FDS 2024: Field and service robotics
Homework 4

Andrea Morghen

Question 1: Describe the buoyancy effect and why it is considered in underwater robotics while it is neglected in aerial robotics.

Buoyancy is the force in the upward direction exerted on an object immersed in a fluid (liquid or gas), it is caused by the difference in density between the object and its surroundings. By considering:

$\overline{g} = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T \in R^3$

$g \in R^3$, the gravity acceleration

$\Delta \in R^3$, the volume of the body

$m \in R^3$, the mass of the body

$\rho \in R^3$, the density of the water

We can define the equation of buoyancy as:

$$b = \rho \Delta ||\overline{g}||$$

We see that the buoyancy force is equal in modulus to the weight of the amount of fluid displaced by the body of the object ($\rho\Delta = m_{fluid}$). The buoyancy force acts at the center of buoyancy, $r_b^b \in R^3$, and it is equal to:

$$f_b^b = -R_b^T \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix}^T = -R_b^T \begin{bmatrix} 0 \\ 0 \\ \rho \Delta g \end{bmatrix}^T$$

The sign $-$ is due to the choice of the frame (z-axis downwards).
In the underwater robot model, the combined effect of gravity (f_g^b) and buoyancy(f_b^b) in the fixed body frame (ENU) is as follows:

$$g_{rb}^b = \begin{bmatrix} f_g^b + f_b^b \\ S(r_c^b) f_g^b + S(r_b^b) f_b^b \end{bmatrix}^T \in R^6$$

Buoyancy is a hydrostatic effect which determines the presence of this phenomenon even in a situation of stasis, so it is different from the effect of added mass and the contribution of inertia which are hydrodynamic, i.e. only present when the body is accelerating.

In underwater robots buoyancy cannot be neglected while in aerial robots it can, this is due to the different density of the two fluids in which the robots move. The density of water can be compared to the density of the underwater robot, whereas for aerial robots this is not the case, in fact the density of air cannot be compared to the density of the drone. Considering the buoyancy formula, it is observed that buoyancy is linearly dependent on ρ, so if the density of the fluid is sufficiently small, the buoyancy effect can be neglected.
Because the centre of buoyancy does not coincide with the centre of mass, a torque is generated that make the body rotates. In order to mitigate this torque there is a tendency to align the two central vertical axes.

Question 2: Briefly justify whether the following expressions are true or false.

a. The added mass effect considers an additional load to the structure.
False: A rigid body moving in a fluid also moves with it part of the fluid covering it, the fluid particles close to the body begin to accelerate with it due to the reaction principle and this acceleration is linked to a reaction force, called the added mass effect. The fluid exerts a reaction force which is equal in magnitude and opposite in direction, in the dynamic model this effect must be considered with the help of the added mass matrix MA (not necessarily positively defined), but the added mass is not a quantity of fluid to add to the system such that it has a bigger mass, in fact the sentence is false because this effect is not a hydrostatic effect, in static coditions this effect is not present.

b. The added mass effect is considered in underwater robotics since the density of the underwater robot is comparable to the density of the water.

True: the added mass effect depends on the density of the fluid, in fact it occurs when the fluid surrounding the drone undergoes an acceleration and the density of this fluid is comparable to the density of the robot. This effect cannot be neglected in water, whereas for aerial robots it can be neglected because the density of air cannot be comparable to that of the drone.

c. The damping effect helps in the stability analysis.

True: the viscosity of the fluid also causes the presence of dissipative drag and lift forces on the body. A common simplification is to consider only quadratic damping terms and group these terms into a damping matrix $D_{RB} \in R^{6x6}$ that is positive definite, the coefficients of this matrix are considered as constant. The viscous effect can be considered as the sum of two forces, the drag and the lift forces. In particular, the drag force acts in a direction opposite to the desired motion, so it can be very useful in a stability test when designing a controller for the robot. About Lyapunov stability analysis, the damping terms help make the derivative of the Lyapunov function more negative (accelerates the convergence of the system's state to the desired equilibrium state). This can be seen in the following formula from page 72 of the book Underwater Robots [1]:

$$\dot{V} = -s_v^T[K_D + D_{RB}]s_v$$

where:
$K_D =$ control gain matrix (positive definite)
$K_D =$ damping matrix (positive definite)

The term D_{RB} clearly contributes to making $\dot{V}$ negative definite, thus ensuring the stability of the system.

d. The Ocean current is usually considered as constant, and it is better to refer it with respect to the body frame.

