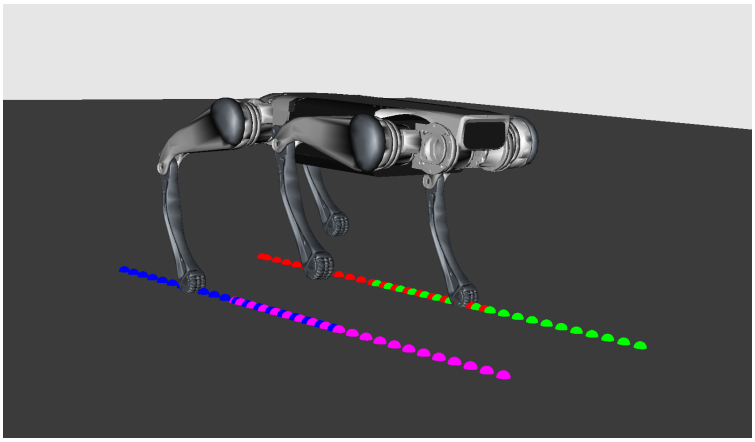
Underactuated Robots project

MPC control of quadruped robot

Paradiso Emiliano 1940454, Piccione Brian 1889051,

Pisapia Vittorio 1918590, Tedeschi Jacopo 1882789.



Abstract

This work presents the development of a simulated quadruped locomotion system controlled through Model Predictive Control. To formulate the problem as a convex Quadratic Program suitable for MPC, several simplifying assumptions were introduced to approximate the robot's dynamics.

Simulations were performed using the Lite3 quadruped robot from DeepRobotics. A dedicated footstep planner, along with ground and swing leg controllers, was implemented and kept as simple as possible to reduce the overall complexity of the work while maintaining effective locomotion behavior.

The simulation results demonstrate that the MPC framework can successfully control quadruped walking. Various gaits were tested under different conditions, showing that the controller is robust and capable of achieving stable and accurate locomotion.

GitHub link: https://github.com/Emilianogith/MPC-for-dynamic-locomotion-in-the-MIT-cheetah-3/tree/main

Contents

1 Related works

Controlling highly dynamic legged robots is difficult because their bodies are often underactuated, especially during fast gaits like bounding or galloping, and because the ground reaction forces must obey strict constraints. For instance, in locomotion modes like bounding and galloping, the robot's body lacks direct control during flight phases. Predictive control offers a solution by anticipating these periods of underactuation and planning accordingly to maintain stability. However, designing such controllers is challenging due to the robots' complex, nonlinear dynamics and the high dimensionality of their states and control inputs. The referenced paper[1] presents a predictive control approach that successfully stabilizes a wide range of gaits, including those with complex orientation dynamics.

Our project builds on the work of the MIT team, particularly their design of a whole-body Model Predictive Controller (MPC). In the referenced paper, MPC was used to compute ground reaction forces that enable various types of gaits. To make the problem tractable, the authors introduced several simplifying assumptions to reduce the complexity of the robot's dynamic model and to formulate the control problem as a convex optimization.

The control problem was formulated as a single convex optimization that models the Cheetah 3 robot as a 3D, 12-degree-of-freedom system, shown in Figure 1. The solution to this optimization can be directly used to compute motor torques that stabilize the robot, eliminating the need for a separate feedback controller.

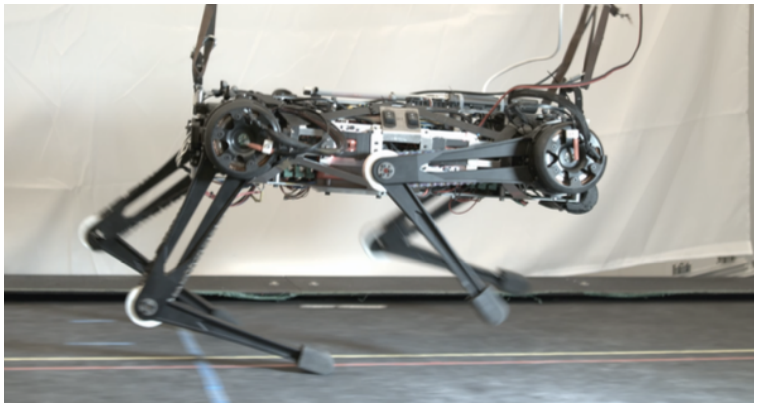


Figure 1: Cheetah 3 robot used in the reference paper.

The controller presented in the paper computes desired 3D ground reaction forces for motions with predefined foot placement and liftoff timings. During swing phases (when a foot is scheduled to be off the ground) the robot uses a swing leg controller designed to be robust to early contact with the ground. When the foot is in contact, a ground force controller is activated, as illustrated in the block diagram in Figure 2.

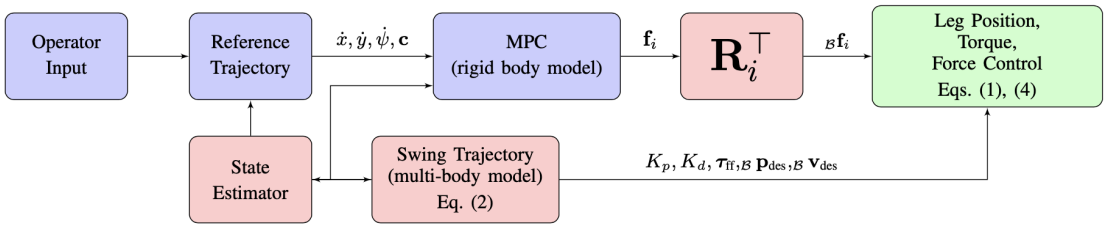


Figure 2: Control system block diagram.

1.1 Sparse vs dense problem

The original optimization problem is inherently sparse, due to its structure and the presence of large, mostly zero-valued matrices. A more efficient strategy is to exploit the problem's sparsity, storing and operating only on the non-zero elements of the sparse matrices in order to save memory. However, the method proposed in the reference [1] simplifies the problem by reducing its size, effectively eliminating the sparsity altogether. This is done by removing the dynamics constraints and associated optimization variables, and including them into the cost function. This reformulation yields a dense representation of the problem resulting in a significant speedup in computation time. In our work, we chose to maintain a simple formulation by preserving the problem in its sparse form, without rearranging it into a dense structure. Instead, we employed a solver capable of exploiting the inherent sparsity, achieving acceptable computation times.

1.2 Reference trajectory

An important aspect to remark is how the reference trajectory is generated. In the referenced paper, the desired trajectory and gait are created in real time using inputs from a video game controller. The operator provides commands such as a non-zero x,y velocity, a z position, yaw angle, and yaw rate. The reference

position is then computed by integrating these velocities. This trajectory is kept short, between 0.3 and 0.5 seconds, and is frequently updated every 0.03 to 0.05 seconds to maintain accuracy, especially if the robot is disturbed. In contrast, our project takes a slightly different approach. Instead of receiving velocity commands in real time, we set all reference parameters (such as x,y velocity, z position, yaw, yaw rate, and gait type) once at the beginning of the simulation. These values remain fixed during the entire run.

The main contribution of the referenced paper lies in the design of a Model Predictive Controller that uses a simplified model of the robot's dynamics. Despite the simplification, the model still captures many key aspects of quadruped locomotion. One of their main findings is that having a highly accurate model of the robot's behavior over the entire prediction horizon is less important than accurately capturing its instantaneous dynamics. The simplifications made by the authors proved to be well-justified. Experimental results demonstrated strong performance across a variety of gaits, including trotting, galloping, pronking, and stair walking. The controller also showed robustness to external disturbances. During a fully 3D galloping motion, the robot achieved impressive agility, reaching a maximum yaw rate of 180 degrees per second and a top linear speed of 3 meters per second.

2 Quadruped robot

Figure 3: Lite 3 quadruped robot.

The robot used in this project is the Lite 3 quadruped robot from Deep Robotics. Since we did not have access to the physical robot, all experiments were carried out in a simulation environment. The Lite 3, shown in Figure 3, is a 8.885 kg quadruped robot. Each of its four legs has three torque-controlled joints: abduction/adduction, hip, and knee.

The body and world coordinate frames used in the simulation are expressed as represented in Figure 4.

