An Implementation of Trajectory Optimization for Walking Robots

Ömer Tarık Banuş

19 January 2025

1 Introduction

The development of legged robots has garnered significant attention in recent years due to their potential for navigating complex and unstructured environments. One of the critical challenges in this field is trajectory optimization, which involves planning efficient and stable movements that accommodate the unique dynamics of legged locomotion.

Recent literature highlights significant advancements in trajectory optimization (TO) and model predictive control (MPC) for legged systems. Wensing [4] discuss how TO and MPC have become fundamental in enabling legged robots to perform complex tasks by fully exploiting their dynamics within a prediction horizon. Studies have shown that TO is effective for planning open-loop trajectories over long horizons, while MPC facilitates rapid re-planning and feedback stabilization over shorter intervals [1,4]. Additionally, Silva and Machado [3] review various optimization strategies, including mimicking biological locomotion and employing genetic algorithms to enhance the efficiency of legged robots. They emphasize the need for robust designs that can adapt to diverse terrains while minimizing energy consumption.

In this project, I will be implementing the work of Winkler et al. [5], a powerful framework specifically designed to facilitate real-time trajectory optimization for legged robots. My implementation focuses exclusively on quadrupedal robots, aiming to enhance their performance in various tasks, such as walking, running, and climbing, while ensuring stability and adaptability in dynamic environments.

2 Theory

Optimization plays a pivotal role in trajectory planning for legged robots. The intricate dynamics of legged locomotion demand sophisticated optimization techniques to achieve efficient and stable movements. The trajectory optimization problem is inherently nonlinear, encompassing constraints and objectives derived from robot dynamics, kinematics, and environmental interactions.

Linear Model Predictive Control (MPC), as explored by Kim et al. [2], has demonstrated potential in controlling dynamic quadrupeds. However, linear assumptions often fail to accurately capture the nonlinearities of legged locomotion, particularly in dynamic and unstructured environments. While linear MPC offers computational efficiency, its applicability is limited in highly dynamic maneuvers or complex terrains.

In contrast, a nonlinear optimization framework directly incorporates the full-body dynamics of the robot. The optimization problem is formulated as a nonlinear programming (NLP) problem, utilizing sequential quadratic programming (SQP) techniques to effectively handle nonlinearities. This approach enables precise modeling of contact forces, robot dynamics, and kinematic constraints.

The mathematical formulation of the trajectory optimization problem is expressed as:

$$\min_{x} \quad J(x) \quad \text{(Objective Function)} \tag{1}$$

$$\text{s.t.} \quad \mathbf{r}''(t) = \frac{1}{m}\sum_{i=1}^{n_i} \mathbf{f}_i(t) - \mathbf{g}, \tag{2}$$

$$\mathbf{I}\boldsymbol{\omega}'(t) + \boldsymbol{\omega}(t) \times \mathbf{I}\boldsymbol{\omega}(t) = \sum_{i=1}^{n_i} \mathbf{f}_i(t) \times (\mathbf{r}(t) - \mathbf{p}_i(t)) \quad \text{(Dynamic Constraints)} \tag{3}$$

$$\mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}(t), \boldsymbol{\theta}(t)), \tag{4}$$

$$|\mathbf{R}(\boldsymbol{\theta})[\mathbf{p}_i(t) - \mathbf{r}(t)] - \bar{\mathbf{p}}_i| \leq \mathbf{b} \quad \text{(Kinematic Constraints)} \tag{5}$$

$$\mathbf{f}_i(t \notin \mathcal{C}_i) = 0, \tag{6}$$

$$\mathbf{p}_i(t \in \mathcal{C}_{i,s}) = \text{const.}, \quad \dot{\mathbf{p}}_i(t \in \mathcal{C}_{i,s}) = 0 \quad \text{(Contact Constraints)} \tag{7}$$

$$\mathbf{p}_i^z(t \in \mathcal{C}_{i,s}) = h_{\text{terrain}}(\mathbf{p}_i^{x,y}(t)), \tag{8}$$

$$\mathbf{f}_n(t \in \mathcal{C}_{i,s}) \geq 0, \quad |f_t(t)| \leq \mu f_n(t) \quad \text{(Terrain Constraints)} \tag{9}$$

$$\ddot{\mathbf{r}}_k(t_k^+) = \ddot{\mathbf{r}}_{k+1}(t_k^-) \quad \text{(Base Acceleration Constraints)} \tag{10}$$

$$x \in \mathcal{X} \quad \text{(Variable Bounds)}. \tag{11}$$

where $J(x)$ represents the cost function, typically minimizing energy expenditure or ensuring trajectory smoothness. The constraints enforce adherence to dynamic equations, kinematic limits, and terrain feasibility.

