Dynamic Mirror Descent based Model Predictive Control for Accelerating Robot Learning

Utkarsh A. Mishra[1,*], Soumya R. Samineni[1,*], Prakhar Goel[2], Chandravaran Kunjeti[3], Himanshu Lodha[1], Aman Singh[1], Aditya Sagi[1], Shalabh Bhatnagar[1] and Shishir Kolathaya[1]

***Abstract*— Recent works in Reinforcement Learning (RL) combine model-free (Mf)-RL algorithms with model-based (Mb)-RL approaches to get the best from both: asymptotic performance of Mf-RL and high sample-efficiency of Mb-RL. Inspired by these works, we propose a hierarchical framework that integrates online learning for the Mb-trajectory optimization with off-policy methods for the Mf-RL. In particular, two loops are proposed, where the Dynamic Mirror Descent based Model Predictive Control (DMD-MPC) is used as the inner loop Mb-RL to obtain an optimal sequence of actions. These actions are in turn used to significantly accelerate the outer loop Mf-RL. We show that our formulation is generic for a broad class of MPC based policies and objectives, and includes some of the well-known Mb-Mf approaches. We finally introduce a new algorithm: Mirror-Descent Model Predictive RL (M-DeMoRL), which uses Cross-Entropy Method (CEM) with elite fractions for the inner loop. Our experiments show faster convergence of the proposed hierarchical approach on benchmark MuJoCo tasks. We also demonstrate hardware training for trajectory tracking in a 2R leg, and hardware transfer for robust walking in a quadruped. We show that the inner-loop Mb-RL significantly decreases the number of training iterations required in the real system, thereby validating the proposed approach.**

I. Introduction

Model-Free Reinforcement Learning (Mf-RL) algorithms are widely applied to solve tasks like dexterous manipulation [1] and agile locomotion [2], [3] as they eliminate the need to model the complex dynamics of the system. However, these techniques are data hungry and require millions of interactions with the environment. Furthermore, these characteristics highly limit successful training on hardware as undergoing such high number of transitions in hardware environments is infeasible. Thus, in order to overcome this hurdle, various works have settled for a two loop model-based approach, typically referred to as Model-based Reinforcement Learning (Mb-RL). Such strategies take the benefit of the explored dynamics of the system by learning the dynamics model, and then determining an optimal policy in this model. Hence this "inner-loop" optimization allows for a better choice of actions before interacting with the original environment.

The inclusion of model-learning in RL has significantly improved sampling efficiency [4], [5] leading numerous works in this direction. Such a learned model has proven to be very beneficial in developing robust control strategies (PDDM [6] and PETS [7]) based on predictive simulations. The process of planning with the learnt model is mainly motivated by the Model Predictive Control (MPC), which is a well known strategy used in classical real-time control. Given the model and the cost formulation, a typical MPC structure can be formulated in the form of a finite horizon trajectory optimization problem. With such a view and exploiting the approximated dynamics, methods like Cross-Entropy Method (CEM) [8] and Model Predictive Path Integral (MPPI) [9] have been used to achieve high reward gains. Model Based Policy Optimisation (MBPO) [10] introduced the outer loop policy to collect transition to train approximate model and sample over it to train the policy. Gradually, POLO [11] introduced the use of value functions for terminal rewards in such model based settings, optimizing for which can motivate the policy to converge faster. All these were particularly incorporated by Model Predictive Actor Critic (MoPAC) [12] along with optimization in the inner loop using MPPI to accelerate Mb-Mf learning. Such works demonstrate model-based (Mb) additions to typical model-free (Mf) algorithms accelerating the latter ones with significant sampling efficiency.

With a view toward strengthening existing Mb-Mf approaches for learning, we propose a generic framework that integrates a model-based optimization scheme with model-free off-policy learning. Motivated by the success of online learning algorithms [13] in RC buggy models, we combine them with off-policy Mf learning, thereby leading to a two-loop Mb-Mf approach. In particular, we implement dynamic mirror descent (DMD) algorithms on a model-estimate of the system, and then the outer loop Mf-RL is used on the real system. The main advantage with this setting is that the inner loop is computationally light; the number of iterations can be large without effecting the overall performance. Since

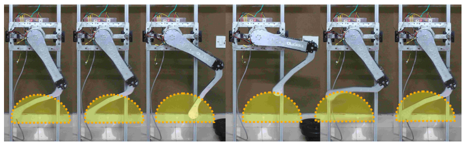

Fig. 1: Here the 2R leg is following a semi-elliptical trajectory after successful hardware-in-loop training. The number of training iterations required for the outer-loop hardware RL decreases with the help of inner-loop DMD-MPC.

[1] Department of Computer Science and Automation, Indian Institute of Science Bangalore

[2] Electronics and Communication Engineering Department, Manipal Institute of Technology India

[3] Electronics and Communication Engineering Department, National Institute of Technology Karnataka, Surathkal India

[*] These authors have contributed equally.

this is a hierarchical approach, the inner loop policy helps improve the outer loop policy, by effectively utilizing the control choices made on the approximate dynamics. This approach, in fact, provides a more generic framework for some of the Mb-Mf approaches (e.g., [12], [6]).