False: because it is true that the ocean current is usually considered as constant but in the world frame. In fact, if it is refer with respect the body frame, then the ocean current cannot be considered constant. Since it represents a disturbance for the robot system and a constant disturbance is much easier to manage in the control design, the representation of current in the world frame where it is constant and irrotational is useful:

$$v_c = \begin{bmatrix} v_{c,x} \\ v_{c,y} \\ v_{c,z} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \dot{v}_c = 0$$

Question 3: Consider the Matlab files within the quadruped_simulation.zip file. Within this folder, the main file to run is MAIN.m. The code generates an animation and plots showing the robot's position, velocity, and z-component of the ground reaction forces. In this main file, there is a flag to allow video recording (flag_movie) that you can attach as an external reference or in the zip file you will submit. You must:

a. implement the quadratic function using the QP solver qpSWIFT, located within the folder (refer to the instructions starting from line 68 in the file MAIN.m);

b. modify parameters in the main file, such as the gait and desired velocity, or adjust some physical parameters in get_params.m, such as the friction coefficient and mass of the robot.

Execute the simulation and present the plots you find most interesting: you should analyze them to see how they change with different gaits and parameters and comment on them.

A whole body control scheme is an option for controlling a quadruped robot, which decouples motion planning from control. This controller sets the goal of achieving the desired acceleration $\ddot{q}_f^*$ and the desired ground reaction forces $\ddot{f}_{gr}^*$ that are obtained from a quadratic problem.

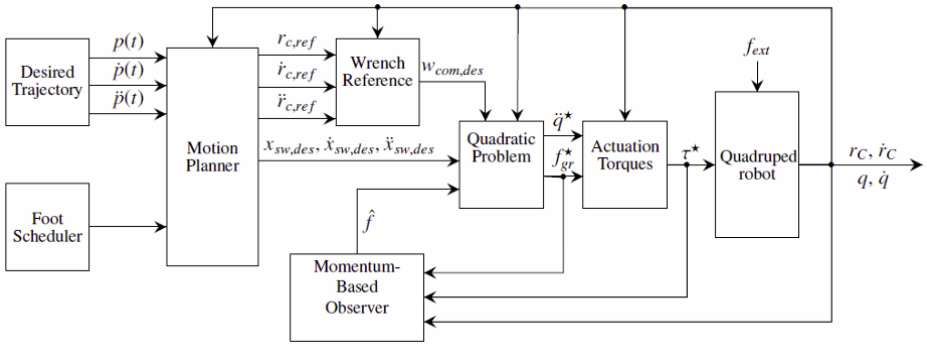

Figure 1: whole-body controller scheme

The quadratic problem's optimal function imposes to follow the desired wrench for CoM and a desired trajectory for the swing feet.

$$\begin{aligned} \underset{\zeta}{minimize} \quad & f(\zeta) \\ \text{s.t.} \quad & A\zeta = b, \\ & D\zeta \leq c. \end{aligned}$$

The constraints regard:

- Dynamic consistency
- Non-sliding contact
- Torque limit
- Swing the legs task

In the matlab file (row 77) the function "qpSWIFT" has been used, it takes in input the matrices representing constraints (equalities and inequalities) and the objective function to solve the constrained optimization problem.

