Sample-efficient Plasma Control by Planning for Optimal Trajectory Information

Viraj Mehta, Ian Char, & Jeff Schneider {virajm, ichar, schneide}@cs.cmu.edu
Robotics Insitute & Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA, USA

Willie Neiswanger & Stefano Ermon {neiswanger, ermon}@cs.stanford.edu
Computer Science Department
Stanford University
Stanford, CA, USA

Joseph Abbate, Rory Conlin, & Mark D Boyer {abbatej, wconlin}@princeton.edu, mboyer@pppl.gov
Princeton Plasma Physics Laboratory
Princeton, NJ, USA

Abstract

Many potential applications of reinforcement learning (RL) are stymied by the large numbers of samples required to learn an effective policy. This is especially true when applying RL to real-world control tasks, e.g. in the sciences or robotics, where executing a policy in the environment is costly. In popular RL algorithms, agents typically explore either by adding stochasticity to a reward-maximizing policy or by attempting to gather maximal information about environment dynamics without taking the given task into account. In this work, we develop a method that allows us to plan for exploration while taking both the task and the current knowledge about the dynamics into account. The key insight to our approach is to plan an action sequence that maximizes the expected information gain about the optimal trajectory for the task at hand. We demonstrate that our method learns strong policies with 2-200x fewer samples compared to 14 baselines on a diverse set of control tasks including 2 nuclear fusion examples in both the open-loop and closed-loop control settings.

Keywords: Reinforcement Learning, Exploration, Model-Predictive Control

1. Introduction

The potential of reinforcement learning (RL) as a general-purpose method of learning solutions to sequential decision making problems is difficult to overstate. Ideally, RL could allow for agents that learn to accomplish all manner of tasks solely through a given reward function and the agent's experience; however, RL has so far broadly fallen short of this. One major reasons for this is that typical RL methods in continuous problems require very large numbers of samples to achieve a near-optimal policy. This is especially challenging when samples are expensive.

Many previous state-of-the-art methods rely on cost functions that either result in behavior that is too greedy—i.e. the policy simply tries to maximize returns during exploration—or too exploratory, i.e. the policy is incentivized to explore the environment dynamics and

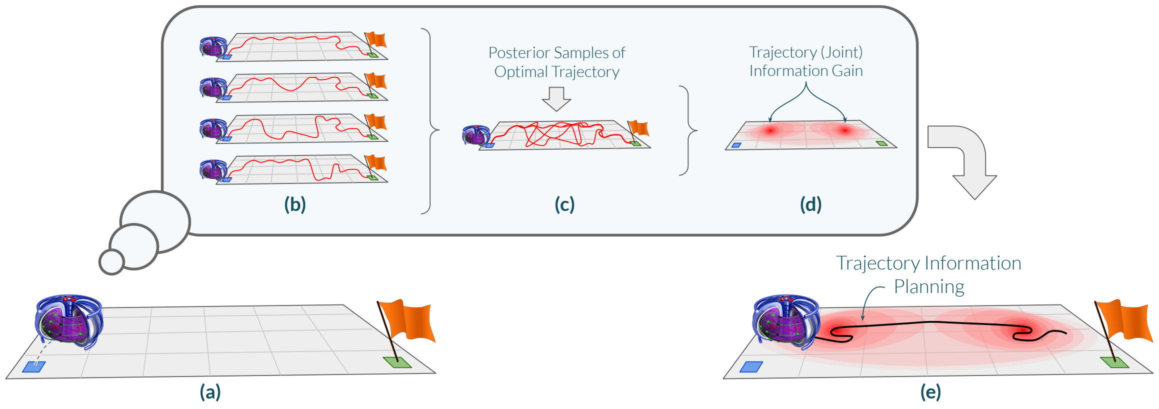
Posterior Samples of Optimal Trajectory
Trajectory (Joint) Information Gain
Trajectory Information Planning
(a)
(b)
(c)
(d)
(e)

Figure 1: A schematic depiction of Trajectory Information Planning (TIP). Suppose the agent in (a) aims to determine where to explore next from its current state. To do so, in (b) the agent samples dynamics models $T' \sim P(T \mid D)$ from its current posterior and finds approximately optimal trajectories $\tau^* \sim P(\tau^* \mid T')$ for each sample. Then in (c) it pools these samples of posterior optimal trajectories τ^*. In (d) it constructs a function that gives the joint expected information gain about the optimal trajectory τ^* given a planned exploration trajectory (i.e. EIG_{τ^*} over the set of points visited). Finally, in (e) the agent can plan an action sequence which maximizes this joint expected information gain.

does not consider the task at hand. We therefore present a cost function that balances out these two extremes. In particular, our cost function captures the amount of information that would be gained about the *optimal trajectory*, if the agent were to explore by following a particular planned trajectory. As depicted in Figure 1, this involves the agent planning out what it would hypothetically do given different realizations of the dynamics and planning actions that are informative about those possibilities. We give a discussion of the related work in Section E.

The contributions of this work are as follows: we develop a novel cost function for exploration that explicitly accounts for both the specific task as well as uncertainty about environment dynamics given past experience, a method for planning which applies the cost function to explore in MDPs with continuous states and actions, and a thorough empirical evaluation of our method across 5 closed-loop and 3 open-loop environments (with a focus on expensive RL tasks in plasma physics) compared against 14 baselines. We find that our proposed method is able to learn policies that perform as well as an agent with access to the ground truth dynamics using 2-200x fewer samples than comparison methods.

2. Problem Setting

In this work we deal with finite-horizon discrete-time *Markov decision processes* (MDPs) which consist of a sextuple $\langle \mathcal{S}, \mathcal{A}, T, r, p_0, H\rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the transition function $T : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$ (using the convention that $P(\mathcal{X})$ is the set of probability measures over $\mathcal{X}$), $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function, $p_0(s)$ is a distribution over $\mathcal{S}$ of start states, and $H \in \mathbb{N}$ is the horizon. We always assume $\mathcal{S}, \mathcal{A}, p_0$, and H are known. We also assume the reward r is known, though our development of the method can easily be generalized to the case where r is unknown. Our primary function of interest is the transition function T, which we learn from data. We address both open and closed loop control settings. In the more common closed loop setting, our aim is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the MDP objective below. We discuss trajectories $\tau \sim p(\tau \mid \pi, T)$ where $\tau = [(s_0, a_0), \ldots, (s_{H-1}, a_{H-1}), s_H]$ generated by $s_0 \sim p_0$, $a_i = \pi(s_i)$, and $s_{i+1} \sim T(s_i, a_i)$. We can write the return of a trajectory as $R(\tau) = \sum_{i=0}^{H-1} r(s_i, a_i, s_{i+1})$ for the states and actions s_i, a_i that make up τ. The MDP objective can then be written as $J_T(\pi) = \mathbb{E}_{\tau \sim p(\tau \mid \pi, T)} [R(\tau)]$. We aim to maximize this objective while minimizing the number of samples from the ground truth transition function T that are required to reach good performance. We denote the optimal policy as $\pi^* = \arg\max_\pi J_T(\pi)$, which we can assume to be deterministic (Sutton and Barto, 1998) but not necessarily unique. We use τ^* to denote optimal trajectories, i.e. $\tau^* \sim p(\tau \mid \pi^*, T)$

