Adaptive control of Quadruped robot under varying load conditions

Vamshi Kumar Kurva[1], Shishir Kolathaya[2]

***Abstract*— Many commonly used control frameworks rely on the robot's dynamic model. However, in practice, this model often proves to be inaccurate due to factors such as incorrect specification of the robot's physical parameters, mechanical wear and tear, and unforeseen changes like the addition of extra payloads during deployment. Significant deviations in the dynamics can severely impact the controller's performance. Our goal is to enhance the controller's model in real-time during deployment using onboard sensors and online learning. Specifically, our work focuses on quadruped locomotion under varying load conditions. We present a model-based force control framework that utilizes classical control techniques to effectively handle substantial changes in the mass and center of mass.**

Keywords: *Quadrupeds, legged locomotion, System Identification, Model Predictive Control(MPC), Force control*

I. INTRODUCTION

The load-carrying capability of quadruped robots is a critical attribute that significantly enhances their utility across a wide range of applications. These robots are designed to navigate diverse terrains with agility and stability, making them ideal for tasks in environments where wheeled robots may struggle. By enabling quadruped robots to carry substantial loads, we expand their potential uses in logistics, search and rescue operations, military missions, and agricultural tasks. This capability not only increases operational efficiency by reducing the need for human intervention in hazardous or inaccessible areas but also opens up new possibilities for automated delivery systems and infrastructure maintenance. Hence, improving load-carrying capability is essential for maximizing the practical value and versatility of quadruped robots in real-world applications.

While legged locomotion has matured, with significant advancements in navigating uneven terrains and handling minor disturbances robustly, there has been less emphasis on enhancing load-bearing capability [1] [2] [3] [4] [5] [6] [7]. However, few of the papers tried to address the issue. For instance, [8] focuses on identifying robot inertial parameters for high-performance model-based control. Their method employs an online recursive approach based on contact forces and joint angles to identify the Center of Mass (CoM) of the base, crucial for ensuring locomotion stability. However, the method is too slow that the robot stops to perform the online identification when the payload is detected and the locomotion is shown in the lab setting.

[1]Vamshi Kumar Kurva is associated with the department of Computer Science and Automation, Indian Institute of Science, Bangalore.

[2]Shishir N. Y. Kolathaya is with the Robert Bosch Center for Cyber Physical Systems and the Department of Computer Science & Automation, Indian Institute of Science, Bangalore.

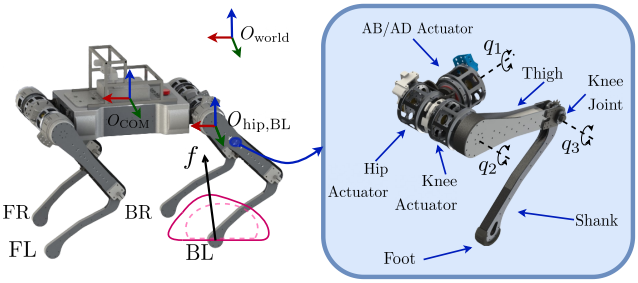


Fig. 1: An illustration of Stoch3 robot. The symbols FL, FR, BL, BR, represent the front-left, front-right, back-left, and back-right legs respectively. For simplicity, only the BL hip frame, $O_{\text{hip, BL}}$ is shown.

Alternatively, [9] adopts a different strategy by learning a locally linear, time-varying residual model around the robot's current trajectory instead of directly identifying physical parameters. This residual model, combined with a nominal model, supports real-time model-based control, as demonstrated experimentally with a 10 kg payload on a 12 kg A1 robot. Nonetheless, since the load distribution remains symmetric around the robot base, it minimally affects the CoM. [10] integrates $\mathcal{L}_1$ adaptive control techniques into a force control framework to handle model uncertainties. Their controller deployed on the A1 robot can carry 6 Kg payload stably while walking. In contrast, [11] proposes a robust min-max model predictive control (MPC) approach based on robust optimization principles for handling uncertain conditions.

