Student names:
NIEDERHAUSER Loïc, ROVINA Hannes, INDUMI Mirko

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.*

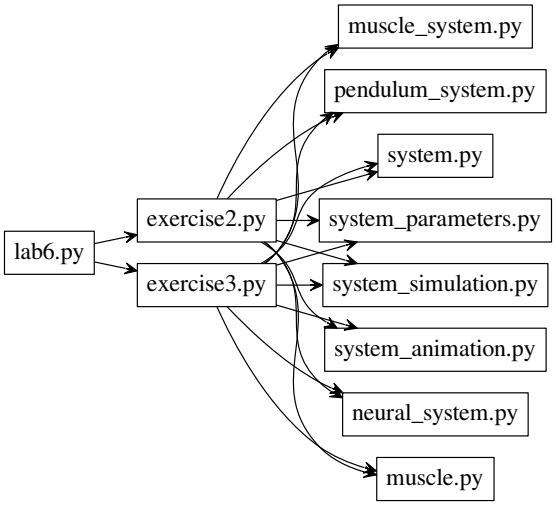


Figure 1: Exercise files dependencies. In this lab, you will be modifying `exercise1.py` *and* `pendulum_system.py`

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

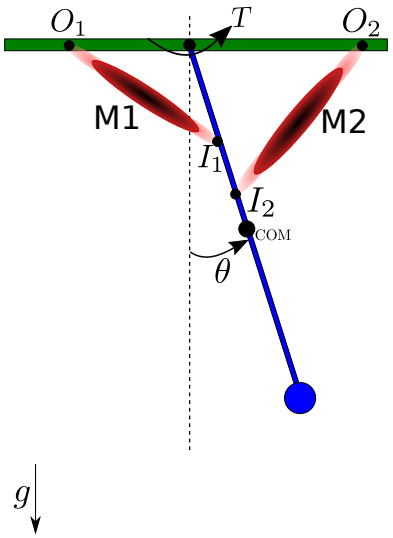


Figure 2: Pendulum with Antagonist Hill Muscles

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

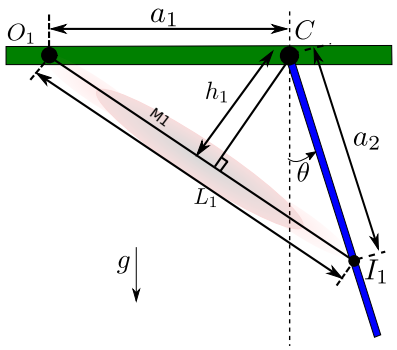


Figure 3: Computation of muscle length and moment arm

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.

From intuition, analysing figure 3, one could expect that with greater angle θ, the muscle length is increasing. At the same time the moment arm is increasing until the pendulum and the muscle form a 90 degrees angle and then decrease again. The implementation of equation 2 leads to the results shown in figure 4. We deduce that our assumptions were correct.

