Student names: Honigmann Simon, Munier Louis & Plett Palomar Kilian Asterio

Instructions: Update this file (or recreate a similar one, e.g. in Word) to prepare your answers to the questions. Feel free to add text, equations and figures as needed. Hand-written notes, e.g. for the development of equations, can also be included e.g. as pictures (from your cell phone or from a scanner). ***This lab is graded.*** *and must be submitted before the* ***Deadline : 11-04-2018 Midnight****.*
Please submit both the source file (.doc/*.tex) and a pdf of your document, as well as all the used and updated Python functions in a single zipped file called lab6_name1_name2_name3.zip where name# are the team member's last names. Please submit only one report per team!*

The file `lab#.py` is provided to run all exercises in Python. The list of exercises and their dependencies are shown in Figure 1. When a file is run, message logs will be printed to indicate information such as what is currently being run and and what is left to be implemented. All warning messages are only present to guide you in the implementation, and can be deleted whenever the corresponding code has been implemented correctly.

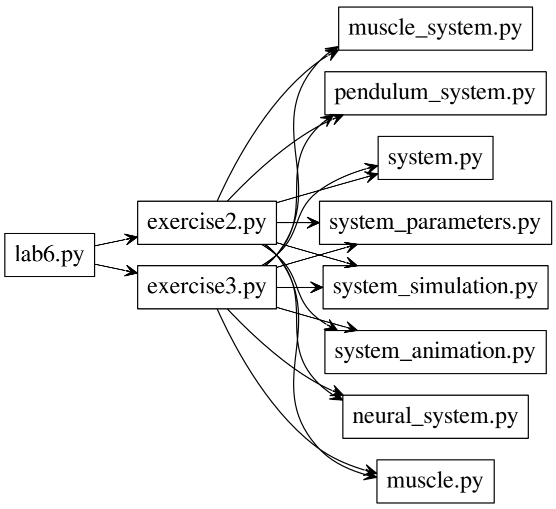


Figure 1: Exercise files dependencies. In this lab, you will be modifying `exercise1.py` and `pendulum_system.py`

Files to complete the exercises

- `lab6.py` : Main file
- `exercise2.py` : Main file to complete exercise 2
- `exercise3.py` : Main file to complete exercise 3
- `system_parameters.py` : Parameter class for Pendulum, Muscles and Neural Network (Create an instance and change properties using the instance. You do not have to modify the file)
- `muscle.py` : Muscle class (You do not have to modify the file)
- `system.py` : System class to combine different models like Pendulum, Muscles, Neural Network (You do not have to modify the file)

- `pendulum_system.py` : Contains the description of pendulum equation and Pendulum class. You can use the file to define perturbations in the pendulum.
- `muscle_system.py` : Class to combine two muscles (You do not have to modify the file)
- `neural_system.py` : Class to describe the neural network (You do not have to modify the file)
- `system_simulation.py` : Class to initialize all the systems, validate and to perform integration (You do not have to modify the file)
- `system_animation.py` : Class to produce animation of the systems after integration (You do not have to modify the file)

NOTE : '*You do not have to modify*' does not mean you should not, it means it is not necessary to complete the exercises. But, you are expected to look into each of these files and understand how everything works. You are free to explore and change any file if you feel so.

Exercise 2 : Pendulum model with Muscles

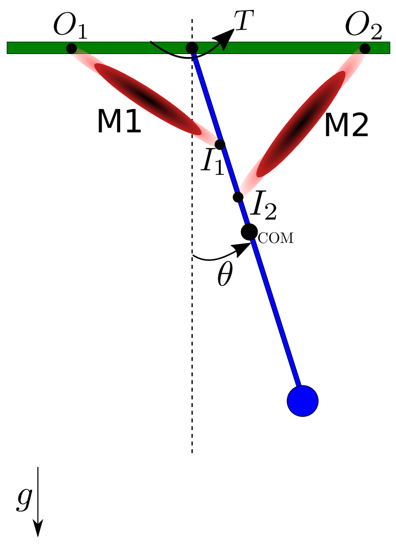


Figure 2: Pendulum with Antagonist Hill Muscles

The system is comprised of a physical pendulum described by equation 1 and a pair of antagonist muscles **M1** and **M2**. Muscle **M1** extends the pendulum (θ increases) and Muscle **M2** flexes the muscle (θ decreases).

Consider the system only for the pendulum range $\theta = [-\pi/2, \pi/2]$

$$I\ddot{\theta} = -0.5 \cdot m \cdot g \cdot L \cdot sin(\theta) \tag{1}$$

Where,

- I - Pendulum inertia about the pendulum pivot joint $[kg \cdot m^2]$
- θ - Pendulum angular position with the vertical $[rad]$
- $\ddot{\theta}$ - Pendulum angular acceleration $[rad \cdot s^{-2}]$
- m - Pendulum mass $[kg]$

- g - System gravity $[m \cdot s^{-2}]$
- L - Length of the pendulum $[m]$

Each muscle is modelled using the Hill-type equations that you are now familiar with. Muscles have two attachment points, one at the origin and the other at the insertion point. The origin points are denoted by $O_{1,2}$ and the insertion points by $I_{1,2}$. The two points of attachment dictate how the length of the muscle changes with respect to the change in position of the pendulum.

