



SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER CONTROL AND MANAGEMENT
ENGINEERING

Trajectory Generation for Legged Robots Based on a Closed-Form Solution of Centroidal Dynamics

AUTONOMOUS AND MOBILE ROBOTICS

Professor:

Prof. Giuseppe Oriolo

Supervisor:

Nicola Scianca

Students:

Francesco Danese 1926188

Giuseppina Iannotti 1938436

Alessia Pontiggia 1892079

Contents

1	Introduction	3
2	Related Works	4
3	Proposed Method	7
3.1	Stiffness-Based Centroidal Dynamics	7
3.2	Closed-Form Solutions and Discrete-Time Equations	9
3.3	Integration of Base-Link Rotation	10
4	Formulation of Trajectory Optimization Problem	11
4.1	State Equation	11
4.2	Formulation of Optimal Control Problem	12
5	Simulation and Results	16
5.1	Trajectory Generation	16
5.2	Solutions	18
5.3	Still Task: results	18
5.4	Walking Task: results	20

Abstract

Hello World !!

1 Introduction

2 Related Works

Understanding Centroidal Dynamics [?] is essential for controlling humanoid robots. Centroidal Dynamics [?] refers to the dynamics of a robot projected onto its Center of Mass (CoM), which represents the average position of all its mass and serves as the point where the overall gravitational force can be considered to act. Although humanoid robots are complex systems with high-dimensional, nonlinear dynamics, the motion of their Center of Mass (CoM) can often be described using simpler, more intuitive models. Focusing on the CoM allows the global dynamics of the robot to be captured without modeling each joint and link individually, significantly reducing system complexity. This approach provides an effective framework for planning, control, and stability, enabling the development of robots with robust, agile, and human-like behavior.

Thanks to these advantages, many reduced order models and control algorithms have been developed that incorporate the CoM and Centroidal Dynamics as key components. Among the earliest and most influential models is the Linear Inverted Pendulum Model (LIPM) [?], a point-mass approximation that simplifies the robot’s dynamics by making two assumptions: (1) there is no angular momentum about the CoM, and (2) the CoM maintains a constant height. These assumptions allow the otherwise nonlinear centroidal dynamics to be linearized, greatly simplifying trajectory planning. Stability within this framework is maintained by ensuring that the Zero Moment Point (ZMP)—the point where the resultant ground reaction force would act—remains inside the support polygon formed by the robot’s feet.

To extend the capabilities of LIPM, the Variable-Height Linear Inverted Pendulum Model (VH-LIPM) [?] was introduced. This model relaxes the constant height assumption, allowing vertical CoM motion, thereby enabling more dynamic and versatile movements such as stepping on uneven terrain or climbing.

Another important model is the Spring-Loaded Inverted Pendulum (SLIP) [?], which represents the CoM supported by a massless, compliant leg modeled as a spring. Unlike the rigid models, SLIP captures the energy storage and return mechanisms crucial for running, hopping, and other dynamic gaits. The ability of SLIP to express vertical oscillations of the CoM makes it essential for modeling and analyzing dynamic, non-walking locomotion.

These reduced order models have been extensively used in trajectory optimization approaches for motion planning. Trajectory optimization involves defining a motion task through state variables, control inputs, cost functions, and constraints, and then solving for the trajectory that optimizes performance while respecting dynamic feasibility. The reduced models serve as simplified system representations within these optimization problems. For instance, LIPM-based trajectory planners might define the desired ZMP trajectory as a cost term or enforce ZMP constraints to guarantee

stability. More advanced formulations integrate full centroidal dynamics, incorporating both CoM motion and base orientation, with stability constraints expressed through support polygons or centroidal wrench cones.

However, numerical trajectory optimization approaches often require fine time discretization—small time steps—to maintain solution accuracy, leading to a high number of decision variables. Moreover, feasibility is typically ensured only at discrete time nodes, and ensuring feasibility between these nodes remains a challenge.

Integration of Trajectory Optimization If a closed-form solution to the dynamic model is available, it enables trajectory generation using far fewer decision variables while ensuring feasibility at all times, not just at discrete points. Some methods have attempted to derive closed-form or analytical solutions for centroidal dynamics, often through simplifying assumptions such as constant CoM height or neglecting vertical-axis rotation. While these approaches are beneficial for efficiency, they limit the applicability of the resulting motions, particularly for more complex tasks like multi-contact planning or acrobatics. One technique for analytical integration involves assuming constant contact wrenches over finite time intervals (zero-order hold), but this introduces unwanted angular momentum variations unless extremely small time steps are used. To address this limitation, the multi-contact Linear Inverted Pendulum Model (mc-LIPM) was proposed, modeling contact forces through a stiffness-displacement relationship between the CoM and the contact points. This method allows for larger integration steps—potentially spanning entire contact phases—with significant angular momentum disturbances. Nevertheless, while mc-LIPM improves over traditional LIPM approaches, it still lacks full expressiveness for modeling complex rotational dynamics. Despite their success, these reduced order models introduce fundamental limitations. LIPM and VH-LIPM are built on strong assumptions that oversimplify the nonlinear nature of real-world dynamics, restricting their use in highly dynamic or irregular environments. SLIP preserves more physical realism but lacks a closed-form solution, complicating its analytical handling in optimization frameworks. Moreover, trajectory optimization methods based on these models often lead to large-scale optimization problems that are computationally expensive, challenging their application to real-time control.

Limitations These simplifications may be tolerable for basic locomotion tasks such as walking or slow stepping, but they become inadequate for advanced applications like multi-contact maneuvers, dynamic manipulation, or acrobatic motions, where high-fidelity and continuous-time feasibility are crucial. Additionally, the lack of closed-form solutions exacerbates the computational burden, making fast online planning difficult. To overcome these challenges, a novel stiffness-based method is introduced. This approach provides a closed-form solution to the trajectory optimization problem

by modeling the interaction forces through stiffness relationships. It enables the generation of long, dynamically feasible trajectories with far fewer decision variables, offering better computational efficiency and making real-time, complex motion generation more practical and reliable.

3 Proposed Method

3.1 Stiffness-Based Centroidal Dynamics

We begin from the standard *centroidal dynamics* equations, which relate the motion of the robot's center of mass (CoM) to the total external wrench (force and moment) acting on the system:

$$m \ddot{p} = f - m g, \quad (1a)$$

$$\dot{L} = \eta. \quad (1b)$$

Here:

- $p \in \mathbb{R}^3$ is the position of the CoM, and \ddot{p} its acceleration.
- $L \in \mathbb{R}^3$ is the total angular momentum about the CoM.
- $f \in \mathbb{R}^3$ and $\eta \in \mathbb{R}^3$ are, respectively, the translational and rotational components of the total external wrench.
- $m > 0$ is the total mass, and $g \in \mathbb{R}^3$ is the gravity vector (e.g. $g = [0, 0, -9.81]^\top$).