In addition to the proposed framework, we introduce a new algorithm Mirror-Descent Model Predictive RL (M-DeMoRL), which uses Soft actor-critic (SAC) [14] in the outer loop as off-policy RL, and CEM with elite fractions in the inner loop as DMD-MPC [13]. We show that the DMD-MPC accelerates the learning of the outer-loop by simply enriching the off-policy experience data with better choices of state-control transitions. Finally, we demonstrate direct hardware training for 2R leg tracking task (Fig. 4) and hardware transfer of policies for quadruped walking with significantly lesser environment interactions.

The paper is structured as follows: Section II will provide the preliminaries for OL for MPC as followed in the paper. Section III will describe the hierarchical framework for the proposed strategy, followed by the description of the DMD-MPC. Section IV formulates the algorithm and discusses our simulation results. Section V presents hardware training on a two-link leg manipulator followed by hardware transfer onto Stochlite quadruped. Finally, we conclude in Section VI.

II. Problem Formulation

We consider an infinite horizon Markov Decision Process (MDP) given by $\{\mathcal{X},\mathcal{U},r,P,\gamma,\rho_0\}$ where $\mathcal{X} \subset \mathbb{R}^n$ refers to set of states of the robot and $\mathcal{U} \subset \mathbb{R}^m$ refers to the set of control or actions. $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, $P : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ refers to the function that gives transition probabilities between two states for a given action, and $\gamma \in (0, 1)$ is the discount factor of the MDP. The distribution over initial states is given by $\rho_0 : \mathcal{X} \to [0, 1]$ and the policy is represented by $\pi_\theta : \mathcal{X} \to \mathcal{U}$ parameterized by $\theta \in \Theta$, a potentially feasible high-dimensional space. If a stochastic policy is used, then $\pi_\theta : \mathcal{X} \times \mathcal{U} \to [0, 1]$. For ease of notations, we will use a deterministic policy to formulate the problem. Wherever a stochastic policy is used, we will show the extensions explicitly. The system model dynamics can be expressed in the form of an equation: $x_{t+1} \sim f(x_t, u_t)$, where f is the stochastic transition map. We can obtain an estimate of this map/model as f_ϕ, which is parameterized by ϕ. The goal is to maximise the expected return :

$$\theta^* := \arg\max_\theta \mathbb{E}_{\rho_0,\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r(x_t, u_t)\right], \tag{1}$$

$$x_0 \sim \rho_0, \quad x_{t+1} \sim f(x_t, \pi_\theta(x_t)). \tag{2}$$

Model Predictive Control (MPC). Given the complexity of solving infinite horizon problems (1) via reinforcement learning (RL), there has been a lot of push toward deployment of Model Predictive Control (MPC) based methods for a finite H-step horizon [4]. Denote the sequence of H states and controls as $\mathbf{x}_t = (x_{t,0}, x_{t,1}, \ldots, x_{t,H})$, and $\mathbf{u}_t = (u_{t,0}, u_{t,1}, \ldots, u_{t,H-1})$, with $x_{t,0} = x_t$. The cost for H steps is given by

$$C(\mathbf{x_t}, \mathbf{u_t}) = \sum_{h=0}^{H-1}\gamma^h c(x_{t,h}, u_{t,h}) + \gamma^H c_H(x_{t,H}) \tag{3}$$

where, $c(x_{t,h}, u_{t,h}) = -r(x_{t,h}, u_{t,h})$ is the cost incurred (for the control problem) and $c_H(x_{t,H})$ is the terminal cost[1]. Each of the $x_{t,h}, u_{t,h}$ are related by

$$x_{t,h+1} \sim f_\phi(x_{t,h}, u_{t,h}), \quad h = 0, 1, \ldots, H-1, \tag{4}$$

with f_ϕ being the estimate of f. We will use the short notation $\mathbf{x_t} \sim f_\phi$ to represent (4).

The solution for (1) with the finite horizon cost, and with the model estimate f_ϕ can be obtained via MPC [4], [5], [15]. At every step t, optimal sequence of actions/controls are obtained. The first action is then applied on the real system to obtain the next state. There are several ways to solve the MPC setup, and online learning (OL) is one such approach, which is described next.