The active and passive forces produced by the muscle are transmitted to the pendulum via the tendons. In order to apply this force on to the pendulum, we need to compute the moment based on the attachments of the muscle.

Using the laws of sines and cosines, we can derive the length of muscle and moment arm as below. The reference to the paper can be found here Reference,

$$L_1 = \sqrt[2]{a_1^2 + a_2^2 + 2 \cdot a_1 \cdot a_2 \cdot \sin(\theta)} \tag{2}$$

$$h_1 = \frac{a_1 \cdot a_2 \cdot \cos(\theta)}{L_1} \tag{3}$$

Where,

- L_1 : Length of muscle 1
- a_1 : Distance between muscle 1 origin and pendulum origin ($|O_1C|$)
- a_2 : Distance between muscle 1 insertion and pendulum origin ($|I_1C|$)
- h_1 : Moment arm of the muscle

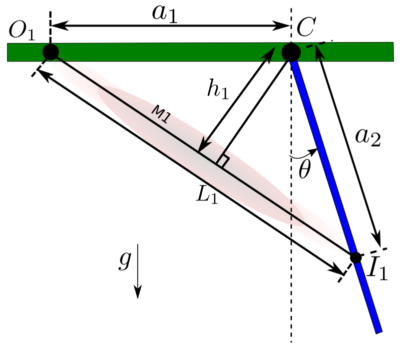


Figure 3: Computation of muscle length and moment arm

Equation 2 can be extended to the Muscle 2 in similar way. Thus, the final torque applied by the muscle on to the pendulum is given by,

$$\tau = F \cdot h \tag{4}$$

Where,

- τ : Torque $[N \cdot m]$
- F : Muscle Tendon Force $[N]$
- h : Muscle Moment Arm $[m]$

In this exercise, the following states of the system are integrated over time,

$$X = \begin{bmatrix} \theta & \dot{\theta} & A_1 & l_{CE1} & A_2 & l_{CE2} \end{bmatrix} \tag{5}$$

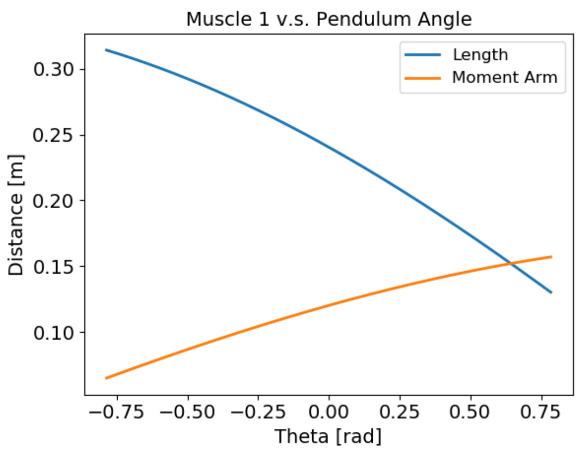
Where,

- θ : Angular position of the pendulum [rad]
- $\dot{\theta}$: Angular velocity of the pendulum [rad/s]
- A_1 : Activation of muscle 1 with a range between [0, 1]. 0 corresponds to no stimulation and 1 corresponds to maximal stimulation.
- l_{CE1} : Length of contractile element of muscle 1
- A_2 : Activation of muscle 2 with a range between [0, 1]. 0 corresponds to no stimulation and 1 corresponds to maximal stimulation.
- l_{CE2} : Length of contractile element of muscle 2

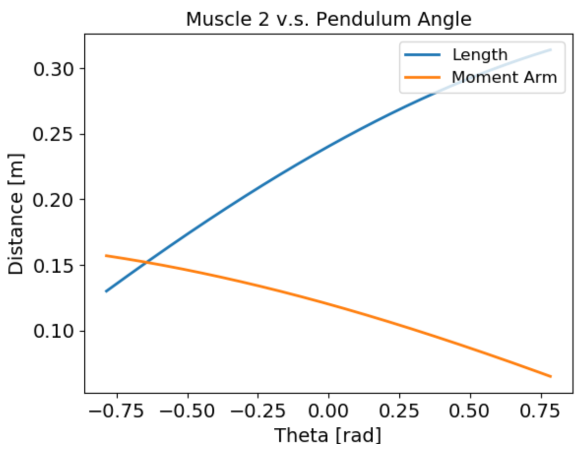
To complete this exercise you will make use of the following files, `exercise2.py`, `system_parameters.py`, `muscle.py`, `system.py`, `pendulum_system.py`, `muscle_system.py`, `system_simulation.py`

2a. For a given set of attachment points, compute and plot the muscle length and moment arm as a function of θ between $[-\pi/4, \pi/4]$ using equations in eqn:2 and discuss how it influences the pendulum resting position and the torques muscles can apply at different joint angles. You are free to implement this code by yourself as it does not have any other dependencies.