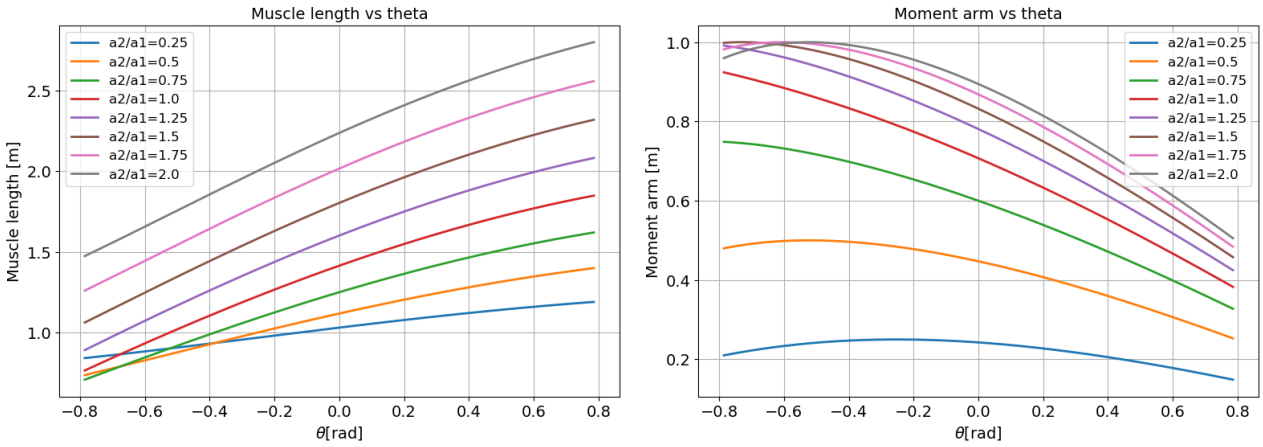
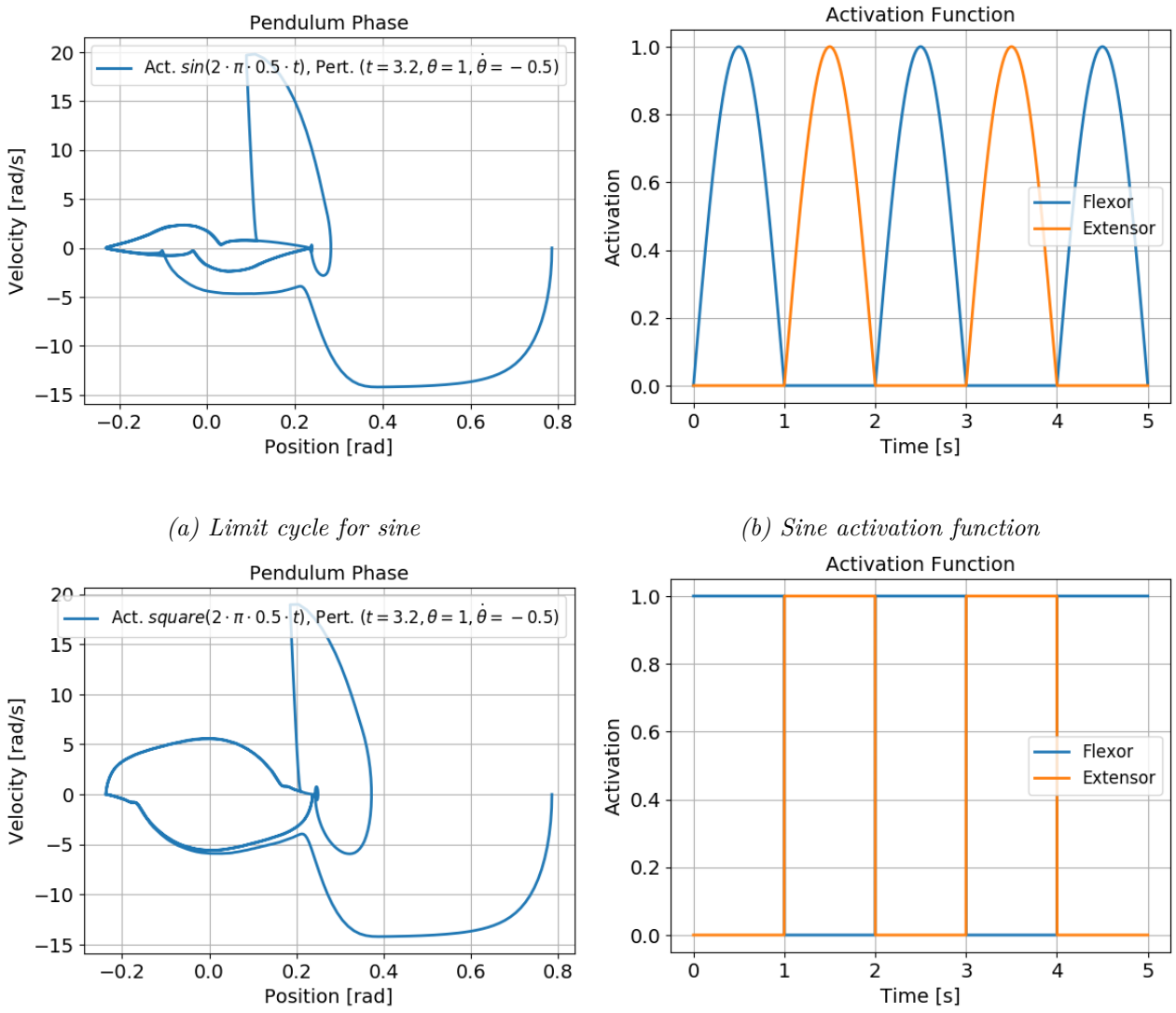


