Experiment Design with Gaussian Process Regression with Applications to Risk-Aware Control

Sean Anderson, Katie Byl, João P. Hespanha

***Abstract*— Learning for control in repeated tasks allows for well-designed experiments to gather the most useful data. We consider the setting in which we use a data-driven controller that does not have access to the true system dynamics. Rather, the controller uses inferred dynamics based on the available information. In order to acquire data that is beneficial for this controller, we present an experimental design approach that leverages the current data to improve expected control performance. We focus on the setting in which inference on the unknown dynamics is performed using Gaussian processes. Gaussian processes not only provide uncertainty quantification but also allow us to leverage structures inherit to Gaussian random variables. Through this structure, we design experiments via gradient descent on the expected control performance with respect to the experiment input. In particular, we focus on a risk-aware minimum expected time control problem. Numerical demonstrations of our approach indicate our experimental design outperforms relevant benchmarks.**

I. Introduction

Many safety-critical tasks such as driving around a race track or repeated robotic motions in a novel environment can leverage learning-based methods to improve control performance. In the setting where the dynamics of the system are unknown, a natural question is how to use a fixed number of trials to gather the most informative data for improving control performance.

We motivate our problem with the example of a race car driver learning the nuances of the vehicle dynamics on a new track. If the driver is given test laps before having to race, the driver should choose each test lap carefully so as to improve their expected performance on the race lap. For our problem, we consider a controller that minimizes an objective (e.g. racing) given dynamics inferred from gathered data. In order to collect the most useful data for control, we design an experiment that aims to improve the data-driven controller's expected performance as much as possible.

In this work we present a novel formulation for experiment design for data-driven control. Section II defines the system dynamics as an unknown discrete-time process with additive noise. We define a data-driven controller that minimizes a control objective given the currently available data. When thinking about experiment design, this naturally leads to a formulation in which we consider how to augment our dataset in order to the improve our control performance as much as possible. We design the experiment to minimize the expected control performance given data available at the time of the experiment.

In Section III, we focus our attention on using Gaussian processes for inference on the system dynamics. In safety-critical settings where we need uncertainty quantification of both epistemic and aleatoric uncertainty of the dynamics, Gaussian processes allow us to derive tractable formulations for both the optimal control and experiment design problems.

Using this structure, we present an approach to solving the experiment design problem that takes the gradient of the expected control performance with respect to the experiment input. We derive a numerical approximation to the analytic gradient and then use stochastic gradient descent to improve the expected performance.

We consider a prototypical risk-aware control problem in Section IV in which we try to minimize the expected time to a target set while avoiding an unsafe set to a desired probability level. In particular, we focus on joint-chance constraints in time such that the total probability of entering the unsafe set is minimized as in [1], [2]. We solve this using a dynamic programming approach and a Lagrangian formulation for the joint-chance constraint.

Numerical results in Section V demonstrate the effectiveness of the experiment design formulation for the minimum time problem. We consider the scenario in which we add one well-designed experiment to an existing dataset. We compare the performance of our algorithm against suitable benchmarks by first considering the percentage of experiments that result in feasible controllers in low-data settings. We then also consider the minimum time for controllers that are feasible. These indicate that our experiment design outperforms the benchmarks and highlights potential future work.

Using Gaussian process regression for learning dynamical models for the purposes of safe control has been explored in numerous works including [3], [4]. These papers focus on using Gaussian processes trained on sufficient data to perform safety-critical control. Exactly how to choose that data when given limited time to gather data is not explored. Furthermore, active learning using Gaussian processes has been demonstrated in [5], [6], [7]. Active learning is often used in the dual control context [8]; the dual control problem differs from this work because we separate out data gathering from the control trial while a dual control approach would try to simultaneously perform control and estimate unknown

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 2139319. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

The authors are with Department of Electrical and Computer Engineering, University of California Santa Barbara, Santa Barbara, CA, 93106 USA seananderson@ucsb.edu, katiebyl@ece.ucsb.edu, hespanha@ece.ucsb.edu

quantities. Some work has been done on experiment design for Gaussian processes, mainly with respect to information theoretic objectives and not necessarily for dynamical systems [9], [10], [11], [12], [13], [14].

Overall then, our approach differs from previous work in mainly two aspects: first, the experiment design criteria differs from information theoretic approaches by using the expected control performance in order to focus learning in control-relevant regions, and second, our approach to solving the resulting experiment design is novel in how it leverages the structure of Gaussian processes.

II. Experiment Design for Data-Driven Control

We consider systems with discrete-time dynamics

$$x_{t+1} = h(x_t, u_t) + w_t, \tag{1}$$

where $x_t \in \mathbb{R}^{n_x}$ denotes the state at time t, $u_t \in \mathbb{R}^{n_u}$ denotes the input, and $w_t \sim \mathcal{N}(0, \Sigma_w)$ is independent, identically distributed process noise. We do not know the function $h(\cdot)$ and will use data gathered from previous trials to inform our belief of the values $h(\cdot)$ will take at particular state-input pairs. We consider a control objective given by

$$G(X, U) \tag{2}$$

over time horizon N such that the matrix $X \in \mathbb{R}^{n_x \times N}$ is a concatenation of the states, x_t, and similarly $U \in \mathbb{R}^{n_u \times N}$.

While $h(\cdot)$ is not known, we have prior information about it such that we can condition our expected control objective on it. We incorporate available data by letting the set:

$$\mathcal{D} = \{x_i, u_i, y_i\}_{i=0}^{M} \tag{3}$$

consist of M triples satisfying

$$y_i = h(x_i, u_i) + w_i \tag{4}$$

with the understanding that the data come from a repeated task in which case (1) provides triples that satisfy (4) for $y_i = x_{t+1}, x_i = x_t, u_i = u_t$.