The figures below show the relationship between angle and muscle length or moment arm. The simulated pendulum's muscles are approximately 24 cm long, each mounted at 45 degrees to the vertical. The muscles are unstretched when the pendulum angle is 0. All other parameters remain as default values (e.g. $l_{pendulum} = 50cm, m_{pendulum} = 1kg$).



(a) Angle vs Muscle 1 Length and Moment Arm. Muscle 1 has a negative moment arm, but it is plotted as a positive value here to allow for direct comparison with muscle 2.



(b) Angle vs Muscle 2 Length and Moment Arm.

Figure 4: Effect of Pendulum Angle on Muscles

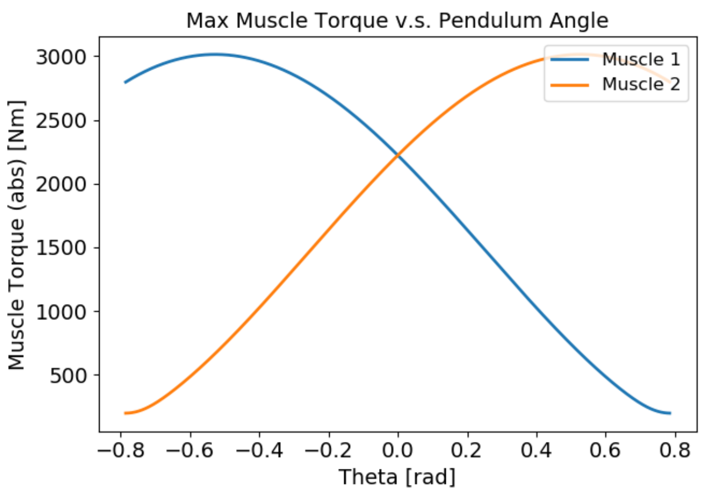


Figure 5: Pendulum Angle vs Maximum Muscle Torque ($v_ce = 0$, Stimulation = 1). Muscle 1 produces negative torques, which were plotted as positive values to allow direct comparison.

From figure 4, it can be noted that the muscles are balanced when the angle is equal to 0 and the pendulum is vertical. Both muscles, having the same properties, have the same length and equal and opposite moment arms, resulting in a net-zero torque on the pendulum.

The torque applied by each muscle is related to the moment arm, and is related to the muscle force, which is in-turn dependent on muscle length. As the muscle length stretches or contracts from the optimal length, it's force capacity decreases. As the muscle gets shorter, this effect is offset by an increasing moment arm. As the muscle lengthens, the moment arm actually decreases for this configuration. As a result, either muscle will be capable of dominating the other when it is allowed to contract first (all other parameters being equal). 5 demonstrates this behaviour, with a peak muscle torque occurring at a pendulum angle of about 35 deg (0.6 rad). When the pendulum is pulled to either side, the muscle which has contracted must counter the torque created by the gravitational pull on the pendulum and the torque created by the other muscle.

2b. Using simple activation wave forms (example : sine or square waves) applied to muscles (use `system_simulation.py::add_muscle_activations` method in `exercise2.py`), try to obtain a limit cycle behavior for the pendulum. Use relevant plots to prove the limit cycle behavior. Explain and show the activation wave forms you used. Use `pendulum_system.py::PendulumSystem::pendulum_system` function to perturb the model.

In this section, a sinusoidal input was applied to each muscle to result in a stable limit cycle behaviour of the pendulum. A 0.5 Hz sine wave was applied to one muscle, and an equivalent sine wave with a 180 degree phase shift was applied to the alternate muscle. As the muscles can only receive positive stimulation between 0 and 1, a unity amplitude was used, and all negative values were set to 0. The resultant waveform are plotted below for initial conditions of $\theta_0 = \pi/4$ and $\omega_0 = 0$.

