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 needs to be submitted before the* ***Deadline : 03-06-2018 Midnight****.*
You only need to submit one final report for all of the following exercises combined henceforth.
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 final_report_name1_name2_name3.zip where name# are the team member's last names. Please submit only one report per team!*

Mouse Model

In this exercise you will be working with the hind-limb musculoskeletal model of the mouse. The goal of the following exercises will be to develop and understand how reflexes alone can generate locomotion behavior. In order to do this we use the work of [1] to develop the reflex model. Here you will integrate the concepts learnt during the course to produce locomotion behavior.

Model Description

For the purpose of these exercises you will be using the mouse model show in figure 1. Since the main focus of the exercises is to study hind limb locomotion, the fore limbs of the mouse are rigidly fixed in a default position and only aid the mouse in balance and stability during locomotion. The spine, head and pelvis are also fixed a default position during locomotion. The hind limbs are the most important entities of the following exercises.

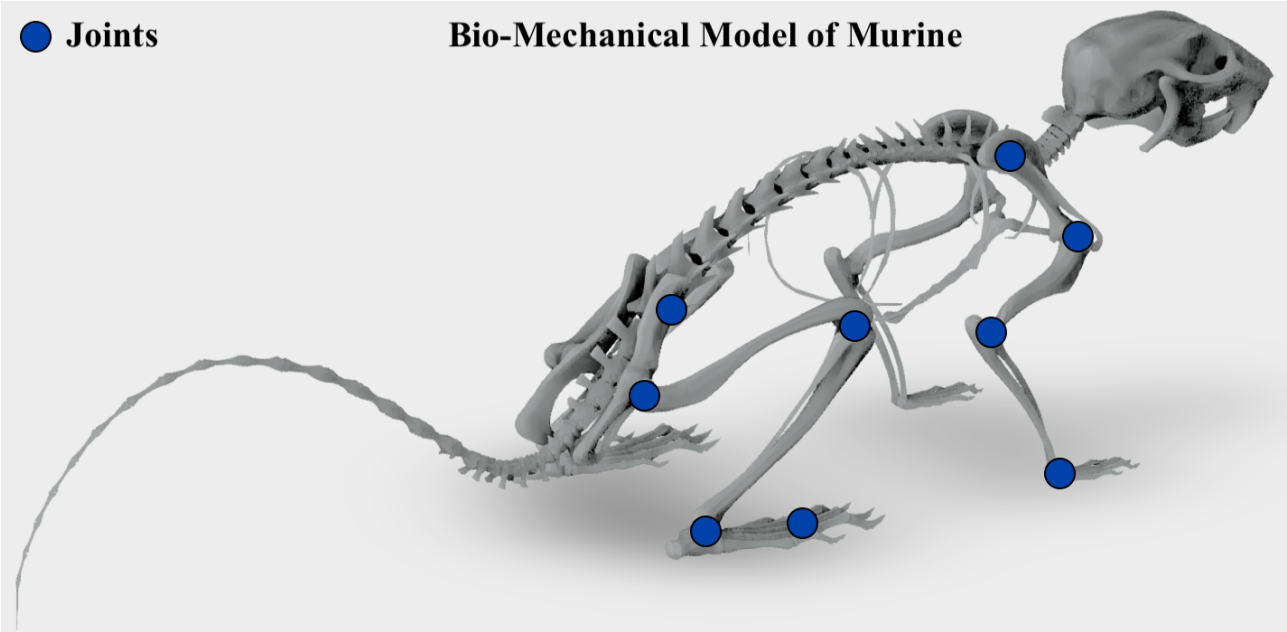


Figure 1: Mouse Model used in the exercises

Figure 2 shows the joints and muscles considered for each of the two hind limbs in the mouse model.

Joints

Each hind limb consists of four revolute joints,

- Hip
- Knee
- Ankle
- MTP

Out of the four joints listed above, every joint actuated by at least one pair of antagonist muscle pairs expect for the MTP joint. MTP joint is passive and contains only a spring-damper to allow the joint to return to its initial position. To simplify the model, simpler bounding box shapes are used to wrap the complex bone meshes.

