DIE TI. UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

# Automation and Robotics Engineering

## Field and Service Robotics Course

Report – Homework 4

Student: Salvatore Granata - P38000219

**Abstract**

This report documents the development of Homework 4, written in LaTeX, where all contents and code were created and written by Salvatore Granata with conceptual organization improvements made thanks to AI tools. It shows the solutions to the proposed exercises concerning application of theoretical knowledge related underwater robotics and the design and control of legged robots

The GitHub repository cointaining the report and the all the code sources of the developed homework can be found at the following link: https://github.com/Salvatoregr/homework4_FSR.git.

## Contents

## 1 Exercise 1 - Buoyancy Effect

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

### 1.1 The Buoyancy Effect in Underwater Robotics

When a rigid body is submerged in a fluid under the effect of gravity, two more forces must be considered: the gravitational force and the buoyancy.

The buoyancy is a hydrostatic effect, since it is not function of the relative movement between the body and the fluid, and expresses the ability or tendency of an object to float in water or other fluid. This arises from the fact that fluid pressure increases with depth and from the fact that the increased pressure is exerted in all directions.

Given the submerged weight of the body $w$:

$$w = m\|\bar{g}\|$$

the Archimedes' Principle states that the Buoyant Force $b$ on a submerged object is equal to the weight of the liquid displaced by the object.

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

and this force acts along the $z$-axis of the system

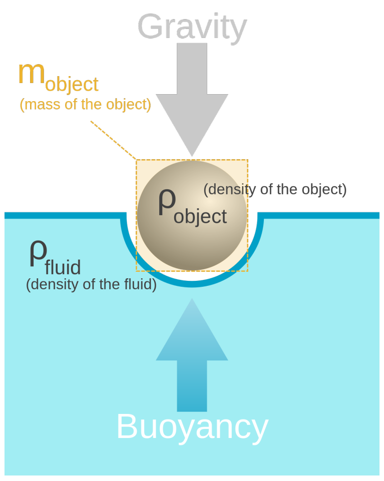


Figure 1: Buoyancy and Gravity effect on a body submerged in a fluid (water)

Buoyancy can be classified into three types: positive, negative, and neutral.

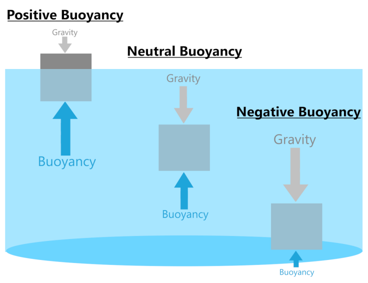


Figure 2: Types of Buoyancy effects

- Underwater Robot with *positive buoyancy* floats upwards instead of sinking because the buoyant force exceeds the robot's weight. This can be an advantage as the robotic system will naturally rise to the surface if there is a failure without using motors, facilitating easy retrieval. However, a drawback is that it will need to use more motor torques to dive and maintain a certain depth.
- Underwater Robot with *neutral buoyancy* neither floats nor sinks but stays at a constant depth. This is advantageous because they can maintain a specific depth without continuous motor usage, since they're only needed to move to different depths or up to surface.
- Underwater Robot with *negative buoyancy* sinks in water because its weight is greater than the buoyant force. This characteristic allows the robotic system to conserve energy while descending, as they naturally sink.

The interaction of gravity and buoyancy forces is fundamental to the design and operation of underwater vehicles, because is important for achieving stable underwater navigation.

In particular, the gravity force acts downward at the center of mass of the underwater vehicle, while the buoyancy force acts upwards at the center of buoyancy which is not always the same of the center of mass. When these centers are aligned, the forces act along the same direction and don't produce any rotational torque, leading to stability. This alignment is particularly important for tasks requiring precise maneuverability and position holding, such as underwater inspection, surveying, and intervention operations. If the center of mass and center of buoyancy are not aligned, it creates a torque effect, causing rotational motion that in the worst case could cause the robot to flip.

A design strategy to align them might involve placing heavy components (like batteries and electronics) to adjust the center of mass and incorporating buoyant materials to fine-tune the center of buoyancy. Additionally, dynamic buoyancy control systems, such as variable ballast tanks, can be used to adjust buoyancy in real-time, aiding in depth control and stability.

## 1.2 Buoyancy Effect in Aerial Robotics

Since the total force along the $z$-axes can be expressed as:

$$f_z = gV(\rho_{fluid} - \rho_{robot})$$

In aerial robotics, such as drones and UAVs, the Buoyancy effect is negligible because the density of air is significantly lower than the density of the moving mechanical system, giving a much smaller buoyant force which is negligible, while underwater applications, instead, the density of the water is comparable with the density of the robot (air has a density of about $1.2kg/m^3$, and water has a density of about $997kg/m^3$)

## 2 Exercise 2 - True or False Expressions

Briefly justify whether the following expressions are true or false.

a. The added mass effect considers an additional load to the structure.

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.

c. The damping effect helps in the stability analysis.

