Mode-constrained Model-based Reinforcement Learning via Gaussian Processes

Aidan Scannell
Aalto University

Carl Henrik Ek
University of Cambridge

Arthur Richards
University of Bristol

Abstract

Model-based reinforcement learning (RL) algorithms do not typically consider environments with multiple dynamic modes, where it is beneficial to avoid inoperable or undesirable modes. We present a model-based RL algorithm that constrains training to a single dynamic mode with high probability. This is a difficult problem because the mode constraint is a hidden variable associated with the environment's dynamics. As such, it is 1) unknown a priori and 2) we do not observe its output from the environment, so cannot learn it with supervised learning. We present a nonparametric dynamic model which learns the mode constraint alongside the dynamic modes. Importantly, it learns latent structure that our planning scheme leverages to 1) enforce the mode constraint with high probability, and 2) escape local optima induced by the mode constraint. We validate our method by showing that it can solve a simulated quadcopter navigation task whilst providing a level of constraint satisfaction both during and after training.

1 INTRODUCTION

Over the last decade, reinforcement learning (RL) has become a popular paradigm for controlling dynamical systems (Hewing et al., 2020; Sutton and Barto, 2018). However, RL algorithms do not typically prevent agents from entering *inoperable* or *undesirable* dynamic modes (Def. 3.1). This would be desirable when flying a quadcopter to a target state whilst avoiding turbulent dynamic modes, or driving a car whilst avoiding dangerous road surfaces. In these examples, we seek agents that can learn sample-efficiently, whilst avoiding these *inoperable* or *undesirable* dynamic modes.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

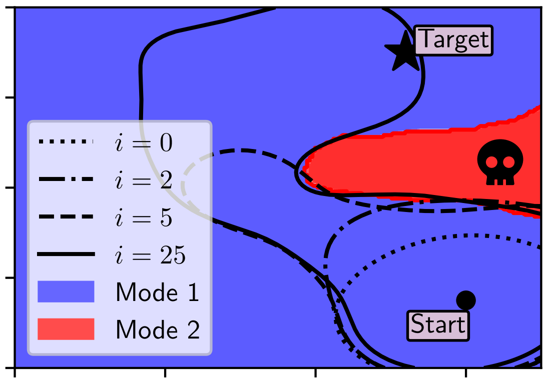

Figure 1: **Mode-constrained quadcopter navigation**. Top-down view of a quadcopter subject to 1) an *operable* dynamic mode (blue) and 2) an *inoperable*, turbulent dynamic mode induced by a strong wind field (red). The goal is to navigate to the target whilst remaining in the *operable* dynamic mode (blue). We achieve this by gradually expanding the δ-mode-constrained region (Def. 3.3) at each episode i, i.e. improving our knowledge of the latent mode constraint, by training our dynamic model on new data $\mathcal{D}_{0:i}$.

One approach to solving this problem is to constrain the agent to a single dynamic mode during training. However, in this mode-constrained setting, the agent does not observe the constraint. Instead, the mode constraint is a hidden variable associated with the environment's dynamics, which are *unknown a priori*. As a result, our constraint must be simultaneously learned and enforced.

When simultaneously learning and enforcing a constraint in RL, it is impossible to guarantee constraint satisfaction. This emphasises the need to learn sample efficiently, as each interaction with the environment could result in a constraint violation. Further to this, constraints can prevent an agent from solving the main task (Roy et al., 2022). This is because they can introduce local optima that make the space of feasible policies hard to navigate.

In this paper, we present ModeRL[1], a Bayesian model-based RL algorithm, which simultaneously learns and enforces the mode constraint using well-calibrated uncer-

[1]Code @ https://github.com/aidanscannell/moderl/

tainty estimates from a learned dynamic model. Our main contributions are as follows:

1. A method for jointly inferring the mode constraint alongside the underlying dynamic modes.
2. A planning algorithm that leverages the dynamic model's well-calibrated uncertainty estimates to 1) enforce the mode constraint up to a given probability, and 2) combat local optima induced by the constraint.
3. We validate ModeRL in a simulated quadcopter navigation task.

2 RELATED WORK

To the best of our knowledge, there is no prior work addressing mode-constrained RL (see Eq. (4)). However, we can compare our method to related works, which we detail here.

Constrained Markov decision processes (CMDPs) A common paradigm when considering constraints in RL is to consider CMDPs (Altman, 1999; Wachi et al., 2018). In this setting, the agent must satisfy a set of constraints defined by additional cost functions, whose output is observed from the environment. In contrast, the output of the constraint function is not observed in mode-constrained RL, so we cannot learn it using supervised learning. Our work has similarities to Schreiter et al., 2015 as they use a Gaussian process (GP) classifier to identify safe and unsafe regions when learning dynamic models in an active learning setting. However, they also assume that they observe the output of their constraint function.

Safe model predictive control (MPC) Koller et al., 2018 and Hewing et al., 2020 consider safe MPC schemes which use GP dynamic models to certify the safety of actions. However, they do not consider environments with multimodal dynamics. Arcari et al., 2020 is the most similar work to our own, as they consider safe stochastic MPC in dynamical systems with multiple operating modes (synonymous to our dynamic mode in Def. 3.1). Our work differs in that we use nonparametric methods to jointly identify the mode constraint and the dynamic modes.

Mode remaining planning Scannell et al., 2021 present a mode remaining trajectory optimisation algorithm that uses a learned dynamic model. However, they do not enforce any constraints. Further to this, they assume access to the environment *a priori*. In contrast, we consider the model-based RL setting, where we enforce the mode constraint during exploration (i.e. data collection) as well.

Safety as stability In low-dimensional continuous-control problems, Berkenkamp et al., 2017 propose to encode safety as stability via a learned dynamic model. However, their method assumes that the environment's dynamics are Lipschitz continuous. Although we do not provide details in this paper, we believe our method could be used to remove this assumption by constraining exploration to a subset of the dynamics that are Lipschitz continuous.

In summary, the constraint function in mode-constrained RL is not only *unknown a priori*, but its output is also not observed. This is because it is a hidden variable associated with the environment's dynamics. As such, it cannot be learned with supervised learning. In the remainder of this paper, we present a model-based RL method that uses a nonparametric dynamic model to infer the mode constraint – as a latent variable – alongside the dynamic modes. Importantly, it simultaneously learns and enforces the mode constraint (with high probability) during training.

3 PROBLEM STATEMENT

We consider environments with states $\mathbf{s}_t \in \mathcal{S} \subseteq \mathbb{R}^{D_x}$, actions $\mathbf{a}_t \in \mathcal{A} \subseteq \mathbb{R}^{D_u}$ and *multimodal*, *stochastic* transition dynamics, given by

$$\mathbf{s}_{t+1} = f_k(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\epsilon}_{k,t}, \quad \text{if } \alpha(\mathbf{s}_t) = k, \tag{1}$$

where the discrete mode indicator function $\alpha : \mathcal{S} \to \{1, \dots, K\}$ indicates which of the K underlying dynamic modes $\{f_k : \mathcal{S}_k \times \mathcal{A} \to \mathcal{S}\}_{k=1}^K$ and associated i.i.d. noise models $\boldsymbol{\epsilon}_{k,t}$ governs the environment at a given time step t. We refer to the output of the mode indicator function as the mode indicator variable $\alpha_t = \alpha(\mathbf{s}_t) \in \{1, \dots, K\}$.

Definition 3.1 (dynamic mode) *Let $\alpha : \mathcal{S} \to \{0, \dots, K\}$ denote a function which partitions a dynamical system's f state space $\mathcal{S}$ into K pair-wise disjoint state domains $\mathcal{S}_k = \{\mathbf{s} \in \mathcal{S} \mid \alpha(\mathbf{s}) = k\}$. Given a dynamical system comprising of K functions, $f = \{f_k : \mathcal{S}_k \times \mathcal{A} \to \mathcal{S}\}_{k=1}^K$, where each function governs the system at a particular region of the state space $\mathcal{S}$, we formally define a dynamic mode as,*

$$f_k : \mathcal{S}_k \times \mathcal{A} \to \mathcal{S}. \tag{2}$$

Notice from Def. 3.1 that our dynamic modes are free to leave their state spaces $\mathcal{S}_k$ and enter other modes $\mathcal{S}$.

Problem statement We consider controlling the stochastic system in Eq. (1) in an episodic setting, over a horizon T. We assume that after each episode the system is reset to a known initial state $\mathbf{s}_0$. We consider general deterministic policies $\pi \in \Pi$, which encapsulates both closed-loop policies $\pi(\mathbf{s}_t)$ and open-loop policies $\pi(t)$. For a known transition dynamic model $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the performance of a policy π is the sum of rewards over the horizon, in expectation over the transition noise,

$$J(\pi, f) = \mathbb{E}_{\boldsymbol{\epsilon}_{0:T}} \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 \right]. \tag{3}$$

The goal of our work is to find the optimal policy π^* whilst remaining in a desired dynamic mode k^*,

$$\operatorname*{argmax}_{\pi \in \Pi} J(\pi, f) \quad \text{s.t. } \alpha(\mathbf{s}_t) = k^* \quad \forall t \in \{0, \dots, T\}. \tag{4}$$

Formally, a mode-constrained system is defined as follows.

Definition 3.2 (mode-constrained) *Let $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denote a multimodal dynamical system and $\mathcal{S}_{k^*} = \{\mathbf{s} \in \mathcal{S} \mid \alpha(\mathbf{s}) = k^*\}$ denote the state domain of the desired dynamic mode k^*. Given an initial state $\mathbf{s}_0 \in \mathcal{S}_{k^*}$ and a policy $\pi \in \Pi$, the controlled system is said to be mode-constrained under the policy π iff:*

$$f(\mathbf{s}_t, \pi(\mathbf{s}_t, t)) + \epsilon_t \in \mathcal{S}_{k^*} \quad \forall t \in \{0, \dots, T\} \tag{5}$$

Given that neither the underlying dynamic modes $\{f_k\}_{k=1}^K$, nor how the system switches between them α, are *known a priori*, it is not possible to solve Eq. (4) with the mode constraint in Def. 3.2. Therefore, we relax the requirement to finding a mode-constrained policy with high probability. We formally define a δ-mode-constrained policy as follows.