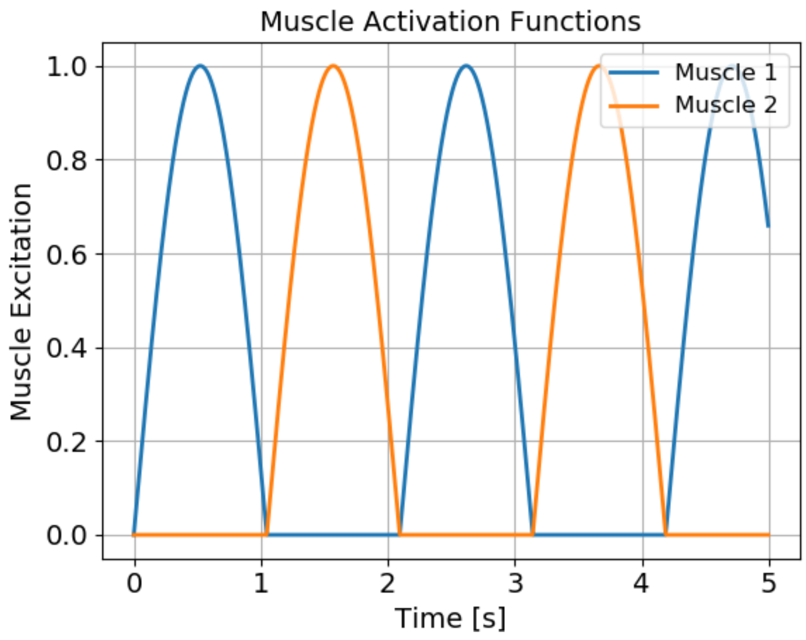


Figure 6: Effect of Pendulum Angle on the Muscles

With these waveform activating each muscle, the system was simulated and a phase plot of the results was created. To prove stability, Poincare analysis was performed. Figure 7 shows the system, starting from initial conditions, then converging onto a limit cycle behaviour. The Poincare cross section is indicated on the plot. Figure 8 shows the intersection angles with the cross section for each crossing. While no clear convergence is shown, the scale of the values is within the margin of uncertainty of the model, given the integration methods and time-step used. Thus, it is fair to say that the system has indeed converged into a limit cycle behaviour.

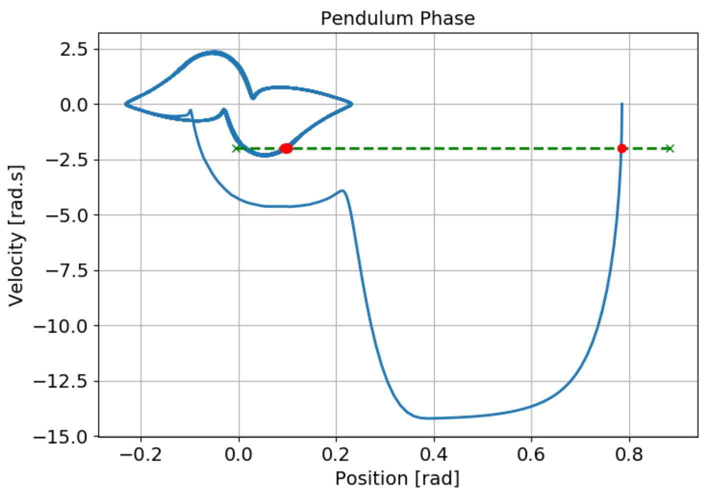


Figure 7: Pendulum Angle-Velocity Phase Plot showing limit cycle behaviour

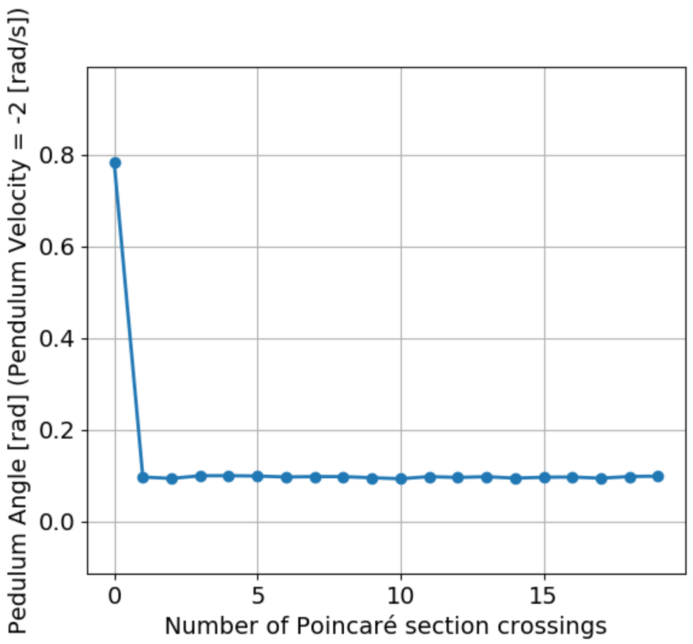
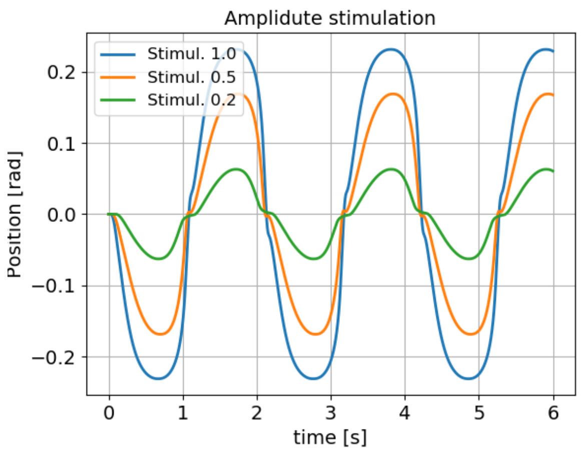


