Barrier Functions Inspired Reward Shaping for Reinforcement Learning

Abhishek Ranjan[1*], Nilaksh[2*], Shreenabh Agrawal[1*], Aayush Jain[2], Pushpak Jagtap[3], Shishir Kolathaya[4]

***Abstract*— Reinforcement Learning (RL) has advanced from addressing basic control problems to tackling complex real-world tasks with large state spaces. Although RL excels in solving these challenging problems, the time required for convergence during training remains a significant limitation. Various techniques have been proposed to mitigate this issue, and reward shaping has emerged as a popular solution. However, most existing reward-shaping methods rely on value functions, which can pose scalability challenges as the environment's complexity grows. Our research proposes a novel framework for reward shaping inspired by Barrier Functions, which is safety-oriented, intuitive, and easy to implement for any environment or task. To evaluate the effectiveness of our proposed reward formulations, we conduct experiments on various environments, including CartPole, Half-Cheetah, Ant, and Humanoid. Our results demonstrate that our method leads to 1.4-2.8 times faster convergence and as low as 50-60% actuation effort compared to the vanilla reward. Moreover, our formulation has a theoretical basis for safety, which is crucial for real-world applications.**

I. INTRODUCTION

Reinforcement Learning (RL) has succeeded significantly in learning policies for performing complex tasks such as [1], [2], Atari games [3], and robotics problems [4], [5], especially when combined with deep learning. This success is often attributed to the neural network's ability to approximate non-linear functions; however, the design of appropriate reward functions is equally important. Typically, reward functions need to be well-designed and capture an implicit specification of the desired behaviour of the agent. A well-designed reward function can lead to the algorithm adopting policies that converge to the intended learning task.

To make RL more accessible, previous works have established reward-shaping to accelerate algorithm convergence [6], [7], [8]. The most well-known work in the reward shaping domain uses the potential-based reward shaping (PBRS) [9] that suggests adding a potential function term initialized to the value function for reward-shaping. However, PBRS needs a good estimate of the value function, which presents a challenge in the case of sparse reward models and in complex environments due to the 'curse of dimensionality.' [10], [11], [12].

This work is supported in part by the Google Research Grant, Pratiksha Young Investigator Fellowship, and the SERB Grants SRG/2022/001807, CRG/2021/008115.

[1]Indian Institute of Science (IISc), Bangalore {abhishekr, shreenabhm}@iisc.ac.in

[2]Indian Institute of Technology (IIT), Kharagpur {nilaksh404, aayushjain}@kgpian.iitkgp.ac.in

[3]RBCCPS, IISc - pushpak@iisc.ac.in

[4]CSA & RBCCPS, IISc - shishirk@iisc.ac.in

*Equal Contribution

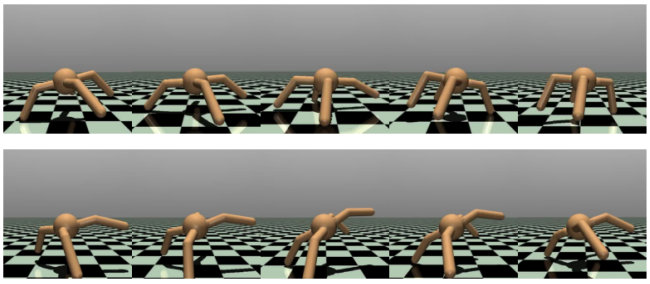

Fig. 1: An example run of Ant with Exponential Barrier reward (Top) and the Vanilla reward (Bottom), trained for the same number of time steps. Unlike the vanilla reward, barrier function-based reward leads to more natural movements and better utilization of all joints.

We present Barrier Function (BF) inspired reward shaping, a simple and safety-oriented framework that results in greater training efficiency and safety. Our method employs BF-based reward-shaping terms to augment an environment's vanilla reward. Barrier Functions are known to be positive when a system's state is within a set of safe states. Thus with this theoretical basis, our reward-shaping terms have the property of enforcing an RL agent's states to be within the safe subset of all the states. We propose two barrier function formulations: *Exponential barrier* and *Quadratic barrier*, and evaluate our framework on several environments ranging from CartPole [13] to Humanoid [14]. Our framework also leads to better gait performance for walkers, which can be qualitatively seen in Fig. 1

The main contributions of this paper are summarized as follows:

- A safety-oriented and easy-to-implement Barrier Function-inspired reward shaping framework that can be intuitively constructed based on the task.
- The framework leads to faster convergence towards the goal and efficient state exploration by enforcing the system within the safe set.
- It also leads to lesser energy expenditure as the barrier function constrains the states within desired limits, thus avoiding extreme actions.

We note that we do not include an energy term in the shaped reward as it often conflicts with the environment's primary objective. For example, CartPole's lowest energy state would correspond to the pendulum being vertically down, which is not the intended goal. Even though we can scale the energy terms in the reward, this leads to an additional tuning of the rewards, which is counterproductive

for bigger robot models.

