Field and Service Robotics Homework 4

Davide Busco P38000177

https://github.com/DavideBusco/-FSR-_Homework4

1 Exercise 1

It is requested to explain the buoyancy effect and why it is considered in underwater applications but not in aerial ones.
In underwater applications, when a body is submerged in a fluid under the effect of gravity, it is necessary to consider two additional forces in the dynamic model:

- Gravitational force,
- Buoyancy.

The buoyancy effect is an hydro-static effect that acts on underwater bodies.
To be an hydro-static effect means that it does not depend on the relative movement of the robot's body and the fluid.
This effect can be modelled as: $b = \rho\Delta||\overline{g}||$
where:

- $\overline{g} = \begin{bmatrix}0 & 0 & g\end{bmatrix}^T$ is the gravity acceleration vector,
- Δ is the volume of the body,
- ρ is the density of the water.

The buoyancy force acts on the center of buoyancy, which usually does not correspond with the center of mass, and its effect is opposed to the gravitational force. This means that, instead of being a force that pulls the body down, it actually pushes the body upwards.
The force can be expressed like:

$$f_b^b = -R_b^T \begin{bmatrix}0 & 0 & \rho\Delta g\end{bmatrix}^T \tag{1}$$

which, as it is possible to see, is opposed to the gravity due to the minus, as previously said.
It is also important to say that it is better to have the center of mass and the center of buoyancy aligned in order to avoid rotational torques.
This effect, as well as the one caused by the gravity acceleration, acts as a wrench in the body fixed frame.

Now that it is clear how this effect acts in underwater applications, it is also simple to explain why it is neglected in aerial robotics. The buoyancy effect can be neglected in aerial robotics because the densities of the robot and of the air are not comparable. Consequently, the contribution of buoyancy is not as appreciable as it is in underwater applications making it possible to neglect it in aerial applications.

2 Exercise 2

- FALSE: It is false because the added mass is not a quantity of fluid to add to the system in order to make the system mass result as a bigger mass. Instead, it has to be considered as an effect due to a reaction force of the fluid accelerated by the robot which is moving in the fluid. Consequently, the added mass contribution is given by the reaction force. In order to take it into account in the dynamic model of an underwater robot it is possible to add this effect with a term named "Restoring Forces". This term is the contribution by the gravity and buoyancy which gives a non-null force contribution when the UUV is still (as well as the current term).

- TRUE: In underwater robot applications when a robot accelerates and moves, it accelerates also the water (fluid) surroundings it. Since the densities of the robot and of the water (fluid) are comparable, then, as a consequence of the comparable densities, there is a reaction force opposed to the robot movement which cannot be neglected. Instead, the contribution of this effect is neglected in aerial robotics because the densities of the air and of the robot are not comparable since the density of the air is much lower than the one of the robot.

- TRUE: The damping effect helps in the stability analysis of the robot. This is true because it is an effect caused by the viscosity of the fluid, which causes the presence of dissipative forces related to drag and lift forces on the robot. The drag force is parallel to the relative velocity between the body and the fluid, whereas the lift force acts perpendicular to the drag force.
  These forces are typically considered to act at the body's center of mass. This consideration is beneficial for stability proofs using the direct Lyapunov method, as the forces contribute a negative term in the derivative of the Lyapunov function.

- FALSE: Ocean currents are phenomena that are very difficult to compute since they depend on numerous parameters such as tidal movements, atmospheric winds and other events which makes it difficult to compute the ocean current. It is true that it is often simplified by assuming that the ocean currents are constant and irrotational, what is wrong in the sentence is that they are not considered with respect to the body frame. Instead, they must be considered with respect to the world frame.
  With this assumption it is possible to express the ocean current as:

$$\mathbf{v}_c = \begin{bmatrix} v_{c,x} \\ v_{c,y} \\ v_{c,z} \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^6, \quad \dot{\mathbf{v}}_c = \mathbf{0}_6. \tag{2}$$

3 Exercise 3

In order to implement the quadratic function used for the minimization using the solver qpSWIFT, it is possible to write this line of code where the used matrices are given:

$$\Big[zval, basic_info eq, adv_info eq\Big] = qpSWIFT(sparse(H), g, sparse(Aeq), beq, sparse(Aineq), bineq) \tag{3}$$

With the purpose of examine the behaviour of the robot in simulation, it has been chosen to study the simulation with the default values reporting all the resulting plots as reference. Then the study has continued by changing one parameter at a time, with the other chosen with default values.
Considering as default values:
mass = 5.5Kg, desired velocity = 0.5m/s and friction coefficient $\mu = 1$.
This has be done for all the gaits, which are respectively trot, bound, pacing, gallop, run trot, crawl.

3.1 Gait 0 - Trot

In that case, using the default parameters, there is a little error on the position p_x tracking and the actual velocity v_x presents an oscillation around the desired value. The GRFs' peak value is 30N.

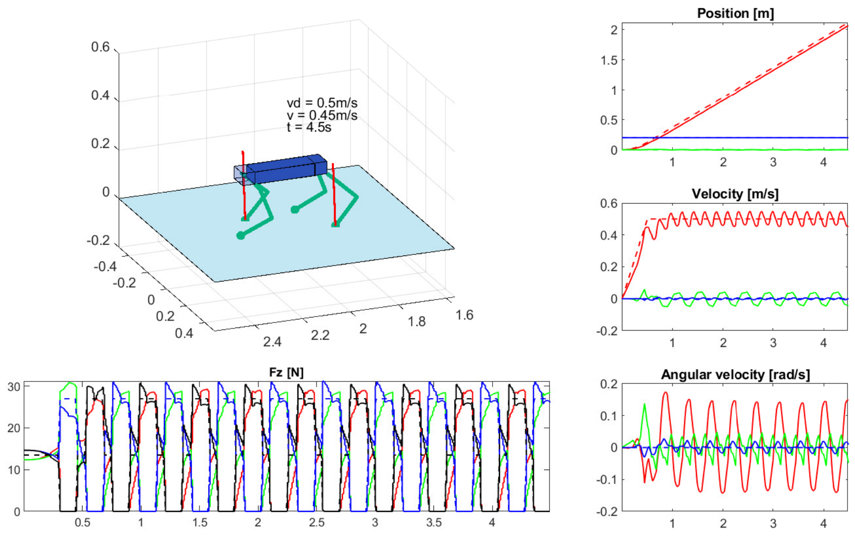

Figure 1: Simulation with default values.

