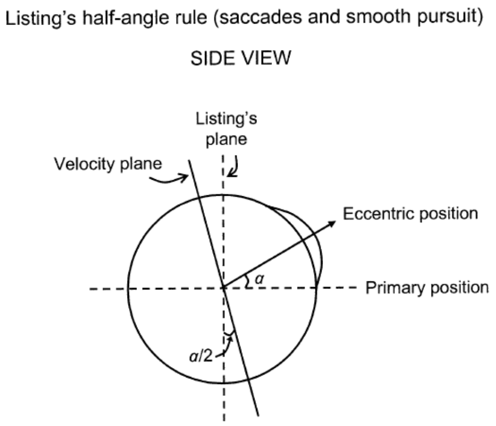


Figure 7. Listing’s half-angle rule for saccades and smooth pursuit.

Passive pulleys make axes of rotation a function of gaze. Appropriately located passive pulleys would cause a muscle’s axis to tilt by half of the angle of eye rotation. In this connection, 1/2 is a “magic number” because Listing’s law is satisfied if the axis of rotation shifts by 1/2 of a shift in eye orientation. However, the passive pulley hypothesis violates the Listing’s law in tertiary gaze positions. So, we will only examine the behavior of our model in the secondary gaze positions.

Muscle Model

In contrast with the skeletal muscles, EOMs are bilaminar and consist of two layers with different fiber types, the global and the orbital layers. This complicated and special architecture contributes to the realization of both fixations and rapid eye movements up to 900°/sec and requires cooperative control of the EOMs involving very complex neural circuits. Here, for simplicity, we do not model this separation into two layers. The Hill-type muscle constitutive model (Hill, 1938) widely applied in describing the force generation mechanism of skeletal muscles has been adopted to describe extraocular muscle mechanics. It consists of an active contractile element (CE), a series elastic element (SE), and a parallel elastic element (PE).

- For the CE, the active force dynamics rely on the activation dynamics, the active force-length relationship and the force-velocity relationship, given by the equation:

$$F_{CE} = a f_{FL}(l) f_{FV}(v)$$

where $a \in [0,1]$ is the muscle activation, with $a = 1$ meaning the muscle is fully innervated, f_{FL} is the force-length relationship, f_{FV} is the force-velocity, l is the muscle length and v is the contraction velocity. Both f_{FL} and f_{FV} have nonlinear characteristics.

Activation dynamics are modelled using a first-order differential equation to relate the rate of change in activation to excitation (i.e. the firing of the motor units):

$$\dot{a} = \frac{(x^2 - xa\ )}{\tau_{rise}} + \frac{(x - a)}{\tau_{fall}}$$

where, α is the activation level, x is the excitation level of muscle, and τ_{rise} and τ_{fall} are the rise and fall time constants for activation, respectively.

- PE models the passive muscle force as a function of muscle length, characterized by the passive FL curve.

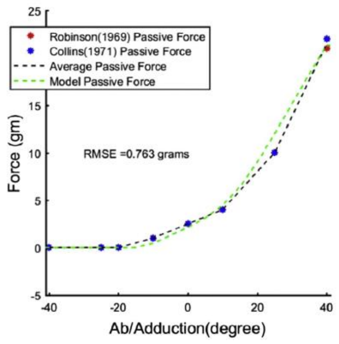
$$F_{PE} = f_p(l)$$

where l is the muscle length and f_p is the passive force-length relationship.

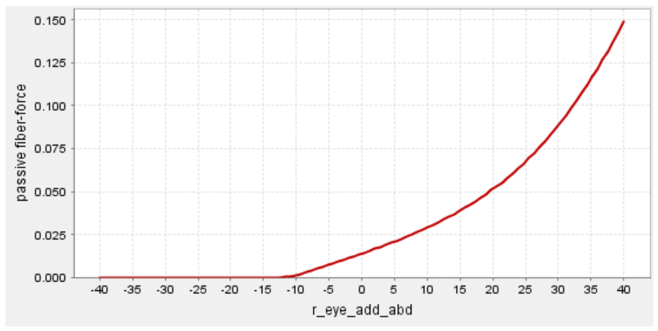
- SE represents the passive muscle force from elastic elements in series with CE, contributed by tendons and connective tissues within muscles. None of the existing computerized ocular models has included the SE component, since the SE behavior has not been quantified yet due to the lack of physiological data. We follow the previous models and we only include the contractile and parallel element in the EOM model.

We use the Millard model (based on the Hill-type model) implemented in OpenSim (Millard, Uchida, Seth & Delp, 2013), which allows to manually fit force-generation dynamics. The six EOMs were modelled using the rigid tendon Hill-type muscle model, that ignores the elasticity of the tendon. This means that the series element (SE) of the Hill muscle model is not included. This assumption is valid when the ratio of the tendon length to the muscle length is less or equal to one, as the in the case of all EOMs. EOMs are considered parallel-fibered muscles, so the pennation angle $\alpha = 0$, and slack length was substituted with the tendon length as we used rigid tendon assumption. Maximum isometric force, optimal fiber length, primary position length and tendon length were adopted from [2], where primary position length was used as default initial length of the EOMs in the simulations.

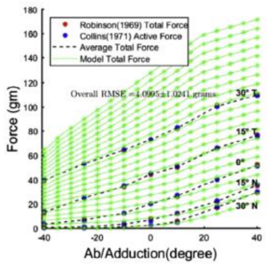
The parameters that describe the above relationships were chosen to fit the curves found in [2]. In the plots below, we try to match the force-length relationships at maximum activation of the lateral rectus muscle. This represents the first part of testing the fidelity of the model. Due to lack of data describing the other muscles' force-length relationships, we used the parameters found describing the normalized curves of active and passive force-length relationships of the LR for the other EOMs as well.



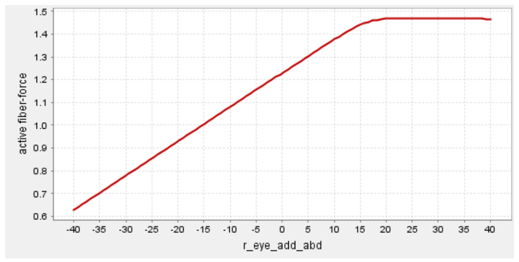
(A) Theoretical Passive F-L curve



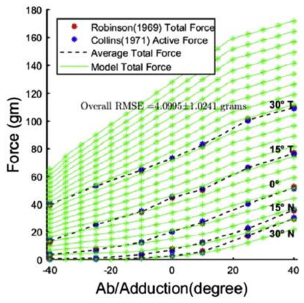
(B) Simulated Passive F-L curve



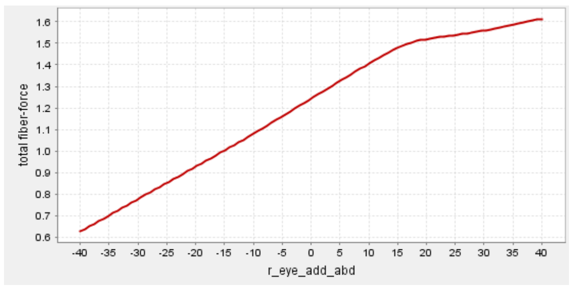
(C) Theoretical Active F-L curve



(D) Simulated Active F-L curve



(E) Theoretical Total F-L curve



(F) Simulated Total F-L curve

Figure 8. (A), (C) and (E) show the Passive, Active and Total force-length curve of Lateral Rectus muscle exerted in different activation levels (green lines). (B), (D) and (F) the corresponding approximated force-length relationships calculated for the Millard LR muscle for maximum activation (a=1).