In all the methods mentioned above, the payload is typically attached very near the center of the torso, thus shifting the CoM by a very small amount. From our experiments, we observed that significant changes in the CoM pose substantial challenges, occasionally leading to controller failures and causing the robot to fall – issues that do not occur with minor CoM changes. Therefore, our objective is to develop a controller capable of handling substantial changes in the CoM. Our contributions are as follows:

- We propose a model-based force controller equipped with an Online System ID to determine the payload mass and CoM of the robot in real-time.
- The proposed controller is validated in simulation using Pybullet physics engine [12] on a custom Stoch3 robot.
- We demonstrate that the controller can accurately track given commands even with significant CoM shifts.

The paper's organization is as follows: Section II describes the robot configuration and specifications. Section III describes force-based controller. Section IV presents simulation

results, assessments. Finally, Section V concludes our work.

II. Robot Description

This section will briefly describe the configuration of the robot used in the simulation. Stoch-3, as shown in Fig 1, is a custom-built dynamic quadruped robot developed for rapid prototyping of controllers. The robot has 4 legs, each leg is equipped with 3 actuators to give the foot 3 Degrees of Freedom (DoF). Thus, the robot model consists of 6 floating and 12 actuated DoF. We treat each leg independently to derive analytical relations for forward and inverse kinematics. Here q_1, q_2, and q_3 represent abduction, hip, and knee joints and form a serial-3R kinematic chain as shown in Fig 1. Some system parameters of the robot are presented in the following table I.

Parameter	Value	Unit
M	25	kg
I_{xx}	0.043	$\mathrm{kg\,m^2}$
I_{yy}	0.1127	$\mathrm{kg\,m^2}$
I_{zz}	0.2288	$\mathrm{kg\,m^2}$
Body length	0.541	m
Body width	0.203	m
abduction length	0.123	m
hip length	0.297	m
shank length	0.347	m

TABLE I: System Parameters of the Stoch3 Robot. M is the total robot of the mass, I_{xx}, I_{yy}, I_{zz} are the diagonal entries of the inertia matrix

III. Force control with Online SysID

This section provides an overview of the force control framework. The control architecture comprises several modules: ConvexMPC, PD controller, and Online SysID, as illustrated in Fig 2. We employ ConvexMPC [2] as a high-level controller to track the command velocities provided by a joystick. The high-level controller uses a gait scheduler that defines the gait timing and contact sequence for each leg. It then determines the desired foot forces for the stance leg and foot positions for the swing leg based on user commands. These high-level commands are converted into torques by the low-level controller. For the swing legs, foot positions are tracked using PD control, while for the stance legs, desired torques are generated using the leg Jacobian, which relates joint torques to end-effector forces. Online SysID is employed to identify changing parameters of interest using past data from the controller. We will briefly explain all the components below

A. ConvexMPC

Let $x = (\Theta, p, \omega, \dot{p}, g)$ represent the state of the base where $p, \dot{p}$ denote the linear position and velocity, Θ, ω, g denote the angular position, velocity and the acceleration due to gravity respectively. $\Theta = (\phi, \theta, \psi)^T$ are the Euler angles where ϕ is the roll, θ is the pitch, and ψ is the yaw. Additionally, let $r_i = (r_{ix}, r_{iy}, r_{iz})$ be the position of the i^{th} foot with respect to the CoM of the torso and $f_i = (f_{ix}, f_{iy}, f_{iz})$ be the control input for the i^{th} leg. The inertial effect of the legs is ignored due to their lightweight design, allowing the robot to be modeled as a Single Rigid Body (SRB). This SRB model, combined with certain approximations in the rotational dynamics, results in the following linear time-varying dynamics:

$$\frac{d}{dt}\begin{pmatrix}\Theta\\ p\\ \omega\\ \dot{p}\\ g\end{pmatrix} = \underbrace{\begin{pmatrix}0_3 & 0_3 & \mathcal{R}_z^T(\psi) & 0_3\\ 0_3 & 0_3 & 0_3 & 1_3\\ 0_3 & 0_3 & 0_3 & 0_3\\ 0_3 & 0_3 & 0_3 & 0_3\\ 0 & 0 & 0 & 0\end{pmatrix}}_{A(t)}\begin{pmatrix}\Theta\\ p\\ \omega\\ \dot{p}\\ g\end{pmatrix} + \underbrace{\begin{pmatrix}0_3 & 0_3 & 0_3 & 0_3\\ 0_3 & 0_3 & 0_3 & 0_3\\ \mathcal{I}^{-1}\hat{r_1} & \mathcal{I}^{-1}\hat{r_2} & \mathcal{I}^{-1}\hat{r_3} & \mathcal{I}^{-1}\hat{r_4}\\ 1_3/m & 1_3/m & 1_3/m & 1_3/m\\ 0 & 0 & 0 & 0\end{pmatrix}}_{B(t)}\begin{pmatrix}f_1\\ f_2\\ f_3\\ f_4\end{pmatrix}$$