After gathering some data, $\mathcal{D}$, we want to conduct a trial that minimizes an expected control objective (e.g. racing a car around a track as fast as possible). In this data-driven setting, our control objective is to minimize a conditional expectation based on the available information and the process noise:

$$J(\mathcal{D}) := \min_{U \in \mathcal{U}} \ \mathbb{E}[G(X, U) \mid \mathcal{D}] \tag{5}$$

where X is described by (1).

In order to gather the data that is most useful for the control performance, we construct our experiment design criteria to minimize the post-experiment expected cost of the optimal control problem. To this effect we construct an experiment that will generate a new dataset to augment the existing dataset $\mathcal{D}$ such that if we were to race after the experiment trial, our control performance would be as good as possible:

$$\min_{\bar{U} \in \mathcal{U}} \ \mathbb{E}\big[J\big(\mathcal{D}_+(\bar{X}, \bar{U})\big) \mid \mathcal{D}\big], \tag{6}$$

where $\bar{U}$ is the control signal for the new dataset, $\bar{X}$ the corresponding state trajectory from (1), $\bar{Y}$ the corresponding measurements, and $\mathcal{D}_+(\bar{X}, \bar{U}) := \mathcal{D} \cup (\bar{X}, \bar{U}, \bar{Y})$. The expectation here is with respect to the possible trajectory, $\bar{X}$, with the current understanding of the dynamics given the data is $\mathcal{D}$.

III. Gaussian Process Experiment Design

We now move from the general setting to one in which uncertainty in $h(\cdot)$ is modeled as a Gaussian process.

A. Model inference

A Gaussian process (GP) is a collection of indexed random variables that are jointly Gaussian, any subset of which is also jointly Gaussian. We assume $h(x, u)$ is a Gaussian process indexed by state-input pairs $z := (x, u)^T$ with mean, $m(\cdot)$, and covariance function, $k(\cdot, \cdot)$:

$$m(z) = \mathbb{E}[h(z)], \tag{7}$$

$$k(z_k, z_j) = \mathbb{E}[(h(z_k) - m(z_k))(h(z_j) - m(z_j))]. \tag{8}$$

When comparing two points, z_k, z_j, the kernel $k(\cdot, \cdot)$ shapes the covariance of the two random variables. In particular, we use the squared exponential kernel, $k(z_k, z_j) = \sigma_f^2 \exp\big(\frac{-(z_k - z_j)^T (z_k - z_j)}{2\ell^2}\big)$, which is infinitely differentiable [15].

Corollary 1: [16]. Assume $h(\cdot)$ is a Gaussian process and the process noise covariance is diagonal such that $\Sigma_w = \text{diag}([\sigma_{w,1}^2, ..., \sigma_{w,n_x}^2])$. Then the conditional distribution of the d-th entry of x_t ($d \in \{1, ..., n_x\}$) given the dataset $\mathcal{D}$, the state $x_t = x$, and the control input $u_t = u$ is a normal distribution with mean and variance:

$$\mu_d(z; \mathcal{D}) := m^d(z) + K_{z\mathbf{Z}}^d (K_{\mathbf{ZZ}}^d + \sigma_{w,d}^2 I)^{-1} (\mathbf{Y}^d - m^d(\mathbf{Z})) \tag{9a}$$

$$\sigma_d^2(z; \mathcal{D}) := K_{zz}^d - K_{z\mathbf{Z}}^d (K_{\mathbf{ZZ}}^d + \sigma_{w,d}^2 I)^{-1} K_{\mathbf{Z}z}^d + \sigma_{w,d}^2 \tag{9b}$$

where $\mathbf{Z} := [z_0, ..., z_M]$ is the matrix of training indices and $\mathbf{Y} := [y_0, ..., y_M]$ is a matrix of noisy measurements in $\mathcal{D}$. K are Gram matrices that are composed of kernel evaluations: $[K_{\mathbf{ZZ}}^d]_{lj} = k^d(z_l, z_j), [K_{\mathbf{Z}z}^d]_j = k^d(z_j, z), K_{zz}^d = k^d(z, z)$.

Proof: We consider multi-output GPs with independent outputs and diagonal process noise matrix $\Sigma_w = \text{diag}([\sigma_{w,1}^2, ..., \sigma_{w,n_x}^2])$. For each triple (x_i, u_i, y_i) in $\mathcal{D}$, (4) provides a noisy measurement of the GP. We incorporate this in the prior on our observations such that

$$\mathbf{Y}^d \sim \mathcal{N}(m^d(\mathbf{Z}), K_{\mathbf{ZZ}}^d + \sigma_{w,d}^2 I). \tag{10a}$$

We go on to express the joint distribution of the measurements at observed points, $\mathbf{Y}^d$ corresponding to $\mathbf{Z}$, and at y^d corresponding to a new test point z:

$$\begin{bmatrix} \mathbf{Y}^d \\ y^d \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m^d(\mathbf{Z}) \\ m^d(z) \end{bmatrix}, \begin{bmatrix} K_{\mathbf{ZZ}}^d + \sigma_{w,d}^2 I & K_{\mathbf{Z}z}^d \\ K_{z\mathbf{Z}}^d & K_{zz}^d \end{bmatrix} \right) \tag{10b}$$