Figure 4: Muscle length and Moment arm in function of theta with varying a2/a1 ratio

For shorter muscles it leads thus to a smaller angle for the muscle at resting position. In addition to that, it can be stated that for small angles, generally higher torques can be applied since the moment arm is longer for constant applied force. This statement can be verified by a simple experiment doing pull-ups. While at the beginning, when the arms are straight, a lot of effort (force) is required to create lift and at the end it is relatively easy to climb over the pull-up bar. This is due to the increasing moment arm while bending the arms.

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.

For the analysis of the limit cycle, the following activation functions were used: First $A \cdot sin(2\pi\omega t)$ and then $A \cdot square(2\pi\omega t)$ with a frequency of $\omega = 0.5Hz$ and amplitude $A = 1$. They are illustrated in figures 5b and 5d.



(a) Limit cycle for sine

(b) Sine activation function

(c) Limit cycle for square

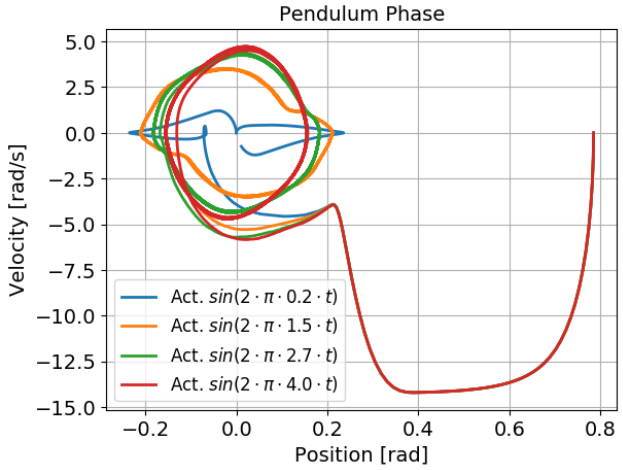
(d) Square activation function

Figure 5: Limit cycle analysis with perturbation for sinusoidal and square wave activation of muscles with stimulation frequency 0.5 Hz

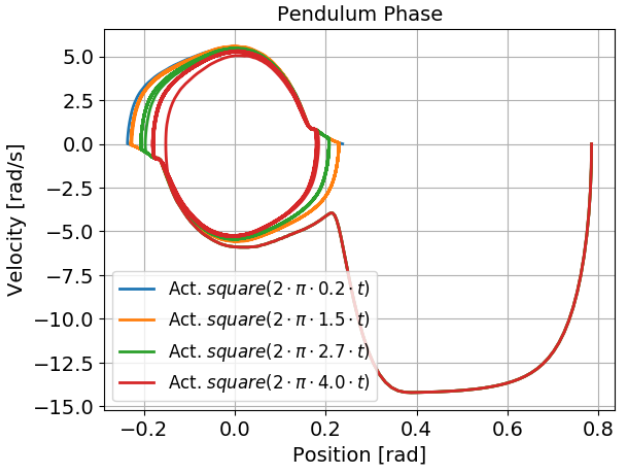
As can be seen in the plots 5a and 5c the pendulum exhibits a limit cycle behaviour. On the left side of the plots, said limit cycle is visible and as illustrated with the rather big perturbation ($\theta = 1, \dot{\theta} = -0.5$) at time step $t = 3.2s$, the response of the system goes back into the limit cycle behaviour.