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

### 2.1 Expression 1 - Added Mass Effect

It's **false**. During the motion of an underwater robot, the fluid surrounding the robot is accelerated with the body itself, so a force is necessary to achieve this acceleration. Also, the fluid exerts a reaction force which is equal in magnitude and opposite in direction and this reaction force is the defined as *added mass contribution*.
It's important to underline that, the added mass is not a quantity of fluid to add to the system such that it has a bigger mass. It's called "added mass" because it's like the robot has an additional mass, to express the fact that it moves like a body which is heavier, but actually the mass of the robot doesn't change. It's because when the robot accelerates, the particles of water surrounding the robot prevent the motion and it's like having an additional mass inside the body, but in truth it's just an opposite force. So the added mass effect is function of the body's geometry and this force is composed by a force given by the product between an acceleration and a mass plus a Coriolis and centripetal contribution, which need to be considered in the dynamic model of an underwater robot.

### 2.2 Expression 2 - Added Mass Effect

It's **true**. As expressed in the previous point, during the robot motion the fluid exerts a reaction force (called added mass contribution) equal in magnitude and opposite in direction, which has a significative module when the fuild is the water, because its density is, in that case, comparable to the density of the robotic system. When the fluid is the air instead, the density is much lower compared to the density of an UAV and for this reason the added mass effect is negligible.

### 2.3 Expression 3 - Damping Effect

It's **true**. Damping effect is very important in the stability analysis of underwater robotics. Since this kind of robot moves through the water, the viscosity of the fluid leads to dissipative forces, including drag and lift forces on the body that improve the stability but they are a disturbances for the motion.

### 2.4 Expression 4 - Ocean Current

It's **false**. Ocean currents, generated from various factors such as tidal movements, wind patterns, temperature gradients, salinity differences, and Coriolis forces, exert a significant effect on underwater vehicles. Unlike unmanned aerial vehicles (UAVs), where the effect of wind can often be neglected, ocean currents cannot be neglected for underwater applications as it's very relevant for most of the application, since even when a vehicle hovers in water, it can still experience drift due to these currents, necessitating compensation.

While modeling ocean currents as a force acting on the vehicle is relatively easy (it can be modeled as a force that pushes the robot along a direction), determining the frame of reference to express it can be difficult.

Since all the studied controllers have been developed in the body frame, it would be preferred to express the ocean current effect in the same frame, but then the formulation becomes very difficult due to the dependency on the robot's orientation and other nonlinearities that would make the expression hardly nonlinar.

For this reason, to simply manage that effect, it's highly suggested to express it in the *world fame* as in this case it can be considered as a *constant* and irrotational effect. The drawback of that tho is that in world frame a lot of sensors measure are not available and in order to apply the controllers seen in aerial robotics, a multiplication with the rotation matrix is required to express this information back in the body frame. In this way the effect can be added to the dynamic model of a rigid body moving in a fluid simply considering the relative velocity in the body-fixed frame.

## 3 Exercise 3 - Quadruped Control and Simulation

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.

### 3.1 Quadratic Function Implementation

The quadratic function has been implemented in the MATLAB script using the solver qpSOLVER. In oparticular, once followed the instructions in the file MAIN.m, by using the defined function:

```
[zval, basic_info, adv_info]=qpSWIFT(sparse(H),g,sparse(Aeq),beq,sparse(Aineq),bineq)
```

The function above solves the quadratic problem with the following form[1]

$$\begin{aligned} \min \ & \tfrac{1}{2}x^T H x + g^T x \\ \text{s.t.} \ & A_{ineq}x \leq b_{ineq} \\ & A_{eq}x = b_{eq} \end{aligned}$$

The result of the QP problem is stored in a variable called zval in order to be used in the resolution

### 3.2 Simulation with different parameters

Once the robot model and control is avaiable, it's worth making some simulations to better understand the robot's behaviour. With the given robot, the available gaits can be listed as:

- trot;
- bound;
- pacing;
- gallop;
- trot run;
- crawl;

The simulations were performed on all types of gaits and comparisons were made by varying some parameters of the robot and the environment in which it moves, such as the friction coefficient, the mass and the desired velocity[2].

[1]Detailed information are in the qpSWIFT.m MATLAB file, cointained in the given quadrued_simulation folder

[2]Only the graphs that provide the most useful information to understand the robot's performance will be shown and discussed. Irrelevant behaviors for certain gaits will not be displayed, but they will be available in the GitHub repository of the homework.

### 3.2.1 Trot

With this gait, the robot moves by synchronously shifting first the left front leg and the right rear leg, and then the right front leg and the left rear leg. Running the script with a default configuration which considers a mass of $5.5Kg$, a friction coefficient $\mu = 1$ and a desired velocity of 0.5 the robot is able to track the reference well and the movement is smooth.