where $\mathcal{R}_z(\psi)$ represent positive rotation of ψ about $z-$axis, $u = (f_1, f_2, f_3, f_4)$, $A \in \mathbb{R}^{13\times 13}$ depends only on the orientation and $B \in \mathbb{R}^{13\times 12}$ depends on mass m, foot step locations r_i and inertia matrix $\mathcal{I}$. MPC then solves for the Ground Reaction Forces (GRFs) needed to track the desired command velocities by solving the following optimisation problem over a finite horizon N

$$\begin{aligned}\min_{\mathbf{u}} \quad & \sum_{k=0}^{N-1}\left(\|\mathbf{x}_k - \mathbf{x}_{k,ref}\|_{\mathbf{Q}}^2 + \|\mathbf{u}_k\|_{\mathbf{R}}^2\right)\\ \text{subject to} \quad & \mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k, \quad k = 0,\dots,N-1\\ & \underline{c}_k \le \mathbf{C}_k\mathbf{u}_k \le \overline{c}_k, \quad k = 1,\dots,N-1\\ & \mathbf{x}_0 = \mathbf{x}(t)\\ & \mathbf{D}_k\mathbf{u}_k = 0, \quad k = 1,\dots,N-1\end{aligned}$$

where $\mathbf{x}_k, \mathbf{u}_k$ are the state and control inputs at time k, $\mathbf{Q}, \mathbf{R}$ are positive definite matrices of weights, $\mathbf{A}_k, \mathbf{B}_k$ represents the discrete time dynamics, $\underline{c}_k, \mathbf{C}_k, \overline{c}_k$ represents the inequality constraints on the control input, and $\mathbf{D}_k$ is a matrix that selects the GRFs corresponding to swing legs.

After finding an optimal sequence of control inputs for the next N steps, only the first optimal action is executed, the system goes into the next state and the process repeats. Since MPC only solves for GRFs and not joint torques, it need not be aware of leg kinematics and hence can be thought of as a high-level controller.

B. Online SysID

As previously discussed, the dynamics of the robot are influenced by system parameters such as mass, Center of Mass (CoM), inertia, and foot positions relative to the CoM. Accurately identifying these varying system parameters enhances the controller's ability to effectively track commands. Let $\mathcal{D} = \{(x_i, u_i);\ i = t-1, t-2, ..1, 0\}$ denote the data buffer containing state-action pairs up to time $t-1$. We

