Supplemental Materials: Model-based Reinforcement Learning for Continuous Control with Posterior Sampling

A. Proof of Lemma 1

We first prove the results in $\mathbb{R}^d$ when $d = 1$: Let $p_1(x), p_2(x)$ be the probability density functions for P_1, P_2 respectively. By the property of symmetric distribution, we have that $p_1(\mu_1 + x) = p_1(\mu_1 - x)$ and $p_2(\mu_2 + x) = p_2(\mu_2 - x)$ for any $x \in \mathbb{R}^d$. Let $\mu_2 > \mu_1$ without loss of generality.

Since $\boldsymbol{\epsilon_1}$ and $\boldsymbol{\epsilon_1}$ share the same distribution, we have that $p_1(x) = p_2(x + \mu_2 - \mu_1), \forall x \in \mathbb{R}^d$.

Note that $p_1(x) = p_2(x)$ at $x = \frac{\mu_1+\mu_2}{2}$. Thus the total variation difference between p_1 and p_2 can be simplified as twice the integration of one side due to symmetry:

$$\int_{-\infty}^{\infty} |p_2(x) - p_1(x)|dx = \int_{-\infty}^{\frac{\mu_1+\mu_2}{2}} |p_2(x) - p_1(x)|dx + \int_{\frac{\mu_1+\mu_2}{2}}^{\infty} |p_2(x) - p_1(x)|dx = 2\int_{\frac{\mu_1+\mu_2}{2}}^{\infty} |p_2(x) - p_1(x)|dx, \tag{12}$$

where the last equation come from

$$\begin{aligned} &p_1(\frac{\mu_1 + \mu_2}{2} - x) = p_1(\mu_1 - x + \frac{\mu_2 - \mu_1}{2}) = p_1(\mu_1 + x - \frac{\mu_2 - \mu_1}{2}) \\ &= p_2(\mu_2 + x - \frac{\mu_2 - \mu_1}{2}) = p_2(\frac{\mu_1 + \mu_2}{2} + x). \end{aligned} \tag{13}$$

Case 1: If the density functions p_1 and p_2 are unimodal, we have $p_2(x) > p_1(x)$ when $x > \frac{\mu_1+\mu_2}{2}$. Let $z_1 = x - \mu_1, z_2 = x - \mu_2$, we have:

$$\begin{aligned} &\int_{\frac{\mu_1+\mu_2}{2}}^{\infty} |p_2(x) - p_1(x)|dx \\ &= \int_{\frac{\mu_1-\mu_2}{2}}^{\infty} p_2(z_2)dz_2 - \int_{\frac{\mu_2-\mu_1}{2}}^{\infty} p_1(z_1)dz_1 \\ &= \int_{\frac{\mu_1-\mu_2}{2}}^{\frac{\mu_2-\mu_1}{2}} p_2(z_2)dz_2 \\ &\leq \int_{\frac{\mu_1-\mu_2}{2}}^{\frac{\mu_2-\mu_1}{2}} p_{max}dz = p_{max}|\mu_2 - \mu_1|, \end{aligned} \tag{14}$$

where p_{max} is the maximum probiblity density of p_2, which is dependent on the variance of the shared noise distribution. The proof is completed by combing (12) and (14).

Case 2: When $p_1(x), p_2(x)$ are not unimodal, there exist C_0 such that $p_2(x)$ would be a descreasing function in x when $x > \frac{\mu_1+\mu_2}{2} + C_0(\mu_2 - \mu_1)$ (otherwise the integration of the density cannot be 1, and C_0 is a constant which is dependent on the specific distribution). Recall that $p_1(x) = p_2(x + \mu_2 - \mu_1)$, so when $x > \frac{\mu_1+\mu_2}{2} + C_0(\mu_2 - \mu_1)$,