2.1 Robot Model

The robot model constrains the position of the i-th end-effector, $\mathrm{p}_i(t)$, within a reachable region, $\mathcal{R}_i(\mathrm{r}, \theta)$, defined as:

$$\mathrm{p}_i(t) \in \mathcal{R}_i(\mathrm{r}, \theta) \quad \Leftrightarrow \quad |\mathbf{R}(\theta)[\mathrm{p}_i(t) - \mathrm{r}(t)] - \bar{\mathrm{p}}_i| < \mathrm{b}, \tag{12}$$

where $\mathbf{R}(\theta)$ denotes the rotation matrix parameterized by θ, $\mathrm{r}(t)$ is the position of the robot's base, $\bar{\mathrm{p}}_i$ is the nominal position of the i-th end-effector, and b represents the bounds of the reachable region.

2.2 Dynamic Model

The dynamic behavior of the robot is governed by Newton-Euler equations, which are incorporated as constraints in the optimization:

$$m\ddot{\mathbf{r}}(t) = \sum_{i=1}^{n_i} \mathbf{f}_i(t) - m\mathbf{g}, \tag{13}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}(t) \times \mathbf{I}\boldsymbol{\omega}(t) = \sum_{i=1}^{n_i} \mathbf{f}_i(t) \times (\mathbf{r}(t) - \mathbf{p}_i(t)), \tag{14}$$

where m is the robot's mass, $\mathbf{r}(t)$ is the position of the robot's center of mass, $\mathbf{f}_i(t)$ represents the contact force at the i-th end-effector, $\mathbf{g}$ is the gravitational acceleration, $\mathbf{I}$ is the inertia tensor, and $\boldsymbol{\omega}(t)$ is the angular velocity. To ensure physical consistency, these constraints are enforced at regular time intervals throughout the trajectory.

2.3 Force Constraints

For physically correct locomotion, contact forces must satisfy friction constraints to prevent slipping:

$$|f_t(t)| < \mu f_n(t), \quad f_n(t) \geq 0, \tag{15}$$

where $f_t(t)$ and $f_n(t)$ are the tangential and normal components of the contact force, respectively, and μ is the friction coefficient. The forces must also stay within feasible limits determined by the robot's actuators.

2.4 Base Acceleration Constraints

Base acceleration constraints ensure smooth transitions between trajectory segments. The acceleration at the junction between two base polynomials must be continuous:

$$\ddot{\mathbf{r}}_k(t_k^+) = \ddot{\mathbf{r}}_{k+1}(t_k^-), \tag{16}$$

where t_k^+ and t_k^- denote the endpoints of consecutive segments. This condition prevents discontinuities in forces or velocities.

2.5 Contact Constraints

Contact constraints define the state of each end-effector during swing and stance phases. For a given contact phase $\mathcal{C}_{i,s}$, the foot position remains stationary:

$$\dot{\mathbf{p}}_i(t \in \mathcal{C}_{i,s}) = 0 \quad \Rightarrow \quad \mathbf{p}_i(t \in \mathcal{C}_{i,s}) = \mathbf{p}_{i,s} = \text{const.} \tag{17}$$

During the swing phase, the contact force is zero:

$$\mathbf{f}_i(t \notin \mathcal{C}_i) = 0. \tag{18}$$

The times at which foot i is in contact for the s-th stance phase are defined as:

$$\mathcal{C}_{i,s} = \left\{ t \,\middle|\, 0 < t - \sum_{j=1}^{2s-1} \Delta T_{i,j} < \Delta T_{i,2s} \right\}, \tag{19}$$

where $\Delta T_{i,j}$ denotes the duration of each swing or stance phase. This ensures that the contact states alternate between swing and stance phases.

