ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Legged Robots

Assignement 3: Three-link Biped walking Controller
Group no. 7

Authors:
Guilhem Azzano
Quentin Vingerhoets
Hédi Fendri

Supervisors:
Auke Jan Ijspeert
Hamed Razavi

December 18,2018

Contents

1 Introduction

After modelling and simulating our three link biped-robot in the first two assignments (developing the kinematics and dynamics model and building a simulator), we have to design a walking controller which enables the three-link biped to walk without falling and which respect a list of requirements. We had the possibility to choose between different requirements. The list of requirements achieved is :

1. The controller is able to generate a family of walking gaits (different velocities). We chose this range of velocities(0.7 , 0.9 , 1.1 , 1.3 and 1.5 m/s) with a maximum of 10% error.
2. The robot starts from zero velocity $\dot{q_0} = [0; 0; 0]$ and from zero pose : $q_0 = [0; 0; 0]$.
3. Maximum available torque of each actuator is 30 Nm. Also, the torque signals have to be continuous (i.e., no spikes) spikes less then 30 Nm at the moment of switching the legs are accepted. error
4. The robot controller can reject minimum external perturbations of 50N in his steady state.
5. The controller should be able to generate energy-efficient gaits.

The robot is a planar three link biped walker on a plane with lumped masses consisting of a torso and two equal length legs connected to the torso at the hip. The coordinates $q = (q1, q2, q3)^T$ are used to describe the configuration. A control moment $u = (u1, u2)^T$ is applied at the hip joint between each leg and the torso. Therefore the angle $q1$ is free to change.

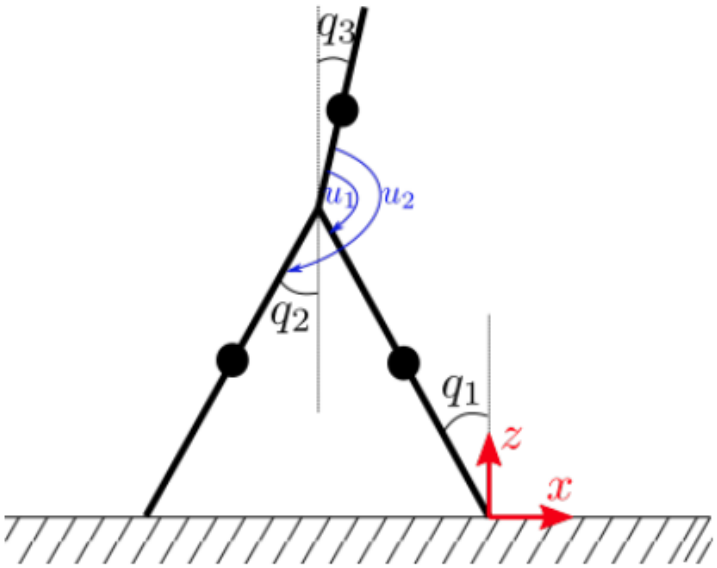

Figure 1: Three link biped robot

2 Methods

The robot is under actuated. He has only two motors for 3 DOFs. Those motors being attached to the torso with one for each leg, we are here working with relationships between q3 and the two other angles. The main idea we have behind locomotion, is that the robot will lean forward to induce a lose of balance in the right direction and fall on his swing foot before repeating the process for the next step. This means, that we use the weight of the torso to make the whole robot tilt frontward, and the swing leg to catch up the next step. Based on that, we decided to implement control on the torso (q3), and alternatively on one or the other leg, depending on which one is swinging (q2). Our free variable is therefore q1.

2.1 Torso controller (u1)

This controller is very important in terms of stability. It is the first one we decided to tackle down, as it is vital for the torso to be stable, before we design the control for the swing foot. We started by fixing a given angle to

it with a linear PD control, so that he will always be leaning forward to induce motion, while staying upright enough to do not fall. But we quickly realised that this angle had to be reduced once the robot was moving faster, because it would otherwise make him trip. At that point, we worked separately, and came up with two different solutions.
The first design was a non-linear PD controller (as seen in class, using virtual constraints) on a **fixed** angle $q3 - c = 0$ with $c \in [0, \frac{\pi}{2}]$. Added to it was a P control on the speed of the hip $dx_h - wantedSpeed = 0$. The aim of this second control is to increase the torque u1 when the robot is too slow, so that he leans more forward, and decrease it when he goes too fast, to make sure he never trips. The final u1 torque is then a combination of the output of those two controls. This design gave very good result, and allowed us to reach speeds of more than 1.5 m/s, but sadly, it was not stable enough to also manage the perturbation of 50N pushing the robot when in steady state.
The second design was a linear PD controller on the constraint: $q3 - q1 = 0$. A.k.a., the torso tries to stay in line with the stance leg. This design gave good results as well, as it naturally make the torso lean forward (see figure 2), but had struggles letting the robot run faster than 1.1 m/s without falling. However, it displayed very good stability results in terms of perturbations, often withstanding forces higher than 100N. This stability mainly comes from the derivative part of this control. When the stance leg trips fast forward, the torso will have an impulse in the opposite direction, balancing the robot and letting him recover.
Seeing that both design held good results while being complementary, we decided to mix the two ideas. We ended up with the following strategy: A non-linear controller on the virtual constraint $y = q3 - q1$, paired with a proportional controller on the speed of the robot. This allowed us to obtain great, controllable, steady-state speed, while keeping a satisfactory stability against perturbations.