Online Learning for MPC. Online Learning (OL) is a generic sequential decision making technique that makes a decision at time t to optimise for the regret over time. Since MPC also involves taking optimal decisions sequentially, [13] proposed to use online learning via dynamic mirror descent (DMD) algorithms. DMD is reminiscent of the proximal update with a Bregman divergence that acts as a regularization to keep the current control distribution close to the previous one. For a rollout time of H, we sample the tuple $\mathbf{u_t}$ from a control distribution (π_η) parameterized by $\eta \in \mathcal{P}$, where $\mathcal{P}$ is the parameter set. To be more precise, η_t is also a sequence of parameters: $\eta_t = (\eta_{t,0}, \eta_{t,1}, \ldots, \eta_{t,H-1})$ which yield the control tuple $\mathbf{u_t}$. Therefore, given the control distribution paramater η_{t-1} at round $t-1$, we obtain η_t at round t from the following update rule:

$$J(x_t, \tilde{\eta}_t) := \mathbb{E}_{\mathbf{u_t}\sim\pi_{\tilde{\eta}_t}, \mathbf{x_t}\sim f_\phi}\left[C(\mathbf{x_t}, \mathbf{u_t})\right], \tilde{\eta}_t := \Phi_t(\eta_{t-1}) \tag{5}$$

$$\eta_t = \arg\min_\eta \left[\alpha_t\langle\nabla_{\tilde{\eta}_t} J(x_t, \tilde{\eta}_t), \eta\rangle + D_\psi(\eta\|\tilde{\eta}_t)\right], \tag{6}$$

where J is the MPC objective/cost expressed in terms of x_t and $\pi_{\tilde{\eta}_t}$, Φ_t is the shift model, $\alpha_t > 0$ is the step size for the DMD, and D_ψ is the Bregman divergence for a strictly convex function ψ. Note that the shift parameter Φ_t is critical for convergence of this iterative procedure. Typically, this is ensured by making it dependent on the state x_t. In particular, for the proposed two-loop scheme, we make Φ_t dependent on the outer loop policy $\pi_\theta(x_t)$. Also note that resulting parameter η_t is still state-dependent, as the MPC objective J is dependent on x_t.

With the two policies, π_θ and π_{η_t} at time t, we aim to develop a synergy in order to leverage the learning capabilities of both of them. In particular, the ultimate goal is to learn them in "parallel", i.e., in the form of two loops. The outer loop optimizes π_θ and the inner loop optimizes π_{η_t} for the MPC Objective.

[1]It will be shown later that in the proposed two-loop scheme, the terminal cost can be the value function obtained from the outer loop.

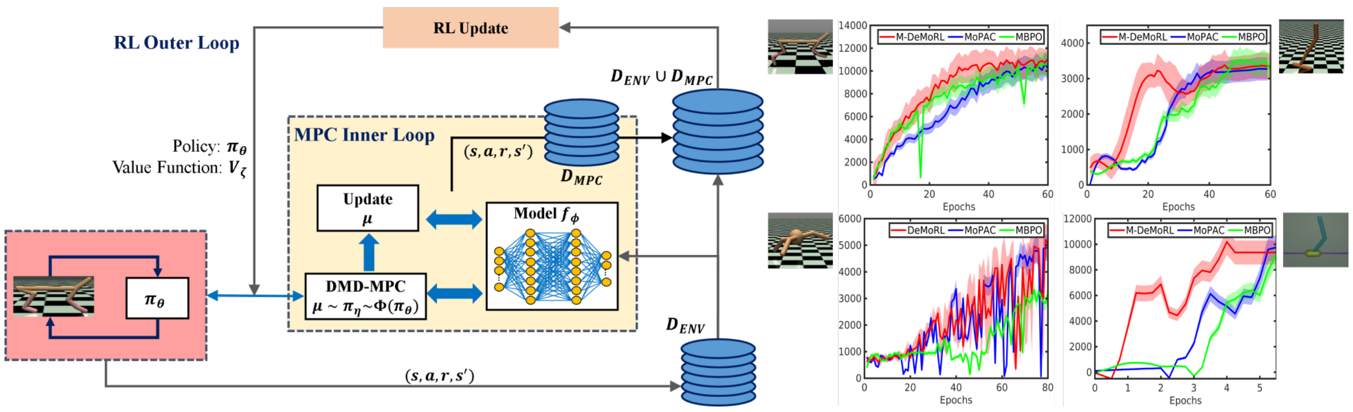

Fig. 2: (Left) The proposed hierarchical structure of Mirror-Descent Model Predictive Reinforcement Learning (M-DeMoRL) with an inner loop DMD-MPC update and an outer loop RL update. (Right) The Mujoco OpenAI benchmark environments solved by the proposed algorithm with corresponding performance plots compared with other model based algorithms: MoPAC and MBPO.