II. Related Work

Reward Shaping: The most well-known work in the reward shaping domain is the potential-based reward shaping (PBRS) method [15]. Positive linear transformation, familiar from utility theory, shows that we can add rewards for transitions between states, expressible as a difference between the values of the arbitrary potential functions applied to the respective states. The PBRS method's application for knowledge extraction during the learning process for reward shaping, rather than directly formulating rewards for the learning process, has been studied by Badnava et al. [16]. Hu et al. [17] have shown how to utilize the reward-shaping function's beneficial part and ignore the unbeneficial part.

Performance and Convergence: Reward shaping can significantly increase the agent's performance and optimality [18], [19], [20]. Natural language can be used as a reward-shaping function for better agent performance [21]. The Lyapunov function from control theory can be used as a reward-shaping function [22]. The concept of automatically learning to restructure a Markov Decision Process (MDP) reward function to accelerate reinforcement learning has been discussed [23]. Still, the stability and safety of the agent and the agent's environment are yet to be considered. As per our findings, the MDP approach [23] takes a lot of energy to get trained. Marom et al. [24] show a Bayesian approach to reward shaping that augments the reward distribution with prior beliefs and decays with experience. This Bayesian approach incorporates the model-based reward, limiting the reward function's performance when tested in different environments. Balakrishnan et al. [25] showed that signal temporal logic can be a formal specification for the reward formulation to achieve the end-desired goal. But this method is also a model-based reward formulation and does not exhibit the notion of generality.

In addition, Reward Shaping has been used for transfer learning [26], but its usage for the stability and safety of the agent has yet to be explored. To our knowledge, we are among the first to explore the agent's safety and stability. Our reward-shaping framework also ensures less energy consumption during training and testing and can be used for any environment with slight modifications.

III. Preliminary

A. Reinforcement Learning

Reinforcement learning (RL) can be described as a discounted Markov decision process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P} : \mathcal{S} \times A \rightarrow \mathcal{S}$ is the deterministic state transition function, $r : \mathcal{S} \times A \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in$ (0,1) is the discount factor. In RL, the goal of the learning algorithm is to converge on a policy π: $\mathcal{S} \rightarrow \mathcal{A}$ that maximizes the total (discounted) reward after performing actions on an MDP, i.e., the objective is to maximize $\Sigma_0^\infty \gamma^t r_t$, where r_t is the output of the reward function r for the sample at instance t. Since a policy maps a state to an action, the value of a policy is evaluated according to the discounted cumulative reward. Given this MDP, we are interested in shaping the rewards by using Barrier Functions. To this end, we have to first describe the concept of Barrier Functions (BFs) formally, which is explained next.

B. Barrier Functions

For practical applications, we often want the system state s to stay within a safe region, denoted as a set $\mathcal{C}$. The set $\mathcal{C}$ is defined as the *super-level set* of a continuously differentiable function $h : \mathcal{S} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ yielding,

$$\mathcal{C} = \{s \in \mathcal{S} : h(s) \geq 0\} \tag{1}$$

$$\partial\mathcal{C} = \{s \in \mathcal{S} : h(s) = 0\} \tag{2}$$

$$\text{Int}\,(\mathcal{C}) = \{s \in \mathcal{S} : h(s) > 0\}, \tag{3}$$

where $\text{Int}\,(\mathcal{C})$ and $\partial\mathcal{C}$ denote the interior and the boundary of the set $\mathcal{C}$, respectively. It is assumed that $\text{Int}\,(\mathcal{C})$ is non-empty and $\mathcal{C}$ has no isolated points, i.e. $\text{Int}\,(\mathcal{C}) \neq \phi$ and $\overline{\text{Int}\,(\mathcal{C})} = \mathcal{C}$. The set $\mathcal{C}$ is said to be forward invariant (safe) if $\forall s(0) \in \mathcal{C} \implies s(t) \in \mathcal{C} \quad \forall t \geq 0$. We can mathematically verify the safety of the set $\mathcal{C}$ by establishing the existence of a barrier function. We have the following definition of a barrier function (BF) from [27].

Definition 3.1: Given the set $\mathcal{C}$ defined by (1)-(3)*, the function h is called the barrier function (BF) defined on the set $\mathcal{S}$ if there exists an extended class $\mathcal{K}$ function κ such that for all $s \in \mathcal{S}$:*

$$\dot{h}\,(s, \dot{s}) + \kappa\,(h(s)) \geq 0, \tag{4}$$

Here $\kappa : (-\infty, \infty) \rightarrow (-\infty, \infty)$ is a strictly increasing continuous function, with $\kappa(0) = 0$. κ is widely called an extended class $\mathcal{K}$ function. Note that $\dot{h}(s, \dot{s}) := \frac{\partial h}{\partial s}\dot{s}$, where $\dot{s}$ is the time derivative of s. Even if the MDP is in discrete time, $\dot{s}$, and consequently $\dot{h}$, can be calculated approximately as $\dot{h} \approx \frac{\partial h}{\partial s}(s_t - s_{t-1})/\Delta t$, where s_t, s_{t-1} are the samples of the states obtained at time steps t and $t-1$, and Δt is the time interval between two samples. It is worth mentioning that the classical definition has the actions or controls u as one of the arguments in (4), thereby making it a *control* barrier function (CBF). For this paper, we avoid the use of inputs and make estimates of the derivative of h by using $\dot{s}$. This is sufficient for the presented work as our focus is on reward shaping.