We assume all external wrenches are contact wrenches at n_e end-effectors ("ends") of the robot. Denote by $p_l \in \mathbb{R}^3$ the world-frame position of the l -th end, and let

$$f_l \in \mathbb{R}^3, \quad \eta_l \in \mathbb{R}^3, \quad l = 1, \dots, n_e$$

be the translational and rotational components of the contact wrench at that end. Then the total wrench is

$$f = \sum_{l=1}^{n_e} f_l, \quad \eta = \sum_{l=1}^{n_e} [(p_l - p) \times f_l + \eta_l]. \quad (2)$$

The cross-product in (2) makes the system *bilinear* in (p, f_l) , coupling CoM motion with contact forces.

Spring-like parametrization of contact wrenches. Rather than holding f_l, η_l constant, we introduce a *stiffness* parameter $\lambda_l \geq 0$ and a *CMP-offset* vector $r_l \in \mathbb{R}^3$ for each end, plus a pure-moment direction $\hat{\eta}_l \in \mathbb{R}^3$. Inspired by a spring model, we set:

$$f_l = m \lambda_l^2 (p - (p_l + r_l)), \quad \eta_l = m \lambda_l^2 \hat{\eta}_l. \quad (3)$$

Intuitively:

- λ_l^2 scales like a contact stiffness: larger $\lambda_l \rightarrow$ stronger repulsive force.
- $p_l + r_l$ is the *virtual pivot* (similar to a Centroidal Moment Pivot, CMP). The force pulls the CoM toward that point.
- $\hat{\eta}_l$ encodes any pure moment about the CoM, scaled consistently by λ_l^2 .

Derivation of the stiffness-based model

By substituting (2) and (3) into (1a) and neglecting $O(\epsilon^2)$ terms one obtains

$$\begin{aligned}
m \ddot{p} &= \sum_{l=1}^{n_e} m \lambda_l^2 (p - (p_l + r_l)) - m g \\
&\approx \sum_{l=1}^{n_e} m \lambda_l^2 (p - (p_l + r_l)) - m g + m \epsilon^2 p \\
&= m \left(\sum_l \lambda_l^2 + \epsilon^2 \right) p - m \left(\sum_l \lambda_l^2 (p_l + r_l) + g \right) \\
&= m \bar{\lambda}^2 (p - \bar{p} - \bar{r}),
\end{aligned} \tag{4}$$

where

$$\bar{\lambda}^2 = \sum_l \lambda_l^2 + \epsilon^2, \quad \bar{p} = \frac{\sum_l \lambda_l^2 p_l + g}{\bar{\lambda}^2}, \quad \bar{r} = \frac{\sum_l \lambda_l^2 r_l}{\bar{\lambda}^2}.$$

Similarly, substituting into (1b) gives

$$\begin{aligned}
\dot{L} &= \sum_{l=1}^{n_e} [(p_l - p) \times m \lambda_l^2 (p - p_l - r_l) + m \lambda_l^2 \hat{\eta}_l] \\
&= \sum_l [(p - p_l) \times m \lambda_l^2 r_l] + \sum_l m \lambda_l^2 \hat{\eta}_l \\
&= p \times m \bar{\lambda}^2 \bar{r} + \sum_l m \lambda_l^2 (\hat{\eta}_l - p_l \times r_l) \\
&\approx (m \ddot{p} + m \bar{\lambda}^2 (\bar{p} + \bar{r})) \times \bar{r} + \sum_l m \lambda_l^2 (\hat{\eta}_l - p_l \times r_l) \\
&= m (\ddot{p} \times \bar{r} + \bar{\eta}),
\end{aligned} \tag{5}$$

where

$$\bar{\eta} = \bar{\lambda}^2 (\bar{p} \times \bar{r}) + \sum_l \lambda_l^2 (\hat{\eta}_l - p_l \times r_l).$$

Closed-form centroidal dynamics. Combining (4) and (5) yields the stiffness-based centroidal equations:

$$\ddot{p} = \bar{\lambda}^2 (p - (\bar{p} + \bar{r})), \tag{6a}$$

$$\dot{L} = m (\ddot{p} \times \bar{r} + \bar{\eta}). \tag{6b}$$

The aggregated parameters $\bar{\lambda}, \bar{p}, \bar{r}, \bar{\eta}$ recover the same expressions as before.

Discussion and special cases.

Remark 1 (Exactness vs. flight phase). If one ignores flight (i.e. always in contact, $\sum_l \lambda_l^2 > 0$), one may set $\epsilon = 0$ and (6) hold exactly. Otherwise $\epsilon > 0$ guarantees a well-defined $\bar{\lambda}$ in airborne phases.

Remark 2 (Ballistic motion). When all ends lose contact ($\lambda_l = 0$ for all l), one finds

$$\ddot{p} = \epsilon^2 p - g \approx -g, \quad \dot{L} = 0,$$

recovering the usual ballistic CoM motion and conservation of angular momentum.

Remark 3 (Relation to existing models). Stiffness-based (or force-to-point) parametrization has appeared before, but typically only at the *total* wrench level. Here we assign a separate $\lambda_l, r_l, \hat{\eta}_l$ to each end, which yields a unified multi-contact description. The classical CoP and (e)CMP emerge naturally as \bar{p} and $\bar{p} + \bar{r}$, respectively.

3.2 Closed-Form Solutions and Discrete-Time Equations

We subdivide the time horizon $[0, T]$ into N consecutive intervals

$$[t_k, t_{k+1}], \quad k = 0, 1, \dots, N-1, \quad t_{k+1} = t_k + \tau_k.$$

We assume that *contact states* (i.e. which ends are in contact) change only at the boundaries t_k . Moreover, we apply a *zero-order hold* on the stiffness-based parameters

$$\{\lambda_l(t), r_l(t), \hat{\eta}_l(t)\} \mapsto \{\lambda_{l,k}, r_{l,k}, \hat{\eta}_{l,k}\} \quad \text{for } t \in [t_k, t_{k+1}),$$

meaning that each parameter is held constant over the interval.

