A Three Dimensional Neural Model For Lamprey Swimming

A Report Submitted in Partial Fulfillment of the Requirements for SYDE556

Josh Bradshaw, 20479758

Faculty of Engineering
Department of Systems Design Engineering

April 14, 2017

Course Instructor: Chris Eliasmith

Contents

List of Figures

1

Background

Lamprey swimming has been studied extensively. The Lamprey is widely thought to be the simplest vertebrate. The isolated lamprey spinal cord will move in an oscillatory pattern, even when it is removed from the fish and bathed in solution [1]. Various experiments performed by neuroscientists have shown each short portion of the spinal cord can be modelled as a biphasic oscillator [2].

A two dimensional model of Lamprey swimming using a top-down design methodology was presented by Eliasmith and Anderson in [1]. This model is designed to operate with no input. The model can be used to simulate the Lamprey swimming at a given frequency. The outputs of the model are the Lamprey's spinal neuron activities, and the muscle tensions that these neuron activities correspond to.

This model of the Lamprey was also studied by P. Dwight Kuo, who adapted the model to simulate the motions of larval zebrafish. Dwight Kuo adapted the dynamics of model so that an external signal could cause the fish to rapidly turn to the left or right. This 'escape sequence' mimics the way larval zebrafish move to escape threats in the wild.

Ekeberg and Grillner presented a three dimensional model of Lamprey Swimming in [3]. They use neural data to show that the Lamprey has 'neural sensors' in its brain stem that let it keep track of its roll and pitch orientation. They take a bottom up design approach, in which they mimic measured spinal cord data and use these signals as their model input. They argue that neuron pools and motor units that control they Lamprey's roll and pitch are separate from the neurons that control the primary fictive swimming motion. They state that roll and pitch are controlled through tonic signals from the brain stem, and that the brain stem constantly works to return the fish to its equilibrium in the absence of a perturbation input.

2

Design Problem

This report focuses on extending the top-down model for Lamprey swimming created by Eliasmith et. al and extending it to be three dimensional, and have three control signals for roll, pitch and yaw. The majority of this model should be implemented using mathematical principles outlined in the Neural Engineering Framework. The model should also be reasonably biologically plausible.

Possible extensions of the model that will not be addressed in this report due to time constraints are:

1. Simulating the dynamics and hydrodynamics of the Lamprey in using computational fluid dynamics to determine its actual position changes in 3D space based on its muscle tensions.

2. Modelling the brain stem regions that measure the Lamprey's roll and pitch orientation.

3. Using data from neuroscience to inform parameter selection for the feedback loops.

3

Two Dimensional Model

The system presented is largely similar to the original model. The target muscle tensions are given by:

$$T(z,t) = \kappa(\sin(\omega t - kz) - \sin(\omega t)) \tag{1}$$

Where κ is a scaling factor, which is set equal to one for simplicity. The neural drive in the spinal cord segment that controls this spinal cord activity is given by:

$$T(z,t) = \sum_i a_i e_i = \kappa \sum_i a_i e^{-(z-i*dx)^2/\sigma^2} \tag{2}$$

Where a_i is the segment activity, and the encoders e_i are sampled from a Gaussian basis function. The segment activity is modelled for each of the segments, and each encoders segments are used to determine the localized effect. Ten segments of the Lamprey spine were simulated.

4

Starting Point: Modelling the Lamprey in 2D

4.1 Mathematical Model

As a starting point, the model created by Eliasmith and Anderson was implemented. In this model, the high level description of the neuron activities is given by:

$$T(z,t,A) = \kappa(A_0 + \sum_{n=i}^{n} A_{2n-1}(t)\sin(2\pi nz)A_{2n}(t)\cos(2\pi nz) \tag{3}$$

This relationship is converted into a dynamics equation that's compatible with the Neural Engineering Framework through a series of derivations. The end result is the following state equation:

$$\dot{A} = M_D A + M_I u(t - t_s) \tag{4}$$

Where M_d is the dynamics matrix that produces the oscillatory behaviour, and M_I is the start-up matrix.

$$M_d = \begin{bmatrix} -\alpha_0 & \omega & -\alpha_0 & 0 \\ -\omega & 0 & 0 & 0 \\ -\alpha_0 & -\omega & -\alpha_0 & 0 \\ 0 & 0 & 0 & -\alpha \end{bmatrix} \tag{5}$$

$$M_i = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 1 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \tag{6}$$

The α terms in the dynamics matrix damp high frequency noise, that would otherwise make the system unstable.

A projection matrix is used to move from the high level activity space to the localized activity space, based on the Gaussian basis functions that were used to calculate the encoders.

$$\Gamma = \int \Phi_n(z)\phi_m(z)dz \tag{7}$$

This projection matrix is used to convert from our high level activity space to the local activity space like so:

$$m_d = \Gamma^{-1} M_d \Gamma \tag{8}$$

$$m_i = \Gamma^{-1} M_i \Gamma \tag{9}$$

The local activity space's recurrent connection equation is:

$$\hat{a} = m_D a + m_I a \tag{10}$$

4.2 Implementation

The model was implemented using nengo. The segment activities were represented using a ten dimensional population of neurons. The dynamics calculations were also carried out entirely using neurons, by setting up a recurrent connection between the two populations of neurons. As in the original model, white noise with variance of 0.1 was inputted into the activities population.

4.3 Results

As expected, the lamprey swims consistently at 1.3Hz. The muscle tensions were computed for the right side of the fish, and they also oscillate as expected.