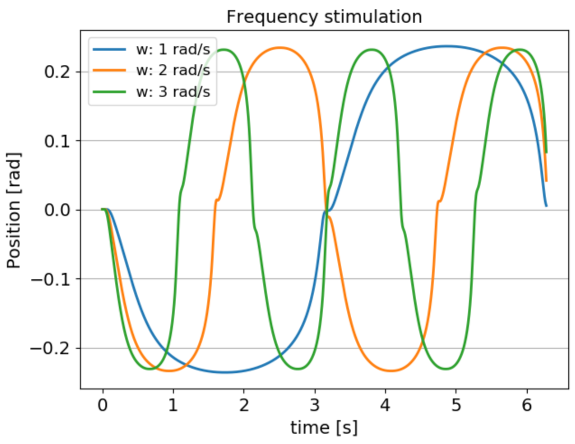
Figure 8: Plot of the crossing location of each Poincare intersection point.

2c. Show the relationship between stimulation frequency and amplitude with the resulting pendulum's behavior.

To show the relationship between pendulum behaviour and the frequency and amplitude of muscle stimuli, the same half-sign wave functions used in Exercise 2b were used. However, in this experiment, the amplitudes and frequencies were varied. Amplitudes of 0.2, 0.5, and 1.0 were tested with a frequency of 3 rad/s. Then frequencies of 1, 2 and 3 rad/s were compared with an amplitude of 1. Figure 9 shows the resultant phase-time relation from different configurations of stimulation.



(a) Pendulum position over time, showing the effect of different stimulation amplitudes.



(b) Pendulum position over time, showing the effect of different stimulation frequencies, where $\omega/2\pi = f$ [Hz], namely 0.159Hz, 0.318Hz and 0.477Hz.

Figure 9: Effect of different amplitude and frequency stimulation on the pendulum.

The stimulation level largely determines the amplitude of the pendulum oscillations. As the muscle force is directly related to muscle stimulation, this makes intuitive sense. As the pendulum crosses the zero position, a concavity change can be noted in all plots. This change indicates the change in stimulation of the muscles at the point of peak muscle velocity. As the pendulum crosses through the 0 angle position, the active forces are zero, but the passive forces (particularly the parallel element) are unbalanced as their magnitude is dependent on the direction of the muscle velocity. Interestingly, in the case with full muscle stimulation, this concavity shift occurs at an angle higher than in the case of the small stimulation amplitude. This indicates a phase lag between the stimulation commands and the pendulum positioning.

Varying the stimulation frequency also yields interesting results. The lower frequency stimulation had the most pronounced concavity change at the zero crossing, but had the smallest phase lag of the 3 tested stimulation amplitudes. The duration of the concavity change is expected to be larger, simply due to the slower rate of change of muscle stimulation causing the slower movement of the pendulum. The phase lag increase with increased frequency is also expected, as the velocities involved are higher and the system's inertia plays a larger role.

Exercise 3 : Neural network driven pendulum model with muscles

In this exercise, the goal is to drive the above system 2 with a symmetric four-neuron oscillator network. The network is based on Brown's half-center model with fatigue mechanism. Here we use the leaky-integrate and fire neurons for modelling the network. Figure 10 shows the network structure and the complete system.

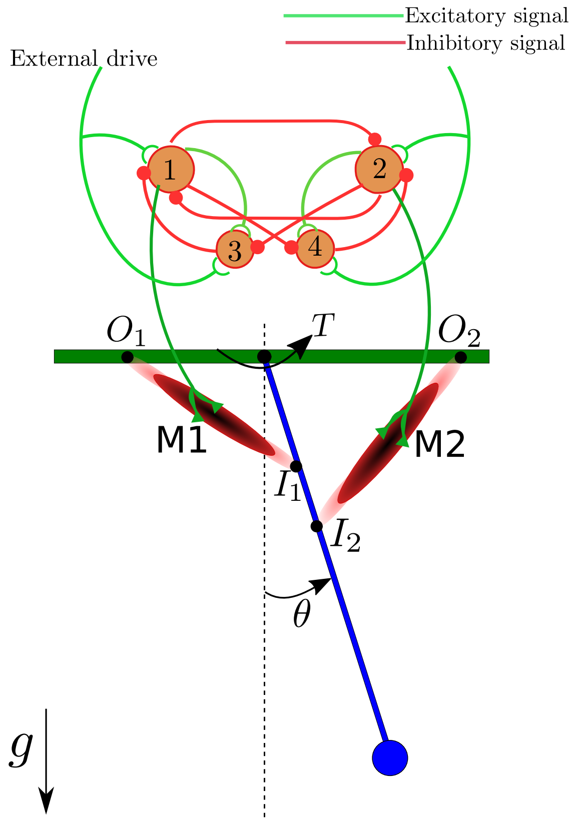


Figure 10: Pendulum with Antagonist Hill Muscles Driven Half Center Neural Network.