If we are able to restrict the states of the system $s, \dot{s}$ in such a way that the inequality (4) is satisfied, then we know that the set $\mathcal{C}$ is forward invariant (safe). We can use this idea to shape the reward functions in such a way that any violation of safety causes a loss of reward. We will formally show the reward shaping methodology in the next section.

IV. Reward Shaping Methodology

A. Reward Shaping using Barrier Functions

Having described BFs and their associated formal results, we now discuss BF-inspired reward shaping in the context of RL. Reward shaping is a method for engineering a reward function to provide more frequent feedback on appropriate

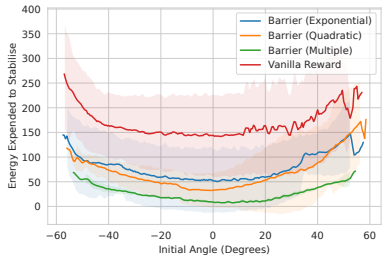

Fig. 2: Energy expended to stabilize each initial angle for all rewards.

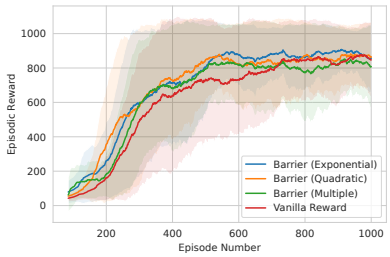

Fig. 3: Unnormalised reward graph for the BF-based rewards.

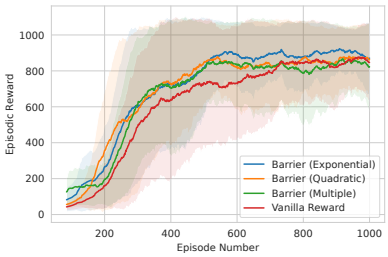

Fig. 4: Normalised reward graph for the BF-based reward formulations based on vanilla reward

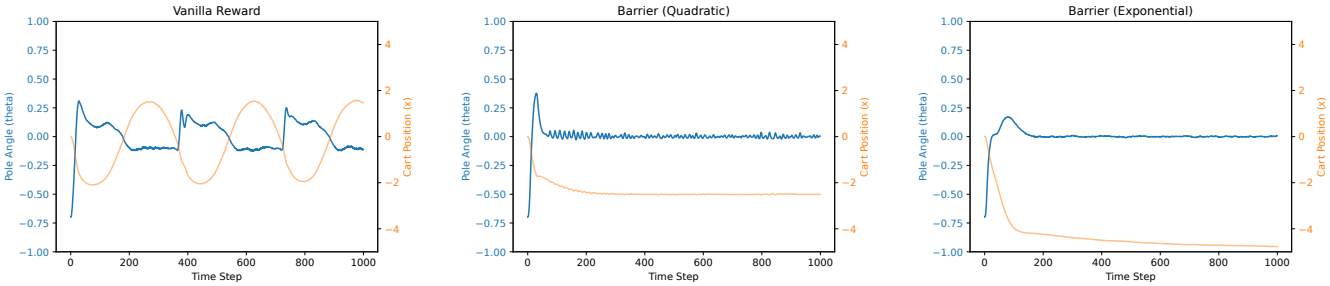

Fig. 5: θ_p and x-position vs time-step graph showing control flow to stabilize -40° pole angle.

behaviours. To illustrate our framework, we propose the shaped reward r' in (5), which we obtain by adding a term to the vanilla reward r.

$$r'(s, \dot{s}) = r(s) + \underbrace{r^{\text{BF}}(s, \dot{s})}_{\text{additional reward}}, \tag{5}$$

$r^{\text{BF}}(s, \dot{s})$ is our barrier function inspired reward shaping term. We emphasise that the new reward r' depends on $\dot{s}$. From definition 3.1, $\mathcal{C}$ is forward invariant if and only if there exists a barrier function h such that it satisfies (4). Thus we define r^{BF} as

$$r^{\text{BF}}(s, \dot{s}) = \dot{h}(s, \dot{s}) + \gamma h(s), \tag{6}$$

In order to satisfy (4) we have taken the extended class $\mathcal{K}$ function κ as $\kappa(m) = \gamma m$, $\gamma > 0$. This gives our shaping term r^{BF} the desirable property of positively rewarding the agent when it is in a set of safe states $\mathcal{C}$ while negatively rewarding otherwise.

The BF h: $\mathcal{S} \to \mathbb{R}$ is chosen to be a suitable function that constrains specific quantities in $\mathcal{S}$, such that it would lead to desirable properties like actuation safety and training efficiency. The following section provides specific examples of BF-based reward shaping.