Due to different histological structures of EOMs compared to skeletal muscles, EOMs have a higher fraction of fast twitch fibers and thus different force-velocity behavior. Models of f_v have been proposed based on Hill's FV curve of skeletal muscles, experimental data on rat EOMs, and maximum saccadic velocity of human eyes (Robinson, 1981). However, in our model the parameters explaining the changes in force-velocity slopes, are left as default in the Millard model, since the behavior of the selected Hill-type muscle model depends mainly on the maximum contraction velocity. In OpenSim, the maximum muscle contraction velocity is given in optimal fiber length per seconds and thus it was set individually for each EOM to reach $900^o/s$ when each muscle contracts, which is consistent with the peak velocity of saccadic eye movements.

Also, because of different structure of neural control of eye movements, activation and deactivation delays are lower than in skeletal muscles, at about 5ms.

Wrapping Surfaces

Wrapping objects help to simulate the proper dynamics. Using wrapping objects implemented in the OpenSim allows us to model realistic force directions as the muscle runs over the eyeball. From the point of origin to the pulley point, the force exerted by the contraction of the muscles is aligned along one straight line. But, from the pulley point to the insertion point, the force is distributed along the surface of a sphere. Two separate wrapping spheres for the rectus muscles and the oblique muscles were created, to avoid abnormal changes on the fiber-length curve as the eyeball rotates in the three directions.

Suspensory Tissue Tension

The passive connective tissues of the eyeball apply a restoring force, which brings the globe back to the central position when the net force from the EOMs is zero. These tissues include all non-muscular suspensory tissues, such as Tenon's capsule, the optic nerve, the fat pad and the conjunctiva. The force-displacement curve of the net elasticity can be represented as:

$$P = k_p q + k_c \cdot 10^{-4} q^3$$

where q is the degree of rotation and k_p, k_c are coefficients found in the literature as $k_p = 0.33\ gm/degrees$ and $k_c = 1.56\ gm/degrees^3$. These forces serve the eye's stabilization. In the proposed model, a limiting spring generalized force is used at each joint to act as the orbital suspension tissues. However, since we used the built-in functionality of the OpenSim, we neglected the cubic term, but it should not result in a big change in the model dynamics. For example, at $20°$ eye rotation the contribution of the cubic component is around 11% of total elasticity force.

2. Fixation and Saccadic movements

Fixation holds the image of an immobile object on the fovea while the head is steady, while a saccade changes the direction of gaze to capture and stabilize the image of an object on the fovea rapidly. In order to validate the model, we tested its ability to generate saccadic movements and lock on a variety of fixation points. Since measurements of neural saccade commands in humans are not available, the neural signals of saccadic movements can be estimated by constructing a controller that minimizes the tracking error between the desired trajectory and the simulated movement of the eye. A scheme of the closed-loop system can be seen in Figure.9. A simple PD controller was used for each antagonist pair of EOMs, which feeds the sum of the position error of rotation angle multiplied by a factor K_p and the angular velocity error multiplied by a factor K_v into the model. Depending on the sign of the error term, one EOM of the antagonist pair is activated, while the other remains inactivated. The output of the controller is saturated so that it is kept in the range of [0,1]. The output of the controller can be interpreted as the excitation levels of the muscle activations.

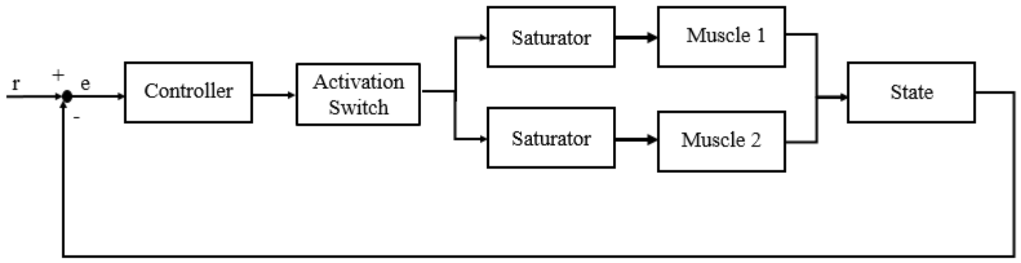


Figure 9. Scheme of the controller applied to drive the model to make saccadic movements.

We tested the model to perform saccadic movements starting from 0^o to -20^o, -30^o, 10°, 30°, $-10°$ and finally 0° for horizontal and vertical rotation angles with duration of 200ms for each fixation point and 50msec as the transition time between two fixation points, leading to varying velocities of $-400°/s$, $-200°/s$, $600°/s$, $200°/s$, $400°/s$, $-800°/s$ and $200°/s$ accordingly for each saccade. This opposes the documented duration of saccade movements found in [10] that states that the normal duration of a 15° and a 30° saccade is about 55 and 80 msec respectively. But since saccades can reach up to $900^o/s$, we tried to evaluate the performance of the model in high velocity scenarios. The total simulation lasts 4 seconds. At the fixation point the velocity is held at $0°/s$, while torsion was tried to be kept approximately zero throughout the simulations to obey Listing's law in secondary positions. The results of the simulation can be seen in Figures.10-12, where we show the rotation angles (the red line shows the desired trajectory and the blue shows the simulated movement) and the rotation velocities in the three axis, and the excitation and the resulting activation levels of the modeled EOMs.

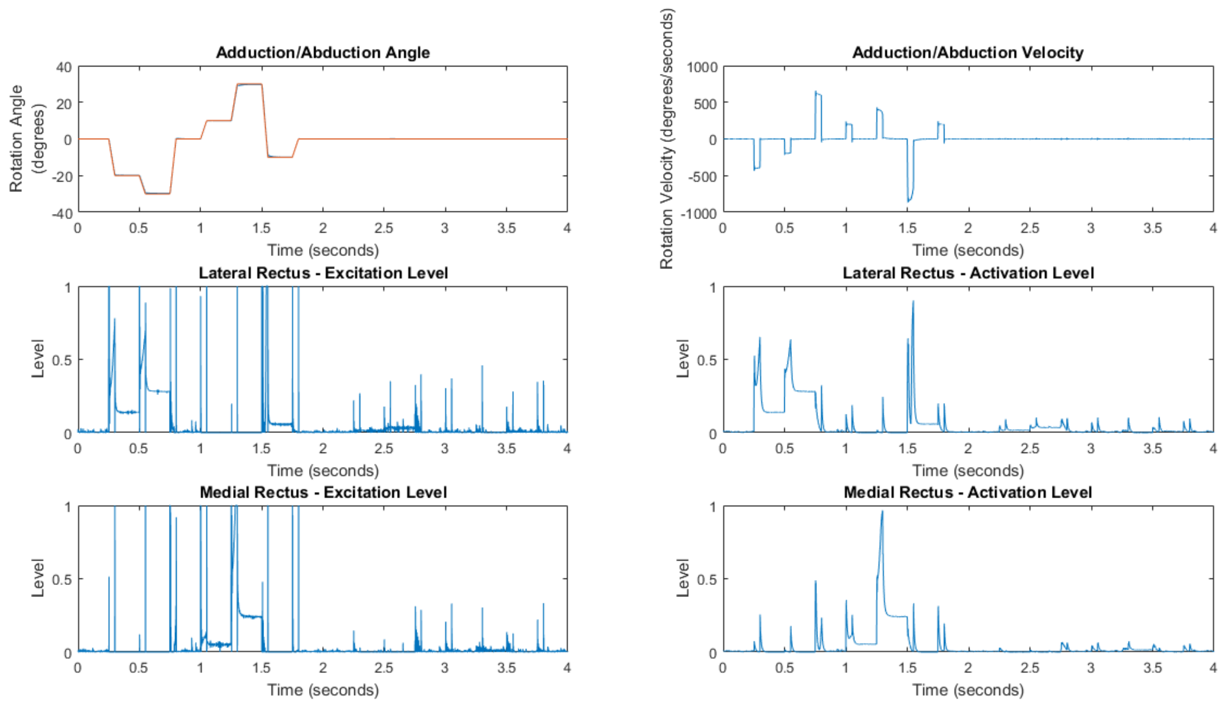


Figure 10. Simulations and Analyses of saccades and fixation movements for Abduction/Adduction rotation angles and velocities, and excitation and activation levels for Lateral and Medial Rectus muscles.