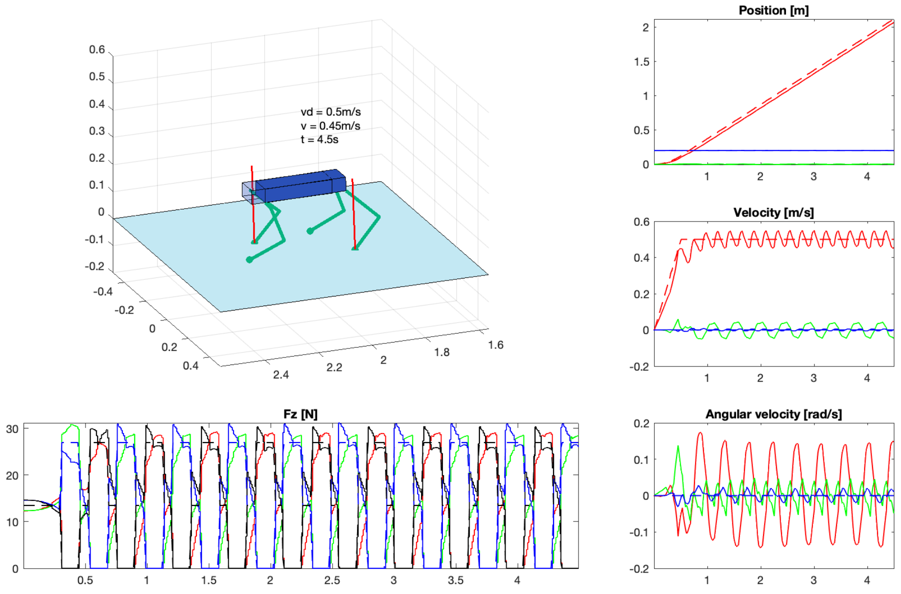


Figure 3: Simulation of Trot gait with $m = 5.5$kg, $\mu = 1$, $v_d = 0.5m/s$

An interesting behaviour could be obtained by studying the robot's movements when the reference changes, for example the desired velocity becomes $v_d = 2m/s$, without varying the other parameters. In particular, running the script with the considered settings, the results can be shown as follows:

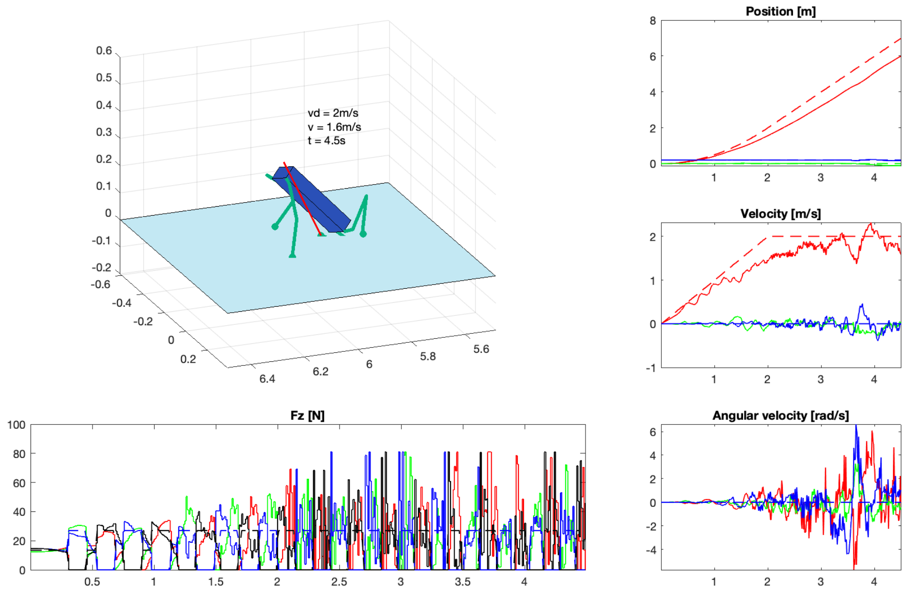


Figure 4: Simulation of Trot gait with $m = 5.5$kg, $\mu = 1$, $v_d = 2m/s$

From the simulations conducted, it was noted that the maximum achievable speed is 1.5m/s, after which the robot does not ensure smooth movement and graphical problems and bugs are highlighted. A further simulation can be conducted by varying the mass parameter, which, although modified, remains robust and is able to reach the set reference speed. By varying the parameter related to the friction coefficient, instead, it can be seen that for $\mu = 0.01$, the robot collapses as it is unable to ensure it reaches the destination without slipping.

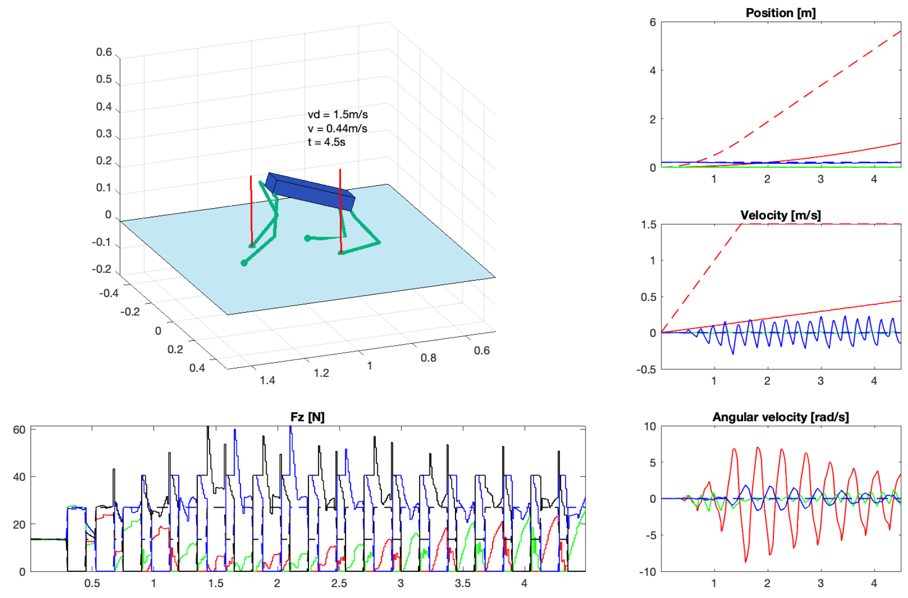


Figure 5: Simulation of Trot gait with $m = 5.5$kg, $\mu = 0.01$, $v_d = 1m/s$

### 3.2.2 Bound

With this gait, the robot moves by simultaneously shifting the rear legs first, and then the front legs. In this simulation, it is particularly interesting to study the robustness to changes in the friction coefficient. Specifically, it has been observed that, while for friction coefficients close to 1 the reference speed is correctly followed, unlike the previous gait, for $\mu = 0.01$ the robot is completely unable to move, essentially remaining stationary, because the legs slip upon contact with the ground.