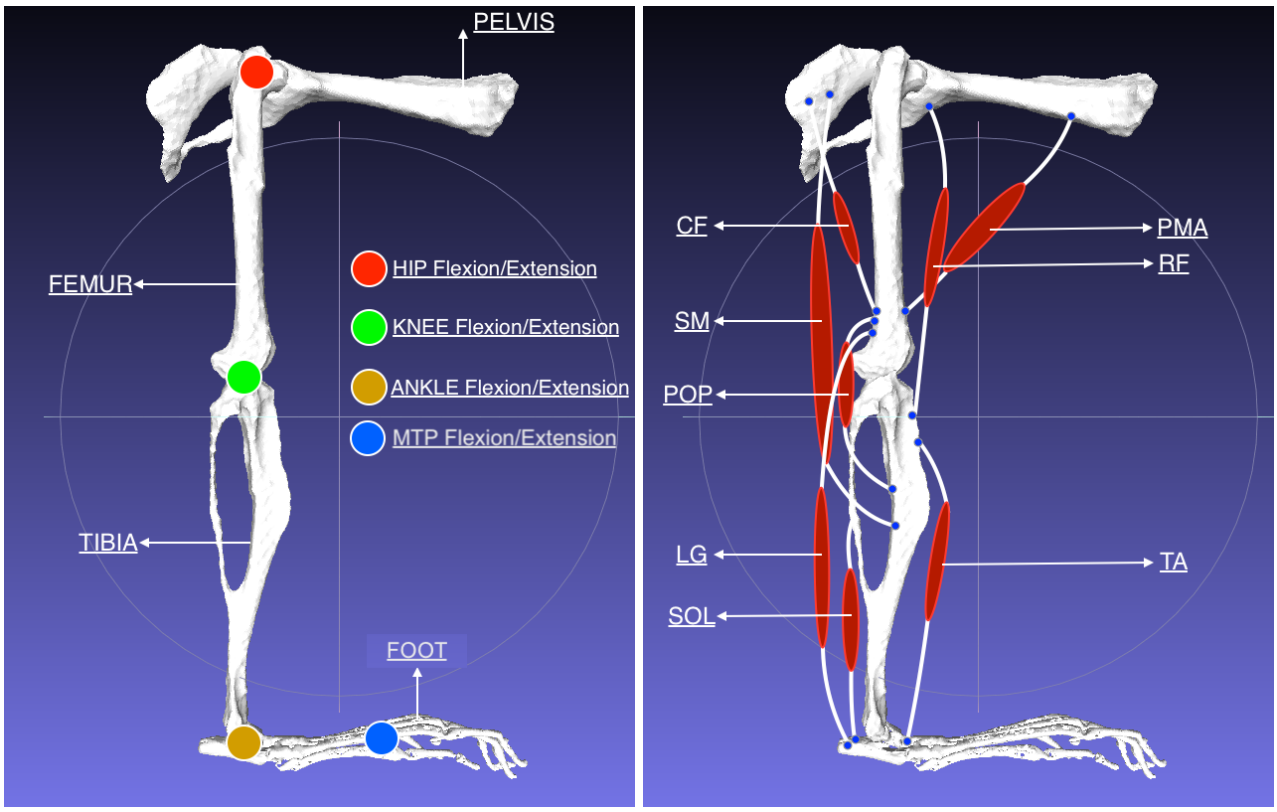
Muscles

Each joint mentioned above is actuated by at least one pair of antagonist muscles so that the joint can have the complete range of motion. The set of muscles in each hind limb are show in figure 2b. A mono-articular muscle applies a joint torque on only one joint and a bi-articular muscle spans over two joints and applies a torque on both.

To summarize, table 1 shows the list of muscles chosen for simulation along with the joints about which they act and their function on these joints in the hind limb.

