On the Benefits of GPU Sample-Based Stochastic Predictive Controllers for Legged Locomotion

Giulio Turrisi[1], Valerio Modugno[2], Lorenzo Amatucci[1], Dimitrios Kanoulas[2], Claudio Semini[1]

***Abstract*— Quadrupedal robots excel in mobility, navigating complex terrains with agility. However, their complex control systems present challenges that are still far from being fully addressed. In this paper, we introduce the use of Sample-Based Stochastic control strategies for quadrupedal robots, as an alternative to traditional optimal control laws. We show that Sample-Based Stochastic methods, supported by GPU acceleration, can be effectively applied to real quadruped robots. In particular, in this work, we focus on achieving gait frequency adaptation, a notable challenge in quadrupedal locomotion for gradient-based methods. To validate the effectiveness of Sample-Based Stochastic controllers we test two distinct approaches for quadrupedal robots and compare them against a conventional gradient-based Model Predictive Control system. Our findings, validated both in simulation and on a real 21Kg Aliengo quadruped, demonstrate that our method is on par with a traditional Model Predictive Control strategy when the robot is subject to zero or moderate disturbance, while it surpasses gradient-based methods in handling sustained external disturbances, thanks to the straightforward gait adaptation strategy that is possible to achieve within their formulation.**

I. INTRODUCTION

The superior mobility of quadrupedal robots allows them to navigate through uneven landscapes, climb obstacles, and maintain stability on slippery surfaces, showcasing a level of adaptability that closely mirrors the capabilities of their biological counterparts [1]. However, the advantages offered by quadrupedal robots come at the price of increased complexity in designing and implementing effective locomotion controllers. In recent years, the field of robotics has seen a significant surge in the development of innovative control solutions for legged robots, particularly quadrupedal systems [2]. This advancement is fueled by the aim to realize sophisticated locomotion strategies that are capable of seamlessly adapting to diverse and challenging real-world scenarios.

The quest for advanced quadrupedal mobility has pushed researchers to explore a variety of control methodologies, from analytical optimization and Model Predictive Control (MPC) [3] and purely end-to-end Reinforcement Learning strategy [4] to cutting-edge machine learning strategies that blend reinforcement learning with traditional optimization techniques [5].

Sample-Based Stochastic (SBS) methodologies are rapidly gaining traction [6] and represent a valid alternative to control mobile robots. This strategy diverges from traditional control methods by relying on a distribution-based generation and evaluation of a multitude of potential solutions sampled from the search space, rather than exclusively depending on gradient-based optimization techniques.

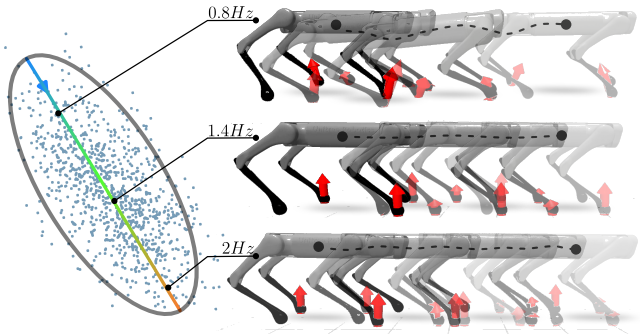

Fig. 1. The illustration depicts the application of a Sample-Based Stochastic (SBS) predictive controller for gait generation in a quadrupedal robot. Each gait sequence, sampled from a distribution, corresponds to a unique set of control actions at various gait frequencies. These sequences are evaluated based on their predicted cost over the forecast horizon. The most promising sequences—referred to as the elite samples—are then used to update the sampling distribution, optimizing the controller's future action selections.

Historically, SBS optimization methods [7] have emerged as a compelling alternative to analytical approaches in instances where gradients are intractable or nonexistent. These methods have proven particularly useful in robotic applications where the complexity of the robot's interactions with its environment precludes the straightforward application of gradient-based techniques. SBS methods can uncover effective strategies for controlling robots in diverse and challenging situations by sampling a wide range of possible solutions and evaluating their performance. We believe that adopting SBS control strategies in quadrupedal robotics could represent a significant advancement, offering a flexible and robust framework for developing locomotion and maneuvering capabilities that can adapt in real-time to the ever-changing demands of real-world environments.

Moreover, the widespread adoption of programming languages and libraries like JAX [8] opens up unprecedented opportunities for implementing real-time SBS control solutions in real robotics platforms. JAX is a high-performance library designed for numerical computing, particularly well-suited for the demands of parallel large-scale computation.

In this study, we show the application of GPU-accelerated, SBS real-time controllers for quadruped robots, demon-

[1] These authors are with the Dynamic Legged Systems Laboratory, Istituto Italiano di Tecnologia (IIT), Genova, Italy. E-mail: name.lastname@iit.it

[2] These authors are with the Department of Computer Science, University College London, Gower Street, WC1E 6BT, London, UK. E-mail: {v.modugno, d.kanoulas}@ucl.ac.uk

This work was partially supported by the UKRI Future Leaders Fellowship [MR/V025333/1] (RoboHike)

strating that SBS methods can match the performance of traditional gradient-based controls while reducing controller complexity. Furthermore, we highlight the SBS controller's adaptability by easily addressing challenges like gait frequency adaptation, a persistent issue in quadrupedal locomotion research [9].

To summarize, our contributions are:

- The first implementation of a GPU-accelerated Sample-Based Stochastic (SBS) control strategy for the direct control of a 12 degrees of freedom quadrupedal robot, fully integrated within a comprehensive locomotion control framework.
- Introduction of a straightforward gait adaptation strategy, illustrating the SBS's ability to optimize robot locomotion under external disturbances.
- A detailed comparative analysis of two SBS control strategies against established gradient-based optimal controllers, underscoring the practical advantages and effectiveness of the SBS approach in real-world scenarios.
- An open-source code repository[1], for the rapid implementation and testing of SBS control strategies for quadrupedal robots.

