Legged Robots
MICRO-507

October 21st, 2025

Design and Optimization of a Quadruped Jumping Controller

Authors

Jules Raphael Simon Streit,	jules.simonstreit@epfl.ch (327732)
Fatih Selim Yilmaz,	fatih.yilmaz@epfl.ch (314970)
Cyril Nils Goffin,	cyril.goffin@epfl.ch (373937)

Group 16

Table of contents

1. Introduction

In this project, the single-leg control principles is extended to a full quadruped robot capable of dynamic 3D jumping. The goal is to design and analyze a controller inspired by the Quadruped-Frog architecture of Bellegarda *et al.* [1], combining model-based control with bio-inspired mechanisms. The controller generates rhythmic foot force profiles that drive jumping behaviors such as forward, lateral, and twist jumps. These forces are tracked at the joint level using the Jacobian, while a Cartesian impedance controller maintains the feet near their nominal positions under the hips. To improve base stability and robustness to uneven terrain, a Virtual Model Control (VMC) component is used to regulate the roll and pitch of the body, inspired from [2] and [3]. By tuning and optimizing the parameters of this architecture, using the Optuna library [4], the impact of different force profiles, gains, and nominal foot positions on the jumping performance and stability is studied.

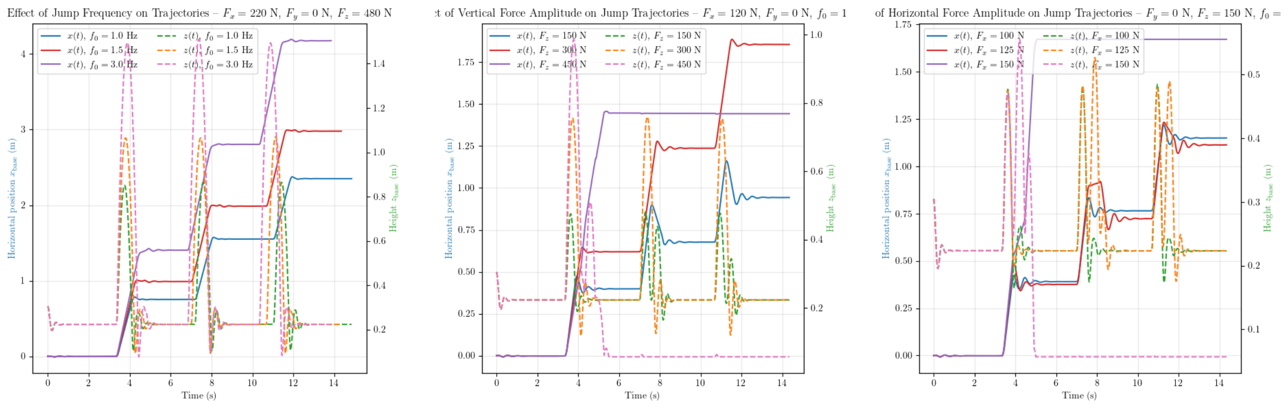
2. Controller design

2.1 Jumping modes implementation

The omnidirectional jumping controller supports three modes: **forward**, **lateral**, and **twist** jumps. Each mode is defined by the direction and distribution of the applied foot forces $\mathbf{F} = [F_x, F_y, F_z]^\top$, which are mapped into joint torques through the leg Jacobians $\boldsymbol{\tau}_i = J_i^\top \mathbf{F}_i$. In order to achieve these jumping behaviors, the following controller parameters were used (empirically tuned by trial and error): $\mathbf{K_p} = 800\mathbf{I_3}$, $\mathbf{K_d} = 30\mathbf{I_3}$ and $k_{\text{VMC}} = 500$. It was also empirically found that a good nominal foot position was $(0, 0.15, -0.20)$.

(a) Forward jump. All legs apply the same forward and upward force $\mathbf{F} = [F_x, 0, F_z]^\top$, where F_x drives forward motion and F_z controls jump height. An effective parameter set was found as $f_0 = 1.5\,\text{Hz}$, $f_1 = 0.25$, $F_x = 120\,\text{N}$, $F_y = 0\,\text{N}$, and $F_z = 300\,\text{N}$. Figure 2.1 shows how varying f_0, F_z, and F_x individually affects the robot's jumping behavior.