III. Methodology

In this section, we discuss a generic approach for combining model-free (Mf) and model-based (Mb) reinforcement learning (RL) algorithms through DMD-MPC. In classical Mf-RL, data from the interactions with the original environment are used to obtain the optimal policy parameterized by θ. While the interactions of the policy are stored in memory buffer, $\mathcal{D}_{ENV}$, for offline batch updates, they are used to optimize the parameters ϕ for the approximated dynamics of the model, f_ϕ. Such an optimized policy can then be used in the DMD-MPC strategy to update the control distribution, π_η. The controls sampled from this distribution are rolled out with the model, f_ϕ, to collect new transitions and store these in a separate buffer $\mathcal{D}_{MPC}$. Finally, we update θ using both the data i.e., from the buffer $\mathcal{D}_{ENV} \cup \mathcal{D}_{MPC}$ via one of the off-policy approaches (e.g. DDPG [16], SAC [14]). In this work, we majorly demonstrate this using Soft Actor-Critic (SAC) [14]. This gives a generalised hierarchical framework with two loops: DMD-MPC forming an inner loop and model-free RL in the outer loop. A graphical representation of the described framework is given in Figure 2.

There are two salient features in the two-loop approach:

1. At round t, we obtain the shifting operator Φ_t by using the outer loop parameter θ. This is in stark contrast to the classical DMD-MPC method shown in [13], wherein the shifting operator is only dependent on the control parameter of the previous round η_{t-1}.
2. Inspired by [11], [12], the terminal cost $c_H(x_{t,H}) = -\ V_\zeta(x_{t,H})$ is the value of the terminal state for the finite horizon problem as estimated by the value function (V_ζ, parameterized by ζ) associated with the outer loop policy, π_θ. This will efficiently utilise the model learned via the RL interactions and will in turn optimize π_θ with the updated setup.

Since there is limited literature on theoretical guarantees of DRL algorithms, it is difficult to show convergences and regret bounds for the proposed two-loop approach. However, there are guarantees on regret bounds for DMD algorithms in the context of online learning [17]. We reuse their following definitions:

$$G_J \triangleq \max_{\eta_t \in \mathcal{P}} \|\nabla J(\eta_t)\|, \quad M \triangleq \frac{1}{2} \max_{\eta_t \in \mathcal{P}} \|\nabla \psi(\eta_t)\|$$
$$D_{max} \triangleq \max_{\eta_t, \eta_t' \in \mathcal{P}} D_\psi(\eta_t \| \eta_t'),$$
$$\Delta_{\Phi_t} \triangleq \max_{\eta_t, \eta_t' \in \mathcal{P}} D_\psi(\tilde{\eta}_t \| \tilde{\eta}_t') - D_\psi(\eta_t \| \eta_t').$$

By a slight abuse of notations, we have omitted x_t in the arguments for J. We have the following:

Lemma 1: Denote the optimal cost obtained for (5) for the real model by J_r. Then the difference in the returns between the real and the approximated model (J_r and J respectively) is

$$J\left(x_t, \tilde{\eta}_t\right) - J_r\left(x_t, \tilde{\eta}_t\right) \le 2c_{max} \frac{(H-1)\gamma^{H+1} - H\gamma^H + \gamma}{(1-\gamma)^2} \epsilon_f + \gamma^H 2\ V_{max} H \epsilon_f \triangleq R_{f,H}$$

using Lemma B.3 in [10]. Here, ϵ_f is the uncertainty in dynamics model approximation.

Theorem 1: Let the $\tilde{\eta}_t$ be sequence obtained from outer loop policy π_θ, and let $\eta_t^\star$ be the policy obtained from $\pi_{\theta^\star}$, where $\theta^\star$ is optimal; then for the class of convex MPC objectives J, the maximum regret incurred for the T decision timesteps, each with H-step planning can be formulated as

$$Re_T\left(\boldsymbol{\eta}_T\right) := \sum_{t=0}^{T} J\left(\tilde{\eta}_t\right) - J_r\left(\eta_t^\star\right)$$
$$\le \frac{D_{\max}}{\alpha_{T+1}} + \frac{4M}{\alpha_T} W_{\Phi_t}\left(\boldsymbol{\eta}_T\right) + \frac{G_J^2}{2\sigma} \sum_{t=1}^{T} \alpha_t + T\ R_{f,H}$$

where $\boldsymbol{\eta}_T$ is the vector of parameters decided at every decision step, W_{Φ_t} is given by

$$W_{\Phi_t} = \sum_{t=0}^{T} \|\eta_{t+1}^\star - \tilde{\eta}_{t+1}\|, \tag{7}$$

and other notations are derived from [17].
Proofs of Lemma 1 and Theorem 1 are provided in Appendix.

As we obtain the shift model at every iteration using the outer loop RL policy and the learned dynamics, the maximum regret decreases as the policy θ converges to $\theta^\star$, the sequence $\tilde{\eta}_{t+1}$ approaches $\eta^\star_{t+1}$. In other words, the regret is minimum when the infinite horizon optimal outer loop policy is efficient enough in identifying a finite H-step horizon optimal inner loop policy.