2.6 Foot Motion Parameterization

The trajectory of each foot is parameterized using a cubic polynomial:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \tag{20}$$

where the coefficients a_i are functions of the time interval ΔT, initial position x_0, initial velocity $\dot{x}_0$, final position x_1, and final velocity $\dot{x}_1$. These polynomials ensure smooth foot motion during swing phases.

2.7 Terrain Constraints

Feet in contact with the terrain must satisfy:

$$\mathbf{p}_i^z(t \in \mathcal{C}_{i,s}) = h_{\text{terrain}}(\mathbf{p}_i^x(t), \mathbf{p}_i^y(t)), \tag{21}$$

where h_{terrain} is the terrain height at the corresponding 2D position $(\mathbf{p}_i^x, \mathbf{p}_i^y)$. This constraint ensures physical consistency with the terrain profile.

3 Implementation

In this project, I carefully considered computational efficiency, as solving real-world robotics problems often demands rapid solution times. Since C++ is widely used in robotics software development, I implemented the project in this language to ensure compatibility and usability.

I built upon resources developed by a research group at ETH Zurich, who published both the nonlinear optimization library IFOPT and the source codes [5]. By utilizing these resources, I was able to focus on implementing constraints

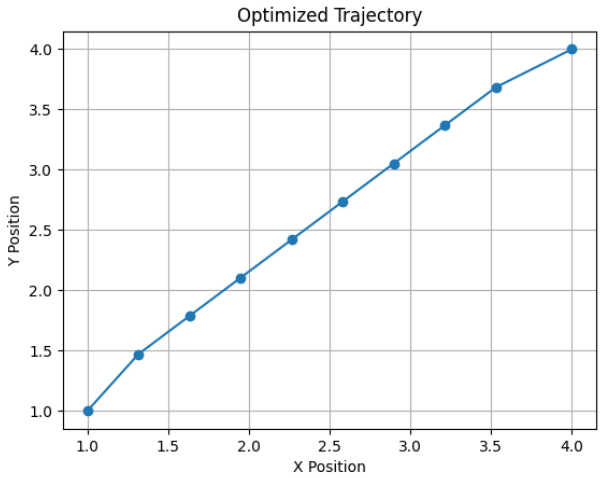

Figure 1: Optimized trajectory of a point mass moving from (1,1) to (4,4) in an unobstructed environment. The solution minimizes the travel distance while satisfying the map constraints.

and equations specific to the problem rather than starting from scratch. I developed custom code to define and incorporate the necessary constraints and equations into the optimization framework.

3.1 Simple 2D Trajectory Optimization

In IFOPT is an extremely useful tool for nonlinear optimization problems in my opinion. Constructing the problems were very easy. By defining classes inheriting from given structure one can easily construct such problems. All I had to do were to define variable and constraint classes with pre-determined Eigen vectors and bounds for values. In the constraint class I am defining constraint functions and defining boundaries for them.

To familiarize myself with the tools and optimization process, I began with simpler problems before progressing to more complex scenarios. The simplest example involved a two-dimensional case where a point mass traveled between two locations in an unobstructed environment. As shown in Figure 1, the problem's constraints included limiting the movement of the point mass within a defined map area, while the cost function minimized the total travel distance. I defined the optimization variables as n points along the trajectory, with a constraint ensuring that the maximum distance between any two consecutive points did not exceed a predefined value k.

Building upon this, I introduced an obstacle into the environment, defined mathematically as a circular constraint. The presence of this obstacle added complexity to the optimization problem. Figures 2 and 3 depict the results

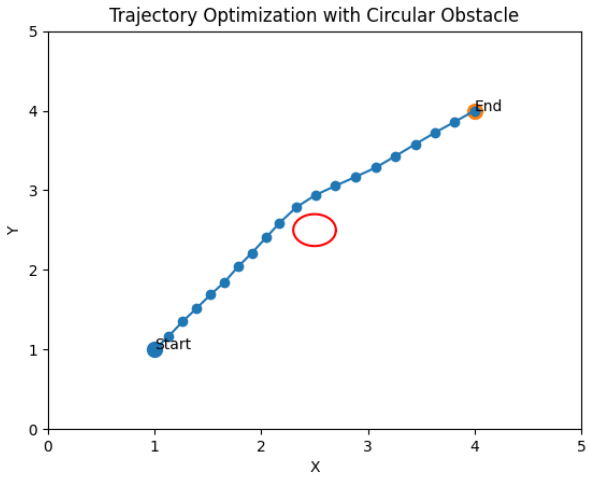

Figure 2: Optimized trajectory of a point mass navigating around a circular obstacle. The solver identifies a feasible path that avoids the obstacle while minimizing travel distance.