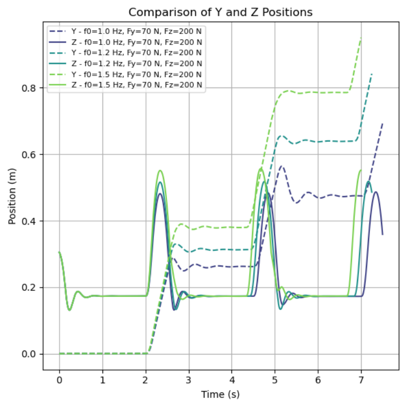
Higher f_0 increases jump frequency and displacement but reduces stability when too large. Raising F_z improves height and range up to about 400 N, beyond which landings become unstable. Similarly, larger F_x enhances forward motion but leads to oscillations and tipping when excessive. Overall, the results highlight a clear trade-off between jump power and stability.



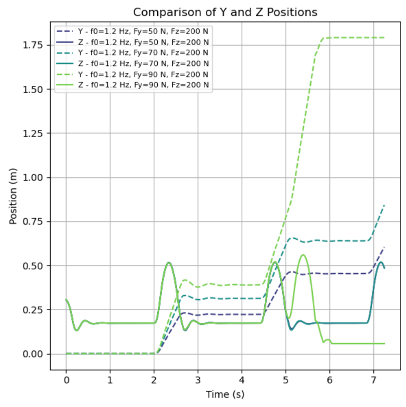
(a) f_0 variations (b) Vertical force F_z variations (c) Horizontal force F_x variations

Figure 2.1: Comparison of frequency and force amplitude effects on forward jumping trajectories.

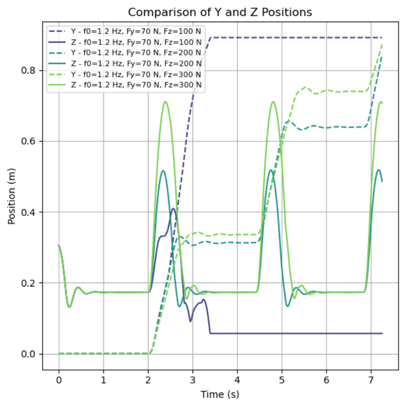
(b) Lateral jump. All legs apply a sideways and upward force $\mathbf{F} = [0, \pm F_y, F_z]^\top$, where the sign of F_y defines the jump direction (left or right). This mode produces motion along the y-axis while maintaining a similar vertical lift through F_z. By experimentation, we found an ideal parameter set: $f_0 = 1.2\,\text{Hz}$, $f_1 = 0.25$, $F_x = 0\,\text{N}$, $F_y = 70\,\text{N}$, and $F_z = 200\,\text{N}$. Increasing f_0 reduces the duration of the applied force. In the first plot, we observe that a higher f_0 improves stability, with fewer oscillations along the motion direction (y). The second plot shows the influence of F_y: a larger value increases the y displacement until instability causes the robot to fall. Finally, the third plot highlights the effect of F_z: greater F_z results in higher jumps and further jump, while too small a value leads to loss of balance, likely because the force remains active when the robot contacts the ground, preventing a stable landing. We also notice that as the robot performs more jumps, the yaw angle gradually drifts away from zero.



(a) f_0 variations



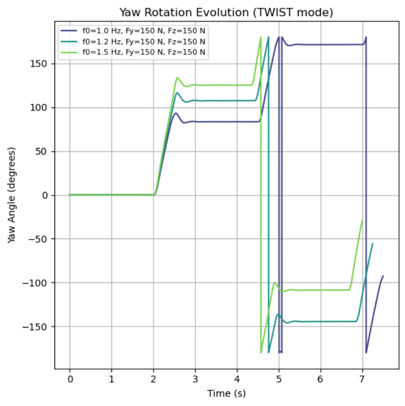
(b) F_y variations



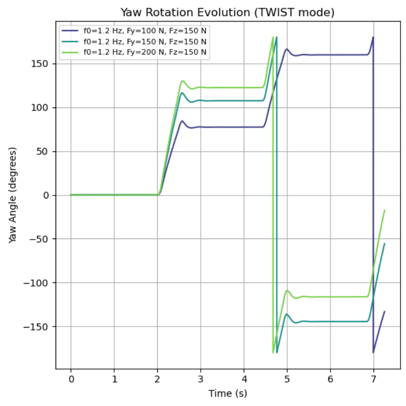
(c) F_z variations

Figure 2.2: f_0, F_y and F_z influences.