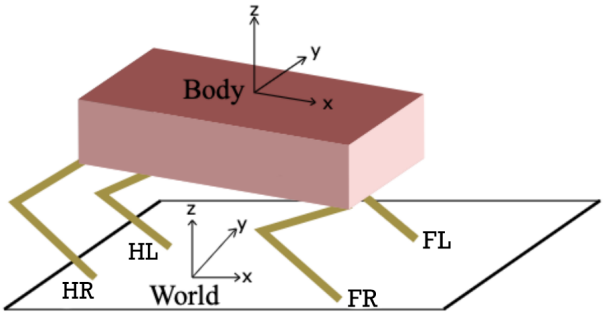


Figure 4: World and body reference frame.

To define the robot's state, we used a simplified dynamic model: the robot is treated as a single rigid body affected by contact forces at the feet, while the dynamics of the legs themselves are neglected. This is a reasonable assumption, as the legs have much less mass compared to the main body (in percentage).

Under this model, the robot's state is represented by the vector: $\mathbf{x} = (\boldsymbol{\Theta} \quad \mathbf{p} \quad \boldsymbol{\omega} \quad \dot{\mathbf{p}})^T \in SE(3) \times \mathbb{R}^6$, where $\boldsymbol{\Theta} \in SO(3)$ is the orientation of the robot frame with respect to the world coordinate frame using the roll, pitch and yaw angle representation, $\mathbf{p} \in R^3$ is the position of the center of mass, $\dot{\mathbf{p}}$ is the velocity of com and $\boldsymbol{\omega} \in \mathfrak{so}(3) \approx \mathbb{R}^3$ is the angular velocity of the rigid body.

3 Footstep planner

In our work, we implemented a footstep planner for quadrupedal locomotion, inspired by similar works but adapted to our experimental framework. Given an initial configuration, the algorithm generates a sequence of foot placements by the use of a virtual unicycle (described by its cartesian coordinates $p = (x \quad y)^T$ and its orientation θ), placed on the projection on the ground of the robot's center of mass. The motion of the unicycle is governed by a reference linear velocity v_{ref} and angular velocity ω_{ref}. The temporal evolution of the gait is controlled by defining single support (SS) and double support (DS) phases, corresponding respectively to periods in which a predefined number of legs are swinging and a period in which all legs are in contact with the ground. The durations of these phases, denoted as *ss_duration* and *ds_duration*, determine the integration time used for generating the unicycle positions used to place the footsteps in the plan. In particular, the integration time adopted is the sum of the *ds_duration* and *ss_duration*, corresponding to the elapsed time between two consecutive steps. This time permits to generate feasible foot positions that allow to track a desired motion of the center of mass, (determined by the same v_{ref} and ω_{ref} adopted for the virtual unicycle).

The update of the virtual unicycle follows:

$$p_{\mathrm{COM}}(t + \Delta t) = p_{\mathrm{COM}}(t) + R(\theta)\, v_{\mathrm{ref}} \cdot \Delta t \tag{1}$$

$$\theta(t + \Delta t) = \theta(t) + \omega_{\mathrm{ref}} \cdot \Delta t \tag{2}$$

Where R represents the rotation matrix of the angle theta around the z-axis with respect to the world reference frame, and Δt represents the simulation time step. Starting from the unicycle pose, the positions of the feet are computed using proper displacements in order to correctly reach the projection of each hip on the ground. The corresponding footsteps will be placed on the previously mentioned projections as shown in Figure 5.

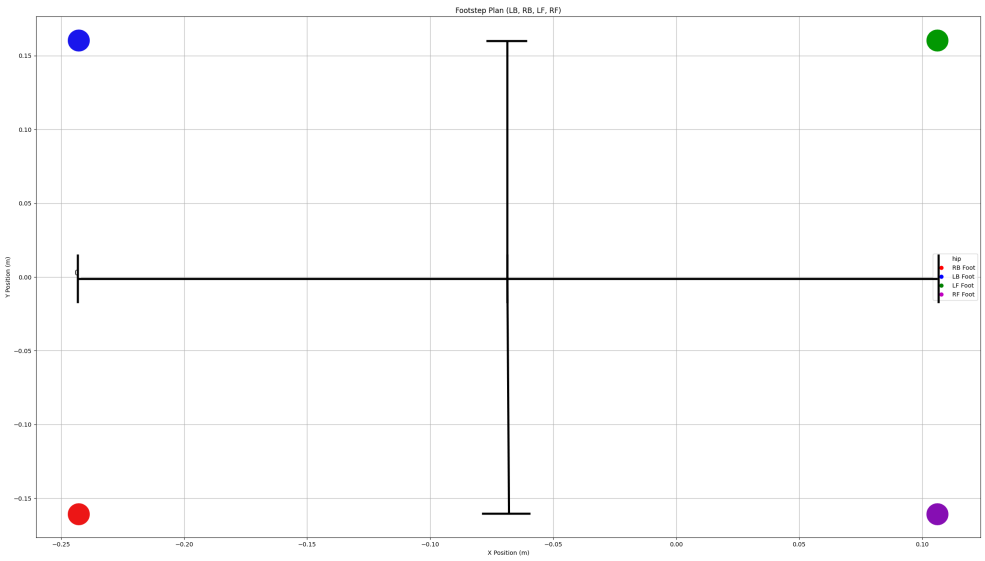


Figure 5: Displacements between feet. Front Left foot (in green), Front Right foot (in purple), Hind Left foot (in blue), Hind Right foot (in red).

The description of the gait is designed through a **forced alteration** of swinging and standing phases for each leg: if a leg was standing in the previous Single Support phase, it will be assigned as a swinging leg for the following one and vice versa, interleaved by a period of all standing legs (double support phase). An example is provided in the Figure 6.

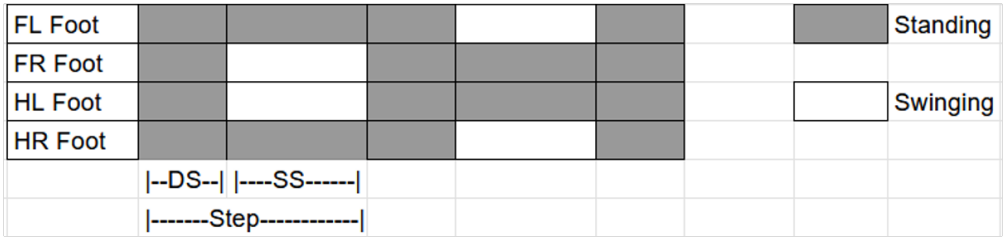


Figure 6: Example of a gait pattern generated by our approach. Each row corresponds to one foot: Front Left (FL), Front Right (FR), Hind Left (HL), and Hind Right (HR). Columns represent the progression of the gait, alternating between Double Support (DS) and Single Support (SS) phases. Empty blocks indicate that the corresponding foot is in the swing phase, while filled blocks indicate it is in contact with the ground.

The simplicity of the proposed gait generator enables a seamless definition of different gaits (which will be

described later). However, it does not support the implementation of more complex gaits, such as galloping, which require consecutive swing phases for the same leg, as shown in the following figure:

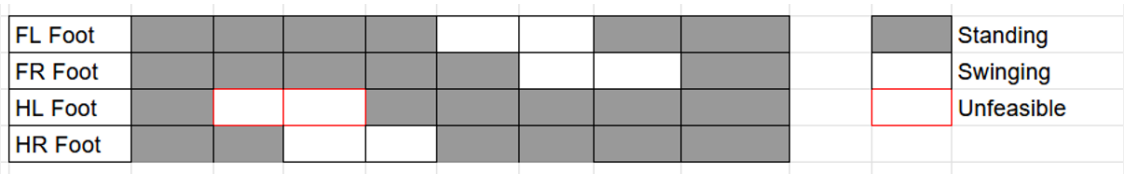


Figure 7: Gait sequence illustrating the eight distinct phases of the gallop. Consecutive swing phases for the same leg are not supported by our gait generator.

The resulting plan consists of a user-defined number of steps, each containing information about the foot pose and the swing/support state for each leg. An example of a complete plan is shown below.