Changing one parameter at a time, with the other parameters set at default values, the results are:

- Changing the mass:
  - with $mass = 11Kg$ the v_x oscillation around the reference value increases, and the position tracking is a bit worse than when the $mass$ is 5.5Kg, the peak for the $GRFs$ is 60N.

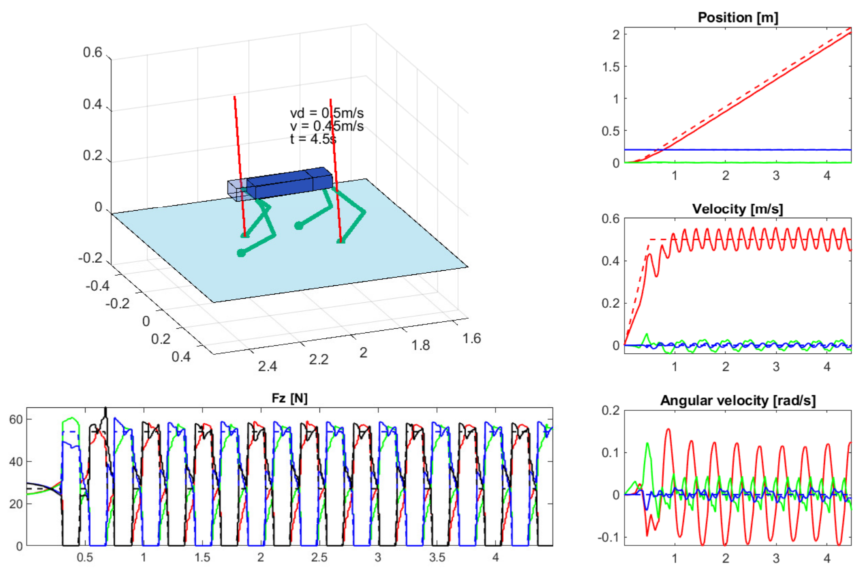

Figure 2: Simulation with mass = 11Kg.

- Setting the *mass* equal to 1.5Kg the robot does a better position tracking of the reference, and the v_x oscillation is lower. Then the peak for the *GRFs* decreases too, it is 8.5N.

- Changing the friction:
  - with $\mu = 0.2$ the robot performs a good trajectory tracking despite the low value for friction. Compared to the default case, there is a slight worsening in position tracking. This is because in that case, the reaction force vectors have a lower horizontal component compared to the vertical one. This may cause the motion to be less influenced by the friction.

- Changing Desired Velocity:
  - Setting the desired value for v_x equal to 0.2m/s, the robot advances slower, and this has as a consequence that it is able to do a better tracking of the position reference compared to the case with default parameters.

The last simulation is done with $mass = 1.5Kg$, $v = 0.2m/s$ and $\mu = 0.2$. As it is possible to see from the Figure 3, both the position tracking and the heading velocity oscillation are improved. Then, since the mass is lower, the GRFs are lower too. As said before, the small value for the friction doesn't affect much in that case.

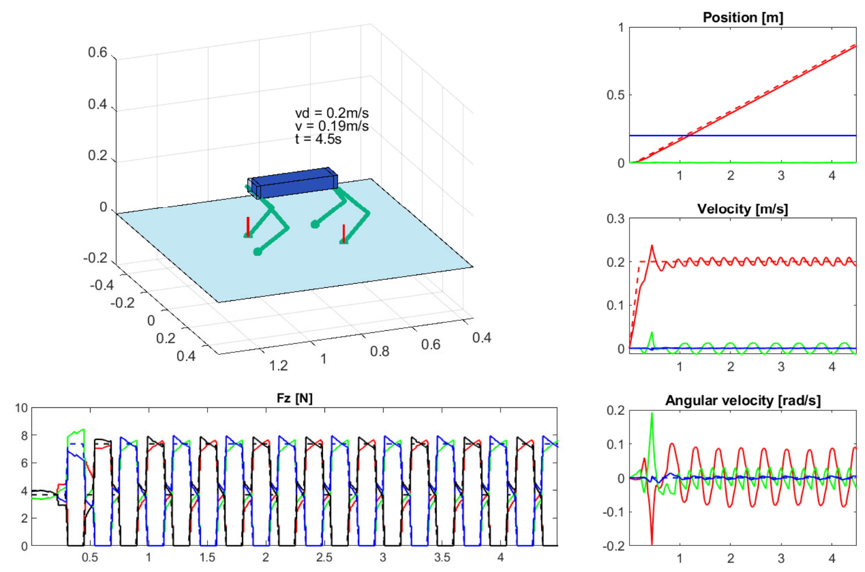

Figure 3: Simulation with $mass = 1.5Kg$, $v = 0.2m/s$ and $\mu = 0.2$.

3.2 Gait 1 - Bound

In that case, using the default parameters, the tracking error on the pos_x diverges and it cannot reach the desired value on v_x. Then, in that case, the desired velocity value is stuck to 2m/s, so changing it doesn't produce any effects. The $GRFs'$ peak value is 100N.