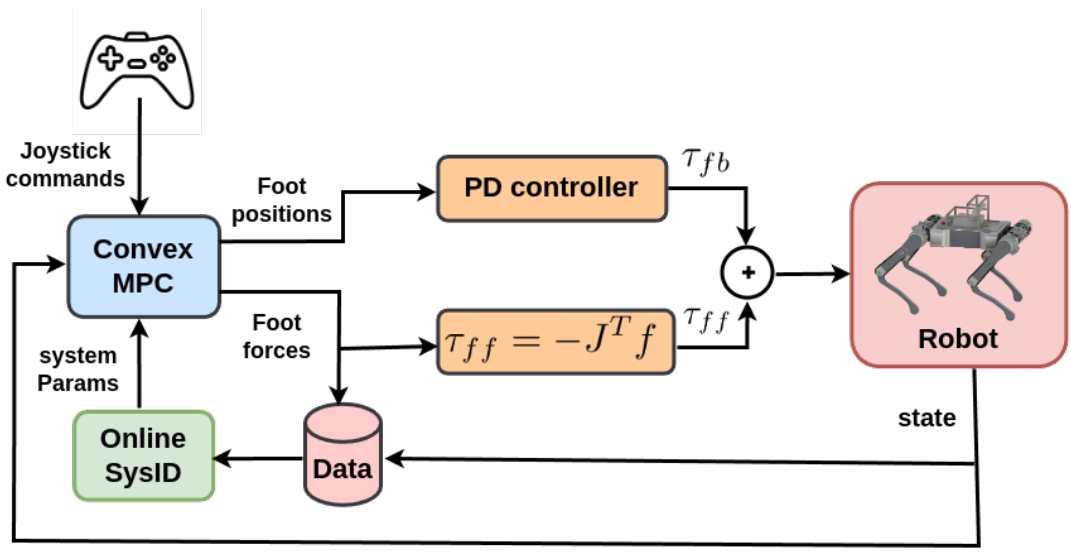


Fig. 2: Overview of the proposed force control framework. Joystick commands consists of (v_x, v_y, w_z). τ_{ff} is the feed-forward torque and τ_{fb} is the feedback torque. SysID module updates the system parameters once every few iterations and the updated parameters are plugged into the MPC controller.

employ heuristic-based algorithms to find the changing mass and CoM using the data collected from the controller.

Algorithm 1 Mass Estimation

Input: Data buffer $\mathcal{D}$, batch size, control time step dt
Output: Estimated mass
Initialization: $mass = [\,]$
For $i = t-1, t-2, \ldots, t-$ batch size:
  Estimate $\ddot{p}(i) = \frac{\dot{p}(i+1) - \dot{p}(i)}{dt} + (0.0, 0.0, -9.8)^T$
  $F_z(i) = f_{1z}(i) + f_{2z}(i) + f_{3z}(i) + f_{4z}(i)$
  $mass(i) = \frac{F_z(i)}{\ddot{p}_z(i)}$
  Append $mass(i)$ to $mass$
Return Average($mass$)

The mass estimation method, outlined in Algorithm 1, calculates the robot's mass by dividing the z-direction force acting on the robot by its z-direction acceleration at each time-step. These individual estimates are averaged to obtain a reliable measure. This approach specifically focuses on z-direction forces and accelerations to mitigate noise introduced by the finite difference method used for acceleration calculation, which is stabilized by incorporating gravitational acceleration.

Another critical parameter of interest is the CoM shift relative to the torso's center. Under no-load conditions, we assume the CoM aligns with the geometric center of the torso. However, under load conditions, Algorithm 2 describes how the CoM shift is determined. The estimation process identifies specific torso points where moments about the x and y axes are zero, assuming negligible GRFs in the x and y directions compared to the z-direction.

Algorithm 2 CoM Shift Estimation

Input: Data buffer $\mathcal{D}$, batch size, control time step dt
Output: Estimated CoM shift
Initialization: $\delta = [\,]$
For $i = t-1, t-2, \ldots, t-$ batch size:
  Estimate $F_z(i) = f_{1z}(i) + f_{2z}(i) + f_{3z}(i) + f_{4z}(i)$
  $\delta_x = \frac{r_{1x}(i)f_{1z}(i) + r_{2x}(i)f_{2z}(i) + r_{3x}(i)f_{3z}(i) + r_{4x}(i)f_{4z}(i)}{F_z(i)}$
  $\delta_y = \frac{r_{1y}(i)f_{1z}(i) + r_{2y}(i)f_{2z}(i) + r_{3y}(i)f_{3z}(i) + r_{4y}(i)f_{4z}(i)}{F_z(i)}$
  Append $(\delta_x, \delta_y, 0.0)$ to δ
Return Average(δ)

Once the CoM shift is computed, the foot positions are adjusted relative to the CoM rather than the center of the base frame. The updated mass and foot positions are then used in computing the GRFs for the subsequent time step.

C. Low-level controller

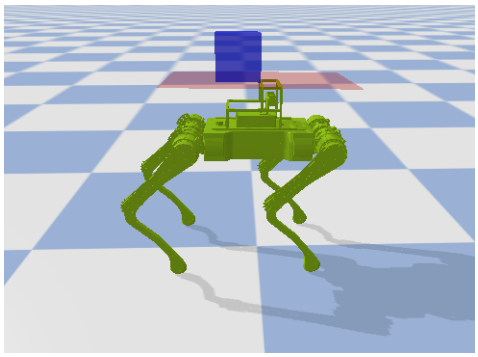
Torques for each leg are generated differently depending on whether the leg is in the swing or stance phase. For the swing leg, desired foot positions are converted to desired joint angles q_d using inverse kinematics. PD control is then used to track the desired joint angles which generates the feedback torques:

$$\tau_{fb} = k_p * (q_d - q) + k_d * (0 - \dot{q})$$

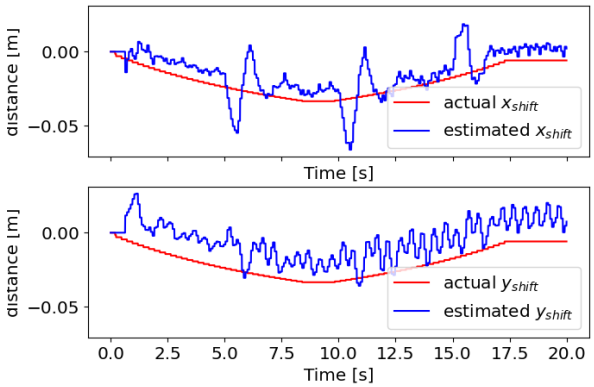
where k_p, k_d are proportional and derivative gains, $q, \dot{q}$ are the current joint angle and velocities. For the stance leg, desired GRFs are converted to the required feed-forward torques using the Jacobian equation

$$\tau_{ff} = -J(q)^T f$$

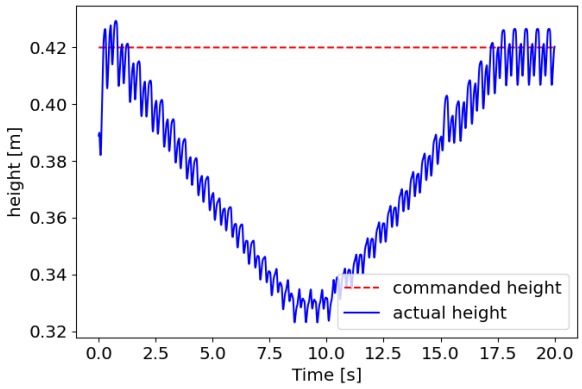
where $J(q)$ is the Jacobian matrix evaluated at joint position q and f is the desired GRF.



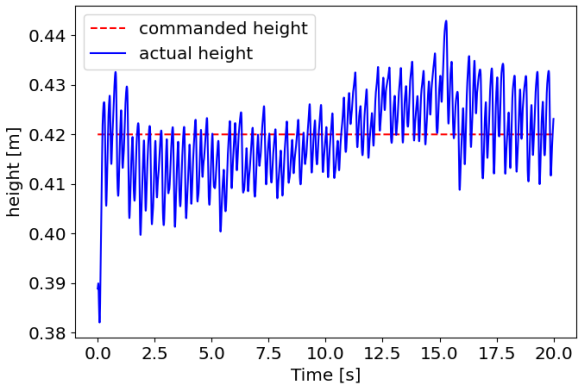
(a)



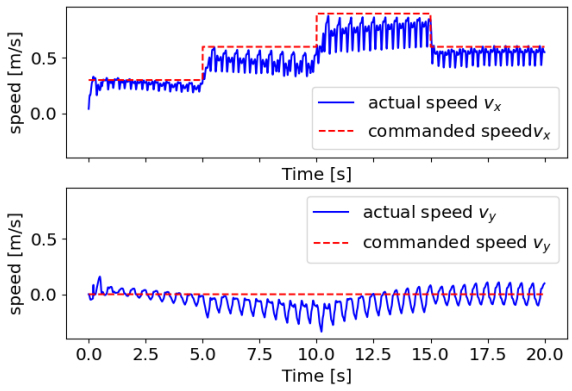
(b)



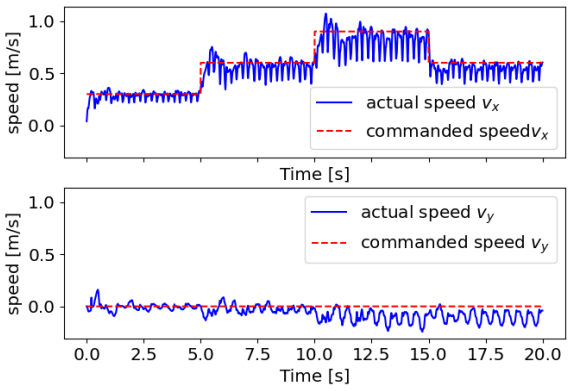
(c)



(d)