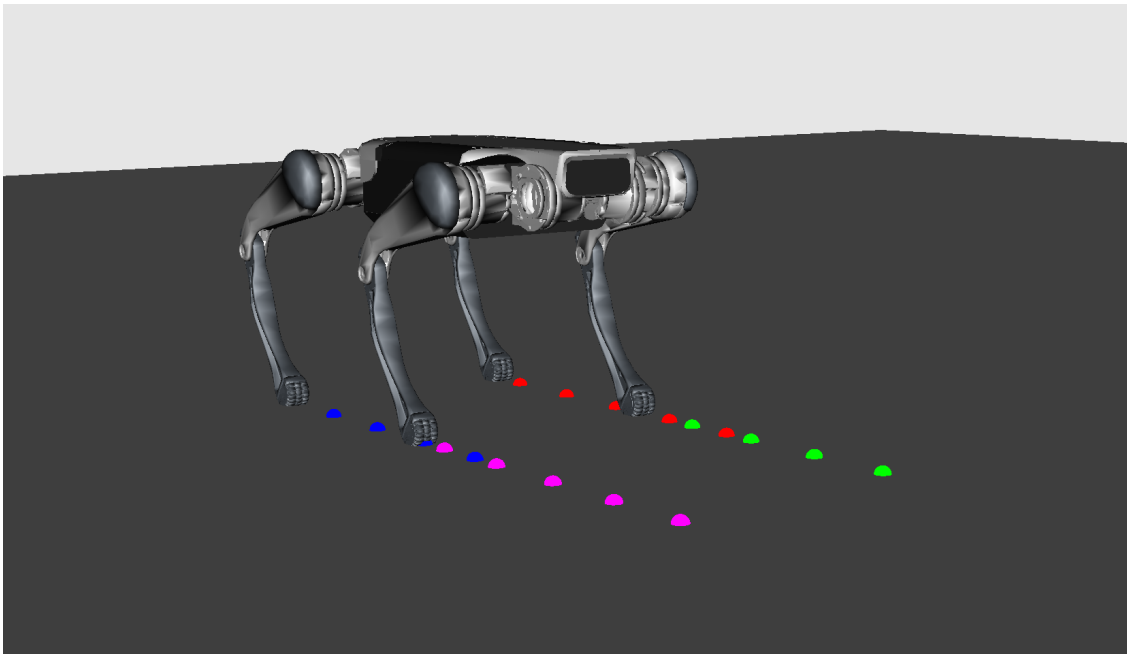


Figure 8: Example of the plan generated by the Footstep planner.

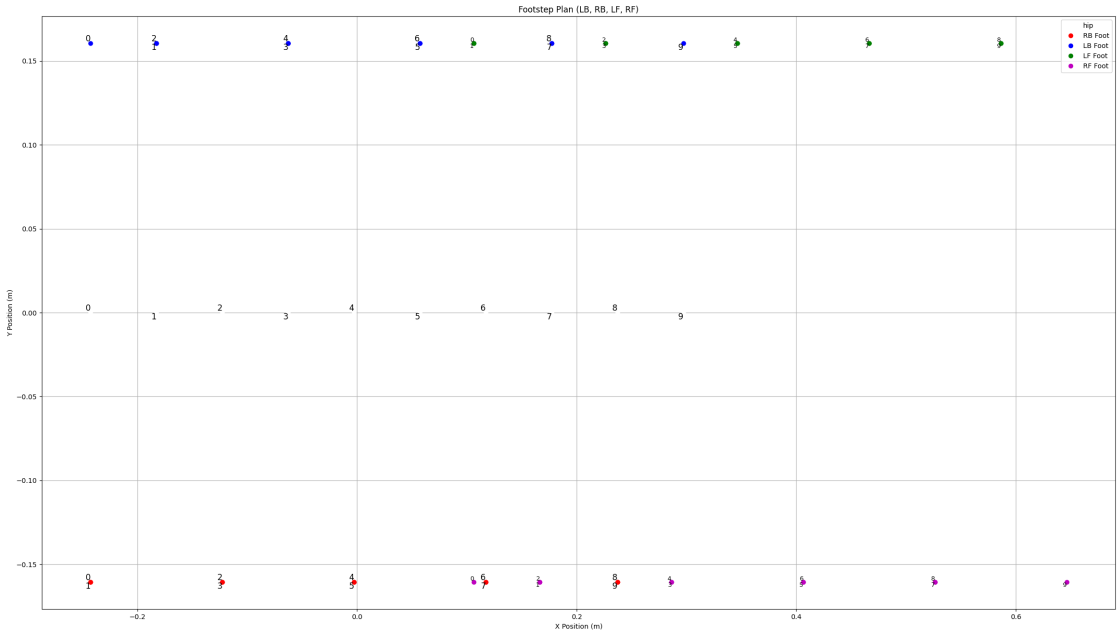


Figure 9: Plot of the planned steps in 2D.

3.1 Trotting gait

In most of our experiments, we adopted a gait characterized by the synchronous swing of the FR foot (front right) and HL foot (hind left), a phase of double support, followed by a synchronous swing of the pair FL foot (front left) and HR foot (hind right). allowing for a stable and symmetric gait cycle. The trotting gait is illustrated in the Figure 6.

3.2 Pseudo-galloping gait

Contrary to the previous gait, the Pseudo-Galloping gait is characterized by the synchronous swing of the front pair legs followed by the simultaneous swing of the hind pair of legs, separated by a double support phase. This gait cycle tries to approximate the more sophisticated gallop gait.

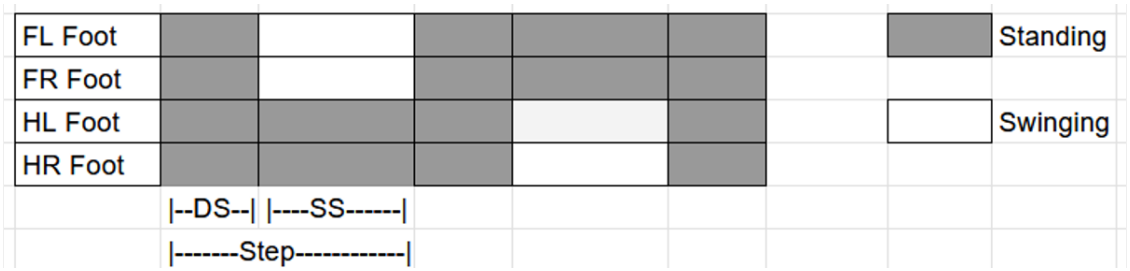


Figure 10: Pseudo-galloping gait.

3.3 Ambling gait

In this gait, the legs selected for swinging are the lateral pairs, alternating between the left side pair and right side pair, as shown in the following figure:

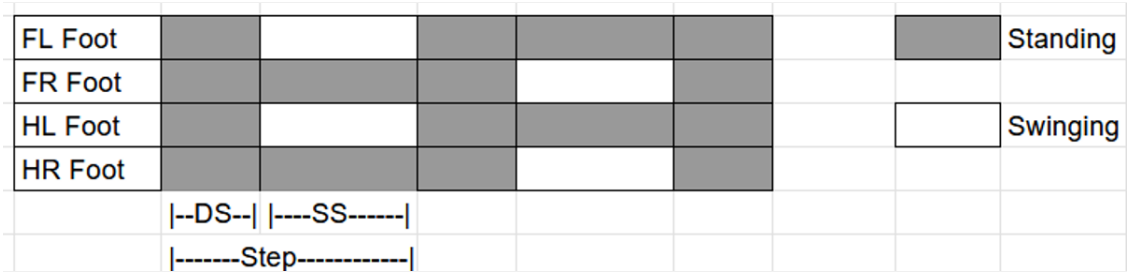


Figure 11: Ambling gait.

3.4 Pronking gait

Lastly, in the pronking gait, all the legs are simultaneously selected for swinging, resulting in the jumping transition between a feet placement and the following one. An illustration of the prancing gait is shown in Figure 12.

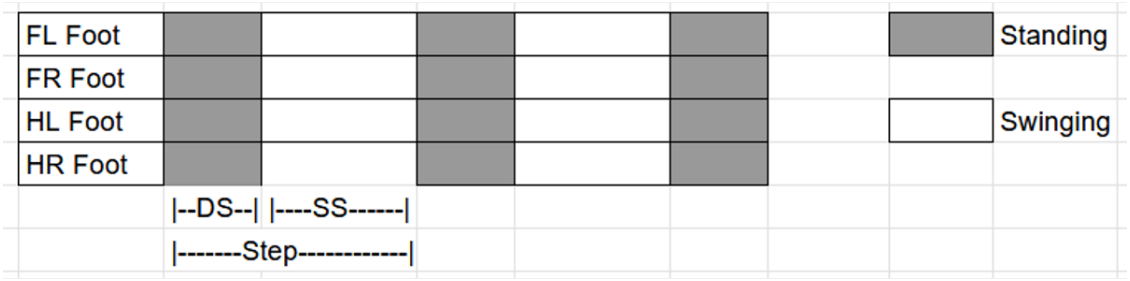


Figure 12: Pronking gait.