This joint distribution, $p(\mathbf{Y}^d, y^d \mid z, \mathbf{Z})$, can be conditioned on the dataset $\mathcal{D}$ using $p(y^d \mid \mathbf{Y}^d, z, \mathbf{Z}) = p(\mathbf{Y}^d, y^d \mid z, \mathbf{Z}) / p(\mathbf{Y}^d)$ to obtain the conditional distribution of y^d

given $\mathcal{D}$ and z as a normal distribution with mean and variance

$$\mu_d(z;\mathcal{D}) = m^d(z) + K^d_{z\mathbf{Z}}(K^d_{\mathbf{ZZ}} + \sigma^2_{w,d}I)^{-1}(\mathbf{Y}^d - m^d(\mathbf{Z})) \tag{10c}$$

$$\sigma^2_d(z;\mathcal{D}) = K^d_{zz} - K^d_{z\mathbf{Z}}(K^d_{\mathbf{ZZ}} + \sigma^2_{w,d}I)^{-1}K^d_{\mathbf{Z}z} + \sigma^2_{w,d}. \tag{10d}$$

Since $y = h(x,u) + w$, by (1) we have that $y = x_{t+1}$ for a given trajectory. We then have that the conditional distribution of x_{t+1},given $\mathcal{D}$ and a particular state and input, is a normal distribution with moments (10c,10d).

Our GPs are independent such that the predictive equations are given by stacking the individual outputs with diagonal matrix $\Sigma(z;\mathcal{D}) := \text{diag}([\sigma^2_1(z;\mathcal{D}), ..., \sigma^2_{n_x}(z;\mathcal{D})])$ and vector $\mu(z;\mathcal{D}) := [\mu_1(z;\mathcal{D}), ..., \mu_{n_x}(z;\mathcal{D})]^T$. We refer to the specific index set (x,u) instead of z for clarity going forward. ■

B. Experiment Design

We develop a gradient descent approach to the experiment design optimization (6) as described in Algorithm 1. We derive a gradient estimator that uses Monte Carlo sampling to approximate the expectation in (6). The resulting algorithm takes in the available data $\mathcal{D}$, an initial guess for the optimization variable $\bar{U}$, and the initial condition for the experiment $\bar{x}_0$; using these and a Monte Carlo sample from the GP, we generate a possible experiment trajectory. For each sample, we solve the optimal control problem in (5) and then compute the gradient of the objective function with respect to $\bar{U}$. We use a batch of samples to perform stochastic gradient descent until a stopping condition is met.

In particular, stochastic gradient descent on the variable $\bar{U}$ requires computation of

$$\frac{\partial}{\partial \bar{U}}\mathbb{E}[J(\mathcal{D}_+(\bar{X},\bar{U})) \mid \mathcal{D}]. \tag{11}$$

Defining $f(\bar{X},\bar{U}) := J(\mathcal{D}_+(\bar{X},\bar{U}))$ and denoting by $p(\bar{X} \mid \bar{U})$ the $\bar{U}$-dependent conditional probability density function of $\bar{X}$ given $\mathcal{D}$, we can rewrite (11) as

$$\frac{\partial}{\partial \bar{U}}\int f(\bar{X},\bar{U})p(\bar{X} \mid \bar{U})d\bar{X}. \tag{12}$$

When both $f(\cdot)$ and $p(\cdot)$ are differentiable with respect to $\bar{U}$, this derivative is given by

$$\int \frac{\partial f(\bar{X},\bar{U})}{\partial \bar{U}}p(\bar{X} \mid \bar{U})d\bar{X} + \int f(\bar{X},\bar{U})\frac{\partial p(\bar{X} \mid \bar{U})}{\partial \bar{U}}d\bar{X}. \tag{13}$$

While the first term in this expression can be easily computed using Monte Carlo integration, the second is generally more difficult. The following result uses the so-called "reparameterization trick" [17] to obviate this difficulty, allowing us to directly use Monte Carlo integration to compute (11).

Theorem 1: Suppose the GP regressor (9) has a kernel and mean function both differentiable with respect to the index. Furthermore, assume that the value of the objective function is differentiable with respect to $\bar{U}$. Then we can construct an estimate of (11) using

$$\sum_{l=1}^{L}\frac{\partial c(v^l_0, v^l_1, ..., v^l_{N-1}, \bar{U})}{\partial \bar{U}}. \tag{14a}$$

Here v is a standard normal random variable and

$$\begin{aligned} &c(v^l_0, v^l_1, ..., v^l_{N-1}, \bar{U}) := \\ &\quad J\big(\mathcal{D}_+([\bar{x}_0, \mu(\bar{x}_0,\bar{u}_0) + \Sigma(\bar{x}_0,\bar{u}_0)^{\frac{1}{2}}v_0, \\ &\quad \mu(\bar{x}_1,\bar{u}_1) + \Sigma(\bar{x}_1,\bar{u}_1)^{\frac{1}{2}}v_1, ..., \\ &\quad \mu(\bar{x}_{N-1},\bar{u}_{N-1}) + \Sigma(\bar{x}_{N-1},\bar{u}_{N-1})^{\frac{1}{2}}v_{N-1}], \bar{U})\big). \end{aligned} \tag{14b}$$

$\bar{x}_t$ is recursively defined by $\bar{x}_t = \mu(\bar{x}_{t-1}\bar{u}_{t-1}) + \Sigma(\bar{x}_{t-1},\bar{u}_{t-1})^{\frac{1}{2}}v_{t-1}$ with $\mu(\cdot)$ and $\Sigma(\cdot)$ given by (9). In our numerical examples the gradient in (14a) of (14b) is computed by automatic differentiation.