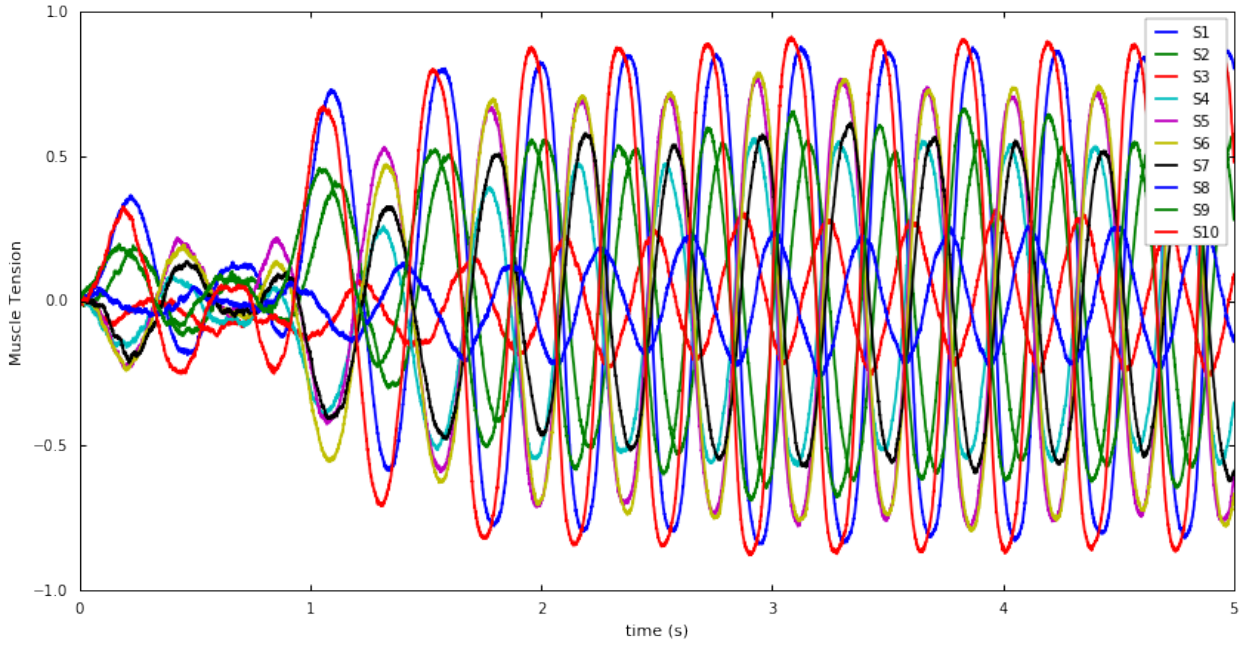

Figure 1: Lamprey segment activities generated using the 2D model.

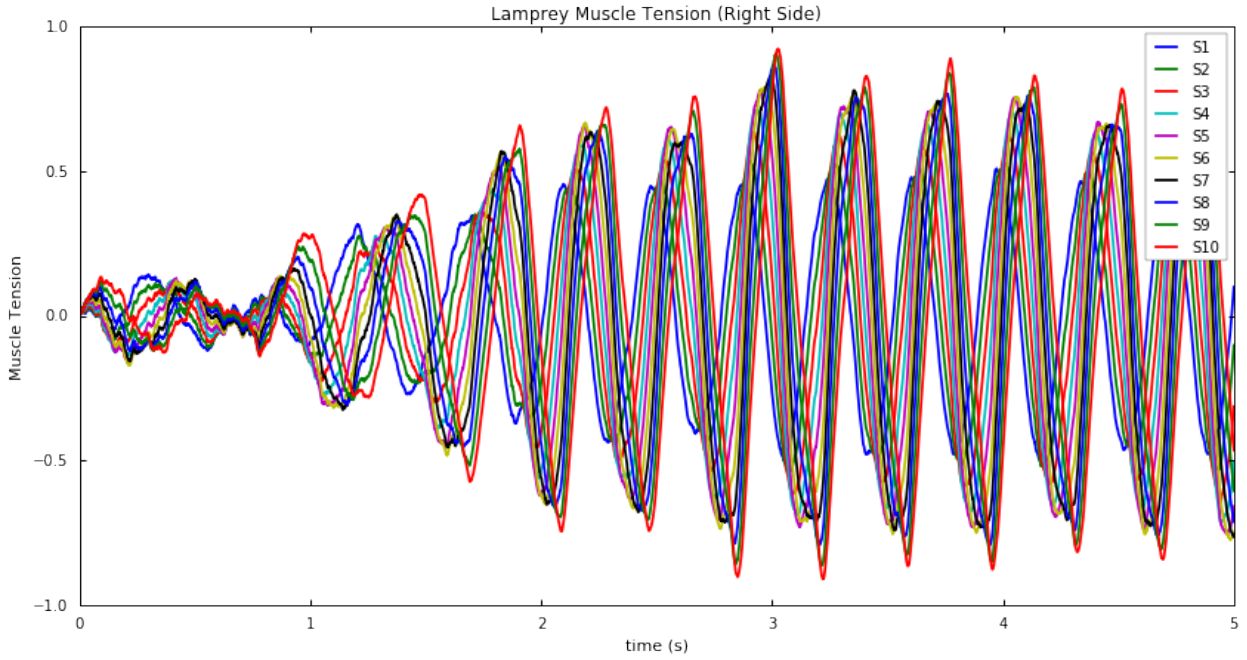

Figure 2: Muscle tension computed based on the segment activities generated using the 2D model.

This model takes more than one cycle to reach steady state. This delay duration depends

on how long the start-up matrix M_I is applied, the frequency of swimming w. It also varies randomly between simulations.