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

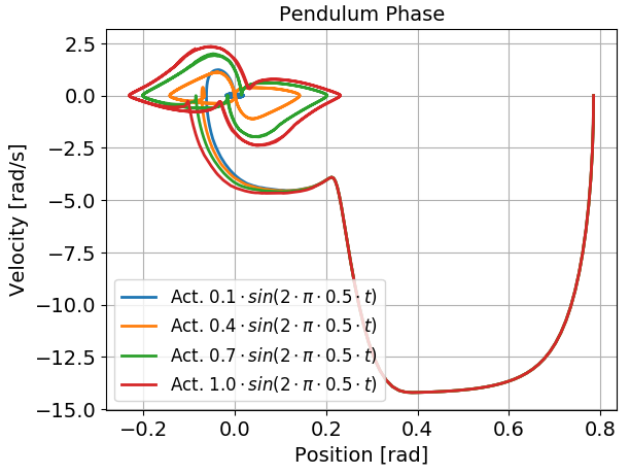
When the frequency of the activation function is increased, the amplitude of the oscillations decreases. This is in accordance with the physical model, since the muscle is stimulated for less time in each cycle but with the same stimulation amplitude as before, it has less time to contract. This is shown in figures 6a and 6b for both aforementioned activation functions.



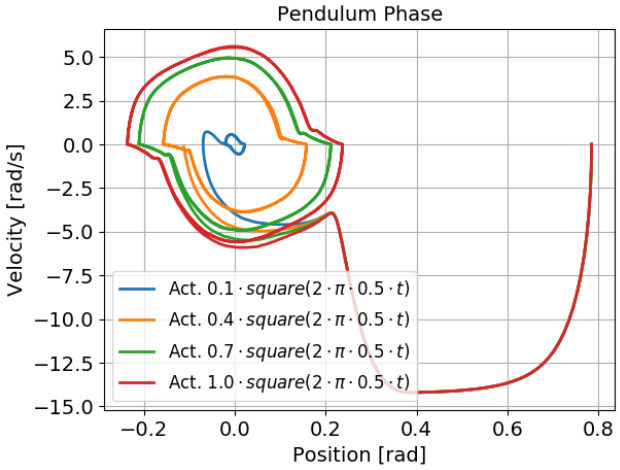
(a) Limit cycle for sine, with increasing ω



(b) Limit cycle for square, with increasing ω



(c) Limit cycle for sine, with increasing A



(d) Limit cycle for square, with increasing A

Figure 6: Limit cycle analysis for different frequencies in the activation function (sinus and square wave) and resulting amplitude of the pendulum

If we vary the stimulation amplitude at a constant frequency of $0.5Hz$ we can see that with increasing stimulation amplitude, the pendulums amplitude increases as well. This corresponds to demanding a larger contraction of the muscles when stimulating with a higher amplitude. The plots in figures 6c and 6d show this behaviour, again for both sinusoidal and square activation functions.

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 7 shows the network structure and the complete system.

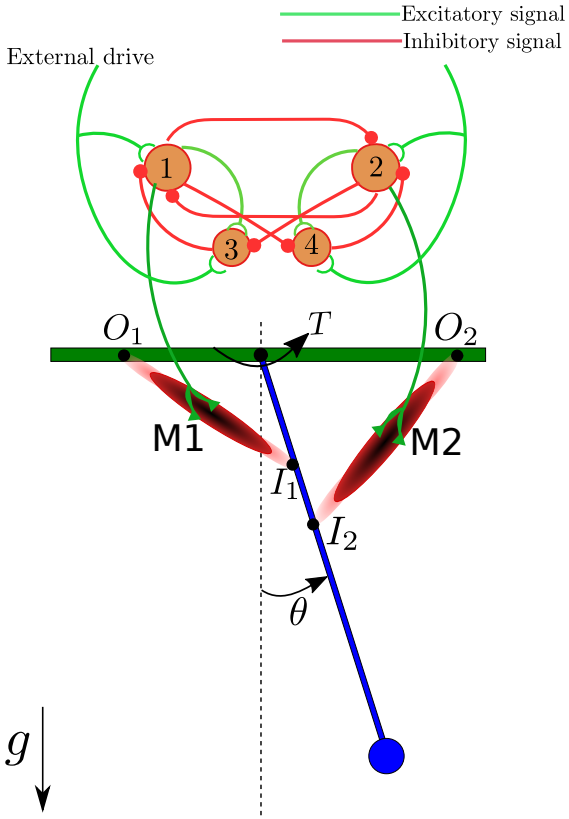


Figure 7: Pendulum with Antagonist Hill Muscles Driven Half Center Neural Network.