State at the beginning of interval k . Let

$$p_k = p(t_k), \quad v_k = \dot{p}(t_k), \quad L_k = L(t_k),$$

and compute the aggregated quantities

$$\begin{aligned} \bar{\lambda}_k &= \sqrt{\sum_{l=1}^{n_e} \lambda_{l,k}^2 + \epsilon^2}, \quad \bar{p}_k = \frac{\sum_{l=1}^{n_e} \lambda_{l,k}^2 p_{l,k} + g}{\bar{\lambda}_k^2}, \quad \bar{r}_k = \frac{\sum_{l=1}^{n_e} \lambda_{l,k}^2 r_{l,k}}{\bar{\lambda}_k^2}, \\ \bar{\eta}_k &= \bar{\lambda}_k^2 (\bar{p}_k \times \bar{r}_k) + \sum_{l=1}^{n_e} \lambda_{l,k}^2 (\hat{\eta}_{l,k} - p_{l,k} \times r_{l,k}). \end{aligned}$$

Analytical solution on $[t_k, t_{k+1}]$. With $\bar{\lambda}_k, \bar{p}_k, \bar{r}_k, \bar{\eta}_k$ constant, the CoM-dynamics

$$\ddot{p} = \bar{\lambda}_k^2 (p - (\bar{p}_k + \bar{r}_k))$$

is a linear second-order ODE whose homogeneous+particular solution reads

$$p(t) = (\bar{p}_k + \bar{r}_k) + C_k(\Delta t) (p_k - (\bar{p}_k + \bar{r}_k)) + \frac{S_k(\Delta t)}{\bar{\lambda}_k} v_k, \quad (7a)$$

$$v(t) = \dot{p}(t) = \bar{\lambda}_k S_k(\Delta t) (p_k - (\bar{p}_k + \bar{r}_k)) + C_k(\Delta t) v_k, \quad (7b)$$

where $\Delta t = t - t_k$ and

$$C_k(\Delta t) = \cosh(\bar{\lambda}_k \Delta t), \quad S_k(\Delta t) = \sinh(\bar{\lambda}_k \Delta t).$$

Finally, substituting into the angular-momentum equation

$$\dot{L} = m(\ddot{p} \times \bar{r}_k + \bar{\eta}_k)$$

and integrating from t_k to t gives

$$L(t) = L_k + m((v(t) - v_k) \times \bar{r}_k + (t - t_k) \bar{\eta}_k). \quad (7c)$$

Remark 4 (Zero-Order Hold). A zero-order hold means we approximate time-varying parameters by piecewise-constant values on each interval. This yields closed-form expressions above, at the cost of not capturing high-frequency parameter variations.

3.3 Integration of Base-Link Rotation

The centroidal state (p, \dot{p}, L) does not specify the *orientation* $R(t) \in SO(3)$ or the *base-link* angular velocity $\omega(t) \in \mathbb{R}^3$. In a multi-body system one shows

$$L = \underbrace{RI R^\top}_{I_{\text{sys}}(R)} \omega + R \hat{L}, \quad (8)$$

where

- $I \in \mathbb{R}^{3 \times 3}$ is the composite inertia in the base-link frame,
- \hat{L} is the angular momentum about the base link due to internal motions.

If we fix reference values I_{ref} , \hat{L}_{ref} (e.g. from a nominal whole-body motion), we solve for

$$\omega(t) = I_{\text{sys}}(R)^{-1}(L - R \hat{L}_{\text{ref}}) \approx R I_{\text{ref}}^{-1}(R^\top L - \hat{L}_{\text{ref}}). \quad (9)$$

Discrete quaternion update. Let $q_k \in \mathbb{H}$ be the unit-quaternion representing $R(t_k)$. Over $[t_k, t_{k+1}]$ we subdivide into n_{div} equal steps

$$t_k = t'_0 < \dots < t'_i < \dots < t'_{n_{\text{div}}} = t_{k+1}, \quad \tau'_k = \frac{\tau_k}{n_{\text{div}}}, \quad t'_i = t_k + i \tau'_k.$$

At each substep we assume ω nearly constant and update

$$q_{k+1} = \underbrace{q(\omega(t'_{n_{\text{div}}}) \tau'_k)}_{\text{quat. for small rotation}} \cdots q(\omega(t'_1) \tau'_k) \cdot q(\omega(t'_0) \tau'_k) \cdot q_k, \quad (10)$$

where $q(\theta)$ is the unit quaternion corresponding to the axis-angle $\theta \in \mathbb{R}^3$. The designer chooses n_{div} to balance integration accuracy against computational cost of gradient evaluation in trajectory optimization.

4 Formulation of Trajectory Optimization Problem

4.1 State Equation

To mathematically describe the evolution of the system over discrete time intervals, we define at each time step k :

- a state vector \mathbf{x}_k , containing all the information necessary to describe the system's configuration and motion;
- a control input vector u_k , based on the stiffness-based control strategy introduced in Section 3.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{q}_k \\ \mathbf{v}_k \\ \mathbf{L}_k \\ t_k \\ \{\mathbf{p}_{l,k}\}_{l=1,\dots,n_e} \\ \{\mathbf{q}_{l,k}\}_{l=1,\dots,n_e} \end{bmatrix} \quad (7)$$

$$u_k = \begin{bmatrix} \tau_k \\ \{\mathbf{v}_{l,k}\}_{l=1}^{n_e} \\ \{\boldsymbol{\omega}_{l,k}\}_{l=1}^{n_e} \\ \{\boldsymbol{\lambda}_{l,k}\}_{l=1}^{n_e} \\ \{\mathbf{r}_{l,k}\}_{l=1}^{n_e} \\ \{\boldsymbol{\eta}_{l,k}\}_{l=1}^{n_e} \end{bmatrix} \quad (8)$$

Each component of the state and input vectors has been introduced and detailed in Section 3.

This formulation enables explicit enforcement of physical constraints such as contact complementarity by treating velocities as control variables. For instance, assigning a high cost to velocities at inactive contacts discourages motion in those regions, effectively ensuring that contacts are only active when physically meaningful.

Equations Update The update to the timestamp is given by:

$$t_{k+1} = t_k + \tau_k \quad (9)$$

The state update for each end-effector's position and orientation is governed by the basic kinematic relations:

$$\mathbf{p}_{l,k+1} = \mathbf{p}_{l,k} + \mathbf{v}_{l,k} \tau_k \quad (10)$$

$$\mathbf{q}_{l,k+1} = q(\boldsymbol{\omega}_{l,k}, \tau_k) \cdot \mathbf{q}_k \quad (11)$$

Here, $q(\boldsymbol{\omega}_{l,k}, \tau_k)$ represents the quaternion corresponding to the angular displacement over the time interval τ_k , computed from the angular velocity $\boldsymbol{\omega}_{l,k}$. Integrating all these elements, the complete system dynamics are compactly expressed as a state transition function:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \quad (12)$$

This function encapsulates both the kinematics and dynamics of the system, allowing for predictive simulation or optimization over multiple time steps.

4.2 Formulation of Optimal Control Problem

The cost function is a crucial component of the trajectory optimization problem, as it quantifies the performance of a given trajectory. The cost function is typically composed of several terms, each representing different aspects of the system’s performance.