Now to summarise this torso controller:
It is split in two parts. The first one is the non-linear virtual constraint $y = q3 - q1$. It will balance the torso and create the walking behaviour (see figure 2). One great feature of this control, is that the torso ends up at a smaller angle at the end of the step, than the one he has at the start of it. This allows it to absorb the shock of the impact when the swing foot hits the ground, without leaning forward enough to lose balance. That control makes the robot really stable, but nothing will stop it to move backward and the speed isn't controllable other than with the step length.

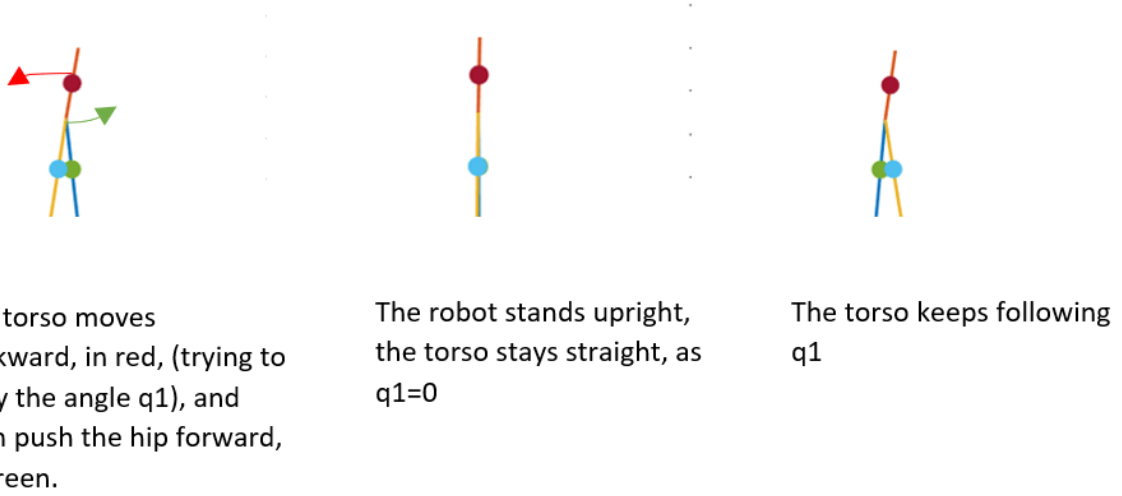

Figure 2: Behaviour of u1

The second part is the proportional on the difference between the speed of the hip and the wanted steady state speed. It allows to solve the problems of the first part. It leans the robot forward to accelerate. Once he has reach his travelling speed, it has almost no effect on it, and the first part takes care of keeping the travelling speed while remaining stable. Not only this control on the speed makes the robot move and accelerate well in the desired direction, it will also help the torso lean more backward if the robot runs too fast, which will help it stabilise.

2.2 Swing foot controller (u2)

Like we did for the torso, we started by simply using a linear PD control on the angle q2 to make sure the leg would go to the desired step angle. But as for the torso, this control was too simple, and the main problem with it is that the leg would hit the floor before crossing the stance leg. We changed the design and decided to make q2 follow some kind of function of q1. As the two legs have the exact same length, we need to make sure the swing leg crosses the stance leg when it is upright, then reach the ground when at the correct step angle (figure 3):

- The swing leg keeps a higher angle than the stance leg to stay above ground.
- The swing leg cross the middle point when the robot is upright, quickly going from left to right to avoid touching the ground out of the clearance space.
- The swing leg keeps an angle higher than the stance leg, until it reaches the desired step angle, where it will impact the ground, and take its turn as the stance leg.