Definition 3.3 (δ-mode-constrained) *Let $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denote a multimodal dynamical system and $\mathcal{S}_{k^*} = \{\mathbf{s} \in \mathcal{S} \mid \alpha(\mathbf{s}) = k^*\}$ denote the state domain of the desired dynamic mode k^*. Given an initial state $\mathbf{s}_0 \in \mathcal{S}_{k^*}$ and $\delta \in (0, 1]$, a controlled system is said to be δ-mode-constrained under the policy π iff:*

$$\Pr(\forall t \in \{0, \dots, T\} : f(\mathbf{s}_t, \pi(\mathbf{s}_t, t)) + \epsilon_t \in \mathcal{S}_{k^*}) \geq 1 - \delta. \tag{6}$$

Policies satisfying this δ-mode-constrained definition should remain in the desired dynamic mode with probability up to $1-\delta$. It is worth noting that the agent would not be able to explore the environment without relaxing the mode constraint from Def. 3.2. Increasing δ promotes exploration but also increases the chance of violating the mode constraint. Intuitively, δ, which we refer to as the *constraint level*, makes the mode-constrained problem feasible, whilst still providing some level of constraint satisfaction during training.

Initial mode remaining controller In robotics applications, an initial set of poor-performing controllers can normally be obtained via simulation or domain knowledge. We assume access to an initial data set of state transitions $\mathcal{D}_0 = \{(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}_{t=1}^{TN_0}$ from N_0 episodes of length T. We use it to learn a predictive dynamic model $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t, \mathcal{D}_0)$ which is locally accurate around the start state $\mathbf{s}_0$.

Assumption 3.1 *A state transition data set has been collected $\mathcal{D}_0 = \{(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}_{t=1}^{TN_0}$ from an initial region of the state space $\mathcal{S}_0 \subseteq \mathcal{S}_{k^*}$, which belongs to the desired dynamic mode k^* and contains the start state $\mathbf{s}_0 \in \mathcal{S}_0$.*

Algorithm 1 ModeRL

```
Require: Start state $\mathbf{s}_0$, desired dynamic mode $k^*$, initial
    data set $\mathcal{D}_0$, policy $\pi_0$, dynamic model $p(\mathbf{s}_{t+1} \mid \hat{\mathbf{s}}_t, \mathcal{D}_0)$
1: for $i \in \{0, 1, \dots, \text{num episodes}\}$ do
2:     while not converged do
3:         Sample $N_b$ state transitions $\mathcal{B} \sim \mathcal{D}_{0:i}$
4:         Update dynamics using Eq. (12) with $\mathcal{B}$
5:     end while
6:     Optimise policy $\pi_{i+1}$ using Eq. (16)
7:     Collect data $\mathcal{D}_{i+1}$ using $\pi_{i+1}$
8:     Update agent's data set $\mathcal{D}_{0:i+1} = \mathcal{D}_{i+1} \cup \mathcal{D}_{0:i}$
9: end for
```

Although such a model can be used to learn an initial policy, it will not work outside of the initial state domain $\mathcal{S}_0$ and may not be able to find a δ-mode-constrained policy, due to the model having high *epistemic uncertainty*. For this reason, we adopt a model-based RL strategy which incrementally explores the environment subject to a δ-mode constraint. At each episode, it collects data and uses it to train its dynamic model. This reduces the dynamic model's *epistemic uncertainty* and expands the δ-mode-constrained region. See Fig. 1.

4 MODE-CONSTRAINED MODEL-BASED RL

We propose to solve the mode-constrained RL problem in Eq. (4) by synergising model learning and planning in a model-based RL algorithm we name mode-constrained model-based reinforcement learning (ModeRL). Our approach is detailed in Alg. 1.

4.1 Probabilistic Dynamic Model

ModeRL learns a single-step dynamic model and we adopt the delta state formulation to regularise the predictive distribution. We denote a state difference output as $\Delta\mathbf{s}_{t+1} = \mathbf{s}_{t+1} - \mathbf{s}_t$ and the set of all state difference outputs as $\Delta\mathbf{S}$. We further denote a state-action input as $\hat{\mathbf{s}}_t = (\mathbf{s}_t, \mathbf{a}_t)$, the set of all state-action inputs as $\hat{\mathbf{S}}$, the set of all state inputs as $\mathbf{S}$ and the state transition data set at episode i as $\mathcal{D}_{0:i}$.

The main goal of our dynamic model is to jointly infer the mode constraint along with the underlying dynamic modes. In particular, we would like our model to,

1. Formulate a prior over the mode constraint where we can encode prior knowledge, potentially enabling ModeRL to find a policy without ever violating the mode constraint. In practice, encoding prior knowledge allows us to exploit Bayesian interpolation (MacKay, 1992) to reduce the number of constraint violations during training.

2. Disentangle the sources of uncertainty in the mode constraint so that ModeRL can escape local optima induced by the constraint (via intrinsic exploration with the mode constraint's *epistemic uncertainty*).

Marginal likelihood Mixtures of Gaussian process experts (MoGPE) models are a natural choice for modelling multimodal systems as they automatically infer the assignment of observations to dynamic modes (experts). Let us start by introducing the MoGPE marginal likelihood,

$$p\left(\Delta\mathbf{S} \mid \hat{\mathbf{S}}\right) = \sum_{\boldsymbol{\alpha}} \underbrace{p\left(\boldsymbol{\alpha} \mid \mathbf{S}\right)}_{\text{gating network}} \left[\prod_{k=1}^{K} \underbrace{p\left(\Delta\mathbf{S}_k \mid \hat{\mathbf{S}}_k, \boldsymbol{\theta}_k\right)}_{\text{dynamic mode } k}\right] \tag{7}$$

where $\hat{\mathbf{S}}_k$ denotes the set of N_k inputs assigned to dynamic mode k, i.e. $\hat{\mathbf{S}}_k = \{\hat{\mathbf{s}}_t \in \hat{\mathbf{S}} \mid \alpha(\mathbf{s}_t) = k\}$. Similarly for the outputs we have $\Delta\mathbf{S}_k = \{\Delta\mathbf{s}_{t+1} \in \Delta\mathbf{S} \mid \alpha(\mathbf{s}_{t+1}) = k\}$. Note that there is a joint distribution corresponding to every possible combination of assignments of observations to dynamic modes. Hence, Eq. (7) is a sum over exponentially many (K^N) sets of assignments, where $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_N\}$ represents a set of assignments for all observations. This distribution factors into the product over modes, where each mode models the joint Gaussian distribution over the observations assigned to it.

Learning the mode constraint The gating network indicates which dynamic mode governs the system at a given state input. It is of particular importance in our work as we use it to represent our mode constraint. Motivated by synergising model learning and planning, we formulate our gating network using input-dependent functions – known as gating functions – and place GP priors over them. In Sec. 4.3 we exploit the disentangled *epistemic uncertainty* represented in our GP-based gating network to help ModeRL escape local optima induced by the mode constraint. Similar to Tresp, 2000, our gating network resembles a GP classification model,

$$p\left(\boldsymbol{\alpha} \mid \hat{\mathbf{S}}\right) = \mathbb{E}_{\underbrace{p(\mathbf{h}(\mathbf{S}))}_{\text{GP prior(s)}}}\left[\prod_{t=1}^{N} \underbrace{p\left(\alpha_t \mid \mathbf{h}(\mathbf{s}_t)\right)}_{\text{classification likelihood}}\right] \tag{8}$$

where $p\left(\alpha_t \mid \mathbf{h}(\mathbf{s}_t)\right)$ represents a classification likelihood parameterised by K gating functions $\mathbf{h} : \mathcal{S} \rightarrow \mathbb{R}^K$. We use a Bernoulli likelihood when $K = 2$ and a softmax when $K > 2$. We place GP priors on each of the gating functions $p(\mathbf{h}(\hat{\mathbf{S}})) = \prod_{k=1}^{K} \mathcal{N}\left(p(h_k(\hat{\mathbf{S}}) \mid \hat{\mu}_k(\hat{\mathbf{S}}), \hat{k}_k(\hat{\mathbf{S}}, \hat{\mathbf{S}})\right)$, where $\hat{\mu}_k(\cdot)$ and $\hat{k}_k(\cdot, \cdot)$ represent the mean and covariance functions associated with the k^{th} gating function h_k. In our gating network formulation, the GP posteriors represent the mode constraint's *epistemic uncertainty*.

Dynamic modes We model the underlying dynamic modes as independent GP regression models,

$$\underbrace{p\left(\Delta\mathbf{S}_k \mid \hat{\mathbf{S}}_k\right)}_{\text{dynamic mode } k} = \underbrace{\mathbb{E}_{p(f_k(\hat{\mathbf{S}}_k))}}_{\text{GP prior}}\left[\prod_{t=1}^{N_k} \underbrace{p\left(\Delta\mathbf{s}_{t+1} \mid f_k(\hat{\mathbf{s}}_t)\right)}_{\text{Gaussian likelihood}}\right],$$

where each mode's GP prior is given by $p(f_k(\hat{\mathbf{S}}_k)) = \mathcal{N}\left(f_k(\hat{\mathbf{S}}_k) \mid \mu_k(\hat{\mathbf{S}}_k), k_k(\hat{\mathbf{S}}_k, \hat{\mathbf{S}}_k)\right)$ with $\mu_k(\cdot)$ and $k_k(\cdot, \cdot)$ representing the mean and covariance functions associated with the k^{th} mode's GP prior respectively. Note that as the assignment of observations to modes is *not known a priori*, we must infer the assignments from observations. In our model, each dynamic mode's Gaussian likelihood represents the mode's *aleatoric uncertainty* (the transition noise in this case) whilst each mode's GP posterior represents the mode's *epistemic uncertainty*.

4.2 Dynamic Model Learning

Performing Bayesian inference in our dynamic model involves finding the posterior over the latent variables $p(\{f(\hat{\mathbf{S}})\}_{k=1}^{K}, \mathbf{h}(\mathbf{S}) \mid \mathcal{D}_{0:i})$, which requires calculating the marginal likelihood in Eq. (7). As such, exact inference in our model is intractable due to the marginalisation over the set of mode indicator variables. For this reason, we resort to a variational approximation.

Following the approach by Titsias, 2009, we augment the probability space with a set of inducing variables for each GP. However, instead of collapsing these inducing variables, we represent them as variational distributions and use them to lower bound the marginal likelihood, similar to Hensman et al., 2013, 2015. Fig. 6 shows the graphical model of the augmented joint probability space.