The paper is organized as follows: Section II reviews related works, providing a foundation and context for our research. Section III delves into SBS control methods and their application to the predictive case. In Section IV, we explore how SBS predictive control methods can be effectively applied for quadrupedal-legged locomotion. Section V presents the experimental results, showcasing SBS controllers on a real Unitree Aliengo[2]. Finally, Section VI concludes the paper, summarizing our findings and suggesting directions for future research.

II. RELATED WORKS

The challenge of achieving effective quadrupedal locomotion has traditionally been addressed through analytical gradient-based methods. For instance, in the approach outlined by [3], the authors develop an optimal control strategy based on linearized rigid body dynamics. This strategy demonstrates the potential for practical application in real quadrupedal robots. In [10] the authors employ a Nonlinear Model Predictive Control (NMPC) to track retargeted animal motions, while in [11] the authors propose an NMPC approach to traverse rough terrain using elevation maps input to optimize footstep locations.

It is important to note that all these analytical methods often depend on complex structures that require the presence of external solvers, such as qpOASES [12], or HPIPM [13], and gradient information. Even though all these methods display great robustness derived from the inherent stability of the optimal control framework, this strength is somewhat offset by their lack of adaptability, stemming from the necessity for continuous and differentiable costs within their formulations.

Another promising avenue for addressing locomotion challenges has been Reinforcement Learning (RL) strategies, which have yielded impressive outcomes in various applications thanks to their ability to handle sparse rewards or costs that are not differentiable. In [14] and [15] the authors propose different RL policies for performing parkour-like motions with real robotic quadrupeds. In [4] the authors introduce an end-to-end RL policy for quadrupeds navigation. Despite the great results achieved by RL approaches, a significant limitation is their lack of generalizability to different tasks without undergoing comprehensive retraining. This constraint means that for each new problem, the RL model often requires a fresh training cycle, which can be resource-intensive. Furthermore, the black-box nature of these methods often presents challenges related to the safe deployment of the policies in the real world.

SBS methods emerge as a powerful hybrid, merging the precision of model-based analytical optimal control techniques with the inherent adaptability of RL approaches. These methods have shown remarkable potential in generating complex behaviors through a limited number of concurrent samples. For instance, in [6], [16], the efficacy of SBS methods in creating sophisticated behaviors is demonstrated only within simulated environments. Several studies have validated the effectiveness of SBS methods on actual robotic platforms [17], [18], highlighting their practical applicability. In [19], the authors attempted to integrate a sample-based controller with RL, but due to the platform's simplicity and its limited degrees of freedom, the application fell short of a comprehensive locomotion scenario. Despite a wide array of methods being successfully applied across various robotic systems, there remains a noticeable gap in their application to locomotion. Finally in [20], the authors introduce an SBS strategy with an auxiliary policy for real-time operation due to CPU limitations on sample size. Conversely, our GPU-accelerated SBS controller allows for direct, real-time robot control without such limitations.

III. SAMPLE-BASED STOCHASTIC OPTIMIZATION

SBS methods, which have been utilized extensively in the last decade in robotics applications [21], [7], often employ the multivariate Gaussian distribution as the cornerstone for exploring the parameters landscape. This distribution, denoted as $\mathcal{N}(\boldsymbol{\theta}, \boldsymbol{C})$, where $\boldsymbol{\theta}$ represents the mean and $\boldsymbol{C}$ the covariance matrix, provides a probabilistic framework for generating parameter samples. Within this context, it is possible to identify a general structure for SBS optimization as shown in Algorithm 1.

The process starts with the **sampling** phase, where K samples $\boldsymbol{\theta}_{k=1\dots K}$ are drawn from the multivariate Gaussian distribution. Following the sample generation, an **evaluation** and **ordering** steps are performed, where each parameter set $\boldsymbol{\theta}_k$ is evaluated using the cost function $J(\boldsymbol{\theta}_k)$. The samples are then ranked in ascending order of their cost values,

[1]code repository: https://github.com/iit-DLSLab/Quadruped-PyMPC
[2]aliengo quadruped robot: https://www.unitree.com/products/aliengo

Algorithm 1 Generic iteration of Sample-Based Stochastic optimizer

1: $\boldsymbol{\theta}_{k=1\dots K} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{C})$ ▷ **Sampling**
2: $J_k = \text{CostEval}(\boldsymbol{\theta}_k)$ ▷ **Evaluation**
3: $\boldsymbol{\theta}_{k=1\dots K}, \leftarrow \text{sort}(\boldsymbol{\theta}_{k=1\dots K}, J_{k=1\dots K})$ ▷ **Sorting**
4: $J_{k=1\dots K} \leftarrow$ extract sorted costs
5: **Update**:
- $\boldsymbol{\theta}^{\text{new}} = \text{UpdateMean}(\boldsymbol{\theta}_{k=1\dots K_e}, J_{k=1\dots K_e})$
- $\boldsymbol{C}^{\text{new}} = \text{UpdateCov}(\boldsymbol{\theta}_{k=1\dots K_e}, J_{k=1\dots K_e})$

prioritizing those that indicate lower costs. Finally, in the **update** phase, the algorithm refines the parameters of the Gaussian distribution, concentrating on the 'elite' samples. This subset consists of the top K_e samples from the sorted list, representing the most promising directions for further exploration. The mean and covariance are updated to guide the subsequent sampling toward more promising regions of the parameter space. To converge to the optimal solution of the cost function, it is necessary to execute these four phases multiple times iteratively.