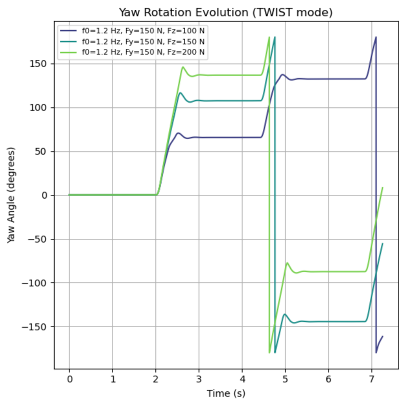
(c) Twist jump. To generate yaw rotation, opposite lateral forces are applied to the front and rear legs: $\mathbf{F}_{\text{front}} = [0, -F_y, 0]^\top$ and $\mathbf{F}_{\text{back}} = [0, +F_y, 0]^\top$. This creates a torque around the vertical axis, inducing clockwise or counterclockwise rotation depending on the sign of F_y. With $f_0 = 1.2$, $F_y = 150$, and $F_z = 150$, the motion remains stable. Increasing f_0, F_y, or F_z amplifies yaw rotation, and when all are raised simultaneously (e.g., $f_0 = 2.3$, $F_y = 400$, $F_z = 400$), the robot can rotate up to 540°. In reality, such behavior would be limited by actuator constraints and the simulation-to-reality gap.



(a) f_0 variations



(b) F_y variations



(c) F_z variations

Figure 2.3: f_0, F_y and F_z influences.

In all cases, the Virtual Model Controller (VMC) provides additional stabilization after landing by applying virtual restoring forces proportional to the body orientation errors.

2.2 Influence of center of mass

After tuning and validating the jumping controller, the influence of the z component of the nominal foot position p_d was analyzed. Forward jump experiments were performed for $(0, 0.10, -0.10)$, $(0, 0.10, -0.20)$, and $(0, 0.10, -0.30)$, with identical controller and force profile settings (Figure 2.4). Increasing the z component (lowering the feet) reduced forward performance: after three jumps, the quadruped reached 2.3 m, 2.0 m, and 0.3 m, respectively, while jump height dropped from 0.7 m to 0.5 m. This effect arises from the change in effective CoM height. A lower nominal foot position increases leg flexion at rest, allowing greater extension and energy transfer during push-off. Conversely, a higher stance limits stroke length and impulse. Thus, a deeper stance enhances propulsion, improving both jump height and distance.

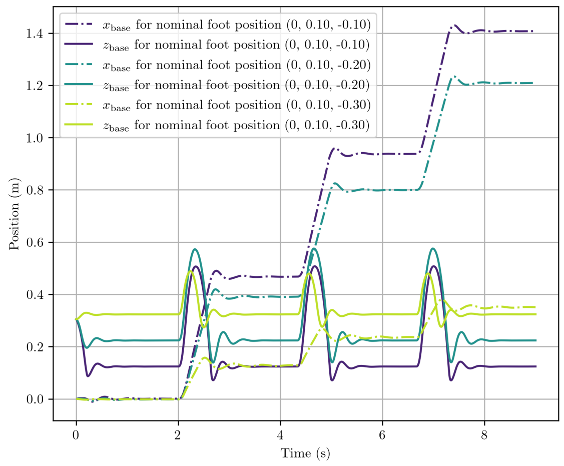


Figure 2.4: Forward jumping performance for different nominal foot positions (three consecutive forward jumps).

2.3 Influence of Virtual Model Control

Finally, the influence of Virtual Model Control (VMC) on the quadruped's stability is evaluated. Figure 2.5 shows the roll, pitch, and yaw evolution over three consecutive forward jumps, with and without VMC. Note that VMC is active only during ground contact phases (shaded blue regions in the figure). With VMC disabled, the body orientation oscillates within approximately $\pm 5°$ after each landing, while with VMC active, these oscillations are reduced to about $\pm 1°$. This improvement results from the virtual spring–damper forces applied at the feet, which generate corrective torques counteracting roll and pitch deviations. Consequently, VMC significantly enhances post-landing stability and allows faster recovery between jumps.