Augmented dynamic modes We sidestep the hard assignment of observations to modes by augmenting each dynamics GP with a set of M_{f_k} separate independent inducing points,

$$p(f_k(\boldsymbol{\zeta}_k)) = \mathcal{N}\left(f_k(\boldsymbol{\zeta}_k) \mid \mu_k(\boldsymbol{\zeta}_k), k_k(\boldsymbol{\zeta}_k, \boldsymbol{\zeta}_k)\right). \tag{9}$$

Introducing separate inducing points from each mode's GP can loosely be seen as "partitioning" the observations between modes. However, as the assignment of observations to modes is *not known a priori*, the inducing inputs $\boldsymbol{\zeta}_k$ and variables $f_k(\boldsymbol{\zeta}_k)$, must be inferred from observations.

Augmented gating network We follow a similar approach for the gating network and augment each gating function GP with a set of M_{h_k} inducing points,

$$p(h_k(\boldsymbol{\xi})) = \mathcal{N}\left(h_k(\boldsymbol{\xi}) \mid \hat{\mu}_k(\boldsymbol{\xi}), \hat{k}_k(\boldsymbol{\xi}, \boldsymbol{\xi})\right). \tag{10}$$

The distribution over all gating functions is denoted $p(\mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} p(h_k(\boldsymbol{\xi}))$. In contrast to the dynamic modes, the gating function GPs share inducing inputs $\boldsymbol{\xi}$.

Marginal likelihood We use these inducing points to approximate the true marginal likelihood with,

$$p\left(\Delta \mathbf{S} \mid \hat{\mathbf{S}}\right) \approx \mathbb{E}_{p(\mathbf{h}(\boldsymbol{\xi}))p(\mathbf{f}(\boldsymbol{\zeta}))}\Bigg[\prod_{t=1}^{N} \sum_{k=1}^{K} \Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi})) p(\Delta \mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k)) \Bigg], \tag{11}$$

where the conditional distributions $p(\Delta \mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k))$ and $\Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi}))$ follow from standard sparse GP methodologies. See Eqs. (19) and (20) in App. A.1. Importantly, the factorisation over observations is outside of the marginalisation over the mode indicator variable, i.e. the mode indicator variable can be marginalised for each data point separately. This is not usually the case for MoGPE methods. Our approximation assumes that the inducing variables, $\{f_k(\boldsymbol{\zeta}_k)\}_{k=1}^K$, are a sufficient statistic for their associated latent function values, $\{f_k(\hat{\mathbf{S}}_k)\}_{k=1}^K$ and the set of assignments $\boldsymbol{\alpha}$. It becomes exact when each mode's inducing points represent the true data partition $\{\boldsymbol{\zeta}_k, f_k(\boldsymbol{\zeta}_k)\}_{k=1}^K = \{\hat{\mathbf{S}}_k, f_k(\hat{\mathbf{S}}_k)\}_{k=1}^K$.

Evidence lower bound (ELBO) Following a similar approach to Hensman et al., 2013, 2015, we lower bound Eq. (11),

$$\begin{aligned}\mathcal{L}(\{\mathbf{m}_k, \mathbf{L}_k, \hat{\mathbf{m}}_k, \hat{\mathbf{L}}_k, \boldsymbol{\zeta}_k\}_{k=1}^K, \boldsymbol{\xi}) = \sum_{t=1}^{N} \mathbb{E}_{q(\mathbf{h}(\hat{\mathbf{s}}_t))q(\mathbf{f}(\boldsymbol{\zeta}))} \Bigg[\\ \log \sum_{k=1}^{K} \Pr\left(\alpha_t = k \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right) p(\Delta \mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k)) \Bigg] \\ - \sum_{k=1}^{K} \mathrm{KL}\left(q(f_k(\boldsymbol{\zeta}_k)) \mid\mid p(f_k(\boldsymbol{\zeta}_k))\right) \\ - \sum_{k=1}^{K} \mathrm{KL}\left(q(h_k(\boldsymbol{\xi})) \mid\mid p(h_k(\boldsymbol{\xi}))\right)\end{aligned} \tag{12}$$

where the dynamic mode's variational posterior is given by $q(\mathbf{f}(\boldsymbol{\zeta})) = \prod_{k=1}^K \mathcal{N}\left(f_k(\boldsymbol{\zeta}_k) \mid \mathbf{m}_k, \mathbf{L}_k \mathbf{L}_k^T\right)$ and the gating network's variational posterior is given by $q(\mathbf{h}(\hat{\mathbf{s}}_t)) = \prod_{k=1}^K \int p(h_k(\hat{\mathbf{s}}_t) \mid h_k(\boldsymbol{\xi})) \mathcal{N}\left(h_k(\boldsymbol{\xi}) \mid \hat{\mathbf{m}}_k, \hat{\mathbf{L}}_k \hat{\mathbf{L}}_k^T\right) \mathrm{d}\mathbf{h}(\boldsymbol{\xi})$.

Optimisation The bound in Eq. (12) induces a local factorisation over observations and has a set of global variables – the necessary conditions to perform stochastic variational inference (SVI) (Hoffman et al., 2013) on $q(\mathbf{f}(\boldsymbol{\zeta}))$ and $q(\mathbf{h}(\boldsymbol{\xi}))$, i.e. optimise $\{\mathbf{m}_k, \mathbf{L}_k, \hat{\mathbf{m}}_k, \hat{\mathbf{L}}_k\}_{k=1}^K$. We treat the inducing inputs $\boldsymbol{\xi}$, $\{\boldsymbol{\zeta}_k\}_{k=1}^K$, kernel hyperparameters and noise variances, as variational hyperparameters and optimise them alongside the variational parameters, using Adam (Kingma and Ba, 2017). We use mini-batches and approximate the expectations over the log-likelihood using Monte Carlo samples.

Our approach can loosely be viewed as parameterising the nonparametric model in Eq. (7) to obtain a desirable factorisation for 1) constructing a GP-based gating network and 2) deriving an ELBO that can be optimised with stochastic gradient methods. Importantly, our approach still captures the complex dependencies between the gating network and dynamic modes.

Predictions Given our variational approximation, we make predictions at a new input $\hat{\mathbf{s}}_t$ with,

$$p(f_k(\hat{\mathbf{s}}_t) \mid \hat{\mathbf{s}}_t, \mathcal{D}_{0:i}) \approx \int p(f_k(\hat{\mathbf{s}}_t) \mid f_k(\boldsymbol{\zeta}_k)) q(f_k(\boldsymbol{\zeta}_k)) \mathrm{d}f_k(\boldsymbol{\zeta}_k),$$
$$p(h_k(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_{0:i}) \approx \int p(h_k(\mathbf{s}_t) \mid h_k(\boldsymbol{\xi})) q(h_k(\boldsymbol{\xi})) \mathrm{d}h_k(\boldsymbol{\xi}).$$

See Eqs. (21) and (22) in App. A.1.

4.3 Planning

We now detail our planning algorithm which leverages the latent structure of our dynamic model to:

1. Enforce the mode constraint with high probability,
2. Escape local optima induced by the mode constraint, by targeting exploration where the agent has high *epistemic uncertainty* in its belief of the mode constraint.

Multi-step predictions with dynamic model Let us first observe that if the controlled system satisfies the mode constraint, the system will be fully governed by the desired dynamic mode f_{k^*}. We leverage this observation and simplify making multi-step predictions by using only the desired dynamic mode f_{k^*}. This enables us to approximate the GP dynamics integration in closed form using moment matching (Deisenroth and Rasmussen, 2011; Girard et al., 2003; Kamthe and Deisenroth, 2018). See App. A.3 for more details. This approximation has been shown to work well in RL contexts (Cutler and How, 2015; Deisenroth et al., 2015; Deisenroth and Rasmussen, 2011; Pan and Theodorou, 2014; Pan et al., 2015).

Objective Given this approach for making multi-step predictions, we use Def. 3.3 to formulate a relaxed version of the mode-constrained problem in Eq. (4),

$$\operatorname*{argmax}_{\pi \in \Pi} \underbrace{\mathbb{E}_{p(f_{k^*} \mid \mathcal{D}_{0:i})}\left[J(\pi, f_{k^*})\right]}_{\text{greedy exploitation}}, \tag{13a}$$
$$\text{s.t.} \underbrace{\Pr(\alpha_t = k^* \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \mathcal{D}_{0:i}) \geq 1 - \delta,}_{\delta\text{-mode constraint}} \quad \forall t. \tag{13b}$$

The expectation is taken over the desired dynamic mode's f_{k^*} GP. Note that the expected objective in Eq. (13a), which we refer to as greedy exploitation, is widely adopted. For example, in PILCO (Deisenroth and Rasmussen, 2011), PETS (Chua et al., 2018) and GP-MPC (Kamthe and Deisenroth, 2018). We calculate the δ-mode constraint

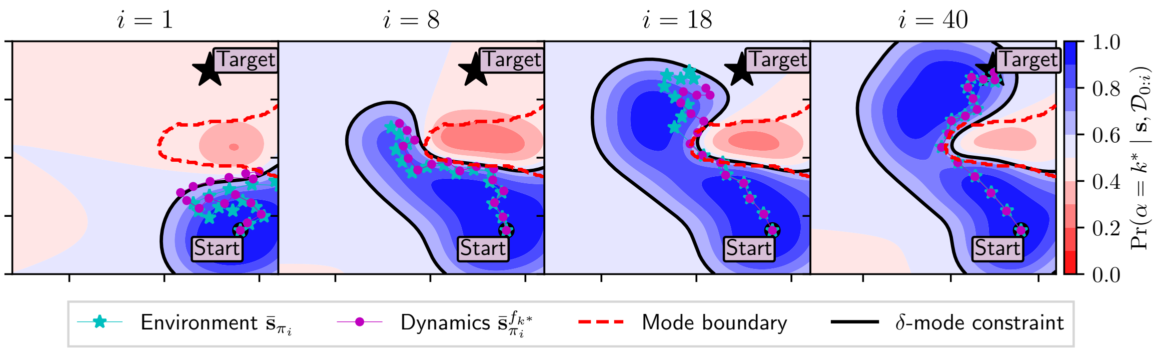

Figure 2: Visualisation of four episodes i of ModeRL in the quadcopter navigation task from Fig. 7. The goal is to navigate to the black star, whilst avoiding the turbulent dynamic mode (dashed red line). The contour plots indicate the agent's belief of being in the desired dynamic mode $\Pr(\alpha = k^* \mid \mathbf{s}, \mathcal{D}_{0:i})$ at each episode, i.e. after training on $\mathcal{D}_{0:i}$. The black lines show the δ-mode constraint (see Eq. (13b)) expanding during training. At each episode i we roll out the policy in the desired dynamic mode's GP (magenta) as well as the in the environment (cyan). Experiments used an exponentially decaying schedule on δ to tighten the mode constraint during training.