Since each leaky-integrate and fire neuron comprises of one first order differential equation, the states to be integrated now increases by four(one state per neuron). The states are,

$$X = \begin{bmatrix} \theta & \dot{\theta} & A_1 & l_{CE1} & A_2 & l_{CE2} & m_1 & m_2 & m_3 & m_4 \end{bmatrix} \tag{6}$$

Where,

- m_1 : Membrane potential of neuron 1
- m_2 : Membrane potential of neuron 2
- m_3 : Membrane potential of neuron 3
- m_4 : Membrane potential of neuron 4

To complete this exercise, additionally you will have to use `neural_system.py` and `exercise3.py`

3a. Find a set of weights for the neural network that produces oscillations to drive the pendulum into a limit cycle behavior. Plot the output of the network and the phase plot of the pendulum

As seen in lecture *"lecture4_Neuron_Models"*, the different weights of the neural network that cause the system to oscillate are as follows. These weights determine how the system of neurons inhibits and excites each other:

$$w = \begin{bmatrix} 0 & -5 & -5 & 0 \\ -5 & 0 & 0 & -5 \\ 5 & -5 & 0 & 0 \\ -5 & 5 & 0 & 0 \end{bmatrix} \tag{7}$$

By doing this, the neuronal network outputs become evident in an stable oscillatory pattern. Here the external excitation accordingly has value of zero. Figure 11 below shows the output of the neurons over time. In particular, the transient between each neuron's initial state and the final limit cycle can be seen between 0 and 0.2 seconds.

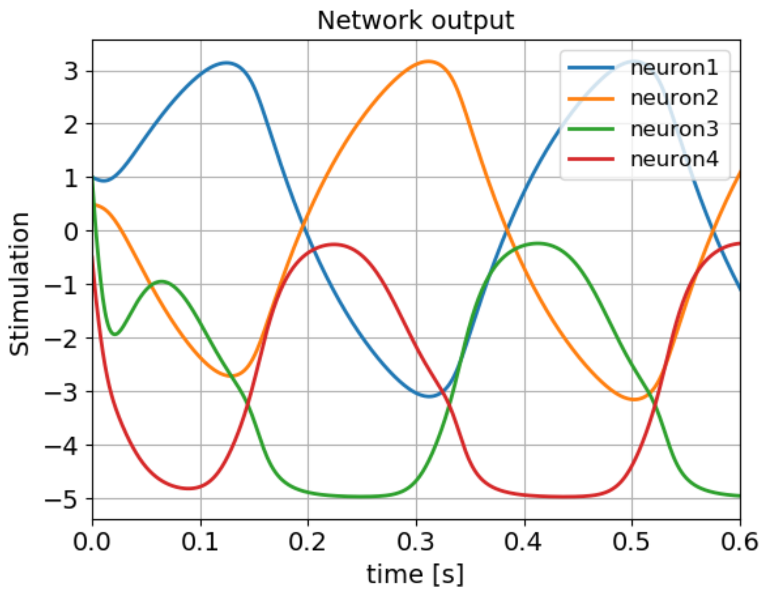


Figure 11: Neuron outputs over time during limit cycle behaviour.

Figure 12 demonstrates the stability of the limit cycle with different initial conditions and disturbances. In each case, the system quickly returns to the limit cycle (purple colour).