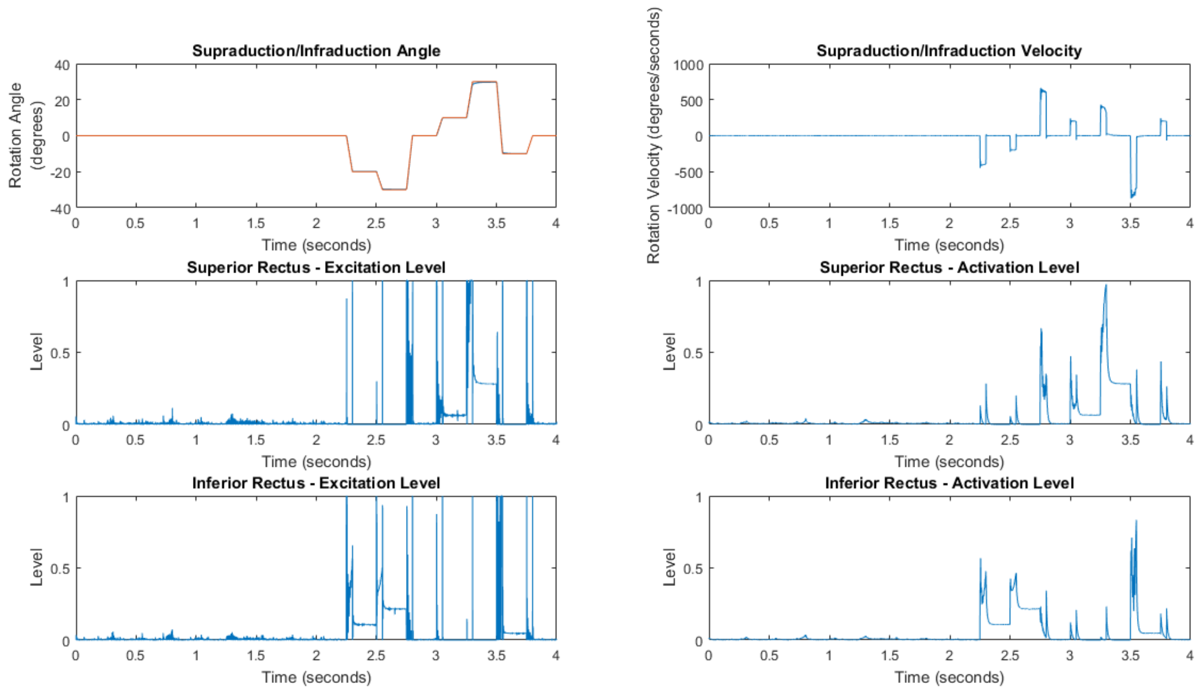


Figure 11. Simulations and Analyses of saccades and fixation movements for Supraduction/Infraduction rotation angles and velocities, and excitation and activation levels for Superior and Inferior Rectus muscles.

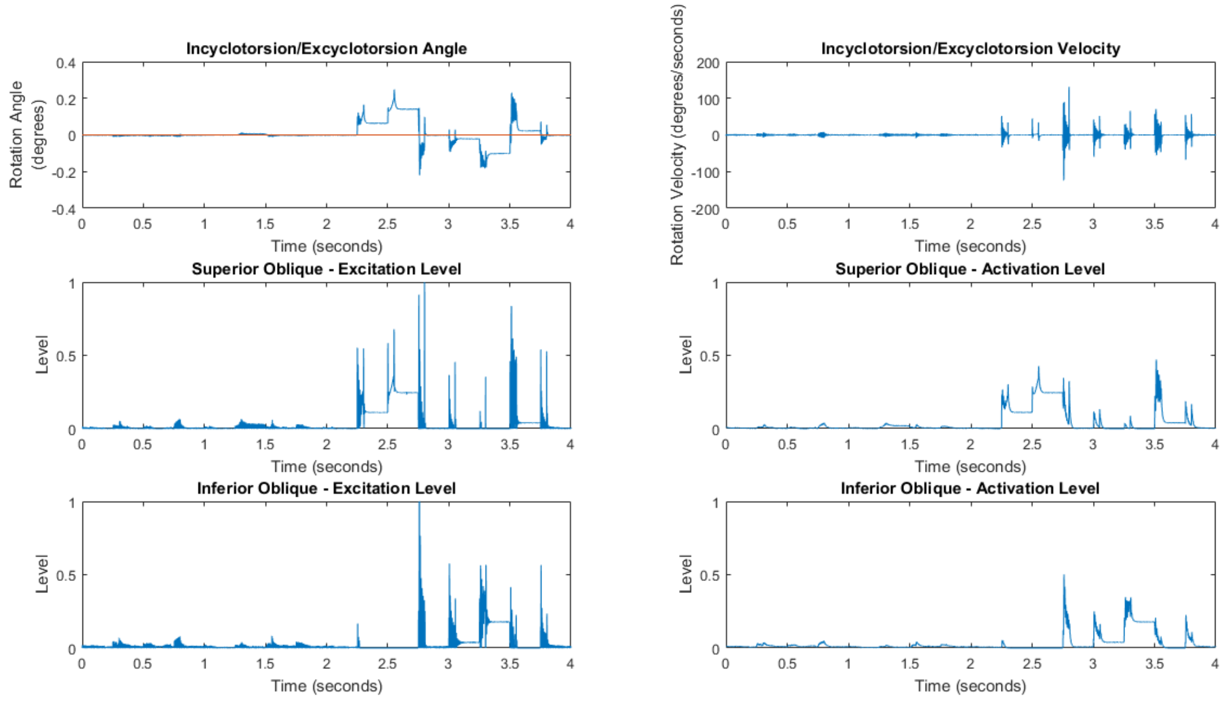
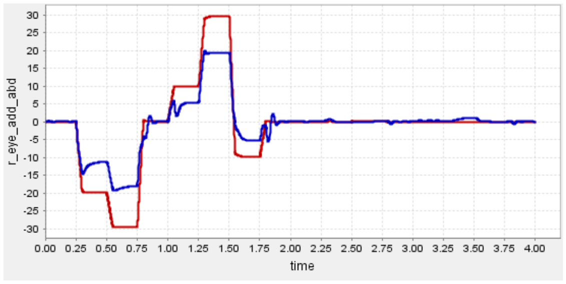


Figure 12. Simulations and Analyses of saccades and fixation movements for Incyclotorsion/Excyclotorsion rotation angles and velocities, and excitation and activation levels for Superior and Inferior Oblique muscles.

