Gait Switching and Enhanced Stabilization of Walking Robots with Deep Learning-based Reachability: A Case Study on Two-link Walker

Xingpeng Xia*,[1], Jason J. Choi*[2], Ayush Agrawal[2], Koushil Sreenath[2], Claire J. Tomlin[2], and Somil Bansal[3]

Abstract— **Learning-based approaches have recently shown notable success in legged locomotion. However, these approaches often lack accountability, necessitating empirical tests to determine their effectiveness. In this work, we are interested in designing a learning-based locomotion controller whose stability can be examined and guaranteed. This can be achieved by verifying regions of attraction (RoAs) of legged robots to their stable walking gaits. This is a non-trivial problem for legged robots due to their hybrid dynamics. Although previous work has shown the utility of Hamilton-Jacobi (HJ) reachability to solve this problem, its practicality was limited by its poor scalability. The core contribution of our work is the employment of a deep learning-based HJ reachability solution to the hybrid legged robot dynamics, which overcomes the previous work's limitation. With the learned reachability solution, first, we can estimate a library of RoAs for various gaits. Second, we can design a one-step predictive controller that effectively stabilizes to an individual gait within the verified RoA. Finally, we can devise a strategy that switches gaits, in response to external perturbations, whose feasibility is guided by the RoA analysis. We demonstrate our method in a two-link walker simulation, whose mathematical model is well established. Our method achieves improved stability than previous model-based methods, while ensuring transparency that was not present in the existing learning-based approaches.**

I. Introduction

A. Motivation & Key Ideas

Locomotion is one of the fundamental modes of mobility for robots. The recent success of deep reinforcement learning (RL) in achieving robust, stable walking for legged robots [1], [2], [3] underscores the strength of learning-based policies. Despite this success, the mechanisms underlying the RL policies remain opaque. For example, the RL policy in [1] stretches the leg when the robot is pushed to keep its balance. However, this action, derived from a black-box neural network, leaves us unable to explain the underlying rationale. Generally, we seek more than just empirical observations to determine why and when a policy works.

To address this limitation, our study focuses on developing explainable learning-based policies for stable locomotion. Specifically, we seek to elucidate

1) the region in which the robot state can be perturbed without compromising its ability to return to a stable gait,
2) the rationale for selecting a feasible target gait among the library of candidate gaits,
3) the way of evaluating the feasibility of gait transition.

By designing a learning-based policy which accompanies answers to these questions, we can furnish learning-based locomotion with a layer of interpretability and assurance.

The central idea of our approach to tackle these questions is to resort to the region of attraction (RoA) concept. The RoA defines the state space region from which a system can stabilize to a desired attractor—in the case of legged robots, a periodic walking gait. A policy informed by the RoAs can determine when stabilization is feasible and how. Although RoA analysis is a classic topic in controls, its application to legged robots has been hindered by the robots' complex hybrid dynamics, with a few notable exceptions [4], [5], [6]. The method in [6] stands out for recovering largest portion of the RoA through reachability analysis, yet it is significantly limited by the computational demands of solving Hamilton-Jacobi (HJ) partial differential equations (PDEs) numerically.

Our approach builds upon [6] but circumvents the computational hurdles of the numerical method by leveraging neural networks to approximate the HJ PDE solutions. Adapting the approach in [7], we tailor the learning of HJ PDE solutions to the hybrid dynamics and diverse gaits of legged robots. Our walking policy is designed around the learned HJ PDE solution. By ensuring that it stabilizes to a feasible gait based on the RoA evaluation, we not only significantly enhance the stabilization capabilities of the robot, but also provide a transparent rationale behind the controller's decisions.

B. Contributions

The main novelty of this work is the first-ever application of the deep learning-based reachability to hybrid system dynamics and locomotion control design. Its primary purpose is to provide users with an RoA estimate from which the robot can stabilize to a set of stable walking gaits. With the learned value function, the solution of the HJ PDE, we are able to estimate RoAs of individual gaits, and their union shapes the feasible space for stable walking.

Second, we design a control policy that stabilizes to an individual gait, based on the learned HJ PDE solution. Rather than resorting to the optimal reachability policy that minimizes the Hamiltonian of the learned value function [6], we devise a one-step predictive control design. It seeks the best control in minimizing the value function at the next timestep,

* The first two authors contributed equally. This work is supported in part by the NSF Grant CMMI-1944722, the NSF CAREER Program under award 2240163, the NASA ULI on Safe Aviation Autonomy, and the DARPA Assured Autonomy and Assured Neuro Symbolic Learning and Reasoning (ANSR) programs. The work of Jason J. Choi received the support of a fellowship from Kwanjeong Educational Foundation, Korea.

[1]Tsinghua University, Beijing, China. [2]University of California, Berkeley, CA, US. [3]University of Southern California, Los Angeles, CA, US `xiaxp19@mails.tsinghua.edu.cn`, `jason.choi@berkeley.edu`, `somilban@usc.edu`

which mitigates the learning errors of the neural network more effectively and achieves enhanced stabilization.

Finally, we devise an effective gait switching strategy whose feasibility is provable by the RoA analysis. This gait transition is inevitable, especially when the robot is subjected to unmodeled perturbations that lead the system state to escape the RoA of the current gait. By evaluating RoAs, we can cleverly select which gait the robot switches to, and continue stable walking without falling.

We demonstrate the aforementioned contributions in a simple two-link walker robot simulation. Although the dynamics of the robot is simplistic as a legged robot, its low dimensionality allows us to conduct a detailed analysis of the verified RoAs and access to its detailed mathematical model allows a fair comparison to existing model-based control methods. Our experiment reveals that 1) RoAs of the gaits can be estimated accurately with the proposed learning-based framework, 2) the designed controller achieves a significantly higher success rate than model-based stabilization controllers like the hybrid zero dynamics-based input-output (IO) linearization controller [8], and nonlinear model predictive controller (NMPC) [9], and 3) gait transitions can be conducted stably when the robot is pursing a sequence of varying gaits or when it is subjected to strong perturbations.