Proof: Starting from the joint-density describing the forward simulation of the dynamics given $\mathcal{D}$, we express the expected value in (11) as

$$\mathbb{E}\big[J\big(\mathcal{D}_+(\bar{X},\bar{U})\big) \mid \mathcal{D}\big] = \int J\big(\mathcal{D}_+(\bar{X},\bar{U})\big)p(\bar{X} \mid \bar{x}_0,\bar{U},\mathcal{D})d\bar{X}, \tag{15a}$$

where the density function describes the distribution of the trajectory given the input sequence $\bar{U}$, the currently available data $\mathcal{D}$, and the process noise for each time step. Using Bayes' rule for probability density functions, the joint distribution of the states $p(\bar{X} \mid \bar{x}_0,\bar{U},\mathcal{D})$ can be recursively expanded as

$$\begin{aligned} &p(\bar{X} \mid \bar{x}_0,\bar{U},\mathcal{D}) \\ &\quad = p(\bar{x}_N, ..., \bar{x}_2 \mid \bar{x}_0,\bar{x}_1,\bar{U},\mathcal{D})p(\bar{x}_1 \mid \bar{x}_0,\bar{U},\mathcal{D}). \\ &\quad = p(\bar{x}_N, ..., \bar{x}_3 \mid \bar{x}_0,\bar{x}_1,\bar{x}_2,\bar{U},\mathcal{D})\times \\ &\qquad p(\bar{x}_2 \mid \bar{x}_0,\bar{x}_1,\bar{U},\mathcal{D})p(\bar{x}_1 \mid \bar{x}_0,\bar{U},\mathcal{D}). \\ &\quad = \Pi^N_{t=1}p(\bar{x}_t \mid \bar{x}_0, ..., \bar{x}_{t-1},\bar{U},\mathcal{D}). \end{aligned} \tag{15b}$$

As a consequence of (1), our dynamics are Markovian such that we have

$$p(\bar{x}_t \mid \bar{x}_0, ..., \bar{x}_{t-1},\bar{U},\mathcal{D}) = p(\bar{x}_t \mid \bar{x}_{t-1},\bar{u}_{t-1},\mathcal{D}). \tag{15c}$$

From Corollary 1, we know $p(\bar{x}_t \mid \bar{x}_{t-1},\bar{u}_{t-1},\mathcal{D})$ is described by (9) for a particular state-input pair. Using the reparameterization trick [17], Gaussian random variables can be expressed in terms of a standard normal, v, where in our case

$$\bar{x}_{t+1} = \mu(\bar{x}_t,\bar{u}_t;\mathcal{D}) + \Sigma(\bar{x}_t,\bar{u}_t;\mathcal{D})^{\frac{1}{2}}v, \tag{15d}$$

and $\Sigma(\bar{x}_t,\bar{u}_t;\mathcal{D})^{\frac{1}{2}}$ is a diagonal matrix of the standard deviation of each output dimension d since from (9) the outputs are independent. Given $\bar{x}_0$, we apply this change of variables recursively starting at $\bar{x}_1$:

$$\mathbb{E}\big[J\big(\mathcal{D}_+(\bar{X},\bar{U})\big) \mid \mathcal{D}\big] = \tag{15e}$$

$$\int ... \int J\big(\mathcal{D}([\bar{x}_0, \mu(\bar{x}_0,\bar{u}_0) + \Sigma(\bar{x}_0,\bar{u}_0)^{\frac{1}{2}}v_0, \bar{x}_2, ..., \bar{x}_N], \bar{U})\big) \times p(v_0)\Pi^N_{t=2}p(\bar{x}_t \mid \bar{x}_{t-1},\bar{u}_{t-1},\mathcal{D})dv_0 d\bar{x}_2 ... d\bar{x}_N \tag{15f}$$

since $d\bar{x}_{t+1} = \Sigma(\bar{x}_t, \bar{u}_t)^{\frac{1}{2}} dv_t$, and for a scalar normal random variable v_t the change of variable leads to $p(\mu + \sigma v_t | \bar{x}_t, \bar{u}_t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}v^2\right)\sigma = p(v_t)$; we can extend this to our problem since the outputs of (9) are independent such that the joint distribution can be considered a product of the individual density functions. We continue the recursion, where x_t is recursively computed based on (15d), and for clarity define

$$\begin{aligned} &c(v_0, v_1, ..., v_{N-1}, \bar{U}) := \\ &\quad J\big(\mathcal{D}_+([\bar{x}_0, \mu(\bar{x}_0, \bar{u}_0) + \Sigma(\bar{x}_0, \bar{u}_0)^{\frac{1}{2}} v_0, \\ &\quad \mu(\bar{x}_1, \bar{u}_1) + \Sigma(\bar{x}_1, \bar{u}_1)^{\frac{1}{2}} v_1, ..., \\ &\quad \mu(\bar{x}_{N-1}, \bar{u}_{N-1}) + \Sigma(\bar{x}_{N-1}, \bar{u}_{N-1})^{\frac{1}{2}} v_{N-1}], \bar{U})\big). \end{aligned} \tag{15g}$$

We express our resulting integral more concisely now as

$$\int ... \int c(v_0, v_1, ..., v_{N-1}, \bar{U}) \Pi_{t=0}^{N-1} p(v_t) dv_0 ... dv_{N-1}, \tag{15h}$$

which we can approximate numerically as

$$\sum_{l=0}^{L} c(v_0^l, v_1^l, ..., v_{N-1}^l, \bar{U}). \tag{15i}$$

Under the assumption that $J(\cdot)$ and the GP moments (9) are differentiable with respect to $\bar{U}$, the Monte Carlo integrator is now differentiable with respect to $\bar{U}$:

$$\sum_{l=0}^{L} \frac{\partial}{\partial \bar{U}} c(v_0^l, v_1^l, ..., v_{N-1}^l, \bar{U}). \tag{15j}$$

Note that while the differentiability of $J(\cdot)$ depends on the particular objective function, the differentiability for the GP moments is met when using a differentiable kernel and mean function. ■