Task Related Cost In trajectory tracking tasks, it is important for the system to follow a planned or reference trajectory as closely as possible. The *task-related cost* measures how much the system’s current state and inputs deviate from their desired (reference) values. Minimizing this cost ensures the system stays close to the intended path during motion. The task-related cost function is formulated as:

$$L_{\text{task},k} = \frac{1}{2} \|W_k^x(\mathbf{x}_k - \mathbf{x}_k^{\text{ref}})\|^2 + \frac{1}{2} \|W_k^u(\mathbf{u}_k - \mathbf{u}_k^{\text{ref}})\|^2 \quad (13)$$

where $(*)^{\text{ref}}$ represents the reference (target) values for the state x_k and the control input u_k . In this work, we focus on a waypoint-tracking task, where a series of intermediate waypoints are specified for the center of mass (CoM), base link, and limb endpoints. Desired positions and velocities along a smooth path connecting these waypoints are generated using spline curves. The system’s stiffness parameters are computed by solving the following least-squares optimization problem at each time step k :

$$\min \left\| \sum_l \lambda_{l,k}^2 \right\|^2 \quad \text{subject to} \quad \sum_l \lambda_{l,k}^2 (\mathbf{p}_k^{\text{ref}} - \mathbf{p}_{l,k}^{\text{ref}}) = \mathbf{g} \quad (14)$$

This optimization distributes stiffness values to maintain support for the CoM against gravity. In addition, the reference values for the Centroidal Moment Pivot (CMP) offset and the moments at the end effectors are typically set to zero unless non-zero values are specifically chosen to induce desired dynamic effects. The weighting matrices W_k^x and W_k^u are design parameters that control the importance given to state and input deviations in the cost function. Finally, when dealing with rotational variables represented by quaternions, the deviation between the actual and reference orientations is calculated by transforming the quaternion difference into an equivalent angle-axis representation:

$$\mathbf{q} - \mathbf{q}^{\text{ref}} := \omega(\mathbf{q}^{\text{ref}^{-1}} \mathbf{q}) \quad (15)$$

where $\omega(\cdot)$ maps a unit quaternion into an angle-axis vector.

Inequality Constraints In physical systems involving contact dynamics, it is essential to ensure that certain physical conditions—such as feasible positions, contact forces, and proper stiffness—are always satisfied. These conditions are formulated as *inequality constraints* and play a key role in maintaining the physical realism of the motion and feasibility of the optimization problem. In this framework, inequality constraints are

used to regulate the position of each contact point relative to the CoM and the base link, enforce non-slipping conditions, and bound parameters like phase duration and stiffness values. First, the box constraint on the position of each end link relative to the CoM and the base link is expressed as:

$$\mathbf{p}_{l,\min} \leq \mathbf{q}^{-1}(\mathbf{p}_l - \mathbf{p}) \leq \mathbf{p}_{l,\max}, \quad (16)$$

where \mathbf{p} is the CoM position, \mathbf{p}_l is the position of the l -th end, and \mathbf{q}^{-1} denotes the inverse transformation to the local frame. Simple range constraints are imposed on the duration of each phase and the stiffness values:

$$\tau_{\min} \leq \tau \leq \tau_{\max}, \quad (17)$$

$$0 \leq \lambda_l \leq \lambda_{\max}, \quad \forall l. \quad (18)$$

Next, for each end in contact, the contact wrench must satisfy non-slip and moment conditions. If we want to avoid relative motion of the contact surfaces, we must ensure sufficient friction. Thus, to achieve no slipping, the force must be within the friction cone, i.e., the tangential force f_t must satisfy

$$|f_t| \leq \mu f_n \implies \sqrt{f_{l,x}^2 + f_{l,y}^2} \leq \mu f_{l,z} \quad (19)$$

where μ is the static friction coefficient, and $(f_{l,x}, f_{l,y}, f_{l,z})$ are the force components at the contact. Constraints on the moments at the contact point are given as:

$$-c_{\max,x} f_{l,z} \leq \eta_{l,x} \leq c_{\max,x} f_{l,z}, \quad (20)$$

$$c_{\min,y} f_{l,z} \leq \eta_{l,y} \leq c_{\max,y} f_{l,z}, \quad (21)$$

$$-\mu_z f_{l,z} \leq \eta_{l,z} \leq \mu_z f_{l,z}, \quad (22)$$

where c_{\min} and c_{\max} define the rectangular bounds of the center-of-pressure (CoP) region, and μ_z is the friction coefficient for the moment. All the inequality constraints can be compactly represented as:

$$g(\mathbf{x}_k, \mathbf{u}_k) \geq 0, \quad (23)$$

where $g(\cdot)$ is a differentiable vector-valued function, evaluated componentwise. To handle these constraints during optimization, a log-barrier function is introduced:

$$L_{\text{limit}}(\mathbf{x}_k, \mathbf{u}_k) = \sum_{i=1}^{n_g} -\log \max(\epsilon, g_i(\mathbf{x}_k, \mathbf{u}_k)), \quad (24)$$

where n_g is the number of constraints, g_i is the i -th constraint function, and ϵ is a small positive constant that ensures numerical stability and prevents the log function from becoming undefined.

Contact Dependent Cost Modeling the interaction between a robot’s ends and the environment is crucial in dynamic contact systems. The contact-dependent cost is introduced to enforce the complementarity between the contact forces, end velocities, and stiffness values. This ensures that if an end effector is in contact, its motion and interaction forces behave physically, and if it is not in contact, unnecessary forces or stiffness values are suppressed. By minimizing these costs, the system ensures physically meaningful transitions between contact and non-contact states throughout the trajectory.

The contact-dependent cost is defined as:

$$J_{\text{compl},k} = w_{\text{compl}}^2 \sum_l \left(\underbrace{\sum_i \delta[\sigma_{l,k} = i] (\boldsymbol{\eta}_i^\top (\mathbf{p}_{l,k} - \mathbf{o}_i))^2}_{\text{contact distance constraint}} \right. \\ \left. + \underbrace{\delta[\sigma_{l,k} \neq \emptyset] (\|\mathbf{v}_{l,k}\|^2 + \|\boldsymbol{\omega}_{l,k}\|^2)}_{\text{zero velocity constraint}} \right. \\ \left. + \underbrace{\delta[\sigma_{l,k} = \emptyset] \lambda_{l,k}^2}_{\text{zero stiffness constraint}} \right) \quad (25)$$

where $\sigma_{l,k}$ denotes the contact state of the l -th end at time step k . In particular,

- $\sigma_{l,k} = i$ if the l -th end is in contact with the i -th contact surface.
- $\sigma_{l,k} = \emptyset$ if it is not in contact.

The operator $\delta[*]$ is an indicator function that returns 1 if the condition inside the brackets is true, and 0 otherwise.