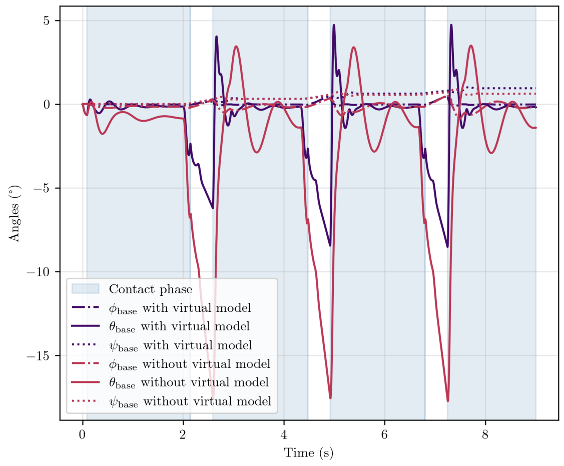


Figure 2.5: Forward jumping stability with and without VMC (three consecutive jumps).

3. Controller optimization

3.1 Furthest jumps controller

Across the three jump types, the optimization aimed to improve specific locomotion behaviors by tuning the foot force amplitudes $[F_x, F_y, F_z]$ and the impulse frequency f_0. For those tasks, the idle frequency f_1 is kept constant ($f_1 = 0.25$ Hz) as it does not impact the jump dynamics as the parameters are optimized over only one jump. For the **forward jump**, the goal was to maximize forward displacement:

$$\mathrm{J}_{\text{furthest/forward}} = x_{\mathrm{f}} - x_0.$$

After 25 iterations across 5 random seeds, the optimized parameters are: $\bar{F_x} = (137.5 \pm 11.9)$ N, $\bar{F_y} = (0.0 \pm 0.0)$ N, $\bar{F_z} = (232.3 \pm 49.6)$ N, and $\bar{f_0} = (0.90 \pm 0.15)$ Hz as you can see in 5.1.
For the *lateral jump*, the objective encouraged sideways motion:

$$\mathrm{J}_{\text{furthest/lateral}} = y_{\mathrm{f}} - y_0.$$

After 25 iterations across 5 random seeds, the optimized parameters are: $\bar{F_x} = (0.0 \pm 0.0)$ N, $\bar{F_y} = (138.5 \pm 8.0)$ N, $\bar{F_z} = (244.0 \pm 51.1)$ N, and $\bar{f_0} = (0.93 \pm 0.13)$ Hz as you can see in 5.2.
Finally, for the *twist jump*, the objective was designed to reward controlled yaw rotation:

$$\mathrm{J}_{\text{furthest/twist}} = \psi - \psi_0.$$

After 25 iterations across 5 random seeds, the optimized parameters are: $\bar{F_x} = (0.0 \pm 0.0)$ N, $\bar{F_y} = (141.6 \pm 7.1)$ N, $\bar{F_z} = (264.9 \pm 66.2)$ N, and $\bar{f_0} = (0.99 \pm 0.11)$ Hz yielding strong, well-balanced twisting jumps. Results can be seen in 5.3.

To ensure stable jumping behavior and prevent the robot from falling during optimization, several measures were implemented in both the simulation setup and the objective functions. First, stability thresholds were defined to detect failure cases: the robot was considered fallen if its torso height dropped below $z_{\min} = 0.1$ m or if the absolute roll or pitch angles exceeded 0.6 rad (about 34°). Any such event immediately resulted in an objective score J set to 0, discouraging unstable behaviors and moving to the next simulation step.