Algorithm 1 Experiment Design

Input $\bar{U}$ (initial), $\mathcal{D}, \bar{x}_0$
Output $\bar{U}$
function SAMPLEGRAD($\bar{U}, V, \mathcal{D}, x_0$)
  $(\bar{X}, \bar{U}, \bar{Y}) \leftarrow$ forward rollout from $(\bar{U}, V, \mathcal{D}, \bar{x}_0)$
  $\mathcal{D}_+ \leftarrow \mathcal{D} \cup (\bar{X}, \bar{U}, \bar{Y})$
  $\mu(z; \mathcal{D}_+), \Sigma(z; \mathcal{D}_+) \leftarrow$ update GP with $\mathcal{D}_+$
  $U^*(\bar{U}, V) \leftarrow$ Solve control with $\mu(z; \mathcal{D}_+), \Sigma(z; \mathcal{D}_+)$
  $\frac{\partial J}{\partial \bar{U}} \leftarrow$ Compute gradient at $U^*(\bar{U}, V)$
end function
while not converged **do**
  for i=1:batch size **do**
    $V \leftarrow [v_0, v_1, ..., v_{N-1}]$ from standard normal
    $\frac{\partial J(V)}{\partial \bar{U}} \leftarrow SAMPLEGRAD(\bar{U}, V, \mathcal{D}, \bar{x}_0)$
  end for
  $\bar{U} \leftarrow$ gradient step from batch, project onto $\mathcal{U}$
end while

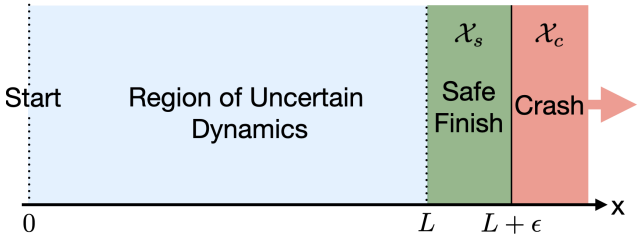

Fig. 1. Starting from the origin, the objective is to reach the green target set, $\mathcal{X}_s$, in minimum time. Overshooting the target set results in a crash in $\mathcal{X}_c$. The dynamics of the system are unknown and need to be learned.

IV. Minimum Expected Time

To demonstrate the approach laid out above, we consider a prototypical stochastic control problem with a risk-aware constraint. We start at the origin and want to reach a safe finishing set, $\mathcal{X}_s$, while avoiding an unsafe set, $\mathcal{X}_c$, that also leads to termination. In a one-dimensional setting we visualize this in Figure 1. Formally this corresponds to the following optimization:

$$\min_U \ \mathbb{E}[T \mid \mathcal{D}] \tag{16a}$$

$$s.t. \ \mathbb{E}\Big[\sum_{t=0}^{T} I(x_t) \mid x_0, \mathcal{D}\Big] \leq \Delta, \tag{16b}$$

$$u_t \in \mathcal{U}, \tag{16c}$$

$$x_0 = 0, \tag{16d}$$

where x_t is described by (1), Δ is our tolerance for failure and the constraint (16b) is a conservative bound on the probability of failure, $P(x_T \in \mathcal{X}_c) \leq \Delta$ [1]. Here $I(x)$ indicates a state in the unsafe set $\mathcal{X}_c$,such that

$$I(x) = \begin{cases} 1, & x \in \mathcal{X}_c, \\ 0, & o/w. \end{cases} \tag{17}$$

We define T, which is a random variable, as the first time for which the state enters either $\mathcal{X}_s$ or $\mathcal{X}_c$, unless the state never reaches either finished set, in which case $T = N$. We express this as

$$T := \min \ \{t : x_t > L\} \cup \{N\}. \tag{18}$$

Corollary 2: [1]. Suppose there exists a solution to the Bellman equation

$$J_t^{\lambda^*}(x_t) = \begin{cases} \min_u \ 1 + \mathbb{E}[J_{t+1}^{\lambda^*}(x_{t+1}) \mid x_t = x, \mathcal{D}], & x_t \notin \mathcal{X}_s \cup \mathcal{X}_c, \\ 0, & x_t \in \mathcal{X}_s, \\ \lambda^*, & x_t \in \mathcal{X}_c, \end{cases} \tag{19a}$$

with terminal cost-to-go given by

$$J_N^{\lambda^*}(x_N) = \begin{cases} \lambda^*, & x_N \notin \mathcal{X}_s, \\ 0, & o/w, \end{cases} \tag{19b}$$

for which trajectories under the policy

$$u_t = \pi(x_t) := \arg\min_u \ 1 + \mathbb{E}[J^{\lambda^*}_{t+1}(x_{t+1}) \mid x_t = x, \mathcal{D}] \tag{19c}$$

satisfy

$$\lambda^* \left(\mathbb{E}\left[\sum_{t=0}^{T} I(x_t) \mid x_0, \mathcal{D} \right] - \Delta \right) = 0, \tag{19d}$$

then (19c) solves the optimization (16).