4 Trajectory generator

In this project, the swing trajectories for the quadruped's feet were designed to ensure smooth motion during the swing phase. The trajectory in the horizontal plane (x and y coordinates) is generated using a cubic polynomial interpolation between the initial position p_i and target position p_f of the foot on the xy-plane. This allows continuous velocity and acceleration profiles, and provides zero derivative at the boundaries.

$$p(t) = p_i + (p_f - p_i)\left(-2\left(\frac{t}{T}\right)^3 + 3\left(\frac{t}{T}\right)^2\right), \qquad t \in [0, T] \tag{3}$$

T is the duration of the single-support phase, which corresponds to the swing time. Differently, for the vertical motion, a quartic polynomial has been employed, producing a bell-shaped profile that ensures the foot lifts off and lands with zero vertical velocity and acceleration. Through this choice, sharp impacts with the ground are avoided. The desired maximum height h of the step is a hyparameter that defines the peak of the vertical trajectory.

$$z(t) = at^4 + bt^3 + ct^2, \qquad t \in [0, T] \tag{4}$$

with $a = 16\frac{h}{T^4}$, $b = -32\frac{h}{T^3}$, $c = 16\frac{h}{T^2}$.

5 Model Predictive Controller

A discrete-time finite-horizon MPC computes the desired ground reaction forces. The MPC contains information about the dynamics of the system (encoded as equality constraints), and on each iteration, starting

from the current configuration, it finds an optimal sequence of inputs for the corresponding desired trajectory of the state over a finite-length horizon, while ensuring feasibility of equality and/or disequality constraints. The first set of input is commanded to the robot while the other $N-1$ inputs are discarded. The process is repeated and new control inputs are computed.

The horizon length is N. Thanks to the simplifications adopted and the linearization of the robot's dynamics (that will be explained in the next paragraph), the optimization problem solved by the MPC can be formulated using a multiple shooting approach, resulting in the following convex optimization problem:

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{i=0}^{N-1} \|\mathbf{x_{i+1}} - \mathbf{x_{i+1,\text{ref}}}\|^2_{\mathbf{Q_i}} + \|\mathbf{u_i}\|^2_{\mathbf{R_i}} \tag{5}$$

$$\text{subject to} \quad \mathbf{x_{i+1}} = \mathbf{A_i x_i} + \mathbf{B_i u_i}, \qquad i = 0, \dots, N-1 \tag{6}$$

$$\underline{\mathbf{c_i}} \leq \mathbf{C_i u_i} \leq \bar{\mathbf{c}}_\mathbf{i}, \qquad i = 0, \dots, N-1 \tag{7}$$

$$\mathbf{D_i u_i} = 0, \qquad i = 0, \dots, N-1 \tag{8}$$

where $\mathbf{x_i}$ is the system state at i-th step of the horizon, $\mathbf{u_i}$ is the control input at step i, $\mathbf{A_i}$ and $\mathbf{B_i}$ represent the discrete time system dynamics, $\mathbf{C_i}$ and $\mathbf{c_i}$ represent inequality constraints for the control input and lastly $\mathbf{D_i}$ is a matrix that selects forces corresponding to swinging feet. $\mathbf{Q_i}$ and $\mathbf{R_i}$ are diagonal positive semidefinite matrices of weights. These constraints will be analyzed in more detail in the following section. It's worth noting that the overall problem takes the form of a Quadratic Programming (QP) problem, which can be efficiently solved using dedicated solvers such as OSQP.

The desired state trajectory $\mathbf{x_{ref}}$ for the whole horizon, is computed by integrating the commanded linear velocity $\mathbf{v_{com\_ref}}$ and angular velocity $\boldsymbol{\omega_{ref}}$.

5.1 Dynamics

The controller employed utilizes a simplified model of the robot, treating it as a rigid body subjected to external forces applied at the contact points. In this formulation, the dynamics of the legs are neglected. This simplification is justified by the fact that the legs constitute only a small fraction of the robot's total mass.

For each ground reaction force $\mathbf{f}_i$, we define the vector $\mathbf{r}_i$ as the displacement from the center of mass to the corresponding point of force application. Under the assumption of rigid body dynamics, and expressed in world coordinates, the equations governing the motion are:

$$\ddot{\mathbf{p}} = \frac{\sum_{i=1}^{n} \mathbf{f}_i}{m} - \mathbf{g} \tag{9}$$

$$\frac{d}{dt}(\mathbf{I}\boldsymbol{\omega}) = \sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{f}_i \tag{10}$$

$$\dot{\mathbf{R}} = [\boldsymbol{\omega}]_{\times}\mathbf{R} \tag{11}$$

To further reduce model complexity and enable real-time simulation, additional approximations are adopted. Considering the robot's orientation represented using Z-Y-X Euler angles (roll–pitch–yaw), the angular velocity in world coordinates can be derived from the time derivatives of these angles:

$$\boldsymbol{\omega} = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ -\sin(\theta) & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{12}$$

Assuming the robot does not assume a near-vertical posture, this relation can be inverted to obtain the rate of change of the Euler angles. Moreover, under the reasonable assumption of small roll and pitch angles, effectively assuming that the robot undergoes rotation primarily about the yaw axis, the angular velocity equation simplifies to:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \mathbf{R}_z(\psi)\,\boldsymbol{\omega} \tag{13}$$

This leads to a commonly used approximation for moment balancing in the literature, which has been shown to be effective. The moment dynamics are then expressed as:

$$\frac{d}{dt}(\mathbf{I}\boldsymbol{\omega}) \approx \mathbf{I}\dot{\boldsymbol{\omega}} \tag{14}$$

Here, the inertia tensor $\mathbf{I}_{\mathcal{B}}$ is transformed into world coordinates by rotating the body-fixed inertia about the z-axis by the yaw angle, leveraging the assumption of small roll and pitch angles:

$$\mathbf{I} \approx \mathbf{R}_z(\theta)\,\mathbf{I}_{\mathcal{B}}\,\mathbf{R}_z(\theta)^{\top} \tag{15}$$

The full translational and rotational dynamics of the system can subsequently be expressed in a matrix form:

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{\Theta}\\ \mathbf{p}\\ \boldsymbol{\omega}\\ \dot{\mathbf{p}}\end{bmatrix} = \begin{bmatrix}\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{R}_z(\psi) & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\end{bmatrix}\begin{bmatrix}\boldsymbol{\Theta}\\ \mathbf{p}\\ \boldsymbol{\omega}\\ \dot{\mathbf{p}}\end{bmatrix} + \begin{bmatrix}\mathbf{0}_3 & \dots & \mathbf{0}_3\\ \mathbf{0}_3 & \dots & \mathbf{0}_3\\ \mathbf{I}^{-1}[\mathbf{r}_1]_\times & \dots & \mathbf{I}^{-1}[\mathbf{r}_n]_\times\\ \frac{\mathbf{1}_3}{m} & \dots & \frac{\mathbf{1}_3}{m}\end{bmatrix}\begin{bmatrix}\mathbf{f}_1\\ \vdots\\ \mathbf{f}_n\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ \mathbf{g}\end{bmatrix} \tag{16}$$

This formulation is dependent on the yaw angle and footstep locations. By computing these quantities in advance and augmenting the model state to include gravity the system dynamics become linear time-varying, suitable for convex model predictive control.

The control problem is formulated as a discrete-time linear system over a finite prediction horizon. At each time step $i+1$, desired values for the foot positions and the robot's yaw angle are defined and used to construct the system dynamics.

The desired yaw angle is computed via numerical integration of a predefined angular velocity, which is specified as part of the gait design.

To compute the vector $\mathbf{r}_i$ at each time step i, the footstep planner is queried to determine the contact state of each foot: If the foot is in swing phase, its position is obtained from the predefined swing trajectory evaluated at the given timestep. While if the foot is in contact phase, its position is retrieved directly from the footstep planner's reference data.

In both cases, the relative position of the foot with respect to the center of mass is calculated by subtracting the CoM position at the corresponding timestep (obtained through numerical integration of the commanded Cartesian velocity) from the absolute foot position.

Using this information, the system matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ are iteratively constructed for each time step i across the finite horizon. This process results in a **discrete-time linear system**, which forms the basis of a convex Quadratic Programming problem.