Remark 4.1: Since, in RL task, our goal is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes total (discounted) reward, as we maximise the proposed reward (5) it will also enforce the condition (4) in Definition 3.1 and thus implies safe execution of the task.

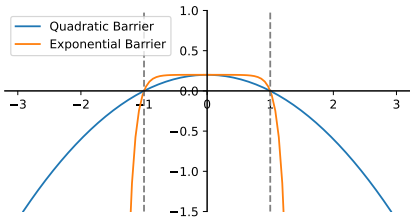

Fig. 6: Plots illustrating the proposed barrier functions. Dashed lines represent the constraint limits (-1,1).

B. Barrier Function Formulation

In an RL task, some state variables should ideally lie between a safe range of values. Violating this range can result in undesirable behaviours. To constrain them within their safe limits, we propose two barrier functions: a quadratic function h_{quad} (7) and an exponential function h_{exp} (8).

$$h_{\text{quad}}(s) = \sum_{l \in \mathcal{L}} \delta_l \left(s_l - s_l^{\max}\right)\left(s_l^{\min} - s_l\right) \tag{7}$$

$$h_{\text{exp}}(s) = \sum_{l \in \mathcal{L}} \lambda_l \left[1 - \left(e^{\delta_l (s_l - s_l^{\max})} + e^{\delta_l (s_l^{\min} - s_l)}\right)\right], \tag{8}$$

where $\lambda_l, \delta_l \in \mathbb{R}$ are some scaling factors, $\mathcal{L} = \{1, 2, 3, \dots\}$ is the set of indices for the elements of the state variables of the model whose values we want to constrain. In particular, $\{s_l\}$ is l^{th} element of state vector s, which is the value of the state variable upon which we wish to enforce the thresholds $(s_l^{\min}, s_l^{\max})$.

For appropriate values of δ_l, h_{exp} has a flat curve with

steep drops near $s_l^{\max}$ and $s_l^{\min}$, thus unlike h_{quad}, it has an additional property of not favouring median values of s_l (illustrated in Fig. 6). This makes the agent explore a larger range of s_l in a safe region. Thus h_{exp} is more suitable when the task involves s_l taking a wide range of values, while h_{quad} is more suitable for constraining to narrower ranges. Since both h_{quad} and h_{exp} are positive for a particular variable $l \in \mathcal{L}$ when $s_l \in (s_l^{\min}, s_l^{\max})$, from (6) the BF-based reward shaping terms can be written as (9).

$$\begin{aligned} r^{\text{BF}}_{\text{quad}}(s, \dot{s}) &= \dot{h}_{\text{quad}}(s, \dot{s}) + \gamma h_{\text{quad}}(s) \\ r^{\text{BF}}_{\text{exp}}(s, \dot{s}) &= \dot{h}_{\text{exp}}(s, \dot{s}) + \gamma h_{\text{exp}}(s), \end{aligned} \tag{9}$$

Thus from (4), the r^{BF} terms give positive reinforcement when the state is safe, i.e. $s \in \mathcal{C}$, and negative otherwise. The final shaped rewards can now be computed using (5).

$$\begin{aligned} r'_{\text{quad}} &= r + r^{\text{BF}}_{\text{quad}} = r + \dot{h}_{\text{quad}} + \gamma h_{\text{quad}} \\ r'_{\text{exp}} &= r + r^{\text{BF}}_{\text{exp}} = r + \dot{h}_{\text{exp}} + \gamma h_{\text{exp}}, \end{aligned} \tag{10}$$

C. Environment Specific Formulation

1) Walker Environments: In walker environments like Half-Cheetah [28], Ant [13], and Humanoid [14], the task is to learn to run as fast as possible. We use the same reward equation for all these environments as essentially they all have a similar state space $\mathcal{S}$ containing x^w_{pos}, Θ^w and Ω^w, where Θ^w and Ω^w are the set of walker agent's joint angles and angular velocities, respectively. The vanilla reward r given by (11) has multiple terms that guide the agent to move forward (r_{forward}) without falling (r_{health}) and minimizing the contact forces (r_{contact}).

$$r = r_{\text{health}} + r_{\text{forward}} - r_{\text{contact}}, \tag{11}$$

Since the task involves running, we seek to constrain the agent's joint angles $\theta_l \in \Theta^w$ within a safe range of angles $(\theta_l^{\min}, \theta_l^{\max})\ \forall l \in \mathcal{L}$ that is either specified by the environment or deduced from the domain knowledge. In this case, $\mathcal{L}$ is the set of all joint angles we wish to constrain, thus θ_l is analogous to s_l. Thus using (7)-(8), the barrier functions are given as (12), substituting them in (5) gives the final shaped reward.

$$\begin{aligned} h^{\text{walker}}_{\text{quad}}(s) &= \sum_{l \in \mathcal{L}} \delta_l \left(\theta_l - \theta_l^{\max}\right)\left(\theta_l^{\min} - \theta_l\right) \\ h^{\text{walker}}_{\text{exp}}(s) &= \sum_{l \in \mathcal{L}} \lambda_l \left[1 - \left(e^{\delta_l(\theta_l - \theta_l^{\max})} + e^{\delta_l(\theta_l^{\min} - \theta_l)}\right)\right], \end{aligned} \tag{12}$$