$p_2(x) > p_2(x+\mu_2-\mu_1) = p_1(x)$. Let $z_1 = x-\mu_1, z_2 = x-\mu_2$, we have

$$\begin{aligned}
&\int_{\frac{\mu_1+\mu_2}{2}}^{\infty} |p_2(x)-p_1(x)|dx \\
&= \int_{\frac{\mu_1+\mu_2}{2}}^{\frac{\mu_1+\mu_2}{2}+C_0(\mu_2-\mu_1)} |p_2(x)-p_1(x)|dx + \int_{\frac{\mu_1+\mu_2}{2}+C_0(\mu_2-\mu_1)}^{\infty} |p_2(x)-p_1(x)|dx \\
&\le C_0 p_{max}(\mu_2-\mu_1) + \int_{\frac{\mu_1-\mu_2}{2}+C_0(\mu_2-\mu_1)}^{\infty} p_2(z_2)dz_2 - \int_{\frac{\mu_2-\mu_1}{2}+C_0(\mu_2-\mu_1)}^{\infty} p_1(z_1)dz_1 \\
&\le C_0 p_{max}(\mu_2-\mu_1) + \int_{\frac{\mu_1-\mu_2}{2}+C_0(\mu_2-\mu_1)}^{\frac{\mu_2-\mu_1}{2}+C_0(\mu_2-\mu_1)} p_{max}dz = (C_0+1)p_{max}|\mu_2-\mu_1|,
\end{aligned} \tag{15}$$

then the proof is complete by combining (12) and (15).

Now we extend the result to $\mathbb{R}^d (d \ge 2)$: Let the shared covariance matrix for the overall noise distribution be $\sigma^2 \boldsymbol{I}_d$, where the noise in each dimension is drawn independently with variance σ^2. We can rotate the coordinate system recursively to align the last axis with vector $\boldsymbol{\mu_1} - \boldsymbol{\mu_2}$, such that the coordinates of $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ can be written as $(0, 0, \cdots, 0, \hat{\mu}_1)$, and $(0, 0, \cdots, 0, \hat{\mu}_2)$ respectively, with $|\hat{\mu}_2 - \hat{\mu}_1| = \|\boldsymbol{\mu_2} - \boldsymbol{\mu_1}\|_2$.

The new covariance matrix after rotation will still be $\sigma^2 \boldsymbol{I}_d$ since the rotation matrix is orthogonal. Notice that rotation is a linear transformation on the original noises, and the original noises are independently drawn from each axis, so the new covariance matrix indicates the noises in each new axis (after rotation) can also be viewed as independent[6]. Without loss of generality, let $\hat{\boldsymbol{\mu}}_{\boldsymbol{1}} \ge \hat{\boldsymbol{\mu}}_{\boldsymbol{2}}$. Using p_1', p_2' to indicate the marginal probability density in d-th dimension, we have:

$$\begin{aligned}
&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} |p_2(\boldsymbol{x}) - p_1(\boldsymbol{x})| dx_1 dx_2 \cdots dx_d \\
&= \int_{-\infty}^{\infty} |p_2'(x_d) - p_1'(x_d)| dx_d
\end{aligned} \tag{16}$$

Then we can follow the same steps in $\mathbb{R}^1$ to finish the proof.

Remark For Gaussian noises with shared covariance $\sigma^2 \boldsymbol{I}_d$, Pinsker's inequality and the KL-divergence of two Gaussian distributions can also show that $\int |p_1(\boldsymbol{x}) - p_2(\boldsymbol{x})| d\boldsymbol{x} \le \frac{1}{\sigma}||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2$. But our upper bound for Gaussian noises is tighter: we have $\int |p_1(\boldsymbol{x}) - p_2(\boldsymbol{x})| d\boldsymbol{x} \le \sqrt{\frac{2}{\pi\sigma^2}}||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2$. Also here we develop upper bounds for a wide class of symmetric distributions.

