Student names: Aurélien Morel, Lucas Wälti, Luca Kiener

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.*

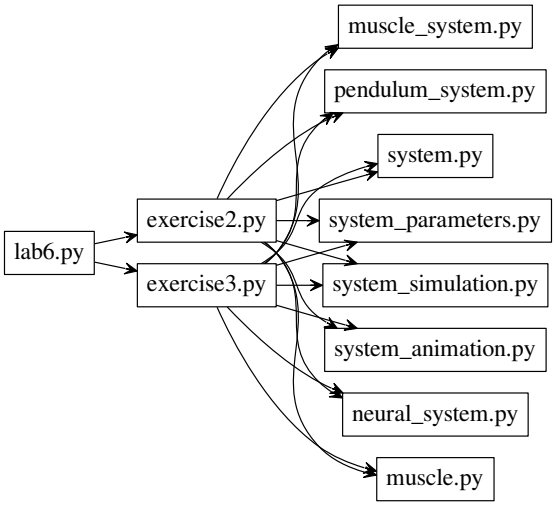


Figure 1: Exercise files dependencies. In this lab, you will be modifying `exercise1.py` *and* `pendulum_system.py`

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

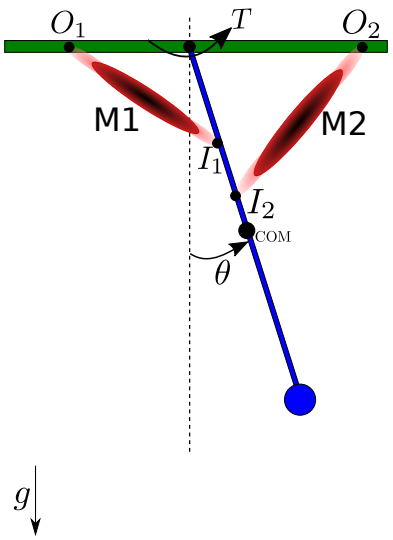


Figure 2: Pendulum with Antagonist Hill Muscles

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

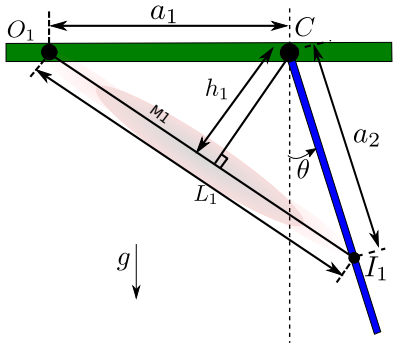


Figure 3: Computation of muscle length and moment arm

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 contracticle 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 contracticle 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.

We see that for asymmetric insertions, the results for both muscle length and moment arm length are not symmetric around $\theta = 0$ so the resting position of the pendulum would be shifted.