Proof: For a dynamic programming (DP) problem with stage costs $g(x)$ and the risk bound (16b), we can form a Lagrangian [1]:

$$\mathcal{L}(x,u,\lambda,x_0,\mathcal{D}) = \mathbb{E}\left[\sum_{t=0}^{N} g_k(x_t) \mid \mathcal{D} \right] + \lambda \left(\mathbb{E}\left[\sum_{t=0}^{N} I_t(x_t) \mid x_0, \mathcal{D} \right] - \Delta \right) \tag{20a}$$

such that the primal problem is given by solving the Bellman equation

$$J^{\lambda}_t(x_t) = \min_u \ g(x_t) + \lambda I_t(x_t) + \mathbb{E}[J^{\lambda}_{t+1}(x_{t+1}) \mid x_t = x, \mathcal{D}] \tag{20b}$$

for a fixed λ and terminal cost

$$J^{\lambda}_N(x_N) = g_N(x_N) + \lambda I_N(x_N). \tag{20c}$$

From [1], we update λ based on a root-finding algorithm on the subgradient of the dual function $q(\lambda, x_0, \mathcal{D}) := \min_u \ \mathcal{L}(x,u,\lambda,x_0,\mathcal{D})$:

$$\frac{\partial q(\lambda, x_0, \mathcal{D})}{\partial \lambda} = \mathbb{E}\left[\sum_{t=0}^{N} I_t(x_t) \mid x_0, \mathcal{D} \right] - \Delta. \tag{20d}$$

When the subgradient is sufficiently close to zero, we have λ^*. The optimality conditions for this approach are in [1].

Our minimum time problem (16) is in discrete-time such that for a DP formulation our stage cost counts unit time for states not finished. By construction, once we enter a finished set we remain there for all future time. We model this using dynamics with $x_t \in \mathcal{X} = \mathbb{R}^{n_x} \cup \{done, bad\}$:

$$x_{t+1} := \begin{cases} h(x_t, u_t) + w_t, & x_t \in \mathbb{R}^{n_x}, \\ done, & x_t \in \mathcal{X}_s \cup \{done, bad\}, \\ bad, & x_t \in \mathcal{X}_c, \end{cases} \tag{20e}$$

such that *bad* corresponds to reaching $\mathcal{X}_c$ and *done* corresponds to having reached the safe terminal set $\mathcal{X}_s$ or *bad* previously. We then define our stage cost $g(x_t)$ to be

$$g(x_t) := \begin{cases} 1, & x_t \in \mathbb{R}^{n_x}, \\ 0, & o/w, \end{cases} \tag{20f}$$

and the cost-to-go as

$$J_t(x_t) = \begin{cases} \min_u \ \mathbb{E}[J^{\lambda^*}_{t+1}(x_{t+1}) \mid x_t = x, \mathcal{D}], & x_t \in \mathbb{R}^{n_x}, \\ 0, & x_t = done, \\ \lambda, & x_t = bad. \end{cases} \tag{20g}$$

This model satisfies the Lagrangian approach above such that we directly apply it. Additionally, the model based on (20e) to 20g) can be more compactly expressed since for $x \in \mathcal{X}_s$, the cost-to-go is zero; for $x \in \mathcal{X}_c$, the cost-to-go is λ; and for $x \notin \mathcal{X}_s \cup \mathcal{X}_c$, the cost-to-go is

$$J^{\lambda}_t(x_t) = \min_u \ 1 + \mathbb{E}[J^{\lambda}_{t+1}(x_{t+1}) \mid x_t = x, \mathcal{D}] \tag{20h}$$

with

$$J^{\lambda}_N(x_N) = \begin{cases} \lambda, & x_N \notin \mathcal{X}_s, \\ 0, & o/w, \end{cases} \tag{20i}$$

giving us the compact expression in (19a,19b). The expected cost-to-go is with respect to the conditional distribution (9) from Corollary 1:

$$\mathbb{E}[J_{t+1}(x_{t+1}) \mid x_t, \mathcal{D}] = \int J_{t+1}(x_{t+1}) p(x_{t+1} \mid x_t = x, u), \mathcal{D}) dx. \tag{20j}$$

■

V. Numerical Results

We present numerical simulations for the minimum expected time problem. To show the benefit of optimal experiment design, we compare the performance improvement of conducting an experiment using our method versus other intuitive methods.

A. Process model

Consider a system modeled by (1) with the initially unknown scalar function

$$h(x,u) = x + \left(1 + 3\tilde{I}_{[0.75,1.4]}(x)\right)u \tag{21}$$

where $\tilde{I}(\cdot)$ is the indicator function approximated by product of arctangents with finite slope. This can be visualized in the dark blue trace in Figure 2. We set the process noise to $\Sigma_w = 0.01^2$.

For the control problem in (16), we start at $x_0 = 0$ and try to finish in the set $\mathcal{X}_s = [L, L+\epsilon] = [1, 1.05]$ (Figure 1). Overshooting $\mathcal{X}_s$ enters the unsafe set $\mathcal{X}_c = (1.05, \infty)$. Our tolerance for failure in (16b) is set to $\Delta = 25\%$. For both the control problem and experiment design, we restrict the control input to $[-0.1, 0.1]$. The max allowable time, N, is 15 time steps for both the control and experiment design problems.

B. Benchmarks

We start with a dataset, $\mathcal{D}$, consisting of a fixed number of random walk trials with positive mean. We then create $\mathcal{D}_+$ by augmenting $\mathcal{D}$ with an additional trajectory using our method (Algorithm 1) referred to as $\bar{U}$ or one of the benchmarks described below. Based on $\mathcal{D}_+$ we compute the new optimal control (16) and evaluate the control performance. In our comparisons, we consider the bounding performance value given perfect knowledge of the process, $h(\cdot)$. In terms of expected time and risk, this is the ideal performance and is referred to as "perfect info" in plots. For the benchmarks, we use the following alternatives to our experiment design:

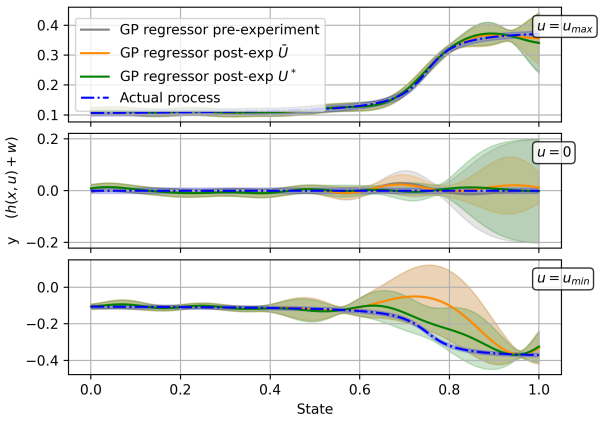

Fig. 2. We visualize our GP approximation of the process mean and variance by taking slices of the input at three different levels, $u_{max} = 0.1, 0, u_{min} = -0.1$, and sweep the state. In blue, we observe the actual process with relatively small 1-σ bounds from the process noise. The pre-experiment GP's dataset has 25 random trials, while the post-experiment GPs' datasets contain the original 25 plus the respective experiment trial. For u_{max}, we observe the GP models have relatively low epistemic uncertainty (variance), which is reasonable given the random walks have positive mean. Our experiment input, $\bar{U}$ reduces the uncertainty for zero-input significantly relative to the pre-experiment or U^*. The closed-loop, $U^*(x)$ is left off here for legibility. While U^* appears to generate a model with less uncertainty for u_{min}, this is less relevant since the optimal control input does not need to be negative in this setup.

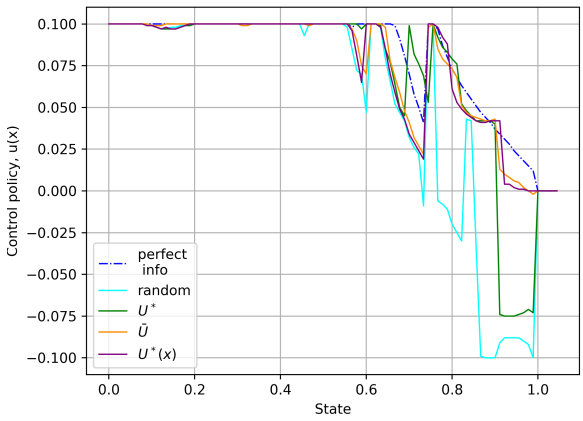

Fig. 3. We illustrate the optimal policies post-experiment. Since we have slightly different policies depending on the experiment outcome, we illustrate the policy closest to the median for our experiment design and the corresponding outcome for the benchmarks. We observe that the policy given perfect information is to apply the max input, 0.1, until the state reaches 0.6, then decrease until around 0.75, and then decrease the input to zero after just before 0.8. In this example, our experiment design $\bar{U}$ tends to approach the optimal without exceeding it. In some areas the benchmarks are closer to the perfect information policy but they also exceed it, which leads to riskier behavior.

- *random*: add another random walk trajectory to the set $\mathcal{D}$.
- U^*: compute an open-loop control based on the optimal control (16) given $\mathcal{D}$ for the most-likely state sequence sampled from our current GP. This open-loop control also serves as the initial value for Algorithm 1.
- $U^*(x)$: use the optimal state feedback policy from (16) given $\mathcal{D}$. While we consider open-loop input sequences for our experiment design, including the relative performance of a closed-loop policy shows the value of feedback in this stochastic setting.

Each experiment trajectory is run for the duration of the experiment time and does not stop upon reaching either of the target sets. This can lead to shorter trajectories if the state reaches the finished set in less than 15 steps for the optimal control benchmarks. However, for the small dataset scenarios that we explore in the coming section, the optimal control tends to use the full-time since it cannot guarantee constraint satisfaction.

C. Computations

For efficient programming, we used JAX [19] and GPJAX [20] in Python such that the stochastic DP can be just-in-time compiled and we can simultaneously use automatic differentiation in (14a). We used 1000 stochastic gradient descent steps instead of waiting until convergence, with a batch size of 80 at each step on a dual core NVIDIA GeForce RTX 3080 GPU.

D. Discussion

Figure 2 helps build intuition about the performance of our method relative to the benchmarks. This figure depicts the uncertainty of three different GP models: the pre-experiment GP has access to 25 trials of data, the GP corresponding to $\bar{U}$ contains the same 25 trials plus our well-designed experiment, and the GP for U^* contains the 25 initial trials plus the trial from applying U^*. We leave off the closed-loop comparison to avoid cluttering the plot. We observe that our method reduces epistemic uncertainty mainly for zero-input near the finishing set. Figure 3 illustrates that given the post-experiment datasets, our experiment design leads to a control policy that approaches the optimal policy given perfect information. The optimal control with perfect information decreases to zero near the target set, which indicates the observed reduction of epistemic uncertainty at zero in Figure 2 by our method is useful for improving control performance. Unlike the benchmarks, our method does not exceed the optimal policy given perfect information, doing which leads to higher risk. This qualitative analysis indicates that our experiment design, $\bar{U}$, leads to learning in areas that are important for control performance.

We quantify the experiment design performance using two metrics: the fraction of controllers (16) that can satisfy the chance-constraint (16b), and from the subset of controllers that are feasible, the expected time to finish. In Figure 4, we show the trends for the two metrics across dataset size. In this setting, 40 random trials are drawn and each dataset of size n is a superset of dataset size m for $n > m$ (i.e. the 10 trial dataset includes the 5 trial dataset plus 5 additional trials). In the first subplot of Figure 4 we observe