3. Trajectory Information Planning

Our method consists of a generic framework for Bayesian or approximately Bayesian model-predictive control and a novel cost function for planning that allows us to explicitly plan to find the maximal amount of new information relevant to our task. The MPC algorithm is given in Algorithm 1 and we give further discussion in Section A.4. Many prior methods such as Chua et al. (2018), Deisenroth and Rasmussen (2011), and Shyam et al. (2019) approximate this framework while using the greedy cost function corresponding to the future negative expected rewards or the pure exploration cost function corresponding to future information about the dynamics. In Section 3.1, we derive our new cost function and describe how it is computed. The overall method is simply applying the planning framework with the new cost function.

Algorithm 1 Bayesian Model-Predictive Control with Cost Function C

```
Inputs: transition function episode query budget $b$, number of posterior function samples $k$, planning horizon $h$
Initialize $D \leftarrow \emptyset$.
for $i \in [1, \ldots, b]$ do
  Sample start state $s_0 \sim p_0$.
  for $t \in [0, \ldots, H-1]$ do
    Sample posterior functions $\{T'_\ell\}_{\ell=1}^k \sim P(T' \mid D)$.
    Approximately find $\arg\min_{a_0, \ldots, a_{h-1}} \sum_{\ell=1}^k \mathbb{E}_{\tau_\ell \sim p(\tau \mid T'_\ell, a_0, \ldots, a_{h-1})} [C(\tau_\ell)]$ via CEM.
    Execute action $a_0$ by sampling $s_{t+1} \sim T(s_t, a_0)$.
    Update dataset $D \leftarrow D \cup \{(s_t, a_0, s_{t+1}\}$.
  end for
end for
return $\pi_g$ for the posterior $P(T' \mid D)$.
```

3.1 A Task-Specific Cost Function based on Trajectory Information

In this work, we aim to explore by choosing actions that maximize the conditional expected information gain (EIG) about the optimal trajectory τ^*. This is the same overall goal as that of Mehta et al. (2022), where the EIG_{τ^*} acquisition function was introduced for this purpose. However, in this paper we generalize this acquisition function in order to allow for sequential information collection that accounts for the redundant information that could be collected between timesteps. As discussed at length in Osband et al. (2019), it is essential to reason about how an action taken at the current timestep will affect the possibility of learning something useful in future timesteps. In other words, exploration must be *deep* and not greedy. Explicit examples are given in Osband et al. (2019) where the time to find an ϵ-optimal policy in a tabular MDP is exponential in the state size unless exploration can be coordinated over large numbers of timesteps rather than being conducted independently at each action. As the EIG_{τ^*} acquisition function is only defined over a single state-action pair and mutual information is submodular, we cannot naively use the acquisition function as is (or sum it over many datapoints) to choose actions that lead to good long-term exploration. As an illustrative example, this is clear in navigation tasks, where the nearby points visited over trajectories will provide redundant information about the local environment.

We therefore give a cost function that generalizes EIG_{τ^*} by taking a set of points to query and computing the *joint* expected information gain from observing the set. Our cost function is non-Markovian in the state space of the MDP, but it is Markovian in the dataset, which represents a point in the belief space of the agent about the dynamics. Let $\tilde{\mathcal{S}} = \{x : x \subseteq \mathcal{S} \times \mathcal{A}, |x| < \infty\}$, the set of finite subsets of the set of all state-action pairs. Our cost function $C_{\tau^*} : \tilde{\mathcal{S}} \to \mathbb{R}$ is defined below to be the negative *joint expected information gain* about the optimal trajectory τ^* for a subset $S \in \tilde{\mathcal{S}}$. In particular, assuming an existing dataset D, a set of h query points $S = \{(s_i, a_i)\}_{i\in[h]}$, and a random set of next states $S' = \{s'_i \sim T(s_i, a_i), i \in [h]\}$,

$$C_{\tau^*}(S) = \mathbb{E}_{S'\sim p(S'|S,D)}\left[\mathbb{H}\left[\tau^* \mid D \cup S'\right]\right] - \mathbb{H}\left[\tau^* \mid D\right]. \tag{1}$$

This formulation of C_{τ^*} forces our method to handle the redundant information among queries—it is likely that there will be some information about τ^* that is duplicated between members of the set and our method of computing this cost function must avoid 'double-counting' it. However, as written, this function relies on computing entropies on high-dimensional trajectories where the form of the joint distribution of the elements is unknown. To tractably estimate this quantity, we use the fact that $C_{\tau^*}(S) = -I(S', \tau^*) = -I(\tau^*, S')$ for the mutual information I. This allows us to exchange τ^* and our set of queries so that τ^* is giving information about the posterior predictive distribution of our set. In other words,

$$C_{\tau^*}(S) = \mathbb{E}_{\tau^*\sim p(\tau^*|D)}\left[\mathbb{H}\left[S' \mid D \cup \tau^*\right]\right] - \mathbb{H}\left[S' \mid D\right]. \tag{2}$$

In order to compute the right-hand term, we must take samples $\tau^*_{ij} \sim P(\tau^* \mid D), i = 1, \ldots, m, j = 1, \ldots, n$. To do this, we first sample m start states $s_0^{(i)}$ from p_0 (we always set $m = 1$ in experiments but derive the procedure in general) and for each start state independently sample n posterior functions $T'_{ij} \sim P(T' \mid D)$ from our posterior over dynamics models. We then run a planning procedure using iCEM (Pinneri et al., 2020) on each of the posterior functions from $s_0^{(i)}$ using T'_{ij} for T (using our assumption that planning can generate approximately optimal trajectories given ground-truth dynamics), giving our sampled τ^*_{ij}. Formally, we can approximate C_{τ^*} via Monte-Carlo as

$$C_{\tau^*}(S) \approx \frac{1}{mn}\left(\sum_{i\in[m]}\sum_{j\in[n]} \mathbb{H}[S'|D \cup \tau^*_{ij}]\right) - \mathbb{H}[S' \mid D]. \tag{3}$$