A. Sample-Based Stochastic Methods for Predictive Control

The essence of applying SBS methods lies in performing a complete rollout over a certain horizon to compute the cost function to optimize w.r.t. to the decision variables. Therefore in the context of predictive SBS with a horizon of N steps, the CostEval$(\cdot)$ function defined for computing J_k associated with the kth sample becomes:

Algorithm 2 Rollout$(\boldsymbol{\theta}_k)$

for $i = 0$ to $N-1$ **do**
  $\boldsymbol{u}_i = \boldsymbol{\pi}(\boldsymbol{\theta}_k, \boldsymbol{x}_i, t_i)$
  $\boldsymbol{x}_{i+1} = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$
  $J_k = J_k + r(\boldsymbol{u}_i, \boldsymbol{x}_i, \boldsymbol{x}_i^r)$
end for
return J_k

where $\boldsymbol{\pi}(\boldsymbol{\theta}_k, \boldsymbol{x}_i, t_i)$ represents a control policy that depends on the sample decision variables $\boldsymbol{\theta}_k$, $\boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$ represents the discretized dynamics of the system to be controlled, $r(\boldsymbol{u}_i, \boldsymbol{x}_i, \boldsymbol{x}_i^r)$ represents the one-step cost associated with the current state-input pair, and $\boldsymbol{x}_i^r$ defines the desired state reference.

Using sampling-based methods in the context of MPC is simple in practice because it does not require the use of any optimizers, like HPIPM or qpOASES. Classical optimizers often come with a host of practical challenges such as scalability issues, handling discontinuities, and dependency on gradient information. On the other hand, SBS methods, being gradient-free, offer the flexibility to compute solutions across a wide range of scenarios, including those with discontinuities, which are particularly prevalent in multi-contact systems.

Even though SBS methods are traditionally viewed as sample-inefficient, this issue is greatly mitigated by leveraging the computational power of modern GPUs. This parallelization does not only enhance efficiency, but also allows for the randomization of other features of the control problem. Such capabilities significantly expand the practical applicability of SBS methods in MPC, as we will illustrate in the subsequent section in the context of legged locomotion.

In a predictive control scenario, for the sake of efficiency, we conduct just a single iteration of Algorithm 1 at every time step. To enhance the search process for successive iterations, we employ a strategy known as *warm-starting*. This entails initiating each new search from the solution obtained in the previous iteration. This significantly accelerates the optimization cycle, ensuring that our methods remain computationally viable even in scenarios demanding real-time decision-making.

In this work, we employed two SBS predictive methods: the Naïve SBS optimizer and the Model Predictive Path integral (MPPI) optimizer [22].

B. Naïve SBS Optimizer

We introduced the Naïve SBS method as a baseline to demonstrate the effectiveness of SBS methods, even when the algorithm's structure is markedly simple. In this approach, we employ a straightforward UpdateMean$(\cdot)$ strategy where, at each iteration, the chosen elite sample is always the best-performing one. To maintain the algorithm's simplicity, we do not update the covariance from one iteration to the next, leaving it unchanged. This decision to keep the covariance constant is crucial for sustaining robust exploration capabilities, which we find to be highly beneficial in this context.

A generic iteration of the Naïve SBS Optimizer is summarized in Algorithm 3, where with $\boldsymbol{\theta}^*$ we define the best sample drawn during one iteration.

Algorithm 3 One iteration of the Naïve Optimizer

$\boldsymbol{\theta}_{k=1\dots K} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{C})$
$J_k = \text{Rollout}(\boldsymbol{\theta}_k)$
$\boldsymbol{\theta}_{k=1\dots K} \leftarrow \text{sort}(\boldsymbol{\theta}_{k=1\dots K}, J_{k=1\dots K})$
Update:
- $\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^*$ ▷ Current best as new mean
- $\boldsymbol{C}^{\text{new}} = \boldsymbol{C}$ ▷ Covariance is unchanged

C. Model Predictive Path Integral Optimizer

The MPPI algorithm originates from fundamental optimal control principles and derives its name from utilizing the Feynman-Kac lemma. This approach reformulates the Hamilton-Jacobi-Bellman (HJB) equations into evaluating an expectation over all possible trajectories (or paths) that the system could take, weighted by their performance of path integral that can be effectively estimated using Monte Carlo methods as shown in [22]. This weighting scheme reflects the principle that trajectories leading to lower costs are more likely to be close to the optimal path. The rollout weighting procedure is defined as

$$\tilde{\omega}_i = \exp\left(-\frac{1}{\lambda} \cdot (J_i - \beta)\right)(\omega_i \cdot \boldsymbol{\theta}_i)$$

$$\Omega = \sum_{i=1}^{K} \tilde{\omega}_i \qquad \boldsymbol{\theta}^{\text{new}} = \sum_{i=1}^{K} (\omega_i \cdot \boldsymbol{\theta}_i)$$

where $\beta = J_1, \quad \lambda = 1$ represent respectively a normalization factor and the temperature parameter of MPPI, $\tilde{\omega}_i$ and $\omega_i = \frac{\tilde{\omega}_i}{\Omega}$ represent the weight and the normalized weight associated with each sample $\boldsymbol{\theta}_i$. A full representation of the MPPI algorithm is shown in Algorithm 4.

Algorithm 4 One iteration of the MPPI Optimzer

$\boldsymbol{\theta}_{k=1\dots K} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{C})$ ▷ Sampling
$J_k = \text{Rollout}(\boldsymbol{\theta}_k)$ ▷ Evaluation
$\boldsymbol{\theta}_{k=1\dots K} \leftarrow \text{sort}(\boldsymbol{\theta}_{k=1\dots K}, J_{k=1\dots K})$ ▷ Sorting
$J_{k=1\dots K} \leftarrow$ extract sorted costs
Update:
- UpdateMean$(\boldsymbol{\theta}_{k=1\dots K}, J_{k=1\dots K})$
- $\boldsymbol{C}^{\text{new}} = \boldsymbol{C}$ ▷ Covariance is unchanged

function UPDATEMEAN$(\boldsymbol{\theta}_{k=1\dots K}, J_{k=1\dots K})$
  $\beta = J_1$ ▷ Best cost after sorting
  $\lambda = 1$
  Initialize $\tilde{\omega}_i$ for each sample
  for $i = 1$ to K **do**
    $\tilde{\omega}_i = \exp\left(-\frac{1}{\lambda} \cdot (J_{k_i} - \beta)\right)$
  end for
  $\Omega = \sum_{i=1}^{K} \tilde{\omega}_i$
  for $i = 1$ to K **do**
    $\omega_i = \frac{\tilde{\omega}_i}{\Omega}$ ▷ Normalize weights
  end for
  $\omega_{k=1\dots,K} \leftarrow [\omega_1, \dots, \omega_K]$
  $\boldsymbol{\theta}^{\text{new}} = \sum_{i=1}^{K} (\omega_{k=1\dots,K} \cdot \boldsymbol{\theta}_{k=1\dots K})$
end function