C. Related work

1) Learning-based approach for locomotion: The success recipe for deep RL-based locomotion includes many elements such as high-fidelity simulations [10], good composite reward design [11], appropriate training schemes like domain randomization [1] or curriculum learning [11], and a suitable model architecture [12]. All in all, carefully trained RL policies achieved state-of-the-art performance in locomotion.

2) Model-based analysis and control design for locomotion: Numerous mathematical tools are proposed for the analysis of stability of locomotion—Lyapunov methods [4], [5], Poincaré map [13], capturability [14], contraction analysis [15], Riemannian partition [16], and reachability [6], [17], [18]. Although each method has its unique strengths and drawbacks, verifying an invariant and stable domain is the central theme of most methods.

The control design approaches also vary from IO linearizaiton [19] to NMPC [9], [20], [21], [22] and many others. The common challenge in these designs is addressing the varying gait sequence and the associated contact dynamics. The main benefit of our approach is that the hybrid dynamics are already accounted for in the construction of the value function, and no further treatment of the contacts is needed for the control synthesis.

3) Combination of model & learning-based locomotion: Combinations of learning and Lyapunov-based constraints are explored in [23], [24], [25]. A residual model of the robot dynamics is learned online in [26]. Combinations of RL and NMPC are also proposed in [27], [28]. The vision of these works is to combine the strengths of model-based design and machine learning. In our work, we hope to enhance explainability of the learning-based controller with the reachability-informed design.

4) Physics-informed Machine Learning for Control: The approach we undertake, which learns solutions of the HJ PDE with neural networks, falls into the category of physics-informed machine learning for control [29]. Solving any PDEs numerically is inherently subjected to the curse of dimensionality [30]; thus, physics-informed neural network (PINN) [31] is suggested as an alternative for finding approximate solutions. Its applications to solving HJ PDEs are proposed in [32], [33], [7], and we develop our method based on [7] which is tailored for reachability.

II. Problem Description

A. Problem Setup

1) Hybrid dynamics model of walking robots: We describe the walking dynamics of the legged robot as

$$\dot{x} = f(x,u), \qquad x \notin S \tag{1a}$$

$$x^{+} = \Delta\left(x^{-}\right), \qquad x^{-} \in S, \tag{1b}$$

where $x \in \mathbb{R}^n$ is the state and $u \in \mathcal{U}$ is the control input. S indicates the switching surface where the reset map Δ is applied; under the reset event at time t, the state $x^{-} := \lim_{\tau \nearrow t} x(\tau)$ instantaneously shifts to x^{+}. The state x consists of generalized coordinates q and its time derivative; $x = [q;\ \dot{q}]$. The trajectory of the robot is composed of a set of continuous trajectories driven by the continuous-mode dynamics f, and discontinuous jumps whenever the state hits the switching surface S, which captures the impact event between the swinging foot and the ground. The continuous mode dynamics (1a) can be derived from Euler-Lagrangian mechanics of the robot, constrained by the contact force at the stance leg. The reset map (1b) can be modeled based on the rigid impact model and the relabeling of the coordinates for switching the swing and stance legs. For more details of the derivation of the dynamics, please refer to [8, Sec.3.4].

2) Stable hybrid limit cycle walking gaits: A stable walking gait of the robot is represented as a non-trivial T_p-periodic solution of the system (1), denoted as $x^*(\cdot)$, a trajectory in time, which undergoes resets at times kT_p for positive integer k. We call $O := \{x \mid \exists t \geq 0, x = x^*(t)\}$ a hybrid limit cycle of the system. The limit cycle is assumed to be forward invariant and stable under some *baseline controller* $\pi_0 : \mathbb{R}^n \to \mathcal{U}$, in a small neighborhood around O.

3) Parametrized walking gaits: There might exist multiple stable hybrid limit cycles of the system, each corresponding to various walking gaits of the robot. Each gait is conditioned on various gait parameters, such as average forward velocity or step length. We denote this gait parameter vector as β, which will result in a parameter-conditioned hybrid limit cycle walking gait $O(\beta)$. We will often refer to the gait parameter β itself as the "gait" in the manuscript for the sake of brevity. Finally, we denote $\mathcal{B}$ as a set of gait parameters which result in a feasible stable walking gait, meaning that for all $\beta \in \mathcal{B}$, the limit cycle $O(\beta)$ exists and is stable.

4) Unmodeled perturbations: We model perturbations not captured in (1), such as push or impact with other objects as a drift (or instantaneous jump) of the robot state. We assume that the perturbations we deal with are temporary.

B. Objectives

The overall objective is twofold. First, for each gait, we seek to compute a region of state space $\Omega \subseteq \mathbb{R}^n$ around $O(\beta)$, from which there exists a feedback control law that asymptotically stabilizes the system to $O(\beta)$. We call $\Omega(\beta)$ the (asymptotically) stabilizable region, or *region of attraction (RoA) for $O(\beta)$*. The goal is to verify in which robot state it is possible to stabilize to an individual gait, and the associated stabilizing feedback control law $\pi : \mathbb{R}^n \to \mathcal{U}$.

Next, we are interested in *when transitioning to a new gait is possible or necessary*, when the robot state is perturbed or if the user wants to command the robot to change its gait. That is, given a state x, we are interested in finding $\mathcal{B}_{\text{feas}}(x) \subseteq \mathcal{B}$ such that for all gait parameters in the set, $\beta \in \mathcal{B}_{\text{feas}}(x)$, the state is inside the RoA of the corresponding gait, $x \in \Omega(\beta)$. Then, a gait transition is necessary if the current walking gait of the robot β is not in $\mathcal{B}_{\text{feas}}(x)$, to ensure stability of the walking. Upon transition, the robot has to select one of the gait β that is included in $\mathcal{B}_{\text{feas}}(x)$.