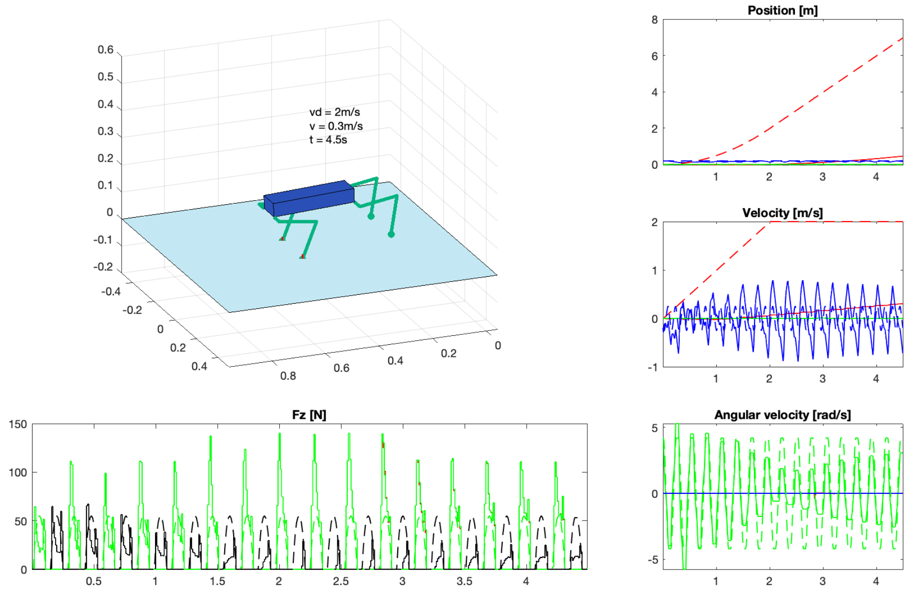


Figure 6: Simulation of Bound gait with $m = 5.5\text{kg}$, $\mu = 0.01$, $v_d = 1m/s$

By decreasing parameters such as the desired speed, better tracking can be observed even for lower friction coefficients, due to the fact that the legs move more slowly. For a simulation reason, instead, by varying the mass, even slightly, there's a system instability, causing a crash.

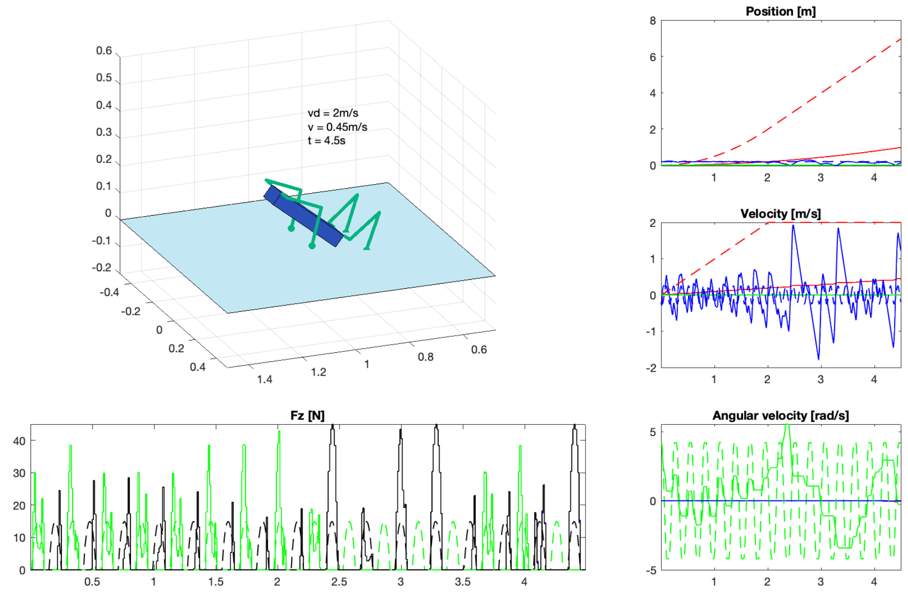


Figure 7: Simulation of Bound gait with $m = 1.5\text{kg}$, $\mu = 0.01$, $v_d = 1m/s$

### 3.2.3 Pacing

With this gait, the robot moves within the environment by simultaneously moving the right legs first and then the left legs. In this case, it has been evaluated through multiple simulations that with values of $\mu = 1$, and a mass of $5.5kg$, the maximum achievable speed by the robot with these mass values is $2m/s$.