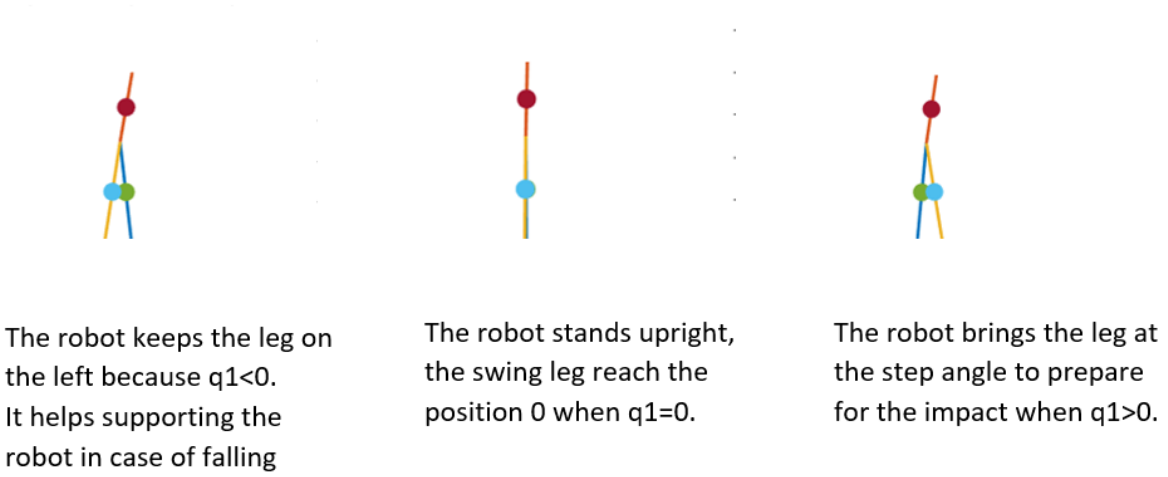

Figure 3: Behaviour of u2

The solution for this behaviour is to use a function of the shape of figure 4. The main characteristic of this kind of function, is that it has a high slope around the origin (fast crossing), and converges to an horizontal asymptote (step angle). We thought about three different functions: arctan, tanh and sigmoid. We started using arctan, but it performed badly when the robot started reaching high speed. The swing leg would sometimes drag on the floor, slowing down the robot, or making it trip. So, we changed it to tanh, which ended up working very well.

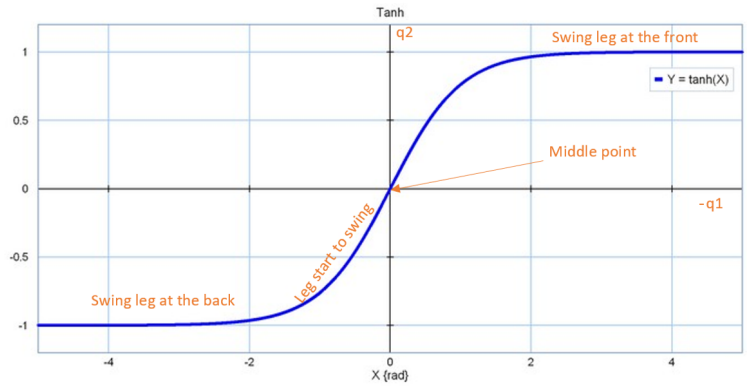

Figure 4: Tanh function behaviour

To make it work with our control, we scaled it to the wanted step angle (tanh being a normalized function) and we added a coefficient to its argument (-q1) to be able to tune the slope. The function is the following:

$q2 = f(q1) = Angle\_required * tanh(coefficient * (-q1))$. The coefficient is there to accelerate the process when crossing the middle point. It makes the slope steeper and therefore the leg will move forward faster when crossing. We implement that controller with a simple PD loop.

2.3 Hyper-parameters Fine tuning

During the process of designing our controller, we had the time to play with its various parameters, and to understand their impact on the control. The list of parameters is the following:

- Kp_1, Kd_1, PD coefficients for the non-linear control on the torso for $u1$.
- Kp_{1_2}, P coefficient for the linear control on the speed for $u1$.
- Kp_2, Kd_2, PD coefficients for the linear control on the swing leg for $u2$.
- $speed, step\_angle, tanh\_c$ Respectively the wanted speed for the speed control and the step angle and tanh coefficient for the swing leg control.

To summarise:

- Kp_1 Allows to more quickly get back to a straighter position for the torso, improving its stability, but can make it oscillate if too big.
- Kd_1 Allows to sooth the torso's motion, greatly improving stability, but can make the robot trip forward if too big, as the torso will be to slow.
- Kp_{1_2} Makes the robot converge faster towards *speed*. But if either of those two is too big, the robot will lean forward too much and fall.
- $tanh\_c$ and $step\_angle$ Those two are related. In general, we want a small $step\_angle$ to diminish the force of the impact on the torso. To make smaller steps, you also want the swing leg to move faster when crossing, which is achieved by increasing $tanh\_c$.
- Kp_2 Defines the reactivity of the swing leg. Needs to be big for fast motion and small steps to work properly. Making it too big will cause oscillations and unstability in the robot's gait.
- Kd_2 Sooth the motion of the swing leg. Needs to be very small to still let it move fast enough, but if too small, the leg will start having a jerky behavior and will accidentally hit the floor.

Knowing all this, we improved the parameters in an heuristic manner with a fail and retry process until we were satisfied with the results. Sadly, we could not understand well enough the mechanics of our controller to come up with a more analytic procedure for the tuning of our parameters. However, the tuning was not too hard and we are happy with the results.

3 Results

To obtain different speed, simply changing the "*speed*" parameter works but can induce instability. Therefore, the other parameters need to be tuned as well.. The parameters Kp2 and Kd2 of the swing foot controller are constant: $Kp2 = 1000$ and $Kd2 = 6$. For the rest, the parameters change slightly from one speed to the other. As you can see, the reached speed is not exactly equal to the *speed* parameter. Our controller could still be improved, in order to have parameters that work in any situation, and a speed that is defined solely by *speed* (see table 1).