III. Background

A. HJ reachability analysis for RoA computation

In this section, we first present an overview of Hamilton-Jacobi (HJ) reachability analysis for continuous systems [34]: the dynamics given by $\dot{x} = f(x,u)$ without the reset. Let $\xi^u_{x,t}(\tau)$ denote the state at time τ by starting at initial state x and initial time t, and applying input signal $u(\cdot)$ over $[t,\tau]$. Given a *target set* $L \subset \mathbb{R}^n$, the *Backward Reachable Tube (BRT)* of L is defined as the set of initial states from which there exist a control signal under which the system will eventually reach L within the time horizon T:

$$\begin{aligned} BRT(L;T) := \{x \mid\ & \exists u : [-T,0] \to \mathcal{U}, \\ & \exists \tau \in [-T,0], \xi^u_{x,-T}(\tau) \in L\}. \end{aligned}$$

For our problem, we define the target set as a small neighborhood of the gait $O(\beta)$, denoted as $L(\beta)$, such that the baseline controller π_0 can stabilize to the limit cycle from anywhere inside L. Such a neighborhood region exists for any asymptotically stable attractor [35]. If we set $O(\beta)$ directly as the target set, only the states that can achieve *finite-time* convergence to the gait can be verified, excluding states that are not finite-time stabilizable but still asymptotically stabilizable to the gait. By the definition of BRTs, any state x in $BRT(L(\beta);T)$ is reachable to $L(\beta)$ in finite-time, and once it reaches the target set, it can be stabilized to $O(\beta)$. Therefore, for every state that can be verified as an element of a finite-time BRT of $L(\beta)$, we can conclude that it is an element of $\Omega(\beta)$, the RoA of the gait. In fact, if $T \to \infty$, the BRT recovers the full RoA of the given gait, i.e. $\lim_{T\to\infty} BRT(L(\beta);T) = \Omega(\beta)$. Thus, the BRTs computed for long enough time horizon can be considered as a maximal estimation of the RoAs of the gaits.

In HJ reachability, the computation of BRT is considered an optimal control problem, which can be solved with dynamic programming. First, a signed distance function to L, $l(x)$, is defined whose zero-sublevel set is L, i.e. $L = \{x : l(x) \leq 0\}$. Here, the dependency on β is dropped for simplicity. Next, we define the minimum signed distance to L over time along the trajectory as

$$J(t,x,u(\cdot)) = \min_{\tau \in [-t,0]} l(\xi^u_{x,-t}(\tau)). \tag{2}$$

When $l(\tau) \leq 0$, the system is inside L at time τ, thus, reaches the target set. Thus, for the goal of computing the BRT, we compute the optimal control that minimizes this distance (so that it achieves a non-positive value of J). As such, we define the value function as

$$V(t,x) = \inf_{u(\cdot)} \Big\{ J\Big(t,x,u(\cdot)\Big) \Big\}. \tag{3}$$

The value function in (3) can be computed using dynamic programming, which results in the following Hamilton-Jacobi partial differential equation (HJ PDE) [36]:

$$\min\big\{ -D_t V(t,x) + H(t,x),\ l(x) - V(t,x) \big\} = 0, \tag{4}$$

with the initial value function $V(0,x) = l(x)$, where

$$H(t,x) := \min_u \nabla V(t,x) \cdot f(x,u). \tag{5}$$

D_t and ∇ represent the time and spatial gradients of the value function. H is the Hamiltonian that encodes the role of dynamics and the optimal control input. Once $V(t,x)$ is obtained by solving the HJ PDE, the BRT is given as the zero sub-level set of the value function $BRT(L(\beta);t) = \{x \mid V(t,x) \leq 0\}$. The corresponding optimal control for reaching the target set L is derived as

$$\pi^*(t,x) = \arg\min_u \nabla V(t,x) \cdot f(x,u). \tag{6}$$

B. HJ reachability for walking robots

We next summarize the extension of the reachability framework in [6], to account for discontinuous state resets. The value function in the presence of state resets, (1b), can be obtained by solving a constrained version of the HJ PDE:

$$\min\Big\{ -D_t V(t,x) + H(t,x),\ l(x) - V(t,x) \Big\} = 0, \quad x \notin S, \tag{7a}$$
$$V(t,x) = V(t,\Delta(x)), \quad x \in S, \tag{7b}$$

with the initial value function

$$V(0,x) = l(x) \text{ if } x \notin S, \quad V(0,x) = l(\Delta(x)) \text{ if } x \in S. \tag{8}$$

In words, the value function can be obtained by solving the usual HJ PDE for the states that are not on the switching surface, and for the states that are on the switching surface, the value is given by that of the corresponding post-reset state. This is because if the state is on the switching surface, it will instantaneously change to the post-reset state. Since (7) reasons about the state resets, the obtained value function and the associated optimal controller (6) implicitly exploit the reset map to reach the target set as quickly as possible.

The HJ PDEs in (4) and (7) can be effectively solved using numerical algorithms like level set methods described in [36], [37]. The only additional step in solving (7), compared to (4) is to enforce (7b) at every timestep, which does not add computational complexity to the original algorithm. Please refer to [6] for more details of the numerical algorithm.

C. Limitations of numerical methods for HJ reachability

Applying the numerical algorithms for solving the HJ PDE to legged robot dynamics encounters several key obstacles. First, since the algorithm is in essence a brute force dynamic programming, it is practically infeasible to be applied to realistic legged robots due to the curse of dimensionality [30]. The computational load and the memory requirements grow exponentially with respect to the state dimension.

Next, the numerical stability of the PDE solutions is tightly coupled with the maximal norm of the Hamiltonian (5) [38]. This necessitates a denser computational grid for robots with stiffer dynamics, characterized by larger magnitudes of their vector fields. Legged robots exhibit stiffer behaviors than simpler systems like mobile robots or near-hover drones [34], previously addressed by HJ reachability. Consequently, balancing computational time against numerical accuracy often results in a value function whose gradient, critical for determining the optimal control, suffers poor accuracy.

Finally, the computation of BRTs for legged robots presents another efficiency challenge when considering the gait parameter β. Solving for BRTs individually across each gait parameter leads to considerable computational redundancy and memory waste. This is because BRTs for neighboring gaits tend to exhibit similar shapes, as we will see later in the paper. In this regard, a parametric approach to representing the gait BRTs can address these inefficiencies effectively. By computing the BRTs in a unified process that accounts for all possible gait parameters, only the shape parameters of the BRT need to be stored and can significantly reduce the computation and memory requirements.