DMD-MPC with Exponential family. We consider a parametric set of probability distributions for our control distributions in the exponential family, given by natural parameters η, sufficient statistics δ and expectation parameters μ [13]. Further, we set Bregmann divergence in (6) to the KL divergence. The natural parameter of control distribution, $\tilde{\eta}_t$, is obtained with the proposed shift model Φ_t from the outer loop RL policy π_θ by setting the expectation parameter of $\tilde{\eta}_t$: $\tilde{\mu}_\mathbf{t} = \pi_\theta(\mathbf{x_t})$. Note that we have overloaded the notation π_θ to map the sequence $\mathbf{x_t}$ to $\tilde{\mu}_\mathbf{t}$, which is the sequence of $\tilde{\mu}_{t,h} = \pi_\theta\left(x_{t,h}\right)$[2]. Then, we have the following gradient of the cost:

$$\nabla_{\tilde{\eta}_t} J(\mathbf{x_t}, \tilde{\eta}_t) = \mathbb{E}_{\mathbf{u_t}\sim\pi_{\tilde{\eta}_t}, \mathbf{x_t}\sim f_\phi}\left[C(\mathbf{x_t}, \mathbf{u}_t)(\delta(\mathbf{u_t}) - \tilde{\mu}_\mathbf{t})\right], \tag{8}$$

In the presented setup, we choose Gaussian distribution for control and $\delta(\mathbf{u_t}) := \mathbf{u_t}$. We finally have the following update rule for the expectation parameter [13]:

$$\mu_\mathbf{t} = (1-\alpha)\,\tilde{\mu}_\mathbf{t} + \alpha\mathbb{E}_{\pi_{\tilde{\eta}_t}, f_\phi}\left[C\left(\mathbf{x_t}, \mathbf{u_t}\right)\mathbf{u_t}\right]. \tag{9}$$

Based on the data collected in the outer loop, the inner loop is executed via DMD-MPC as follows:

- Step 1. Considering H-step horizon, for $h = 0, 1, 2, \ldots, H-1$, obtain

$$\tilde{\eta}_{t,h} = \Sigma^{-1}\tilde{\mu}_{t,h}, \quad \tilde{\mu}_{t,h} = \pi_\theta(x_{t,h}) \tag{10}$$
$$u_{t,h} \sim \pi_{\tilde{\eta}_{t,h}} \tag{11}$$
$$x_{t,h+1} \sim f_\phi(x_{t,h}, u_{t,h}). \tag{12}$$

  where Σ represents the covariance for control distribution.
- Step 2. Collect $\tilde{\eta}_t = (\tilde{\eta}_{t,0}, \tilde{\eta}_{t,1}, \ldots, \tilde{\eta}_{t,H-1})$, and apply DMD-MPC (6) to obtain η_t.

For the presented work, we use CEM with the method of elite fractions that allows us to select only the best transitions. This is given by the following:

$$J(\mathbf{x_t}, \tilde{\eta}_t) := -\log \mathbb{E}_{\pi_{\tilde{\eta}_t}, f_\phi}\left[\mathbf{1}\left\{C\left(\mathbf{x_t}, \mathbf{u_t}\right) \leq C_{t,\max}\right\}\right] \tag{13}$$

where we choose $C_{t,max}$ as the top elite fraction from the estimates of rollouts. It is worth noting that both CEM and MPPI belong to the same family of objective function utilities [13].

[2]Note that if the policy is stochastic, then $\tilde{\mu}_{t,h} \sim \pi_\theta\left(x_{t,h}\right)$. This is similar to the control choices made in [12, Algorithm 2, Line 4].

Algorithm 1: M-DeMoRL Algorithm

```
1  Initialize SAC and Model $f_\phi$, Environment Parameters
2  Initialize memory buffer: $D_{ENV}$
3  for max iterations do
4      $D_{ENV} \leftarrow D_{ENV} \cup \{x, u, r, x'\}, u \sim \pi_\theta(x)$
5      for each model learning epoch do
6          Train model $f_\phi$ on $D_{ENV}$ with loss : $J_\phi = \|\{(x'-x), r\} - f_\phi(x,u)\|_2$
7      end
8      Initialize $D_{MPC}$
9      Calculate M and H from schedule
10     for DMD-MPC iterations do
11         Sample $x_{t,0}$ uniformly from $D_{ENV}$
12         Simulate M trajectories of H steps horizon: (10), (11) and (12)
13         Perform CEM to get optimal action sequence: $\mu_\mathbf{t}$ (14) and (9)
14         Collect complete trajectory: $\mathbf{x_t}, \mathbf{r_t} \sim f_\phi(x_{t,0}, \mu_\mathbf{t})$
15         Add all transitions to $D_{MPC}$ : $D_{MPC} \leftarrow D_{MPC} \cup \{x_{t,h}, u_{t,h}, \hat{r}_{t,h}, x_{t,h+1}\}$
16     end
17     for each gradient update step do
18         Update SAC parameters using data from $D_{ENV} \cup D_{MPC}$
19     end
20 end
```

IV. Implementation and Results

In this section, we implement the two-loop hierarchical framework as explained in the previous section and structure the specific details about the algorithm associated with the proposed work. This will be compared with the existing approaches MoPAC [12] and MBPO [10] on the benchmark MuJoCo control environments.