One of the first parameters that it is posssible to change is the **GAIT**. There are 6 different gait that we can impose on our quadruped robot:

- **Trot gait**: cross legs move, allows fast locomotion but sacrifices stability for speed (support polygon is a line)
- **bound gait**: the front and the rear legs work in pairs, there is some time where there is no ground contact with any foot, so it is one of the most dynamic
- **pacing gait**: lateral legs work in coordination, less dynamic than the bound gait
- **gallop gait**: is the fastest gait, typical of animals like horses. Maximum speed for a quadruped robot but small margin of stability because at certain times it has only one foot in contact with the ground
- **trot-run**: is very similar to trotting, but the legs in phase of oscillation cover less space, from the simulation it is observable that there is no stance phase with all 4 legs but there is a very short moment when all legs are in swinging phase
- **crawl gait**: very stable because it always has 3 feet in contact with the ground (triangular support polygon) , but low speeds

Shown below are simulations of the six gait types with standard parameters, for videos of the full simulations see the folder in the github repository "recordings/no_change"[2]:

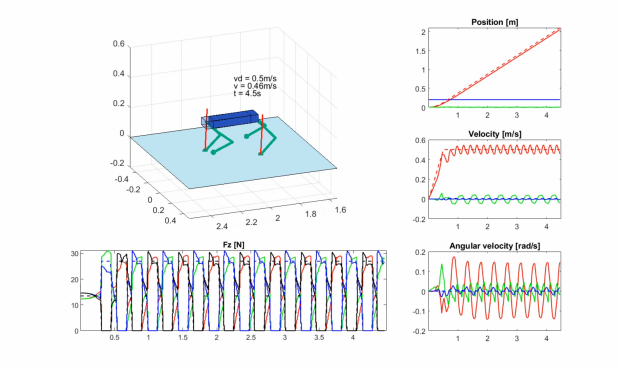

Figure 2: Trot simulation no change

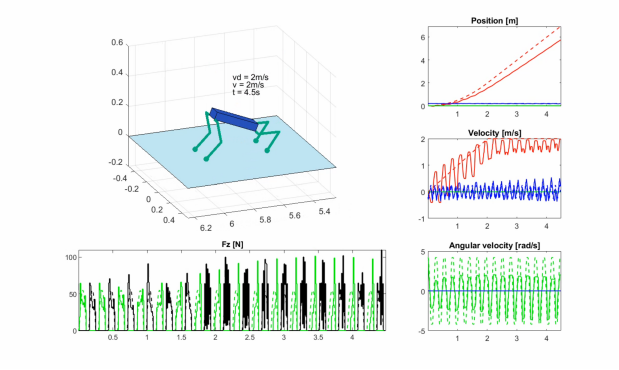

Figure 3: Bound simulation no change

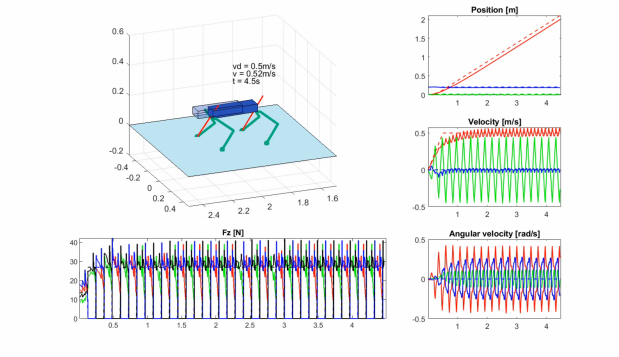