IV. LEGGED LOCOMOTION WITH SBS PREDICTIVE CONTROLLER

In this section, we demonstrate the applicability of the SBS predictive control framework to the problem of legged locomotion. Specifically, we showcase the versatility of SBS methods in extending beyond basic locomotion control to optimize additional aspects of robot walking, such as gait frequency, which is known to be complex to optimize within a gradient-based approach in real-time [23].

Our proposed control architecture operates by sampling K sets of decision variables, denoted as $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2]$, at each time step. Here, $\boldsymbol{\theta}_1$ controls the method computeContactSequence$(\boldsymbol{\theta}_1)$ which generates different gaits $\boldsymbol{\delta}$, whereas $\boldsymbol{\theta}_2$ determines the behavior of the mapping policy $\boldsymbol{\Gamma} = \boldsymbol{\sigma}(\boldsymbol{\theta}_2)$ that provides the input contact forces $\boldsymbol{\Gamma}$. To evaluate each parameter set $\boldsymbol{\theta}_k$, we calculate the associated costs performing parallel rollout on the GPU. For this, we employ a discretized model $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\delta})$ based upon the Single Rigid Body Dynamics (SRBD) to advance the system dynamics over N steps and a single step cost $r(\boldsymbol{u}_i, \boldsymbol{x}_i, \boldsymbol{x}_i^r)$, which represent the tracking cost between the reference state $\boldsymbol{x}_i^r$ and the current system state $\boldsymbol{x}_i$. Following this evaluation, we update the mean of the Gaussian distribution $\mathcal{N}(\boldsymbol{\theta}, \boldsymbol{C})$, which is then passed to a low-level control block that computes the robot's torques. In parallel, we compute the foothold reference where the robot needs to step in, which is then passed to the next MPC control loop, and the swing trajectory for the legs that are not currently in contact with the ground. This separation of tasks is generally adopted for controllers that utilize the SRBD model. More details are provided in Section IV-A.

A depiction of the methodology is presented in the blocks scheme in Figure 2, where with *Leg Control Framework* we refer to the foothold and swing references, and the low-level control computation routines described above. Finally, a detailed description of the SBS predictive controller adapted for legged locomotion is provided in Algorithm 5.

Algorithm 5 SBS Predictive Controller for Locomotion

Given:
$\boldsymbol{x}_0$ current state estimation
$\boldsymbol{\theta}_{k=1\dots K} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{C})$
for each sample k (parallel GPU execution) **do**
  $J_k = \text{Rollout}(\boldsymbol{\theta}_k, \boldsymbol{x}_0)$ ▷ See Algorithm 2
end for
$\boldsymbol{\theta}_{k=1\dots K} \leftarrow \text{sort}(\boldsymbol{\theta}_{k=1\dots K}, J_{k=1\dots K})$
$[\boldsymbol{\theta}_1^{\text{new}}, \boldsymbol{\theta}_2^{\text{new}}] = \text{UpdateMean}(\boldsymbol{\theta}_{k=1\dots K_e}, J_{k=1\dots K_e})$
$\boldsymbol{C}^{\text{new}} = \boldsymbol{C}$

function $\boldsymbol{\pi}(\boldsymbol{\theta}, t_j)$
  $[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2] \leftarrow \boldsymbol{\theta}$
  $\boldsymbol{\delta}_j \leftarrow \text{computeContactSequence}(\boldsymbol{\theta}_1)$
  $\boldsymbol{\Gamma}_j \leftarrow \boldsymbol{\sigma}((t^{\text{knot}}_{1,\dots,P}, \boldsymbol{\theta}_{2,k}), t_j)$
  $\boldsymbol{u}_j = [\boldsymbol{\Gamma}_j, \boldsymbol{\delta}_j]$
end function

A. Model Formulation

This section outlines the model-based approach for a quadrupedal robot used in the Rollout$(\cdot)$ function, adopting a simplified Single Rigid Body Dynamics (SRBD) model from [24]. This model focuses on the quadruped's core translational and rotational movements, omitting the swinging legs' dynamics. This is a suitable approximation since the bulk of a quadruped's mass is typically in its trunk. The robot's dynamics are described using two reference frames: an inertial frame $\mathcal{W}$, and a body-aligned frame $\mathcal{C}$ at the Center of Mass (CoM), simplifying the inertia tensor representation. The SRBD model is articulated through a state representation in the CoM frame

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{\text{c}} \\ \dot{\boldsymbol{v}}_{\text{c}} \\ \dot{\Phi} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{\text{c}} \\ 1/m \sum_{i=1}^{4} \delta_i \boldsymbol{\Gamma}_i + \boldsymbol{g} \\ \boldsymbol{E}'^{-1}(\boldsymbol{\Phi}) {}_{\mathcal{C}}\boldsymbol{\omega} \\ -{}_{\mathcal{C}}\boldsymbol{I}_{\text{c}}^{-1} ({}_{c}\boldsymbol{\omega} \times {}_{\mathcal{C}}\boldsymbol{I}_{\text{c}})_{\mathcal{C}}\boldsymbol{\omega} + \sum_{i=1}^{4} \delta_i {}_{\mathcal{C}}\boldsymbol{I}_{\text{c}}^{-1} {}_{\mathcal{C}}\boldsymbol{p}_{\text{cf},i} \times {}_{\mathcal{C}}\boldsymbol{\Gamma}_i \end{bmatrix} \tag{1}$$

with m characterizing the robot mass subjected to gravitational acceleration $\boldsymbol{g}$. Moreover, $\boldsymbol{v}_c \in \mathbb{R}^3$ and $\dot{\boldsymbol{v}}_c \in \mathbb{R}^3$, respectively, depict the CoM velocity and acceleration. The interaction between the robot and the environment is