5.2 Force constraints

The equality constraints described in (8) are used to set all the forces associated with legs that are in the swing phase at each time step i to zero, ensuring that the desired gait pattern can be correctly executed. In contrast, the forces applied by the legs in contact with the ground at the same time step are optimized to accurately track the reference state trajectory and to support future gait execution over the prediction horizon.

To achieve this, the contact feet must satisfy the constraints in (7) , which enforce:

$$f_{\min} \leq f_z \leq f_{\max} \tag{17}$$

$$-\mu f_z \leq \pm f_x \leq \mu f_z \tag{18}$$

$$-\mu f_z \leq \pm f_y \leq \mu f_z \tag{19}$$

These constraints are crucial for limiting the vertical (z-axis) force to be within specified maximum and minimum values, and ensuring that the contact forces lie within the bounds of the friction cone essential condition in order to prevent slipping.

6 Ground controller

Since the optimization problem is formulated to compute the ground reaction forces that satisfy all constraints (including dynamics and friction cone conditions), we can directly apply the resulting forces to the legs planned to be in contact with the ground. This is done by mapping the computed ground forces to joint torques using the following relation:

$$\boldsymbol{\tau}_i = \mathbf{J^T}_i \mathbf{R^T}_i \mathbf{f}_i \tag{20}$$

where $\mathbf{R}_i$ is the i-th rotation matrix that transforms vectors from the robot frame to the world frame, $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the Jacobian of the i-th foot, and $\mathbf{f}_i$ is the ground reaction force obtained from the MPC for the i-th leg.

7 Swing controller

A smooth trajectory for the swing leg has already been derived in (3), (4). The next step involves tracking this trajectory over time. To achieve this, a simple control strategy combining **feedback** and **feedforward** components is employed to compute the joint torques for the corresponding leg.

The control law is given by:

$$\boldsymbol{\tau}_i = \mathbf{J}_i^\top \left[\mathbf{K}_p(\mathbf{p}_{i,\text{ref}} - \mathbf{p}_i) + \mathbf{K}_d(\mathbf{v}_{i,\text{ref}} - \mathbf{v}_i)\right] + \boldsymbol{\tau}_{i,\text{ff}} \tag{21}$$

Where $\boldsymbol{\tau}_i$ is the joint torque vector for the i-th leg, $\mathbf{J}_i$ is the Jacobian matrix of the i-th leg, $\mathbf{p}_{i,\text{ref}}$ and $\mathbf{p}_i$ are the reference and current positions of the end-effector (foot) in the world frame, and $\mathbf{v}_{i,\text{ref}}$ and $\mathbf{v}_i$ are the reference and current Cartesian velocities of the foot. $\mathbf{K}_p$ and $\mathbf{K}_d$ are diagonal positive-definite gain matrices representing the proportional and derivative control gains, respectively, and $\boldsymbol{\tau}_{i,\text{ff}}$ is a feedforward torque term.

The feedforward torque $\boldsymbol{\tau}_{i,\text{ff}}$ accounts for the Coriolis and gravitational effects as well as desired swing-leg accelerations in task space. It is computed as:

$$\boldsymbol{\tau}_{i,\text{ff}} = \mathbf{J}_i^\top \mathbf{M}_i \left(\mathbf{a}_{i,\text{ref}} - \dot{\mathbf{J}}_i \dot{\mathbf{q}}_i\right) + \mathbf{C}_i \dot{\mathbf{q}}_i + \mathbf{G}_i \tag{22}$$

Where $\mathbf{a}_{i,\text{ref}}$ is the desired acceleration of the foot in Cartesian space, $\dot{\mathbf{J}}_i$ is the time derivative of the Jacobian, $\dot{\mathbf{q}}_i$ is the joint velocity vector, $\mathbf{M}_i$ is the operational space inertia matrix (i.e., the apparent mass of the leg along the i-th direction), $\mathbf{C}_i$ is the Coriolis matrix for the leg, and $\mathbf{G}_i$ is the gravity vector for the leg.

8 Results

The proposed results were obtained using the Python-based DARTPy simulation environment. Given the available hardware, the MPC solver achieved solve frequencies in the range of 40–60 Hz across all simulations with a horizon length of 60. Although reducing the horizon length improves computational speed, it typically results in slightly reduced performance in tracking the reference trajectory. For instance, with a horizon

length of 30, some gaits achieved average solve frequencies exceeding 100 Hz, making real-time, on-board implementation feasible, as the solve time remained below the simulation time step of 0.01 seconds.

- Trotting gait

Parameter	**Value**	**Parameter**	**Value**
ss duration [s]	0.1	h [m]	0.285
ds duration [s]	0.1	μ	1.5
$v_{\text{com\_ref\_x}}$ [m/s]	0.08	N	60
ω_{ref} [rad/s]	0	dt [s]	0.01
steps	40	step_height [m]	0.08

Table 1: Simulation parameters of trotting gait

The trotting gait achieved the best performance, resulting in the lowest tracking error relative to the reference Cartesian trajectory, as shown in Figure 13. While Cartesian position tracking remained highly accurate, orientation tracking exhibited slight deviations from the desired values. Nonetheless, these errors remained bounded within 0.02 rad/s, as depicted in Figure 14. Figure 15 displays the MPC-predicted center of mass trajectory and contact forces at timestep 80. The plot highlights the controller's ability to effectively compensate for disturbances and maintain accurate tracking, thereby confirming its robustness and error-recovery capabilities.

Focusing on a single foot (specifically the FL foot, as shown in Figure 16), the force along the z-axis computed by the MPC during the stance phase is positive and reaches a feasible value of approximately 70 N. In contrast, the forces corresponding to the swing phase are set to zero, as dictated by the imposed constraints. During the single support phase, the swinging performance proved to be highly accurate, with minimal tracking errors observed in both the z and x components (see Figure 17 and Figure 18). The torque values generated by the ground controller for all legs remained bounded in magnitude

by 7.5 N·m, as illustrated for the FL leg in Figure 19.

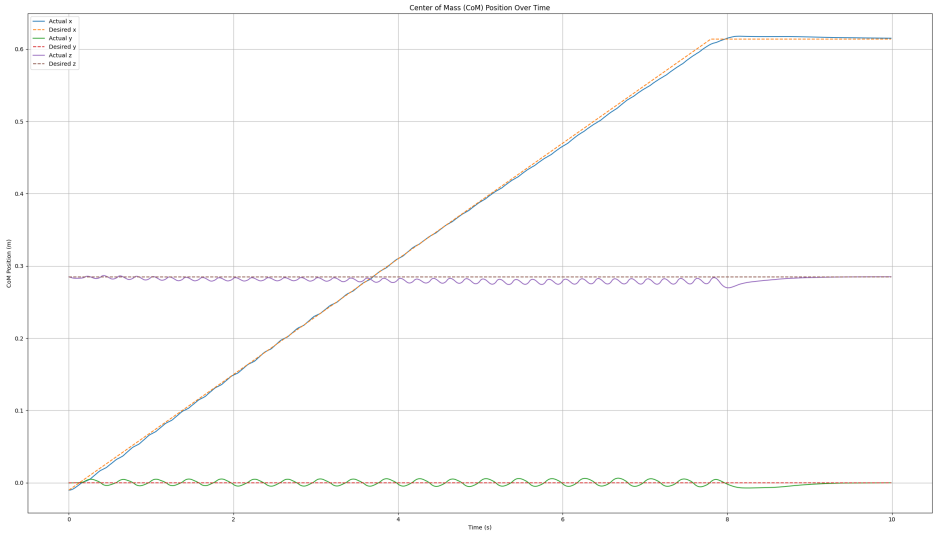


Figure 13: Cartesian tracking performance of the CoM across the simulation duration.