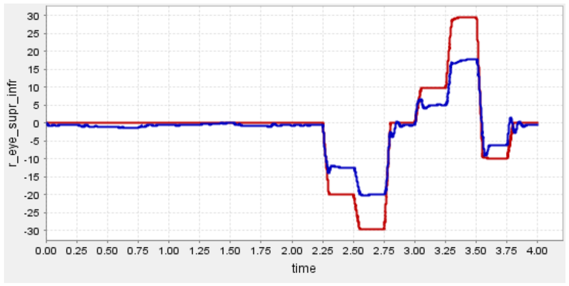
The parameters of the PD controller were found after trial and error, by trying to minimize transition and settling time and also reduce undesired oscillations in the simulated outputs and excitation levels when the muscles are not activated. The optimal parameters of the PD controller found were $K_p = 50$ and $K_v = 1.5$ for the rectus muscles and $K_p = 100$ and $K_v = 0.5$ for the oblique muscles. Decreased values of K_p lead to increased rise time and settling time, whereas increased values of K_p caused “noisy” excitation levels.

The results show that the simulated saccade reasonably follows the desired saccadic trajectory, indicating a satisfactory performance. Also, note that the computed innervations realistically model the behavior of antagonist motoneurons during “off” direction saccades. Most of them completely cease firing after saccade onsets. Small activation levels during inactivation of the muscles are mostly attributed to the suspensory tension of the connective tissues in the orbit. In order to evaluate the performance, we measured the root mean square (RMSE) in each rotation direction. The RMSE for adduction/abduction is $0.2792°$ and for supraduction/ infraduction is $0.2429°$. The RMSE for incyclotorsion/excyclotorsion for the whole duration of the simulation is $0.0554°$, while the minimum rotation angle achieved is $-0.2197°$ and the maximum rotation angle is $0.2477°$. Thus, the associated torsion is very small, which further demonstrates the accuracy of the nonlinear EOM model in reproducing saccades. However, the torsion maintains a constant offset around zero for large rotation angles. The maximum offset found was about $0.14°$.

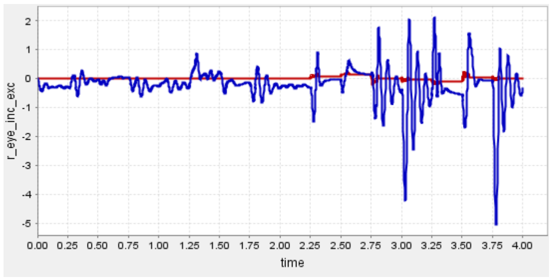
Moreover, we compared the simulated results of the fixation controller with the case where the activation levels are calculated by the static Optimization Tool in OpenSim. The static optimization tool computes the activation of the muscles to perform the desired kinematics. Figure.13 shows the calculated rotation angles (in blue) after applying forward simulation with the controls computed in the static optimization tool, compared with the simulated movement with the application of the fixation controller (in red).



(A) Adduction/Abduction



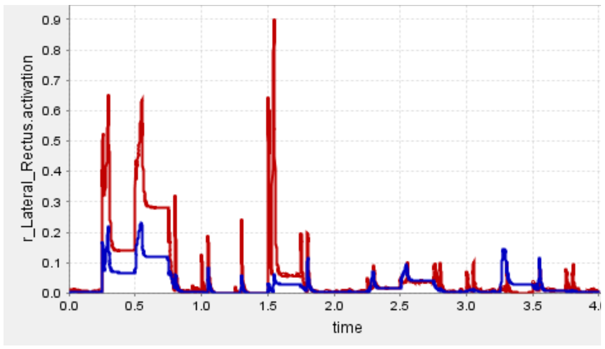
(B) Supraduction/Infraduction



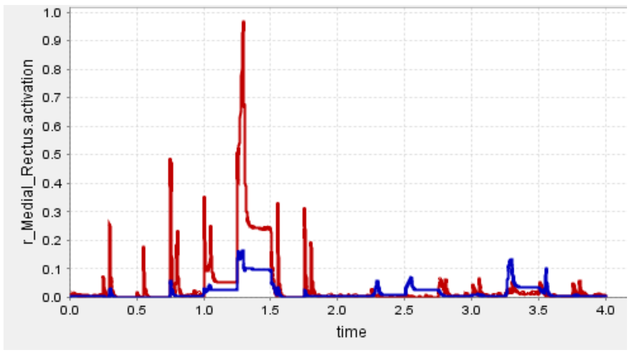
(C) Incyclotorsion/Excyclotorsion

Figure 13. Rotation angles with the application of PD controller (shown in red) and forward simulation with excitation levels computed with the Static Optimization tool (shown in blue).

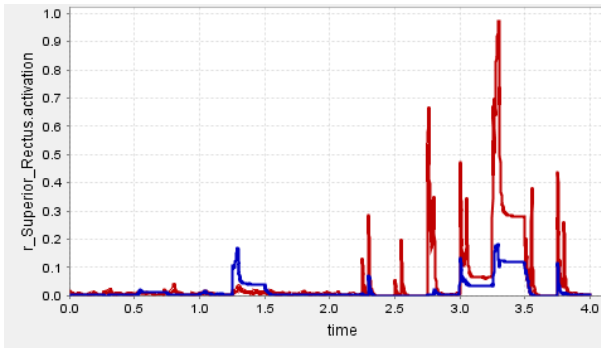
In all movement directions, the optimization tool fails to achieve satisfactory performance by showing increased deviation from the desired trajectory for the horizontal and vertical movements and increased oscillations for the torsional movements. Accordingly, the activation levels of the six EOMs, shown in Figure.14, despite of displaying similar behavior with the corresponding ones computed with the PD controller, they have decreased magnitude overall, thus failing to innervate the muscles to reach the desired fixation point.



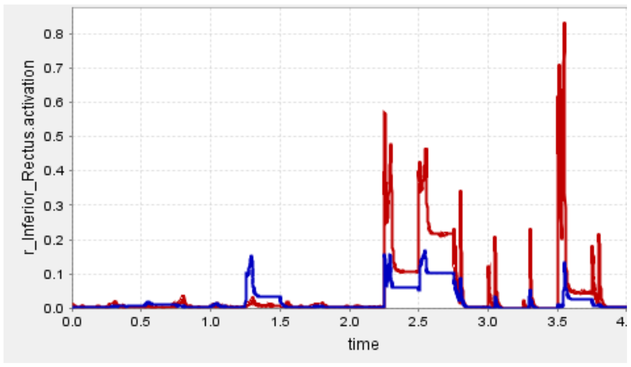
(A) Lateral Rectus activation level



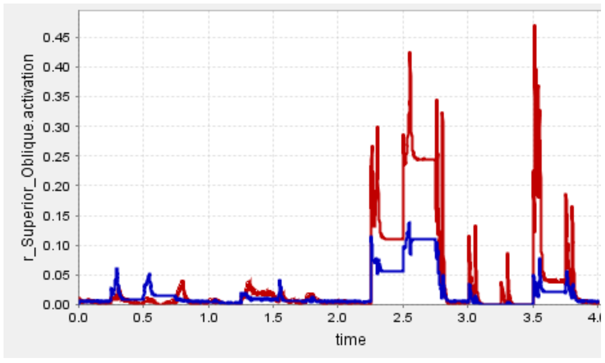
(B) Medial Rectus activation level



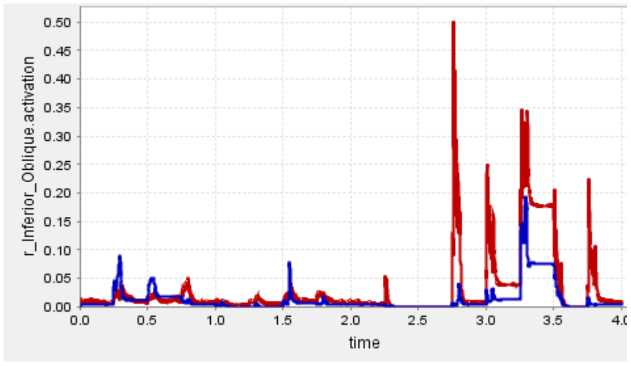
(C) Superior Rectus activation level



(D) Inferior Rectus activation level



(E) Superior Oblique activation level



(F) Inferior Oblique activation level

Figure 14. (A) Lateral Rectus, (B)Medial Rectus, (C) Superior Rectus (D) Inferior Rectus (E) Superior Oblique (F) Inferior Oblique activation levels computed with the application of PD controller (shown in red) and with the Static Optimization tool (shown in blue).