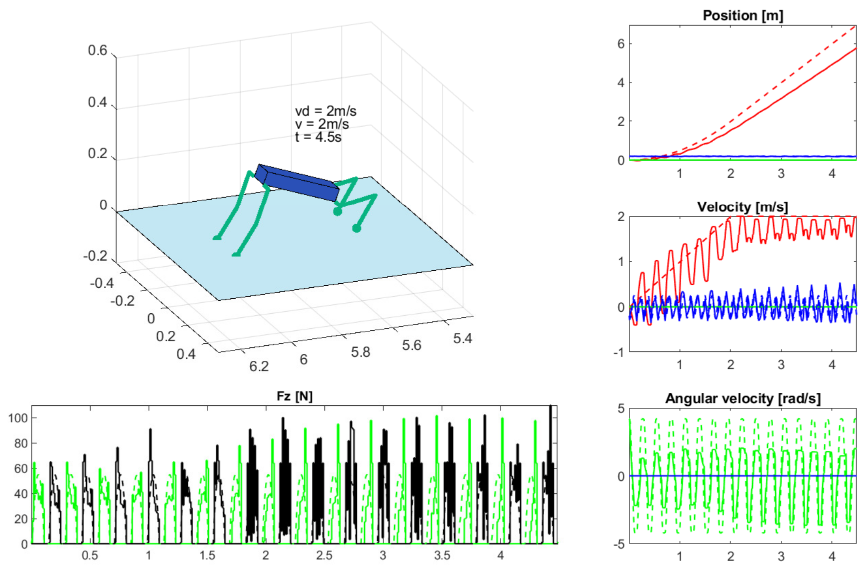

Figure 4: Simulation with default values.

Changing one parameter at a time, with the other set at default values, the results are:

- Changing the mass:
  - setting the $mass = 11Kg$ the robot crashes repeatedly, not following the references. The $GRFs$' peak value is much higher and it is 324N.
  - Setting the $mass = 2.7Kg$ the main difference is that the GRFs are smaller then the default case, and the

v_x has a smaller oscillation. The $GRFs'$ peak value is lower and it is 50N.

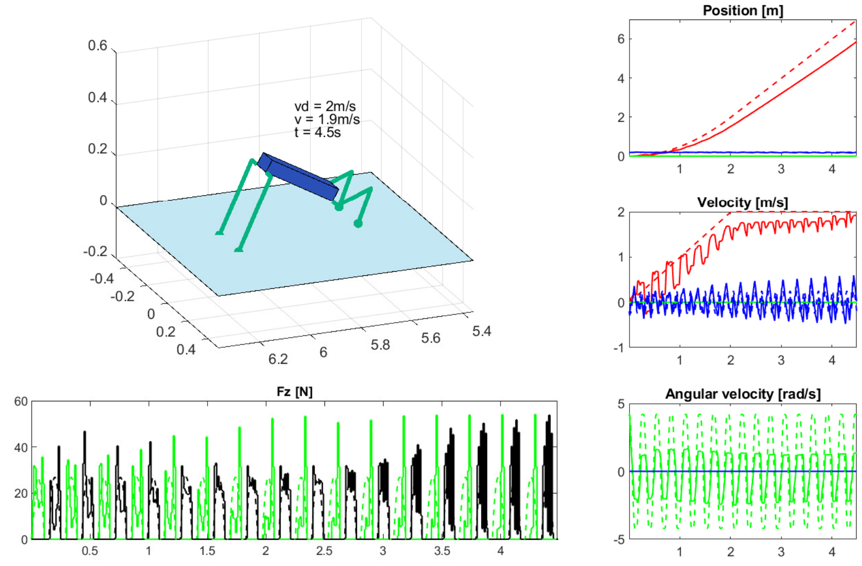

Figure 5: Simulation with mass = 2.7Kg.

- Changing the friction:

  - Setting $\mu = 0.2$, the robot, in the first seconds goes backward. Then, when its velocity is near to the desired one (2m/s), it goes faster and faster crashing on the ground.

  As it is possible to see in the scopes, in the beginning of the simulation, the error and the pos_x is negative since the robot is moving backward. Once the v_x reaches the desired value, it continues increasing causing the fall of the robot.

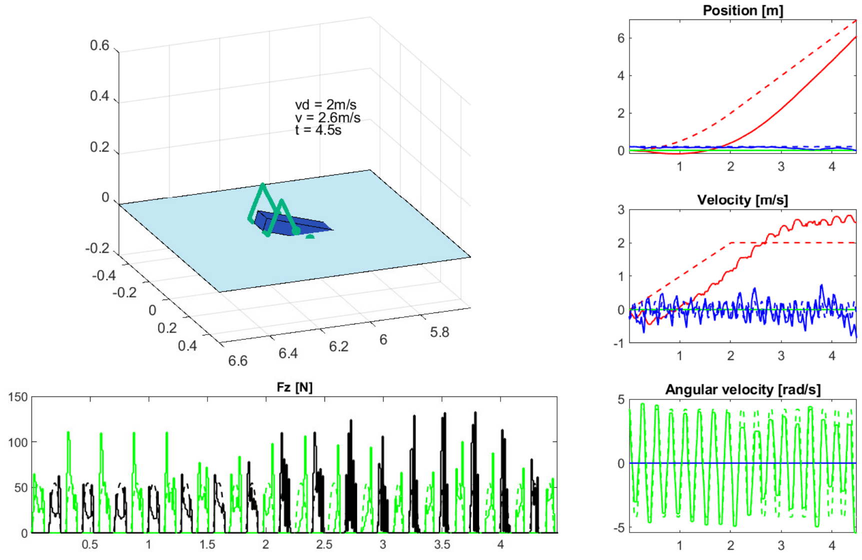

Figure 6: Simulation with v and m as default values and $\mu = 0.2$.

3.3 Gait 2 - Pacing

In that case has been noticed that using the default parameters, there is an increasing error on pos_x and happens that during the motion it rotates itself increasing its orientation error with respect to the heading motion.