Muscle	**Abbreviation**	**Type**	**Hip**	**Knee**	**Ankle**
Psoas Major	PMA	Mono-articular	Flexion	-	-
Caudofemoralis	CF	Mono-articular	Extension	-	-
Semimembranosus	SM	Bi-articular	Extension	Flexion	-
Popliteus	POP	Mono-articular	-	Flexion	-
Rectus Femoris	RF	Mono-articular	-	Extension	-
Tibialis Anterior	TA	Mono-articular	-	-	Dorsiflexion
Soleus	SOL	Mono-articular	-	-	Plantarflexion
Lateral Gastrocnemous	LG	Bi-articular	-	Flexion	Plantarflexion

Table 1: Summary of mouse hind limb muscles used in simulation



(a) Mouse hind limb joints *(b) Mouse hind limb muscles*

Figure 2: Mouse hind limb joints and muscles description

Model Parameters

The model consists of two sets of important parameters. One, the skeletal parameters and second the muscle parameters. All parameters for joints and muscles are described in musculoskeletal::mouse.json file.

Skeletal Parameters:

- $mass$: Mass of the bone $[kg]$
- $size$: Bone length, width and breadth $[m]$. Mesh bounding box defines these elements in the simulation.
- $inertia$: Moment of inertia $[kg - m^2]$
- $centerofmass$: Center of mass of each segment $[m]$

NOTE : Refer to section Model Scaling for units used in the simulation

The properties of each skeletal segment can be found in the world editor in Webots .wbt file.

The muscle parameters are defined in a parameter .json file. The json file contains the following parameters for each muscle and joint present in the hind limb.

Muscle Parameters:

- l_{opt} : Muscle optimal fiber length $[m]$
- l_{slack} : Muscle tendon slack length $[m]$
- $theta\_ref(\theta_{ref})$: Joint angle at which muscle is at its resting length $[deg]$

- *theta_max*(θ_{max}) : Joint angle at which maximal muscle moment arm [deg]
- *pennation* : Fiber angle [deg]
- r_0 : Maximum muscle moment arm [m]
- f_{max} : Maximum muscle force [N]
- v_{max} : Maximum muscle velocity [l_{opt}/s]
- *joint_attach* : Joint to which the muscle exerts a moment
- *direction* : '*clockwise / cclockwise*' : Direction of muscle moment
- *muscle_type* : '*mono*' if muscle spans across only one joint. '*bi*' if muscle spans across two joints. If muscle is bi-articular then r_{02}, $theta\_ref2$, $theta\_ref2$, $joint\_attach2$ and $direction2$ must be additionally defined for the second joint.

Joint Parameters:

- *joint_type* : 'CONSTANT' - Muscle moment arm computation type
- *theta_min* : Minimum joint angle [deg]
- *theta_max* : Maximum joint angle [deg]
- $reference\_angle$: Offset in joint angle [deg]

Model Scaling

As you have seen in Lab 1 of this course, numerical integration have different sources of numerical errors and approximations. One source is the magnitude of numbers being integrated. Numbers that are closely to numerical precision of the system can quickly lead to numerical rounding-off errors. In the actual mouse model, parameters such as masses and inertia's are of the order 1e-8 to 1e-10. Such low numbers are usually a cause for errors. One solution to avoid these errors is by changing the units of the system. In the Webots simulator, the following units are used :

Parameter	**Old**	**New**	**Scale factor**
Length	m	cm	100
Mass	Kg	g	1000
Force	Kg - m/s^2	g - cm/s^2	1e5
Moment of Inertia	$Kg - m^2$	$g - cm^2$	1e7
Angle	radians	radians	1
Torque	Kg-m/s^2	g-cm/s^2	1e7
Time	s	s	1

Table 2: Scaling of parameters for mouse model

Reflex Model

The goal of the following exercises is to replicate the reflex model similar to [1]. In their work, the gait if divided into four sub stages as shown in figure 3.