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,

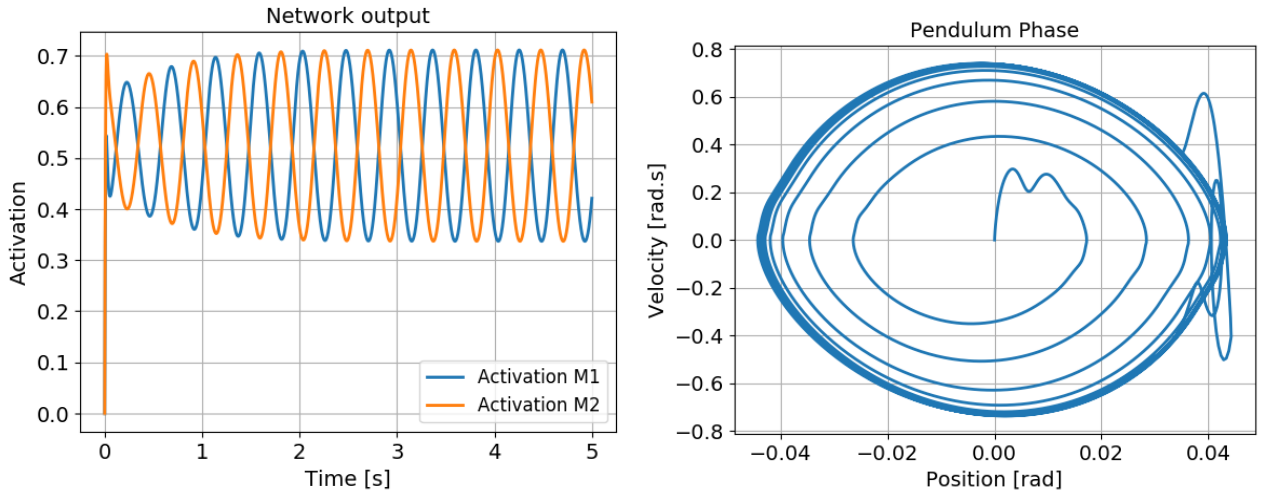
$$X = \begin{bmatrix} \theta & \dot{\theta} & A_1 & l_{CE1} & A_2 & l_{CE2} & m_1 & m_2 & m_3 & m_4 \end{bmatrix} \tag{6}$$

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



(a) Activation functions of flexor and extensor *(b) Limit cycle behaviour with perturbation*

Figure 8: Output of the neurons and pendulum phase plot

In order to obtain an oscillator, two mechanisms are required: Firstly, a pair of neurons inhibiting each other and secondly a fatigue mechanism suspending continuous dominance of one neuron. The pair of neurons used for the output are the neurons 1 and 2 on figure 7. The fatigue mechanism is implemented through a pair of inhibiting neurons with a much larger time constant. When neuron 1 is active, it starts stimulating its "fatigue neuron"(3) which after a short time will inhibit 1 letting the potential of 2 rise.

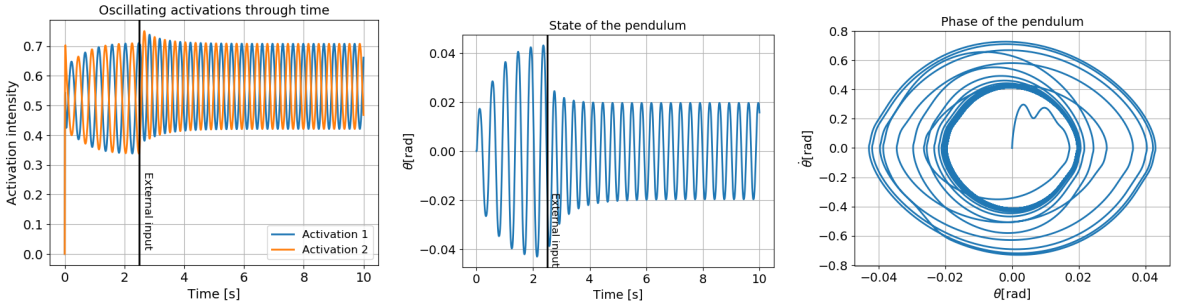
The parameters of the oscillator are implemented according to the lecture slides[1].