in Eq. (13b) as,

$$\Pr(\alpha_t = k^* \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \mathcal{D}_{0:i}) = \int \Pr(\alpha_t = k^* \mid h(\mathbf{s}_t)) \underbrace{p(h(\mathbf{s}_t) \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \mathcal{D}_{0:i})}_{\text{gating posterior at } t} \mathrm{d}h(\mathbf{s}_t), \tag{14}$$

where $\Pr(\alpha_t = k^* \mid h(\mathbf{s}_t))$ resembles a classification likelihood (see App. A.2). We approximate the gating function posterior predicted t steps into the future,

$$\underbrace{p(h(\mathbf{s}_t) \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \mathcal{D}_{0:i})}_{\text{gating posterior at } t} = \int \underbrace{p\left(h(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_{0:i}\right)}_{\text{gating posterior}} \underbrace{p(\mathbf{s}_t \mid \mathbf{s}_0, \mathbf{a}_{0:t-1}, \mathcal{D}_{0:i})}_{\text{dynamics posterior}} \mathrm{d}\mathbf{s}_t, \tag{15}$$

by propagating the state's uncertainty using moment matching (see App. A.3), where the state distribution $p(\mathbf{s}_t \mid \mathbf{s}_0, \mathbf{a}_{0:t-1}, \mathcal{D}_{0:i})$ is obtained by cascading single-step predictions using moment matching (see App. A.3). Importantly, the δ-mode constraint considers 1) both the *epistemic* and *aleatoric uncertainties* in the desired dynamic mode and 2) the *epistemic uncertainty* in the gating network. This ensures that the controlled system is δ-mode-constrained (Def. 3.3) under the uncertainty of the learned dynamic model. Intuitively, it will remain where the dynamic model's *uncertainty* is low because the expectation results in lower probabilities for more uncertain states.

Exploration In our experiments, the constraint in Eq. (13b) hinders exploration and prevents the agent from solving the task. This can be seen in the second from the left plot in Fig. 3, where the mode constraint has induced a local optimum and stopped the agent from reaching the target state. We name the approach in Eq. (13) the greedy constrained strategy. We propose to overcome the issue of local optima induced by the mode constraint, by targeting exploration where the agent has high *epistemic uncertainty* in the mode constraint. We do this by augmenting our objective with an intrinsic exploration term that encourages the agent to explore where its gating network is *uncertain*. We use the entropy of the desired mode's gating function h_{k^*} over a trajectory $\bar{\mathbf{s}} = \{\mathbf{s}_0, \dots, \mathbf{s}_T\}$ as our intrinsic exploration term. This leads to our strategy taking the form,

$$\operatorname*{argmax}_{\pi \in \Pi} \underbrace{\mathbb{E}_{p(f_{k^*} \mid \mathcal{D}_{0:i})} \left[J(\pi, f_{k^*}) \right]}_{\text{greedy exploitation}} + \beta \underbrace{\mathcal{H}\left[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:i} \right]}_{\text{exploration}} \tag{16a}$$

$$\text{s.t.} \quad \underbrace{\Pr\left(\alpha_t = k^* \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \mathcal{D}_{0:i}\right) \geq 1 - \delta}_{\delta\text{-mode constraint}} \quad \forall t. \tag{16b}$$

where β is a hyperparameter that sets the level of exploration. Intuitively, the entropy term should enable the agent to escape local optima induced by the mode constraint as it encourages the agent to explore away from regions of the mode constraint that it has already observed. This is because the gating network's *epistemic uncertainty* will be low where it has observed the environment.

Epistemic vs aleatoric uncertainty When adopting this approach, it is extremely important to disentangle the gating network's *epistemic uncertainty* from its *aleatoric uncertainty*. For example, what is the meaning of the agent's belief in the mode indicator variable tending to a uniform distribution i.e. $\Pr(\alpha = k \mid \mathbf{s}, \mathcal{D}_{0:i}) = \frac{1}{K}$? Does it mean that the agent has not observed the environment near $\mathbf{s}$ (high *epistemic uncertainty*), or that it has observed the environment near $\mathbf{s}$ but is still uncertain which mode governs the dynamics (high *aleatoric uncertainty*)? We show the importance of disentangling these sources of uncertainty in our experiments. This motivated our GP-based gating network in the MoGPE dynamic model, as it principally disentangles the sources of uncertainty over α, by representing the *epistemic uncertainty* in the gating network's GPs.

Non-myopic exploration Further to providing a principled

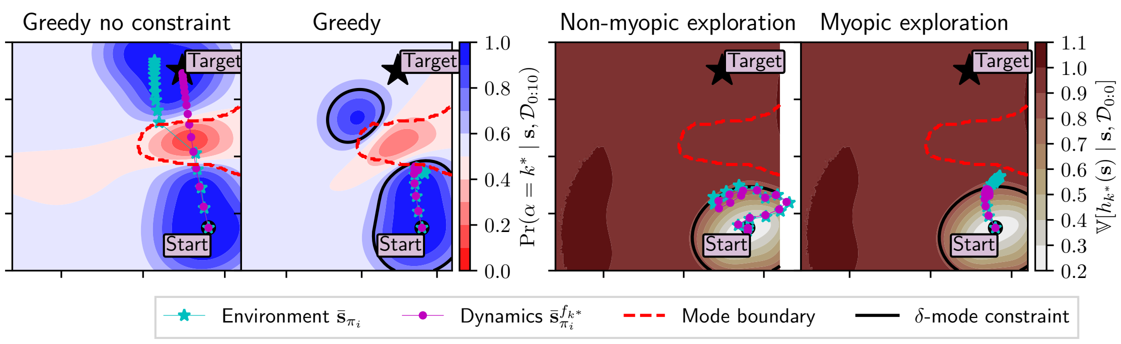

Figure 3: (Left) show trajectories found by the greedy exploitation strategy in Eq. (13), with and without the δ-mode constraint (black line). (Right) shows trajectories found by our strategy in Eq. (16) when using our non-myopic exploration term $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_0]$, compared to using the myopic exploration term $\frac{1}{T}\sum_{t=1}^{T}\mathcal{H}[h_{k^*}(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_0]$. We overlay the trajectories on the GP posterior variance associated with desired mode's gating function $\mathbb{V}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_0]$ at episode $i = 0$.

approach for disentangling sources of uncertainty, our GP-based gating network enables us to deploy a non-myopic exploration strategy. That is, we are able to steer the agent along a trajectory that will maximise uncertainty reduction over the entire trajectory, by considering the joint entropy over a trajectory $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_0]$. In contrast, myopic exploration considers uncertainty reduction of each state independently without considering the influence of other states in the trajectory, e.g. $\frac{1}{T}\sum_{t=0}^{T}\mathcal{H}[h_{k^*}(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_0]$.

Trajectory optimisation We use an open-loop trajectory optimisation policy because it naturally handles the δ-mode constraint in Eq. (16). Given a start state $\mathbf{s}_0$, ModeRL finds the action sequence $\bar{\mathbf{a}} = \{\mathbf{a}_0, \dots, \mathbf{a}_{T-1}\}$ solving the following optimal control problem,

$$\operatorname*{argmax}_{\mathbf{a}_0,\dots,\mathbf{a}_{T-1}} \underbrace{\mathbb{E}_{p(f_{k^*} \mid \mathcal{D}_{0:i})}\left[J(\pi, f_{k^*})\right]}_{\text{greedy exploitation}} + \beta \underbrace{\ln\left(\left|(2\pi e)\boldsymbol{\Sigma}_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}})\right|\right)}_{\text{exploration}} \tag{17a}$$

$$\text{s.t. } \underbrace{\Pr\left(\alpha_t = k^* \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \mathcal{D}_{0:i}\right) \geq 1 - \delta}_{\delta\text{-mode constraint}} \quad \forall t \in \{0, \dots, T\}, \tag{17b}$$

where the greedy exploitation term is an expectation over the state distribution obtained from making multi-step predictions in the desired dynamic mode's GP using moment-matching, see Eq. (32). $\boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}})$ is the predictive covariance of the desired mode's gating function posterior over the trajectory $\bar{\mathbf{s}}$, given in Eq. (25). It is worth noting a closed-loop policy can be obtained using MPC. However, this would require our algorithm to be made faster, for example, via locally linear dynamics approximations. Alternatively, a closed-loop policy could be learned, for example, via guided policy search (Levine and Koltun, 2013).

5 EXPERIMENTAL RESULTS

We test ModeRL on a 2D quadcopter navigation example, where the goal is to navigate to a target state $\mathbf{s}_f$, whilst avoiding a turbulent dynamic mode. See App. B for a schematic of the environment and details of the problem. Our experiments seek to answer the following questions:

1. Why does the greedy exploitation strategy in Eq. (13) fail to solve the mode-constrained problem in Eq. (4)?
2. Does ModeRL, our strategy in Eq. (16), solve the mode-constrained problem in Eq. (4)?
3. Is it important to disentangle the sources of uncertainty in the mode constraint?
4. Does our non-myopic exploration strategy help?
5. How does the constraint level δ influence training?

To evaluate ModeRL, we compare against the greedy baseline strategy in Eq. (13) (whose objective is used by PILCO (Deisenroth and Rasmussen, 2011), PETS (Chua et al., 2018), GP-MPC (Kamthe and Deisenroth, 2018)), both with and without the δ-mode constraint in Eq. (13b). Our experiments' configurations are detailed in App. C.