3. Assessment of the Excitation and Activation patterns

The neural drive of a saccadic eye movement can be characterized by a pulse component to overcome the viscoelasticity of the orbital plant, a slide component where the control signal increases sharply, and a step component to stabilize the eye in the new position. However, since we haven't considered any fiber damping and viscosity of the suspension tissues, the pulse component can be attributed to the large velocity error during the onset of the saccades.

Figure.15 shows the excitation level of the Lateral Rectus when abducted to 20^o, and the comparison between the case where we take into account the activation dynamics and where the activation dynamics are disabled. In the latter case, the output of the controller is the direct activation of the EOMs. It is obvious that the latency of the activation dynamics causes the prolonged duration and large oscillations of the pulse component. Although, this does not affect significantly the simulated movement. During the saccade, the activation shows a gradual increase in level reaching a value around 0.7. In the disabled-activation-dynamics case the activation level increases almost linearly. At the end of the saccade, the activation drops abruptly to a lower level around 0.2, which is maintained to stabilize the eye in the target fixation point. These values and the overall behavior of the activation pattern match the documented ones found in the bibliography, thus allowing us to make confident assumptions about the ability of the model to predict the excitation and activation patterns.

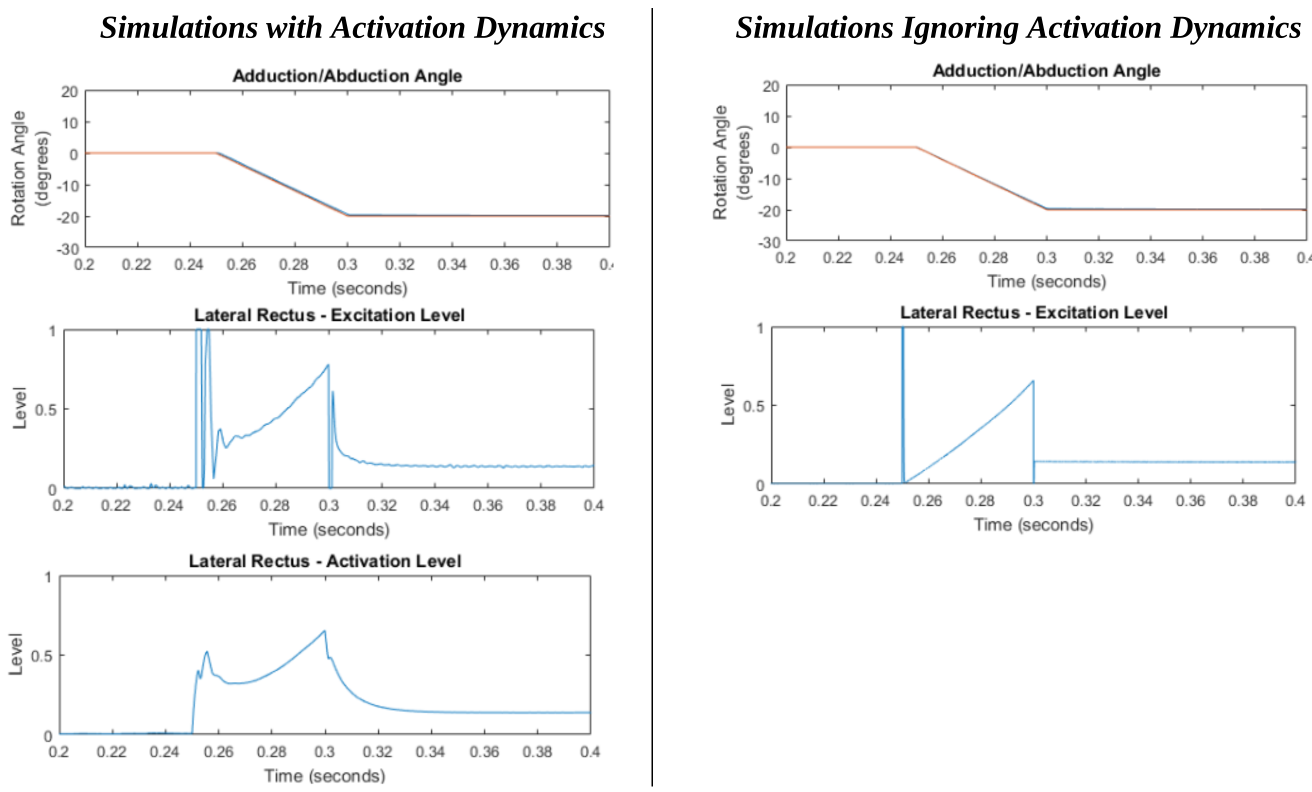


Figure 15. Comparison of Abduction to 20º with incorporating activation dynamics (left column) vs when activation dynamics are ignored (right column). In the later, the output of the PD controller is the activation level of the Lateral Rectus.

The control signals computed with the fixation controller were further used in the forward dynamics tool in OpenSim to reassure their validity. By applying the computed excitation signal in a forward open-loop manner, we manage to attain the same saccadic movements, with the exception of some minor wobbles of the eye globe produced around the fixation points due to the suspension tissues, which were modeled as non-damping springs, that try to restore the globe back to the primary position.

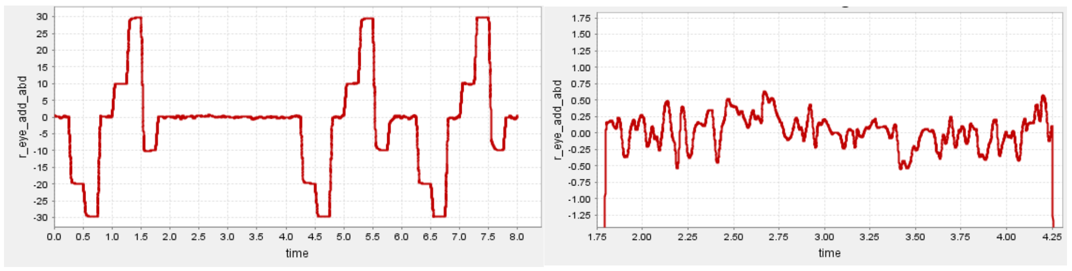


Figure 16. Results of Forward Simulation in OpenSim by applying the control signals computed with the fixation controller. (A) Adduction and Abduction rotation angles, (B) Minor fluctuations around the fixation point.

4. Validation and Verification

To meet good fidelity criteria, a model requires to be verified and validated. In our study, verification in oculomotor models was performed by comparing the force-length characteristic curves of the modelled EOMs to published data. These data include only the lateral rectus characteristic curves. However, we can make a safe assumption that the other muscles have similar properties. More characteristic curves showing the force-length relationship of the other EOMs, as well as the changes in muscle length with rotation in all directions, are presented in the end of this report. Further verification can be done by comparing the model's joint forces produced during horizontal movement with clinically collected data, but these data are not available.

Validation was performed by simulating and analyzing synthesized eye movements while ensuring that Listing Law was obeyed by maintaining zero torsion in secondary gaze positions, and by testing if the assessed innervations are sufficient in fixating the eye at a desired position.