- $b = [3, 3, -3, -3]$
- $D = 1$
- $\tau = [0.02, 0.02, 0.1, 0.1]$
- $w = \begin{bmatrix} 0 & -5 & -5 & 0 \\ -5 & 0 & 0 & -5 \\ 5 & 5 & 0 & 0 \\ -5 & 5 & 0 & 0 \end{bmatrix}$

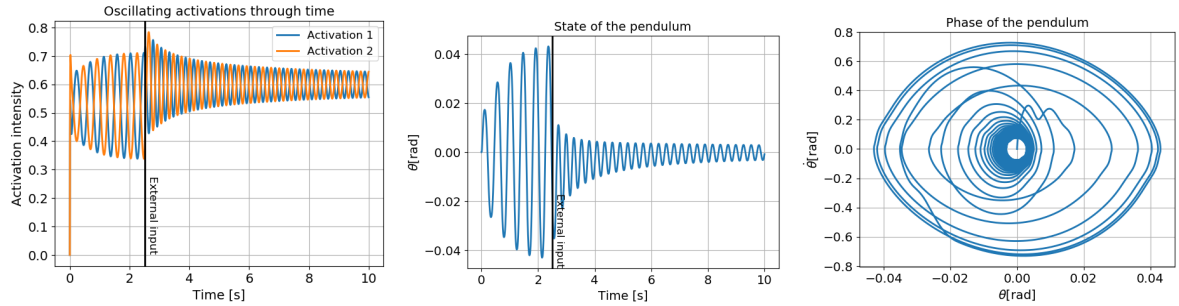
Using these parameters, an oscillating system is generated as shown in figure 8

[1] Lecture 4, Slide 85

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`



(a) Activation function of both muscles with a drive of 0.5 *(b) Oscillation of the pendulum with a drive of 0.5* *(c) Phase of the pendulum with a drive of 0.5*



(d) Activation function of both muscles with a drive of 1.0 *(e) Oscillation of the pendulum with a drive of 1.0* *(f) Phase of the pendulum with a drive of 1.0*

Figure 9: Output of the neurons and pendulum phase plot with an increase in the external drive at $t = 2.5$

Observations: In figure 9 we can see that the application of external drive has a few effects:

- Increase in the average muscle activation. The higher the drive the higher the average activation.
- Increase in the oscillation frequency. The higher the drive the higher the frequency.
- Decrease in the oscillation amplitudes. The higher the drive the smaller the amplitude

Explanations and hypothesis: The increase in average drive is a logical consequence of the drive increase. Indeed having a bigger input on the system should lead to a bigger output in average.
Concerning the frequency: as the drive is higher, the fatigue mechanism is more active. Hence, the "slow" neurons' potential are rising faster. The frequency is therefore increased.
The decrease in amplitude is linked to the increase in frequency. The oscillation being faster, the neurons have less time to increase their potential in each cycle leading to a smaller amplitude.

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)

As we can see on figure 9, we deduce that with increasing external drive the muscle activation (force) and the frequency is increased. Both are coupled. Let's assume that we would like to build a legged

robot using this Brown's half-center model that should walk up a slope and walk straight forward at the same speed. If our robot is about to walk up the slope, more force has to be applied to the muscle. Therefore the external drive has to be increased. But this leads in the same time also to a faster walking robot doing smaller steps. The smaller step cycle could be problematic for a robot that is not robust anymore at this sampling time. This is just an example where the coupling between the cycle frequency and the neuron activity could be seen as a limitation [2].

With sensory feedback, training of the set of weights for the neural network could be possible. Therefore precise movement adapted for a specific task could be possible. Furthermore the system will be better in rejecting disturbances.

[2]Source: David A. McCreaa, Ilya A. Rybakb, Organization of mammalian locomotor rhythm and pattern generation, Published online 2007 Sep 5. doi: 10.1016/j.brainresrev.2007.08.006 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2214837/