5

Yaw Implementation Attempt I

In this revision, lamprey yawing was implemented by modifying the dynamics and startup matrices based on the zebrafish model in [4]. In the zebrafish model, M_I and M_D are given as:

$$M_d = \begin{bmatrix} -\alpha_0 & \omega(1-E) & -\alpha_0 & 0 \\ -\frac{1}{2}\omega(1-E) & 0 & \frac{1}{2}\omega(1-E) & 0 \\ -\alpha_0 & -\omega(1-E) & -\alpha_0 & 0 \\ 0 & 0 & 0 & -\alpha \end{bmatrix} \tag{11}$$

$$M_i = \begin{bmatrix} \frac{1}{2} & -v(x_1-1) & -\frac{1}{2} \\ 0 & 1 & -vEx_2 \\ -\frac{1}{2} & vE(x_3+1) & \frac{1}{2} \end{bmatrix} \tag{12}$$

The parameters ω, and α have exactly the same meanings as in the original lamprey model. When the parameter E changes to +1 or -1, it the disables the dynamics matrix, and the M_I matrix becomes active once more. The dynamics are designed such that when E=1 or E=-1 the fish will make a sharp turn. This model was implemented in hopes that the escape sequence could be used to implement the yaw signal.

5.1 Implementation

The core algorithm of this model is given below:

```
model = nengo.Network('Neural Lamprey', seed=77)
white_noise = nengo.processes.FilteredNoise(synapse=0.005, dist=Gaussian(mean=0, std=np.sqrt(0.1)), see

with model:
    turning_function = piecewise({0:0, 3:1, 4:0})

    def compute_m_d(t, x):
        E = turning_function(t)

        M_d = np.array(
```

```
        [
            [damp0, freq*(1-E), damp0],
            [-(1/2)*freq*(1-E), 0, (1/2)*freq*(1-E)],
            [damp0, -freq*(1-E), damp0],
        ])
        m_d = np.dot(np.dot(Gamma_inv, M_d), Gamma)
        return tau * np.dot(m_d, x) + x

    def compute_m_i(t, x):
        E = turning_function(t)
        turn_velocity = 0.2
        if E == 0:
            return [0,0,0,0,0,0,0,0,0,0]
        else:
            M_i = np.array([
                [0.5, -turn_velocity*E, -0.5],
                [0, 1, 0],
                [-0.5, turn_velocity*E, 0.5],
            ])
            m_i = np.dot(np.dot(Gamma_inv, M_i), Gamma)
            return tau*np.dot(m_i, x)

    turning_input = nengo.Node(output=turning_function)
    A = nengo.Ensemble(n_neurons, dimensions=10, radius=5, encoders=encoders) # spinal cord neurons
    T = nengo.Node(size_in=10, output=Tension) # compute muscle tension (no neurons involved)
    M_D_compute = nengo.Node(size_in=10, output=compute_m_d)
    M_I_compute = nengo.Node(size_in=10, output=compute_m_i) # compute startup matrix (no neurons involv

    conn = nengo.Connection(A, M_I_compute, synapse=tau)
    conn = nengo.Connection(M_I_compute, A, synapse=tau)

    conn = nengo.Connection(A, M_D_compute, synapse=tau)
    conn = nengo.Connection(M_D_compute, A, synapse=tau)

    conn = nengo.Connection(A, T)
    stim_A = nengo.Probe(A, synapse=0.02)
    stim_T = nengo.Probe(T, synapse=0.02)
    turn_stim = nengo.Probe(turning_input)
```

5.2 Results

The results of the model are shown below. The top signal triggers the escape sequence. The plots below show the resulting segment activities and muscle tensions. As you can see, the Lamprey's muscle tensions become biased in one direction when the input is applied, so that the Lamprey's path will deviate from a straight line.