2) Cartpole: The state space $\mathcal{S}$ for CartPole contains the pole angle θ_p, pole angular velocity ω_p along with the cart's position and velocity x_c and v_c respectively. The task in the Cartpole environment is to learn to balance a pole attached to a moving cart, i.e. we want $\theta_p = 0$. The vanilla reward is r = +1 for every step taken. We define $(\theta_p^{\min}, \theta_p^{\max}) = (s_l^{\min}, s_l^{\max})$ as the pole angle's desired threshold range for the goal (θ_p = 0), around which we want the pendulum to stabilize. The corresponding BFs become :

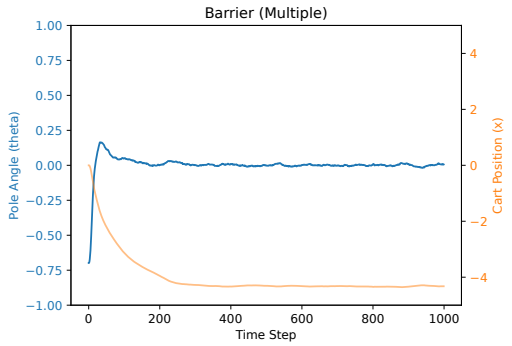

Fig. 7: θ_p and x-position vs time-step graph showing stabilization from an initial angle of -40°.

$$\begin{aligned} h^{\text{cart}}_{\text{quad}}(s) &= \delta_p(\theta_p^{\min} - \theta_p)(\theta_p - \theta_p^{\max}) \\ h^{\text{cart}}_{\text{exp}}(s) &= \lambda_p \left[1 - \left(e^{\delta_p(\theta_p - \theta_p^{\max})} + e^{\delta_p(\theta_p^{\min} - \theta_p)}\right)\right], \end{aligned} \tag{13}$$

Replacing $\delta_p = 1$ and $\theta_p^{\min} = -\phi$ and $\theta_p^{\max} = \phi$, with ϕ being the angle limit, we get

$$\begin{aligned} h^{\text{cart}}_{\text{quad}}(s) &= \phi^2 - \theta_p^2 \\ \dot{h}^{\text{cart}}_{\text{quad}}(s) &= -2\theta_p\dot{\theta}_p = -2\theta_p\omega_p, \end{aligned} \tag{14}$$

Thus according to our formulation in (5), the quadratic BF shaped reward is given as (15)

$$\begin{aligned} r'_{\text{quad}} &= 1 + \dot{h}^{\text{cart}}_{\text{quad}}(s, \dot{s}) + \gamma h^{\text{cart}}_{\text{quad}}(s) \\ &= 1 - 2\theta_p\omega_p + \gamma(\phi^2 - \theta^2), \end{aligned} \tag{15}$$

One can similarly get the exponential BF shaped reward from (13).

For the Cartpole environment, we further experiment by adding BFs constructed on ω_p and v_c in the same spirit as θ_p.

$$\begin{aligned} h^{\text{cart}}_{\text{multi}}(s) = \delta_1(\theta_p^{\min} - \theta_p)(\theta_p - \theta_p^{\max}) + \\ \delta_2(\omega_p^{\min} - \omega_p)(\omega_p - \omega_p^{\max}) + \\ \delta_3(v_c^{\min} - v_c)(v_c - v_c^{\max}), \end{aligned} \tag{16}$$

where $\delta_1, \delta_2, \delta_3 \in \mathbb{R}$ are some scaling factors, and the superscripts on ω, v denoting the min, max limits. We observe that adding more constraints improves the performance. This is shown in Fig. 2 and Fig. 7.

V. Experiments

We experimentally evaluate our BF-based reward-shaping formulation on OpenAI Gym's Cartpole [29] environment and MuJoCo environments like Half-Cheetah [28], Humanoid [14], and Ant [13]. We used the Twin Delayed DDPG (TD3) [30] algorithm with two variants of the BF-based reward: *quadratic* and *exponential*, and the environment's *vanilla* reward as the baseline. For the Cartpole environment, we used an additional step reward formulation. All experiments were performed on a Ryzen Threadripper 5995W CPU, 64GB RAM, and an NVIDIA GeForce RTX 3080.