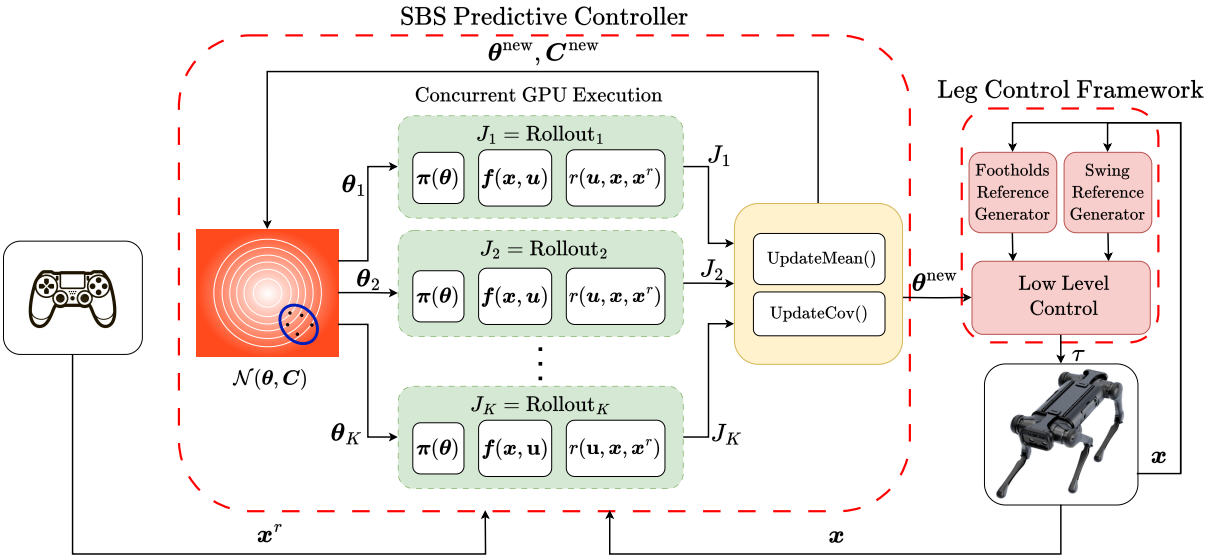

Fig. 2. Block scheme of the proposed control method. From left to right: first, we command the robot with a user-defined reference, which is then passed to the SBS controller that performs a parallel evaluation of the best control actions in GPU; concurrently, a leg control framework generates foothold references, swing trajectories, and low-level control actions, considering the MPC solution.

mediated by the Ground Reaction Forces (GRFs) $\boldsymbol{\Gamma}_i \in \mathbb{R}^3$ at each foot i, while the robot's orientation and motion dynamics are captured by the inertia tensor ${}_{\mathcal{C}}\boldsymbol{I}_c \in \mathbb{R}^{3\times3}$ at the CoM and the base's angular acceleration ${}_{\mathcal{C}}\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$. The positional relationship between the CoM $\boldsymbol{p}_c \in \mathbb{R}^3$ and each foot i's position $\boldsymbol{p}_{f,i} \in \mathbb{R}^3$ is denoted as $\boldsymbol{p}_{cf,i} \in \mathbb{R}^3$. $\Phi = (\phi, \theta, \psi)$ represents the robot body orientation where ϕ, θ, ψ, are the roll, pitch, and yaw respectively, while $\boldsymbol{E}'^{-1}$ is a mapping from the SRBD angles to Euler rates (see [24]).

To model the capacity of each foot i to exert contact forces, binary parameters $\delta_i = \{0, 1\}$ are employed, indicating the presence or absence of ground contact where the index i indicates one of the four legs. The system's state and control are represented by vectors $\boldsymbol{x} = (\boldsymbol{p}_c, \boldsymbol{v}_c, \Phi, {}_{\mathcal{C}}\boldsymbol{\omega})$ and $\boldsymbol{\Gamma} = (\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_4)$. The input to the system model can be defined as $\boldsymbol{u} = [\boldsymbol{\Gamma}, \boldsymbol{\delta}]$, making it possible to formulate the discrete dynamics of the robot as:

$$\boldsymbol{x}_{j+1} = \boldsymbol{f}_i(\boldsymbol{x}_j, \boldsymbol{u}_j),$$

where $\boldsymbol{\delta}_j$ encapsulates the contact state of all four legs at the jth timestep, which normally, in model-based controllers, are pre-computed to attain a tractable optimization problem (see Section IV-C).

In the following, we show how we parametrize the GRFs and the contact parameters $\boldsymbol{\delta}_j$ to solve the aforementioned control problem efficiently.

B. GRFs Parametrization

In this study, we implement a simple parametrization to represent the actual control actions $\boldsymbol{\Gamma}_i$ for input into the SRBD model over the prediction horizon N. The introduction of this intermediate representation serves a dual purpose: firstly, it enables the projection of the control variable across the prediction horizon into a more manageable, lower-dimensional space; secondly, it inherently ensures the smoothness of the control actions. As in [6], our approach utilizes cubic splines for this task. Specifically, within each sample $\boldsymbol{\theta}_k$, the decision variable $\boldsymbol{\theta}_{2,k} \in \mathbb{R}^P$ represents a time-indexed sequence of P knots, each allocated to specific time instants $t^{\text{knot}}_{1,\ldots,P}$.

For any given input time t, the spline's evaluation is then determined as

$$\boldsymbol{\Gamma} = \boldsymbol{\sigma}((t^{\text{knot}}_{1,\ldots,P}, \boldsymbol{\theta}_{2,k}), t) \tag{2}$$

and its output value is then constrained to respect friction cone constraints to guarantee non-slipping conditions.