A. Algorithm:

M-DeMoRL algorithm derives from other Mb-Mf methods in terms of learning dynamics and follows a similar ensemble dynamics model approach. We have shown it in Algorithm 1. There are three parts in this algorithm: Model learning, Soft Actor-Critic and DMD-MPC. We describe them below.

Model learning. The functions to approximate the dynamics and reward function of the system are K-probabilistic deep neural networks [15] cumulatively represented as $\{f_{\phi_1}, f_{\phi_2}, \ldots, f_{\phi_K}\}$ Using the inputs as the current state and actions, the ensemble model fits all the probabilistic models to output change in states and the reward obtained during the transition. Such a configuration is believed to account for the epistemic uncertainty of complex dynamics and overcomes the problem of over-fitting generally encountered by using single models [7].

SAC. Our implementation of the proposed algorithm uses Soft Actor-Critic (SAC) [14] as the model-free RL counterpart. Based on principle of entropy maximization, the choice of SAC ensures sufficient exploration motivated by the soft-policy updates, resulting in a good approximation of the underlying dynamics.

DMD-MPC. Here, we solve for $\mathbb{E}_{\pi_{\tilde{\eta}_t}, f_\phi}\left[C\left(\mathbf{x_t}, \mathbf{u_t}\right) \mathbf{u_t}\right]$ using a Monte-Carlo estimation approach. For a horizon length of H, we collect M trajectories using the current policy π_{θ_t} and the more accurate dynamic models from the ensemble having lesser validation losses. For all trajectories, the complete cost is calculated using a deterministic reward estimate and the value function through (2). After getting the complete state-action-reward H-step trajectories we execute the following based on the CEM [8] strategy:

- Step 1. Choose the $p\%$ elite trajectories according to the total H-step cost incurred. We set $p = 10\ \%$ for our experiments, and denote the chosen respective action trajectories and costs as U_{elites} and C_{elites} respectively. Note that we have also tested for other values of p, and the ablations are shown later in this section.
- Step 2. Using U_{elites} and C_{elites} we calculate $\mathbb{E}_{\pi_{\tilde{\eta}_t}, f_\phi}\left[C\left(\mathbf{x_t}, \mathbf{u_t}\right) \mathbf{u_t}\right]$ as the reward weighted mean of the actions i.e.

$$\mathbf{g_t} = \frac{\sum_{i \in elites} C_i\ U_i}{\sum_{i \in elites} C_i} \tag{14}$$

and update the current policy actions, $\tilde{\mu}_\mathbf{t} = \pi_{\theta_i}(\mathbf{x_t})$ according to (9).

B. Experiments and Comparison

Several experiments were conducted on the MuJoCo [18] continuous control tasks with the OpenAI-Gym benchmark and the performance was compared with recent related works MoPAC [12] and MBPO [10]. First, we discuss the hyperparameters used for all our experiments and then the performance achieved in the conducted experiments.

As the baseline of our framework is built upon MBPO implementation, we derive the "same hyperparameters" for our experiments and all the baseline algorithms. We compare the results of three different seeds and the reward performance plots are shown in Figure 2(right).

For the inner DMD-MPC loop we choose a varying horizon length from $5 - 15$ and perform 100 trajectory rollouts. With our elite fraction as 10%, the updated model-based transitions are added to the MPC-buffer. This process is iterated with a varying batch-size with maximum of $10,000$ thus completing the DMD-MPC block in Algorithm 1. These variable batch size and horizon length allows us to exploit the models more when we have achieved significant learning considering uncertainties and distribution shifts. Also, as evident from Theorem 1, increasing H directly increases the maximum regret incurred. Thus, we start from low horizon length as the regret incurred is more in the initial phases and gradually increase the horizon length to exploit the capabilities of MPC. Further, the number of interactions with the true environment for outer loop policy were kept constant to 1000 for each epoch, same as MoPAC and MBPO.

We clearly note an accelerated progress for all the environments, with approximately 30% faster rate in the reward

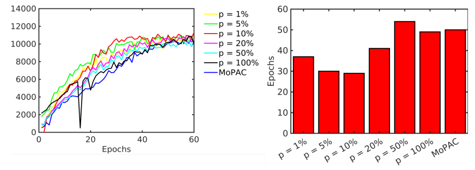

Fig. 3: Ablation study for elite percentage: Reward performance curve (left) and Acceleration analysis as epochs to reach 10000 rewards (right)

performance curve as compared to best of prior works. Our rewards in Ant-v2 were comparable with MoPAC but still significantly better than MBPO. We would like to emphasize that our final rewards are eventually the same as achieved by MoPAC and MBPO, however the progress rate is faster for all our experiments with lesser true environment interactions.

C. Ablation study on elite percentage

Given the sequence of controls $\mu_\mathbf{t}$, we collect the resulting trajectory and add them to our buffer. Therefore, the quality of $\mu_\mathbf{t}$ is a significant factor affecting the quality of data used for the outer loop RL-policy. The selection strategy being CEM, a quality metric is dependent on the choice of elite fractions p.