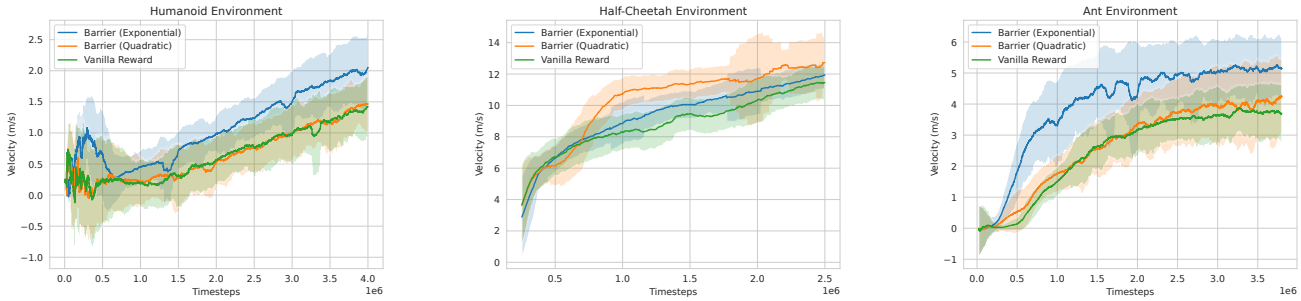

Fig. 8: The plots for each MuJoCo walker environment, averaged for ten random seeds, show the episodic velocity for each training time step. Since these environments aim to achieve as high a velocity as possible, these plots provide a good metric to judge the training speed.

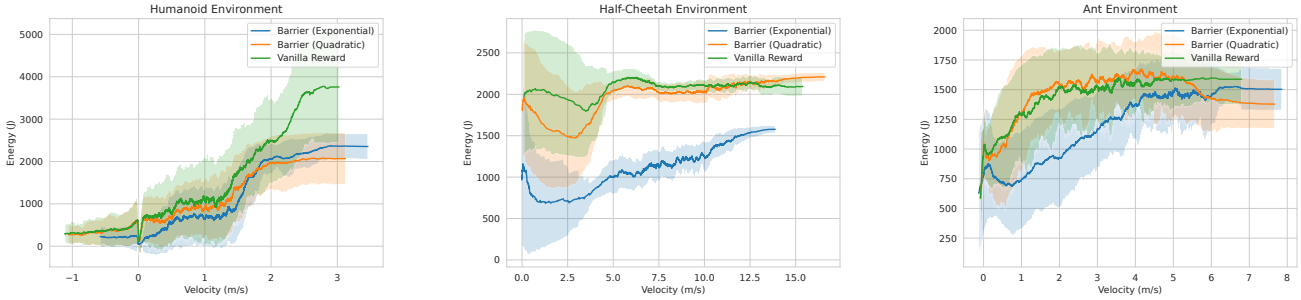

Fig. 9: The episodic energy (in joules) spent by the agent for achieving a particular velocity, averaged over ten random seeds. We see that exponential BF-based reward formulation performs the best across all the environments, consuming the least energy.

Reward	Humanoid		Half-Cheetah		Ant	
	Rel. Actuation Effort	Rel. Time	Rel. Actuation Effort	Rel. Time	Rel. Actuation Effort	Rel. Time
Vanilla	1.00	1.00	1.00	1.00	1.00	1.00
Quadratic BF	0.66 ± 0.08	0.89 ± 0.12	0.97 ± 0.08	**0.68 ±0.03**	**0.62 ±0.05**	0.76 ± 0.15
Exponential BF	**0.49 ±0.03**	**0.64 ±0.06**	**0.92 ±0.05**	0.96 ± 0.02	0.63 ± 0.06	**0.36 ±0.05**

TABLE I: Experiment results for different environments and reward formulations. The *Relative Actuation Effort* field is the episodic energy expended in actuation (in joules) divided by the squared velocity (m^2s^{-2}) relative to vanilla reward. The result is averaged over 100 best runs. *Relative Time* is the relative training time steps taken by a reward formulation to achieve the same maximum velocity as the Vanilla reward (averaged over 10 random seeds).

A. *Cartpole*

For Cartpole, four different reward formulations were explored, and a comparison of their training and test data is presented in Figures 2-5, averaged over three random seeds. The normalised graph in Fig. 4 represents the number of time steps for which the pole was balanced in each episode and is presented for a fair comparison.

Drawing an inference from Fig. 4, it can be easily seen that Barrier based rewards significantly enhance the convergence time. For example, BF-based rewards only take about 500 episodes to reach a reward of 800, while Vanilla rewards consistently took more than 700 steps, resulting in 40% slower convergence.

Fig. 2 shows energy expended till stabilisation vs initial angle for a range of angles. As can be read, the performance of vanilla reward is very inconsistent (large variation across seeds, indicating strong dependence on the seed) and energy expensive. In comparison, the barrier function-based rewards expend significantly less energy to stabilise the same initial conditions with seed independence. For 1 unit of energy spent by the Vanilla reward, Barrier (Exponential) spends 0.79 units, Barrier (Quadratic) spends 0.59 units, and Barrier (Multiple) spends 0.14 units.

Hence, we see more reliable and energy-efficient performance using BF-based rewards. Fig. 5 and 7 describe the control flow for each reward formulation for the same initial state. Using the Vanilla reward function resulted in highly chaotic behaviour with no convergence to the θ_p=0 value. In contrast, implementing barrier functions resulted in minimal to no oscillations. This was especially evident in the case of the Barrier (multiple) function, which exhibited smooth control flow and required minimal energy to stabilize a wide range of initial states.