Assuming the dynamics are modelled with a Gaussian process, we can compute the joint Gaussian probability of the next states S' (Rasmussen, 2003). As the entropy of a multivariate Gaussian depends only on the log-determinant of the covariance, $\log|\Sigma|$, we can tractably compute the joint entropy of the model predictions $\mathbb{H}[S' \mid D]$ and optimize it with a zeroth order optimization algorithm. Finally, we must calculate the entropy $\mathbb{H}[S'|D \cup \tau^*_{ij}]$. For this, we follow a similar strategy as Neiswanger et al. (2021): since τ^*_i is a set of states output from the transition model, we can treat them as additional noiseless datapoints for our dynamics model and condition on them before computing the joint covariance matrix for S'. Given this newly generalized acquisition function, we can instantiate a method of planning in order to maximize future information gained. We give the concrete procedure for computing our acquisition function in Algorithm 2, noting that trajectories τ^*_{ij} do not depend on the query set S and can be cached for various values of S as long as the dataset D does not change.

Our ultimate procedure, which we name *Trajectory Information Planning* (TIP), is quite simple: run model-based RL using MPC as in Algorithm 1 but set the cost function to be $C_{\tau^*}(\tau)$ instead of C_g or C_e and compute this cost function using Algorithm 2. At test time, we simply return to planning with C_g as the cost function and greedily attempt to maximize returns over exploration. We can also formulate an open-loop variant of our method, oTIP, which simply involves planning once and then executing the entire action sequence.

Algorithm 2 Computation of C_{τ^*}

```
  Inputs: dataset $D = \{(s_k, a_k, s'_k)\}$, query set $S$, number of start state samples $m$, number of posterior function samples $n$.
  Sample $m$ start states $\{s_0^{(i)}\}_{i=1}^m \sim p_0$.
  for $i \in [m]$ do
      Sample $n$ posterior functions $\{T'_j\}_{j=1}^n \sim P(T' \mid D)$.
      for $j \in [n]$ do
          Set $\pi^*_j \leftarrow \pi_{\text{MPC}}$ using $C_g$ and singleton posterior $P(T \mid D) = \delta(T'_j)$ via Algorithm 1.
          Compute $\tau^*_{ij}$ by executing $\pi^*_j$ on $T'_j$ starting from $s_0^{(i)}$.
      end for
  end for
  Compute joint posterior covariance $\Sigma^{S'} \mid D$ across all points in $S$.
  Compute joint posterior covariances $\Sigma^{S'}_{ij} \mid D \cup \tau_{ij}\ \forall i \in [n], j \in [m]$ across all points in $S$.
  return $\log|\Sigma^{S'}| - \frac{1}{nm}\sum_{i\in[n],j\in[m]} \log|\Sigma^{S'}_{ij}|$.
```

4. Experiments

The aim of our development of the TIP algorithm and the C_{τ^*} acquisition function for RL is to reduce the sample complexity of learning effective policies in continuous MDPs given limited access to expensive dynamics. In this section we demonstrate the effectiveness of TIP in quickly learning a good policy by comparing against a variety of state-of-the-art reinforcement learning algorithms and strong baselines (including some that use the TQRL setting from (Mehta et al., 2022), which is also known as RL with a generative model in Kakade (2003) and other works (Azar et al., 2013; Agarwal et al., 2020)). In particular, we compare the average return across five evaluation episodes across five random seeds of each algorithm on 5 closed-loop control problems. We also assess the median amount of data taken by each algorithm to 'solve' the problem across five seeds, which we take to mean performing as well as an MPC controller using the ground truth dynamics.

Environment	TIP	sTIP	DIP	MPC	PETS	SAC	TD3	PPO	HUCRL	TS	BARL	EIG$_T$
Pendulum	**21**	36	36	46	5.6k	7k	26k	14k	>50k	>50k	**21**	56
Cartpole	131	141	161	201	1.63k	32k	18k	>1M	>6k	>6k	**111**	121
β Tracking	**46**	76	276	76	330	12k	17k	39k	480	420	186	>1k
β + Rotation	**201**	>500	>500	>500	400	30k	>50k	>50k	>5k	>5k	>500	>1k
Reacher	**251**	>400	>1k	751	700	23k	13k	>100k	6.6k	4.5k	**251**	>1.5k

Table 1: **Sample Complexity:** Median number of samples across 5 seeds required to reach 'solved' performance, averaged across 5 trials. We determine 'solved' performance by running an MPC policy (similar to the one used for evaluation) on the ground truth dynamics to predict actions. We record $> n$ when the median run is unable to solve the problem by the end of training after collecting n datapoints. The methods in the rightmost section operate in the TQRL setting and therefore have more flexible access to the MDP dynamics for data collection. The full set of methods are shown in Section D.

We evaluate the open-loop variant of our method, oTIP, against three comparison methods on three control problems we see as suitable for open-loop control. In particular, to be suitable for open-loop control, the problem cannot be dynamically unstable (as Pendulum and Cartpole famously are) and must have a relatively short control horizon and fixed start state. Here too, we assess the average return as open-loop trials are conducted as well as the number of timesteps required to achieve 'solved' performance.

We give a description of each of our comparison methods including the ablations and relationship to our TIP and oTIP methods in Section B.

We give full descriptions of our non-plasma control problems in Section C.

4.1 Plasma Control Problems

The plasma control problems (β Tracking and β + Rotation) are based on controlling a tokamak, a toroidally shaped device for confining a thermonuclear plasma using magnetic fields. Achieving net positive energy from fusion requires confining a plasma at high enough temperature and density long enough for hydrogen isotopes to collide and fuse. However, as the temperature and density are increased, a wide variety of instabilities can occur which degrade confinement, leading to a loss of energy. Full physics simulation of tokamak plasmas requires 10s-1000s of CPU hours to simulate a single trajectory, and often require hand tuning of different parameters to achieve accurate results. Following the work of Abbate et al. (2021), each of our plasma control problems used neural networks trained on data as the ground truth dynamics models. We used the MDSPlus tool (Stillerman et al., 1997) to fetch historical discharges from the DIII-D tokamak in San Diego. In total, we trained our models on 1,479 historical discharges. The data was pre-processed following the procedure outlined in Abbate et al. (2021). We describe how each environment was constructed in more detail in Sections C.1 and C.2.