Given the quadratic reward function in Eq. (33) our objective has closed form. Further to this, we use two dynamic modes so our δ-mode constraint also has a closed form. At each episode, we then solve,

$$\operatorname*{argmax}_{\mathbf{a}_0,\dots,\mathbf{a}_{T-1}} \underbrace{-\|\boldsymbol{\mu}^{\text{MM}}_{\mathbf{s}_T} - \mathbf{s}_f\|_{\mathbf{H}} + \operatorname{Tr}\left(\mathbf{H}\boldsymbol{\Sigma}^{\text{MM}}_{\mathbf{s}_T}\right)}_{\text{terminal state reward}}$$
$$- \sum_{t=0}^{T-1}\Big(\underbrace{\|\boldsymbol{\mu}^{\text{MM}}_{\mathbf{s}_t} - \mathbf{s}_f\|_{\mathbf{Q}} - \operatorname{Tr}\left(\mathbf{Q}\boldsymbol{\Sigma}^{\text{MM}}_{\mathbf{s}_t}\right)}_{\text{state difference reward}} + \underbrace{\|\mathbf{a}_t\|_{\mathbf{R}}}_{\text{control reward}}\Big)$$
$$+ \beta \underbrace{\ln\left(\left|(2\pi e)\boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}})\right|\right)}_{\text{joint gating entropy}} \tag{18a}$$

$$\text{s.t. } \underbrace{\Phi\left(\frac{\boldsymbol{\mu}_{k^*}(\mathbf{s}_t)}{\sqrt{1 + \boldsymbol{\Sigma}^2_{k^*}(\mathbf{s}_t, \mathbf{s}_t)}}\right) \geq 1 - \delta}_{\delta\text{-mode constraint}} \quad \forall t \in \{0, \dots, T\} \tag{18b}$$

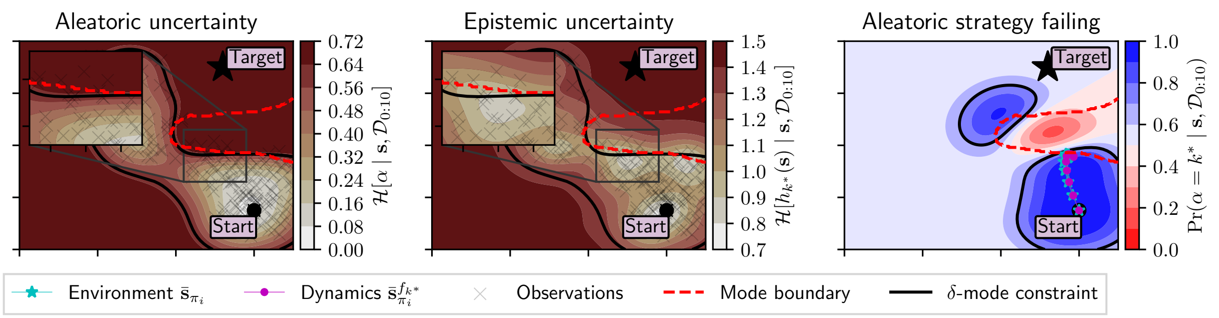

Figure 4: **Epistemic vs aleatoric uncertainty** Illustration of the shortcomings of using the entropy of the mode indicator variable (*aleatoric uncertainty*) for exploration. (Left) shows that the entropy of the mode indicator variable $\mathcal{H}[\alpha \mid \mathbf{s}, \mathcal{D}_{0:10}]$ (*aleatoric uncertainty*) at a region of the mode boundary which has been observed (black crosses), is high (red), even though we have observed the environment at these states. In contrast, (middle) shows that the entropy of the desired mode's gating function $\mathcal{H}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_{0:10}]$ (*epistemic uncertainty*) is low (white). (Right) visualises ModeRL converging to a local optimum when we replace the intrinsic exploration term in Eq. (16) with the entropy of the mode indicator variable.

where $\boldsymbol{\mu}_{k^*}(\mathbf{s}_t)$ and $\boldsymbol{\Sigma}^2_{k^*}(\mathbf{s}_t, \mathbf{s}_t)$ are the predictive mean and covariance of the desired mode's gating function posterior at $\mathbf{s}_t$, given in Eqs. (24) and (25). $\boldsymbol{\mu}^{\text{MM}}_{\mathbf{s}_t}$ and $\boldsymbol{\Sigma}^{\text{MM}}_{\mathbf{s}_t}$ are the mean and covariance of the state predicted t steps into the future $p(\mathbf{s}_t \mid \mathbf{s}_0, \mathbf{a}_{0:t-1}, \mathcal{D}_{0:i})$, obtained by cascading single-step predictions through the desired dynamic mode's GP with moment matching, given in Eq. (32). $\mathbf{H}$ and $\mathbf{Q}$ are user-defined, real symmetric positive semi-definite weight matrices and $\mathbf{R}$ is a user-defined, positive definite weight matrix. We solve Eq. (18) using sequential least squares quadratic programming (SLSQP) in SciPy (Virtanen et al., 2020), using the TensorFlow (Abadi et al., 2015) wrapper provided by GPflow (Matthews et al., 2017).

Why does greedy exploitation fail? Using the greedy strategy without the mode constraint results in the optimisation finding trajectories that leave the desired dynamic mode. This is illustrated in Fig. 3 (left), which shows the unconstrained greedy strategy converged to a solution navigating straight to the target state. As a result, the trajectory leaves the desired dynamic mode and passes through the turbulent dynamic mode. This is expected as the strategy is not aware of the turbulent dynamic mode. In contrast, the constrained greedy strategy in Fig. 3 (second left), does not leave the desired dynamic mode. However, the optimisation does not converge to the global optimum, i.e. the trajectory does not navigate to the target state $\mathbf{s}_f$. Instead, it gets stuck at the mode boundary, i.e. a local optimum.

Does our uncertainty-guided exploration work? The failure of the greedy strategy motivated our nonparametric dynamic model which learns the δ-mode constraint – as a latent variable – alongside the dynamic modes. Importantly, this enabled ModeRL (in Eq. (16)), to adopt an intrinsic exploration term, which targets exploration where the mode constraint's *epistemic uncertainty* is high. Fig. 2 shows four episodes i of ModeRL in the quadcopter navtigation task. Reading from left to right, the contours show how the agent's belief of being in the desired dynamic mode $\Pr(\alpha = k^* \mid \mathbf{s}, \mathcal{D}_{0:i})$ changes as the agent interacts with the environment, collects data $\mathcal{D}_{0:i}$ and updates its dynamic model, i.e. trains on $\mathcal{D}_{0:i}$. It shows the δ-mode-constrained region (black line) expanding as the agent trains on the new observations. The middle two plots show that ModeRL escaped the local optimum induced by the constraint. They also show that ModeRL provides some level of constraint satisfaction during training. Finally, the right-hand plot shows that ModeRL successfully navigated to the target state, i.e. it solved the task.

Is it important to disentangle sources of uncertainty? We now evaluate the importance of disentangling the sources of uncertainty in the mode constraint. In the left plot of Fig. 4 we visualise the entropy of the mode indicator variable $\mathcal{H}[\alpha \mid \mathbf{s}, \mathcal{D}_{0:10}]$ (*aleatoric uncertainty*) at a region of the mode boundary that the agent has observed (black crosses). The entropy is high (red) even though we have observed the environment at these states. As such, the agent would keep exploring the mode boundary even though it has been observed. In contrast, the middle plot shows the entropy of the gating function $\mathcal{H}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_{0:10}]$. The entropy is low (white) around the observations (black crosses), indicating that our GPs are capturing the mode constraint's *epistemic uncertainty*. The entropy of the desired mode's gating function (Fig. 4 middle) is a much better objective for exploration because it encourages the exploration away from the mode boundary once it has been observed. The right plot of Fig. 4 shows results when using the entropy of the mode indicator variable for the intrinsic exploration term. It shows that the agent is not able to escape the local optimum induced by the mode constraint.

Non-myopic vs myopic exploration? We now test the importance of our non-myopic exploration term, i.e. using the joint entropy of the desired mode's gating function over a trajectory $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_0]$, instead of taking the mean of the gating function's entropy at each time step $\frac{1}{T}\sum_{t=0}^{T} \mathcal{H}\left[h_{k^*}(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_{0:i}\right]$. In the right plots of

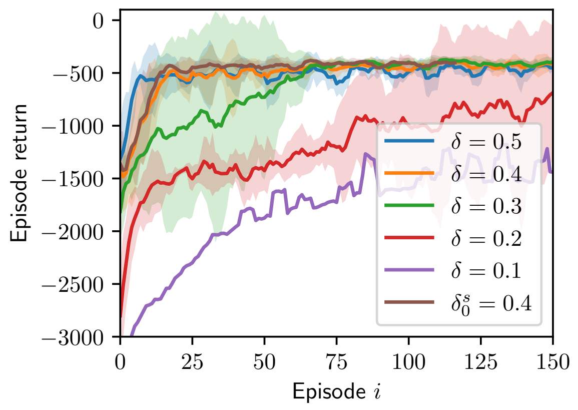

Figure 5: **Constraint level ablation** Training curves for different constraint levels δ. They show that looser constraints (high δ) have better sample efficiency. They further show that if the constraint is too tight (i.e. $\delta \leq 0.2$) then ModeRL gets stuck in local optima and cannot solve the task. Curves show the mean and 95% confidence interval of the episode return for five random seeds, at each episode i of training. The $\delta_0^s = 4$ experiment used an exponential schedule to tighten the constraint during training.

Fig. 3, we overlay the trajectories found with the non-myopic and myopic exploration terms, over the GP posterior variance associated with desired mode's gating function $\mathbb{V}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_0]$ at episode $i = 0$. The right-hand plot of Fig. 3 shows that using the myopic exploration term results in the trajectory navigating to a single state of high entropy and remaining in that state for the rest of the trajectory. This is an undesirable behaviour. In contrast, the second from right plot in Fig. 3 shows our non-myopic exploration term $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_0]$ spreading out. This is a desirable behaviour because it reduces the number of environment interactions that are required to solve the task, i.e. it improves sample efficiency.

Constraint satisfaction during training Finally, we evaluate how the constraint level δ influences training. Fig. 8 confirms that tightening the constraint (i.e. decreasing δ) leads to less constraint violations. This is shown by the accumulated number of episodes with constraint violations N_i^{α} increasing more slowly for lower δ's. Fig. 5 shows the training curves for five constraint levels δ. It shows that relaxing the constraint results in higher sample efficiency. This is indicated by the training curves for lower δ's converging in fewer episodes, e.g. $\delta = 0.5$ (blue). Fig. 5 further shows that ModeRL is not able to solve the task when the constraint is too tight, e.g. $\delta \leq 0.2$ (red/purple). This is indicated by the asymptotic performance not matching that of lower δ's. Finally, we place an exponentially decaying schedule on δ (δ_0^s brown), which tightens the constraint during training. This strategy solved the task in fewer episodes than the fixed δ experiments, indicating that it is more sample efficient. It also resulted in fewer constraint violations both during and after training.

5.1 Practical Considerations

Warm start trajectory optimisation In practice, the constrained optimisation in Eq. (16) fails if the initial trajectory does not satisfy the δ-mode constraint. We overcome this by solving an unconstrained optimisation to a "fake" target state in the initial state domain $\mathcal{S}_0$ and using it to warm start our trajectory optimiser.