Table 1

Required speed$[m/s]$	Obtained speed$[m/s]$	Error [%]	Max perturbation[N]	Controller parameters	cost of transport[J/m]
v=1.5	1.5	0%	60N	$Kp1 = 6500$, $Kd1 = 230$, $Kp1\_2 = 78$ $tanh\_c = 75$, $step\_angle = \frac{\pi}{23}$ $speed = 1.58$	1.5×10^5
v= 1.3	1.305	0.38%	73N	$Kp1 = 6800$, $Kd1 = 240$, $Kp1\_2 = 78$ $tanh\_c = 65$, $step\_angle = \frac{\pi}{23}$ $speed = 1.29$	1.37×10^5
v= 1.1	1.102	0.18%	83N	$Kp1 = 6300$, $Kd1 = 250$, $Kp1\_2 = 84$ $tanh\_c = 65$, $step\_angle = \frac{\pi}{22}$ $speed = 1.11$	1.1×10^5
v= 0.9	0.9046	0.51%	76N	$Kp1 = 6000$, $Kd1 = 250$, $Kp1\_2 = 83$ $tanh\_c = 65$, $step\_angle = \frac{\pi}{22}$ $speed = 1.03$	7×10^4
v= 0.7	0.7031	0.44%	83N	$Kp1 = 6000$, $Kd1 = 250$, $Kp1\_2 = 62$ $tanh\_c = 45$, $step\_angle = \frac{\pi}{22}$ $speed = 0.8$	4.32×10^4

Next, we present the various graphs showcasing the behaviour of our robot when running at its maximum speed of 1.5 m/s. You can see in figure 5 that he reaches his steady-state speed quite rapidly. He then withhold the perturbation with minimal changes in the speed, rapidly converging back to 1.5 m/s. As you can see, the instantaneous speed of the hip varies a lot. This is mainly due to the impact of the swing leg when it hits the floor that will slow it down at each step.

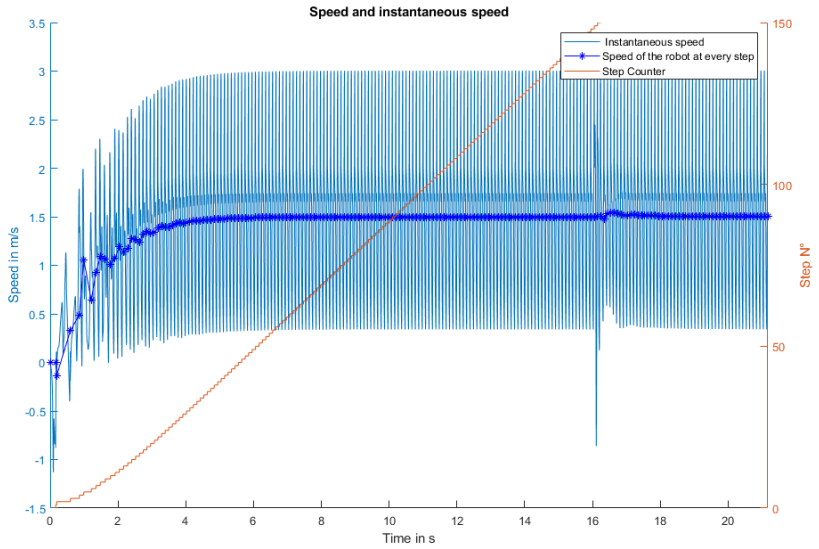

Figure 5: Average step speed (dark blue) and instantaneous speed (light blue) against distance and number of steps

The following figures (figure 6, 7) show both torques $u1$ and $u2$ in function of time. They showcase an antagonistically similar behaviour, with one big spike at each step to compensate the impact, and one smaller spike in-between when the swing leg is quickly passing to the front. That big spike is also due to the swapping of the legs. Indeed when the swing foot touch the ground we are swapping the variables to simplify our model. Our torso controller was leaning the torso forward before the impact but at the impact the stance leg become swing leg and q1's sign is inverted. Suddenly the torso has to move back as half of the controller try to follow the angle q1. Of course that behaviour is expected, we want to push the hip forward to create the walking cycle.

