Student names: . . . (please update)

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 need to be submitted before the* ***Deadline : 23-04-2018 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 lab5_name1_name2_name3.zip where name# are the team member's last names. Please submit only one report per team!*

In the previous week you explored the behavior of a simple pendulum model with passive elements such as springs and dampers and then explored the properties of a single but more realstic muscle model with both active and passive components.

The main goal of this exercise is to explore the behavior of the pendulum model attached with two antagonist Hill-type muscles and then connect a half-center neural network model to drive the muscles in the pendulum. The system is as shown in figure 1.

Files to complete the exercises

- `lab5.py` : Main file
- `exercise3.py` : Main file to complete exercise 3
- `exercise4.py` : Main file to complete exercise 4
- `SystemParameters.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)
- `PendulumSystem.py` : Contains the description of pendulum equation and Pendulum class. You can use the file to define perturbations in the pendulum.
- `MuscleSystem.py` : Class to combine two muscles (You do not have to modify the file)
- `NeuralSystem.py` : Class to describe the neural network (You do not have to modify the file)
- `SystemSimulation.py` : Class to initialize all the systems, validate and to perform integration (You do not have to modify the file)
- `SystemAnimation.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 3 : Pendulum model with Muscles

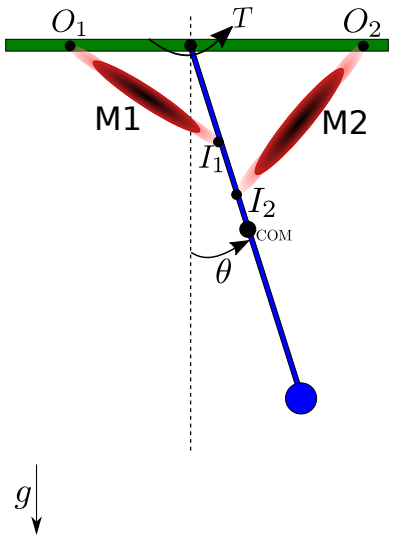

Figure 1: Pendulum with Antagonist Hill Muscles

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).

decreases

increases

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 $[m]$

Difference between O and I => gives L changes wrto pendulum position ==> gives momentum that we need to compute forces.

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

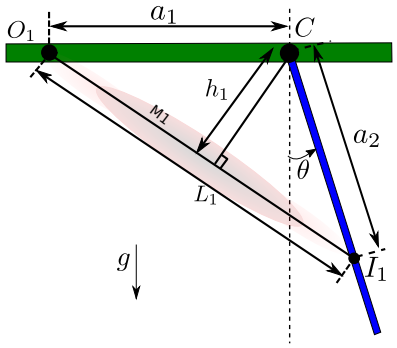

Figure 2: Computation of muscle length and moment arm

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, `exercise3.py`, `SystemParameters.py`, `Muscle.py`, `System.py`, `PendulumSystem.py`, `MuscleSystem.py`, `SystemSimulation.py`

3a. For a given set of attachment points, compute and plot the muscle length and moment arm as a function of θ between $[-\pi/2, \pi/2]$ using equations in eqn:2 and discuss how it influences the pendulum resting position and the torques muscles can apply at different joint angles.

3b. Under passive conditions (muscle is not activated), Discuss the effect of muscle attachment points on the pendulum's behavior. You can use the results from previous question to support and explain the pendulum's behavior.

The resting point is not the same as when activation for same params

3c. Using simple activation wave forms (example : sine or square waves) applied to muscles (use `SystemSimulation.py::add_muscle_activations` method in `exercise3.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. If needed, use `PendulumSystem.py::Pendulum::derivative` function to perturb the model.

3d. Based on the previous limit cycle, show how the physical properties of the pendulum such as mass, length and inertia (Inertia is a result of mass and length) affect the amplitude and frequency of the pendulum oscillations for a given stimulation.

3e. Discuss the relationship between stimulation frequency and amplitude with the resulting pendulum's behavior.

3f. Show how the pendulum's behavior is affected by the maximal muscle force F_{max} for a given stimulation.

When Fmax is too low (depending on the pendulum's mass amongst others), the pendulum's weight has more contribution to the pendulum's behaviour, but is still damped by the muscles, until the point where the oscillatory amplitudes aren't big enough and the contribution of gravity drops below the active force generated by the muscle, which corresponds to the moment where the system enters a limit cycle behaviour.

So for a big enough F_max, one expects the limit cycle to take place faster in the timecourse.

Exercise 4 : Neural network driven pendulum model with muscles

In this exercise, the goal is to drive the above system 1 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 3 shows the network structure and the complete system.

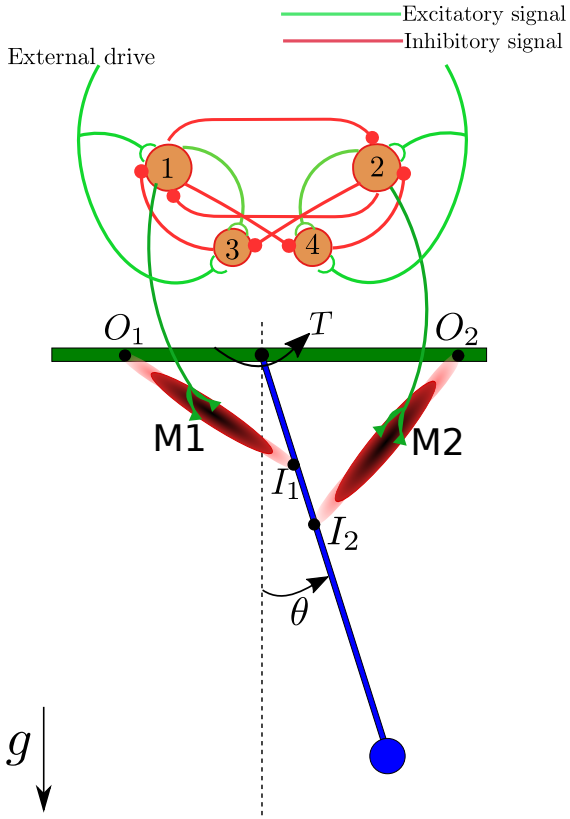

Figure 3: Pendulum with Antagonist Hill Muscles Driven Half Center Neural Network.

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 `NeuralSystem.py` and `exercise4.py`

4a. Find a set of weights for the neural network that produces oscillations to drive the pendulum into a limit cycle behavior. Tâtonnement

4b. Show how the network weights ,bias and time constant parameters affect the behavior of the pendulum.

4c.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
`SystemSimulation.py::add_external_inputs_to_network`

4d. [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?