Each term inside the cost function has a specific physical meaning:

- The first term enforces that if the l -th end is in contact with surface i , namely the distance in the normal direction (defined by normal vector $\boldsymbol{\eta}_i$) from the end’s position $\mathbf{p}_{l,k}$ to the contact surface origin \mathbf{o}_i must be zero.
- The second term requires that if the l -th end is in contact with any surface, its linear velocity $\mathbf{v}_{l,k}$ and angular velocity $\boldsymbol{\omega}_{l,k}$ must also be zero, representing static contact.
- The third term ensures that if the l -th end is not in contact, the associated stiffness $\lambda_{l,k}$ must be zero. Physically, this prevents generating unnecessary contact forces when there is no actual contact.

By properly tuning the weight parameter w_{compl} , these complementarity errors can be made acceptably small after optimization. A sufficiently large value of w_{compl} is necessary to enforce these physical consistency conditions without overly penalizing the overall optimization performance.

Final Cost Function and Problem Formulation After defining the task-related, limit-related, and contact-dependent costs, the overall cost function is constructed by summing these individual terms over all time steps. This aggregated cost captures all important objectives and physical constraints of the motion planning task.

The overall cost function is defined as:

$$J[\boldsymbol{\sigma}] = \sum_k [L_{\text{task},k} + L_{\text{limit},k} + L_{\text{compl},k}[\boldsymbol{\sigma}_k]] \quad (26)$$

The planning problem can then be formulated as the following optimal control problem:

$$\begin{aligned} & \text{find } \mathbf{x}, \mathbf{u} \text{ that minimizes } J[\boldsymbol{\sigma}](\mathbf{x}, \mathbf{u}) \\ & \text{subject to } \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \quad (27)$$

Here, x and u denote the state and control input trajectories, respectively. The function $f(\mathbf{x}_k, \mathbf{u}_k)$ represents the discrete-time system dynamics, ensuring that the state evolution is dynamically feasible.

5 Simulation and Results

This section aims to visualize the results obtained with the proposed method. In the following, there is first defined the trajectory used for the task reference. Then, there are shown plots, tables and simulations. Before defining the trajectories implemented, let's denote with:

- 0, when the foot is in contact with the surface
- -, when the foot is off contact with the surface

The trajectories are defined with respect to contact point, not time.

5.1 Trajectory Generation

Still Task

In this task the robot should not move at any instant of time and thus keeping the same initial position. The contact sequence used is reported in Table ?? where the first row concerns the right foot and the second row the left foot.

Task	N	Contact Sequence
Still	4	0 0 0 0 0 0

Table 1: Contact sequence for still task

The trajectory is simply defined as follows:

Algorithm 1 Reference Trajectory Initialization and Update

- 1: Initialize $X_{\text{ref}} \in \mathbb{R}^{28 \times (N+1)}$, $U_{\text{ref}} \in \mathbb{R}^{27 \times N}$
 - 2: Set initial CoM state, feet positions and orientations in X_{ref}
 - 3: time $\leftarrow 0$
 - 4: **for** $t = 0$ to $N - 1$ **do**
 - 5: Read current contact state from σ
 - 6: Set phase duration and contact force gains λ
 - 7: Update U_{ref} with the above
 - 8: Set CoM and feet position, velocities and orientation as the initial state values
 - 9: time = time + phase duration
 - 10: Update $X_{\text{ref}}(t + 1)$ with the above
 - 11: **end for**
 - 12: **return** $X_{\text{ref}}, U_{\text{ref}}$
-

Walking Task

For the walking task, the trajectory used relies on the following concepts: during the double-contact phases, in which both feet are on the ground, namely [0, 0], the COM does not move forward. Instead, it moves forward (along the X axis) only during single-support phases (a step), namely when right/left foot is lifting ([-, 0] or [0, -] respectively). Therefore if the preceding phase is a double support, the current phase COM keeps the same position as before, while if the preceding phase is of single support, the current phase COM has the previous position + some displacement; The Z (height) of each foot is always zero since at the beginning of each phase the feet are on the ground, while the X increases in an alternating manner depending on which foot was lifted in the previous phase. Also, the feet moves simulating a real walk, where the foot that is moving "surpasses" the one in contact, touching the ground beyond it. Lastly, each foot is interpolated in the intra-phase height position by following a parabolic profile that starts and arrives at the corresponding positions given by the phases solutions, instead of using the fixed inputs (zero-hold). The contact sequence used is reported in Table ?? where the first row concerns the right foot and the second row the left foot.

Task	N	Contact Sequence
Walk	24	000-000-000-000-000-0 0-000-000-000-000-000

Table 2: Contact sequence for walking task

The algorithm used for this task is shown below:

Algorithm 2 Reference Trajectory Initialization and Update

```
1: Initialize  $X_{\text{ref}} \in \mathbb{R}^{28 \times (N+1)}$ ,  $U_{\text{ref}} \in \mathbb{R}^{27 \times N}$ 
2: Set initial CoM state, feet positions and orientations in  $X_{\text{ref}}$ 
3: time  $\leftarrow 0$ , sig_idx  $\leftarrow 0$ 
4: for  $t = 0$  to  $N - 1$  do
5:   Read contact states from  $\sigma$ 
6:   Set phase duration and contact force gains  $\lambda$ 
7:   Update  $U_{\text{ref}}$  with phase duration and  $\lambda$ 
8:   sig_idx  $\leftarrow$  sig_idx + 2
9: end for
10: for  $t = 1$  to  $N$  do
11:   Read current and previous contact states
12:   Set CoM velocity  $(v_x, v_y)$  and feet step velocities
13:   Update CoM and feet positions based on velocity and duration
14:   Update time and orientations in  $X_{\text{ref}}$ 
15:   sig_idx  $\leftarrow$  sig_idx + 2
16: end for
17: return  $X_{\text{ref}}, U_{\text{ref}}$ 
```

5.2 Solutions

Solutions of those problems and tasks are tuples (X, U) where X is the state vector with dimension N , and U is the input vector with dimension $N-1$.

The solution is computed with ipopt solver from Casadi optimizer. Both the reference and solution are contact-dependent, meaning that X and U contain values for each contact phase, thus to find the time dependent solution we used the dynamics.

5.3 Still Task: results

In this case the state vector has dimension 3, and the input vector has dimension 2.

CoM Plots

The following plots report the CoM trajectory in time. The alternation between light and dark grey on the background suggests the switch between phases. As plots suggest there's no significant motion in any direction.