C. Gait Optimization

Gradient-based methods for legged locomotion often precompute the contact status of each leg as a fixed vector $\boldsymbol{\delta}$ due to the challenges posed by Linear Complementary Constraints, which complicate analytical gradient computations [25]. To address this, fixed periodic gaits like trotting or pacing are commonly used for their simplicity and effectiveness, despite the trade-off between robustness and energy efficiency [26]. While some studies have attempted to optimize gait periodicity using heuristics, these optimizations typically occur outside the control loop [9], [27].

SBS controllers, however, are not limited by the constraints of gradient-based methods, allowing for the direct optimization of gait parameters within the MPC framework. A periodic gait is characterized by the step frequency f_s and the duty factor D_f, defining the stance and swing times for each leg as $T_{st} = \frac{D_f}{f_s}$ and $T_{sw} = \frac{1-D_f}{f_s}$, respectively.

This work optimizes step frequency f_s by discretizing it over a fixed range and parametrizing it with $\boldsymbol{\theta}_1$. The function computeContactSequence$(\boldsymbol{\theta}_1)$ calculates lift-off events and $\boldsymbol{\delta}$ for the MPC horizon. To promote energy efficiency, we introduce a regularization term in J for $\boldsymbol{\theta}_1$ towards a nominal stepping frequency.

D. Leg Control Framework

The outer control loop, operating concurrently with the MPC, consists of three modules: a *foothold reference generator*, a *swing reference generator*, and a *low-level* torque command module.

The foothold reference generator calculates the stepping points for each leg using:

$$\boldsymbol{p}_{f,i} = \boldsymbol{p}_{hip,i} + \frac{T_{st}}{2}\boldsymbol{v}_c^d + \sqrt{\frac{\boldsymbol{p}_{c,z}}{g}}(\boldsymbol{v}_c - \boldsymbol{v}_c^d), \tag{3}$$

where $\boldsymbol{p}_{hip,i}$ represents the hip position, $\boldsymbol{v}_c^d$ the desired velocity, and $\boldsymbol{p}_{c,z}$ the height of the CoM. The last term in (3) adjusts footholds for handling external disturbances.

Swing trajectories for legs in the air are generated by the swing reference generator using cubic splines, linking lift-off points to the touchdown points $\boldsymbol{p}_f$ with a nominal stepping height.

The low-level module translates Ground Reaction Forces from Algorithm 5 into motor torques for stance legs as $\boldsymbol{\tau}_{st} = -\boldsymbol{J}^\top(\boldsymbol{q})\boldsymbol{\Gamma}$, and for swing legs using inertia, Jacobian matrices, and PD control for foot trajectory adherence.

These components ensure dynamic, disturbance-compensated locomotion with precise control over foot placement and swing motion.

V. RESULTS

We validated our proposed controllers through simulations and real-world experiments on the 21kg Unitree Aliengo robot. We compared three controllers:

1) acados [28]: an optimization-based approach using the same model from Section IV-A,
2) Naïve Optimizer: described in Section III-B,
3) MPPI: detailed in Section III-C.

Their performance was evaluated under various disturbances in Mujoco simulations [29] and actual hardware tests. All methods were tested with a horizon length of $N = 12$, a discretization time of 0.02 seconds, and a trotting gait using a nominal stepping frequency of $f_s^n = 1.3$Hz. Acados ran on an Intel 13700H CPU, while the SBS controllers utilized an Nvidia 4050 mobile GPU that can draw a maximum of 30W of power, making it ideally suitable for being carried directly on board a quadruped robot.

We used a quadratic cost function for defining the control task

$$J = (\boldsymbol{x} - \boldsymbol{x}^r)^\top \boldsymbol{Q}(\boldsymbol{x} - \boldsymbol{x}^r) + (\boldsymbol{u} - \boldsymbol{u}^r)^\top \boldsymbol{R}(\boldsymbol{u} - \boldsymbol{u}^r),$$

where $\boldsymbol{Q}$, $\boldsymbol{R}$, are positive diagonal weighting matrices.

For gait optimization, we regularized the stepping frequency towards a nominal, more energy-efficient value, using

$$J = J + \rho(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_1^r)^2,$$

with ρ as a scalar weight and $\boldsymbol{\theta}_1^r = f_s^n$. The variable $\boldsymbol{\theta}_1$ ranged between $[f_s^n, 2.4]$Hz, and sampled using a uniform distribution. We choose to discretize for simplicity $\boldsymbol{\theta}_1$ with three different values, such as f_s^n, 2.0, 2.4, with f_s^n defining a nominal more energy-efficient gait while 2.4 a more robust one, but a more fine-grained discretization step can be employed at need.

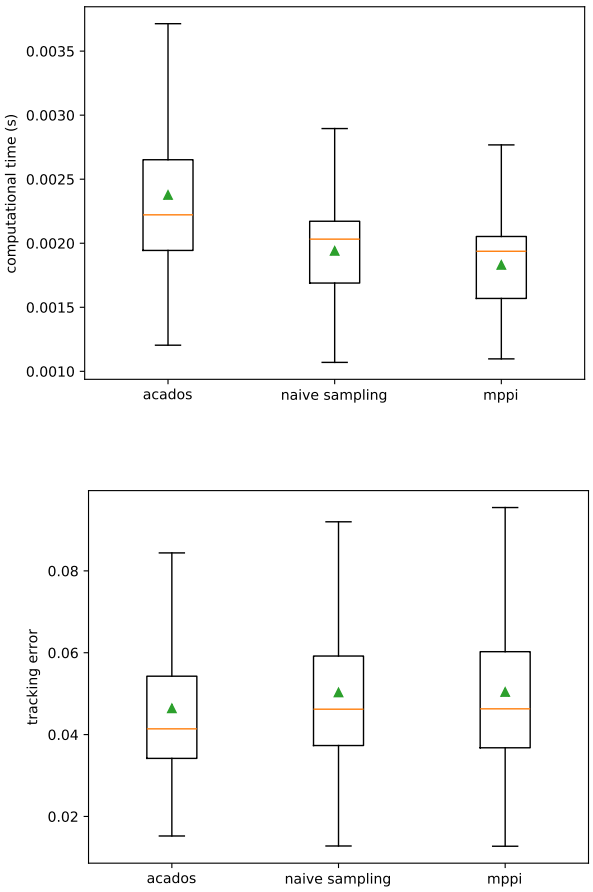

Fig. 3. Comparison between acados, naïve optimizer, and mppi, commanding to the robot a forward linear velocity of 0.5 m/s. Top: box plot of the computation time (SBS methods perform 10000 rollouts at each time step); bottom: box plot of the tracking error. The box extends from the first to the third quartile of the data, with the orange line and the green triangle that depict the median and the mean values, respectively.