Figure 4: Pacing simulation no change

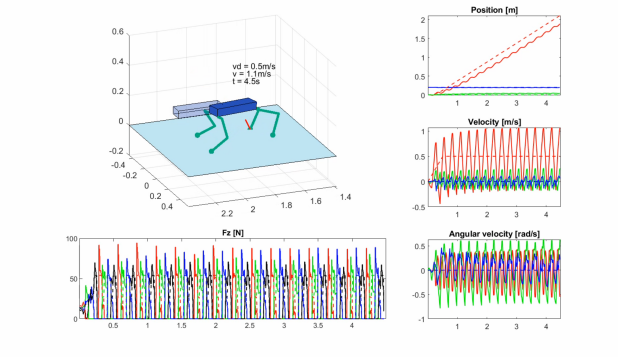

Figure 5: Gallop simulation no change

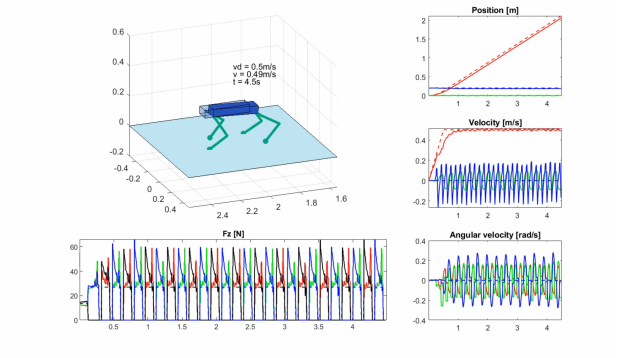

Figure 6: Trot-run simulation no change

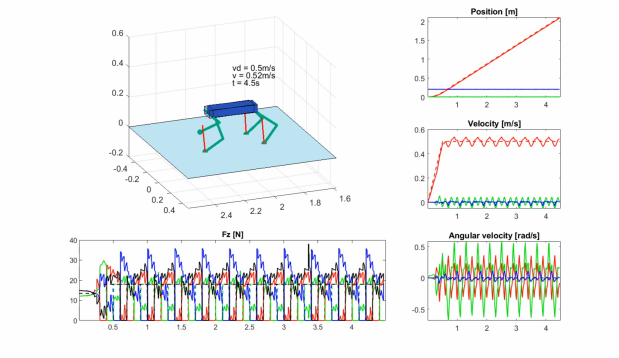

Figure 7: Crawl simulation no change

A second parameter that can be changed is the **DESIRED CRUISING SPEED**.
Various simulations were carried out with different increasing speeds, the results of these simulations are consistent with what was written earlier about the various gaits, however, past a certain speed threshold all gaits degenerate. Simulations at speeds of 2.0 m/s are shown where it is possible to observe some of these gaits degenerate. It is possible to observe that the bound gait and the gallop are not working really good at this high speeds and in fact degenerate into unrealistic gaits, this is due to the fact that these gait are not really robust (small stability margin). The crawl gait although it does not degenerate, loses its main characteristics and becomes much more similar to the gallop gait. Finally trot gait, trot-run gait and pacing gait simulations present marginal degeneration with this speed (larger stability margin) so they are viable although they have big oscillation on linear and angular velocity tracking.

Shown below are simulations of the six gait types with standard parameters besides velocity equal to 2.0 m/s, for videos of the full simulations see the folder in the github repository "recordings/vel_change" [2]:

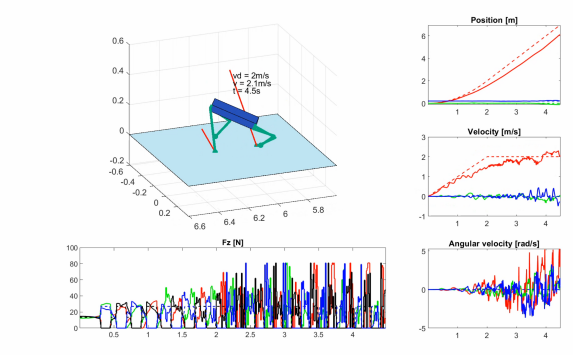

Figure 8: Trot simulation velocity 2.0 m/s

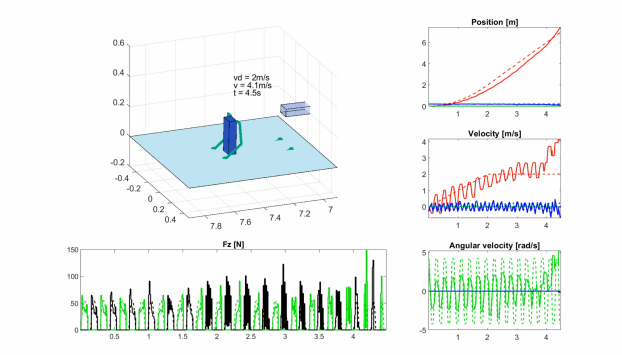

Figure 9: Bound simulation velocity 2.0 m/s

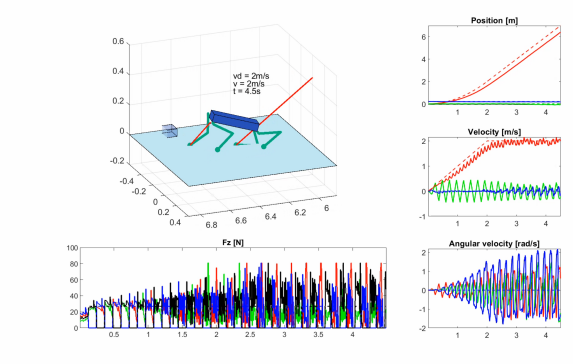

Figure 10: Pacing simulation velocity 2.0 m/s

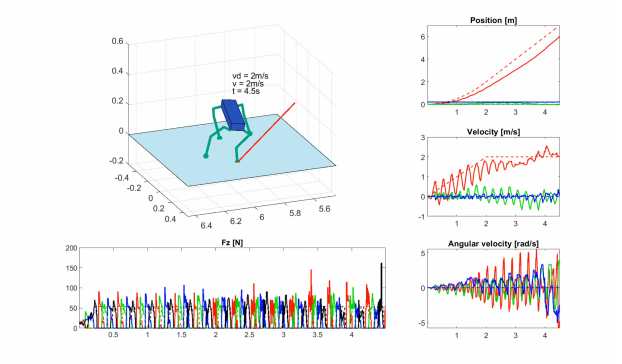

Figure 11: Gallop simulation velocity 2.0 m/s

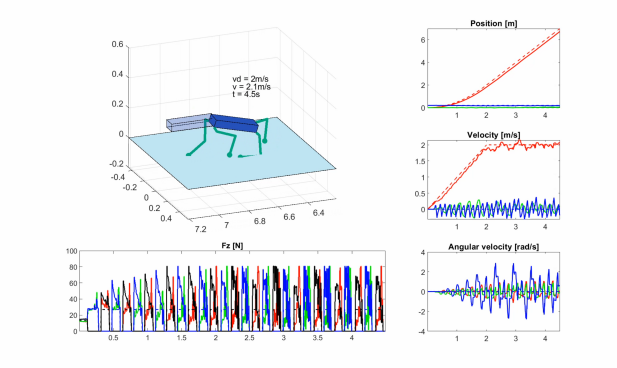

Figure 12: Trot-run simulation velocity 2.0 m/s

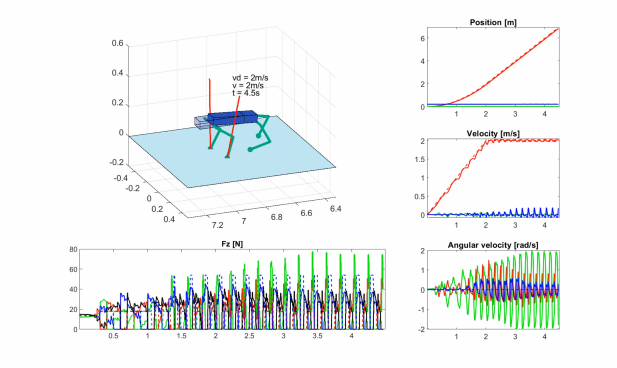

Figure 13: Crawl simulation velocity 2.0 m/s

As a third parameter, the **COEFFICIENT OF FRICTION** was considered.
This coefficient governs the contact between the robot's feet and the ground and depends on the material of both. Previous simulations present a value of 1, which is quite high. Nevertheless, some tests with a even higher value, $\mu = 2$, were done (difficult to realize in reality) showing that a higher value of μ leads to higher margins of stability of the robot's gait. Especially with high value of friction it is possible to maintain gait at higher speeds without degenerating. Later simulations were performed with a reduced friction coefficient of 0.5 (friction coefficient of concrete and rubber is 0.4 - 0.75 under wet conditions), decreasing μ we are increasing the possibility for the robot to become unbalanced. Another thing that we can note decreasing μ is that we have a delay in the tracking position.
The crawl gait works just perfect with reduced μ, it is in fact the most stable of the gaits, as so it has nearly no delay, the trot gait and the trot-run gait simulation work fine (fairly stable gait) with small delay, while the bound gait has unwanted rotations (angular velocity oscillations) and a delay in linear velocity tracking. Finally the pacing and gallop gaits are not stable and so brings the robot to fall down.

Shown below are simulations of the six gait types with standard parameters besides μ equal to 0.5, for videos of the full simulations see the folder in the github repository "recordings/f_change" [2]:

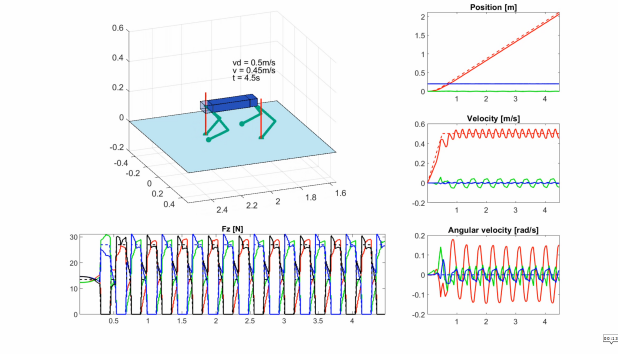

Figure 14: Trot simulation $\mu = 0,5$

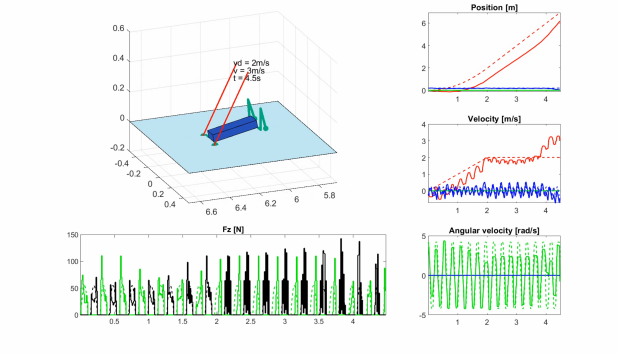

Figure 15: Bound simulation $\mu = 0,5$

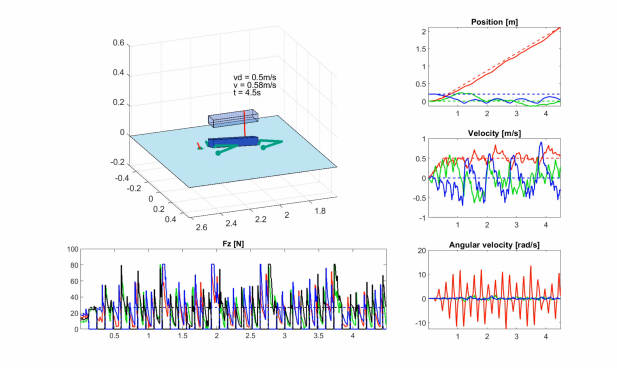

Figure 16: Pacing simulation $\mu = 0,5$

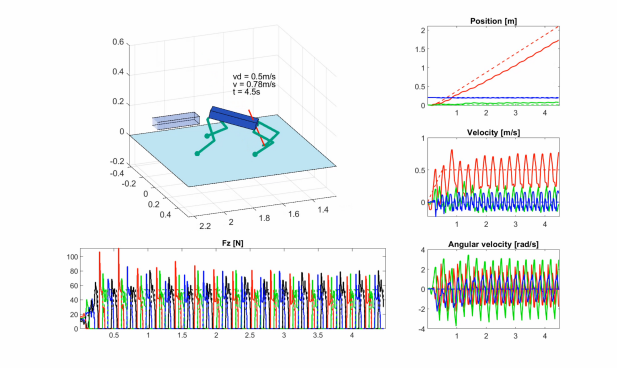

Figure 17: Gallop simulation $\mu = 0,5$

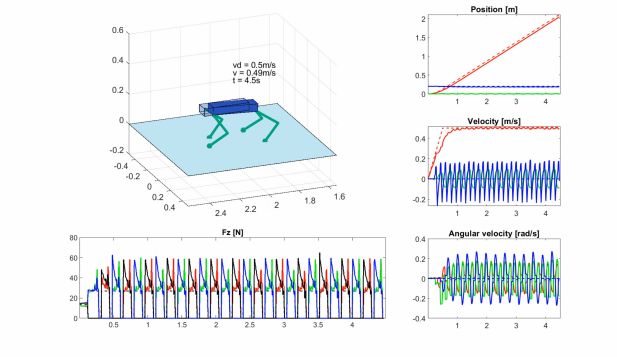

Figure 18: Trot-run simulation $\mu = 0,5$

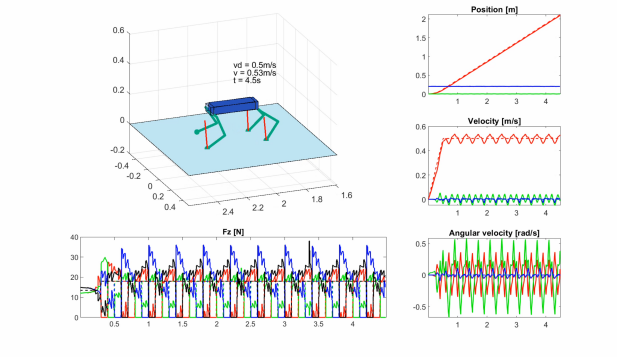

Figure 19: Crawl simulation $\mu = 0,5$

The last parameter modified was the **MASS**, decreasing the value of the mass of the robot enhance the tracking of the position and of course decreases the values of F_z. On the other hand the increase of the mass does not generate problems to the gait stable as the crawl and the trot, in fact they maintain more than one point of contact with the ground, while the more dynamic gaits, which have