B. *MuJoCo Walker Environments*

The walker environments consist of the Humanoid, the Half-Cheetah, and the Ant running on the MuJoCo physics engine [31]. They have considerably more complexity than Cartpole, with Half-Cheetah having six joints, to Humanoid

with 17 joints and a much larger observation space. These environments have a common task of walking forward as fast as possible without falling over.

The metrics used to evaluate the performance of the BF-based quadratic and exponential rewards are (i) *Relative Actuation Effort*, computed from the episodic energy-velocity curve (Fig. 9), (ii) *Training Speed*, computed from the velocity-timestep curve (Fig. 8) and (iii) *Safety*, demonstrated by maximum episodic angle reached by each joint throughout an episode (Fig. 10). All the experiments were repeated for ten random seeds. The optimal scaling factors mentioned in Eq. (12) were found by performing a grid search.

The energy $\mathcal{E}_w$ for a time step for a walker w was calculated by (17).

$$\mathcal{E}_w = \sum_{j \in \mathcal{J}_w} \Delta\theta_j \cdot \tau_j \tag{17}$$

Where $\mathcal{J}_w$ is the set of all joints of the walker, τ_j is the torque exerted by the agent on the joint j, and $\Delta\theta_j$ is the change in the joint angle. The episodic energy is the sum of these over an episode. Since the task of these environments is to maximise the velocity v_w, we define the *Actuation Effort* (given in Table. I) as

$$(\text{Actuation Effort})_w = \frac{\mathcal{E}_w^{\text{episodic}}}{v_w^2} \tag{18}$$

This term represents the energy expended, or the effort made by the agent to achieve a certain kinetic energy.

1) Humanoid: The Humanoid environment is complex, with 17 actions and 376 observations by default. We reduced the observation space dimension to 129, containing all essential information while still being an MDP. Table I shows that when trained with the exponential BF formulation, the agent takes only about 49% actuation energy to achieve the same kinetic energy as the vanilla reward. Fig. 9 (Left) also shows that the BF-based rewards reach a higher maximum velocity for the same training timesteps.

From Fig. 8 (Left), we can also observe that the exponential BF-based reward converges to a higher velocity in much fewer training timesteps. Table I shows that exponential BF reward converges 1.56 times faster, while quadratic BF reward is no worse than the vanilla reward.

2) Ant: The Ant environment, with eight actions and 27 observations, also follows the previous trends. From Table I, we see that for the exponential BF reward, the agent uses 63% actuation energy as that of the vanilla reward. Quadratic BF reward has even better efficiency and consumes only 62% of energy. Further, we can see a drastic increase in training speed, with exponential BF converging 2.77 times faster than vanilla (from Table I). These results are also qualitatively seen in Fig. 8 (Right).

3) Half-cheetah: Although the Half-Cheetah environment is much simpler, with only six actions and 17 observations, it follows the same trends as the Humanoid environment. From Table I, the exponential BF reward consumes 92% compared to the vanilla reward; however, without much improvement in training speed. In contrast, the quadratic BF reward trains 1.47 times faster than vanilla.

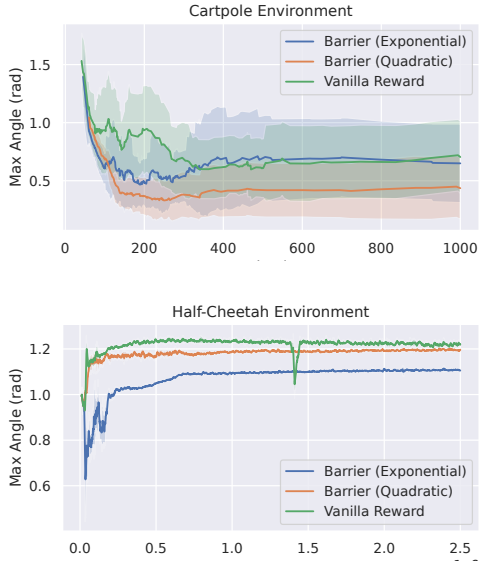

Fig. 10: (Top) Maximum angle (rad) of the Cartpole throughout each episode. (Bottom) Maximum angle (rad) of the Half-Cheetah across all joints throughout each episode.

C. Safety

We utilized the maximum joint angle throughout an episode as a metric to assess the safety performance of our BF-Based reward formulations, which we present in Fig. 10 for both the Cartpole and Half-Cheetah environments. To account for the multiple joints in the Half-Cheetah environment, we normalized the angles using their maximum and minimum values and recorded the maximum normalized angle for each episode. Our results indicate that the BF-based rewards preserve safety by constraining the maximum joint angle within a safe margin from the normalized angle limits, as demonstrated in the plots. This finding provides qualitative evidence of the efficacy of our reward formulations in maintaining agent safety.