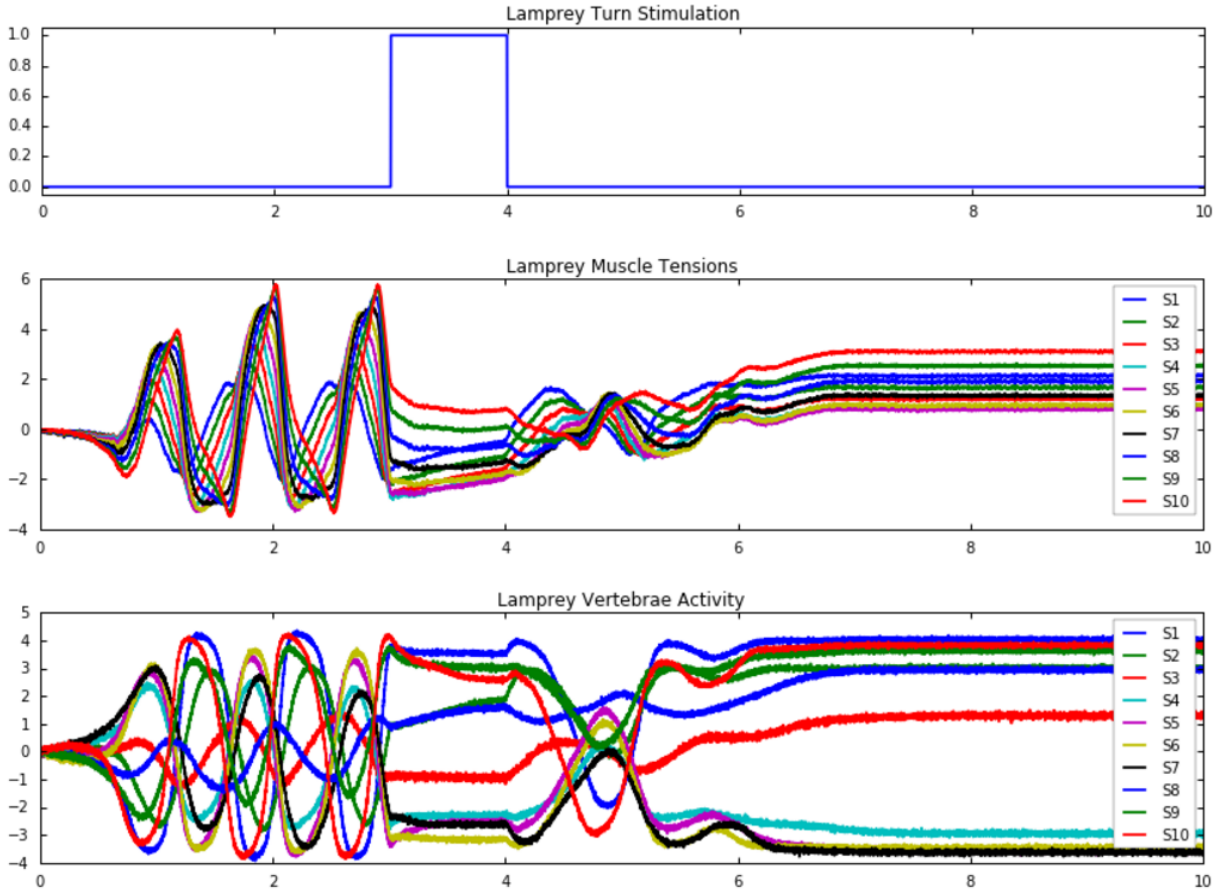

Figure 3: Escape signal and Lamprey muscle tension.

Unfortunately, while the escape signal did cause the fish to turn, it also drove the neurons into saturation. This made it impossible for the fish to recover and continue its oscillatory swimming. This may be a result of a mistake in the model implementation, because the paper provides minimal details about implementation of the start-up and escape signal sequences.

6

Yaw Implementation Attempt II

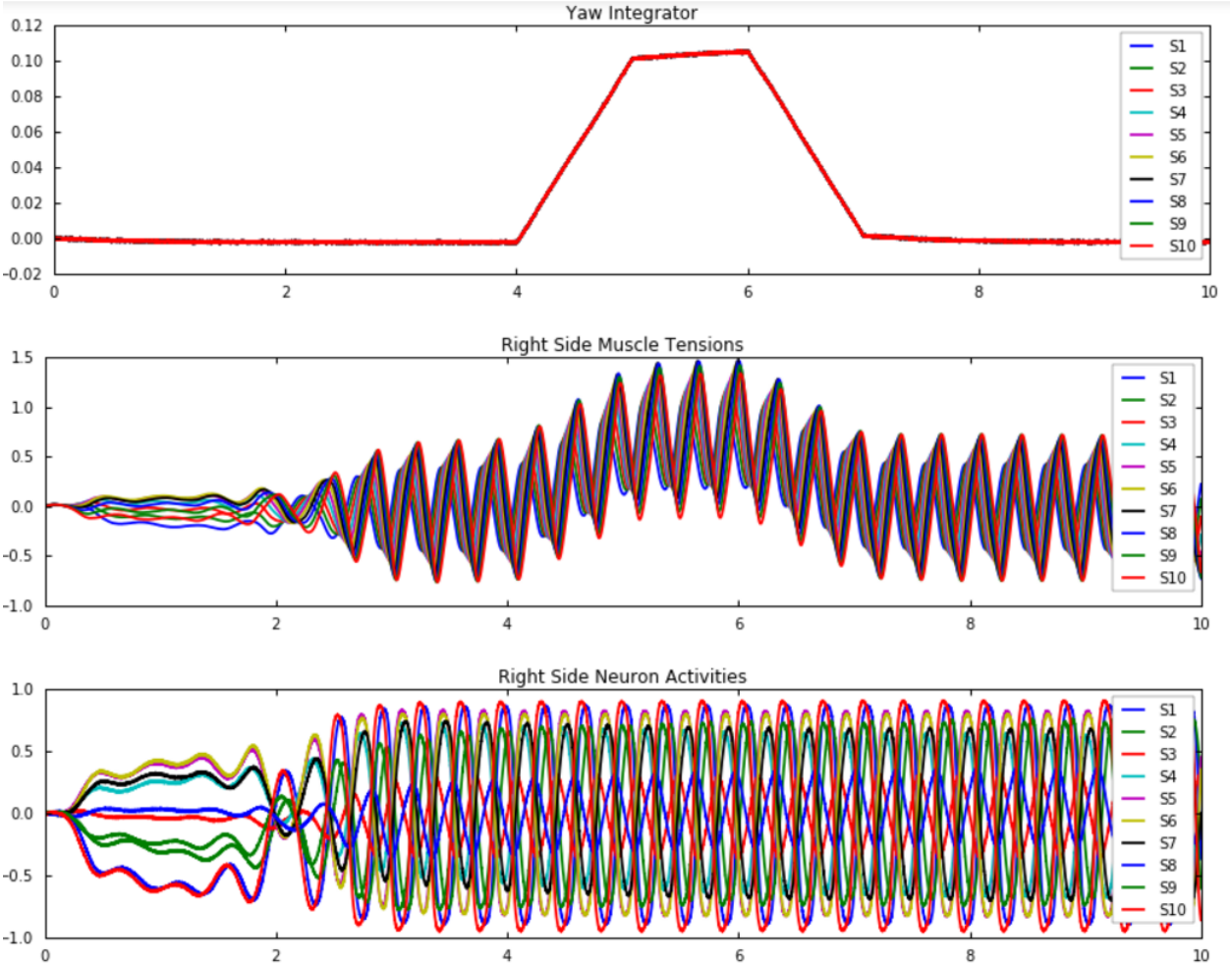

Figure 4: Lamprey turning achieved by summing an integrator with the oscillator.