D. Deep learning-based reachability for continuous systems

In light of the limitations of the brute force numerical methods in solving the HJ PDE, a deep learning-based approximate solution for HJ reachability, named DeepReach, was proposed in [7] for continuous systems. DeepReach utilizes sinusoidal activation functions [39] to represent the value function and employs a loss function that learns the HJ PDE solution in self-supervised fashion.

During the training, DeepReach samples a batch of time and state samples from the target domain. The loss function for a given sample (t_i, x_i) where i is the index of the sample, is given as $h(t_i, x_i) = \lambda_1 h_1 + \lambda_2 h_2$ where

$$\begin{aligned} h_1 =& \big|\min\{-D_t V_\theta(t_i, x_i) + H(t_i, x_i), \\ & \qquad l(x_i) - V_\theta(t_i, x_i)\}\big|, \\ h_2 =& |l(x_i) - V_\theta(0, x_i)|. \end{aligned} \tag{9}$$

h_1 evaluates the left hand side of (4) at the training sample, and h_2 evaluates the initial condition of the PDE.

Since h_1 depends on gradients of the value function, the neural network should not only approximate the value function well but also its gradients. The widely popular ReLU-based neural networks struggle to accurately represent their gradients, which can lead to a poor approximation of the value function. Thus, DeepReach employs a sinusoidal activation function [39], which is known to produce accurate gradients due to its inherent differentiability. After the training, the BRT can be represented by the zero-sublevel set of the learned value function.

IV. Our Method

A. Extension of DeepReach for learning gait BRTs

Our method mainly extends the DeepReach framework to address the parameter-conditioned varying target gaits and the hybrid dynamics of the legged robots. In our framework, we incorporate four key features not present in the original DeepReach work: 1) parameterization of the value function with respect to the gait parameter β, 2) an additional loss term that captures the condition (7b) resulting from the state reset, and 3) sequence-to-sequence (Seq2seq) training scheme, to mitigate the "forgetting" effect in long-horizon training. We provide the details of each extension below.

1) Extension of DeepReach to parameterized BRTs of hybrid limit cycle: The neural network value function will be denoted as $V_\theta(t, x)$, where θ indicates the weights of the sinusoidal network. As proposed in [40], we can augment the input of the network to treat the gait parameter as a virtual state. Consequently, our value function is expressed as $V_\theta(t, x; \beta)$. This enables us to derive RoAs for different gaits using a single learned value function model and through a single training procedure.

2) Loss function: In each training iteration, we sample a batch of time, state, and gait parameter samples from the target domain of each input entity. The loss function for a given sample $(t_i, x_i; \beta_i)$, consists of four loss terms as below:

$$h(t_i, x_i; \beta_i) = \lambda_1 h_1 + \lambda_2 h_2 + \lambda_3 h_3 + \lambda_4 h_4, \tag{10}$$

where h_1 and h_2 are given in (9), and

$$\begin{aligned} h_3 =& |V_\theta(t_i, x_i; \beta_i) - V_\theta(t_i, \Delta(x_i); \beta_i)| \text{ (for } x_i \in S\text{)}, \quad (11) \\ h_4 =& \max\{V(t_i, x_i, \beta_i) - l(x_i; \beta_i), 0\} \\ & + \max\{V(t_i, x_i, \beta_i) - V(t_j, x_i, \beta_i), 0\} \text{ where } t_j < t_i. \end{aligned}$$

Here, λ_i for $i = 1, 2, 3, 4$ are weights of each loss term. We introduce two new loss terms, h_3 and h_4. The loss term h_3 ensures that the states before and after the reset share the same value function values, to satisfy the condition (7b). This is the essential loss term that captures the effect of the impact event of the walking behavior to the gait stabilization. From the definition of the value function in (3), it can only monotonically decrease in time due to $\min_{\tau \in [-t, 0]}$ in (2). Thus, we impose this condition by adding the loss term h_4, which penalizes V_θ if the monotonically decreasing condition is not met. Including this term incentivizes the neural network to learn a more accurate value function.

3) Seq2seq Training: In DeepReach, during the training, the sample time domain $[0, T]$ is scheduled in a curriculum learning fashion, by gradually increasing the maximum time T. It is important that the value function is shaped from the initial condition constraint in the beginning, and it is carved out as the training proceeds through the PDE loss h_1 and other loss terms. Thus, the initial condition serves as an "anchor" for the value function. However, the anchoring effect of the initial condition gets less effective for t that is further away from 0. Thus, an apparent issue with training a single model for the entire time domain $[0, T]$ is that the value function starts to forget information of l when the time horizon is longer. This issue is more severe when the dynamics are stiff and involve state jumps, as in our problem.

To mitigate this issue, we employ a Seq2seq training scheme from [41]. Basically, Seq2seq splits the time domain into multiple subdomains, and trains separate neural network models for each subdomain. This divide-and-conquer approach is effective in mitigating the forgetting phenomenon; for each subdomain, we can introduce the initial condition loss again, which will anchor the value function not only at $t = 0$ but also at the intervals of the subdomains. Each subsequent sequence benefits from the model trained on the preceding sequence as its supervision signal.

B. One-step predictive stabilizing controller

Once we train the value function $V_\theta(t, x; \beta)$, we can easily derive the gradient-based optimal controller in (6), to stabilize the states within the RoA to the gait. However, in our evaluation, applying (6) was not successful mainly due to the learning errors of the value function and its gradients. The effect of the error is severe for the closed-loop performance of the gradient-based controller. This is because when the trajectory evolves, the accumulation of the error effect is tightly coupled with the stability of the closed-loop dynamics. It becomes a more severe issue for legged robots that involve discrete contact dynamics (1b).