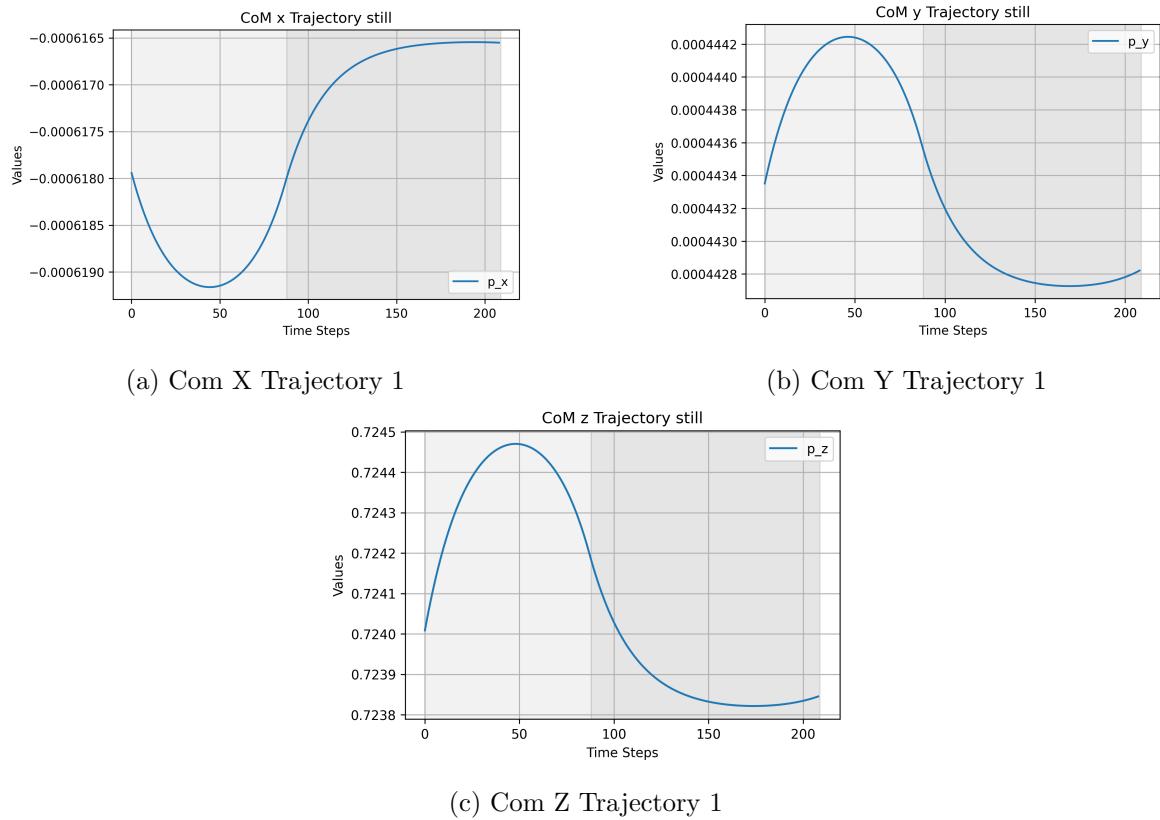


Figure 1: Com Trajectory

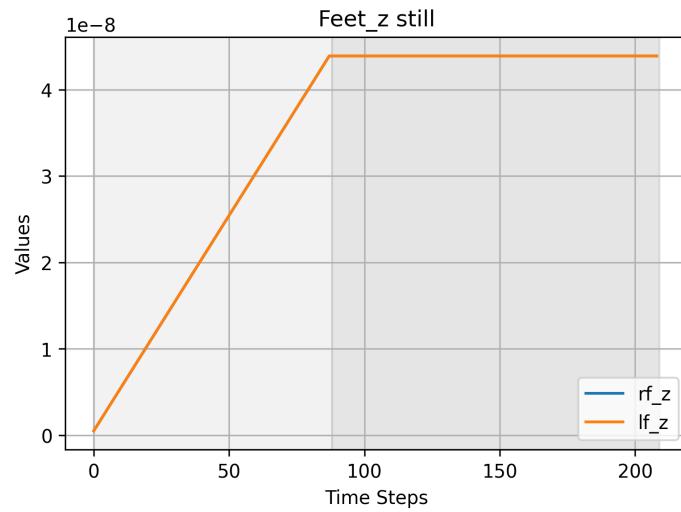


Figure 2: Feet along Z (scale is 1e-8)

State and Input Contact Values

In the following plots, the main components of the state and input vectors solutions are compared with the reference at each contact step.

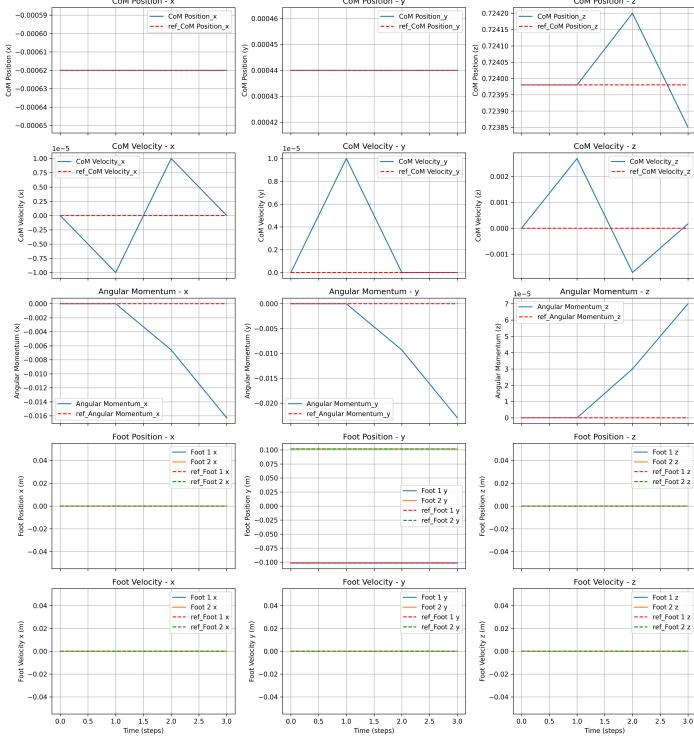


Figure 3: Trajectory vs Reference: state vector

Contact Forces

Another important dynamic aspect regards forces. In the input vector, the contact wrench that describes all the mechanical influence that a contact point (like a foot or hand) exerts on the robot or vice versa. To balance dynamic laws, along the Z-axis the environment should exert a reaction force equal to the gravity factor multiplied by the mass of the robot. In the table ?? below there are listed the contact forces exerting from the environment to both feet. As the table shows, when both feet are on the ground the gravity force is equally distributed on left and right foot.

Right Foot Z	Left Foot Z	Σ_L^k
49.0644	49.0531	[0., 0.]
48.8976	49.9925	[0., 0.]
48.0847	49.0895	[0., 0.]