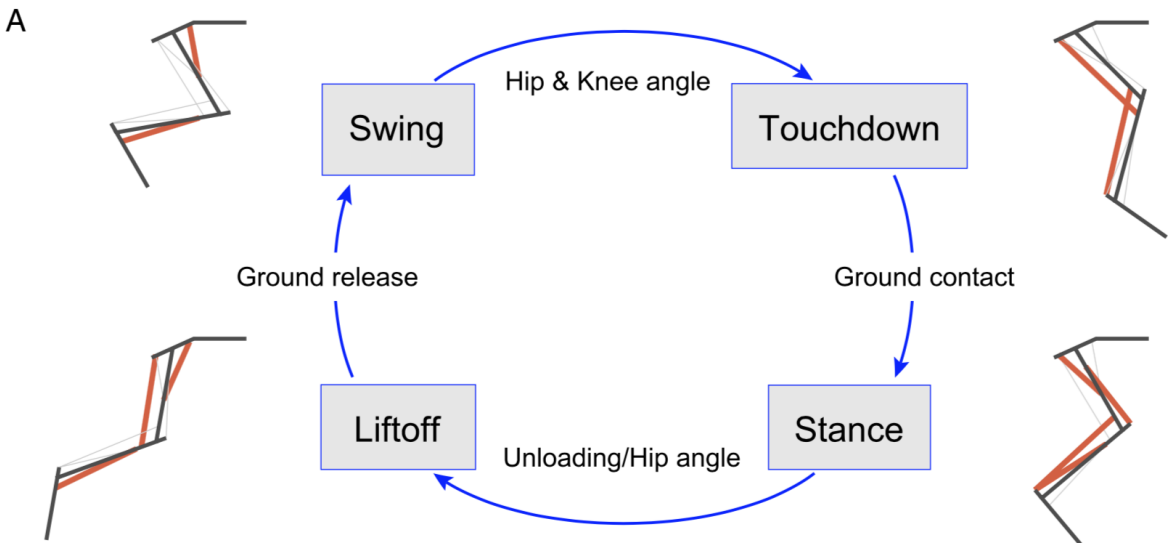


Figure 3: Four stages of gait cycle as defined by [1]

- SWING
- TOUCH-DOWN
- STANCE
- LIFT-OFF

A predefined set of muscle activations make the transition between each of the above mentioned states. The timing between transitions is governed by the feedback the reflex model receives. The reflex model receives three main feed backs.

- Muscle forces from each muscle in the hind limb
- Joint angles from each each joint in the hind limb
- Foot ground contact from the hind limb

In this model, from the feed backs mentioned above we explore the following feed backs.

- Hip and Knee joint angle
- Ankle muscle forces
- Foot ground contact

Using the above feed backs, we define the reflex conditions for transition between the four stages of the gait and eventually produce stable locomotion.

During the exercises in the file reflexes.py you need to tune the muscle activations to obtain the phase transitions and then eventually locomotion.

Code organization

- Path to the controller files and modules - Lab7::Webots::controllers::mouse
- musculoskeletal::MusculoSkeletalSystem.py - Class to read, initialize and update muscles joint system.
- musculoskeletal::Joint.py - Class describing the Joints in the system
- musculoskeletal::Muscle.py - Class describing the Muscle model in the system
- musculoskeletal::MuscleJoint.py - Class describing the interface between muscles and joints. Converts muscle forces to joint torques.
- musculoskeletal::SystemParameters.py - Class containing the access properties for system parameters.
- mouse.py - Main class containing the Webots controller file. You can implement all your code in this file. Or implement in an external file and then do function calls from within this file as per your convenience.
- reflexes.py - Class containing the reflex model. This is where you need to tune the muscle activations
- muscle_visualization.py - Class for visualizing the muscles. You do not have to edit this file.

Running the simulation

Make sure you have successfully installed Webots by following the instructions outlined in Lab 6

Complete the tutorial and practice examples of Webots as outlined in Lab 6

Open the cmc_mouse.wbt file in Webots. This should launch the mouse model in simulation world

Compiling the physics plugin to make the mouse fixed

You will need to compile the physics plugin in Webots that allows you to fix the mouse in air. To do this follow the steps below,

- Navigate to the WorldInfo in the SceneTree
- Expand the WorldInfo Node
- Click on the physics entity
- A small dialog should now appear below. Now choose Edit option in the dialog
- This should open a text editor window
- Now go to Build in the main menu and choose the option *Clean*
- Next in the same Build menu choose the option *Build*
- This should now compile and build the fix_mouse physics plugin.

Running the simulation

Now when you run the simulation, the hind limb legs should drop while the mouse is fixed in air. At this point you can now start to tune your reflex model.