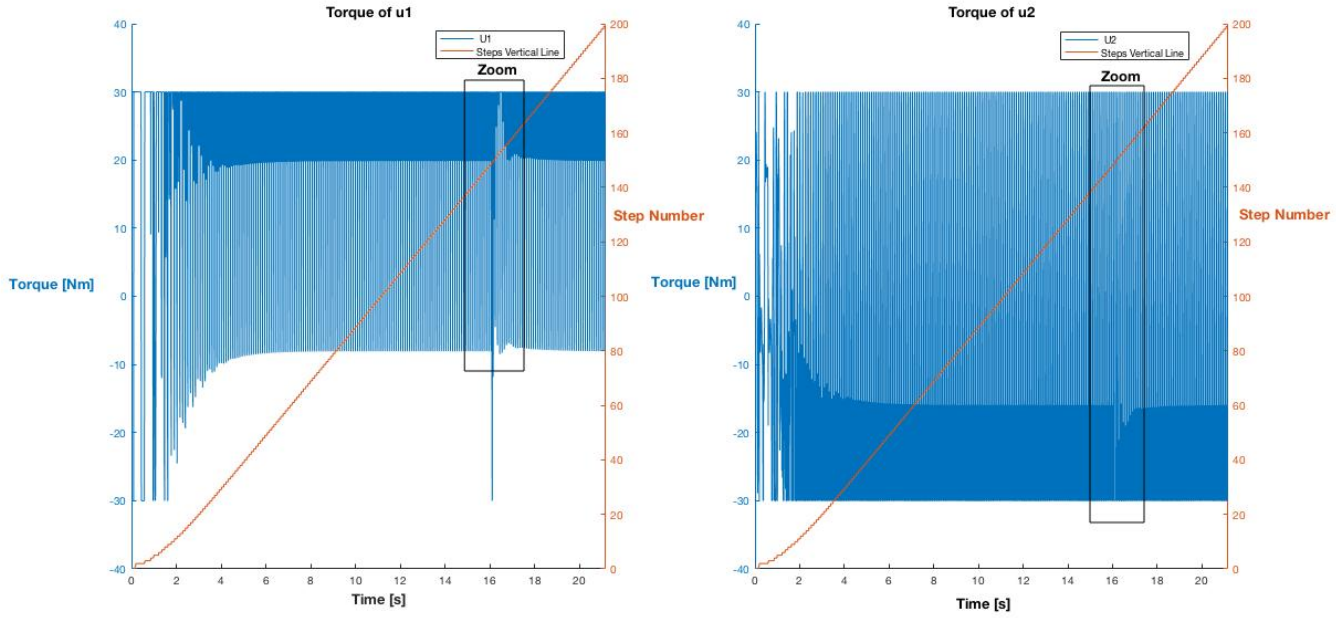

Figure 6: Torques u1 and u2 plots against time at maximum speed =1.5[m/s]

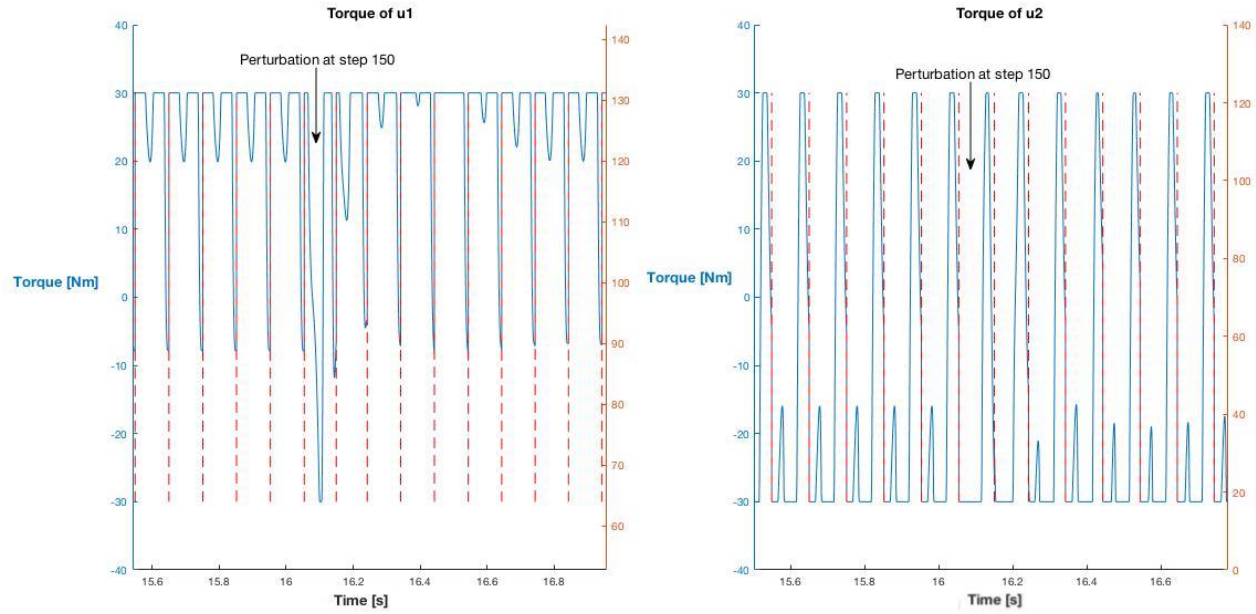

Figure 7: Torques u1 and u2 plots zoomed in the steady state during the perturbation at maximum speed =1.5 [m/s]

By looking at the torque during a normal step we can estimate how robust is our controller. In the two following figures (figure 8, 9) we can see the torque for each controller at the lowest and highest speed.

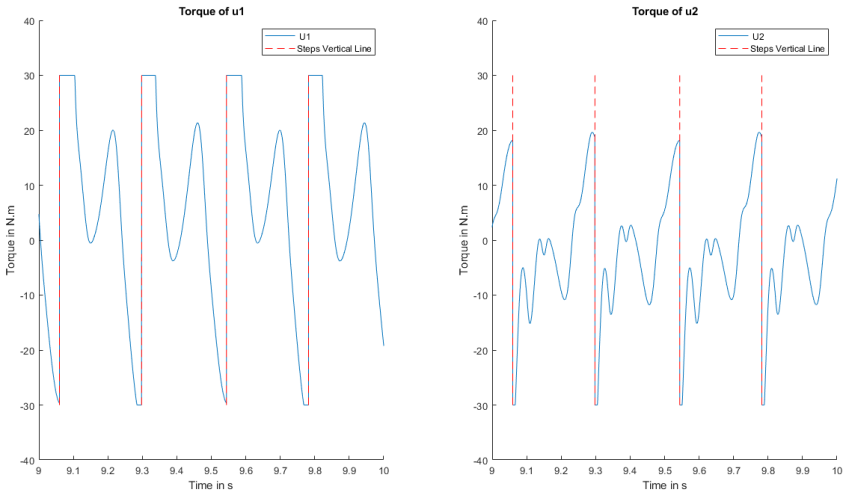

Figure 8: Torques u1 and u2 plots against time at minimum speed=0.7 [m/s]