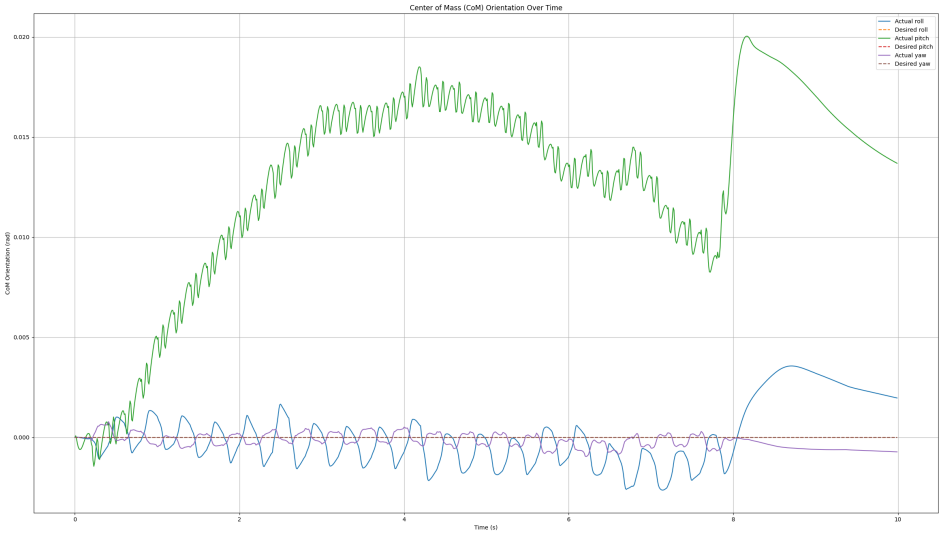


Figure 14: Orientation tracking performance across the simulation duration.

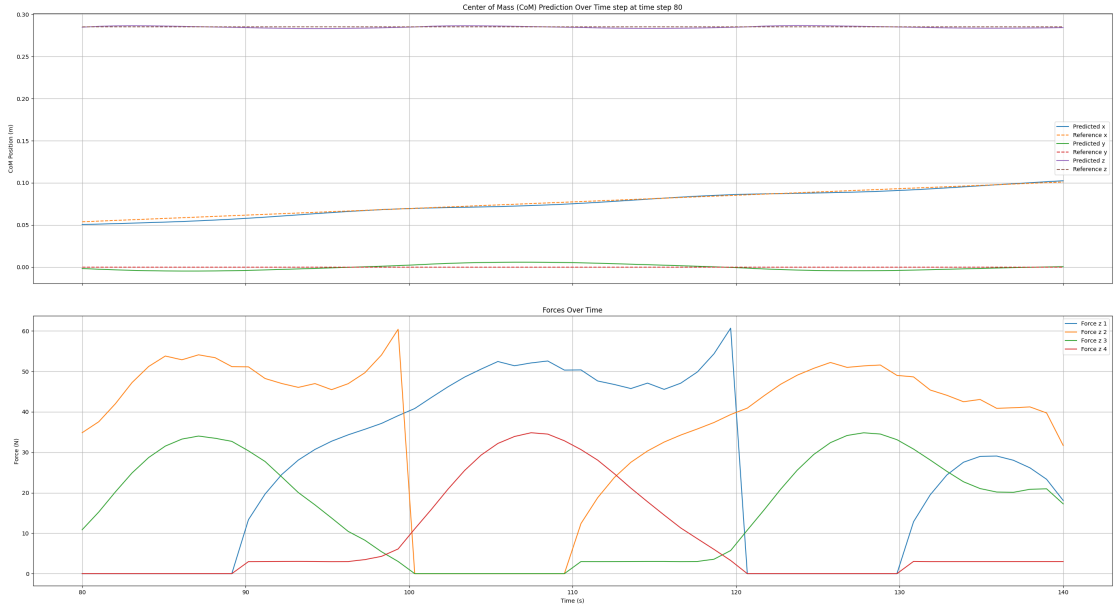


Figure 15: MPC predictions for CoM x, y, z and forces over the prediction horizon at timestep 80.

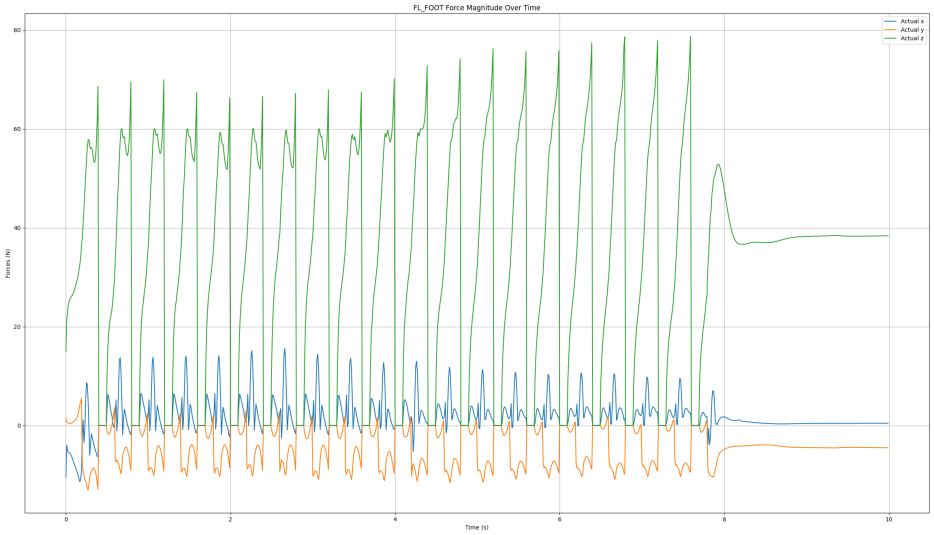


Figure 16: Front Left foot forces for x, y, z coordinates.

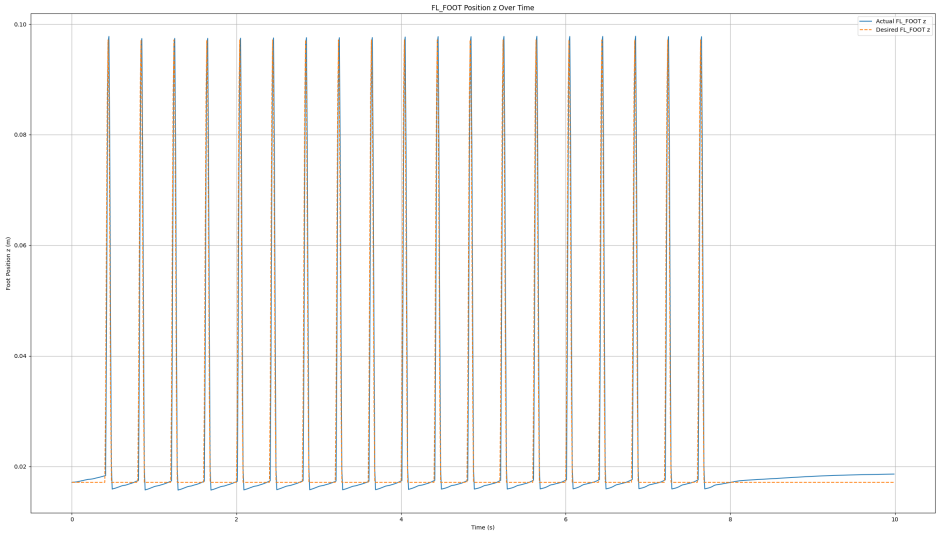


Figure 17: Front Left foot tracking performance for z coordinate.

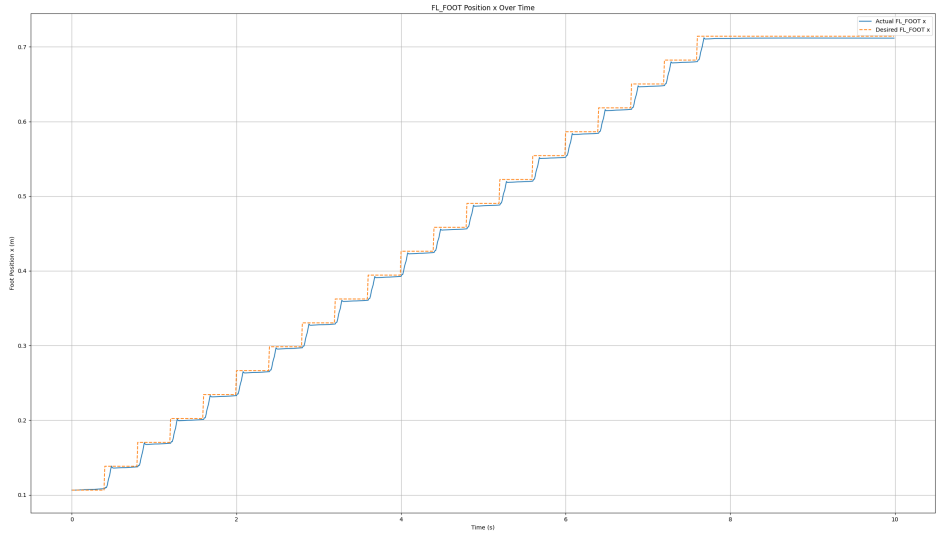


Figure 18: Front Left foot tracking performance for x coordinate.