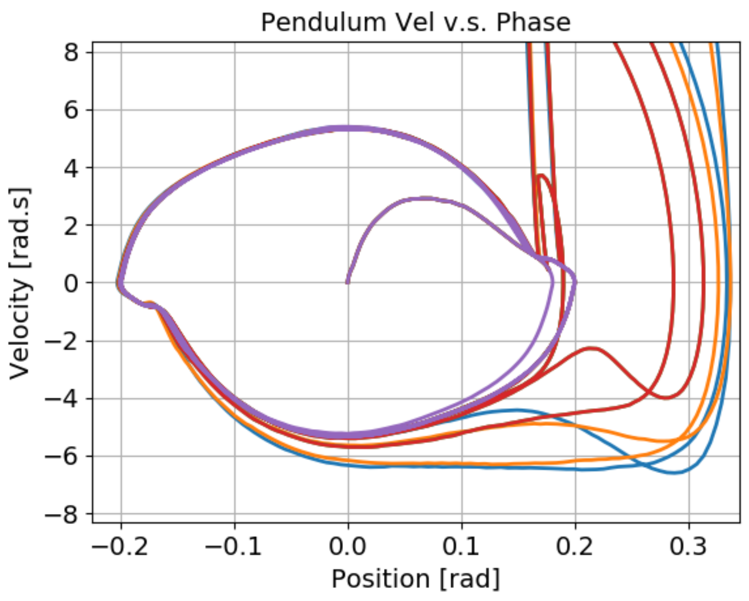
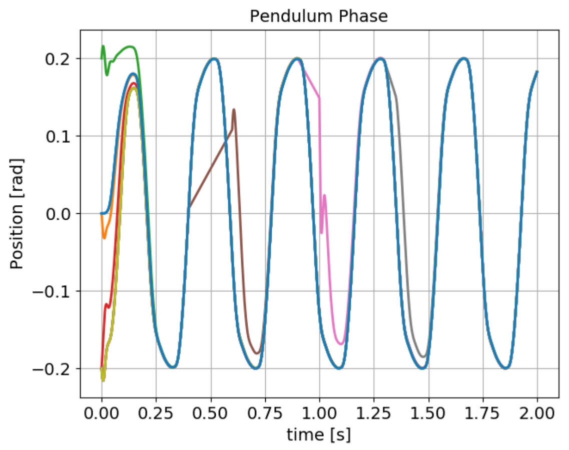
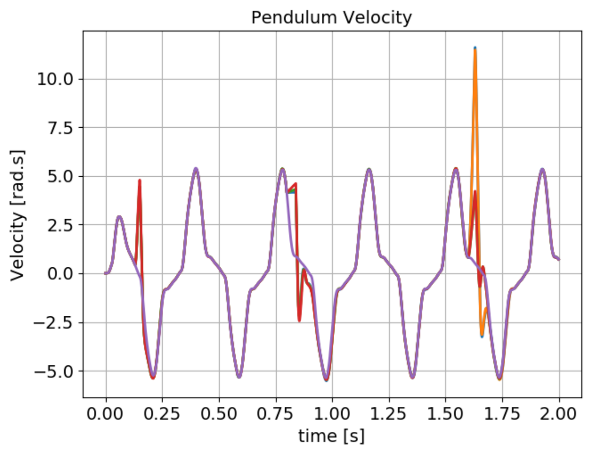


Figure 12: Limit cycle behaviour of a neural network controlled muscle-pendulum system. Different starting positions and disturbances are shown in different colors. All traces return to the same stable limit cycle.

In Figure 13 some disturbances have been applied to the system in order to demonstrate that it will always go back to the limit cycle behavior. To obtain them, some runs were done with different disturbances (change in angular velocity, applying a torque or different initial conditions) and plot on the same graph to see the behaviour without disturbances. In the Phase graph all disturbances converge to the blue curve (line without dirturbances). In the Velocity graph all perturbations converge to the purple curve. So in all cases we achieve a limit cycle.



(a) Pendulum position over time, showing the effect of different initial conditions and disturbance (angular speed and torque), shown in orange, green, brown, and pink. The blue one is the behavior without any disturbances.



(b) Pendulum velocity over time, showing the effect of different disturbances (angular speed and torque), shown in orange, red and green. The violet one is the behavior without any disturbances.

Figure 13: Effect of disturbances on the pendulum.

3b. As seen in the course, apply an external drive to the individual neurons and explain how the system is affected. Show plots for low [0] and high [1] external drives. To add external drive to the network you can use the method `system_simulation.py::add_external_inputs_to_network`

For all Figures in this section there is a change in stimulation (from 0.0 to 1.0) of all neurons in the network at time 2 seconds. Observable is the increase in oscillating frequency of the pendulum, with a decrease in oscillating amplitude.

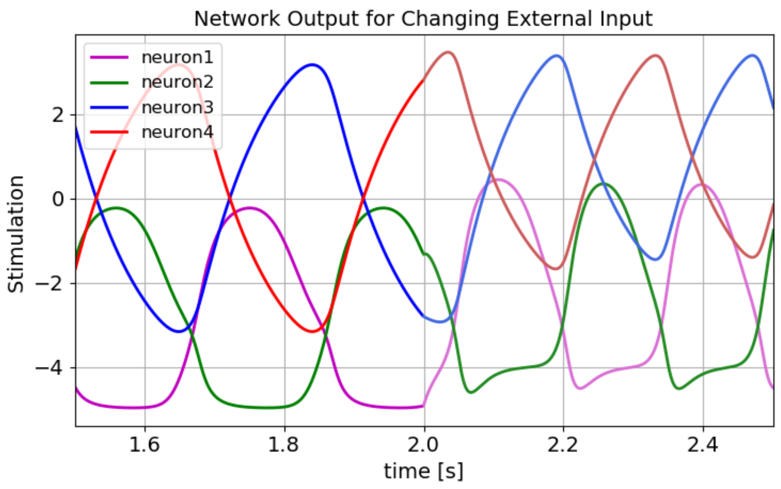
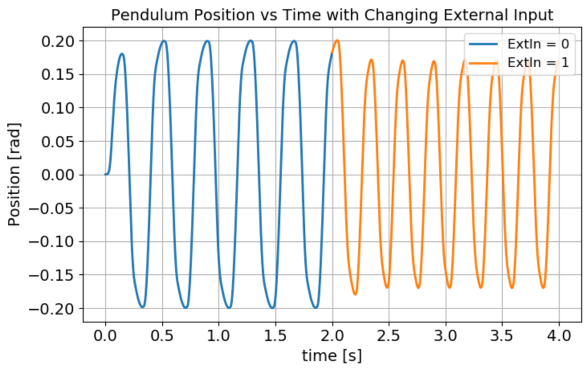
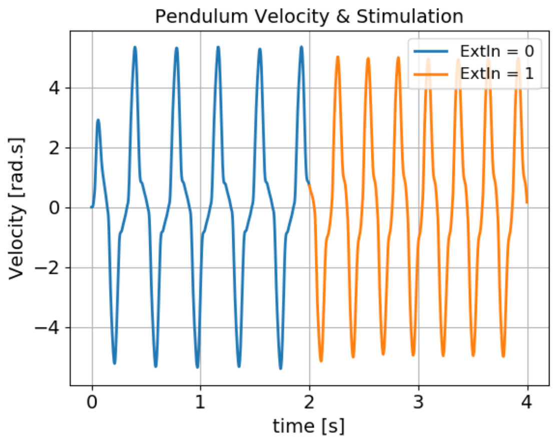