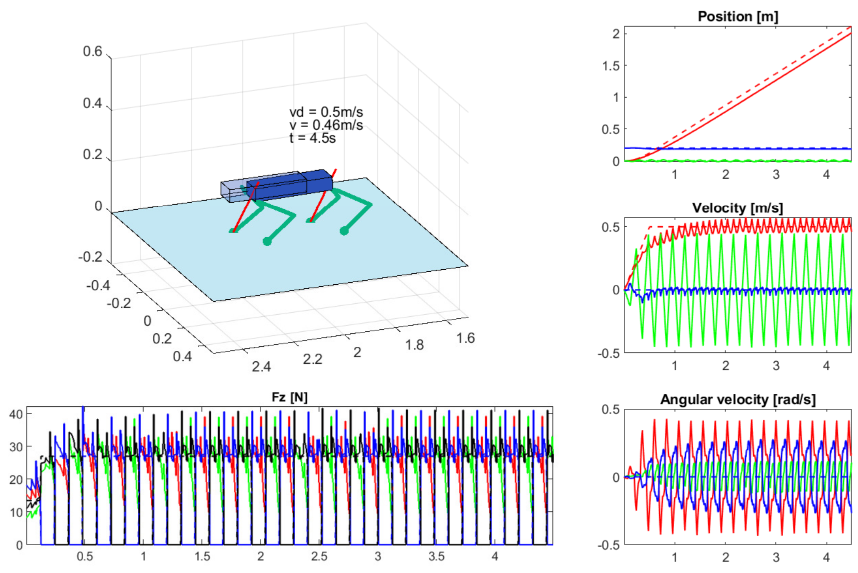

Figure 7: Simulation with default values.

Changing one parameter at a time, with the other set at default value, the results are:

- Changing the mass:
  - $mass = 11Kg$ the v_x has a greater oscillation around the reference value and the position tracking is a bit worse than the default case. Moreover after some seconds the robot collapse due to the weight.
  - $mass$ equal to 1.5Kg the robot's does a significantly better position tracking of the reference and the velocity oscillation around the reference value is lower than the default case. The GRFs' peak value is 9N.
- Changing the friction:
  - with $\mu = 0.2$ the robot collapse on the ground. This is because in that case the movement is obtained with the GRFs in which the horizontal component and the vertical one are comparable.
- Changing Desired Velocity:
  - Setting the desired velocity equal to 0.2m/s, the robot advances slower, but the position tracking error is improved.

In figure is shown a simulation with $v_x = 0$, $\mu = 0.5$ and $mass = 1$. The improvement achieved with these values are a very little position error and a very little oscillation of the v_x around the desired value. In that case $\mu = 0.5$ is enough to complete the simulation also because of a lower mass, compared to the previously done simulation with $\mu = 0.2$.

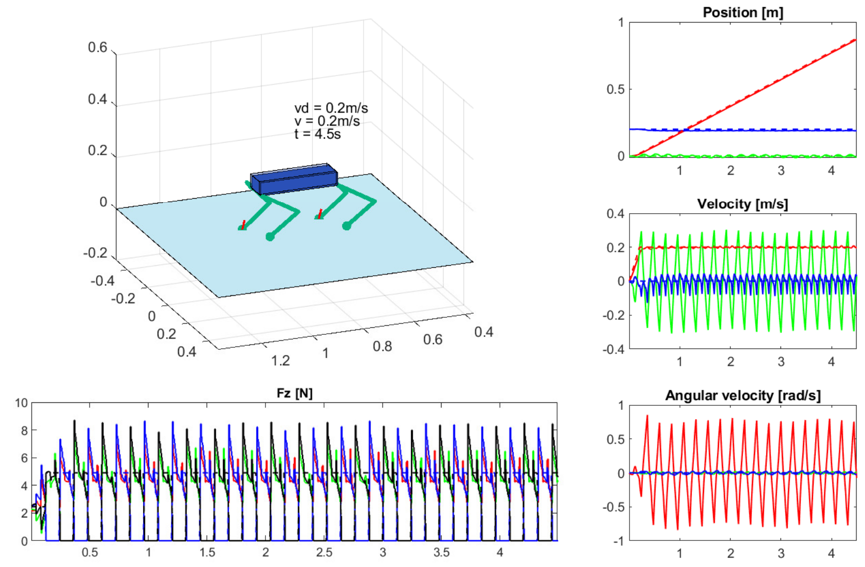

Figure 8: Simulation with $v_x = 0.2$, $\mu = 0.5$ and $mass = 1$.

3.4 Gait 3 - Gallop

In that case, using the default parameters, the robot doesn't do a good position tracking. Not only it is slow in tracking it, it also happens that during the motion it rotates itself increasing its orientation error.

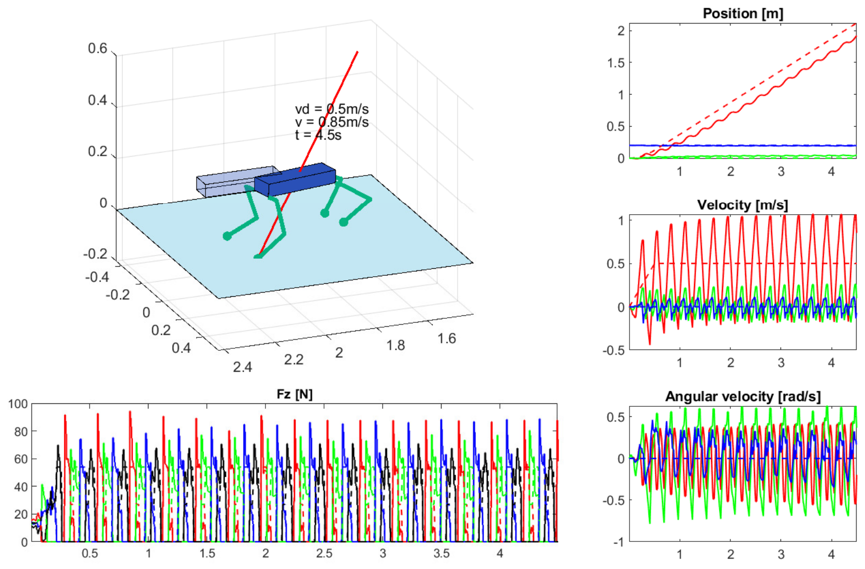

Figure 9: Simulation with default values.