Results As can be seen in Table 1, TIP is able to reach solved performance more quickly across the board than the model-based and model-free external baselines, often using a fraction or even orders of magnitude less data than other methods. We give further results and discussions in Section D.

Environment	oTIP	oMPC	oDIP	BO
Nonlinear Gain 1	**41**	91	51	210
Nonlinear Gain 2	**51**	61	>200	60
Lava Path	**41**	101	101	>2k

Table 2: **Open Loop Sample Complexity:** Same protocol as in Table 1 for open-loop actions.

References

Joseph Abbate, R Conlin, and E Kolemen. Data-driven profile prediction for diii-d. *Nuclear Fusion*, 61(4):046027, 2021.

Jan Achterhold and Joerg Stueckler. Explore the context: Optimal data collection for context-conditional dynamics models. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3529–3537. PMLR, 13–15 Apr 2021. URL `https://proceedings.mlr.press/v130/achterhold21a.html`.

Alekh Agarwal, Sham Kakade, and Lin F. Yang. Model-based reinforcement learning with a generative model is minimax optimal. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 67–83. PMLR, 09–12 Jul 2020. URL `https://proceedings.mlr.press/v125/agarwal20b.html`.

Jordan T. Ash, Cyril Zhang, Surbhi Goel, Akshay Krishnamurthy, and Sham M. Kakade. Anti-concentrated confidence bonuses for scalable exploration. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=RXQ-FPbQYVn`.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013.

Philip Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Ready policy one: World building through active learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 591–601. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/ball20a.html`.

A. Bondeson and D.J. Ward. Stabilization of external modes in tokamaks by resistive walls and plasma rotation. *Physical Review Letters*, 72(17):2709–2712, 1994.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1lJJnR5Ym`.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. UCB and infogain exploration via q-ensembles. *CoRR*, abs/1706.01502, 2017. URL `http://arxiv.org/abs/1706.01502`.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf`.

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. In *NeurIPS*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/a36b598abb934e4528412e5a2127b931-Abstract.html`.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. In *Aaai/iaai*, pages 761–768, 1998.

Richard Dearden, Nir Friedman, and David Andre. Model-based bayesian exploration. *CoRR*, abs/1301.6690, 1999. URL `http://arxiv.org/abs/1301.6690`.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 465–472, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/fujimoto18a.html`.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *CoRR*, abs/1609.04436, 2016. URL `http://arxiv.org/abs/1609.04436`.

R. J. Groebner, K. H. Burrell, and R. P. Seraydarian. Role of edge electric field and poloidal rotation in the l-h transition. *Physical Review Letters*, 64(25):3015–3018, 1990. ISSN 00319007. doi: 10.1103/PhysRevLett.64.3015.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1025–1033, Red Hook, NY, USA, 2012. Curran Associates Inc.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in neural information processing systems*, 27, 2014.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning.* University of London, University College London (United Kingdom), 2003.

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1701–1710. PMLR, 09–11 Apr 2018. URL `https://proceedings.mlr.press/v84/kamthe18a.html`.

J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th annual international conference on machine learning*, pages 513–520, 2009.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

David JC MacKay et al. Bayesian nonlinear modeling for the prediction competition. *ASHRAE transactions*, 100(2):1053–1062, 1994.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Willie Neiswanger, and Stefano Ermon. An experimental design perspective on model-based reinforcement learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=0no8Motr-zO`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Radford M Neal. *Bayesian Learning for Neural Networks.* PhD thesis, University of Toronto, 1995.

Willie Neiswanger, Ke Alexander Wang, and Stefano Ermon. Bayesian algorithm execution: Estimating computable properties of black-box functions using mutual information. In *International Conference on Machine Learning.* PMLR, 2021.

Nikolay Nikolov, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause. Information-directed exploration for deep reinforcement learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Byx83s09Km`.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf`.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL `http://jmlr.org/papers/v20/18-339.html`.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. *arXiv preprint arXiv:2008.06389*, 2020.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In *NIPS*, pages 1225–1232, 2007.

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/301ad0e3bd5cb1627a2044908a42fdc2-Paper.pdf`.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1): 1–96, 2018.

Ilya O Ryzhov and Warren B Powell. Information collection on a graph. *Operations Research*, 59(1):188–201, 2011.

Ilya O Ryzhov, Martijn RK Mes, Warren B Powell, and Gerald van den Berg. Bayesian exploration for approximate dynamic programming. *Operations research*, 67(1):198–214, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5779–5788. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/shyam19a.html`.

JA Stillerman, TW Fredian, KA Klare, and G Manduchi. Mdsplus data acquisition system. *Review of Scientific Instruments*, 68(1):939–942, 1997.

Malcolm Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0-262-19398-1. URL `http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html`.

Matthew Tesch, Jeff Schneider, and Howie Choset. Expensive function optimization with stochastic binary outcomes. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1283–1291, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `https://proceedings.mlr.press/v28/tesch13.html`.

Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. 1996.

James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pages 10292–10302. PMLR, 2020.

Control Problem	Pendulum	Cartpole	β Tracking	β + Rotation	Reacher
Sample τ^* mn times	24	31	7	25	130
Plan actions that minimize C_{τ^*}	16	15	15	50	295
Total for TIP Iteration	40	46	22	75	425
Evaluation for one episode	5-20	2-10	2-5	3 - 18	100-500

Table 3: Runtime in seconds for the phases of the TIP algorithm on all problems when run on the authors' CPU machines. The ranges given show the runtime for the operation at the beginning and at the end of training, as some operations run longer as more data is added.

Appendix A. Implementation Details

A.1 Wall Times

Though TIP and oTIP are designed for applications where samples are expensive and computation is relatively inexpensive, we present in this section data on the running time of these methods. We ran all experiments on a shared research cluster available to us on large machines with hundreds of GB of memory and between 24 and 88 CPU cores. In general our implementation did not make use of more than 20 CPU cores concurrently. In Table 3, we give the running time of the phases of the TIP algorithm. We note that the bulk of the computation in the planning procedure actually goes towards the just-in-time compilation of the JAX code that computes the cost function C_{τ^*} on sampled future trajectories. In order to allow for this compilation cost, we modified the iCEM algorithm from (Pinneri et al., 2020) to take fixed batch sizes as the compilation (e.g. for the β tracking problem) takes approximately 90% of the time required for planning. Unfortunately this compilation process must be repeated at every iteration due to the limitations of the JAX compiler. We believe that a similarly JIT-compiled implementation of the planning algorithm for sampling τ^* on posterior samples could lead to a substantial speedup and a more flexible compiler could do more still.