of these scenarios. The solver successfully identified feasible trajectories that avoided the obstacle. However, as the obstacle's size increased, the number of nodes needed to represent the trajectory also had to increase to meet the constraint of maximum allowable distance between consecutive points. This experience highlighted an important lesson: despite the problem being optimization-based, it is essential to understand the solution method and apply heuristics to appropriately tune the problem's parameters.

3.2 Quadruped Trajectory Optimization

After completing the 2D case, I began constructing a 3D problem. To simplify the initial setup, I modeled a point mass in 3D subjected to an applied force F. In this scenario, my variables were $\{p, f\}$, where p represents the position vector and f the force vector. The state-space representation was defined as:

$$x = \begin{bmatrix} p \\ \dot{p} \end{bmatrix}, \quad u = [f], \quad x_{k+1} = Ax_k + Bu_k,$$

where $A = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Initially, I faced challenges controlling the mass due to an imbalance between the number of constraints and the available states.

Subsequently, I implemented the constraints and variables described in the paper. The primary variables included base position, base angles, foot positions, foot forces, and contact schedules. Base positions and angles were initialized

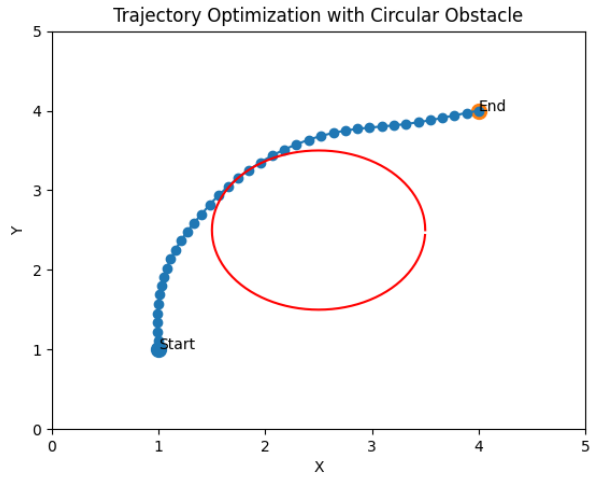

Figure 3: Trajectory optimization with a larger circular obstacle. The increased size of the obstacle necessitates a higher number of trajectory nodes to meet the maximum distance constraint between consecutive points.

as equally spaced values between the starting and final configurations. Foot positions and forces were parameterized using quintic polynomials defined by:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \quad a_i = f(\Delta T, x_0, \dot{x}_0, x_1, \dot{x}_1)$$

where a_i are

$$\begin{aligned} a_0 &= x_0, \\ a_1 &= \dot{x}_0, \\ a_2 &= -\Delta T^{-2}[3(x_0 - x_1) + \Delta T(2\dot{x}_0 + \dot{x}_1)], \\ a_3 &= \Delta T^{-3}[2(x_0 - x_1) + \Delta T(\dot{x}_0 + \dot{x}_1)]. \end{aligned}$$

Here, the start and end points of the polynomials and their durations were optimization variables. In my initial trials, I kept the contact durations ΔT constant, simplifying the problem.

I proceeded to define the constraints. Dynamics constraints ensured that each solved point satisfied the dynamics equation relative to the previous one. Base motion constraints were introduced to keep the base position within an acceptable range. Terrain constraints enforced foot contact at the ground height, defined as zero for flat terrain or sampled from a heightmap for uneven surfaces. Force constraints were implemented as described in the theory section.

This configuration yielded workable solutions, as shown in Figures 4 and 4. However, in some cases, it generated impractical solutions or failed to solve certain instances. One particular issue was that the feet path had a apex height

of 6 meters which is non-sense. Since I am not utilizing the entire model of the robot in this case, the foot contact points had to be constrained.