Table 3: Z-axis gravity force and Σ_L^k values

Components of contact wrench, rotational and translational forces, are graphically expressed in the following plots:

5.4 Walking Task: results

In this case the state vector has dimension 24 and the input vector has dimension 23.

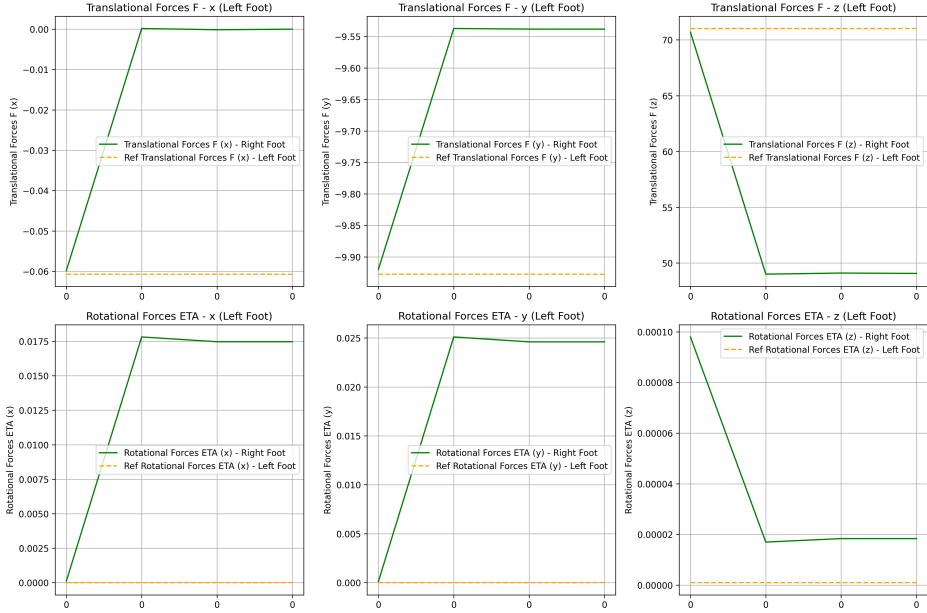


Figure 4: Trajectory vs Reference: Forces

CoM Plots

The following plots report the CoM trajectory in time. The alternation between light and dark grey on the background suggests the switch between phases. As Fig. ?? shows, the CoM exhibits linear motion in the x-direction. Along the y-axis ?? the CoM goes towards the foot staying on the ground when lifting the other. Let's consider the first two contacts: in the first one, the robot is in double support (both feet are on the ground), while the second one expects the left foot to raise. Infact, the CoM at the end of the first phase has moved towards the right foot. This mechanism is repeated through all the alternances between double support and single support. In the z-direction ??, the CoM goes up and down in a small range of motion.

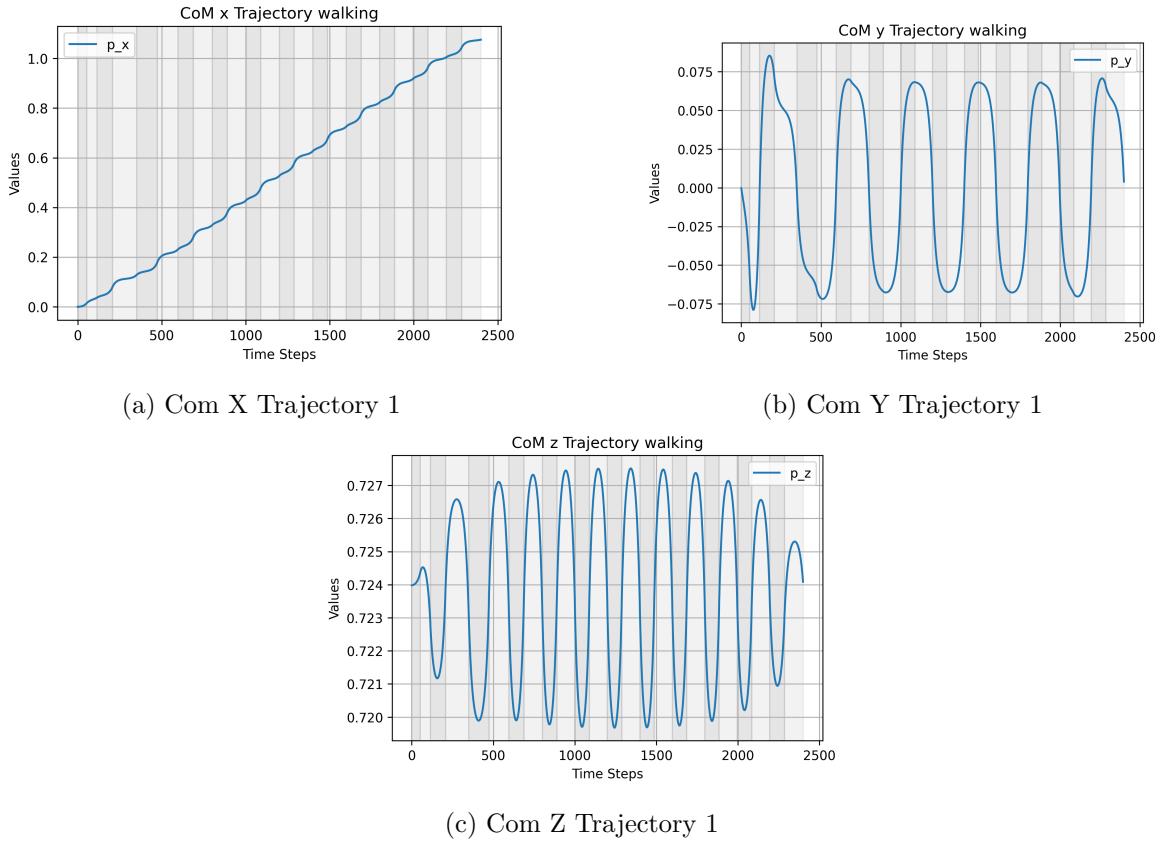


Figure 5: Com Trajectory

Furthermore, the image below represent the foot position along the Z-axis in time. As one can see, they follow a parabolic profile.

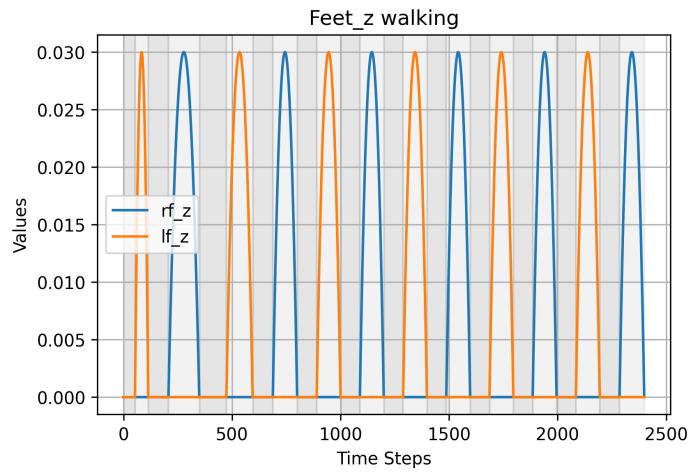


Figure 6: Feet along Z

State and Input Contact Values

In the following plots, the main components of the state and input vectors solutions are compared with the reference at each contact step.