A.2 GP Model Details

For all of our experiments, we use a squared exponential kernel with automatic relevance determination (MacKay et al., 1994; Neal, 1995). The parameters of the kernel were estimated by maximizing the likelihood of the parameters after marginalizing over the posterior GP (Williams and Rasmussen, 1996).

To optimize the transition function, we simply sampled a set of points from the domain, evaluated the acquisition function, and chose the maximum of the set. This set was chosed uniformly for every problem but β + Rotation and Reacher, for which we chose a random subset of $\cup_i \cup_j \tau^*_{ij}$ (the posterior samples of the optimal trajectory) since the space of samples is 10-dimensional and uniform random sampling will not get good coverage of interesting regions of the state space.

A.3 Cost Function Details

We set $n = 15$ and $m = 1$ for our Monte Carlo estimate of the cost function for each problem.

A.4 Model-Predictive Control in Bayesian Model-Based RL

In this section, we give a formulation of Bayesian planning for control that generalizes ideas from methods such as PILCO (Deisenroth and Rasmussen, 2011) and PETS (Chua et al., 2018). This formulation highlights these methods' inherently greedy nature and hints at a possible solution. The objective of Bayesian planning is simply to find the h-step action sequence that maximizes the expected future returns under model uncertainty. That is,

$$\operatorname*{argmin}_{a_0,\dots,a_{h-1}} \mathbb{E}_{T'\sim P(T|D),\tau_e\sim P(\tau|s_0=s,a_{0:h-1},T')}\left[C(\tau_e)\right] \tag{4}$$

for some cost function C over trajectories and some start state s. If in the open-loop control setting, the agent simply executes the sequence of actions found. This procedure can also be extended to closed-loop control via model-predictive control (MPC). This procedure simply involves re-planning (4) at every state the agent visits and playing the first action from the optimal sequence. Concretely, the MPC policy for our Bayesian setting is as follows:

$$\pi_{\text{MPC}}(s) = \arg\min_{a_0} \min_{a_1,\dots,a_{h-1}} \mathbb{E}_{T'\sim P(T|D),\tau_e\sim P(\tau|s_0=s,a_{0:h-1},T')}\left[C(\tau_e)\right] \tag{5}$$

Whether we do open-loop control or closed-loop control via MPC, the cost function, C, is integral to how the agent will behave. Prior work has predominantly focused on two types of cost function:

$$\underbrace{C_g(\tau) = -R(\tau)}_{\text{Greedy Exploration}} \qquad \underbrace{C_e(\tau) = -\sum_{i=0}^{h} \mathbb{H}\left[T(s_i,a_i) \mid D\right]}_{\text{Task-Agnostic Exploration}} \tag{6}$$

Previous works such as Kamthe and Deisenroth (2018) and PETS (Chua et al., 2018) use the greedy exploration cost function, C_g. This cost function simply plans trajectories that achieve high rewards over the next h transitions on average. In works that focus on task-agnostic exploration such as Sekar et al. (2020) and Shyam et al. (2019), the cost function C_e (or similar) is used to encourage the agent to find areas of the state space in which the model is maximally uncertain.

The optimization problem in (5) is typically approximately solved in one of 3 ways: Deisenroth and Rasmussen (2011) and Curi et al. (2020) directly backpropagate through the estimated dynamics and reward functions to find policy parameters that would generate good actions, Janner et al. (2019) use an actor-critic method trained via rollouts in the model alongside the data collected to find a policy, and Chua et al. (2018) and Mehta et al. (2022) use the cross-entropy method (De Boer et al., 2005) to find action sequences which directly maximize the reward over the estimated dynamics. In this work we use a version of the last method given in Pinneri et al. (2020) to directly find action sequences that optimize the cost function being used. We approximate the expectation by playing the actions on multiple samples from the posterior $P(T \mid D)$. We give the hyperparameters used in planning in Section A.5.

Control Problem	Pendulum	Cartpole	β Tracking	β + Rotation	Reacher
Number of samples	25	30	25	50	100
Number of elites	3	6	3	8	15
Planning horizon	20	15	5	5	15
Number of iCEM iterations	3	5	3	5	5
Replanning Period	6	1	2	1	1

Table 4: Hyperparameters used for optimization in MPC procedure for closed-loop control problems.

Control Problem	Nonlinear Gain 1	Nonlinear Gain 2	Lava Path
Number of samples	50	50	25
Number of elites	6	6	4
Planning horizon	10	10	20
Number of iCEM iterations	6	8	6

Table 5: Hyperparameters used for optimization in MPC procedure for open-loop control problems.

A.5 Details on Planning Method

As mentioned in the main text, we use the iCEM method from Pinneri et al. (2020) with one major modification: a fixed sample batch size. This is in order to take advantage of the JIT compilation features of JAX and avoid recompiling code for each new batch size.

In Tables 4 and 5, we present the hyperparameters used for the planning algorithm across each problem. The same hyperparameters were used for the TIP, MPC, EIG$_T$, DIP, sDIP, and sTIP methods. As recommended by the original paper, we use $\beta = 3$ for the scaling exponent of the power spectrum density of sampled noise for action sequences, $\gamma = 1.25$ for the exponential decay of population size, and $\xi = 0.3$ for the amount of caching.

Appendix B. Description of Comparison Methods

We compare against 14 different methods across open and closed-loop problems. Of these, 7 used the same model and planning algorithm (including hyperparameters) as TIP and oTIP. **DIP** and **oDIP** use the cost function $C(\tau) = -\mathbb{H}\left[T(S') \mid D\right]$ and **sDIP** (summed DIP) uses the cost function $C(\tau) = -\sum_{i=0}^{h} \mathbb{H}\left[T(s_i, a_i) \mid D\right]$. These are all pure exploration methods, but DIP and oDIP are more sophisticated in that they plan for future observations with a large amount of *joint* information as opposed to sTIP which sums the individual information expected at each timestep. oDIP is simply the open loop variant of DIP. **EIG$_T$** uses the same objective as sDIP but operates in the TQRL setting, querying points that approximately maximize the predictive entropy of the dynamics model. **BARL** similarly operates in the TQRL setting but uses the EIG$_{\tau^*}$ acquisition function from Mehta et al. (2022). We use the authors' implementation of that work for comparison. **MPC** uses C_g from (6) and plans to directly maximize expected rewards. This method can be seen as quite similar to Kamthe and Deisenroth (2018) and a close cousin of Deisenroth and Rasmussen (2011) in that it optimizes the same objective with a similar model. **oMPC** is simply the open loop variant of MPC.