Figure 14: Effect of external inputs of 0.0 and 1.0 to the neurons

In the system we can observe the change in frequency and in amplitude of the system by applying external stimulation (t>2s). The transition between the two limit cycles, as shown in Figure 16 and 15, is smooth and the system resettles quickly.



(a) Pendulum Position over Time with and without External Input



(b) Pendulum Velocity over Time with and without External Input

Figure 15: Effect of Pendulum Angle on the Muscles. at 2 seconds the stimulation changes to 1.0, producing a decrease on phase and angular velocities but an increase in frequency.

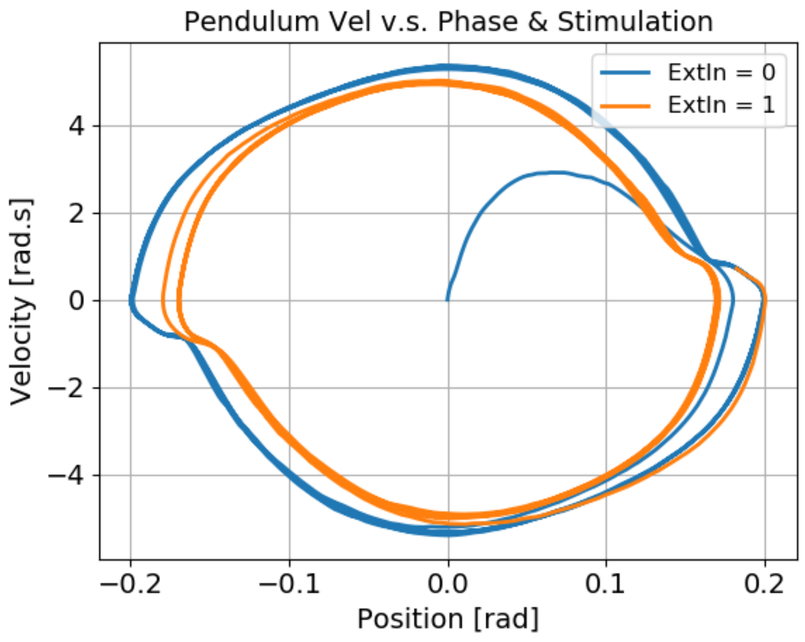


Figure 16: Pendulum Phase Portrait before and after external input is introduced.

3c. [Open Question] What are the limitations of the half center model in producing alternating patterns to control the pendulum? What would be the effect of sensory feedback on this model? (No plots required)

Limitations

This model is idealized and is not robust to the severing of neuronal connections. If a connection is broken between two neurons it will not correctly react anymore. There is also no implementation of any reflexes so it can not well react to unexpected events or external disturbances.

Sensory feedback

The effect of sensory feedback on this model will result in a better imitation of the real behavior of the system and conduct to a more robust system because the neurons could be more implicate in the fine tuning of the parameters at each moment instead of also tune the stability. It will mainly improve when the mechanical characteristics are more like an over-damped system. In this case the strength of sensory feedback will be more important for the behavior than the mechanical properties.

It will also allow to know if the whole system is entirely functional or not by studying the feedback. This leads to redundancy and to have a more robust system. The reflexes of sensory feedback could also be implemented and avoid falling down on some situation like in the "Running over rough terrain" [1] paper where reflex give to the guinea fowl the capability of keeping its dynamic and do not falling down.

[1] "Running over rough terrain": guinea fowl maintain dynamic stability despite a large unexpected change in substrate height Monica A. Daley et al. J ExpBiol2006;209:171-187