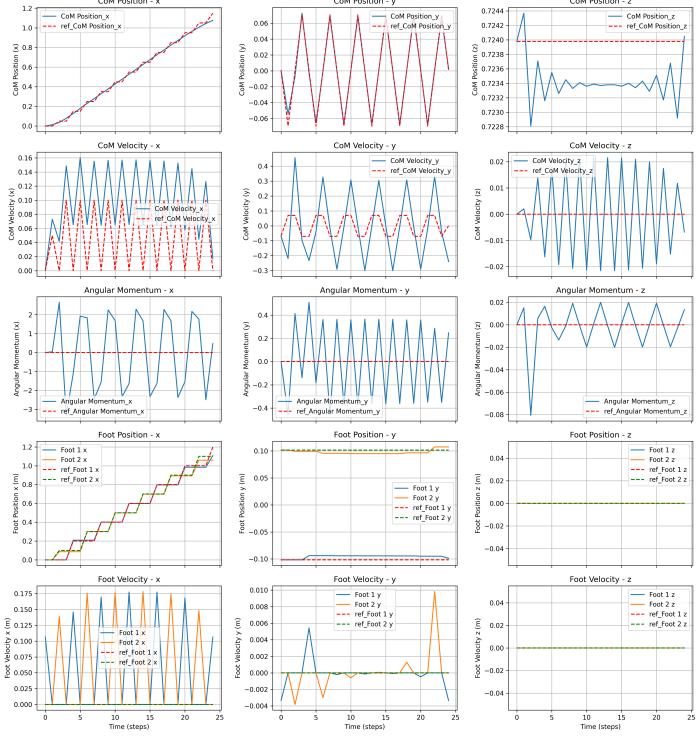


Figure 7: Trajectory vs Reference: state vector

Contact Forces

Another important dynamic aspect regards forces. In the input vector, the contact wrench that describes all the mechanical influence that a contact point (like a foot or hand) exerts on the robot or vice versa. To balance dynamic laws, along the Z-axis the environment should exert a reaction force equal to the gravity factor multiplied by the mass of the robot. In the table ?? below there are listed the contact forces exerting from the environment to both feet. As the table shows, when both feet are on the ground the gravity force is equally distributed on left and right foot, while when one foot lifts the gravity force acts only on the foot staying on the ground.

Right Foot Z	Left Foot Z	Σ_L^k
49.0644	49.0531	[0., 0.]
97.9551	0	[0., -]
48.8973	49.65	[0., 0.]
0	97.4904	[-, 0.]
49.6528	49.1563	[0., 0.]
97.3081	0	[0., -]
49.2399	49.7241	[0., 0.]
0	97.2091	[-, 0.]
49.7388	49.293	[0., 0.]
97.1669	0	[0., -]
49.3007	49.7613	[0., 0.]
0	97.1486	[-, 0.]
49.762	49.31	[0., 0.]
97.1443	0	[0., -]
49.3062	49.7655	[0., 0.]
0	97.1495	[-, 0.]
49.7557	49.3045	[0., 0.]
97.1685	0	[0., -]
49.2815	49.7467	[0., 0.]
0	97.216	[-, 0.]
49.7003	49.254	[0., 0.]
97.3264	0	[0., -]
49.1286	49.6485	[0., 0.]
0	97.5828	[-, 0.]

Table 4: Z-axis gravity force and Σ_L^k values

Components of contact wrench, rotational and translational forces, are graphically expressed in the following plots:

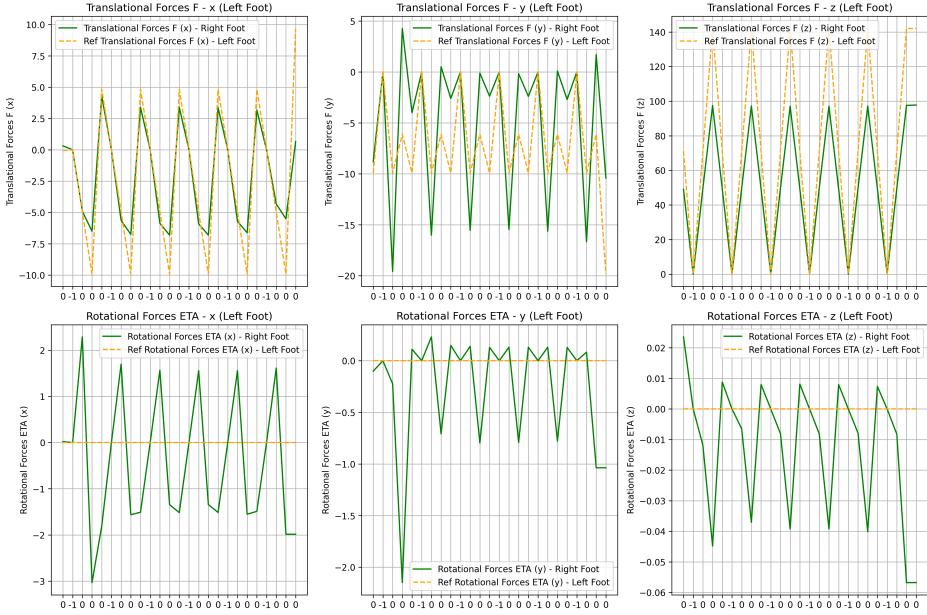


Figure 8: Trajectory vs Reference: Forces

Robot representation

In this section it is possible to visualize first the path followed by the CoM, right and left foot in time:

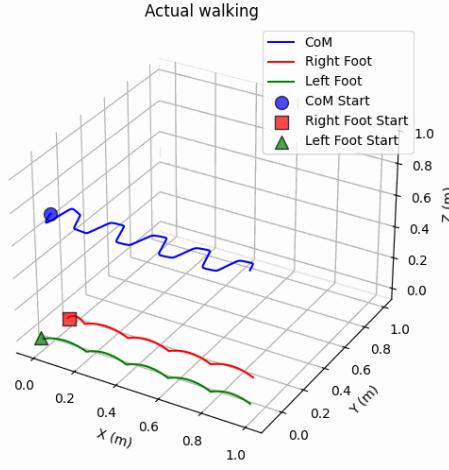


Figure 9: Walking