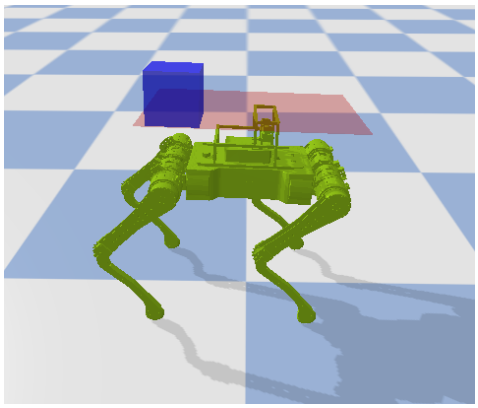


(e)

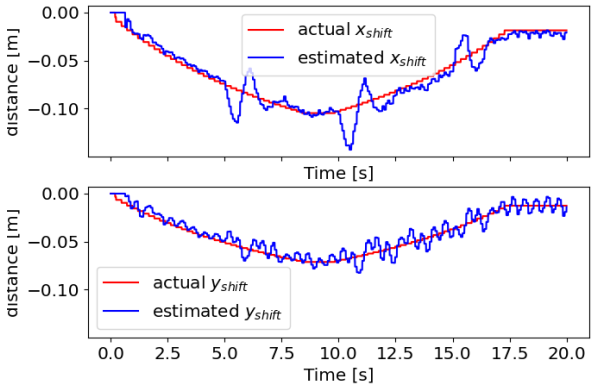


(f)

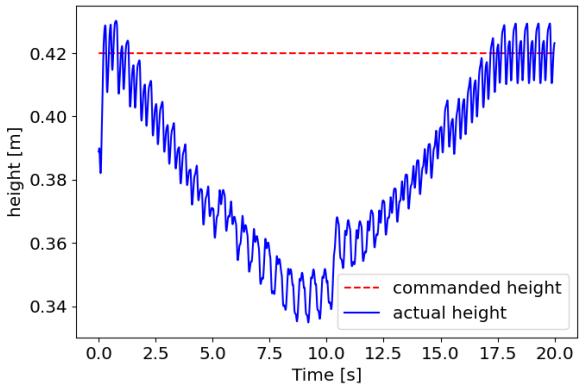
Fig. 3: Results of test1 in Pybullet: In the beginning, an unknown payload is put on the robot close to the center of the base frame and the mass is changed over time. (a) simulation scenario in pybullet (b) actual and estimated CoM shifts when using sysID (c) height tracking with base controller (d) height tracking with our controller (e) velocity tracking of base controller (f) velocity tracking of our controller



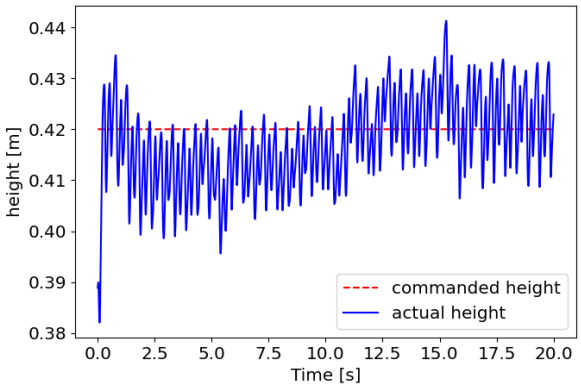
(a)



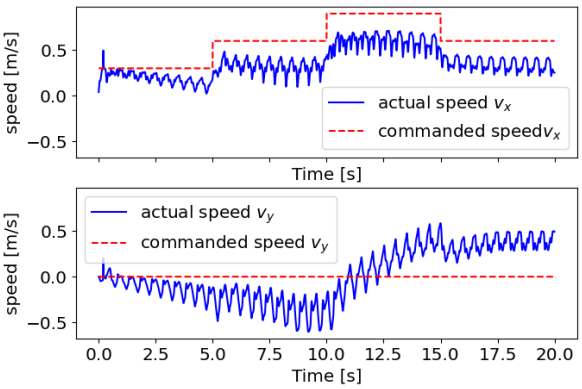
(b)