Conclusions and Limitations of the model

We manage to construct a complete ocular model that represents the ocular motility of a normal human eye. The model can be used to drive simulations of different eye movement systems and be derive some inference about the excitation and activation patterns. The verification and validation of the model showed that it matches the data found in literature and it is able to produce synthetized movements. We even showed that the simulation results produced by static optimization and forward dynamics in OpenSim are less satisfactory than the proposed solution where we used a PD controller to minimize the tracking error between the desired and estimated trajectory of saccadic movements. The desired kinematics were tracked within a maximum RMSE of 0.2429^o and 0.2792^o in horizontal and vertical saccades respectively. With the application of the controller, a very rapid instantaneous acceleration and a constant velocity to sustain clear vision is attained. The produced activation levels were in accordance with the descriptions found in the bibliography and the highest activation levels were shown only during the time intervals when the main agonist muscle was activated.

While we make an effort to model the biomechanics of the oculomotor plant at a high level of realism and we attain a satisfactory performance in comparison with other models, the constructed model has a few limitations. First, the force-length relationships were based on actual data only for the lateral rectus muscle and the force-length curves were created by trying to match the shape of the corresponding documented characteristic curves. Second, the force-velocity parameters were set at the default settings of the Millard model in OpenSim after considering that the maximum contraction velocity plays the main role on the active force exerted by the muscles. Third, we used the passive pulley model that is applicable only in secondary gaze positions, and we ignored the separation of EOMs into global and orbital layer. Lastly, for the passive forces of the connective tissues in the orbit, we considered only the linear model and ignored the cubic term. For gaze locations far from the primary gaze position this is likely to become inaccurate. These limitations need to be resolved for more accurate and realistic results in future studies.

References

[1] Q. Wei, S. Sueda, and D. K. Pai, “Physically-based modeling and simulation of extraocular muscles,” *Prog. Biophys. Mol. Biol.*, vol. 103, no. 2–3, pp. 273–283, Dec. 2010.

[2] J. Iskander, M. Hossny, S. Nahavandi, and L. Del Porto, “An ocular biomechanic model for dynamic simulation of different eye movements.,” *J. Biomech.*, vol. 71, pp. 208–216, Feb. 2018.

[3] Q. Wei, S. Sueda, and D. K. Pai, “Biomechanical Simulation of Human Eye Movement,” 2010, pp. 108–118.

[4] A. Priamikov, M. Fronius, B. Shi, and J. Triesch, “OpenEyeSim: A biomechanical model for simulation of closed-loop visual perception,” *J. Vis.*, vol. 16, no. 15, p. 25, Dec. 2016.

[5] J. L. Hicks, T. K. Uchida, A. Seth, A. Rajagopal, and S. L. Delp, “Is My Model Good Enough? Best Practices for Verification and Validation of Musculoskeletal Models and Simulations of Movement,” *J. Biomech. Eng.*, vol. 137, no. 2, p. 20905, Feb. 2015.

[6] R. A. Association for Research in Vision and Ophthalmology., J. M. Miller, and J. L. Demer, “Three-dimensional Location of Human Rectus Pulleys by Path Inflections in Secondary Gaze Positions,” *Invest. Ophthalmol. Vis. Sci.*, vol. 41, no. 12, pp. 3787–3797, Nov. 1977.

[7] P. Bach-y-Rita, C. C. Collins, Smith-Kettlewell Institute of Visual Sciences., and University of the Pacific. Department of Visual Sciences., *The control of eye movements.* Academic Press, 1971.

[8] J. M. Miller, “Understanding and misunderstanding extraocular muscle pulleys,” *J. Vis.*, vol. 7, no. 11, p. 10, Aug. 2007.

[9] R. Association for Research in Vision and Ophthalmology., R. Clark, and J. Demer, “Active Pulleys: Magnetic Resonance Imaging (MRI) of Rectus Muscle Paths in Tertiary Gazes,” *Invest. Ophthalmol. Vis. Sci.*, vol. 43, no. 13, pp. 1460–1460, Dec. 1977.

[10] D. A. Robinson, D. M. O’Meara, A. B. Scott, and C. C. Collins, “Mechanical components of human eye movements.,” *J. Appl. Physiol.*, vol. 26, no. 5, pp. 548–53, May 1969.

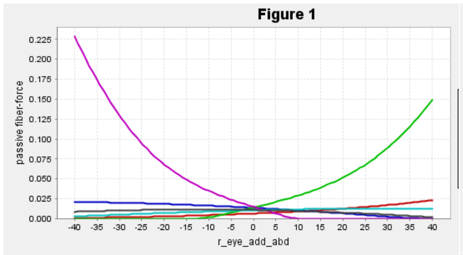
[11] Wong, A.M.F., 2004. Listing’s law: clinical significance and implications for neural control. Surv. Ophthalmol. 49 (6), 563–575.

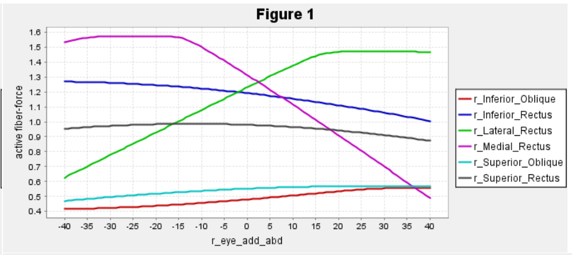
[12] Angelaki, D. E., & Hess, B. J. (2004). Control of eye orientation: Where does the brain’s role end and the muscle’s begin? European Journal of Neuroscience, 19, 1–10.