Besides these methods which directly compare cost functions, we include 8 additional baselines from published work. **PETS** is a method given in Chua et al. (2018) which uses a similar cross-entropy based planner and a probabilistic ensemble of neural networks for an uncertainty-aware estimate of the dynamics. PETS also plans to minimize C_g. **HUCRL** (Curi et al., 2020) learns a policy via backpropagation through time using a hallucinated perturbation to the dynamics that maximizes discounted rewards subject to the one-step confidence interval of the dynamics. HUCRL also uses a probabilistic ensemble of neural networks. Using the same implementation we also tested Thompson Sampling (**TS**), which acts optimally according to a network drawn from the posterior over models, and **BPTT** which plans to minimize C_g using a neural network policy and backpropagation through time. BPTT can also be viewed as a cousin of PILCO (Deisenroth and Rasmussen, 2011) as it attempts to take stochastic gradients of the expected cost. We also compare against **SAC** (Haarnoja et al., 2018), **TD3** (Fujimoto et al., 2018), and **PPO** (Schulman et al., 2017). SAC uses entropy bonuses to approximate Boltzmann exploration in an actor-critic framework. TD3 and PPO include various tricks for stable learning and add Ornstein-Uhlenbeck noise in order to explore.

Appendix C. Description of Control Problems

C.1 β Tracking

In this environment the goal is to adjust the total injected power (PINJ) of the neutral beams so that the normalized plasma pressure, β_N (defined as the ratio of thermal energy in the plasma to energy in the confining magnetic fields), reaches a target value of 2%. Reliably controlling plasmas to sustain high performance is a major goal of research efforts for fusion energy, so even this simple scenario is of interest. The ground-truth dynamics model takes in the current β_N and PINJ, the β_N and PINJ at some Δt time in the past, and the PINJ at some Δt time in the future (we assume that we have complete control over the values of PINJ at all times). Given these inputs, the model was trained to output what β_N will be Δt time into the future. In total, the state space is 4D and the action space is 1D. For this environment, we set $\Delta t = 200ms$, and we specify the reward function to be the negative absolute difference between the next β_N and the target $\beta_N = 2\%$.

C.2 β + Rotation Tracking

This environment is a more complicated version of the β tracking environment in several ways. First of all, the controller now must simultaneously track both β_N and the core toroidal rotation of the plasma. To do so, the controller is also allowed to set the total torque injected (TINJ) of the neutral beams (DIII-D has eight neutral beam injectors at different positions around the tokamak, so it is generally possible to control both total power and total torque independently). Controlling both of these quantities simultaneously is of interest since rotation shear often results in better confinement and less chance of instabilities in the plasma (Bondeson and Ward, 1994; Groebner et al., 1990). In addition, we assume a multi-task setting where the requested targets for β_N and rotation can be set every trajectory. Specifically, the β_N target is drawn from $U(1.5\%, 2.5\%)$ and the rotation target is drawn from $U(25, 125)$ krad/s every trajectory. These targets are appended to the state space.

The learned, ground-truth dynamics model is also more sophisticated here. In addition to the inputs and outputs used by the β tracking environment model, the inputs for this model also include rotation and TINJ at times t, $t - \Delta t$, and $t + \Delta t$ for TINJ only. This model receives additional information about the plasma (e.g. the shape of the plasma); however, we have assumed these inputs are fixed to reasonable values in order to avoid partial observability problems. In total, the state space of this problem is 10D (targets plus current and past observations for β_N, rotation, PINJ, and TINJ) and the action space is 2D (next PINJ and TINJ settings).

C.3 Robotics Problems

Pendulum The pendulum swing-up problem is the standard one found in the OpenAI gym (Brockman et al., 2016). The state space contains the angle of the pendulum and its first derivative and action space simply the scalar torque applied by the motor on the pendulum. The challenge in this problem is that the motor doesn't have enough torque to simply rotate the pendulum up from all positions and often requires a back-and-forth swing to achieve a vertically balanced position. The reward function here penalizes deviation from an upright pole and squared torque.

Cartpole The cartpole swing-up problem has 4-dimensional state (position of the cart and its velocity, angle of the pole and its angular velocity) and a 1-dimensional action (horizontal force applied to the cart). Here, the difficulty lies in translating the horizontal motion of the cart into effective torque on the pole. The reward function is a negative sigmoid function penalizing the distance betweent the tip of the pole and a centered upright goal position.

Reacher The reacher problem simulates a 2-DOF robot arm aiming to move the end effector to a randomly resampled target provided. The problem requires joint angles and velocities as well as an indication of the direction of the goal, giving an 8-dimensional state space along with the 2-dimensional control space.

Appendix D. Additional Results

Due to space constraints in the main paper, we omitted results for the methods sDIP and BPTT. The are included alongside the rest in Table 6. They are outperformed across the board by TIP. For many of our ablation methods we see failures to solve some of the problems even though the model is demonstrated by TIP to be able to sufficiently predict the dynamics. This is especially apparent on the harder plasma control environment, β+Rotation, where TIP is the only method using our GP which is able to solve the problem. We believe that this is because the data acquired through exploration by the ablation methods is less useful for control than the data TIP collects. This is underscored by the second column of Figure 2, where it is clear that TIP achieves the lowest modeling error on the points actu-

Environment	TIP	sTIP	DIP	sDIP	MPC	PETS	SAC	TD3	PPO	HUCRL	TS	BPTT	BARL	EIG$_T$
Pendulum	**21**	36	36	46	46	5.6k	7k	26k	14k	>50k	>50k	>50k	**21**	56
Cartpole	131	141	161	141	201	1.63k	32k	18k	>1M	>6k	>6k	>6k	**111**	121
β Tracking	**46**	76	276	131	76	330	12k	17k	39k	480	420	450	186	>1k
β + Rotation	**201**	>500	>500	>500	>500	400	30k	>50k	>50k	>5k	>5k	>5k	>500	>1k
Reacher	**251**	>400	>1k	>1k	751	700	23k	13k	>100k	6.6k	4.5k	3.7k	**251**	>1.5k