In this work, we design our controller as an one-step predictive (OSP) control problem, which does not directly rely on the value function gradient. This formulation is in fact the discrete-time approximation of the optimal control law in (6), and if the value function is accurate, they should produce the same result with small enough timestep. However, with the neural network value function, in our experiments, we observed that the new formulation achieves a much higher success rate of stabilization than (6). At each time step i, the OSP controller solves an optimization problem

$$\begin{aligned} &\min_{u_i \in \mathcal{U}} V_\theta(t_i, \hat{x}_{i+1}; \beta) \\ &\text{where } t_i \text{ is such that } V_\theta(t_i, x_i; \beta) = 0, \\ &\qquad \hat{x}_{i+1} = \mathrm{f}(x_i, u_i). \end{aligned} \tag{12}$$

The time t_i is determined as a (minimal) *Time-to-Reach (TTR)* at the current state x_i [38], clipped by the time horizon T. Since the value function is non-increasing with respect to time, such TTR value is uniquely determined. We can find the value by doing a binary search for when $V_\theta(t_i, x_i; \beta)$ becomes zero. By taking TTR as the time index for the value function evaluation, the controller is trying to "slide along with" the zero-level set of the value function—the BRT—over time, until the BRT shrinks to the target set. The discrete-time dynamics $\mathrm{f}(x, u)$ in (12) is determined based on the continuous system dynamics described in (1). Finally, the objective determines a control input that leads the state at the next time step to the one where the value function is maximally decreased. By iteratively updating the TTR and descending the value function, the controller can ultimately converge to the target set.

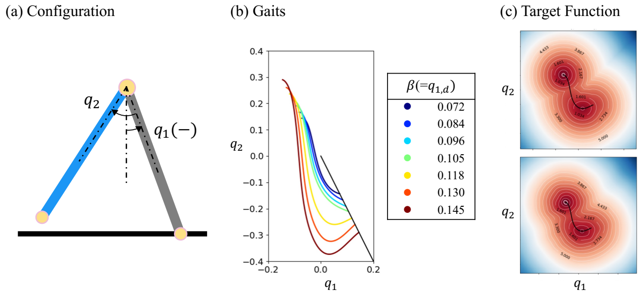

Fig. 1: (a) Configuration of the two-link walker (grey: stance leg, blue: swing leg). (b) Hybrid limit cycle gaits in q_1-q_2 space with various walking step lengths. Black line indicates the switching surface. (c) q_1-q_2 slice of the numerical (top) and learned (bottom) target function $l(x; \beta)$ for $\beta = 0.13$.

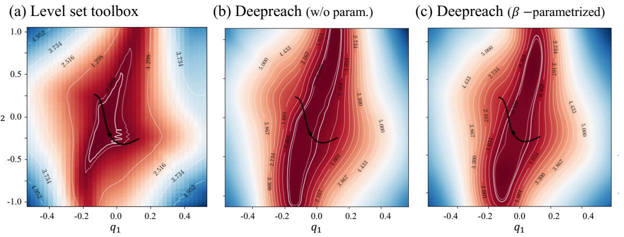

Fig. 2: Comparison between the value functions $V(T, x; \beta)$ for the gait with $\beta = 0.13$, obtained by (a) the numerical method in [37], [6], and (b) (c) our method, where in (b), V_θ is not parametrized and β is fixed as 0.13. The values are visualized in color contour map in $q_1 - q_2$ slices along the gait. The zero-level sets (thick white line) represent the estimated BRTs.

Given that the optimization problem involves a non-convex cost function and the constraint represented by a neural network, one can adopt a discretization or sampling-based search (evaluating over discrete samples in the input set $\mathcal{U}$) to perform the optimization. For our one-step problem, this can be done promptly in real-time as it only requires a single-instance batch neural network inference for all the samples. However, the complexity can increase if (12) is extended to a multi-step predictive control problem.

C. Gait switching controller

Based on the OSP controller in (12), we are able to effectively stabilize the robot state to a single gait $O(\beta)$, when the state x_i is inside $BRT(L(\beta); T)$, where T is the maximum time horizon the value function is trained for. However, if $x_i \notin BRT(L(\beta); T)$, although in practice we can still deploy (12) by setting the TTR t_i as T, since the state is outside of the RoA estimate, there is no guarantee that it will eventually stabilize to the gait. Recalling the second objective in Section II-B, we always want to commit to a gait to which stabilization is feasible. The main advantage of having access to the parametrized value function is being able to switch the target β actively online to achieve this.

We assume that a desired gait parameter β^* is specified by a user command or by a pre-specified sequence of desired gaits. Whenever before (12) is executed with $\beta = \beta^*$, we can check whether $x_i \in BRT(L(\beta^*); T)$, and if this condition is not satisfied, change β to a member of $\mathcal{B}_{\text{feas}}(x_i)$. In the case

Algorithm 1: Gait switching strategy

```
Input  : Current state $x_i$, Desired gait parameter $\beta^*$
Output: Selected gait parameter $\beta_i$
if $V_\theta(T, x_i; \beta^*) \leq 0$ then
    return $\beta^*$
end
$\mathrm{min\_value} \leftarrow +\infty,\ \ \beta_i \leftarrow$ None
for $\beta$ in $\mathcal{B}$ do
    if $\mathrm{min\_value} > V_\theta(T, x_i; \beta)$ then
        $\mathrm{min\_value} \leftarrow V_\theta(T, x_i; \beta),\ \ \beta_i \leftarrow \beta$
    end
end
if $\mathrm{min\_value} > 0$ then
    Warning: $\mathcal{B}_{\mathrm{feas}}(x_i)$ is empty
end
return $\beta_i$
```

where the state x_i is perturbed by unmodeled disturbance, the feasibility condition will be checked for all possible gaits and the gait will be switched to a new feasible gait. For perturbations whose magnitude is significant so that $\mathcal{B}_{\mathrm{feas}}(x_i)$ is empty, the user will be aware that the walking stability is not ensured anymore. This gait switching strategy is summarized in Algorithm 1.

V. Case Study: Two-link Walker

We consider a compass-gait walker, which consists of two links with an actuated joint between them. We consider a pinned model of the robot, with the configuration variable $q := [q_1, q_2]^T$ as illustrated in Fig. 1 (a), and we define the state as $x := [q, \dot{q}]^T$. The switching surface is defined as where the swing foot hits the ground with a negative velocity and the stance leg angle crosses a predefined threshold $\bar{q}_1$, $S := \{x \mid q_1 \leq \bar{q}_1, 2q_1 + q_2 = 0, 2\dot{q}_1 + \dot{q}_2 < 0\}$.