We perform an ablation study for 6 values of $p = 1, 5, 10, 20, 50$ and 100% on HalfCheetah-v2 OpenAI gym environment. The analysis was performed based on the reward performance curves as shown in Fig. 3 (left). Additionally, we realize the number of the epochs required to reach a certain level of performance as a good metric to measure acceleration achieved. Such an analysis is provided in Fig. 3 (right). We make the following observations:

- Having a lesser value of p might ensure that learned dynamics is exploited the most, but decreases the exploration performed in the approximated environment.
- Similarly, having higher value of p on the other hand will do more exploration using a "not-so-perfect" policy and dynamics.

Thus, the elite fraction balances between exploration and exploitation.

V. Hardware Training and Transfer

In this section, we extend M-DeMoRL to more complex dynamical systems like 2R quadruped leg and a complete quadruped walking task as well. In the first case, we do direct algorithm transfer to execute learning with hardware, whereas for quadruped we show generation of hardware transferable policies.

Leg trajectory tracking: The main idea here is to track a fixed end-foot trajectory with a 2R quadruped robot leg (as shown in Figure 4 left). The leg is torque controlled and equipped with hall effect sensors and BLDC motors. In order to perform direct hardware training, we accomplish torque control at 500 Hz, with the motor controller running at 40KHz. We define the task as tracking an elliptical trajectory

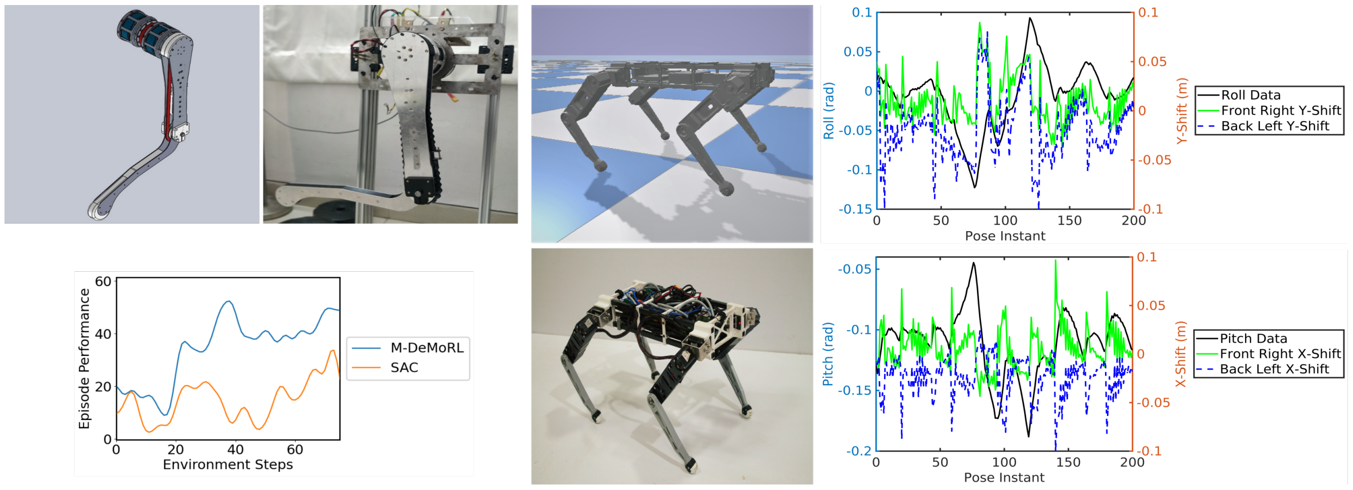

Fig. 4: (Left) Two link Leg Manipulator and corresponding hardware test setup followed by Stochlite quadruped Model Design, Pybullet Simulation and Hardware as incorporated for hardware transfer. The hardware-in-loop training for the leg shows faster reward gains compared SAC. Similarly, the transfered policy in StochLite shows desirable variations in the X and Y shifts based on the torso orientation.

(as shown in Figure 4. The reward is chosen as follows:

$$r_{leg} := 0.8e^{-\|p-p_d\|^2} + 0.2e^{-\|\dot{p}-\dot{p}_d\|^2}, \tag{15}$$

where $p, p_d \in \mathbb{R}^2$ are the actual and desired end-effector positions of the leg. We observed enhanced learning performance as compared to baseline SAC within 20 epochs (20k interactions with the real system). Fig. 1 shows the tile of trajectory tracking from start to end. The evaluation on convergence of M-DeMoRL was observed till 80 epochs and we concluded at reward gains of 2 times the observed performance with SAC baseline training. This clearly illustrates the accelerating nature of M-DeMoRL and justifies feasibility of training directly on the hardware setup.