Table 6: **Sample Complexity Comparison of All Methods:** Median number of samples across 5 seeds required to reach 'solved' performance, averaged across 5 trials. We determine 'solved' performance by running an MPC policy (similar to the one used for evaluation) on the ground truth dynamics to predict actions. We record $> n$ when the median run is unable to solve the problem by the end of training after collecting n datapoints. The methods in the rightmost section operate in the TQRL setting and therefore have more flexible access to the MDP dynamics for data collection.

ally needed during the execution of the policy but not on the uniform test set. In particular we find it interesting that TIP outperforms BARL on the β + Rotation environment, as BARL should in principle have a strictly stronger access to the problem and is optimizing the same quantity with fewer constraints. We hypothesis that this may be due to the fact that BARL optimizes the acquisition function EIG$_{\tau^*}$ by simply uniformly sampling a set of points and choosing the one that evaluates to the largest value. Our more sophisticated optimization algorithm and forced initialization at the start state distribution seems to allow us to collect more information in this case. This interpretation is bolstered by the fact that on the problems where TIP outperforms BARL, we see that TIP is actually collecting more information per action than BARL as evidenced by larger EIG values. We also see clearly that there is value in computing the C_{τ^*} function rather than summing over EIG$_{\tau^*}$ values, as TIP outperforms sTIP across the board. Additionally, there is clear evidence for the value of task-specific exploration as the task-agnostic exploration methods (EIG$_T$, DIP, sDIP) underperform both in terms of returns and model error on the trajectories visited.

For the open-loop experiments (Table 2), we see similarly strong performance from oTIP. As the model-based methods benefit hugely from the fact that they observe many model transitions for each open-loop trial, it is unsurprising that they are more sample-efficient than the model-free BO method. Within the model-based techniques, oTIP is the most sample efficient. We believe that this is for much the same reasons as in the closed-loop case—exciting evidence that the C_{τ^*} cost function can be applied in a variety of settings.

Appendix E. Related Work

E.1 Bayesian Experimental Design: BOED, BO, BAX, and BARL

There is a large literature on Bayesian optimal experiment design (BOED) (Chaloner and Verdinelli, 1995) which focuses on efficiently querying a process or function to get maximal information about some quantity of interest. When the quantity of interest is the location of a function optimum, related strategies have been proposed as the predictive entropy search family of Bayesian optimization (BO) algorithms (Hennig and Schuler, 2012; Hernández-Lobato et al., 2014). Recently, a flexible framework known as Bayesian algorithm execution (BAX) (Neiswanger et al., 2021) has been proposed to efficiently estimate properties of

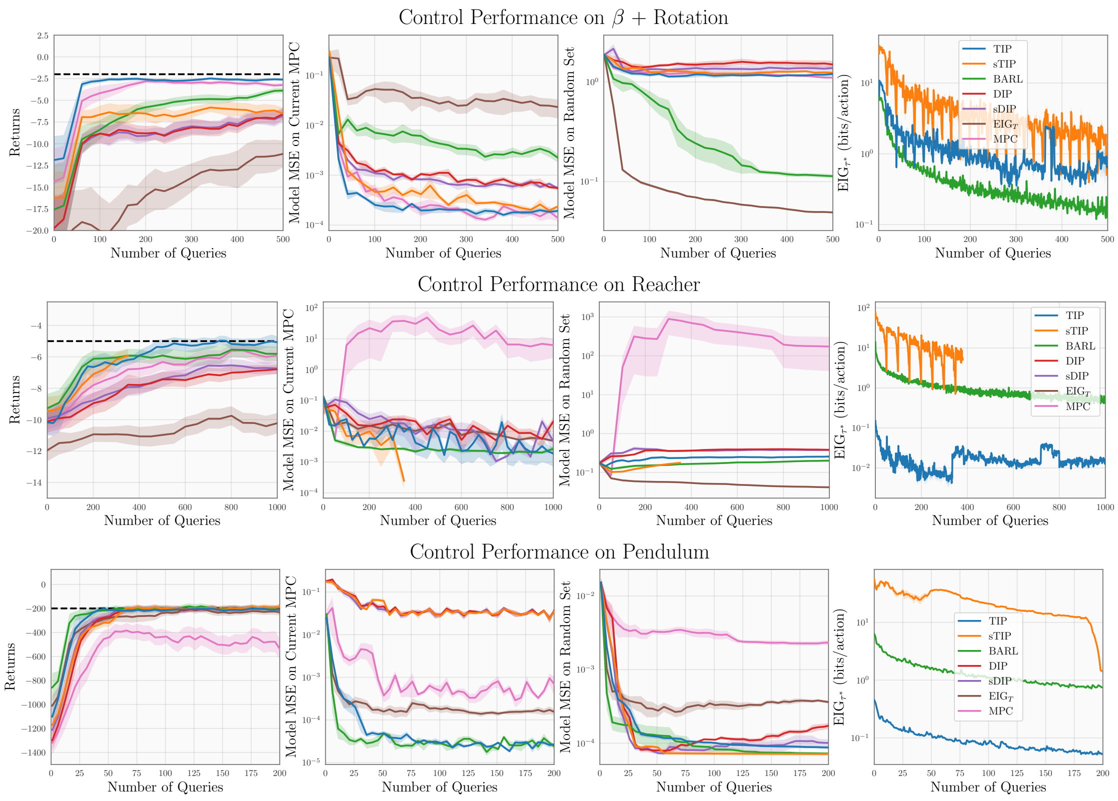

Figure 2: **Control and Modeling Details for TIP and Ablations.** Column 1: Learning curves for our ablation methods, all of which use the same planner and model. Column 2: Dynamics model accuracy on the points used by the planner to choose actions during MPC. Column 3: Dynamics model accuracy on a uniformly random test set in $\tilde{\mathcal{S}}$. Column 4: EIG_{τ^*} values normalized by the number of actions planned.

expensive black-box functions; this framework gives a general procedure for sampling points which are informative about the future execution of a given algorithm that computes the property of interest, thereby allowing the function property to be estimated with far less data.

