Developing a Walking Controller for a Three-link 2D Biped

Jessica Lanini, Andrea Di Russo, Dimitar Stanev, Laura Paez,
Vivek Ramachandran, Mehmet Mutlu and Auke Ijspeert
{jessica.lanini - andrea.dirusso - dimitar.stanev - laura.paez - vivek.ramachandran - mehmet.mutlu}@epfl.ch

Start: 24/09/2019 - Deadline: 20/12/2019

Introduction

I am sure by now you have seen many legged robots. The question is, if I give you a legged robot how would you make it walk? Where do you start? Of course, there is an interface where you can receive sensory data and send commands, but will you start by sitting and sending commands and see what happens? No. So, what is the starting point? The starting point is to model, then design control and finally simulate. The whole point of the mini-project is to realize the following **Control Design Pipeline**:

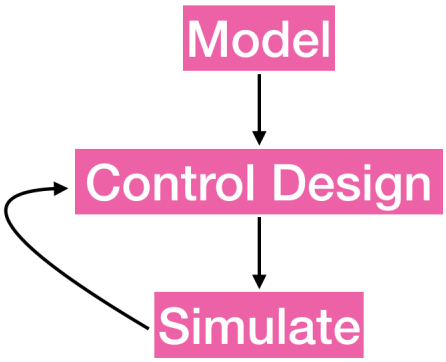

Figure 1: **Control Design Pipeline**

In order to understand the main principles behind the control design, we will work with a simple model: the **Three-link 2D Biped**, as represented in Fig. 2.

The project will be divided into three main parts:

- Modelling and visualization of the 3-link;
- Solving the equations of motion of the 3-link biped (simulation);
- Design of two different walking controllers, evaluate the resultant gaits and compare the performances.

You are asked to **develop these three parts and report your methods, results and observations in a final report**.

You are asked to structure your report according to the following sections which will be evaluated with the corresponding percentages:

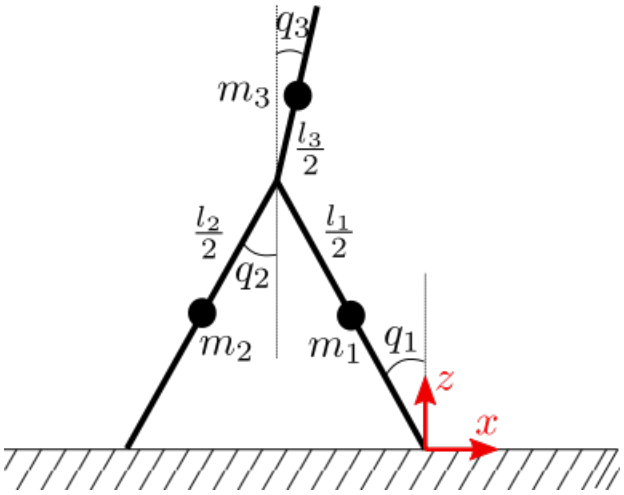

Figure 2: **Three Link Biped**

- Introduction : 3%
- Methods : 5%
- Results : 40% (kinematics and dynamics: 10%; controllers: 30%)
- Discussion : 50%
- Conclusion : 2%

In this lecture and in the next you will complete the third part: *Design of two different walking controllers, evaluate the resultant gaits and compare the performances.*

Preliminary Step

As first thing to do, you need to copy the files 'eqns.m','event_func.m' and 'solve_eqns.m' you developed last week into 'solve_eqns' folder, while the file 'animate.m' into 'visualize' folder.

Exercise 4.1: Implement two different walking controllers

Complete functions 'control.m' and 'control2.m' with two different walking controllers, considering the following:

- Each controller should able to generate a family of walking gaits (different velocities)
- Maximum available torque of each actuator is 30 Nm. Also, the torque signals have to be continuous (i.e., no spikes). Gaits which require torques more than this value are not accepted. It is OK if your torque signals have spikes at the beginning or end of each step; however, you should try to avoid any spikes in between.

Hints

- Since this robot does not have feet, clearly ZMP-based approaches would fail here. In contrast, trajectory based approaches or the methods discussed in the lecture might be helpful.

- Note that you have only two actuators and three degrees of freedom. Hence, the robot has one degree of underactuation. It is important to decide which variables or combination of variables you would like to fully control. Think about, which one of these seem more intuitive to directly control? Which one can be regarded as a "free variable"?
- It is recommended to use a function 'control_hyper_parameters.m' which includes the controller parameters (e.g., desired step length, frequency if applicable)
- In case you decide to continue with virtual constraints or trajectory control, you can consider using splines or Bezier polynomials for designing your virtual constraints (or trajectories).
- When designing your controller, be careful with the convention for the positive and negative signs

Exercise 4.2: Add perturbation

Add a perturbation torque to your control torque in 'eqns.m'. You can build the perturbation torque in an additional file from a predefined external force. **Make sure that the perturbation is applied when the model is in steady-state**

Exercise 4.3: Evaluation and Comparison

Evaluate each walking controller through the 'analyze.m' function, considering:

- Min-Max velocities reached
- Min-Max perturbation force tolerated
- plots of the angles vs time
- velocity of the robot vs time
- displacement in each step vs step number
- step frequency vs step number
- torques vs time
- cost of transport, defined as follows:

$$CoT = \frac{\int_0^T max(0, u_1 \cdot (\dot{q}_1 - \dot{q}_3))dt + \int_0^T max(0, u_2 \cdot (\dot{q}_2 - \dot{q}_3))dt}{x(T) - x(0)}$$

- plots of $\dot{q}$ vs q for all three angles

Compare the performances of the two controllers.