Unfortunately, the attempt to achieve controlled turning through modification of the dynamics matrix was unsuccessful. In light of this failure, another approach was tested. In this case, a low frequency yaw signal was integrated and summed with the oscillatory neurons directly, using an additional synapse. This approach is somewhat similar to the dynamics approach used by Ekeberg et. al [3]. In their model, all turning was achieved through low frequency 'tonic simulations' that summed with the oscillator signals.

The Ekeberg paper also describes a process by which the Lamprey naturally tends to right itself move towards an equilibrium state of swimming straight and upright. This process was trivial to implement for the yaw signal. The yaw signal is integrated by a an intermediate populations, but the integrator is designed to decay quickly, so that the fish will straighten out when now input is applied.

$$f(x) = \tau s_{yaw} \tag{13}$$

The feedback function for the integrator is:

$$f(x) = 0.9x \tag{14}$$

This model was implemented using the code shown below.

```
model = nengo.Network('Neural Lamprey', seed=77)

with model:
    def compute_m_d(x):
        return tau * np.dot(m_d, x) + x

    def compute_m_i(t, x):
        if t > t_s:
            return [0,0,0,0,0,0,0,0,0,0]
        else:
            return tau*np.dot(m_i, x)

    def Tension(t,x):
        t = []
        for z in range(10):
            y = 0
            for m in range(10):
                y += x[m]*gaussian_dist(z*0.1,m)
            t.append(y)
        return t

    def integrate(x):
        return x

    def map_over_dimensions(t,x):
        return x, x, x, x, x, x, x, x, x, x

    yaw =       nengo.Node(piecewise({0: 0, 4:0.1, 5:0, 6:0}))

    yaw_node =  nengo.Node(size_in=1, size_out=10, output=map_over_dimensions)

    A = nengo.Ensemble(n_neurons, dimensions=10, radius=1, encoders=encoders) # spinal cord neurons
    T = nengo.Ensemble(n_neurons, dimensions=10, radius=1)
    T_CALC = nengo.Node(size_in=10, output=Tension) # compute muscle tension (no neurons involved)
```

```
M_I_compute = nengo.Node(size_in=10, output=compute_m_i) # compute startup matrix (no neurons invol

YAW_INTEGRATOR =   nengo.Ensemble(800, dimensions=1, radius=0.2)
ROLL_INTEGRATOR =  nengo.Ensemble(800, dimensions=1, radius=1)
PITCH_INTEGRATOR = nengo.Ensemble(800, dimensions=1, radius=1)

nengo.Connection(A, A, function=compute_m_d, synapse=tau)
nengo.Connection(A, M_I_compute, synapse=tau)
nengo.Connection(M_I_compute, A, synapse=tau)

nengo.Connection(yaw, YAW_INTEGRATOR, transform=tau)
nengo.Connection(YAW_INTEGRATOR, YAW_INTEGRATOR, transform=0.9, synapse=tau)
nengo.Connection(YAW_INTEGRATOR, yaw_node, synapse=tau)

nengo.Connection(A, T, synapse=tau)
nengo.Connection(yaw_node, T, synapse=tau)
nengo.Connection(T, T_CALC, synapse=tau)

stim_A = nengo.Probe(A, synapse=0.02)
stim_T = nengo.Probe(T_CALC, synapse=0.02)
stim_MI = nengo.Probe(M_I_compute)
stim_roll = nengo.Probe(roll, synapse=0.02)
roll_probe = nengo.Probe(yaw_node)
```