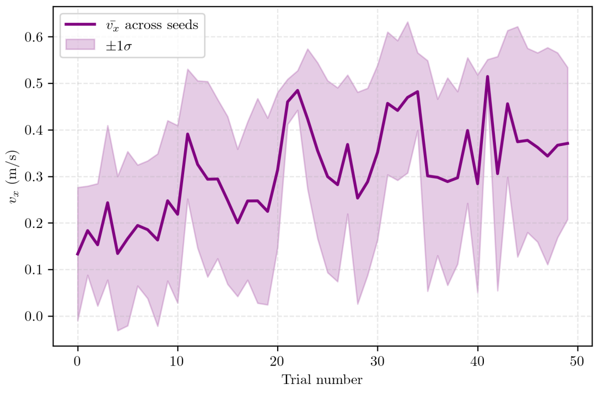
3.2 Fastest forward continuous hopping controller

To determine the parameter set yielding the fastest continuous forward hopping, a Bayesian optimization was performed over five consecutive jumps. The objective was to maximize the robot's mean forward velocity,

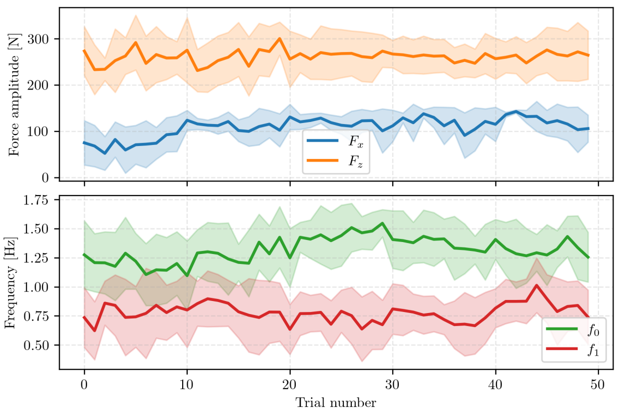
$$J_{fastest/forward} = v_x = \frac{\Delta x}{T} = \frac{x_T - x_0}{N_{\text{jumps}}\left(\frac{1}{2f_0} + \frac{1}{2f_1}\right)}, \tag{3.1}$$

where f_0 and f_1 denote the stance- and flight-phase frequencies. The cost was set to zero whenever a fall occurred (roll or pitch $> 30°$). All four parameters (F_x, F_z, f_0, f_1) were optimized jointly, since f_1 is strongly coupled with the take-off dynamics and cannot be tuned independently.

The search was repeated across 10 random seeds to assess robustness. At each iteration, the mean and standard deviation of the forward speed across seeds were computed, as well as the evolution of the mean parameter values. Figure 3.1a shows the convergence of the average forward speed, while Figure 3.1b illustrates the stabilization of the optimized parameters.



(a) Evolution of the mean forward speed across seeds.



(b) Evolution of the optimized parameters across seeds.

Figure 3.1: Optimization results across ten random seeds. (a) Convergence of the average forward speed and its standard variation; (b) Convergence of the optimized parameters and their standard deviation.

After 50 iteration across 10 random seeds, here are the parameters obtained: $\bar{F_x} = (139 \pm 7)$ N, $\bar{F_z} = (265 \pm 47)$ N, $\bar{f_0} = (1.40 \pm 0.16)$ Hz and $\bar{f_1} = (0.78 \pm 0.19)$ Hz.

3.3 Discussion

Before porting the controller from simulation to hardware we should take into account the actuators limitations(torque, speed and delay), like we saw in the twist jump previously. Motor dynamics and friction are taken into account in the simulation part. Sensors also introduce noise. The optimization on hardware should start from parameters validated in simulation to ensure a safe baseline. Bayesian Optimization can then be run with cautious exploration and cost functions that penalize instability or falls. After each jump, performance metrics such as distance, stability, and orientation are evaluated to iteratively update the model. A few trials should be sufficient to converge, while safe Bayesian Optimization can be used to keep forces and frequencies within hardware limits.

Designing a walking controller is different than designing a hopping one because it requires continuous coordination of leg contacts and smooth transitions between stance and swing phases. The controller must manage gait timing, balance the center of mass, and adapt to changing terrain through contact detection and compliance control. Walking also demands efficient energy use and precise friction management to ensure stable and natural locomotion.

4. Generative AI Declaration

1. Aspects where generative AI was used: Generative AI (ChatGPT) was used primarily to help with plot generation and visualization during the analysis phase, and occasionally to make written sections of the report more concise and stylistically consistent. It was not used for conceptual reasoning, data collection, or implementation of the optimization logic itself. The overall strategy was to rely on AI only for supporting or polishing tasks, while keeping all core design, coding, and interpretation steps fully manual.