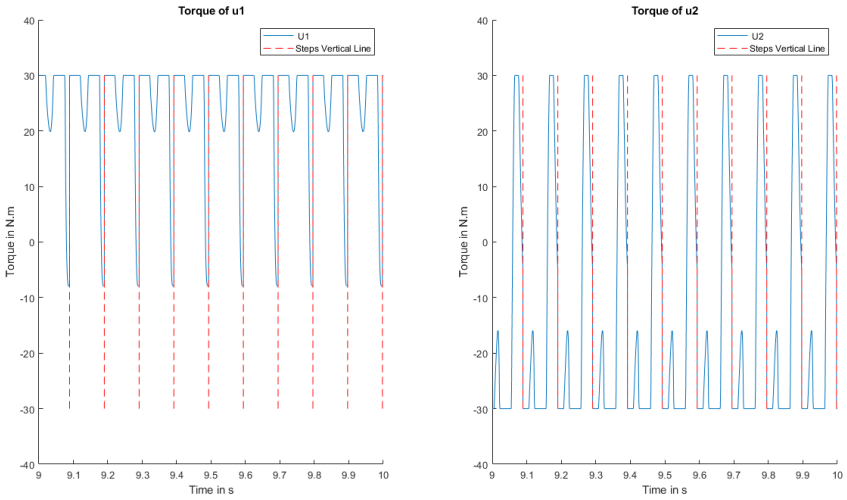

Figure 9: Torques u1 and u2 plots against time at maximum speed=1.5 [m/s]

At low speed we can see that the controller isn't saturating as much as for high speed. This explain why we have a higher stability against perturbation forces at 0.7 m/s than at 1.5m/s. Also as the controller is almost always saturating in figure 9 we can assume that we achieve one of the highest speed possible for our approach.

We also plotted the energy consumed and work produced by our robot to move win the figures 10 and 11 .We can see some spike at the beginning while the robot stabilise itself. Once it reaches its steady state, the required energy and the work produced by our actuators stay constant. We can also notice that the swing leg needs more work than the torso. At the step 150 the energy and work are fluctuating to compensate the perturbation.

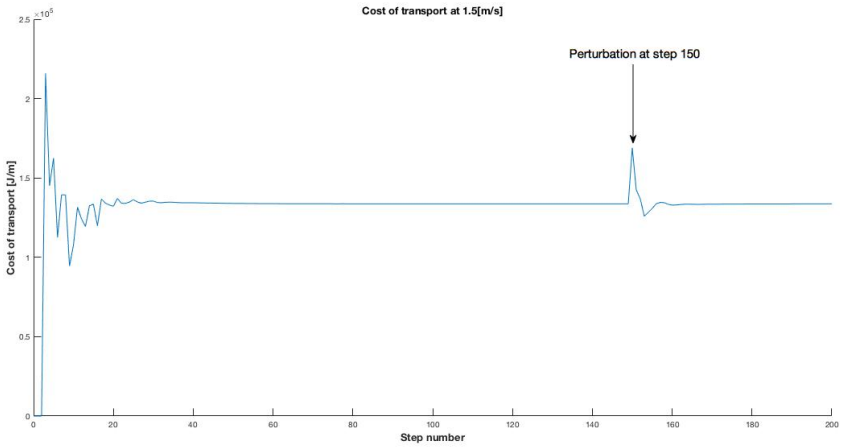

Figure 10: Cost of transport of the robot at minimum speed = 1.5 [m/s]