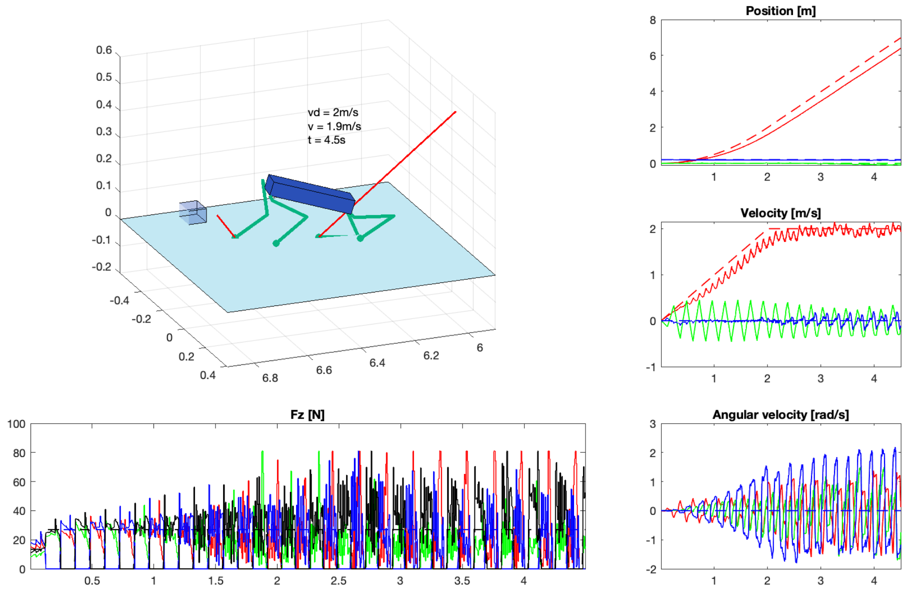


Figure 8: Simulation of Pacing gait with $m = 5.5\text{kg}$, $\mu = 1$, $v_d = 2m/s$

Based on the results obtained, the behavior of the robot was evaluated by varying the friction coefficient while keeping the mass constant. In particular, it was noted that with even a slight decrease in the friction coefficient, the robot performs position and speed tracking but collapses to the ground, showing little robustness to parameter variations.

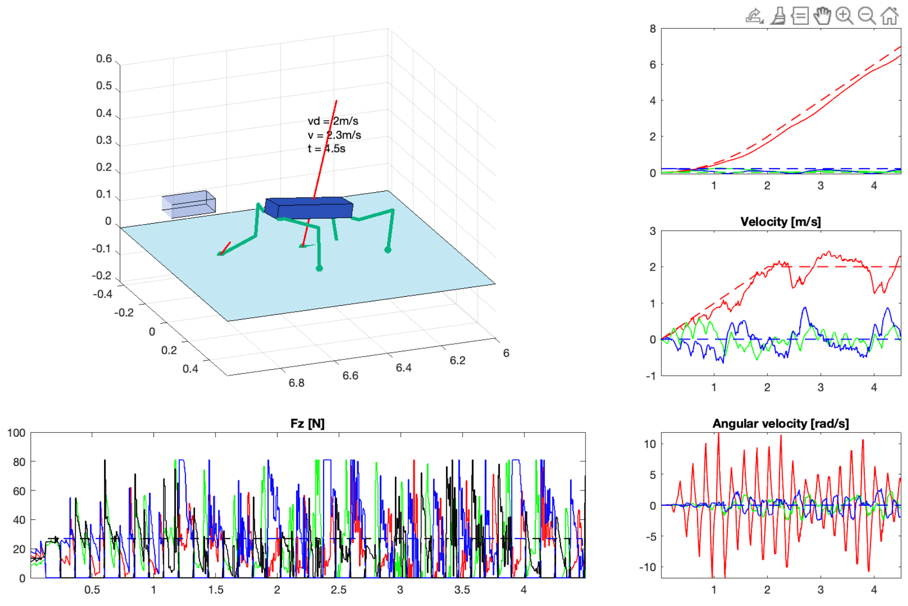


Figure 9: Simulation of Pacing gait with $m = 5.5$kg, $\mu = 0.5$, $v_d = 1m/s$

### 3.2.4 Gallop

With this gait, the robot moves emulating the gallop of a horse. Simulating the system with reference speed, mass, and friction values equal to 5.5kg, $\mu = 1$ and $v_d = 1$, the result obtained is a good tracking of the position, but an orientation error occurs.