Force-Length Characteristic Curves

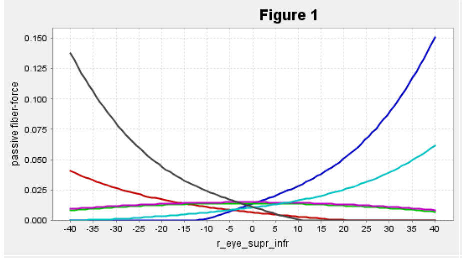
Passive force / Active force

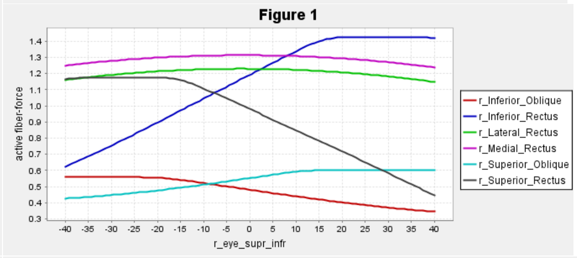
Adduction/
Abduction



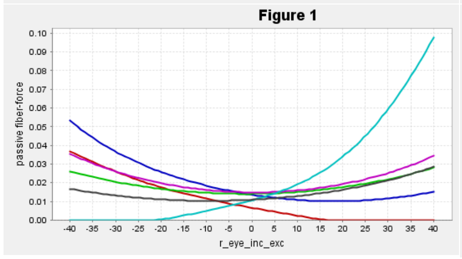


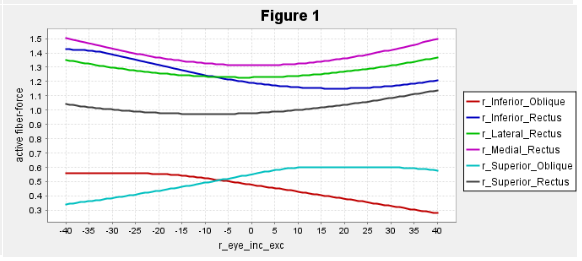
Supraduction/
Infraduction





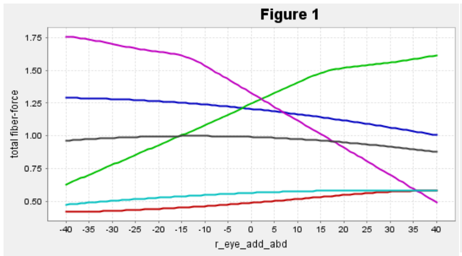
Excyclotorsion/
Incyclotorsion

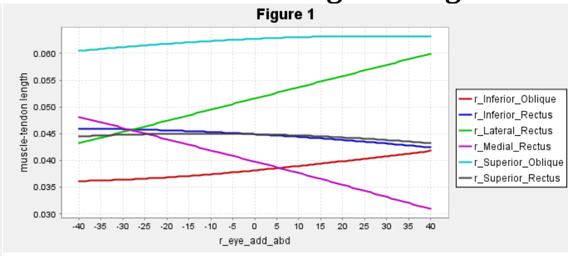




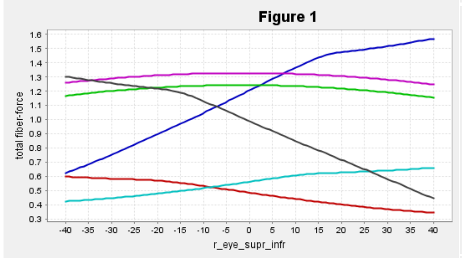
Total Force / Fiber + Tendon Length change