A subsequent related work (Mehta et al., 2022), known as Bayesian Active Reinforcement Learning (BARL), uses ideas from BOED and BAX to sample points that are maximally informative about the optimal trajectory in an MDP. However, BARL relies on a setting the authors call Transition Query Reinforcement Learning (TQRL), which assumes that the environment dynamics can be iteratively queried at an arbitrary sequence of state-action pairs chosen by the agent. TQRL is thus a highly restrictive setting which is not suitable when data can only be accessed via a trajectory (rollout) of environment dynamics; it typically relies on an accurate environment simulator of sufficient expense to warrant its use. Even then, there will likely be differences between simulators and ground truth dynamics for complex systems. Thus, one would ideally like to collect data in real environments. However, this often requires leaving the TQRL setting, and instead collecting data via trajectories only.

In this paper, we aim to apply similar information-theoretic ideas, but extend them to the general MDP setting, as well as learning open loop model-based controllers. The typical method for learning to solve open-loop control problems was demonstrated successfully in Tesch et al. (2013), where a value function was learned from action sequences to task success. Our method takes a model-based approach to this problem, using similar exploration strategies to Bayesian optimization but benefitting from the more substantial supervision that is typical in dynamics model learning.

We give a discussion of methods for exploration in reinforcement as well as background on Bayesian exploration in RL and the use of Gaussian Processes in RL in section E.

E.2 Exploration in Reinforcement Learning

The most common strategy for exploration in RL is to execute a greedy policy with some form of added stochasticity. The simplest strategy, ϵ-greedy exploration as used in Mnih et al. (2013), simply takes the current action thought to be best with probability $1 - \epsilon$ and a random action with probability ϵ. Other methods use added Ornstein-Uhlenbeck action noise (Lillicrap et al., 2015) to the greedy policy, or entropy bonuses (Haarnoja et al., 2018) to the policy or value function objecties to add noise to a policy which is otherwise optimizing the RL objective.

Tabular RL is often solved by choosing actions based on upper confidence bounds on the value function (Chen et al., 2017; Lee et al., 2021), but explicitly computing and optimizing these bounds in the continuous setting is substantially more challenging. Recent work (Curi et al., 2020) approximates this method by computing one-step confidence bounds on the dynamics and training a 'hallucinated' policy which chooses perturbations within these bound to maximize expected policy performance. Another recent work (Ash et al., 2022) uses anti-concentration inequalities to approximate upper confidence bounds in MDPs with discrete actions.

Thompson sampling (TS) (Russo et al., 2018), which samples a realization of the MDP from the posterior and acts optimally as if the realization was the true model, can be applied for exploration in a model-free manner as in (Osband et al., 2016) or in a model-based manner as in Strens (2000). As the posterior over MDP dynamics or value functions can be high-dimensional and difficult to represent, the performance of TS can be hindered by approximation errors using both Gaussian processes and ensembles of neural networks. Curi et al. (2020) recently investigated this and found that this was potentially due to an insufficiently expressive posterior over entire transition functions, implying that it may be quite difficult to solve tasks using sampled models. Similarly, the posterior over action-value functions in Osband et al. (2016) is only roughly approximated by training a bootstrapped ensemble of neural networks.

There is also a rich literature of Bayesian methods for exploration, which are typically computationally expensive and hard to use, though they have attractive theoretical properties. These methods build upon the fundamental idea of the Bayes-adaptive MDP (Ross et al., 2007), which we detail in Section E.3 alongside a discussion of this literature.

Additionally, a broad set of methods explore to learn about the environment without a task in mind. This line of work is characterized by Pathak et al. (2017), which synthesizes a task-agnostic reward function from model errors. Other techniques include MAX (Shyam et al., 2019), which optimizes the information gain about the environment dynamics, Random Network Distillation (Burda et al., 2019), which forces the agent to learn about a random neural network across the state space, and Plan2Explore (Sekar et al., 2020), which prospectively plans to find areas of novelty where the dynamics are uncertain.

E.3 Bayesian Exploration Techniques

Given unlimited computation and an accurate prior, solving the Bayes-adaptive MDP (Ross et al., 2007) gives an optimal tradeoff between exploration and exploitation by explicitly accounting for the updated beliefs that would result from future observations and planning to find actions that result in high rewards as quickly as can be managed given the current posterior. However, this is computationally expensive even in small finite MDPs and totally intractable in continuous settings. Kolter and Ng (2009) and Guez et al. (2012) show that even approximating these techniques can result in substantial theoretical reductions in sample complexity compared to frequentist PAC-MDP bounds as in Kakade (2003). Another line of work (Dearden et al., 1998, 1999) uses the myopic value of perfect information as a heuristic for similar Bayesian exploration in the tabular MDP setting. Further techniques for exploration include knowledge gradient policies (Ryzhov et al., 2019; Ryzhov and Powell, 2011), which approximate the value function of the Bayes-adaptive MDP and information-directed sampling (IDS) (Russo and Van Roy, 2014), which takes actions based on minimizing the ratio between squared regret and information gain over dynamics. This was extended to continuous-state finite-action settings using neural networks in Nikolov et al. (2019). Another very relevant recent paper (Ball et al., 2020) gives an acquisition strategy in policy space that iteratively trains a data-collection policy in the model that trades off exploration against exploitation using methods from active learning. Achterhold and Stueckler (2021) use techniques from BOED to efficiently calibrate a Neural Process representation of a distribution of dynamics to a particular instance, but this calibration doesn't include information about the task. A tutorial on Bayesian RL methods can be found in Ghavamzadeh et al. (2016) for further reference.

E.4 Gaussian Processes (GPs) in Reinforcement Learning

There has been substantial prior work using GPs (Rasmussen, 2003) in reinforcement learning. Most well-known is PILCO (Deisenroth and Rasmussen, 2011), which computes approximate analytic gradients of policy parameters through the GP dynamics model while accounting for uncertainty. The original work is able to propagate the first 2 moments of the occupancy distribution through time using the GP dynamics and backpropagate gradients of the rewards to policy parameters. In Wilson et al. (2020), a method is developed for efficiently sampling functions from a GP posterior with high accuracy. One application show in their work is a method of using these samples to backpropagate gradients of rewards through time to policy paramters, which can be interpreted as a different sort of PILCO implementation. Most related to our eventual MPC-based method is (Kamthe and Deisenroth, 2018), which gives a principled probabilistic model-predictive control algorithm for GPs. We combine ideas from this paper, PETS (Chua et al., 2018), and the ability to sample posterior functions discussed above to give our eventual MPC component as discussed in Section A.4.