Quadrupedal walking: We extend our approach to StochLite (shown in Figure 4 right) for walking, which is a low-cost robot consisting of 12 servos and an inertial measurement unit (IMU) to detect the body pose. StochLite is also capable of measuring joint angles (hip and knee) and torques via encoders and motor current sensors respectively. The robot dynamics model consists of 6 floating degrees-of-freedom and 12 actuated degrees-of-freedom. The simulator for StochLite is PyBullet [19]. The states consist of the torso pose, velocity (both translational and rotational components) and the slope orientation (roll and pitch). Slope orientations are obtained via the IMU orientation and the joint angle encoders.

We execute tracking end-foot trajectories via a low-level control law, while the higher level walking control policy focuses on yielding optimal trajectory parameters similar to [20], [21]. Those parameters form the actions/controls, to be obtained from the RL policy. We use a well-defined trajectory generator, which is in turn shaped via a neural network based policy. This framework, when combined with our sample efficient M-DeMoRL algorithm, allows to train policies in much shorter time ($<$ 60 episodes of true environment interactions with 1000 interactions per episode). Figure 4 (right) shows the reactive behavior of policy in terms of the modulations applied by the neural network controller over the pre-defined elliptical trajectory. These modulations are able to shift the trajectories along X, Y and Z directions. The learned response validates that the neural policy to model the correspondence of pitch and roll with the X and Y shifts. As the pitch orientation increases, the quadruped tries to increase the X-shifts so that the leg moves forward and stabilizes the torso. Similarly, the increase in roll is accompanied by again an increase in the Y-shifts in the rolling direction. We also validate the policy with successful transfer to hardware. Previously, [22] trained policies in Minitaur quadruped in 160 epochs using SAC. Due to increasing complexity, we will consider direct hardware training for quadrupeds in future.

VI. Conclusion

We have investigated the role of leveraging the model-based optimisation with online learning to accelerate model-free RL algorithms. With the emphasis to develop a real-time controller, this work primarily defines a generalised framework that could be used with the existing MPC algorithms and off the shelf Mf-RL algorithms to train efficiently. Simulation results show that our formulation yields sample efficient algorithms, as the underlying online learning tracks for the best policy benefiting the convergence of the Mf-RL policy. We also show that the training is faster than prior Mf-Mb methods. Finally, we show hardware-in-loop training with a 2R leg and we successfully generate hardware transferable policies for quadruped walking. While preliminary hardware results are shown, we look forward to better gaits and more efficient direct training on hardware as future works. The video for our experiment can be found at: stochlab.github.io/redirects/MDeMoRLPolicies.html.

References

[1] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in Proceedings of Robotics: Science and Systems, (Pittsburgh, Pennsylvania), June 2018.

[2] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in Robotics: Science and Systems, 07 2020.

[3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," Science robotics, vol. 5, no. 47, 2020.

[4] S. Levine and V. Koltun, "Guided policy search," in Proceedings of the 30th International Conference on Machine Learning (S. Dasgupta and D. McAllester, eds.), vol. 28 of Proceedings of Machine Learning Research, (Atlanta, Georgia, USA), pp. 1–9, PMLR, 17–19 Jun 2013.

[5] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7559–7566, IEEE, 2018.

[6] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in Conference on Robot Learning, pp. 1101–1112, PMLR, 2020.

[7] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in Advances in Neural Information Processing Systems (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[8] Pourchot and Sigaud, "CEM-RL: Combining evolutionary and gradient-based methods for policy search," in International Conference on Learning Representations, 2019.

[9] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1714–1721, 2017.

[10] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in Advances in Neural Information Processing Systems (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[11] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," in International Conference on Learning Representations, 2019.

[12] A. S. Morgan, D. Nandha, G. Chalvatzaki, C. D'Eramo, A. M. Dollar, and J. Peters, "Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning," arXiv preprint arXiv:2103.13842, 2021.

[13] N. Wagener, C. an Cheng, J. Sacks, and B. Boots, "An online learning approach to model predictive control," in Proceedings of Robotics: Science and Systems, (FreiburgimBreisgau, Germany), June 2019.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in International Conference on Machine Learning, pp. 1861–1870, PMLR, 2018.

[15] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," in Advances in Neural Information Processing Systems (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning.," in ICLR (Poster), 2016.

[17] E. Hall and R. Willett, "Dynamical models and tracking regret in online convex programming," in International Conference on Machine Learning, pp. 579–587, PMLR, 2013.

[18] 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, pp. 5026–5033, IEEE, 2012.

[19] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." http://pybullet.org, 2016–2021.

[20] 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, vol. 4, no. 26, 2019.

[21] K. Paigwar, L. Krishna, S. Tirumala, N. Khetan, A. Sagi, A. Joglekar, S. Bhatnagar, A. Ghosal, B. Amrutur, and S. Kolathaya, "Robust quadrupedal locomotion on sloped terrains: A linear policy approach," 2020.

[22] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," arXiv preprint arXiv:1812.11103, 2018.