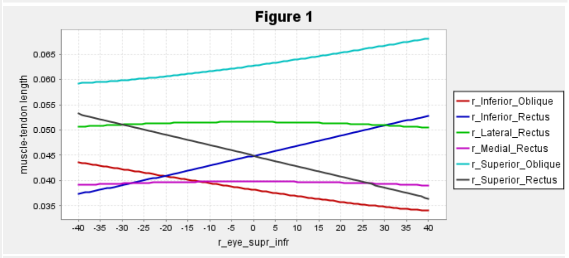
Adduction/
Abduction



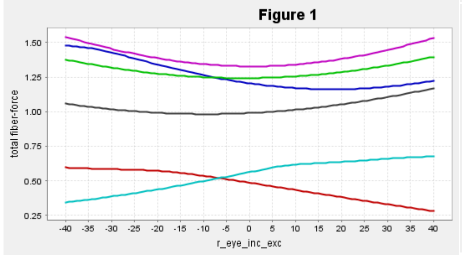


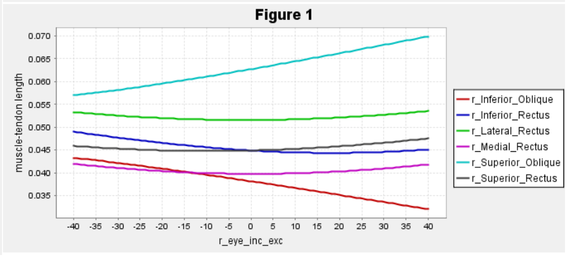
Supraduction/
Infraduction





Excyclotorsion/
Incyclotorsion





Fixation Controller Implementation

```
// Include OpenSim and functions
#include <OpenSim/OpenSim.h>
#include <iostream>
#include <fstream>
using namespace OpenSim;
using namespace SimTK;

//*/
// Ramp Functions
double desiredModelZPosition(double t) {
        double pos = 0;
        if (t < 0.25) pos = 0;
        else if (t < 0.3) pos = (-400 * t + 100)*Pi / 180; else if (t < 0.5) pos = -20 * Pi / 180;
        else if (t < 0.55) pos = (-200 * t +80)*Pi / 180; else if (t < 0.75) pos = -30 * Pi / 180;
        else if (t < 0.8) pos = (600 * t -480)*Pi / 180; else if (t < 1) pos = 0;
        else if (t < 1.05) pos = (200 * t -200)*Pi / 180; else if (t < 1.25) pos = 10 * Pi / 180;
        else if (t < 1.3) pos = (400 * t -490)*Pi / 180; else if (t < 1.5) pos = 30 * Pi / 180;
        else if (t < 1.55) pos = (-800 * t +1230)*Pi / 180; else if (t < 1.75) pos = -10 * Pi / 180;
        else if (t < 1.8 )pos = (200* t -360)*Pi / 180;
        else pos = 0;
        return pos;
}
double desiredModelZVelocity(double t) {
        double vel = 0;
        if (t < 0.25) vel = 0;
        else if (t < 0.3) vel = (-400)*Pi / 180; else if (t < 0.5) vel = 0;
        else if (t < 0.55) vel = (-200 )*Pi / 180; else if (t < 0.75) vel = 0;
        else if (t < 0.8) vel = (600 )*Pi / 180; else if (t < 1) vel = 0;
        else if (t < 1.05) vel = (200 )*Pi / 180; else if (t < 1.25) vel = 0;
        else if (t < 1.3) vel = (400 )*Pi / 180; else if (t < 1.5) vel = 0;
        else if (t < 1.55) vel = (-800 )*Pi / 180; else if (t < 1.75) vel = 0;
        else if (t < 1.8 )vel = (200 )*Pi / 180;
        else vel = 0;
        return vel;
}

//*/
class EyeModelController : public Controller {
        OpenSim_DECLARE_CONCRETE_OBJECT(EyeModelController, Controller);

        // This section contains methods that can be called in this controller class.
public:

        EyeModelController(double aKp, double aKv, double aKpV, double aKvV, double aKpT, double aKvT)
:
                Controller(), kp(aKp), kv(aKv), kpV(aKpV),kvV(aKvV), kpT(aKpT), kvT(aKvT) {}

        void computeControls(const SimTK::State& s, SimTK::Vector &controls) const
        {
                // Get the current time in the simulation.
                double t = s.getTime();
                std::cout << "Time = " << t << std::endl; //Simple test output to check progress

                // Get pointers to each of the muscles in the model.
                Muscle* lat_rect = dynamic_cast<Muscle*>(&getActuatorSet().get(0));
                Muscle* med_rect = dynamic_cast<Muscle*>(&getActuatorSet().get(1));
                Muscle* sup_rect = dynamic_cast<Muscle*>(&getActuatorSet().get(2));
                Muscle* inf_rect = dynamic_cast<Muscle*>(&getActuatorSet().get(3));
                Muscle* sup_oblq = dynamic_cast<Muscle*>(&getActuatorSet().get(4));
                Muscle* inf_oblq = dynamic_cast<Muscle*>(&getActuatorSet().get(5));

                // Compute the desired position and velocity.
                double ydes = desiredModelZPosition(t)+ desiredModelZPosition(t-4);
                double ydesv = desiredModelZVelocity(t)+ desiredModelZVelocity(t - 4);
                double zdes = desiredModelZPosition(t-2) + desiredModelZPosition(t - 4);
                double zdesv = desiredModelZVelocity(t-2)+ desiredModelZVelocity(t - 4);
        double xdes = 0; double xdesv = 0;

                // Get the z rotation coordinate in the model.
                const Coordinate& yCoord = _model->getCoordinateSet().get("r_eye_add_abd");
                const Coordinate& zCoord = _model->getCoordinateSet().get("r_eye_supr_infr");
                const Coordinate& xCoord = _model->getCoordinateSet().get("r_eye_inc_exc");

                // Get the current position and velocity
                double y = yCoord.getValue(s);double yv = yCoord.getSpeedValue(s);
                double z = zCoord.getValue(s);double zv = zCoord.getSpeedValue(s);
                double x = xCoord.getValue(s);double xv = xCoord.getSpeedValue(s);

                // Compute the postion and velocity error and feed it to a PD controller
                double pErrTermHor = kp * (ydes - y);
                double vErrTermHor = kv * (ydesv - yv);

                double pErrTermVert = kpV * (zdes - z);
                double vErrTermVert = kvV * (zdesv - zv);

                double pErrTermTors = kpT * (xdes - x);
                double vErrTermTors = kvT * (xdesv - xv);

                // The sum of errors are used as Excitation levels in the model.
                double sumErrY= pErrTermHor + vErrTermHor;
                double sumErrZ= pErrTermVert + vErrTermVert;
                double sumErrX= pErrTermTors + vErrTermTors;

                // If desired force is in direction of one muscle's pull
                // direction, then set that muscle's control based on the position and velocity error.
                // Otherwise, set the muscle's control to zero.
                double leftControl = 0.0, rightControl = 0.0;
                if (sumErrY < 0) {
                        leftControl = abs(sumErrY);
                        rightControl = 0.0;
                }
                else if (sumErrY > 0) {
                        leftControl = 0.0;
                        rightControl = abs(sumErrY);
                }
                double UpControl = 0.0, DownControl = 0.0;
                if (sumErrZ > 0) {
                        UpControl = abs(sumErrZ);
                        DownControl = 0.0;
                }
                else if (sumErrZ < 0) {
                        UpControl = 0.0;
                        DownControl = abs(sumErrZ);
                }
                double UpTorControl = 0.0, DownTorControl = 0.0;
                if (sumErrX < 0) {
                        UpTorControl = abs(sumErrX);
                        DownTorControl = 0.0;
                }
                else if (sumErrX > 0) {
                        UpTorControl = 0.0;
                        DownTorControl = abs(sumErrX);
                }
                // Saturate of the input to the model.
                // Don't allow any control value to be greater than one.
                if (leftControl > 1.0) leftControl = 1.0;
                if (rightControl > 1.0) rightControl = 1.0;
                if (DownControl > 1.0) DownControl = 1.0;
                if (UpControl > 1.0) UpControl = 1.0;
                if (UpTorControl > 1.0) UpTorControl = 1.0;
                if (DownTorControl > 1.0) DownTorControl = 1.0;

                // Set the activation inputs to the model.
                // Millard muscle has only one control
```

```cpp
		Vector muscleControl(1, leftControl); // left control -> Lateral Rectus
		lat_rect->addInControls(muscleControl, controls);

		muscleControl[0] = rightControl; // right control -> Medial Rectus
		med_rect->addInControls(muscleControl, controls);

		muscleControl[0] = UpControl; // Up control -> Superior Rectus
		sup_rect->addInControls(muscleControl, controls);

		muscleControl[0] = DownControl; // Down control -> Inferior Rectus
		inf_rect->addInControls(muscleControl, controls);

		muscleControl[0] = UpTorControl; // Up Torsion Control -> Superior Oblique
		sup_oblq->addInControls(muscleControl, controls);

		muscleControl[0] = DownTorControl; // Down Torsion Control -> Inferior Oblique
		inf_oblq->addInControls(muscleControl, controls);
	}

private:

	/** Position  and Velocity gain for this controller */
	double kp; double kpV; double kpT;
	double kv;double kvV; double kvT;
};

int main()
{
	try {
		// Create an OpenSim model from the model file provided.

		Model osimModel("EyeModelMillard.osim");

		// Define the initial and final simulation times.
		double initialTime = 0.0;
		double finalTime = 4.0;

		// Set gain for the controller.
		double kp = 50;
		double kv = 1.5;

		double kpV = 50;
		double kvV = 1.5;

		double kpT = 100;
		double kvT = 0.5;
		// Create the controller.
		EyeModelController *controller = new EyeModelController(kp, kv,kpV,kvV,kpT,kvT);

		// Give the controller the Model's actuators so it knows
		// to control those actuators.
		controller->setActuators(osimModel.updActuators());

		// Add the controller to the Model.
		osimModel.addController(controller);

		// Initialize the system and get the state representing the
		// system.
		SimTK::State& si = osimModel.initSystem();

		// Define states for the free joint.
		CoordinateSet& modelCoordinateSet = osimModel.updCoordinateSet();
		Coordinate& yCoord = modelCoordinateSet.get("r_eye_add_abd");
		Coordinate& xCoord = modelCoordinateSet.get("r_eye_inc_exc");
		Coordinate& zCoord = modelCoordinateSet.get("r_eye_supr_infr");
		// Set initial rotation speed value.
		yCoord.setSpeedValue(si, 0);
		xCoord.setSpeedValue(si, 0);
		zCoord.setSpeedValue(si, 0);
```

```
                // Define the initial muscle states.
                const Set<Muscle>& muscleSet = osimModel.getMuscles();
                Millard2012EquilibriumMuscle* muscle1 =
dynamic_cast<Millard2012EquilibriumMuscle*>(&muscleSet.get(0));
                Millard2012EquilibriumMuscle* muscle2 =
dynamic_cast<Millard2012EquilibriumMuscle*>(&muscleSet.get(1));
                Millard2012EquilibriumMuscle* muscle3 =
dynamic_cast<Millard2012EquilibriumMuscle*>(&muscleSet.get(2));
                Millard2012EquilibriumMuscle* muscle4 =
dynamic_cast<Millard2012EquilibriumMuscle*>(&muscleSet.get(3));
                Millard2012EquilibriumMuscle* muscle5 =
dynamic_cast<Millard2012EquilibriumMuscle*>(&muscleSet.get(4));
                Millard2012EquilibriumMuscle* muscle6 =
dynamic_cast<Millard2012EquilibriumMuscle*>(&muscleSet.get(5));

                if ((muscle1 == NULL) || (muscle2 == NULL) || (muscle3 == NULL) || (muscle4 == NULL) ||
(muscle5 == NULL) || (muscle6 == NULL)) {
                        throw OpenSim::Exception("ControllerExample: muscle1 or muscle2 is not an
ActivationFiberLengthMuscle and example cannot proceed.");
                }

                muscle1->setActivation(si, 0.01); // muscle1 activation
                muscle1->setFiberLength(si, 0.0513); // muscle1 fiber length

                muscle2->setActivation(si, 0.01); // muscle2 activation
                muscle2->setFiberLength(si, 0.0398); // muscle2 fiber length

                muscle3->setActivation(si, 0.01); // muscle3 activation
                muscle3->setFiberLength(si, 0.04495); // muscle3 fiber length

                muscle4->setActivation(si, 0.01); // muscle4 activation
                muscle4->setFiberLength(si, 0.0449); // muscle4 fiber length

                muscle5->setActivation(si, 0.01); // muscle5 activation
                muscle5->setFiberLength(si, 0.06137); // muscle5 fiber length

                muscle6->setActivation(si, 0.01); // muscle6 activation
                muscle6->setFiberLength(si, 0.03774); // muscle6 fiber length


                // Create the integrator and manager for the simulation.
                SimTK::RungeKuttaMersonIntegrator integrator(osimModel.getMultibodySystem());
                integrator.setAccuracy(1.0e-3);
                Manager manager(osimModel, integrator);

                // Integrate from initial time to final time.
                manager.setInitialTime(initialTime);
                manager.setFinalTime(finalTime);
                manager.integrate(si);

                // Save the simulation results.
                osimModel.printControlStorage("Eye_Model_controls.sto");
                manager.getStateStorage().print("Eye_Model_states.sto");
        }
        catch (const std::exception &ex) {
                // In case of an exception, print it out to the screen.
                std::cout << ex.what() << std::endl;

                // Return 1 instead of 0 to indicate that something
                // undesirable happened.
                getchar();
                return 1;
        }
        return 0;
}
```