Changing one parameter at a time, with the other set at default value, the results are:

- Changing the mass:
  - $mass = 11Kg$ the tracking problem is more evident and the orientation error as well is bigger. The $GRFs'$ peak value is 163N.
  - $mass = 1Kg$ the robot does a significantly better position tracking of the reference and the orientation

error is improved too. The robot is always facing in the same direction as the one of the reference for all the simulation. The $GRFs'$ peak value is 19N.

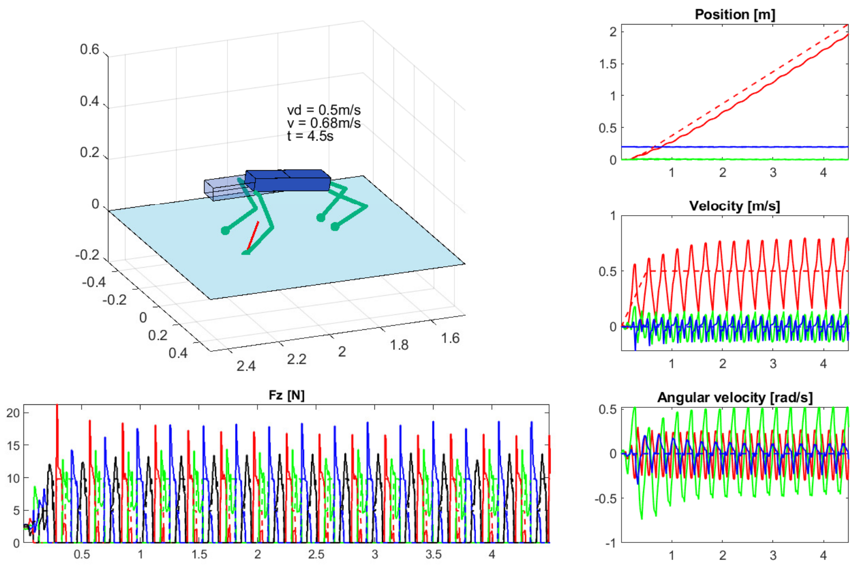

Figure 10: Simulation with mass = 1Kg.

- Changing the friction:

  - $\mu = 0.5$ the robot doesn't perform a good trajectory due to the slippage. The effect is that the robot rotates on itself while trying to move straight. This is because the $GRFs'$ horizontal component is necessary for the motion.

- Changing Desired Velocity:

  - Setting the desired velocity equal to 0.2m/s, the robot goes slower, improving the position tracking.
  - Setting the desired velocity equal to 2m/s, the robot tries to go faster, but it happens that the orientation error increases so much that the robot starts walking on its side in order to keep following the reference.

It's interesting to see how the robot, even if rotated, can still move its actuators in order to track the position and velocity reference. However the position errors are relevant, and the v_x doesn't track well the desired value. The v_x oscillation is huge, as well as the v_y and v_z which oscillation is caused mainly by the fact that the robot is moving with a great rotation error.

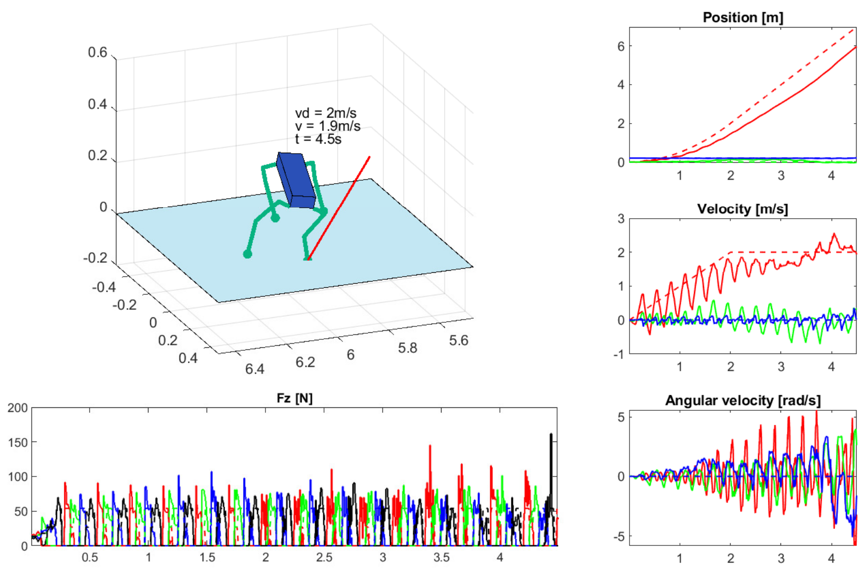

Figure 11: Simulation with $v_x = 2$m/s and the other values set as default.

3.5 Gait 4 - Run Trot

In that case the robot doesn't follow very well the position reference. It is slow in tracking it, and it also happens that in the start of the motion it rotates a bit itself increasing its z-rotation error.