Fixing model parameters during training Initially, ModeRL explores the desired dynamic mode and does not observe any state transitions from other modes. As such, we fix the kernel hyperparameters (e.g. lengthscale and signal variance) associated with the gating network GP. This prevents the δ-mode-constrained region from expanding significantly further than the observed data.

Inducing points We initialise each of the sparse GPs with a fixed number of inducing points uniformly sampled from $\mathcal{D}_0$. Although this approach worked well in our experiments, it is unlikely to scale to larger problems. As such, an interesting direction for future work is to study methods for dynamically adding new inducing points to each GP.

6 CONCLUSION

We introduced ModeRL, a Bayesian model-based RL algorithm for low-dimensional continuous control problems, that constrains exploration to a single dynamic mode (up to a given probability). Intuitively, ModeRL relaxes the mode-constrained RL problem in Def. 3.2, making it feasible, whilst still providing "some level" of constraint satisfaction during training. It uses a nonparametric dynamic model to learn the mode constraint – as a latent variable – alongside the dynamic modes. Importantly, it disentangles the sources of uncertainty in the learned mode constraint. Our experiments show that our nonparametric formulation of the mode constraint is essential for solving the quadcopter navigation task, as it enabled our planning algorithm to escape local optima that other strategies could not.

Limitations The main limitation of ModeRL is that it is restricted to lower dimensional problems, due to the difficulties of defining GP priors in high dimensions. Another (potentially unavoidable) downside of ModeRL, is that it must leave the desired dynamic mode in order to learn about the mode constraint. This is because we used internal sensing. However, in some applications, it may be possible to infer the mode constraint using external sensors. For example, in autonomous driving, it may be possible to infer dynamic modes associated with different road surfaces without leaving the desired dynamic mode, by using cameras. Finally, ModeRL uses an open-loop policy. An interesting direction for future work is to make our algorithm faster so that it can be used to formulate a closed-loop policy via MPC.

Acknowledgements

We thank ST John, Martin Trapp, Arno Solin, and Paul Chang for valuable discussions and feedback. This work was conducted whilst Aidan Scannell was a PhD student at the EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems (FARSCOPE) at the Bristol Robotics Laboratory. It was finished whilst funded by the Finnish Center for Artificial Intelligence (FCAI).

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*.

Altman, Eitan (1999). *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge.

Arcari, Elena, Lukas Hewing, Max Schlichting, and Melanie Zeilinger (July 2020). "Dual Stochastic MPC for Systems with Parametric and Structural Uncertainty". In: *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. PMLR, pp. 894–903.

Berkenkamp, Felix, Matteo Turchetta, Angela Schoellig, and Andreas Krause (2017). "Safe Model-based Reinforcement Learning with Stability Guarantees". In: *Advances in Neural Information Processing Systems*. Vol. 30, pp. 908–918.

Chua, Kurtland, Roberto Calandra, Rowan McAllister, and Sergey Levine (2018). "Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models". In: *Advances in Neural Information Processing Systems*. Vol. 31.

Cutler, M. and J. P. How (May 2015). "Efficient Reinforcement Learning for Robots Using Informative Simulated Priors". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2605–2612.

Deisenroth, M. P., D. Fox, and C. E. Rasmussen (Feb. 2015). "Gaussian Processes for Data-Efficient Learning in Robotics and Control". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2, pp. 408–423.

Deisenroth, Marc and Carl Rasmussen (Jan. 2011). "PILCO: A Model-Based and Data-Efficient Approach to Policy Search." In: *International Conference on Machine Learning*. Vol. 28, pp. 465–472.

Girard, Agathe, Carl Edward Rasmussen, Joaquin Quinonero-Candela, and Roderick Murray-Smith (2003). "Gaussian Process Priors with Uncertain Inputs? Application to Multiple-Step Ahead Time Series Forecasting". In: *Becker, S.* Vancouver, Canada: MIT Press.

Hensman, James, Nicolo Fusi, and Neil D Lawrence (2013). "Gaussian Processes for Big Data". In: *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*. Vol. 29, pp. 282–290.

Hensman, James, Alexander Matthews, and Zoubin Ghahramani (Feb. 2015). "Scalable Variational Gaussian Process Classification". In: *Artificial Intelligence and Statistics*. PMLR, pp. 351–360.

Hewing, Lukas, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger (2020). "Learning-Based Model Predictive Control: Toward Safe Learning in Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1, pp. 269–296.

Hoffman, Matthew D., David M. Blei, Chong Wang, and John Paisley (2013). "Stochastic Variational Inference". In: *Journal of Machine Learning Research* 14.4, pp. 1303–1347.

Kamthe, Sanket and Marc Deisenroth (Mar. 2018). "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1701–1710.

Kingma, Diederik P. and Jimmy Ba (Jan. 2017). "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]*.

Koller, T., F. Berkenkamp, M. Turchetta, and A. Krause (Dec. 2018). "Learning-Based Model Predictive Control for Safe Exploration". In: *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066.

Levine, Sergey and Vladlen Koltun (May 2013). "Guided Policy Search". In: *International Conference on Machine Learning*. PMLR, pp. 1–9.

MacKay, David J. C. (May 1992). "Bayesian Interpolation". In: *Neural Computation* 4.3, pp. 415–447.

Matthews, Alexander G. de G., Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman (Apr. 2017). "GPflow: A Gaussian Process Library Using TensorFlow". In: *Journal of Machine Learning Research* 18.40, pp. 1–6.

Pan, Yunpeng and Evangelos Theodorou (2014). "Probabilistic Differential Dynamic Programming". In: *Advances in Neural Information Processing Systems*. Vol. 27, pp. 1907–1915.

Pan, Yunpeng, Evangelos Theodorou, and Michail Kontitsis (2015). "Sample Efficient Path Integral Control under Uncertainty". In: *Advances in Neural Information Processing Systems*. Vol. 28.

Quiñonero-Candela, Joaquin and Carl Edward Rasmussen (2005). “A Unifying View of Sparse Approximate Gaussian Process Regression”. In: *Journal of Machine Learning Research* 6.65, pp. 1939–1959.

Roy, Julien, Roger Girgis, Joshua Romoff, Pierre-Luc Bacon, and Chris J. Pal (June 2022). “Direct Behavior Specification via Constrained Reinforcement Learning”. In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, pp. 18828–18843.

Scannell, Aidan, Carl Henrik Ek, and Arthur Richards (2021). “Trajectory Optimisation in Learned Multimodal Dynamical Systems Via Latent-ODE Collocation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE.

Schreiter, Jens, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint (2015). “Safe Exploration for Active Learning with Gaussian Processes”. In: *Machine Learning and Knowledge Discovery in Databases*. Cham: Springer International Publishing, pp. 133–149.

Sutton, R.S. and A.G. Barto (2018). *Reinforcement Learning, Second Edition: An Introduction*. Adaptive Computation and Machine Learning Series. MIT Press.

Titsias, Michalis (Apr. 2009). “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Artificial Intelligence and Statistics*. PMLR, pp. 567–574.

Tresp, Volker (2000). “Mixtures of Gaussian Processes”. In: *Advances in Neural Information Processing Systems*. Vol. 13, pp. 654–660.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors (2020). “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17, pp. 261–272.

Wachi, Akifumi, Yanan Sui, Yisong Yue, and Masahiro Ono (Apr. 2018). “Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1.

A DYNAMIC MODEL

This section provides details of our nonparametric dynamic model, including the sparse GP approximations it is built upon, our GP-based gating network, and how we propagate uncertainty when making multi-step predictions.

A.1 Sparse Gaussian Processes

Each dynamic mode's predictions conditioned on its inducing variables follows from the properties of multivariate normals and are given by,

$$p(\Delta\mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k)) = \mathbb{E}_{p(f_k(\hat{\mathbf{s}}_t) \mid f_k(\boldsymbol{\zeta}_k))}\left[p(\Delta\mathbf{s}_{t+1} \mid f_k(\hat{\mathbf{s}}_t))\right] \tag{19a}$$

$$p(f_k(\hat{\mathbf{s}}_t) \mid f_k(\boldsymbol{\zeta}_k)) = \mathcal{N}\left(f_k(\hat{\mathbf{s}}_t) \mid k_k(\hat{\mathbf{s}}_t, \boldsymbol{\xi})k_k(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}f_k(\boldsymbol{\zeta}_k), k_k(\hat{\mathbf{s}}_t,\hat{\mathbf{s}}_t) - k_k(\hat{\mathbf{s}}_t,\boldsymbol{\xi})k_k(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}k_k(\boldsymbol{\xi},\hat{\mathbf{s}}_t)\right), \tag{19b}$$

Similarly for the gating network we have,

$$\Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi})) = \mathbb{E}_{p(\mathbf{h}(\hat{\mathbf{s}}_t) \mid \mathbf{h}(\boldsymbol{\xi}))}\left[\Pr\left(\alpha_t = k \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right)\right] \tag{20a}$$

$$p(\mathbf{h}(\hat{\mathbf{s}}_t) \mid \mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} \mathcal{N}\left(h_k(\hat{\mathbf{s}}_t) \mid \hat{k}_k(\hat{\mathbf{s}}_t,\boldsymbol{\xi})\hat{k}_k(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}h_k(\boldsymbol{\xi}), \hat{k}_k(\hat{\mathbf{s}}_t,\hat{\mathbf{s}}_t) - \hat{k}_k(\hat{\mathbf{s}}_t,\boldsymbol{\xi})\hat{k}_k(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}\hat{k}_k(\boldsymbol{\xi},\hat{\mathbf{s}}_t)\right), \tag{20b}$$

Predictive posteriors As each GP's inducing variables are normally distributed, the functional form of their predictive posteriors are given by,

$$p(f_k(\hat{\mathbf{s}}_t) \mid \hat{\mathbf{s}}_t, \mathcal{D}_{0:i}) \approx \int p(f_k(\hat{\mathbf{s}}_t) \mid f_k(\boldsymbol{\zeta}_k))q(f_k(\boldsymbol{\zeta}_k))\mathrm{d}f_k(\boldsymbol{\zeta}_k) = \mathcal{N}\left(f_k(\hat{\mathbf{s}}_t) \mid \mathbf{A}_k\mathbf{m}_k, k_k(\hat{\mathbf{s}}_t,\hat{\mathbf{s}}_t) + \mathbf{A}_k(\mathbf{S}_k - k_k(\boldsymbol{\xi},\boldsymbol{\xi}))\mathbf{A}_k^T\right) \tag{21}$$

$$p(\mathbf{h}(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_{0:i}) \approx \prod_{k=1}^{K} q(h_k(\boldsymbol{\xi})) = \prod_{k=1}^{K} \mathcal{N}\left(h_k(\hat{\mathbf{s}}_t) \mid \hat{\mathbf{A}}_k\hat{\mathbf{m}}_k, \hat{k}_k(\hat{\mathbf{s}}_t,\hat{\mathbf{s}}_t) + \hat{\mathbf{A}}_k(\hat{\mathbf{S}}_k - \hat{k}_k(\boldsymbol{\xi},\boldsymbol{\xi}))\hat{\mathbf{A}}_k^T\right), \tag{22}$$

where $\mathbf{A}_k = k_k(\hat{\mathbf{s}}_t,\boldsymbol{\xi})k_k(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}$ and $\hat{\mathbf{A}}_k = \hat{k}_k(\hat{\mathbf{s}}_t,\boldsymbol{\xi})\hat{k}_k(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}$. Importantly, our predictive posteriors marginalise the inducing variables in closed form, with Gaussian convolutions.