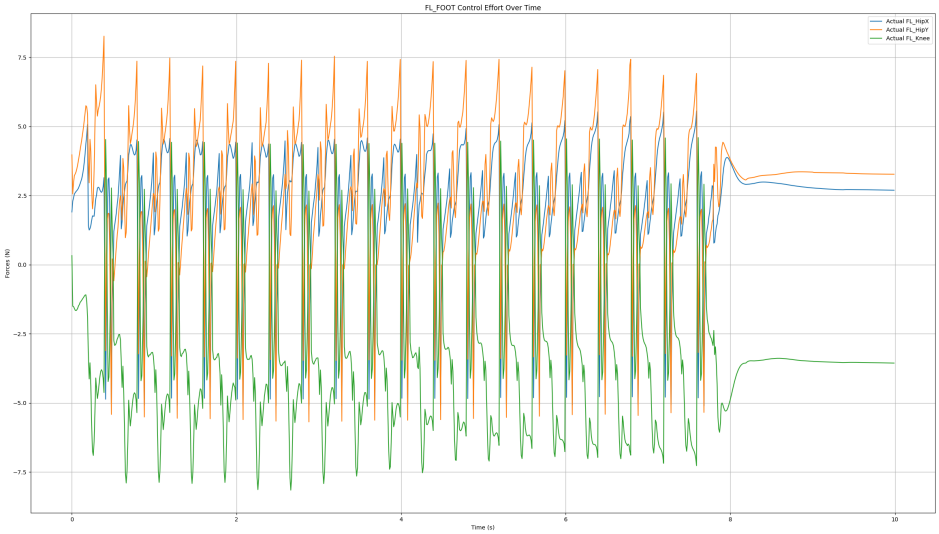


Figure 19: Control effort for the Front Left leg of the trotting gait.

- Pseudo-galloping gait

Parameter	**Value**	**Parameter**	**Value**
ss duration [s]	0.1	h [m]	0.285
ds duration [s]	0.04	μ	1.1
$v_{\text{com\_ref\_x}}$ [m/s]	0.18	N	60
ω_{ref} [rad/s]	0	dt [s]	0.01
steps	28	step_height [m]	0.08

Table 2: Simulation parameters of pseudo-galloping gait

This gait demonstrated the highest robustness with respect to increasing Cartesian velocity, achieving stable locomotion at 0.18 m/s along the x-direction. However, this came at the cost of increased tracking errors in both Cartesian position and orientation (Figure 20 and Figure 21).

Compared to the trotting gait, both the vertical ground reaction forces and the control effort were

higher, reaching up to 80 N along the z-axis and 15 N·m in torque, respectively.

Despite the increased control demands, the swing leg performance remained comparable to that of the trotting gait.

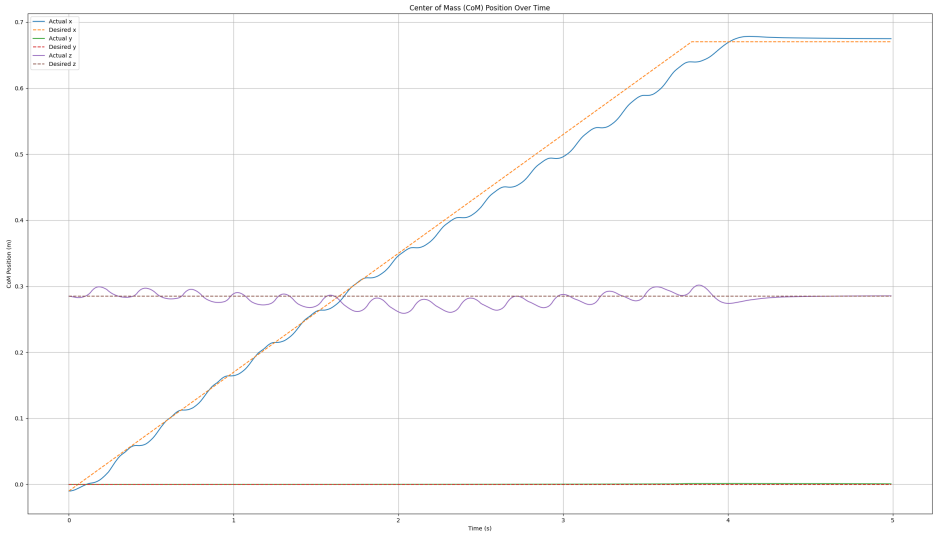


Figure 20: Cartesian tracking performance of the CoM across the simulation duration.

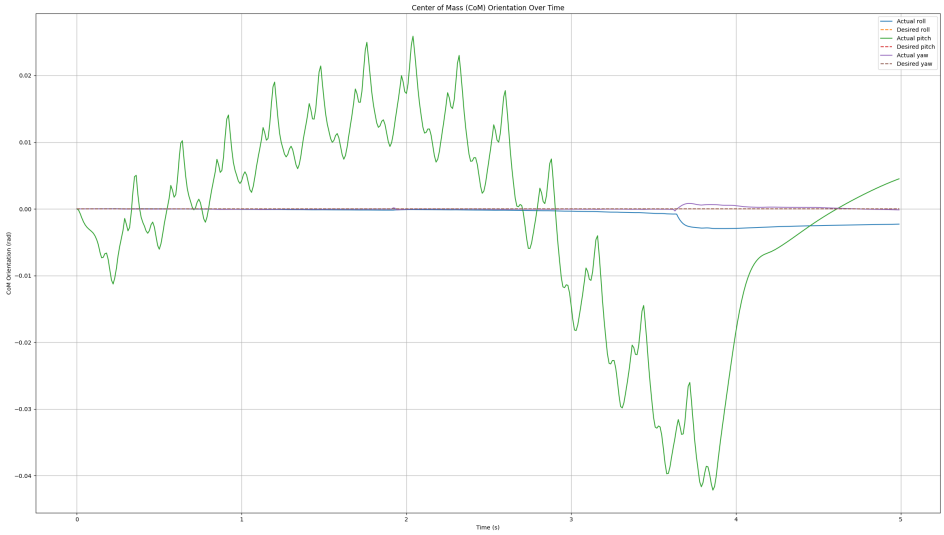


Figure 21: Orientation tracking performance across the simulation duration.

- Ambling gait

Parameter	**Value**	**Parameter**	**Value**
ss duration [s]	0.1	h [m]	0.285
ds duration [s]	0.1	μ	1.5
$v_{\text{com\_ref\_x}}$ [m/s]	0.08	N	60
ω_{ref} [rad/s]	0	dt [s]	0.01
steps	40	step_height [m]	0.08

Table 3: Simulation parameters of ambling gait

Performance was analog to the trotting gait results.

- Pronking gait

Parameter	**Value**	**Parameter**	**Value**
ss duration [s]	0.1	h [m]	0.285
ds duration [s]	0.08	μ	1.5
$v_{\text{com\_ref\_x}}$ [m/s]	0.1	N	60
ω_{ref} [rad/s]	0	dt [s]	0.01
steps	28	step_height [m]	0.08

Table 4: Simulation parameters of pronking gait

The most peculiar gate, shows an expected oscillation in the Cartesian tracking along the z-component around the desired position, due to the jumping (shown in the Figure 22).

The effect of the jump phase is clearly visible in the MPC predictions, both in the vertical ground reaction forces and in the predicted Cartesian z-position (as shown in Figure 23).

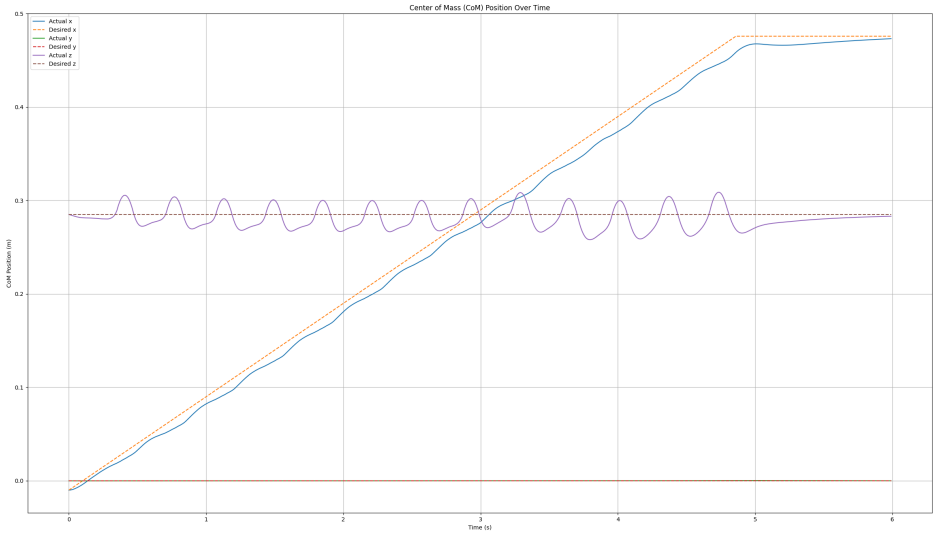


Figure 22: Cartesian tracking performance of the CoM across the simulation duration.