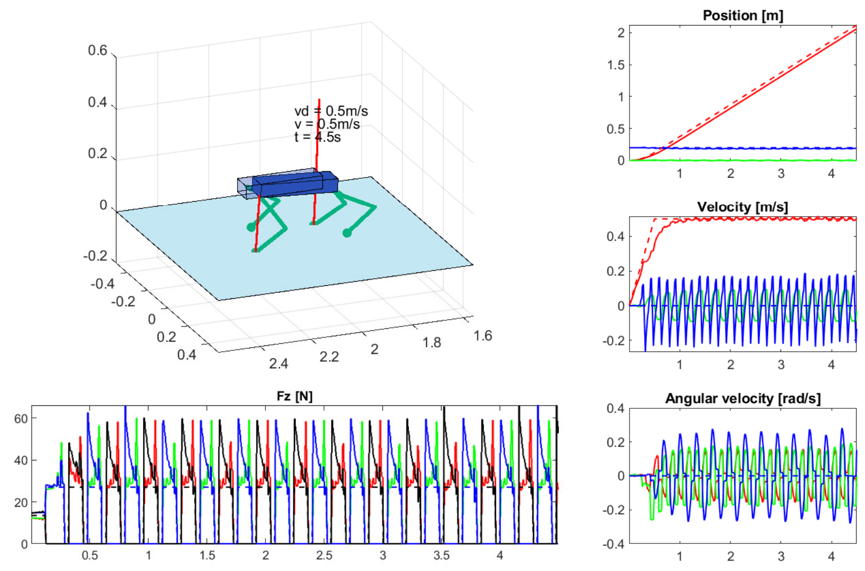

Figure 12: Simulation with default values.

Changing one parameter at a time, with the other set at default values, the results are:

- Changing the mass:

  - $mass = 11Kg$, the tracking problem is more evident and the rotation error also is bigger. The $GRFs'$ peak value is 107N.

  - $mass = 2kg$, the robot does a significantly better position tracking of the reference. Also the rotation error is very improved. The GRFs' peak value is 30N.

- Changing the friction:

  - Setting $\mu = 0.2$, the robot can still perform a trajectory tracking. It is not perfect since it seems to have a little unwanted side movement, but it is still tracking the velocity with a little value oscillation. This is because, in that case, the reaction force vectors have a very low horizontal component compared to the vertical one.

- Changing Desired Velocity:

  - $v_x = 0.2$m/s, the robot advances slower, but the pos_x error is improved.

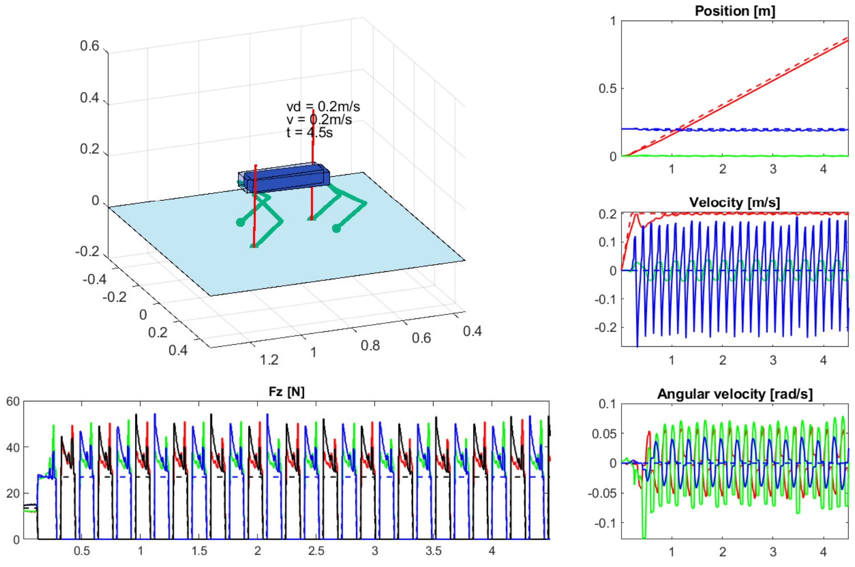

Figure 13: Simulation with $v_x = 0.2m/s$.

  - $v_x = 2$m/s, the robot tries to advance faster, but the pos_x error is bigger than the default case.

In the presented simulation it has been set $yaw = 10$. What happens is that the robot, at the beginning of the simulation, turns around the z axis in order to satisfy the request, in this case it starts moving in order to follow the reference. Despite it is moving backward, it is able to track the desired velocity.

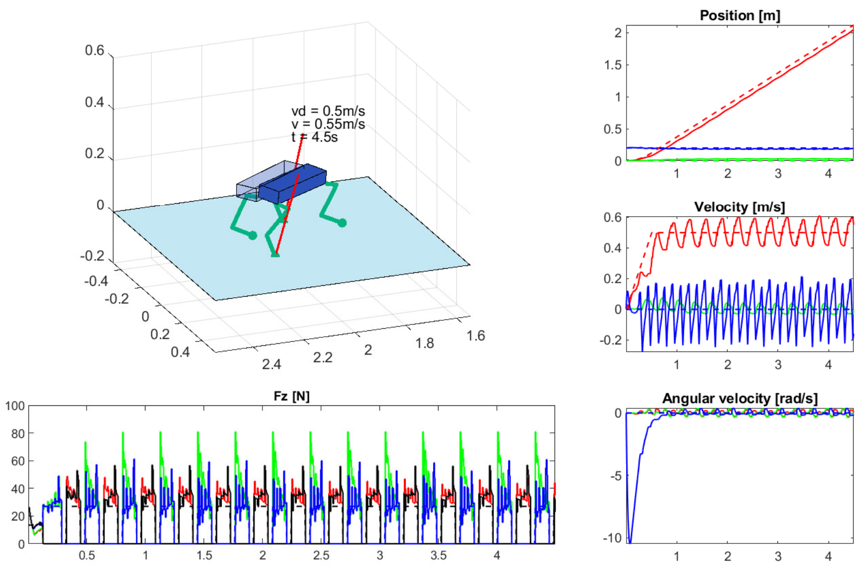

Figure 14: Simulation with $yaw = 10$.

3.6 Gait 5 - Crawl

In that case it has been noticed that using the default parameters, the robot can track the position reference with a little error on the pos_x. The v_x is tracked with an oscillation of ± 0.4m/s.