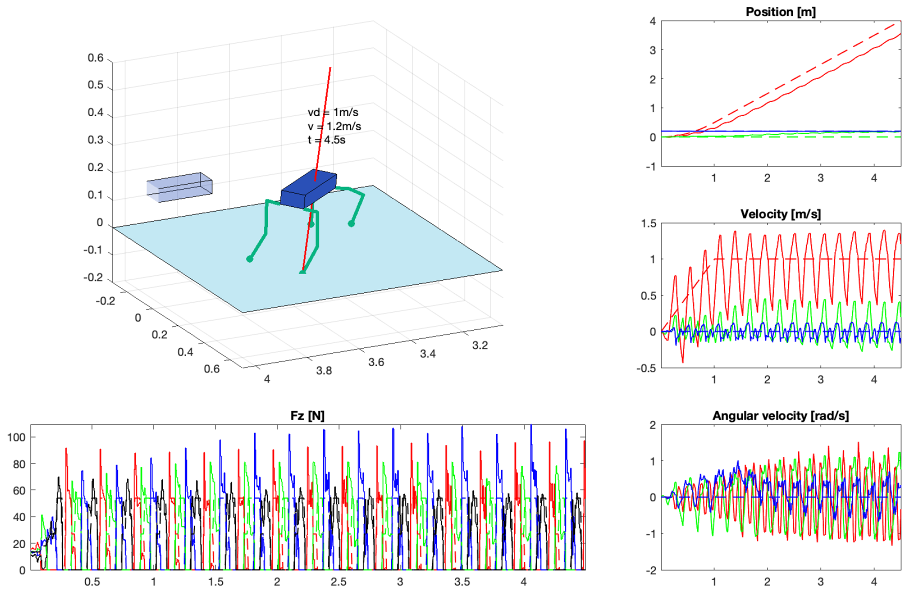


Figure 10: Simulation of Galop gait with $m = 5.5$kg, $\mu = 1$, $v_d = 1m/s$

In this case, lowering the mass value to m=1.5kg, it is possible to observe an improvement in the ability to track position and speed. Additionally, a significant decrease in the force $Fz$ and angular velocity can be noted.

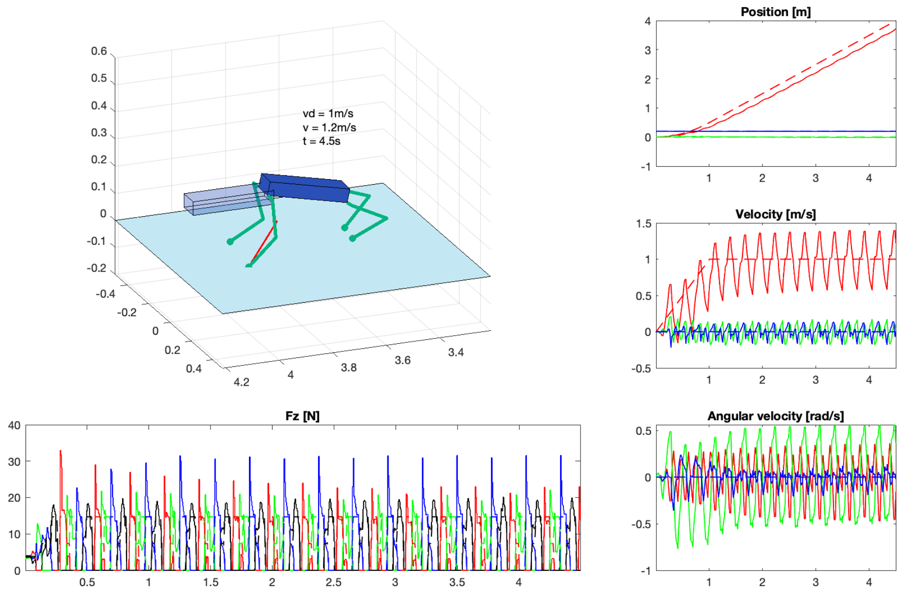


Figure 11: Simulation of Galop gait with $m = 1.5$kg, $\mu = 1$, $v_d = 1m/s$

### 3.2.5 Trot run

In this configuration, the robot moves similarly to a trot, but in running conditions. After simulating with the default parameters, it was interesting to note the robustness of the gait to a parametric variation of the mass. In fact, the system can move even with mass parameters significantly greater than 5.5kg (up to 50kg) and for friction coefficients less than one:

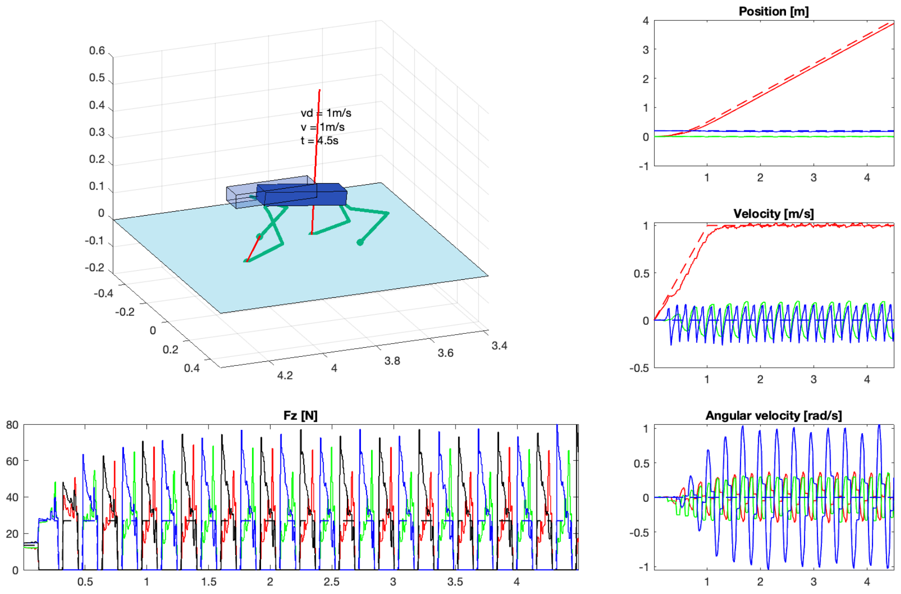


Figure 12: Simulation of Trot run gait with $m = 5.5$kg, $\mu = 0.5$, $v_d = 1m/s$

By lowering the mass to a lower value, as expected and as already happened for the other gaits, it is possible to balance the issue of having a low friction coefficient, and the robot significantly improves its tracking abilities.