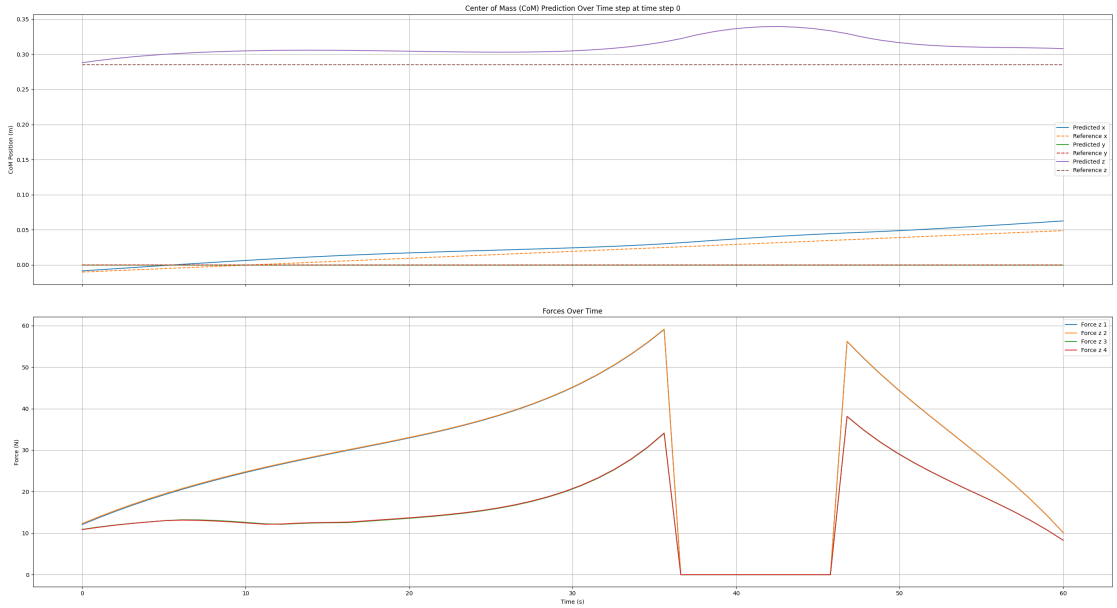


Figure 23: MPC predictions for CoM x, y, z and forces over the prediction horizon at timestep 0.

9 Problems

During the development of the project, several issues were encountered.

Initially, the URDF of the Cheetah 3 robot was implemented. However, due to the absence of a precise reference frame on the robot's feet, the resulting behaviour was unsatisfactory. A different URDF was then adopted, specifically, that of the Lite 3 robot as previously mentioned. This, however, introduced a new issue: all the visual meshes of the feet appeared to be rotated around the y-axis by approximately 40 degrees, resulting in an erroneous analysis of the robot's behaviour. This problem was resolved by manually adjusting the mesh orientation using Blender.

Finally, a major issue arose during the simulations: the robot's feet consistently penetrated the ground, leading to falls. Some instances of these failures are shown in Figures 24 and 25. To address this, the collision spheres associated with the feet were enlarged, and a small mass and inertia were assigned to those collision objects.

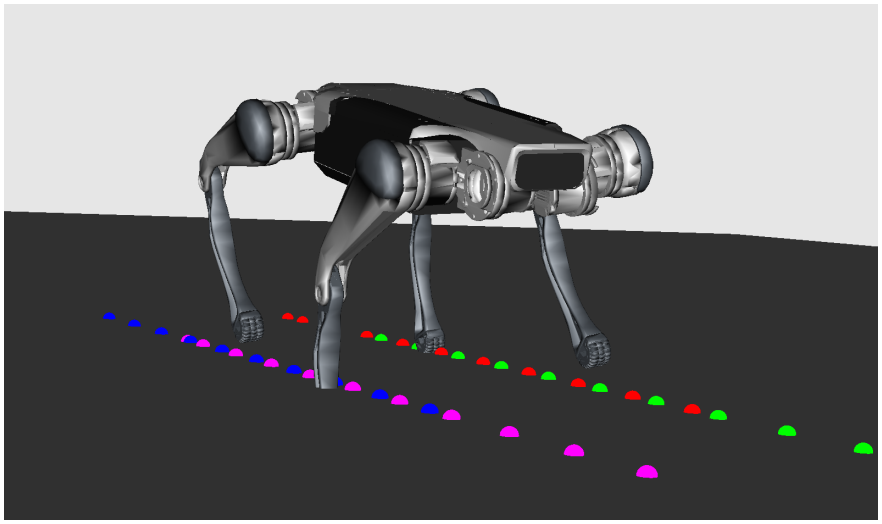


Figure 24: Illustration of ground penetration by the robot's feet, observed during simulation.

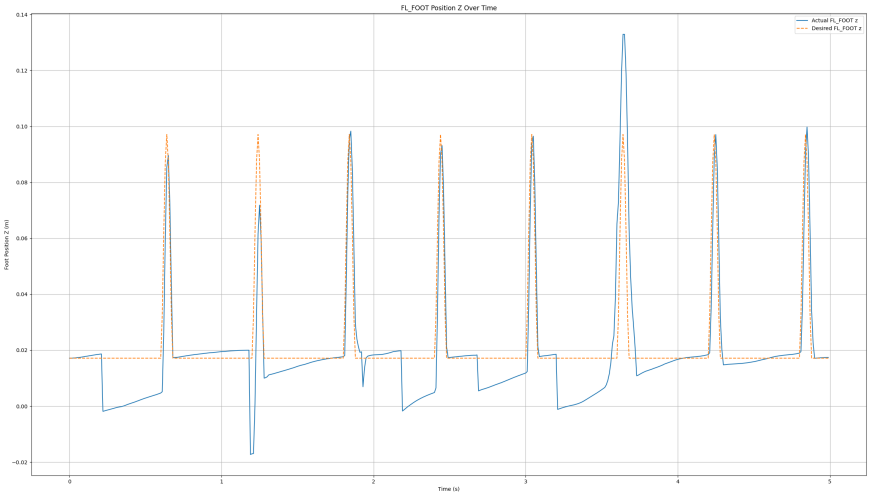


Figure 25: Graph showing the vertical position (z-coordinate) of the foot, highlighting the penetration problem.

10 Conclusion

In this project, a complete locomotion control pipeline for the Lite 3 quadruped robot using Model Predictive Control has been both implemented and validated. The system integrates all key components: a footstep planner, capable of adapting to produce different gaits, a trajectory generator, and both ground and swing leg controllers. Each module was designed specifically for quadrupedal locomotion and tailored to the characteristics of the Lite 3 platform.

Despite facing several challenges mostly related to the simulation environment, such as frequent foot penetrations into the ground, we managed to implement a functioning MPC that demonstrates convergence in the simulation results. These results confirm that the controller can compute valid ground reaction forces that allow the robot to follow the desired trajectories, successfully performing a reasonably smooth gait achieving the main objective of this project.

Additionally, the custom footstep planner and trajectory generator modules produce reasonable outputs that result in a stable locomotion of the Lite 3 robot. The overall framework lays a solid foundation for future real-world deployment and extensions to more complex gait patterns or terrains.

References

[1] J. Di Carlo , P. M. Wensing , B. Katz , G. Bledt , and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control", 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Madrid, Spain, October 1-5, 2018

[2] F. Stark , J. Middelberg , D. Mronga , S. Vyas, F. Kirchner, "Benchmarking Different QP Formulations and Solvers for Dynamic Quadrupedal Walking", arXiv:2502.01329v1 [cs.RO] 3 Feb 2025

[3] Yijie Zhu, "Lite 3 urdf", https://github.com/TopHillRobotics/quadruped-robot/blob/mpc-wbc/quadruped/config/lite3/lite3_robot.yaml,