The stable gaits of the robot are represented as swing leg angle $q_{2,r}$ being a polynomial function of q_1. These polynomial gaits are obtained from trajectory optimization [8] for desired stance leg angle at the event of impact, $q_{1,d}$, ranging from 0.072 to 0.145. We set $\beta = q_{1,d}$ which decides the walking step length of the gait. The closed-loop dynamics under an input-output (IO) linearization controller [19] is considered in the optimization, thus, the obtained gaits are stable limit cycle under the IO linearization controller. The IO linearization controller also provides the baseline stabilzing controller π_0 discussed in Section II-A. The obtained parametrized gaits are visualized in Fig. 1 (b).

1) Training details: The target function $l(x; \beta)$ is constructed by evaluating the distance between the state x and the gait $O(\beta)$. Evaluating the distance numerically for all samples in each training iteration significantly slows down the training. Instead, we use a neural network to represent $l(x; \beta)$. We generate 100,000 samples with 21 values of β from $\mathcal{X}$ and $\mathcal{B}$, and learn the target function with supervised learning. The learned $l(x; \beta)$ is shown in Fig. 1 (c).

In the value function training, we employ a 3-layer neural network with 512 hidden nodes in each layer to represent the learned value function and we utilize the sinusoidal function as the activation function. Additionally, we set the time span of the BRTs to $T = 0.5$. We break up the time span to four sequences for Seq2seq training. In each sequence, we uniformly sample 130,000 samples of $(t_i, x_i; \beta_i)$. It takes approximately 8 hours to complete one sequence of training on the RTXA5000 GPU, and total time cost for the entire training of the parametrized BRTs is 32 hours. This is notably shorter than the direct numerical method in [6], as a computation for a *single* gait BRT takes 12 hours.

2) Learned Value function: The learned value function is visualized in Fig. 2 for $\beta = 0.13$, where we compare our solution to the numerical solution obtained in [6]. Note that the numerical solution is not necessarily the "ground-truth", as discussed in Sec. III-C, due to numerical errors. The value functions of our method are calibrated after training, based on the approach in [42], which provides an empirical 95% success rate of stabilization from the calibrated BRT. Overall, our learned value function generates larger estimation of the BRT compared to the numerical solution. Meanwhile, our parametrization does not sacrifice much accuracy in BRT estimation, as depicted in Fig. 2 (b) and (c).

3) Regions of Attraction: The trained BRTs, representing the RoAs of the gaits, given as the zero-level sets of the learned value function $V_\theta(T, x; \beta)$, are visualized in Fig. 3. For any states encapsulated in the BRT of β, theoretically, we can ensure their convergence to the target gait of parameter β. From the overlapping region of all BRTs, any gait parameter can be selected as the target gait. In contrast, at a state that does not belong to any RoAs, it might not be feasible to stabilize the robot, no matter how good the control policy is.

4) Stabilization performance and comparison to other controllers: The mechanism of the OSP controller in (12) is visualized in Fig. 4, along a trajectory that is stabilized from a perturbed initial state to the gait. It shows that over time, the BRT evaluated at TTR shrinks to the target set and guides the trajectory to successfully converge to the gait. Next, we do a quantitative analysis of the performance of our stabilization controller. We evaluate the success rate of the stabilization within two walking steps among 6,600 trajectories initialized within the state space grid. The states whose swing foot lies below the ground are filtered out, as they represent physically unrealistic configurations. The learned gait BRT encapsulates 13.74% of the tested initial states. We compare the success rates of our controller, the IO linearization controller, and a simple receding-horizon NMPC controller that minimizes the tracking error to the gait. The NMPC prediction horizon is set to 0.35, and increasing the horizon decreased the success rate, due to more occurrence of infeasible solutions. The success rates and the hit map of the successful initial states are reported in Fig. 5. The success rate of our controller (25.7%) surpasses the success rates of the others by 10%.

5) Gait switching for enhanced stabilization: When the user commands to change the gait or if a perturbation occurs to our robot, it becomes necessary for the robot to transition and stabilize to a new gait based on the gait switching strategy in Algorithm 1. We first demonstrate scenarios where the user commands a sequence of desired gaits. We

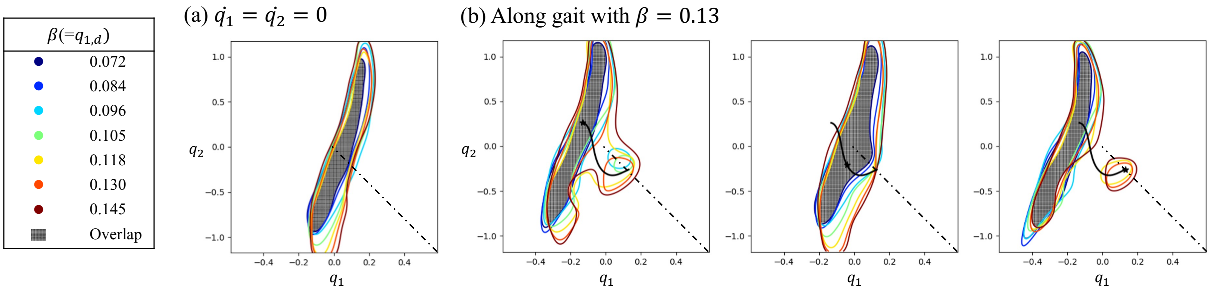

Fig. 3: q_1-q_2 slices of gait BRTs (a) when angular velocities are 0, (b) values that lie on the gait with $\beta = 0.13$ (* indicates where the slice is taken). Each BRT captures states from which the robot can be stabilized to the corresponding gait, and in the overlap region, pursuing any gait is feasible.