Furthermore, in my initial trials I was unable to reach to successful solutions. I saw that many of the variables had to be warm started to reach a solution faster. One downside of nonlinear optimization is that nonlinear problems are dependent on initial conditions.

After fixing the broken variables, I succeeded in obtaining sensible solutions. Subsequently, I added contact duration optimization as well. To incorporate contact durations as variables, I introduced total duration as a constraint for the solver to ensure feasible solutions. Figure 5 presents the unoptimized contact schedule scenario, whereas Figure 6 is taken from the optimized contact schedule scenario. It is notable that in the optimized contact schedule scenario, the contact forces remain constant during the contact phase, while the yaw exhibits greater variability. Furthermore the body states were followed better in this case.

4 Results

In this project, I successfully configured a solver capable of generating feasible solutions for quadrupedal locomotion. By inputting a quadruped model similar to ANYmal in terms of weight and dimensions, the solver produced comparable results. This outcome was anticipated, as the solver employs a simplified model focusing on the base and foot positions, without accounting for leg configurations, joint torques, or leg mass and inertia. Consequently, the solver demonstrates generalizability across various quadrupedal robots.

Comparing Figures 6 and 5, it is evident that optimizing contact schedules results in minimal changes. This is because the symmetric trot gait, used in the fixed contact schedule scenario, is often considered optimal for flat terrain. Many solutions in optimal control and reinforcement learning converge to this gait under such conditions. However, optimizing contact schedules becomes more significant in uneven terrains.

In this project, the quadruped transitions from coordinates (0,0) to (2,0) on flat ground within 2.4 seconds. While the original implementation can handle uneven terrain, I was unable to achieve valid solutions in such scenarios. Additionally, without contact schedule optimization, the solver computes the 2.4-second trajectory in approximately 0.8 seconds. When contact schedule optimization is included, computation time increases to about 4.6 seconds on a laptop equipped with an Intel i9 processor. This extended computation time may pose challenges for real-time hardware deployment.

During development, I observed that the solver is sensitive to initial conditions and problem parameters. This fragility necessitates careful parameter tuning to ensure reliable performance.

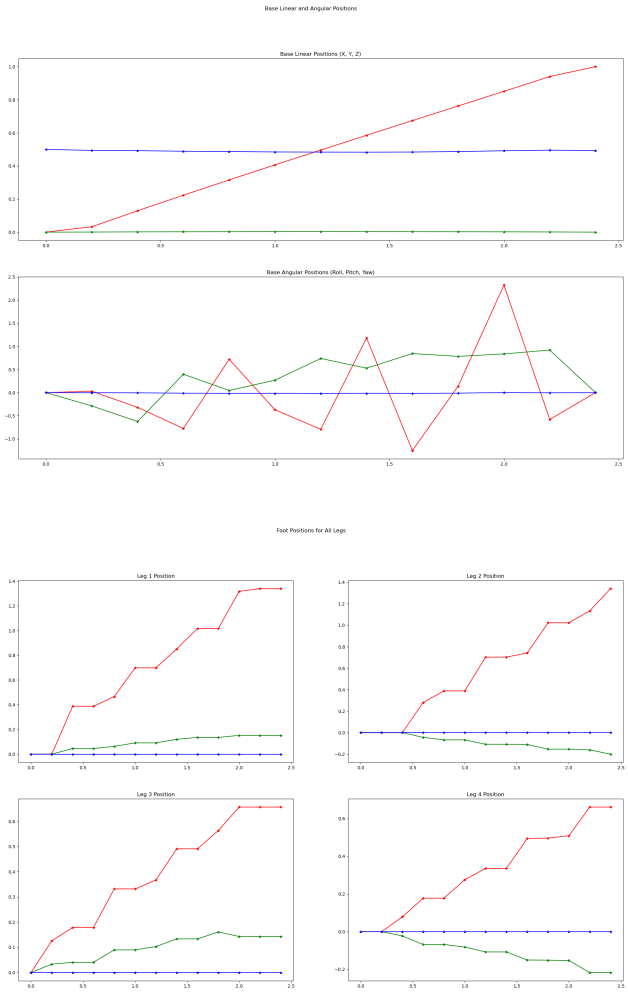

Figure 4: Base positions and angles (top) and foot positions (bottom) during trajectory optimization. The top plot shows the trajectory of the base in terms of positions and angles, while the bottom illustrates the optimized foot positions.