D. Statistical Tests

To test whether the difference between the performance of the BF-based reward formulations and the vanilla reward is statistically significant, we perform Welch's t-test [32] for the walker environments. We seek to reject the null hypothesis H_0 that $\mu_{\text{diff}} = 0$, where $\mu_{\text{diff}} = \mu_{r'} - \mu_r$, is the difference between the average episodic velocity of BF based reward r' and vanilla reward r. The p-value, which is the probability of a false positive, for $N_E = 100$ experiments over ten random seeds are reported below. For Ant, $p_{\text{exp}} = 2.15 \cdot 10^{-3}$ and $p_{\text{quad}} = 3.81 \cdot 10^{-3}$. For Half-Cheetah, $p_{\text{exp}} = 1.28 \cdot 10^{-4}$ and

p_{quad} = $8.03 \cdot 10^{-5}$. Finally for Humanoid, p_{exp} = $9.9 \cdot 10^{-4}$ and p_{quad} = $2.85 \cdot 10^{-2}$. Thus we can conclude that except for the Quadratic-BF reward in the Humanoid environment, all the results are significant at level $\alpha = 10^{-2}$ since all the p-values are less than 10^{-2}.

VI. CONCLUSIONS

We presented Barrier Function (BF) inspired reward shaping, a safety-oriented and easy-to-implement reward shaping formulation. This approach is based on theoretical principles of safety provided by Barrier Functions. The shaping term is designed to incentivize agents to stay within a set of safe states during training, thus increasing training efficiency and exploration safety. To illustrate our formulation process, we proposed two barrier functions: *Exponential* and *Quadratic*.

While prior works, e.g. by Cheng et al. [7] achieved high training efficiency and state safety, their framework needs the system dynamics model. In contrast, our method eliminates this need, thus being easy to implement in complex environments like Humanoid while performing comparatively well.

We evaluate our reward-shaping formulation in environments like CartPole, Humanoid, Half-Cheetah and Ant using metrics like training speed, energy and safety. The results indicate that our formulation is an easy way to introduce safety and efficiency in RL training.

REFERENCES

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[4] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic systems*, vol. 1, no. 2, pp. 123–140, 1984.

[5] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013. [Online]. Available: https://doi.org/10.1177/0278364913495721

[6] B. Marthi, "Automatic shaping and decomposition of reward functions," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 601–608. [Online]. Available: https://doi.org/10.1145/1273496.1273572

[7] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3387–3395, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4213

[8] M. Grzes and D. Kudenko, "Plan-based reward shaping for reinforcement learning," in *2008 4th International IEEE Conference Intelligent Systems*, vol. 2, 2008, pp. 10–22–10–29.

[9] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99. Citeseer, 1999, pp. 278–287.

[10] Y. Dong, X. Tang, and Y. Yuan, "Principled reward shaping for reinforcement learning via lyapunov stability theory," *Neurocomputing*, vol. 393, pp. 83–90, 2020.

[11] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete event dynamic systems*, vol. 13, no. 1-2, pp. 41–77, 2003.

[12] R. Bellman, "Dynamic programming," *science*, vol. 153, no. 3731, pp. 34–37, 1966.

[13] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[14] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[15] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99. Citeseer, 1999, pp. 278–287.

[16] B. Badnava and N. Mozayani, "A new potential-based reward shaping for reinforcement learning agent," *arXiv preprint arXiv:1902.06239*, 2019.

[17] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: A new approach of reward shaping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 931–15 941, 2020.

[18] M. Grzes and D. Kudenko, "Plan-based reward shaping for reinforcement learning," in *2008 4th International IEEE Conference Intelligent Systems*, vol. 2. IEEE, 2008, pp. 10–22.

[19] S. M. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proceedings of the 11th international conference on autonomous agents and multiagent systems*. IFAAMAS, 2012, pp. 433–440.

[20] M. Grzes, "Reward shaping in episodic reinforcement learning," 2017.

[21] P. Goyal, S. Niekum, and R. J. Mooney, "Using natural language for reward shaping in reinforcement learning," *arXiv preprint arXiv:1903.02020*, 2019.

[22] Y. Dong, X. Tang, and Y. Yuan, "Principled reward shaping for reinforcement learning via lyapunov stability theory," *Neurocomputing*, vol. 393, pp. 83–90, 2020.

[23] B. Marthi, "Automatic shaping and decomposition of reward functions," in *Proceedings of the 24th International Conference on Machine learning*, 2007, pp. 601–608.

[24] O. Marom and B. Rosman, "Belief reward shaping in reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[25] A. Balakrishnan and J. V. Deshmukh, "Structured reward shaping using signal temporal logic specifications," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3481–3486.

[26] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Policy transfer using reward shaping." in *AAMAS*, 2015, pp. 181–188.

[27] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[28] P. Wawrzyński, "A cat-like robot real-time learning to run," in *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers 9*. Springer, 2009, pp. 380–390.

[29] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[30] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[31] 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*, 2012, pp. 5026–5033.

[32] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "How many random seeds? statistical power analysis in deep reinforcement learning experiments," *arXiv preprint arXiv:1806.08295*, 2018.