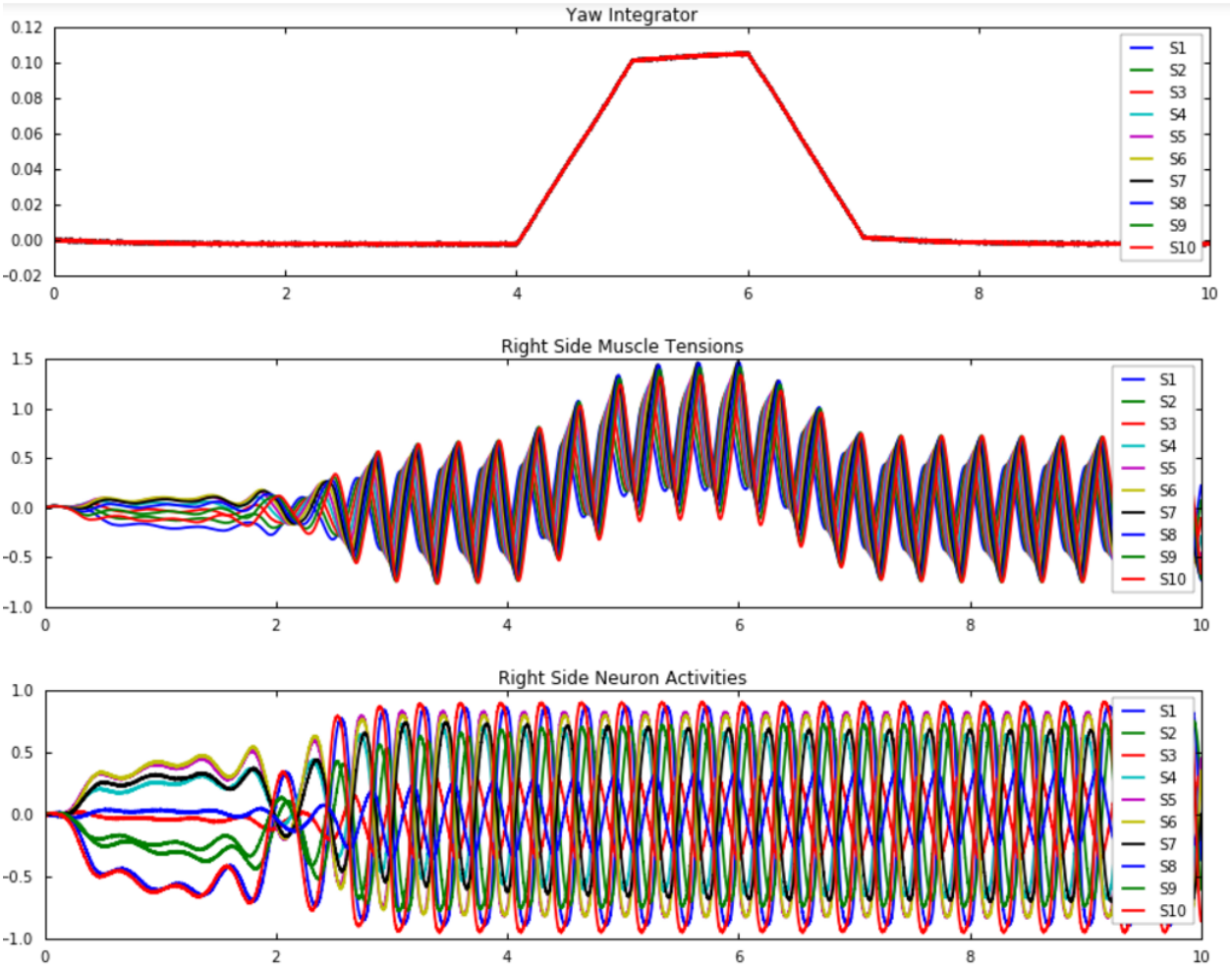

Figure 5: Lamprey turning achieved by summing an integrator with the oscillator.

7

Roll and Pitch Implementation

The Ekeberg paper explains that the roll and pitch signals of the Lamprey are controlled through tonic stimulation. They state that the Lamprey has several independent muscle fibers on the ventral and dorsal sides of its body, and that unbalanced contractions of these muscles is what triggers roll and pitch turns.

For the purposes of this model, the dorsal and ventral muscles were represented using only two pairs of muscles on the corners of the vertebrae. Obviously, the actual Lamprey would have a much more complicated system of muscle fibers on its dorsal and ventral sides. The propagation time for the pitch and roll control signals between the spinal segments was also disregarded. This is a lumped model, that treats the dorsal and ventral muscles of the entire fish as a single unit. This model does not include any sort of position sensing, so the Lamprey's ability to right itself was not modelled.

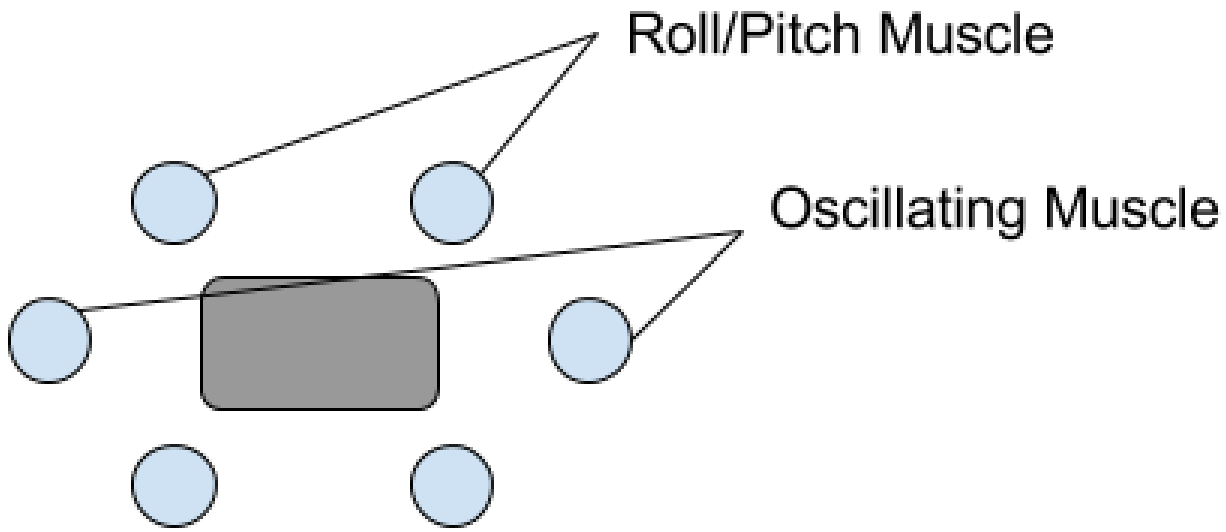

Figure 6: The lamprey's dorsal and ventral muscle fibers, as modelled.

As in the roll and pitch model, the roll and pitch signals were integrated and then

connected to their respective motor units. The pitch integrator was connected such that the lamprey will turn upwards when a positive stimulation is applied. The roll integrator was connected such that the Lamprey will roll to the right when a positive stimulation is applied.