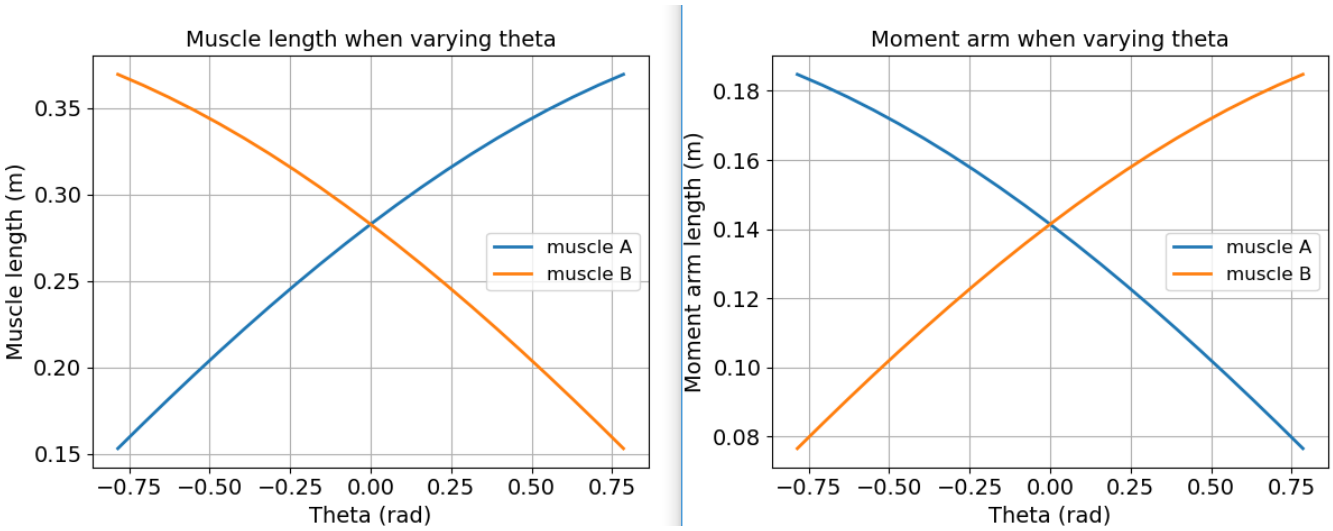


Figure 4: Muscle length and moment arm length with varying θ, symmetric attachment.

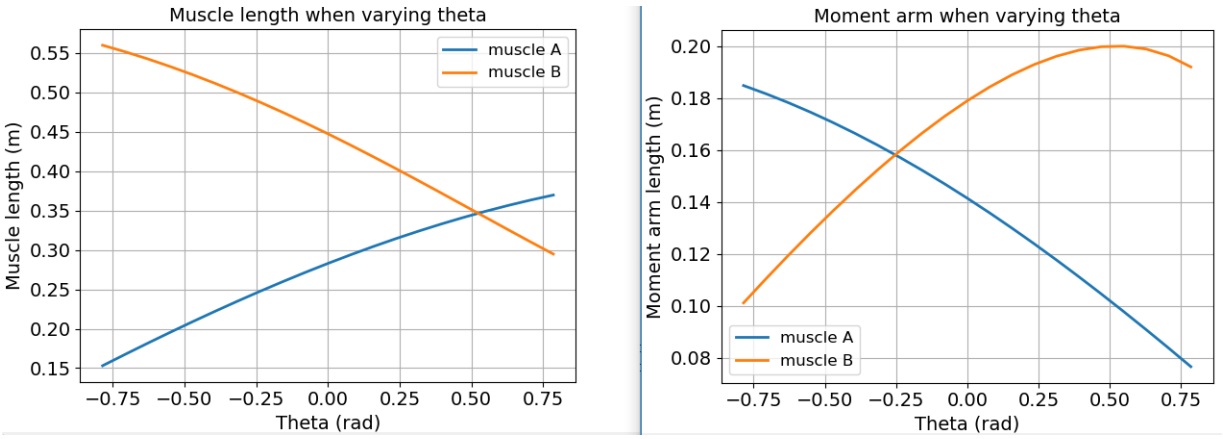
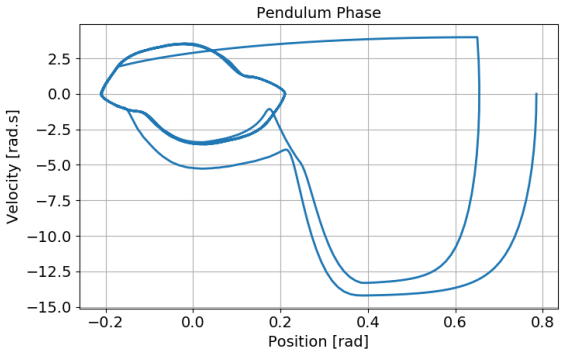


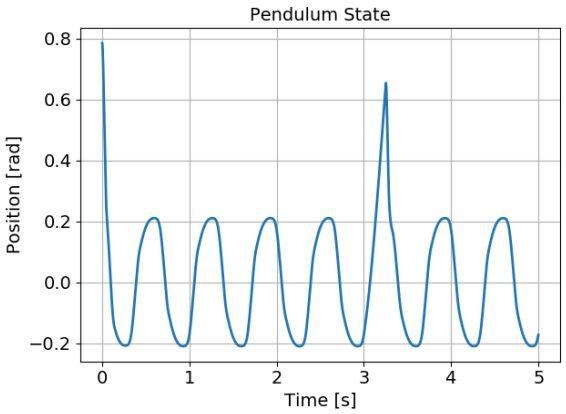
Figure 5: Muscle length and moment arm length with varying θ, asymmetric attachment

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 activations wave forms you used. Use `pendulum_system.py::PendulumSystem::pendulum_system` function to perturb the model.