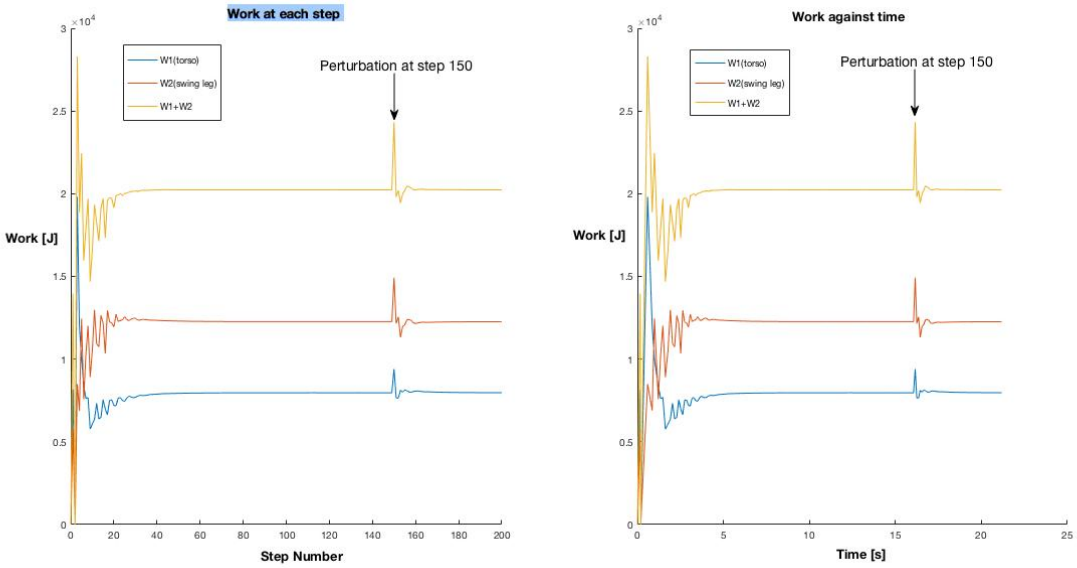

Figure 11: Work of each controller against step number (left) and distance (right) at maximum speed = 1.5 [m/s]

With the figure 12 we can compare the energy needed for the different velocities. We can see that slower is the robot less energy it will consume when it reach his steady state. So, for our controller the slowest gate is the most energy efficient. Indeed a slowest gate will have smoother movement and then the impact will be smaller and we will lose less energy. However we can see that during the first steps the slowest gates have a huge energy peak to reach their steady state. Thanks to the figure 11 we can see that the huge energy peak we saw previously is due to the balancing of the torso. This could probably be avoided by tuning the parameters and make our controller more low-speed oriented.

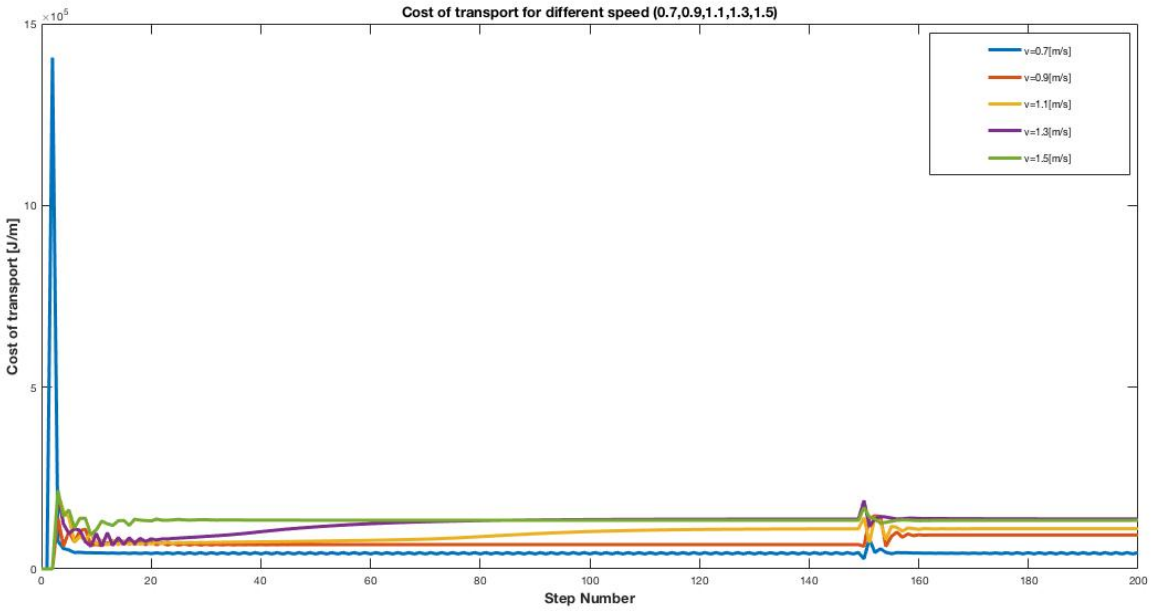

Figure 12: Cost of transport of the robot at different speed

We can analyse the stability of our system in the figures 13,14 and 11.In the figure 14 the three variables follow a cyclic path formed between the speed and the position. The lines which are not in the cycles shows the starting point and the perturbation.

This and the constant value of the displacement, works, energy and the pattern of the torque give us an heuristic proof of stability of our gate.