some instants with single or none contact with the ground, have a significant worsening in performance, to the point of not performing the gait set no more.

The simulations of the six gait types with standard parameters and mass equal to 10 kg and 1.5 kg are in the folder in the github repository "recordings/mass_change" [2].

Question 4: Consider a legged robot as in the picture below. The foot and the leg are assumed to be massless. The point T represents the toe, the point H represents the heel, and the point P is the ankle. The value of the angle θ is positive counterclockwise and it is zero when aligned to the flat floor. Answer to the following questions by providing a brief explanation for your them.

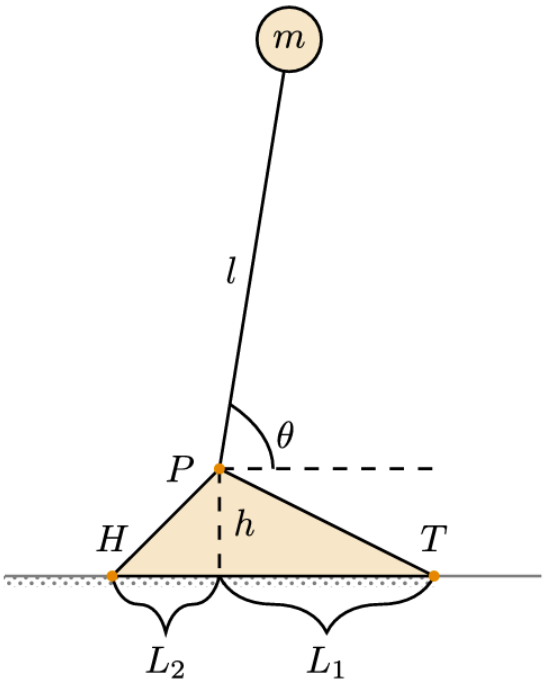

Figure 20: scheme of the legged robot

a. Without an actuator at the point P is the system stable at $\theta = \frac{\pi}{2} + \epsilon$?