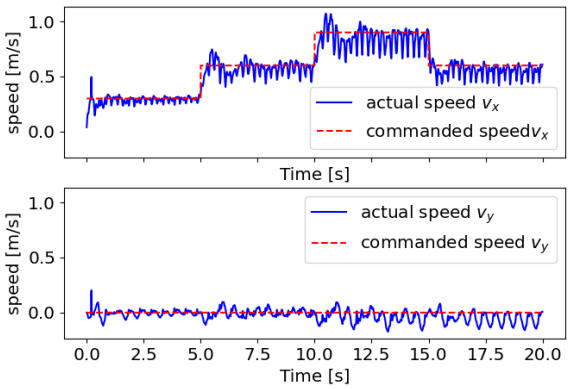
(c)



(d)



(e)



(f)

Fig. 4: Results of test2 in Pybullet: In the beginning, an unknown payload is put on the back of the robot and the mass is changed over time. (a) simulation scenario in pybullet (b) actual and estimated CoM shifts when using sysID (c) height tracking with base controller (d) height tracking with our controller (e) velocity tracking of base controller (f) velocity tracking of our controller

IV. RESULTS

To validate our approach, we used the PyBullet physics engine with a custom Stoch3 URDF model as our environment. In the simulation, we have access to all state variables, including joint positions and velocities. ConvexMPC serves as our base controller and is configured as follows:

- MPC runs at a frequency of 250 Hz and the PD controller at 1 kHz.
- Desired body height is set to 0.42 meters
- The controller always runs trot gait.

We compared the base controller and the adaptive controller under the following conditions:

- Test1: The payload is placed near the center, with its mass changing over time, resulting in less CoM shift.
- Test2: The payload is positioned at one of the corners, with its mass varying over time, resulting in a significant CoM shift.

The performance of the controller is evaluated based on its ability to track the commanded velocities and maintain the commanded height.

A. CoM Shift is less

In this scenario, we initially placed a payload of 2 kg at a distance of $(0.1, 0.1)$ meters from the center of the base frame. The payload mass was then incrementally increased until it reached 18 kg, followed by a similar decrement back to 2 kg. Consequently, the CoM shift varied correspondingly, reaching its maximum when the payload mass was highest and its minimum when the payload mass was lowest. Tracking performance under these conditions is presented in Fig 3. The adaptive controller clearly outperforms the base controller in tracking both height and velocity commands. However, the base controller's velocity tracking is comparable to our controller due to the minimal CoM shift.

B. CoM shift is more

This setting is similar to the previous one, but the initial payload location is $(0.25, 0.17)$ meters from the center of the base frame. As shown in Fig 4, the CoM shift is more pronounced in this case. The base controller struggles to track velocity commands, causing the robot to drift toward the CoM, and height tracking deteriorates with increasing payload mass. However, by identifying the changing parameters, our adaptive controller successfully tracks both height and velocity commands.

V. CONCLUSION

In this work, we presented a force-based controller capable of adapting to changes in load conditions. We employed a system identification technique to identify changing parameters of interest using data collected from the base controller in real-time. We validated our controller in PyBullet simulator under two different test conditions, demonstrating superior command tracking compared to the base controller. For future work, we plan to explore reinforcement learning-based methods to adapt to various other uncertainties and unstructured terrains.

REFERENCES

[1] H.-W. Park, P. Wensing, and S. Kim, "High-speed bounding with the mit cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, p. 027836491769424, 03 2017.

[2] 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," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[3] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control," 08 2022.

[4] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8484–8490.

[5] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021. [Online]. Available: https://arxiv.org/abs/2107.04034

[6] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," 2022.

[7] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," 2023.

[8] G. Tournois, M. Focchi, A. Del Prete, R. Orsolino, D. G. Caldwell, and C. Semini, "Online payload identification for quadruped robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4889–4896.

[9] Y. Sun, W. L. Ubellacker, W.-L. Ma, X. Zhang, C. Wang, N. V. Csomay-Shanklin, M. Tomizuka, K. Sreenath, and A. D. Ames, "Online learning of unknown dynamics for model-based controllers in legged locomotion," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8442–8449, 2021.

[10] M. Sombolestan, Y. Chen, and Q. Nguyen, "Adaptive force-based control for legged robots," *CoRR*, vol. abs/2011.06236, 2020. [Online]. Available: https://arxiv.org/abs/2011.06236

[11] S. Xu, L. Zhu, H.-T. Zhang, and C. P. Ho, "Robust convex model predictive control for quadruped locomotion under uncertainties," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4837–4854, 2023.

[12] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.