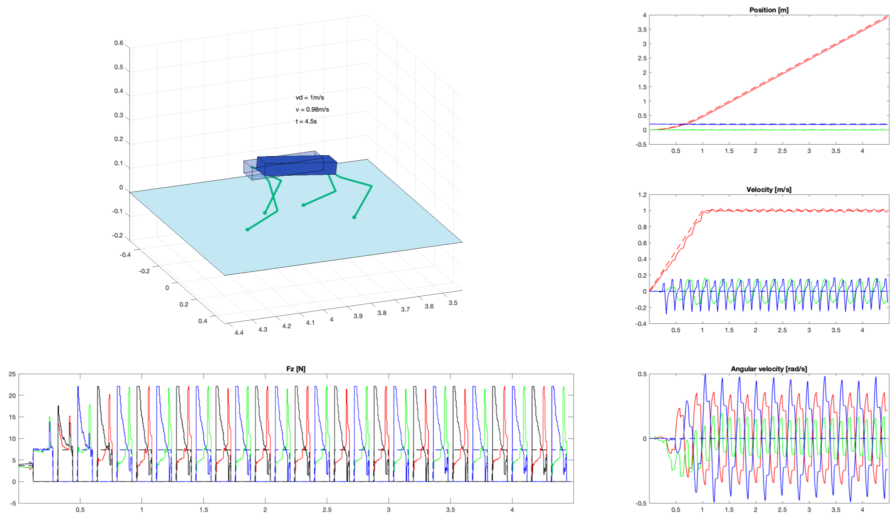


Figure 13: Simulation of Trot run gait with $m = 1.5$kg, $\mu = 1$, $v_d = 1m/s$

### 3.2.6 Crawl

With this gait, the robot moves in the environment by shifting one leg at a time. The tracking performances in this situation result very good even with a low values of friction. By maintaining the default value of $v_d = 1m/s$ and $m = 5.5$kg, with a friction coefficient set as $\mu = 0.5$, the result is the following:

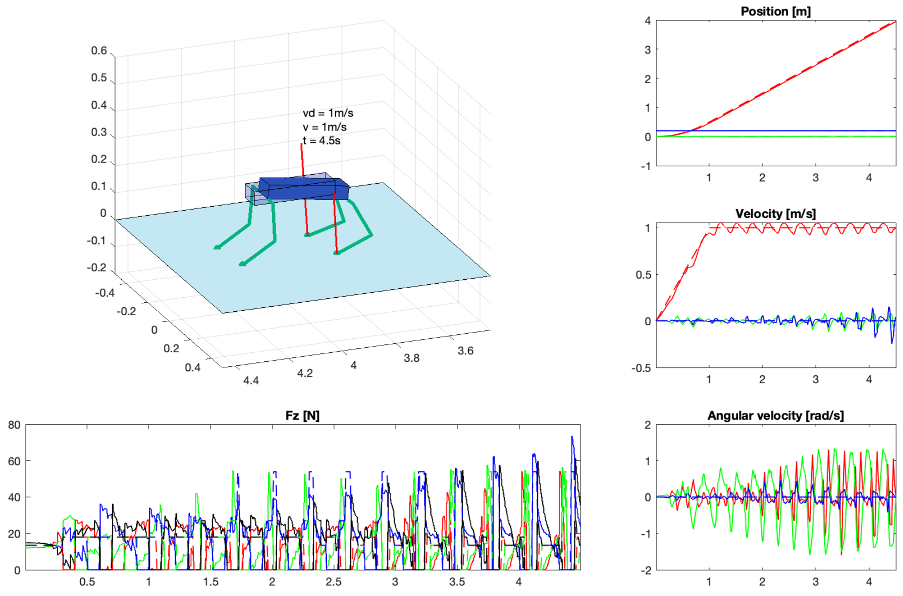


Figure 14: Simulation of Crawl gait with $m = 5.5$kg, $\mu = 0.5$, $v_d = 1m/s$

Further analysis can be conducted by studying the robot's behavior with a variation in mass. Increasing the value of the mass to $m = 10.5$kg, the obtained result is that the tracking is robust enough to track the position with smooth movements.