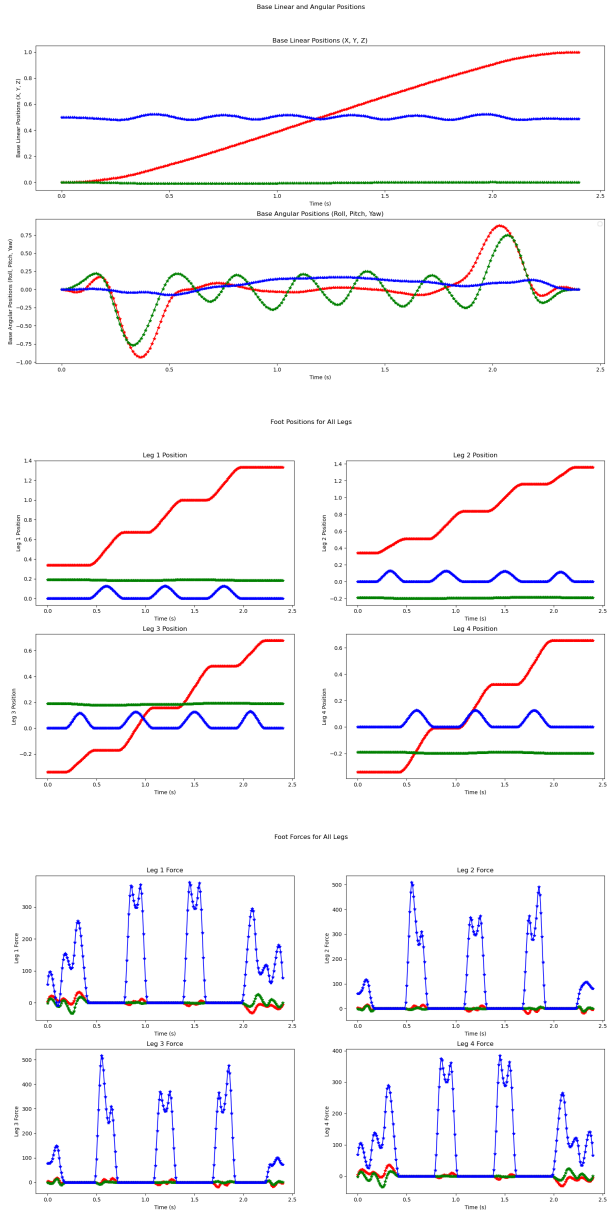

Figure 5: Base positions and angles (top), foot positions (middle), and foot forces (bottom) in a successful trajectory optimization scenario without optimizing contact schedules. The sequence illustrates the relationship between base motion, foot placement, and applied forces.