Explore the main mouse controller file mouse.py and understand how the controller is set-up

mouse.py steps the Webots simulation at set time step of 1ms. During each step, the controller calls the pre-initialized musculoskeletal::MusculoSkeletalSystem.py class and updates. This update computes the muscle forces based on the joint angles and muscle activations received from the reflex controller class. Then the joint torques from the update are applied to Webots rotational motors. This cycle is repeated every time step.

Questions

7a. Now that you have the simulation model set-up, the first goal is to tune the muscle activations for each of the four gait stages/phases mentioned in the reflex model description 3. To do so, in reflexes.py::step start un-commenting each phase and then tune the muscle activation constants (Kx) ranging between [0-1] to make the hind limbs move to the particular posture of the phase. Once you tune a particular posture comment the same lines in reflexes.py::step and repeat the process for other phases. At this point it is not necessary to be very precise with the parameter tuning. Report the parameters used and show a picture of the model for each of the four phases

7b. Once you have all the phases tuned individually, uncomment the left and right leg state transition lines in reflexes.py::step. (Note : Make sure to comment the individual phase tuning lines in reflexes.py::step). You should now observe a walk like gait if everything goes well. Global position of the hind feet is saved in the controllers::mouse::Results by default after you revert the simulation. Plot the trajectory of both the hind feet and explain if the behavior is a limit cycle or not. You may use the function controllers::mouse::Results::load_data.py to read the saved data.

REFERENCES

1. *Computer Simulation of Stepping in the Hind Legs of the Cat: An Examination of Mechanisms Regulating the Stance-to-Swing Transition* : Orjan Ekeberg and Keir Pearson. Journal of Neurophysiology 2005 94:6, 4256-4268 download link

APPENDIX

Muscle Attachment Simplification

In Lab 5 you have explored the role of muscle attachment points across a given joint. From the results it was also clear that the role of muscle attachment can be analytically pre-computed. In this model, an analytical approach is used to pre-compute the change in muscle length and muscle moment arm for a given joint angle.

Moment Arm

The force produced by the muscles needs to be transformed into a moment before being applied to the joints.

$$\tau = M_{\theta \to \tau}(\theta, \theta_{max}) \times F_{mtu} \tag{1}$$

Where,

- τ : Moment/Torque produced by the muscle $[N - m]$
- $M_{\theta \to \tau}(\theta, \theta_{max})$: Moment Arm as a function of θ and $\theta_{max}[m]$
- θ_{max} : Joint angle at which maximum moment is applied $[rad]$
- F_{mtu} : Force produced by muscle tendon unit $[N]$

Equation 1 shows the transfer of muscle force to joint torque. In order to simplify the model and the complexities that arise due to the muscle moment arm change, we simplify the equation 1 by avoiding the joint torque as function of joint Angle θ. Equation 3 shows the simplified equation where moment arm is now a constant maximum value r_0.

$$M_{\theta \to \tau}(\theta_{max}) = r_0 \tag{2}$$

$$\tau = r_0 \times F_{mtu} \tag{3}$$

Where,

- r_0 : Maximum muscle moment arm $[m]$

Muscle Length

As with the moment arm, change in muscle length can be also be defined as a function of joint angle. Equation 4 shows the how the change in muscle length can be computed as a function of joint angle.

$$\triangle l_{mtu} = pennation \times \int_{\theta_{ref}}^{\theta} M_{\theta \to \tau}(\theta_{max}) \tag{4}$$

Substituting 2 in 4 and solving the integral, we obtain the equation for change in muscle length as a function of joint angle. The equation is shown in 5

$$\triangle l_{mtu} = pennation \times r_0 \times (\theta - \theta_{ref}) \tag{5}$$

Where,

- $\triangle l_{mtu}$: Change in muscle length $[m]$
- $pennation$: Angle of muscle fibers in the muscle $[rad]$
- r_0 : Maximum muscle moment arm $[m]$
- θ_{ref} : Joint angle at which muscle is at its resting length $[rad]$

Finally the muscle length is computed as,

$$l_{mtu} = l_{opt} + l_{slack} + \triangle l_{mtu} \tag{6}$$

Where,

- l_{mtu} : Muscle Tendon Unit length $[m]$
- l_{opt} : Muscle optimal fiber length $[m]$
- l_{slack} : Muscle tendon slack length $[m]$