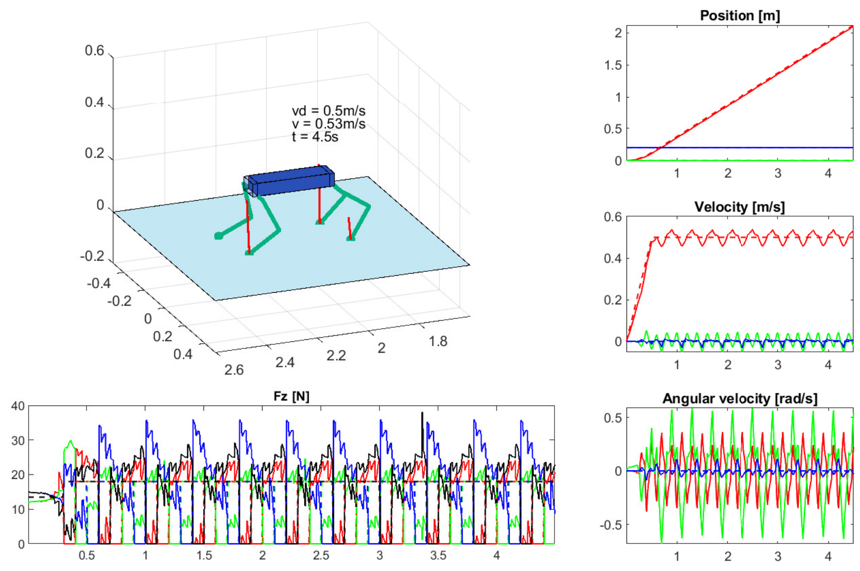

Figure 15: Simulation with default values.

Changing one parameter at a time, with the other set at default values, the results are:

- Changing the mass:

  - $mass = 11Kg$, the oscillation of the v_x, compared to the default case, is bigger. The $GRFs'$ peak value is 64N.

  - $mass = 2Kg$, the oscillation of the v_x is significantly less compared to the default case. The $GRFs'$ peak value is 12N.

- Changing the friction:

  - $\mu = 0.5$, the velocity and position tracking are very similar to the default case. This is because in that case, the reaction force vectors have a low horizontal component compared to the vertical one. This may cause the motion to be less influenced by the friction.

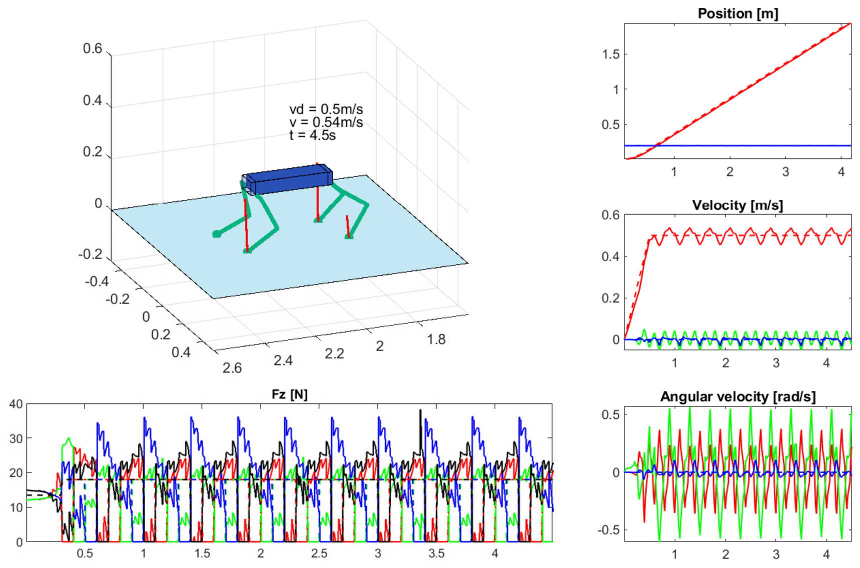

Figure 16: Simulation with $\mu = 0.5$.

- Changing Desired Velocity:

  - Setting the desired velocity equal to 0.2m/s, the robot goes slower, reducing significantly the pos_x error.

  - $v_x = 2$m/s, the robot tries to go faster, but the pos_x error is bigger than the default case.

In the presented simulation the link dimensions was changed. $l_1 = 0.04$ and $l_2 = 0.24$. It's interesting to see that the robot is able to track the desired velocity and position reference in the same way as the default case.

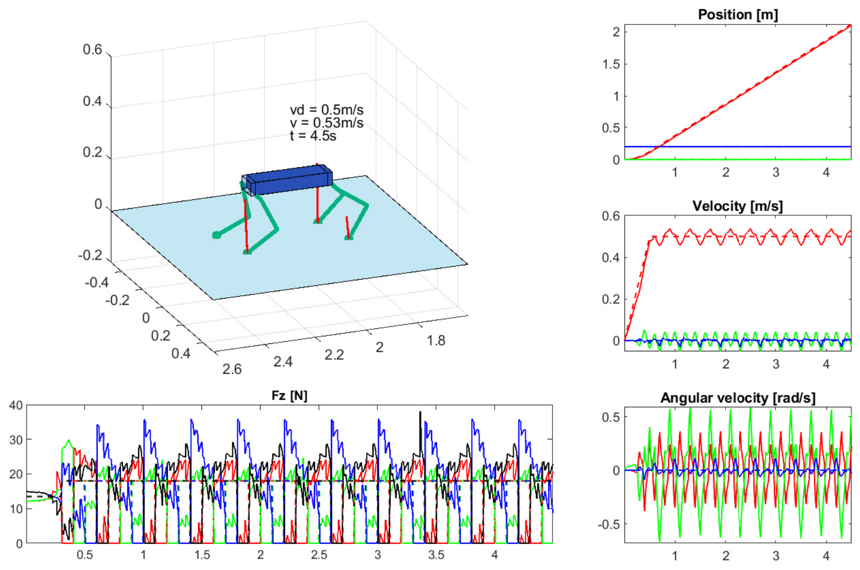

Figure 17: Simulation with different values for links dimensions.