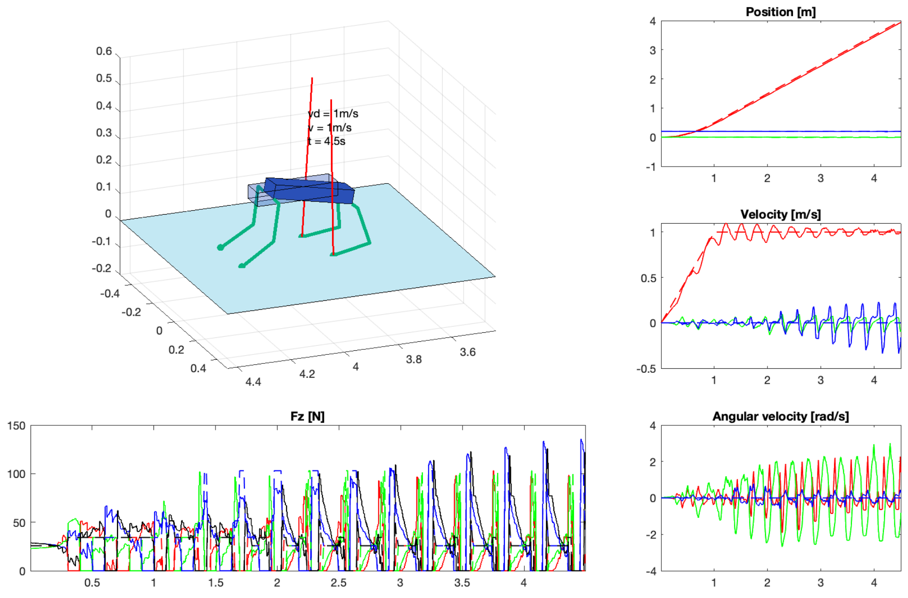


Figure 15: Simulation of Galop gait with $m = 10.5$kg, $\mu = 1$, $v_d = 1m/s$

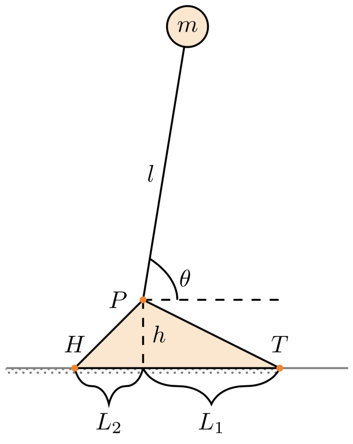
## 4 Exercise 4 - UAV External Disturbances Estimation through r-th order Momentum-Based Estimator

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 $\theta$ 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.

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

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 $\theta$ and the geometric and constant parameters (if necessary).

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 $\theta$ can you achieve without falling?



### 4.1 Stability Analysis at $\theta = \frac{\pi}{2} + \epsilon$

It's the possible to associate the movement of the body to the one of an inverse pendulum.

The system is unstable at $\theta = \frac{\pi}{2} + \epsilon$ because without an actuator at the point P, the mass is subject to the gravity, and it fall to the ground. So, in that point the static postures isn't a stable fixed point where the robot can safety stand still. In addition, given the assumption of no actuated the robot isn't able to avoid to fall so, the concept of viability and controllability aren't verify.

### 4.2 Zero-moment point on the ground

Given the mathematical expression of Zero Moment Point:

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

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

Considering the body of the robot like a point of mass $m$ that rotates only about the $y$-axis, it's possible to consider only the $x$-component of the Zero Moment Point equation that becomes:

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

Considering the following reference system:

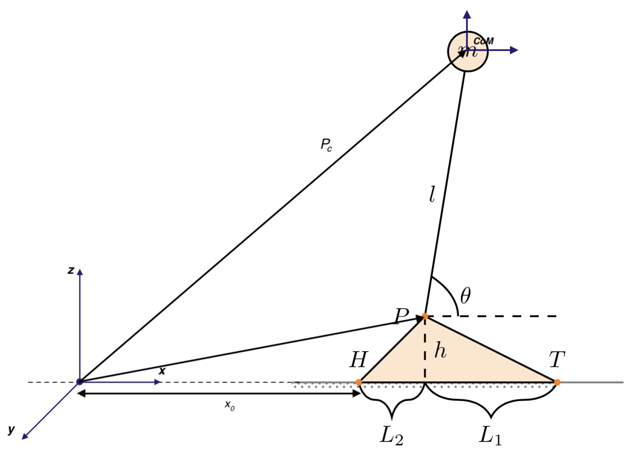


Figure 16: Reference frame of the system

It's possible to compute the ZMP[3] as:

$$ZMP = \frac{gx_0 + gl\cos\theta + hl\ddot{\theta}\sin\theta + l\ddot{\theta}x_0\cos\theta + hl\dot{\theta}^2\cos\theta - l\dot{\theta}^2 x_0\sin\theta}{-l\dot{\theta}^2\sin\theta + g + l\ddot{\theta}\cos\theta}$$

## 4.3 Maximum value $\theta$ without falling

Assuming $\dot{\theta} = \ddot{\theta} = 0$,this is the case of static condition where the robot is not moving. If the robot remains stationary without falling, this is referred to as a *fixed point* of the system, which is a posture in which the robot stands still. In static conditions, this happens when $\ddot{p}_c = \dot{L} = 0$.

These assumptions allow for the computation of the values of $\theta$ that will not cause the robot to fall. Using the previously mentioned formula under these hypotheses and considering the previous reference frame , it is possible to compute the range of $\theta$ as follows:

$$\begin{aligned}
x_0 - L_2 \leq l\cos\theta + x_0 \leq x_0 + L_1 \\
-L_2 \leq l\cos\theta \leq L_1 \\
-\frac{L_2}{l} \leq \cos\theta \leq \frac{L_1}{l} \\
\arccos(\frac{L_1}{l}) \leq \theta \leq \pi - \arccos(\frac{L_2}{l}) \qquad \text{with } \theta \in [0, \pi]
\end{aligned}$$

These represent the value of $\theta$ for which the center of mass is projected on the ground along the gravity vector inside the convex hull of the contact points of the stance legs, which is also defined as *support polygon*

[3] The computation with all the terms has been fully developed on MATLAB. The script has been uploaded on the GitHub repository associated to the Homework