B. Detailed comparison with previous works in Section 3.4

Here we compare our result with Corollary 2 in (Osband & Van Roy, 2014). In their Corollary 2 of linear quadratic systems, the regret bound is $\tilde{O}(\sigma C \lambda_1 n^2 \sqrt{T})$, where λ_1 is the largest eigenvalue of the matrix Q in the optimal value function $V_1(s) = s^T Q s$, where V_1 denotes the value function counting from step 1 to H within an episode, s is the initial state, reward at the i-th step $r_i = s_i^T P s_i + a_i^T R a_i + \epsilon_{P,i}$, and the state at the $i+1$-th step $s_{i+1} = As_i + Ba_i + \epsilon_{P,i}$, $i \in [H]$. However, the largest eigenvalue of Q is actually exponential in H: Recall the Bellman equation we have $V_i(s_i) = \min_{a_i} \mathbb{E}[s_i^T P s_i + a_i^T R a_i + \epsilon_{P,i} + V_{i+1}(As_i + Ba_i + \epsilon_{P,i})]$, $V_{H+1}(s) = 0$. Thus in $V_1(s)$, we can observe a term of $(A^{H-1}s)^T P (A^{H-1}s)$, and the eigenvalue of the matrix $(A^{H-1})^T P A^{H-1}$ is exponential in H.

Even if we change the reward function from quadratic to linear, say $r_i = s_i^T P + a_i^T R + \epsilon_{P,i}$ the Lipschitz constant of the optimal value function is still exponential in H since there is still a term of $(A^{H-1}s)^T P$ in $V_1(s)$. Chowdhury & Gopalan (2019) maintain the assumption of this Lipschitz property, thus there exists $\mathbb{E}[L^*]$ in their bound. As a result, there is still no

[6] If noises in each new axis are not independent, they can only be linearly related, which would result in non-zero covariance and causes contradiction.

clear dependency on H in their regret, and in their Corollary 2 of LQR, they follow the same steps as Osband & Van Roy (2014), and still maintain a term with λ_1, which is actually exponential in H as discussed. Although Osband & Van Roy (2014) mention that system noise helps to smooth future values, but they do not explore it although the noise is assumed to be subgaussian. The authors directly use the Lipschitz continuity of the underlying function in the analysis of LQR, thus they have an exponential bound on H which is very loose, and it can be improved by our analysis. (Chowdhury & Gopalan, 2019) do not explore how the system noise can improve the theoretical bound either.

C. Details for bounding the sum of posterior variances in Section 3.4

Here we slightly modify the Proof of Lemma 5.4 in (Srinivas et al., 2012) and show that $\Sigma_{i=1}^{n}\sigma_i^2(h_i) = \mathcal{O}((d_s+d_a)\log(n))$, then we can write $\Sigma_{k=1}^{[\frac{T}{H}]}\sigma_k^{'2}(h_{\text{kmax}}) = \mathcal{O}((d_s+d_a)\log[\frac{T}{H}])$ with just changes of notations.

For any $s^2 \in [0, \sigma_f^{-2}C_1]$ we have $s^2 \leq C_2\log(1+s^2)$, where $C_2 = \frac{\sigma_f^{-2}C_1}{\log(1+\sigma_f^{-2}C_1)}$. We treat C_1 as the upper bound of the variance (note that the bounded variance property for linear kernels only requires the range of all state-action pairs actually encountered in M^* not to expand to infinity as T grows, which holds in general episodic MDPs).

Lemma 5.3 in (Srinivas et al., 2012) shows that the information gain for dataset $\{h_1, ..., h_n\}$ is equal to $\frac{1}{2}\Sigma_{i=1}^{n} log(1+\sigma_f^{-2}\sigma_i^2(h_i))$, and we also have $\sigma_i^2(h_i) \leq C_2\log(1+\sigma_i^2(h_i))$. Thus we can use the upper bound of the information gain of linear kernels, which is $\mathcal{O}((d_s+d_a)\log(n))$ as presented in Theorem 5 in (Srinivas et al., 2012), to upper bound the sum of posterior variances.