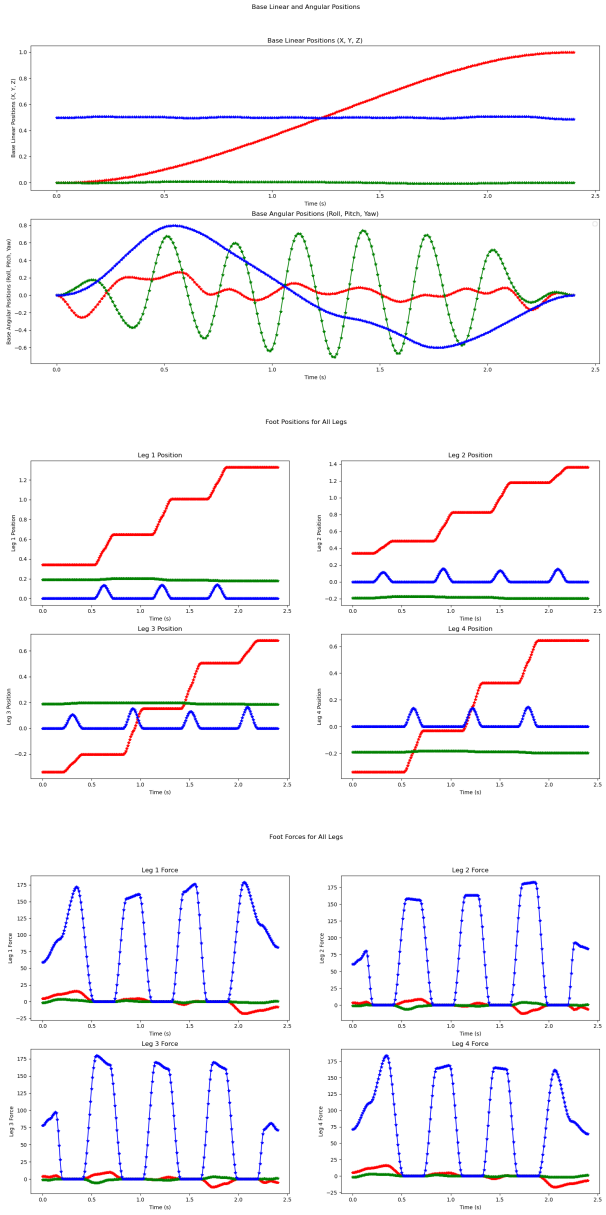

Figure 6: Base positions and angles (top), foot positions (middle), and foot forces (bottom) in a successful trajectory optimization scenario with optimized contact schedules. The sequence illustrates the relationship between base motion, foot placement, and applied forces.

5 Conclusion

This study presented a trajectory optimization framework capable of generating dynamic, efficient, and stable motion plans for quadrupedal robots. By integrating nonlinear dynamics, kinematic constraints, and terrain interactions, the framework achieved robust solutions across various test scenarios. From simple 2D point mass navigation to 3D quadrupedal motion with adaptive gaits, the results demonstrated the solver's versatility and physical realism. Key findings include the successful application of quintic polynomial parameterization for trajectory generation, the effective handling of contact forces and terrain constraints, and the capability to discover suitable gait patterns automatically.

Although this paper remains one of the most cited works in legged locomotion, my implementation highlighted several limitations. The trajectory optimization framework employs a simplified model that does not incorporate the full dynamics of the robot. As a result, the solver can converge to infeasible solutions when trapped in poor local minima. To address this, the search space is heavily restricted, which, while preventing infeasible solutions, limits the solver's generalizability. Furthermore, the contact model used is insufficient for real-world applications. In the published simulations, instances of the robot stepping through vertical walls were observed, which would not be realistic in practical scenarios.

The solver relies heavily on warm-starting, employing heuristics tailored to flat ground. While these heuristics work in simple cases, they fail in highly uneven terrains, where linearized MPCs may outperform the nonlinear approach due to better reliability and adaptability. The author claims that the planner is tested on actual hardware and is thus valid for real-world use. However, the published trials show the robot taking only one step, lacking demonstrations of the robot traversing challenging terrains that would necessitate a nonlinear MPC. Additionally, hardware trials reveal issues with the solved contact forces, stemming from sparse node optimization followed by quintic polynomial interpolation. This approach sacrifices the precision of force calculations at each time step, unlike linear MPCs, which compute forces dynamically, resulting in more responsive behaviors.

Despite these shortcomings, this work laid the foundation for numerous subsequent studies in the field of legged locomotion. It inspired researchers to address its limitations and advance the state-of-the-art in trajectory optimization. This paper remains a milestone in the literature, representing a significant step forward in nonlinear optimization for walking robots.

Implementing this framework was a valuable experience for me, providing firsthand insight into the open problems and limitations of modern approaches. These practical challenges, often understated in published works, have deepened my understanding of the field and motivated me to explore solutions for these pressing issues in future research.

References

[1] Dylan M. Asmar, Ransalu Senanayake, Shawn Manuel, and Mykel J. Kochenderfer. Model Predictive Optimized Path Integral Strategies, March 2023. arXiv:2203.16633 [eess].

[2] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control, September 2019. arXiv:1909.06586 [cs].

[3] Manuel Fernando Silva and Ja Tenreiro Machado. A literature review on the optimization of legged robots. *Journal of Vibration and Control*, 18(12):1753–1767, October 2012.

[4] Patrick M. Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-Based Control for Dynamic Legged Robots. *IEEE Transactions on Robotics*, 40:43–63, 2024. Conference Name: IEEE Transactions on Robotics.

[5] Alexander W. Winkler, C. Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, July 2018. Conference Name: IEEE Robotics and Automation Letters.