2. Most useful aspects: Generative AI was most useful for producing clean and clear plotting scripts that automatically handled averaging, variance visualization, and multi-seed results. For example, the following prompt:

“Could you modify my function so that it plots the mean and standard deviation of the optimized parameters across 10 seeds, with two subplots for forces and frequencies?”

produced a ready-to-use, well-structured Matplotlib script with clear legends and axis labels. This significantly accelerated the visualization stage and improved figure readability.

3. Least useful aspects: Generative AI was least useful when trying to obtain precise formatting or context-specific scientific phrasing, as the suggestions sometimes lacked the exact terminology or conciseness expected in an academic report. For example, when asked:

“Can you write a short paragraph in the same style as the paper explaining the optimization setup?” the first drafts were too verbose and required manual editing to match the report’s tone and level of technical precision.

4.Overall impact on project workflow: The use of generative AI made the project smoother to complete, particularly by reducing time spent on repetitive plotting and figure-tuning tasks. It helped maintain a consistent style across figures and avoided small coding mistakes in visualization.

5. Impact on learning and understanding: The use of AI did not hinder conceptual understanding of the project. On the contrary, it allowed more focus on interpreting results and refining the optimization strategy, since less time was spent on auxiliary coding tasks. The learning process behind the methodology, control design, and optimization remained entirely intact.

Bibliography

[1] G. Bellegarda, M. Shafiee, M. E. Özberk, and A. Ijspeert, "Quadruped-frog: Rapid online optimization of continuous quadruped jumping," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1443–1450, arXiv preprint: 2403.06954. [Online]. Available: https://arxiv.org/abs/2403.06954

[2] M. Ajallooeian, S. Pouya, A. Spröwitz, and A. Ijspeert, "Central pattern generators augmented with virtual model control for quadruped rough terrain locomotion," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3321–3328. [Online]. Available: https://doi.org/10.1109/ICRA.2013.6631028

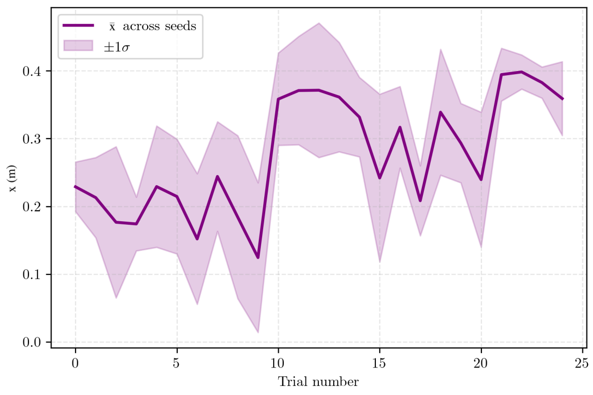
[3] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," vol. 20, no. 2, pp. 129–143. [Online]. Available: https://doi.org/10.1177/02783640122067309

[4] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery (ACM), pp. 2623–2631. [Online]. Available: https://doi.org/10.1145/3292500.3330701

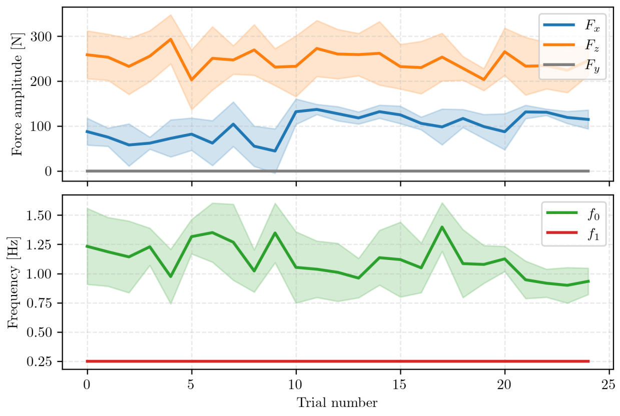
5. Appendix

Table 5.1: Search space of the optimized parameters for furthest jumps and fastest continuous forward hopping.