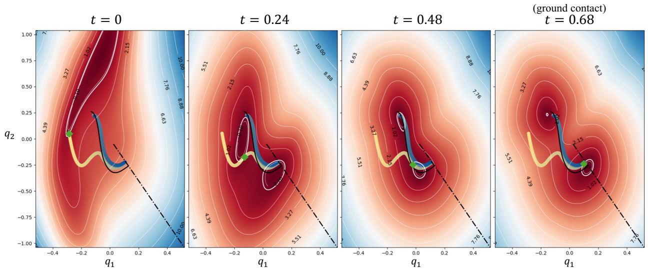

Fig. 4: Snapshots of the phase portrait of the trajectory, initialized at a perturbed state, stabilizing to the gait with $\beta = 0.13$, under the OSP controller (12). The trajectory evolves from yellow to blue while taking two walking steps, and we show the first walking step portion. Green dots represent the state where the snapshot is taken. The color contour map visualizes $V_\theta(t_i, \cdot; \beta)$ in (12). [Video (https://youtu.be/P7Vnr8jwSPc)]

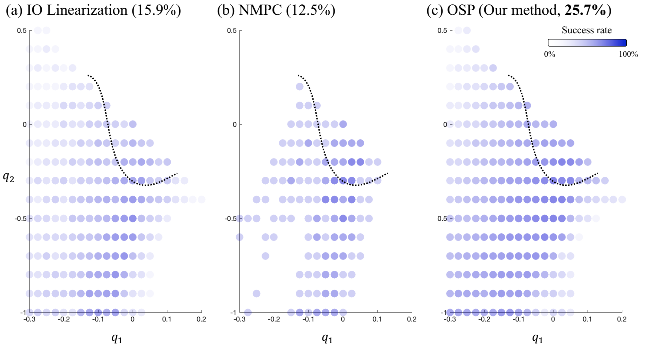

Fig. 5: Success rate of stabilization evaluated over a grid of 6,600 initial states. At each $(q_1, q_2) \in [-0.3, 0.3] \times [-1.0, 1.0]$ value, we evaluate 25 combinations of $(\dot{q}_1, \dot{q}_2) \in \times[-1.0, 1.0] \times [-2.5, 2.5]$, and visualize the rate of the trajectories successfully converging to the gait ($\beta = 0.13$).

evaluate our controller under the three different commands of desired gait sequences: (a) gradually increasing β, (b) gradually decreasing β, and (c) dramatically switching β between its minimum and maximum values for every two steps. The results are displayed in Fig. 6. The proposed algorithm enables the robot to switch between different gaits while trading off stability against the commanded gait. This can be particularly seen in the third case where the robot doesn't immediately switch to the minimum gait as that might result in loss of stability—instead, the robot switches to intermediate gaits determined by the algorithm.

We also introduce a strong perturbation to the robot which is stably tracking an initial gait (β_1), mandating it to transition to a new gait (β_2) since the perturbed state is not included in the $BRT(\beta_1)$. We utilize our gait switching controller to identify a new feasible gait and stabilize the robot to it. The results are shown in Fig.7.

VI. Conclusion

In this work, we utilized deep learning-based reachability analysis to create a library of Regions of Attraction (RoAs) for various gaits of legged robots with hybrid dynamics. The analysis with the estimated RoAs provides a transparent logic behind our gait-stabilizing controller and the gait switching strategy.

In future research, several intriguing directions are worth exploring. First, although our use of neural networks to approximate solutions to the HJ PDE has shown promise, our learned value function can be still inaccurate due to accumulated errors in the learning process. Therefore, the estimated RoAs cannot precisely guarantee safety for the robots. Investigating recent advancements that provide probabilistic guarantees on learned solutions [42] could mitigate this limitation. Additionally, we aim to explore scenarios with persistent disturbances, such as payloads, and design control policies that are robust against bounded disturbances. This could be achieved by employing a differential game-based robust reachability formulation [36]. Finally, applying our approach to higher-dimensional, real-world walking robots will be an exciting direction.

References

[1] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, “Reinforcement learning for robust parameterized locomotion control of bipedal robots,” in *IEEE ICRA*, 2021.

[2] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, “Blind bipedal stair traversal via sim-to-real reinforcement learning,” in *Robotics: Science and Systems (RSS)*, 2021.

[3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Science robotics*, 2020.

[4] I. R. Manchester, M. M. Tobenkin, M. Levashov, and R. Tedrake, “Regions of attraction for hybrid limit cycles of walking robots,” *IFAC Proceedings Volumes*, 2011.

[5] M. Posa, M. Tobenkin, and R. Tedrake, “Stability analysis and control of rigid-body systems with impacts and friction,” *IEEE TAC*, 2015.

[6] J. J. Choi, A. Agrawal, K. Sreenath, C. J. Tomlin, and S. Bansal, “Computation of regions of attraction for hybrid limit cycles using reachability: An application to walking robots,” *IEEE RA-L*, 2022.

[7] S. Bansal and C. Tomlin, “Deepreach: A deep learning approach to high-dimensional reachability,” in *IEEE ICRA*, 2021.

[8] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[9] G. H. Negri, L. K. Rosa, M. S. Cavalca, L. A. Celiberto Jr, and E. B. de Figueiredo, “Nonlinear predictive control applied to a biped walker with adjustable step length using a passive walking-based reference generator,” *Optimal Control Applications and Methods*, 2020.

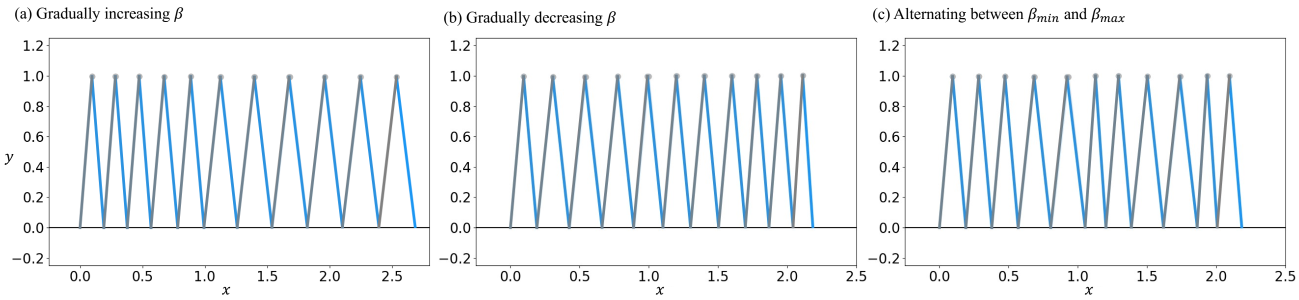

Fig. 6: Two-link walker trajectory snapshots under the gait switching strategy in Algorithm 1 and the OSP controller, when the sequence of gait is commanded. (a) gradually increasing β, (b) gradually decreasing β, and (c) switching β between its min and max values for every two steps. [Video]