Given our GP-based gating network, we are able to model the joint distribution over the gating function values $h_{k^*}(\bar{\mathbf{s}})$ along a trajectory $\bar{\mathbf{s}}$ with,

$$p(h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:i}) \approx q(h_{k^*}(\bar{\mathbf{s}})) = \mathcal{N}\left(h_{k^*}(\bar{\mathbf{s}}) \mid \boldsymbol{\mu}_{k^*}(\bar{\mathbf{s}}), \boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}},\bar{\mathbf{s}})\right) \tag{23}$$

where $\boldsymbol{\mu}_{k^*}(\cdot)$ and $\boldsymbol{\Sigma}^2_{k^*}(\cdot,\cdot)$ are sparse GP mean and covariance functions, given by,

$$\boldsymbol{\mu}_{k^*}(\bar{\mathbf{s}}) = \hat{k}_{k^*}(\bar{\mathbf{s}},\boldsymbol{\xi})\hat{k}_{k^*}(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}\hat{\mathbf{m}}_{k^*} \tag{24}$$

$$\boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}},\bar{\mathbf{s}}) = \hat{k}_{k^*}(\bar{\mathbf{s}},\bar{\mathbf{s}}) + \hat{k}_{k^*}(\bar{\mathbf{s}},\boldsymbol{\xi})\hat{k}_{k^*}(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}\left(\hat{\mathbf{S}}_{k^*} - \hat{k}_{k^*}(\boldsymbol{\xi},\boldsymbol{\xi})\right)\hat{k}_{k^*}(\boldsymbol{\xi},\boldsymbol{\xi})^{-1}\hat{k}_{k^*}(\boldsymbol{\xi},\bar{\mathbf{s}}), \tag{25}$$

where $\hat{k}_{k^*}$ and $\boldsymbol{\xi}$ are the kernel and inducing inputs associated with the desired mode's gating function respectively. This sparse approximation arises because our dynamical model uses sparse GPs and approximates the posterior with,

$$q(h_{k^*}(\bar{\mathbf{s}})) = \int p(h_{k^*}(\bar{\mathbf{s}}) \mid h_{k^*}(\boldsymbol{\xi}))q(h_{k^*}(\boldsymbol{\xi}))\mathrm{d}h_{k^*}(\boldsymbol{\xi}), \tag{26}$$

where $q(h_{k^*}(\boldsymbol{\xi})) = \mathcal{N}\left(h_{k^*}(\boldsymbol{\xi} \mid \hat{\mathbf{m}}_{k^*}, \hat{\mathbf{S}}_{k^*}\right)$.

A.2 Gating Network

Bernoulli ($K = 2$) Instantiating the model with two dynamic modes, $\alpha_t \in \{1, 2\}$, is a special case where only a single gating function is needed. This is because the output of a function $h(\hat{\mathbf{s}}_t)$ can be mapped through a sigmoid function sig : $\mathbb{R} \to [0, 1]$ and interpreted as a probability,

$$\Pr(\alpha_t = 1 \mid h(\hat{\mathbf{s}}_t)) = \mathrm{sig}(h(\hat{\mathbf{s}}_t)). \tag{27}$$

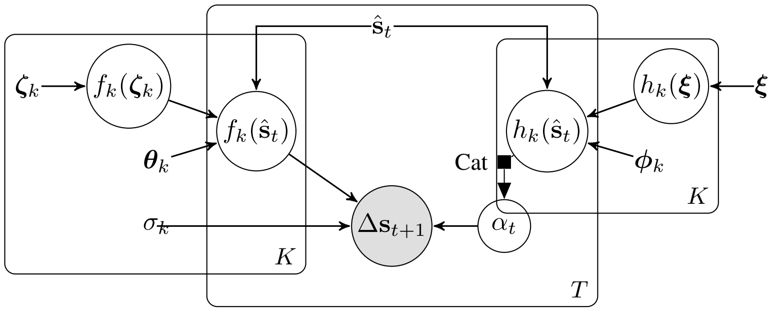

Figure 6: Graphical model of our augmented dynamic model where each state difference output $\Delta\mathbf{s}_{t+1}$ is generated by mapping the state-action input $\hat{\mathbf{s}}_t$ through the latent processes. The dynamic modes are shown on the left and the gating network on the right. The generative processes involve evaluating the gating network and sampling a mode indicator variable α_t. The indicated mode's latent function f_k and noise model $\mathcal{N}(0, \sigma_k^2)$ are then evaluated to generate the output $\Delta\mathbf{s}_{t+1}$.

If this sigmoid function satisfies the point symmetry condition then the following holds, $\Pr(\alpha_t = 2 \mid h(\hat{\mathbf{s}}_t)) = 1 - \Pr(\alpha_t = 1 \mid h(\hat{\mathbf{s}}_t))$. This only requires a single gating function and no normalisation term needs to be calculated. If the sigmoid function in Eq. (27) is selected to be the Gaussian cumulative distribution function $\Phi(h(\cdot)) = \int_{-\infty}^{h(\cdot)} \mathcal{N}(\tau|0,1)\mathrm{d}\tau$, then the mode probability can be calculated in closed form,

$$\begin{aligned}\Pr(\alpha_t = 1 \mid \hat{\mathbf{s}}_t) &= \int \Phi\left(h(\hat{\mathbf{s}}_t)\right) \mathcal{N}\left(h(\hat{\mathbf{s}}_t) \mid \mu_h, \sigma_h^2\right) \mathrm{d}h(\hat{\mathbf{s}}_t) \\ &= \Phi\left(\frac{\mu_h}{\sqrt{1+\sigma_h^2}}\right), \end{aligned} \tag{28}$$

where μ_h and σ_h^2 represent the mean and variance of the gating GP at $\hat{\mathbf{s}}_t$ respectively.

Softmax ($K > 2$) In the general case, when there are more than two dynamic modes, the gating network's likelihood is defined as the Softmax function,

$$\Pr\left(\alpha_t = k \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right) = \text{softmax}_k\left(\mathbf{h}(\hat{\mathbf{s}}_t)\right) = \frac{\exp\left(h_k(\hat{\mathbf{s}}_t)\right)}{\sum_{j=1}^{K} \exp\left(h_j(\hat{\mathbf{s}}_t)\right)}. \tag{29}$$

Each mode's mode probability $\Pr(\alpha_t = k \mid \hat{\mathbf{s}}_t)$ is then obtained by marginalising **all** of the gating functions. In the general case where $\Pr\left(\alpha_t = k \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right)$ uses the softmax function in Eq. (29), this integral is intractable, so we approximate it with Monte Carlo quadrature.

A.3 Uncertainty Propagation: Moment Matching

To obtain the state distributions $p(\mathbf{s}_1 \mid \mathbf{s}_0, \mathbf{a}_0, \mathcal{D}_{0:i}), \ldots, p(\mathbf{s}_T \mid \mathbf{s}_0, \mathbf{a}_{0:T-1}, \mathcal{D}_{0:i})$, for a given set of actions $\mathbf{a}_0, \ldots, \mathbf{a}_{T-1}$, we cascade single-step predictions through the desired dynamic mode's GP using moment matching, (Deisenroth and Rasmussen, 2011; Girard et al., 2003; Kamthe and Deisenroth, 2018; Quiñonero-Candela and Rasmussen, 2005). That is, we iteratively calculate,

$$p(\mathbf{s}_{t+1} \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \mathcal{D}_{0:i}) = \int\int \underbrace{p(\mathbf{s}_{t+1} \mid f_{k^*}(\hat{\mathbf{s}}_t))}_{\text{Gaussian likelihood}} \underbrace{p(f_{k^*}(\hat{\mathbf{s}}_t) \mid \hat{\mathbf{s}}_t, \mathcal{D}_{0:i})}_{\text{dynamics posterior}} p(\mathbf{s}_t \mid \mathbf{s}_0, \mathbf{a}_{0:t-1}, \mathcal{D}_{0:i}) \mathrm{d}f_{k^*}(\hat{\mathbf{s}}_t)\mathrm{d}\mathbf{s}_t, \tag{30}$$

with $p(\mathbf{s}_0) = \delta(\mathbf{s}_0)$. Importantly, the moment-matching approximation allows us to formulate uncertainty propagation using a deterministic system function,

$$\mathbf{z}_{t+1} = f_{k^*}^{\text{MM}}(\mathbf{z}_t, \mathbf{a}_t), \quad \mathbf{z}_t = [\boldsymbol{\mu}_{\mathbf{s}_t}^{\text{MM}}, \boldsymbol{\Sigma}_{\mathbf{s}_t}^{\text{MM}}], \tag{31}$$

where $\boldsymbol{\mu}_{\mathbf{s}_t}^{\text{MM}}$ and $\boldsymbol{\Sigma}_{\mathbf{s}_t}^{\text{MM}}$ are the mean and covariance of $p(\mathbf{s}_t \mid \mathbf{s}_0, \mathbf{a}_{0:t-1}, \mathcal{D}_{0:i})$. As we use deterministic actions we define the deterministic system function as

$$\mathbf{z}_{t+1} = f_{k^*}^{\text{MM}}(\hat{\mathbf{z}}_t), \quad \hat{\mathbf{z}}_t = [\boldsymbol{\mu}_{\hat{\mathbf{s}}_t}^{\text{MM}}, \boldsymbol{\Sigma}_{\hat{\mathbf{s}}_t}^{\text{MM}}], \quad \boldsymbol{\mu}_{\hat{\mathbf{s}}_t}^{\text{MM}} = [\boldsymbol{\mu}_{\mathbf{s}_t}^{\text{MM}}, \mathbf{a}_t], \quad \boldsymbol{\Sigma}_{\hat{\mathbf{s}}_t}^{\text{MM}} = \text{blkdiag}[\boldsymbol{\Sigma}_{\mathbf{s}_t}^{\text{MM}}, \mathbf{0}]. \tag{32}$$