Parameter	**Symbol**	**Unit**	**Range**
Horizontal foot force amplitude	F_x	N	[0, 150]
Lateral foot force amplitude	F_y	N	[0, 150]
Vertical foot force amplitude	F_z	N	[150, 350]
Contact-phase frequency	f_0	Hz	[0.75, 1.75]
Flight-phase frequency	f_1	Hz	[0.25, 1.25]

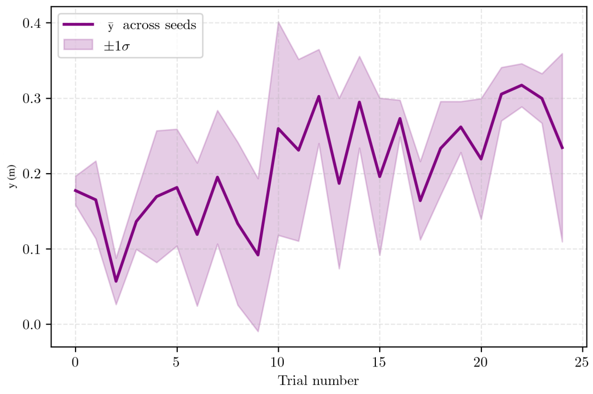


(a) Evolution of the mean foward position across seeds.

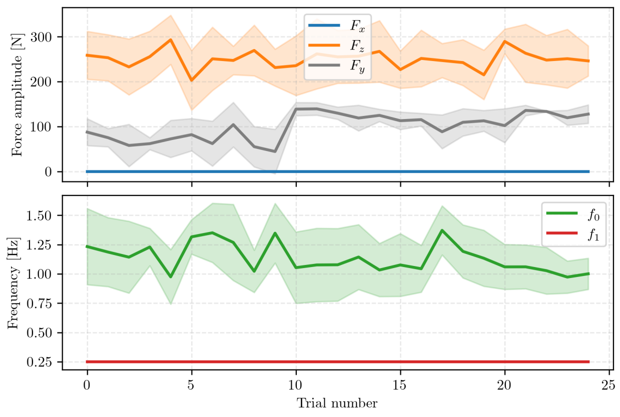


(b) Evolution of the optimized parameters across seeds.

Figure 5.1: Optimization results for furthest forward jumping across 5 random seeds. (a) Convergence of the average forward position and its standard deviation; (b) Convergence of the optimized parameters and their standard deviation.

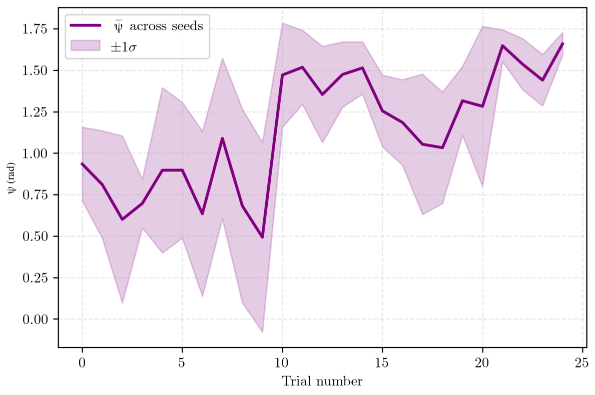


(a) Evolution of the mean lateral position across seeds.

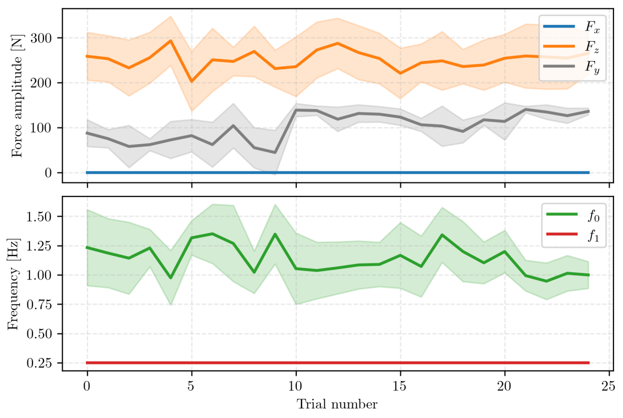


(b) Evolution of the optimized parameters across seeds.

Figure 5.2: Optimization results across 5 random seeds. (a) Convergence of the average lateral position and its standard deviation; (b) Convergence of the optimized parameters and their standard deviation.



(a) Evolution of the mean yaw angle across seeds.



(b) Evolution of the optimized parameters across seeds.

Figure 5.3: Optimization results across 5 random seeds. (a) Convergence of the average yaw angle and its standard deviation; (b) Convergence of the optimized parameters and their standard deviation.