This system can be seen as an inverse pendulum where the equilibrium point $\theta = \frac{\pi}{2}$ is unstable in the case of free evolution. Without the actuator in point P, in fact, there is no way to counteract the torque created by the force of gravity. With a small angular shift $+\epsilon$ from the equilibrium point the system move away indefinitely, so the system in the $\theta = \frac{\pi}{2} + \epsilon$ configuration is not stable.

b. Without an actuator at the point P (i.e.,$\ddot{\theta} \neq 0, \dot{\theta} \neq 0$), compute the zero-moment point on the ground as a function of θ and the geometric and constant parameters (if necessary).

From the slides [3]:" It is possible to recognize that the definition of cop implies that the horizontal moments of the ground reaction forces relative to cop are zero. The cop is also referred to as the zero moment point (ZMP)":

$$p_z^{x,y} = p_c^{x,y} - \frac{p_c^z}{\ddot{p}_c^z - g_0^z}(\ddot{p}_c^{x,y} - g_0^z) + \frac{1}{m(\ddot{p}_c^z - g_0^z)} S\dot{L}^{x,y}$$

with $S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \in SO(2)$

For the legged robot system in Figure ZMP is defined only with an x coordinate because the Support Polygon is a segment. So it applies:

$$p_z^x = p_c^x - \frac{p_c^z}{\ddot{p}_c^z - g_0^z}(\ddot{p}_c^x - g_0^z) + \frac{1}{m(\ddot{p}_c^z - g_0^z)}\dot{L}^y$$

Where L is the angular momentum and p_c are the coordinate of the centre of mass, in the case treated the only mass present is the mass m (The foot and the leg are assumed to be massless), so the CoM is the position in the plane $x - z$ of the mass m. So it applies:

$$p_c^x = l\cos(\theta) \implies \ddot{p}_c^x = -l\sin(\theta)\ddot{\theta} - l\cos(\theta)\dot{\theta}^2$$

$$p_c^z = h + l\sin(\theta) \implies \ddot{p}_c^z = l\cos(\theta)\ddot{\theta} - l\sin(\theta)\dot{\theta}^2$$

$$L^y = I\dot{\theta} \implies \dot{L}^y = I\ddot{\theta}$$

$$g_0^x = 0, g_0^z = -g$$

The ZMP:

$$p_z^x = l\cos(\theta) - \frac{h + l\sin(\theta)}{l\cos(\theta)\ddot{\theta} - l\sin(\theta)\dot{\theta}^2 - g}\left(-l\sin(\theta)\ddot{\theta} - l\cos(\theta)\dot{\theta}^2\right) + \frac{I\ddot{\theta}}{l\cos(\theta)\ddot{\theta} - l\sin(\theta)\dot{\theta}^2 - g}$$

c. Supposing to have an actuator at the ankle capable of perfectly cancelling the torque around P due to the gravity (i.e.,$\ddot{\theta} = 0, \dot{\theta} = 0$), what value of θ can you achieve without falling?

In the static case ($\dot{\theta} = 0$ and $\ddot{\theta} = 0$) the robot falls when the projection of CoM on the ground is external to the polygon support. In the treated case (projection on a 1D space), the projection of the CoM is worth $p_c^x = l\cos(\theta)$ while the supporting polygon in this is the $\overline{HT}$ segment. Then the range of values of θ that the robot can reach without falling are regulated by inequalities:

$$-L_2 \leq l\cos(\theta) \leq L_1 \implies \arccos\left(\frac{L_1}{l}\right) \leq \theta \leq \arccos\left(\frac{-L_2}{l}\right)$$

References

[1] G. Antonelli. Underwater Robots. 2003.

[2] Andrea Morghen. FSR2024. 2024. URL: `https://github.com/Andremorgh/FSR2024.git`.

[3] PRISMA UNINA. FSR_Slides_2024. 2024.