D. Handling $\mathbb{E}[\Sigma_{k=1}^{[\frac{T}{H}]}\tilde{\Delta}_k(r)|\mathcal{H}_k]$ in Section 3.4

Recall that $\tilde{\Delta}_k(r) = \Sigma_{i=1}^{H}(\bar{r}^k(h_i) - \bar{r}^*(h_i))$, so we can omit section 3.2 and directly follow steps in section 3.3 to derive another upper bound which is similar to Lemma 3: $[\Sigma_{k=1}^{[\frac{T}{H}]}\tilde{\Delta}_k(r)|\mathcal{H}_k] \leq \Sigma_{k=1}^{[\frac{T}{H}]}2\sqrt{\sigma_k(h_{\text{kmax}})\log(\frac{4T}{\delta})}$ with probability at least $1-\delta$.

Then we can follow similar steps in section 3.4 to develop that $\mathbb{E}[\Sigma_{k=1}^{[\frac{T}{H}]}\tilde{\Delta}_k(r)|\mathcal{H}_k] = \tilde{\mathcal{O}}(\sqrt{dHT})$.

E. Experimental details

For model-based baselines, the average number of episodes required for convergence is presented below (convergence results for Cartpole and Pendulum can be found in Figure 1 and Figure 2 in the main paper): For model-free methods, we

Method	Reacher(r)	Pusher(r)	Reacher	Pusher
Ours	**20.2**	**146.6**	**29.8**	**151.6**
MBPO	34.6	209.4	54.2	225.0
PETS	26.2	193.4	-	-

have provided the convergence results in Figure 2 for SAC (blue dots). Model-free methods generally converge after 100 episodes for Cartpole and Pendulum, and around 1000 episodes for Pusher and Reacher.

Hyperparameters for MBPO:

env	cartpole	pendulum	pusher	reacher
env steps per episode	200	200	150	150
model rollouts per env step	400			
ensemble size	5			
network architecture	MLP with 2 hidden layers of size 200	MLP with 2 hidden layers of size 200	MLP with 4 hidden layers of size 200	MLP with 4 hidden layers of size 200
policy updates per env step	40			
model horizon	1->15 from episode 1->30	1->15 from episode 1->30	1	1->15 from episode 1->30

Table 1. Hyperparamters for MBPO

And we provide hyperparamters for MPC and neural networks in PETS:

env	pusher	reacher
env steps per episode	150	150
popsize	500	400
number of elites	50	40
network architecture	MLP with 4 hidden layers of size 200	
planning horizon	25	25
max iter	5	
ensemble size	5	

Table 2. Hyperparamters for PETS

Below are hyperparameters of our planning algorithm, which is the same with PETS, except for ensemble size (since we do not need ensembled models, hence our ensemble size is actually 1):

env	cartpole	pendulum	pusher	reacher
env steps per episode	200	200	150	150
popsize	500	100	500	400
number of elites	50	5	50	40
network architecture	MLP with 2 hidden layers of size 200	MLP with 2 hidden layers of size 200	MLP with 4 hidden layers of size 200	MLP with 4 hidden layers of size 200
planning horizon	30	20	25	25
max iter	5			

Table 3. Hyperparamters for our method

For SAC and DDPG, we use the open-source code (https://github.com/dongminlee94/deep_rl) for imple-

mentation without changing their hyperparameters. Here we thank the authors for sharing the code!

We run all the experiments on a single NVIDIA GeForce RTX-2080Ti GPU. For smaller environments like Cartpole and Pendulum, all experiments are done within an hour to run 150k steps. For Reacher and Pusher, our algorithm takes about four hours to run 150k steps, while PETS and MBPO take about three hours to run 150k steps (our extra computation cost comes from Bayesian update, and we plan to explore acceleration for that as future work). SAC and DDPG take about one hour for training 150k steps which is much faster than other baselines since they are model-free algorithms and no need to train models.