Computational Motor Control, Spring 2017
Webots exercise, Lab 8, GRADED

Student Names: … (please update)

Instructions: Update this file (or recreate a similar one, e.g. in Latex) to prepare your answers to the questions. Add text, equations and figures as needed. Please submit both the source file (.doc) and a pdf of your document, as well as all the used and updated Matlab functions in a single zipped file called lab8_name1_name2_name3.zip where name# are the team member's last names. Please submit **only one report per team**!*

11 Walking with Salamandra Robotica – CPG Model

In the previous exercise you made Salamandra Robotica swim and in the next step you will extend its controller to make it walk. To achieve that, you need to extend the CPG network you created last time with limb oscillators. Although similar to the previous exercise, please use newly provided Matlab scripts.

11.A Implement limb oscillators

In Lab 7 you implemented a double chain of oscillators, which was used to control the spine of Salamandra Robotica. Now you should add 4 more oscillators (1 per limb) to control the limb motion. The final CPG network should have 24 oscillators. Couple those oscillators among each other and to the spine as you did in Lab 5. For walking, use a frequency of **1Hz**.
Notice that each limb has only one rotational degree of freedom and its motion is directly controlled by the oscillators' phase, which is done in `matlabController.m`:

```
% limb angles correspond to limb phase
qlimb=theta(21:24);
```

1) Expand the CPG network in the function `runCPGNetwork`. Your output vectors should now have size of 20+4. **Include your implemented code from this function in the report.** For the parameters check slide 22, lecture 6. Test your CPG by running the Matlab simulation.

Hint 1: Notice that the intrinsic frequency **f** and nominal radius **R** are now vectors with 24 elements (values assigned to each of the oscillators). For questions 11.A and 11.B, their elements can all have the same values. In 11.C you will need to modify the values assigned to limb oscillators.

11.B Limb – Spine coordination

In this next part you will explore the importance of a proper coordination between the spine and the limb movement.

1) Analyze the spine movement: What are your phase lags along the spine during walking? How does the spine movement compare to the one used for swimming?

2) Notice that the phase between limb and spine oscillators affects the robot's walking speed. Run a parameter search on the phase offset between limbs and spine. Set the nominal radius **R** to 0.3.

Include plots showing how the phase offset influences walking speed and comment the results. How do your findings compare to body deformations in the salamander while walking?

3) Explore the influence of the oscillation amplitude along the body with respect to the walking speed of the robot. Run a parameter search on the nominal radius **R** with a fixed phase offset between limbs and the spine. For the phase offset take the optimal value from the previous sub-exercise. While exploring **R**, start from 0 (no body bending).

Include plots showing how the oscillation radius influences walking speed and comment on the results.

Hint 2: In the parameter search explore at least 10 different values of the search parameter. To perform it, you can use a grid search functions. To limit the grid search to only one dimension (representing the parameter you want to explore), set the second parameter of the function `initializeGridSearch` to a scalar value.

11.C Land-to-water transitions

The CPG model in Lab 5 was able to switch the robot's behavior between swimming and walking by using a high level drive signal and a saturation function.
While walking, due to a high coupling strength between the limbs and the spine, the spine oscillators are forced to follow the limb oscillators. By reducing the nominal radius of the limb oscillators, the spine oscillators are able to respect the commanded phase lags along the spine.
In this part of the exercise you need to implement a mechanism which will allow for switching between walking and swimming.

Before starting, move the robot in Webots closer to water, e.g. set the x coordinate of the robot's **translation** field to -3m. The land-to-water transition line is passing through point (0,0,0)m and is parallel to the z-axis.

1) Implement a switching mechanism in the function `limbCPGsaturation`. The function receives intrinsic frequencies **f** and **f_limb**, nominal amplitudes **R** and **R_limb** and **x-coordinate** of the robot in the world retrieved from a GPS reading. Based on the GPS reading, you should determine if the robot should walk (it's on land) or swim (it reached water). Depending on the current state of the robot, you should modify **f, f_limb, R** and **R_limb**. Just by modifying these parameters, the robot should perform the transition. **Include your implemented code from this function in the report.**

2) Run the Webots simulation and report spine and limb angles, together with the x coordinate from the GPS signal. Record a video showing the transition from land to water and submit the video together with this report.

3) (BONUS) Achieve **water-to-land** transition. Report spine and limb angles, the x-coordinate of the GPS and record a video.

Hint 3: You can reuse parts of the code from the Lab 5, specially the `saturation_function`.

Hint 4: Experiment with a point (x-coordinate) where the transition occurs (it does not have to be strictly at the line where water starts). Furthermore, try to have a smooth transition between walking and swimming (by gradually changing the **f, f_limb, R** and **R_limb** as a function of the **x-coordinate**).

Hint 5: Use Webots' internal video recording tool to easily record videos.

Hint 6: The function `limbCPGsaturation` returns a Boolean flag **is_swimming** which can be used in `matlabController` to set the limbs into swimming position (by modifying the variable **qlimbs**).