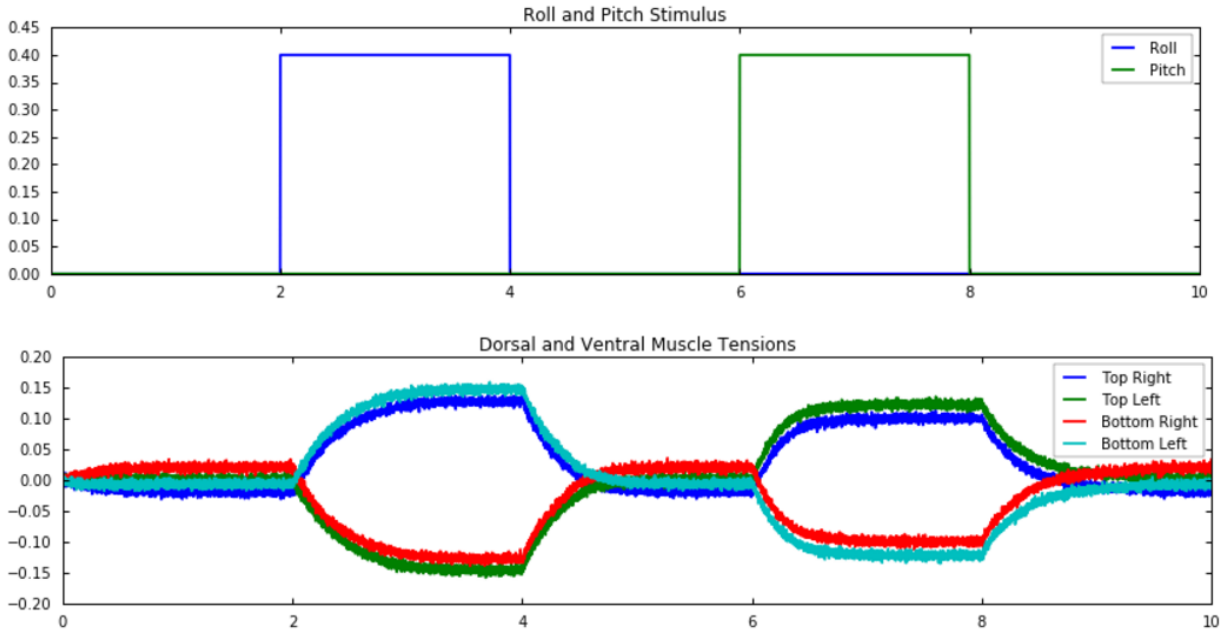

Figure 7: The lamprey's dorsal and ventral muscle fibers response to a roll and pitch input.

8

Testing the Model's Sensitivity to Swimming Frequency

One of the characteristics that I observed while building this model is that the behaviour of the dynamics is was somewhat sensitive to the synaptic time constants of the connections used to compute the dynamics matrix. The results for various swimming frequencies with $\tau = 0.02$ are shown below. For the purposes of this experiment, the stability of the model is evaluated by visually comparing one period of the oscillator to the next.

Simulations of the Lamprey swimming at 0.25 Hz, 5Hz and 10Hz are shown below. For this simulation, all of the synaptic time constants were equal to 20ms.

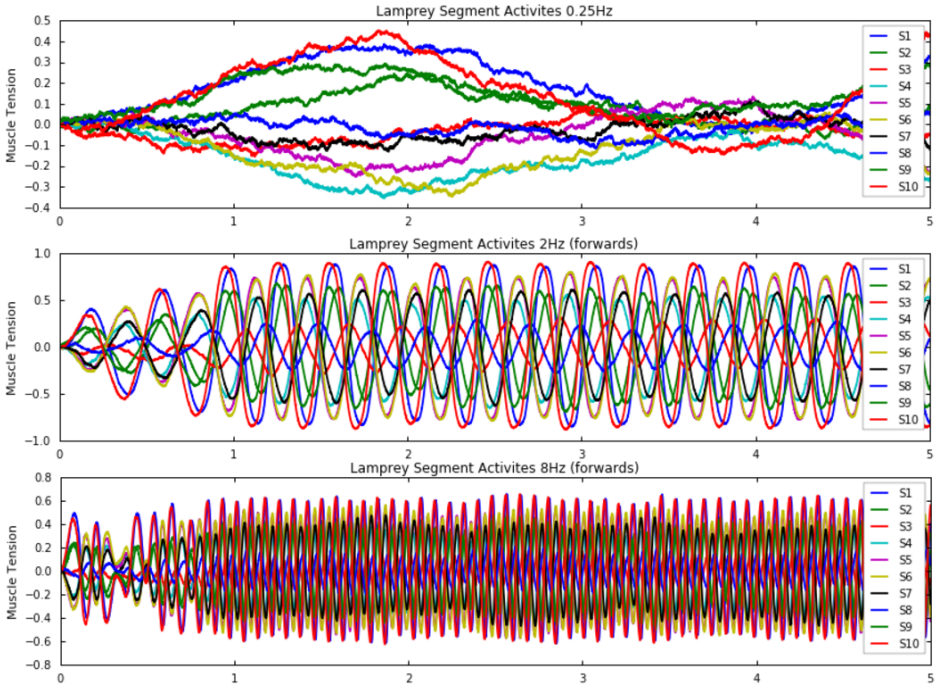

Figure 8: Swimming at various frequencies with $\tau = 0.02$.

The same simulation was carried out with a time constant of 50 ms. The resulting plots

are shown below.