4 Exercise 4

It is requested to answer some questions about the legged robot shown in figure, taking into account the following assumptions:

- Foot and leg are mass-less,
- Point T represents the Toe,
- Point H represents the Heel,
- Point P represents the Ankle.

4.1 Answer to Question A

Starting from the assumption of having a mass-less Foot and leg, it is possible to model the system like an inverted pendulum. Since there aren't any actuators in the point P, the system is free to evolve without any damping forces. Then it is possible to say that $\theta = \frac{\pi}{2}$ is an unstable equilibrium point. Which means that adding a little perturbation to the system in this state, the system goes to a state different from $\theta = \frac{\pi}{2}$.
Said that, it is possible to state that, without an actuator in P, $\theta = \frac{\pi}{2} + \epsilon$ is an unstable state for the system.

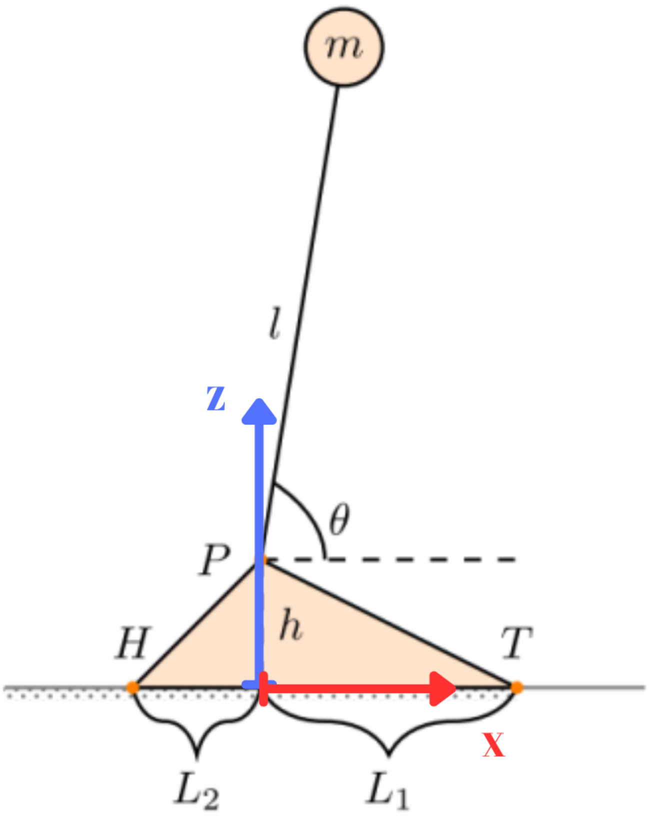

Figure 18: Given system.

4.2 Answer to Question B

It is request to compute the Zero Moment Point (ZMP), which, in a generic case, can be computed like:

$$ZMP : p_z^{x,y} = p_c^{x,y} - \frac{p_c^z}{\ddot{p}_c^z - g_0^z}(\ddot{p}_c^{x,y} - g_0^{x,y}) + \frac{1}{m(\ddot{p}_c^z - g_0^z)} S\dot{L}^{x,y} \tag{4}$$

In the studied case, the body of the robot is a point of mass which can only rotate around the y-axis since it lies on the $x - z$ plane. For this reason the y-component of the zero moment point can be considered to be null.
So it is possible to consider only the x-component of the zero moment point which can be computed as:

$$ZMP : p_z^x = p_c^x - \frac{p_c^z}{\ddot{p}_c^z - g_0^z}(\ddot{p}_c^x - g_0^x) + \frac{1}{m(\ddot{p}_c^z - g_0^z)} S\dot{L}^y \tag{5}$$

where the components of the center of mass and its derivatives are:

- $p_c^x = l \cdot \cos\theta$,
- $p_c^z = l \cdot \sin\theta + h$,
- $\ddot{p}_c^x = -l \cdot \ddot{\theta} \cdot \sin\theta - l \cdot \dot{\theta}^2 \cdot \cos\theta$,
- $\ddot{p}_c^z = l \cdot \ddot{\theta} \cdot \cos\theta - l \cdot \dot{\theta}^2 \cdot \sin\theta$,

the components of the gravity vector are:

- $g_0^x = 0$ because of the flat horizontal surface,
- $g_0^z = -g$ because of the flat horizontal surface,

the angular momentum and its derivative are:

- $L^y = m \cdot l^2 \cdot \dot{\theta}$,
- $\dot{L}^y = m \cdot l^2 \cdot \ddot{\theta}$,

the S matrix can be expressed as:

- $\mathrm{S} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$,

So it is now possible to compute the x-component of the zero moment point.

$$p_z^x = l \cdot \cos\theta - \frac{l \cdot \sin\theta + h}{(l \cdot \ddot{\theta} \cdot \cos\theta - l \cdot \dot{\theta}^2 \cdot \sin\theta) + g}\left(-l\ddot{\theta}\sin\theta - l\dot{\theta}^2\cos\theta\right) - \frac{l^2 \cdot \ddot{\theta}}{l \cdot \ddot{\theta} \cdot \cos\theta - l \cdot \dot{\theta}^2 \cdot \sin\theta + g} \tag{6}$$

4.3 Answer to Question C

It is requested to state the limit values for θ to avoid falling, with an actuator in P.
In order to answer this question the assumption is that the actuator is capable of perfectly cancelling the torque around the point P due to the gravity (i.e., $\ddot{\theta} = 0$, $\dot{\theta} = 0$).
To avoid to fall, the stability condition to achieve is to keep the projection on the ground of the mass m inside the support polygon, which, in this study case, is equal to the line L = L1 + L2.
If the following stability condition is satisfied, it is possible to keep the stability of the robot without having it falling:

$$-L_2 \leq l \cdot \cos\theta \leq L_1 \implies \arccos\left(\frac{L_1}{l}\right) \leq \theta \leq \arccos\left(-\frac{L_2}{l}\right) \tag{7}$$