In order to achieve a limit cycle behavior, we stimulate our muscles with sinusoidal signals (figure 7). To have an oscillatory behavior, we include a phase shift of π between the two stimulations signals. We reach a limit cycle when the pendulum stabilizes as seen in figure 6. The limit cycle is stable: indeed, when we add a perturbation, the pendulum stabilizes and goes back to the limit cycle behavior.



(a) Pendulum state phase with perturbation demonstrating the limit cycle behavior.



(b) Pendulum position as a function of time demonstrating regular oscillations, recovering after the perturbation.

Figure 6: The state-space representation clearly displays a limit cycle behavior. A perturbation is applied but the system converges quickly back to the limit cycle.

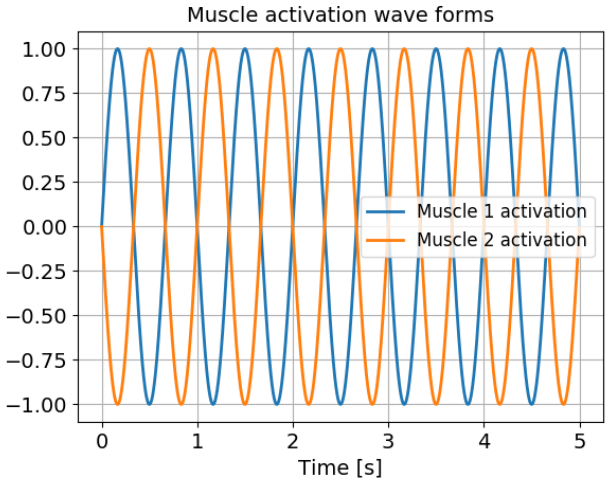
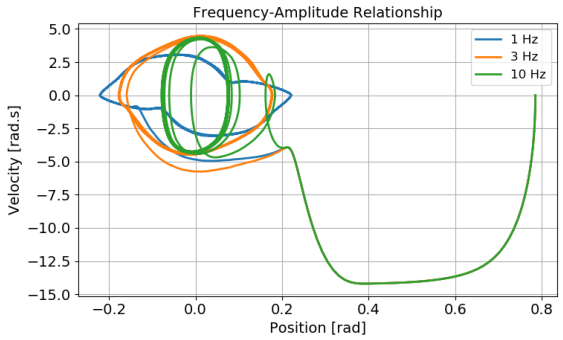


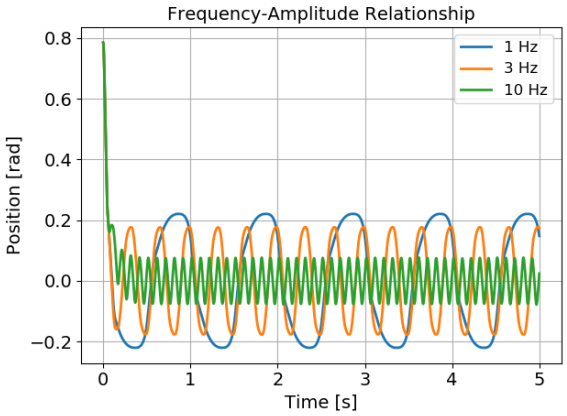
Figure 7: Activation signal sent to each muscle as a sine wave.

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

Figure 8 clearly shows that at higher muscle activation frequency (10 Hz) the amplitude is smaller than at lower muscle activation frequency (1 Hz). It seems logical because at high frequencies, the pendulum has less time to complete its swing motion. Therefore, the motion amplitude is reduced.



(a) State phase plot for different stimulation frequencies.



(b) Pendulum position as a function of time for different stimulation frequencies.

Figure 8: Measurements on the effect of the stimulation frequency on the pendulum behavior.

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 9 shows the network structure and the complete system.