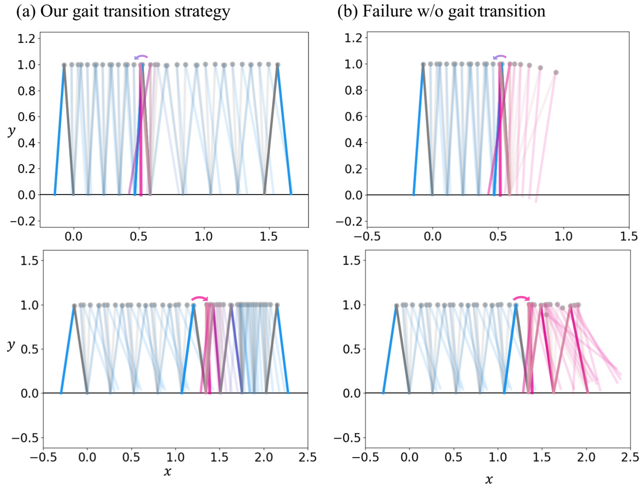

Fig. 7: Strong perturbations (red) causing the robot state to exit the BRT of the initial gait β_1 (top: $\beta_1 = 0.096$, bottom: $\beta_1 = 0.1485$) (a) Algorithm 1 effectively switch the gait and prevents the robot from falling, whereas (b) maintaining the initial gait leads to failure. [Video]

[10] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, 2019.

[11] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *arXiv preprint arXiv:2401.16889*, 2024.

[12] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Malik, "Humanoid locomotion as next token prediction," *arXiv preprint arXiv:2402.19469*, 2024.

[13] M. S. Motahar, S. Veer, and I. Poulakakis, "Composing limit cycles for motion planning of 3d bipedal walkers," in *IEEE CDC*, 2016.

[14] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research (IJRR)*, 2012.

[15] S. A. Burden and S. D. Coogan, "On infinitesimal contraction analysis for hybrid systems," in *IEEE CDC*, 2022.

[16] Z. Gu, N. Boyd, and Y. Zhao, "Reactive locomotion decision-making and robust motion planning for real-time perturbation recovery," in *IEEE ICRA*, 2022.

[17] G. Piovan and K. Byl, "Reachability-based control for the active slip model," *The International Journal of Robotics Research*, 2015.

[18] Y. Ding, M. Zhang, C. Li, H.-W. Park, and K. Hauser, "Hybrid sampling/optimization-based planning for agile jumping robots on challenging terrains," in *IEEE ICRA*, 2021.

[19] K. Sreenath, H.-W. Park, and I. Poulakakis, "A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel," *The International Journal of Robotics Research*, 2011.

[20] O. Dosunmu-Ogunbi, A. Shrivastava, and J. W. Grizzle, "Demonstrating a robust walking algorithm for underactuated bipedal robots in non-flat, non-stationary environments," *arXiv preprint arXiv:2403.02486*, 2024.

[21] J. Choe, J.-H. Kim, S. Hong, J. Lee, and H.-W. Park, "Seamless reaction strategy for bipedal locomotion exploiting real-time nonlinear model predictive control," *IEEE RA-L*, 2023.

[22] H. Li, W. Yu, T. Zhang, and P. M. Wensing, "A unified perspective on multiple shooting in differential dynamic programming," in *IEEE/RSJ IROS*, 2023.

[23] J. Choi, F. Castañeda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," in *RSS*, 2020.

[24] T. Westenbroek, F. Castaneda, A. Agrawal, S. Sastry, and K. Sreenath, "Lyapunov design for robust and efficient robotic reinforcement learning," in *CoRL*, 2022.

[25] Y. Meng and C. Fan, "Hybrid systems neural control with region-of-attraction planner," in *L4DC*, 2023.

[26] 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 RA-L*, 2021.

[27] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," in *IEEE ICRA*, 2020.

[28] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *CoRL*, 2022.

[29] T. X. Nghiem, J. Drgoňa, C. Jones, Z. Nagy, R. Schwan, B. Dey, A. Chakrabarty, S. Di Cairano, J. A. Paulson, A. Carron *et al.*, "Physics-informed machine learning for modeling and control of dynamical systems," in *ACC*. IEEE, 2023.

[30] R. Bellman, *Dynamic Programming*. Princeton university press, 1957.

[31] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, 1998.

[32] J. Sirignano and K. Spiliopoulos, "Dgm: A deep learning algorithm for solving partial differential equations," *J. Comput. Phys.*, 2018.

[33] J. Darbon, G. P. Langlois, and T. Meng, "Overcoming the curse of dimensionality for some hamilton–jacobi partial differential equations via neural network architectures," *Research in the Math. Sciences*, 2020.

[34] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *IEEE CDC*, 2017.

[35] F. Camilli, L. Grüne, and F. Wirth, "Control Lyapunov functions and Zubov's method," *SIAM Journal on Control and Optimization*, 2008.

[36] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *HSCC*, 2015.

[37] I. Mitchell, "A toolbox of level set methods," *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS.pdf*, 2004.

[38] ——, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D. dissertation, Stanford University, 2002.

[39] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *arXiv preprint arXiv:2006.09661*, 2020.

[40] J. Borquez, K. Nakamura, and S. Bansal, "Parameter-conditioned reachable sets for updating safety assurances online," in *IEEE ICRA*, 2023.

[41] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," *NeurIPS*, 2021.

[42] A. Lin and S. Bansal, "Generating formal safety assurances for high-dimensional reachability," in *IEEE ICRA*, 2023.