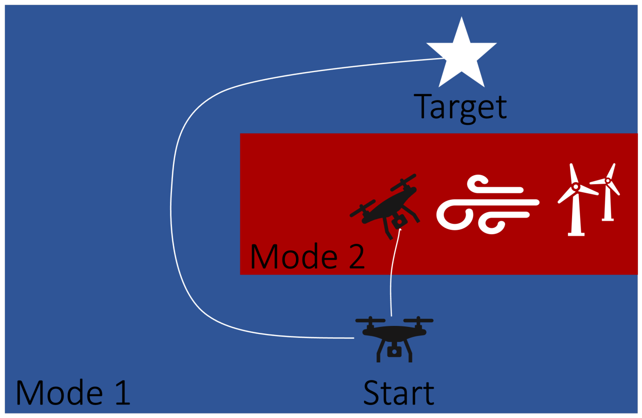

Figure 7: **Mode-constrained quadcopter navigation** - Diagram showing a top-down view of a quadcopter subject to two dynamic modes: 1) an *operable* dynamic mode (blue) and 2) an *inoperable*, turbulent dynamic mode induced by a strong wind field (red). The goal is to navigate to the target state $\mathbf{s}_f$ (white star), whilst avoiding the turbulent dynamic mode (red).

B ILLUSTRATIVE EXAMPLE

ModeRL is tested on a 2D quadcopter navigation example shown in Fig. 7. The goal is to fly the quadcopter from an initial state $\mathbf{s}_0$, to a target state $\mathbf{s}_f$ (white star). However, it considers a quadcopter operating in an environment subject to spatially varying wind – induced by a fan – where two dynamic modes can represent the system,

Mode 1 is an *operable* dynamic mode away from the fan,

Mode 2 is an *inoperable*, turbulent dynamic mode in front of the fan.

The turbulent dynamic mode is subject to higher drift (in the negative x direction) and to higher diffusion (transition noise). It is hard to know the exact turbulent dynamics due to complex and uncertain interactions between the quadcopter and the wind field. Further to this, controlling the system in the turbulent dynamic mode may be infeasible. This is because the unpredictability of the turbulence may cause catastrophic failure. Therefore, when flying the quadcopter to the target state $\mathbf{s}_f$, it is desirable to find trajectories that avoid entering this turbulent dynamic mode.

The state-space of the velocity-controlled quadcopter example consists of the 2D Cartesian coordinates $\mathbf{s} = (x, y)$ and the controls consist of the speed in each direction, given by $\mathbf{a} = (v_x, v_y)$.

The reward function is given by,

$$r(\bar{\mathbf{s}}, \bar{\mathbf{a}}) = -\left(\mathbf{s}_T - \mathbf{s}_f\right)^T \mathbf{H} \left(\mathbf{s}_T - \mathbf{s}_f\right) - \sum_{t=0}^{T-1} \left(\left(\mathbf{s}_t - \mathbf{s}_f\right)^T \mathbf{Q} \left(\mathbf{s}_t - \mathbf{s}_f\right) + \mathbf{a}_t^T \mathbf{R} \mathbf{a}_t \right) \tag{33}$$

$$= -\|\mathbf{s}_T - \mathbf{s}_f\|_{\mathbf{H}} - \sum_{t=0}^{T-1} \left(\|\mathbf{s}_t - \mathbf{s}_f\|_{\mathbf{Q}} + \|\mathbf{a}_t\|_{\mathbf{R}} \right) \tag{34}$$

where $\mathbf{H}$ and $\mathbf{Q}$ are user-defined, real symmetric positive semi-definite weight matrices and $\mathbf{R}$ is a user-defined, positive definite weight matrix. In our experiments we set both $\mathbf{Q}$ and $\mathbf{R}$ to be the identify matrix $\mathbf{I}$ and $\mathbf{H}$ to be $100\mathbf{I}$.

C EXPERIMENT CONFIGURATION

This section details how the experiments were configured.

Initial data set $\mathcal{D}_0$ The initial data set was collected by simulating 50 random trajectories with horizon $T = 15$ from the start state $\mathbf{s}_0 = \{x_0, y_0\}$ and terminating them when they left the initial state domain $\mathcal{S}_0 = \{\mathbf{s} \in \mathcal{S} \mid x_0 - 1 < x < x_0 + 1, y_0 - 1 < y < y_0 + 1\}$.

Model learning In all experiments, the dynamic model was instantiated with $K = 2$ modes. Each dynamic mode's GP used a separate RBF kernel with automatic relevance determination (ARD) for each output dimension d but shared its inducing variables for each output dimension d. Further to this, each dynamic mode learned a separate constant mean function c_{f_k} and separate noise variances for each output dimension. The gating network used a single gating function with an RBF kernel with ARD and a zero mean function. An early stopping callback was used to terminate the dynamic model's training. The early stopping callback used a min delta of 0 and a patience of 50. This meant that training terminated after 50 epochs of no improvement. All of the dynamic model's initial parameters are shown in Tab. 1. We also fix the kernel hyperparameters (e.g. lengthscale and signal variance) associated with the gating network, along with the noise variance in the non-desired (turbulent) dynamic mode.

Policy In all experiments, ModeRL used a horizon of $T = 15$ and was configured with $\delta = 0.3$. At each episode, ModeRL uses the previous solution as the initial solution for the trajectory optimiser.

Intrinsic schedule In our experiments, we used an exponential decay schedule (with decay rate 0.96 and decay steps 10) on the exploration weight β.

Table 1: Experiment configuration and parameter settings.

	DESCRIPTION	**SYMBOL**	**VALUE**
Environment	Start state	$\mathbf{s}_0$	$[2.0, -2.5]$
	Target state	$\mathbf{s}_f$	$[1.2, 3.0]$
	Terminal state reward weight	$\mathbf{H}$	$\text{diag}([100, 100])$
	State reward weight	$\mathbf{Q}$	$\text{diag}([1, 1])$
	Control reward weight	$\mathbf{R}$	$\text{diag}([1, 1])$
Policy π	Constraint level	δ	0.3
	Horizon	T	15
	Exploration weight (initial)	β_0	10.0
Dynamics optimiser	Batch size	N_b	64
	Num epochs	N/A	20000
	Num gating samples	N/A	1
	Num expert samples	N/A	1
	Learning rate	N/A	0.01
	Epsilon	N/A	1×10^{-8}
Dynamic mode 1 f_1	Constant mean function	c_{f_1}	$[0, 0]$
	Kernel variance ($d = 1$)	$\sigma^2_{f_1}$	1
	Kernel lengthscales ($d = 1$)	l_{f_1}	$[1, 1, 1, 1]$
	Kernel variance ($d = 2$)	$\sigma^2_{f_1}$	1
	Kernel lengthscales ($d = 2$)	l_{f_1}	$[1, 1, 1, 1]$
	Likelihood variance	σ^2_1	$\text{diag}([1, 1])$
	Num inducing points	M_{f_1}	50
	Inducing inputs	$\boldsymbol{\zeta}_1$	$\boldsymbol{\zeta}_1 \subseteq \hat{\mathbf{S}}_0$ with $\#\boldsymbol{\zeta}_1 = M$
Dynamic mode 2 f_2	Constant mean function	c_{f_2}	$[0, 0]$
	Kernel variance ($d = 1$)	$\sigma^2_{f_2}$	1
	Kernel lengthscales ($d = 1$)	l_{f_2}	$[1, 1, 1, 1]$
	Kernel variance ($d = 2$)	$\sigma^2_{f_2}$	1
	Kernel lengthscales ($d = 2$)	l_{f_2}	$[1, 1, 1, 1]$
	Likelihood variance	σ^2_2	$\text{diag}([1, 1])$
	Num inducing points	M_{f_2}	50
	Inducing inputs	$\boldsymbol{\zeta}_2$	$\boldsymbol{\zeta}_2 \subseteq \hat{\mathbf{S}}_0$ with $\#\boldsymbol{\zeta}_2 = M$
Gating function 1 h_1	Kernel variance	$\sigma^2_{h_1}$	1
	Kernel lengthscales	l_{h_1}	$[0.8, 0.8]$
	Active dims	N/A	$[0, 1]$
	Num inducing points	M_{h_1}	90
	Inducing inputs	$\boldsymbol{\xi}$	$\boldsymbol{\xi} \subseteq \hat{\mathbf{S}}_0$ with $\#\boldsymbol{\xi} = M_h$

D FURTHER EXPERIMENTS

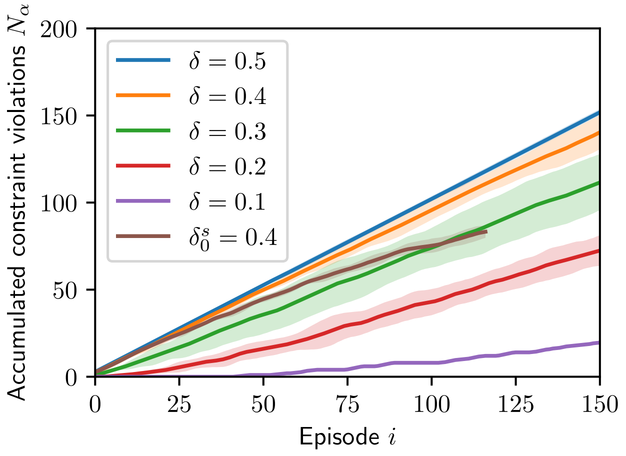

Figure 8: **Constraint level ablation** Accumulated number of constraint violations at each episode i of training, for different constraint levels δ. It shows that tighter constraints (i.e. lower δ) result in less constraint violations during training. Lines show the mean and 95% confidence interval of the accumulated number of constraint violations for five random seeds, at each episode i of training. The $\delta_0^s = 4$ experiment used an exponential schedule to tighten the constraint during training.