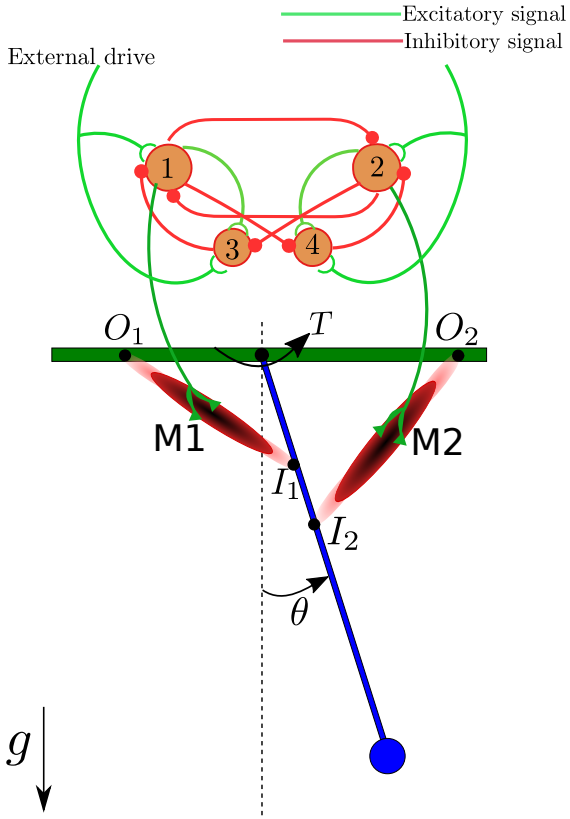


Figure 9: Pendulum with Antagonist Hill Muscles Driven Half Center Neural Network.

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

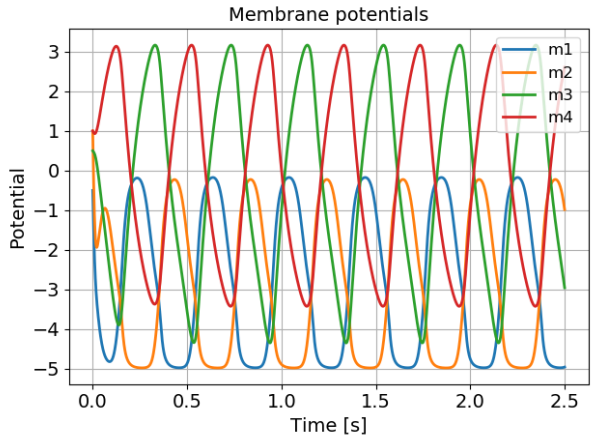
To complete this exercise, additionally you will have to use `neural_system.py` and `exercise3.py`

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

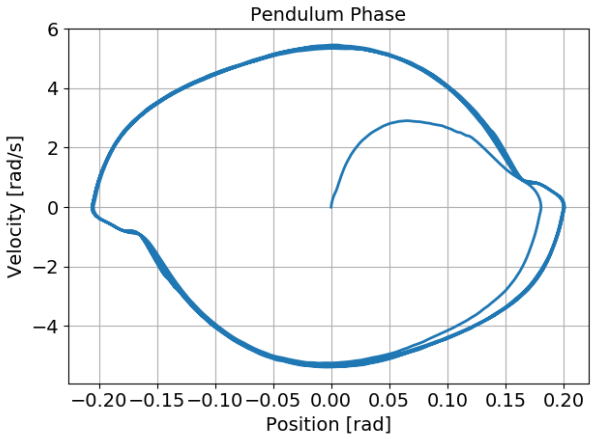
We applied the following parameter values to the network, as discussed in the course:

```
N_params.D   = 2.
N_params.tau = [0.02,0.02,0.1,0.1]
N_params.b   = [3.0,3.0,-3.0,-3.0]
N_params.w = [[0,-5,-5,0],
              [-5,0,0,-5],
              [5,-5,0,-5],
              [-5,5,0,0]]
```

This resulted in the following behavior, as shown in figure 10.



(a) Membrane potential of neurons 1,2,3 and 4.



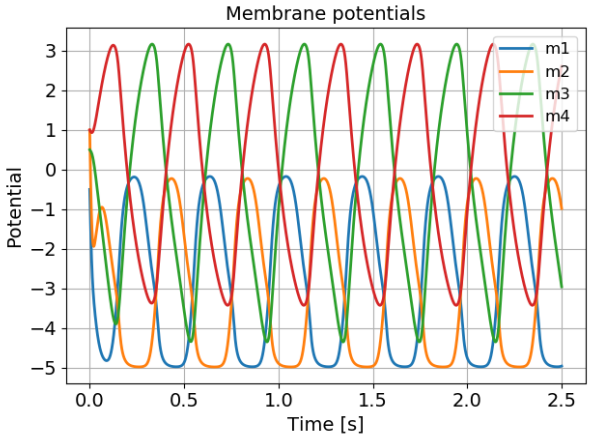
(b) Phase plot of angle and rotation speed of the pendulum illustrating the limit phase behavior of the system.