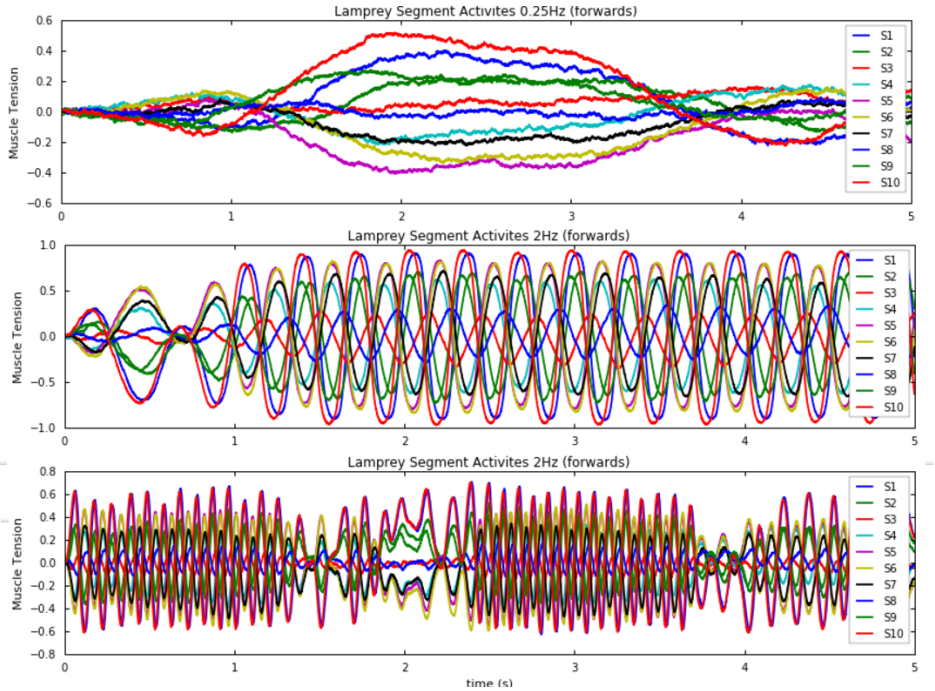

Figure 9: Swimming at various frequencies with $\tau = 0.05$.

Finally, the same simulation was carried out with a time constant of 120ms. The resulting plots are shown below.

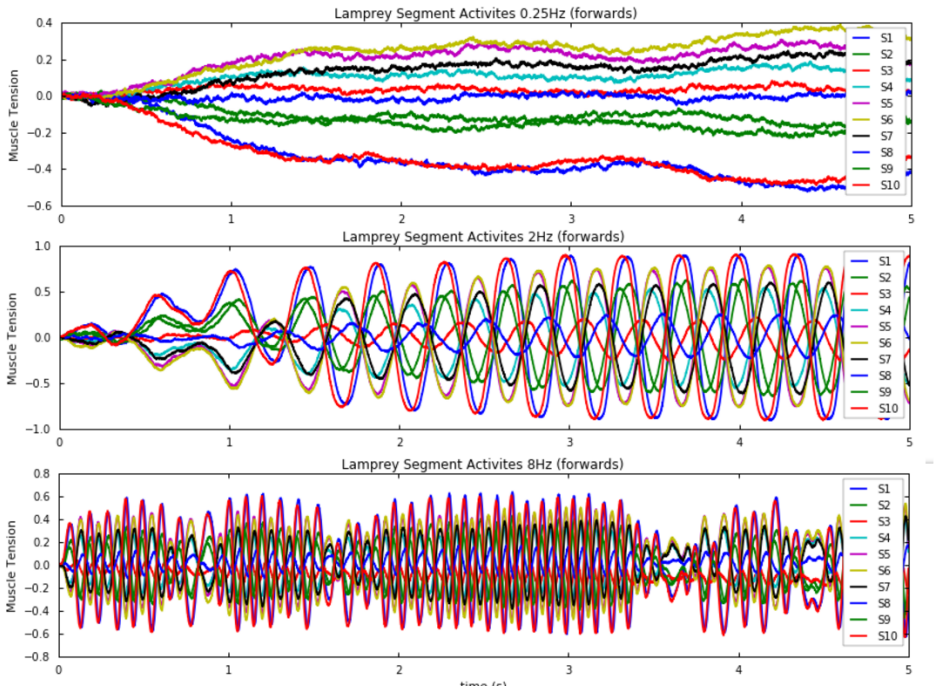

Figure 10: Swimming at various frequencies with $\tau = 0.120$.

As you can see, the oscillator because less stable as the synaptic time constant increases. At 5ms, the oscillator is accurate and consistently achieves the same amplitude of 0.9 on every period. When the time constant is increased to 50ms, the 8Hz simulation becomes unstable and begins to exhibit jitter in the frequency and amplitude of the signal. When the time constant is increased to 120ms, the stability becomes worse still. At 0.25Hz, the system shows no evidence of oscillation, and at 8Hz the system suffers from significant frequency and amplitude drift.

This make sense from a biophysical perspective. In the Lamprey the oscillatory behaviour is implemented using spinal neurons which have a very short synaptic time constant, as opposed to neurons in the brain which have relatively longer time constants. This model helps to explain why the biphasic oscillator of the Lamprey's spine is implemented using spinal neurons as opposed to being implemented in the brain.

9

Conclusion

A three dimensional model for Lamprey swimming was created. The model takes three control inputs for roll, pitch and yaw, and it generates the neuron activities related to the lateral, ventral and dorsal muscles of the fish. This model is based on several simplifying assumptions that reduce its biological plausibility.

The assumption that reduces its biological plausibility the most is the assumption that the signals from the fish's brain stem to the spinal cord are as simple as a roll, pitch and yaw signalling system. The scientific literature shows that the connections between the Lamprey's brain stem and spinal cord are actually much more complex. In the Lamprey, the brain stem is able to modulate the frequency at which the fish swims, and the roll and pitch simulations are much more complex than those used in this model [2].

The roll and pitch portion of this model could be improved significantly by taking more of a bottom up design approach, and informing the modelling based on actual measurements of the ventral and dorsal segments of the Lamprey spinal cord. It could also be improved by implementing the pitch and yaw sensing system of the Lamprey brain stem that the Lamprey uses to right itself [3].

References

[1] Anderson C. H. Eliasmith C. *Neural Engineering*. The MIT Press, Massachusetts Institute of Technology, 1 edition, 2002. (The course text).

[2] Deliagina T. G. Orlovsky G.N. and Wallen P. Vestibular control of swimming in lamprey. *Experimental Brain Research*, (90):479–488, 1991.

[3] Grillner S. Ekeberg, O. Simulations of neuromuscular control in lamprey swimming. *Philosophical Transactions*, (354):895–902, 1999.

[4] P. Adams. Understanding interactions between networks controlling distinct behaviours: Escape and swimming in larval zebrafish. *Neurocomputing*, (2):541–547, 2004.