A. Simulation Results without Gait Adaptation

We evaluated the computational efficiency and tracking accuracy of each control method, with SBS methods optimizing over 10,000 rollouts. We aim to demonstrate the practicality of SBS methods, benefiting from simple implementation without dependency on QP solvers, as detailed in Section III-A. Simulations involved random external disturbances (wrenches within +/- 5N/Nm) applied to the robot CoM with a duration of 2 seconds every 2 seconds.

Figure 3 shows competitive computational times for all methods and slightly quicker execution for SBS approaches, advantageous for minimizing control delays. In our case, MPPI is faster than (2) since in the last we sample from multiple Gaussian distributions. Tracking performance under a forward velocity command of 0.5m/s was comparable across methods, with acados (1) marginally outperforming the Naïve approach (2) and MPPI (3) as shown in Figure 3 thanks to the more fine-grained control action obtained by exploiting the gradient information.

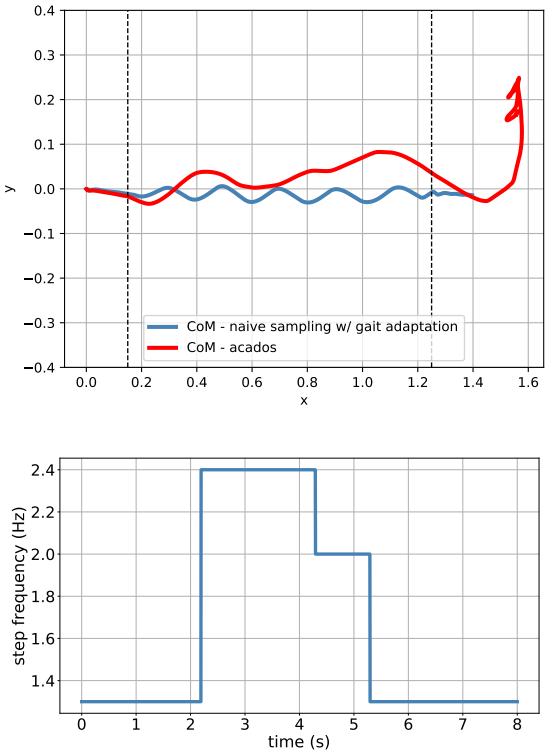

Fig. 4. Comparison between acados and naïve optimizer with gait adaptation, under a strong lateral disturbance (40N) applied for 3 seconds. Top: the CoM evolution under the two control laws, with the dotted line that represents the position range (for the SBS method) in which the external disturbance is applied to the robot; bottom: the step-frequency adaptation profile achieved with the naïve optimizer.

During our tests, we have observed that with a large sample size, differences between methods (2) and (3) are less noticeable, but become significant with fewer rollouts. Hence, in the following, we compare acados and Naïve sampling for brevity.

B. Simulation Results with Gait Adaptation

The straightforward simplicity of sampling-based methods and their capability of optimizing over zero-gradient information, as explained in Section IV-C, make sampling-based controllers an appealing method for legged locomotion. To validate this claim, in Figure 4, we compare the CoM trajectory of Aliengo using acados and naïve sampling with gait adaptation using a strong later push of 40N applied at the CoM of the robot. In this simulation, we furthermore commanded a lateral velocity of 0.1 m/s to the system. As can be seen in Figure 4 (bottom), naïve sampling can optimize the stepping frequency at need, increasing the overall robustness of the controller and avoiding the loss of stability of the robot which occurs with the gradient-based controller. In fact, after the application of the disturbance at around second 2, the stepping frequency is automatically increased, and after the disturbance vanishes (around second 4.2), the frequency is slowly restored to the nominal value f_s^n.

We confirmed this result statistically by analyzing the success rate of methods (1), (2) with and without gait adaptation over 50 different episode simulations, where we applied to the robot CoM random wrench disturbances in the range of +/- 20 [N,Nm]. In Table I, we report the obtained results, showing that the gait adaptation strategy embedded inside the SBS controller greatly increases the system's capability to prevent accidental falls and mean tracking errors. The reader can refer to the accompanying video[3] to visualize the simulated results.

TABLE I
STATISTICAL ANALYSIS OVER 50 DIFFERENT EPISODES

Description	Mean Cost	Success Rate %
acados	0.12	32
naïve sampling	0.14	37
naïve sampling w/ gait adaptation	**0.08**	**80**

C. Experiment Results on Real Quadruped

The naïve sampling controller was validated on the real robot on two different gaits, such as trotting and pacing (see the accompanying video). Furthermore, we replicated the simulation results shown in Figure 4, perturbing both acados and the sampling controller with strong lateral pushes. As can be seen in Figure 5, we obtain a similar behavior to the one achieved in simulation, with the sampling-based controller able to modulate the step frequency at need. In the accompanying video, we report on the same experiment performed using method (1).

VI. CONCLUSION

In conclusion, this paper demonstrated the efficacy of SBS control strategies for quadrupedal robots, offering a robust alternative to conventional gradient-based controls. Our study highlights the SBS methods' success in real-world applications, particularly in achieving real-time gait frequency adaptation—a significant hurdle for gradient-based approaches.