Figure 10: The output of the network is illustrated in figure 10a while the phase of the pendulum is shown in figure 10b.

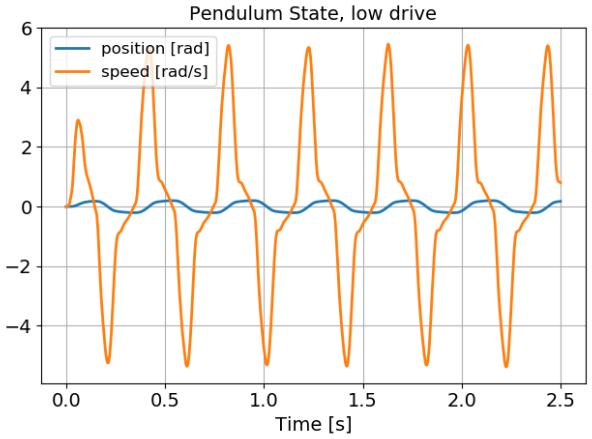
3b. As seen in the course, apply an external drive to the individual neurons and explain how the system is affected. Show plots for low [0] and high [1] external drives. To add external drive to the network you can use the method `system_simulation.py::add_external_inputs_to_network`

Figures 11 and 12 clearly show the effect of the drive. There is a clear change in the oscillations frequency. The oscillations seem to be approximately 25% faster with high drive in comparison with the case where there is no drive.

It is important to note that the amplitude is not affected by the change of drive, only the frequencies are altered.

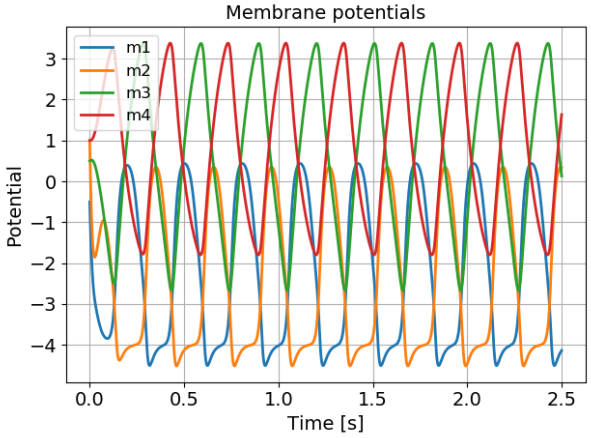


(a) Membrane potential of neurons 1,2,3 and 4.

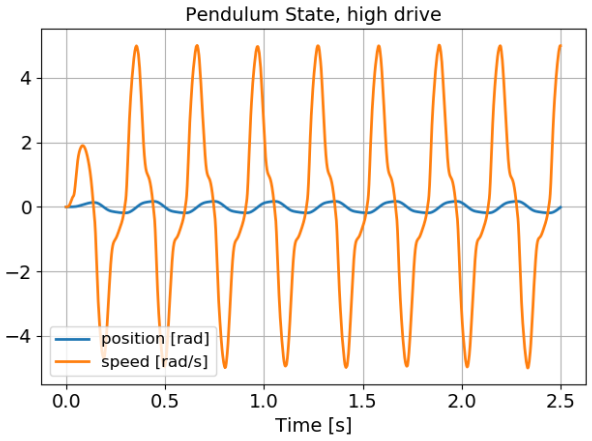


(b) Angle and rotation speed of the pendulum as a function of the time.

Figure 11: Illustration of the behavior for low drive.



(a) Membrane potential of neurons 1,2,3 and 4.



(b) Angle and rotation speed of the pendulum as a function of the time.

Figure 12: Illustration of the behavior for high drive.

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)

This network model does not offer a large variety of frequencies which might be too limiting for a good muscle control. Furthermore, the amplitudes are also an important feature that might need to be adaptable, which is not something the model seems to offer.

A sensory feedback would automatically set the drive level according to some defined stimulus. This would be a way of automatically switching from walk to swimming as we saw in the course. The frequency of the oscillations are faster when swimming than when walking.