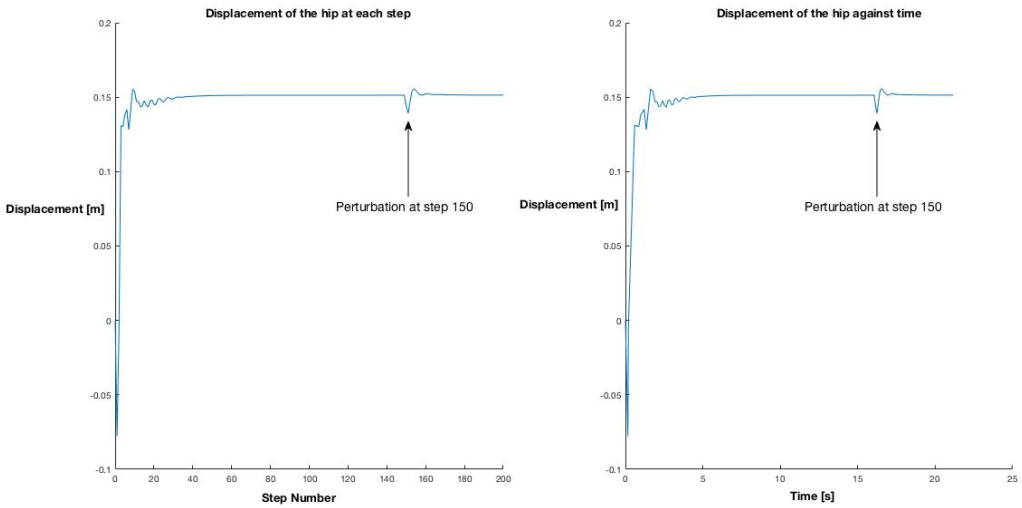

Figure 13: Displacement of the robot against the step number (left) and against the time [right]

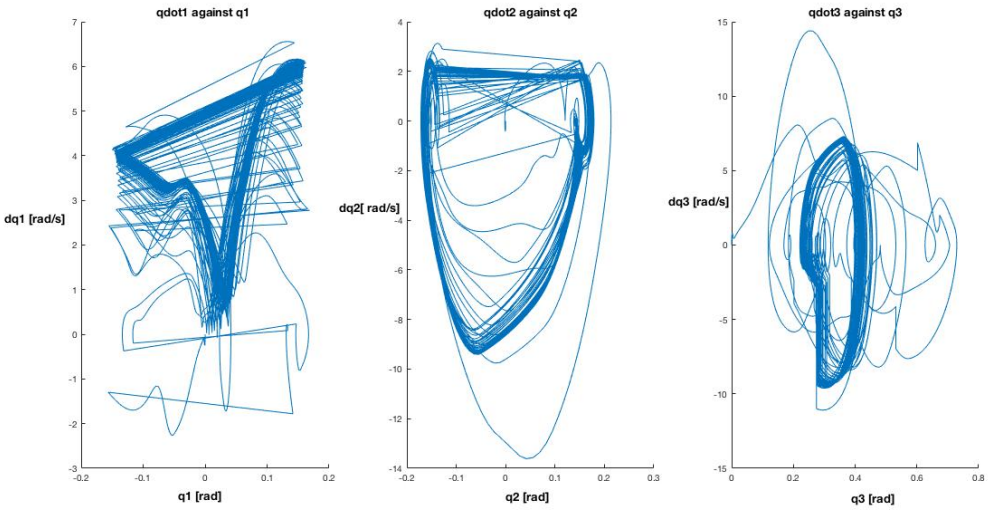

Figure 14: Limit Cycle

And finally, we show the relationship between the three different angles and the time in figure 15.

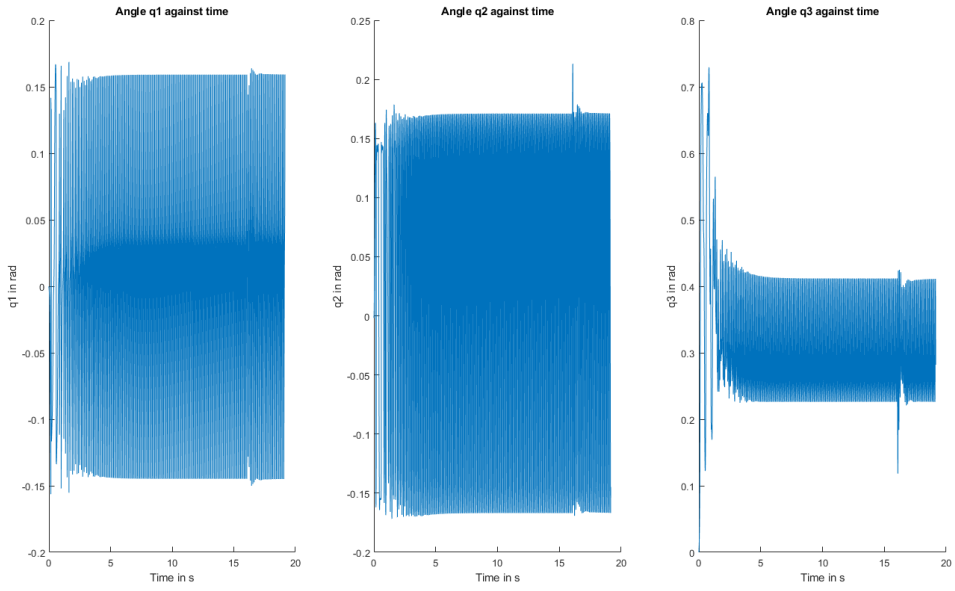

Figure 15: Angles in function of time

4 Discussion

We are satisfied of our results. We did not think we could go over 1.1 m/s, but getting knowledge about the mechanics of our controllers and trying different things to improve them finally brought up a working solution. It can however still be improved. The parameters can be more tuned in order to reach an optimal behaviour, and a stability criterion could be used to bring another perspective about the robots behaviour and characteristics. Also, the implementation of a flying phase could allow it to run at an even faster pace, but at the cost of a more complex control and model.