Comparative testing against traditional MPC approaches underline the SBS controllers' comparable performance under minimal or moderate disturbances, and superior handling of severe disturbances due to their flexible formulation that allows the optimization of other controller aspects like the gait frequency.

Future work will focus on applying SBS controllers with the full dynamics of quadrupedal systems for direct torque control and integrating visual feedback to enhance their operational capabilities in complex environments.

REFERENCES

[1] P. Biswal and P. K. Mohanty, "Development of quadruped walking robots: A review," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 2017–2031, 2021.

[2] G. Lunardi, T. Corbères, C. Mastalli, N. Mansard, T. Flayols, S. Tonneau, and A. D. Prete, "Reference-Free Model Predictive Control for Quadrupedal Locomotion," *IEEE Access*, vol. 12, pp. 689–698, 2024.

[3] 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] video: https://youtu.be/E4iz9fZsfxA

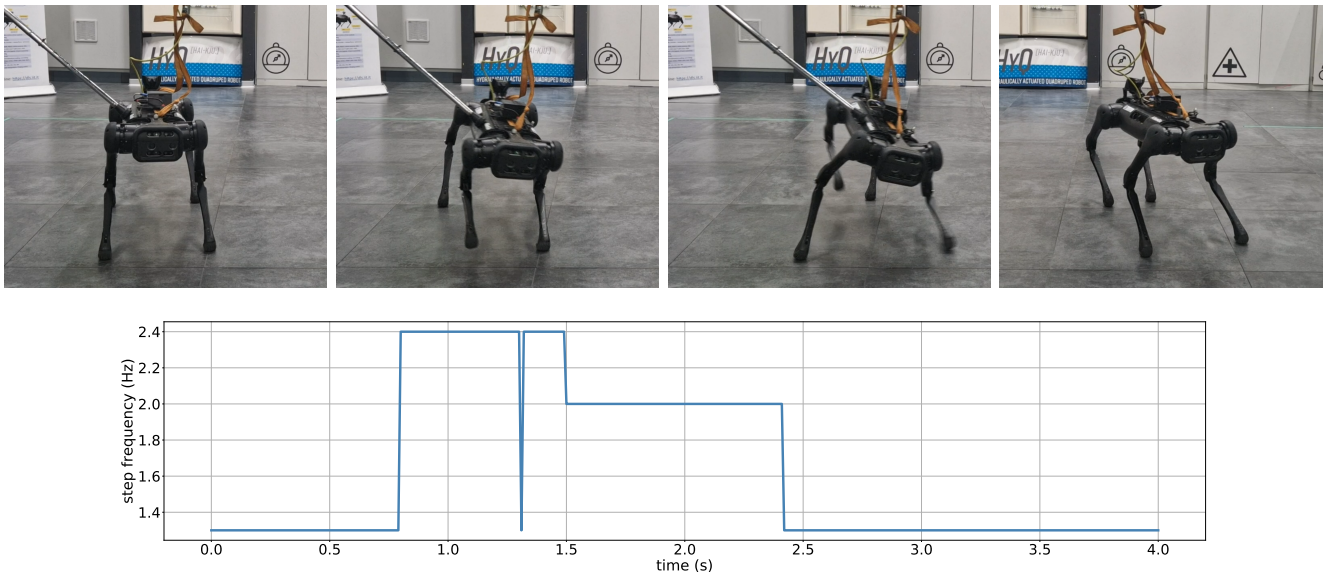

Fig. 5. Top: snapshots of the robot response during trotting, controlled by the naïve sampling technique, to an external disturbance. Bottom: step-frequency adaptation profile during the same experiment.

[4] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2497–2503.

[5] S. Omar, L. Amatucci, V. Barasuol, G. Turrisi, and C. Semini, "Safesteps: Learning safer footstep planning policies for legged robots via model-based priors," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023, pp. 1–8.

[6] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo," 2022.

[7] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," *ArXiv*, vol. abs/1206.4621, 2012.

[8] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[9] L. Amatucci, J.-H. Kim, J. Hwangbo, and H.-W. Park, "Monte carlo tree search gait planner for non-gaited legged system control," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4701–4707.

[10] D. Kang, F. De Vincenti, N. C. Adami, and S. Coros, "Animal motions on legged robots using nonlinear model predictive control," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 11 955–11 962.

[11] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3402–3421, 2023.

[12] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, pp. 327 – 363, 2014.

[13] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020, 21st IFAC World Congress.

[14] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv:2309.14341*, 2023.

[15] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning (CoRL)*, 2023.

[16] D. M. Asmar, R. Senanayake, S. Manuel, and M. J. Kochenderfer, "Model predictive optimized path integral strategies," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3182–3188, 2022.

[17] J. Pravitra, E. Theodorou, and E. N. Johnson, "Flying complex maneuvers with model predictive path integral control," in *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, 2021.

[18] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, pp. 1603–1622, 2017.

[19] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Conference on Robot Learning (CoRL)*, 2019.

[20] J. Carius, R. Ranftl, F. Farshidian, and M. Hutter, "Constrained stochastic optimal control with learned importance sampling: A path integral approach," *The International Journal of Robotics Research*, vol. 41, pp. 189 – 209, 2021.

[21] V. Modugno, G. Nava, D. Pucci, F. Nori, G. Oriolo, and S. Ivaldi, "Safe trajectory optimization for whole-body motion of humanoids," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 763–770.

[22] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning." *Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[23] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[24] N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad, "Model predictive control with environment adaptation for legged locomotion," *IEEE Access*, vol. 9, pp. 145 710–145 727, 2021.

[25] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[26] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.

[27] C. Boussema, M. J. Powell, G. Bledt, A. J. Ijspeert, P. M. Wensing, and S. Kim, "Online gait transitions and disturbance recovery for legged robots via the feasible impulse set," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1611–1618, 2019.

[28] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, pp. 147 – 183, 2019.

[29] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.