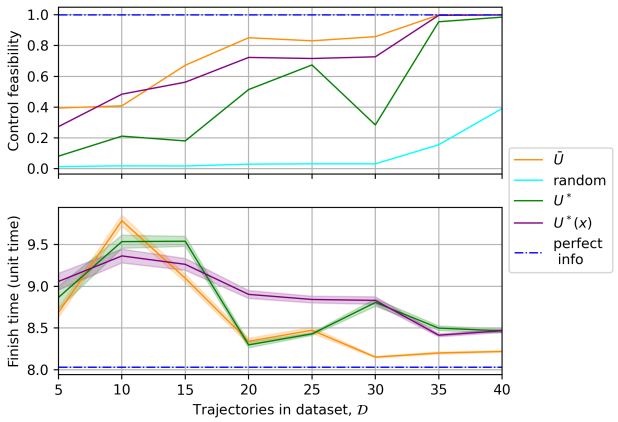

Fig. 4. We visualize the benefit of adding a designed experiment to a dataset of random trials (e.g. 5 random trials plus one experiment) and test the control performance with this augmented dataset. Control feasibility indicates the fraction of controllers that satisfy the chance constraint over 1000 experiment outcomes. Of the controllers that satisfy the constraint, we can compute the expected finish time and 95 percent confidence intervals. Our experiment design, $\bar{U}$, outperforms all other benchmarks for control feasibility except for $U^*(x)$ for the dataset of size 10. Finishing time performance becomes more relevant as the feasibility metric nears 100 percent whereupon we see our time decreasing significantly relative to the others. Due to low control feasibility, we omit the finish time for the random case.

the fraction of controllers that satisfy our risk tolerance level, $\Delta = 25\%$, where the data is from 1000 possible experiment outcomes for each experiment design method. This shows that our method in general generates a controller that satisfies the constraint more than any of the benchmarks. While we significantly outperform the open-loop optimal control, we also do better than the closed-loop optimal control, which has the advantage over all other methods of having access to state feedback. One takeaway from including the closed-loop optimal control is that closed-loop experiment design methods can increase performance significantly. Lastly, while we might expect monotonic improvement with respect to dataset size across all methods, this is not the case because we are dealing with a particular set of 40 random trials.

The second subplot in Figure 4 illustrates the expected time for feasible controllers. For the smaller datasets, the significance of this metric is much secondary to the control feasibility because most controllers do not satisfy the chance constraint. We omit the finish time for "random" since the control feasibility is very low for all datasets. It is worth noting that besides the dataset of size 10, our method is comparable in expected time or better than the benchmarks. For larger datasets, we observe that as the control feasibility approaches 100 percent, our expected time decreases significantly relative to the others as desired.

VI. Conclusion

We present a general approach to experiment design that improves data-driven control performance as much as possible when the dynamics are initially unknown. We focus on using Gaussian processes for inference and derive a tractable formulation for our experiment design optimization. We consider a minimum expected time problem with chance-constraints and numerically demonstrate that our experiment design method outperforms suitable benchmarks. While the minimum expected time control problem was shown in a particular setting, this demonstrates an instance of our broader experiment design approach.

This work scratches the surface for experiment design for stochastic optimal control problems where the goal of the design is to improve the control performance. Finding efficient relaxations that reduce the need to recompute an optimal solution for each sample of the gradient can improve the complexity of this approach significantly. For Gaussian process regressors, efficient augmentation of the data matrix will enable faster computations such as via rank-n updates for Cholesky decompositions. Additionally, preliminary testing indicates that designing an experiment policy will allow for greater performance than the current open-loop methodology. Finally, adding inequality constraints to the experiment design problem is critical for real-world applications.

References

[1] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, "Chance-constrained dynamic programming with application to risk-aware robotic space exploration," *Autonomous Robots*, vol. 39, no. 4, pp. 555–571, Dec. 2015. [Online]. Available: http://link.springer.com/10.1007/s10514-015-9467-7

[2] J. A. Paulson, E. A. Buehler, R. D. Braatz, and A. Mesbah, "Stochastic model predictive control with joint chance constraints," *International Journal of Control*, vol. 93, no. 1, pp. 126–139, Jan. 2020. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207179.2017.1323351

[3] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control using Gaussian Process Regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020, arXiv: 1705.10702. [Online]. Available: http://arxiv.org/abs/1705.10702

[4] C. S. Vallon and F. Borrelli, "Data-Driven Strategies for Hierarchical Predictive Control in Unknown Environments," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1434–1445, Jul. 2022, conference Name: IEEE Transactions on Automation Science and Engineering.

[5] A. Jain, T. Nghiem, M. Morari, and R. Mangharam, "Learning and Control Using Gaussian Processes," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. Porto: IEEE, Apr. 2018, pp. 140–149. [Online]. Available: https://ieeexplore.ieee.org/document/8443729/

[6] V.-A. Le and T. X. Nghiem, "A Receding Horizon Approach for Simultaneous Active Learning and Control using Gaussian Processes," *arXiv:2101.10351 [cs, eess]*, May 2021, arXiv: 2101.10351. [Online]. Available: http://arxiv.org/abs/2101.10351

[7] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively Learning Gaussian Process Dynamics," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. PMLR, Jul. 2020, pp. 5–15, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v120/buisson-fenet20a.html

[8] A. Mesbah, "Stochastic model predictive control with active uncertainty learning: A Survey on dual control," *Annual Reviews in Control*, vol. 45, pp. 107–117, 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1367578817301232

[9] A. Boukouvalas, D. Cornford, and M. Stehlık, "Approximately Optimal Experimental Design for Heteroscedastic Gaussian Process Models," p. 14.

[10] B. P. Weaver, B. J. Williams, C. M. Anderson-Cook, and D. M. Higdon, "Computational Enhancements to Bayesian Design of Experiments Using Gaussian Processes," *Bayesian Analysis*, vol. 11, no. 1,

pp. 191–213, Mar. 2016, publisher: International Society for Bayesian Analysis.

[11] G. M. Hoffmann and C. J. Tomlin, "Mobile Sensor Network Control Using Mutual Information Methods and Particle Filters," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 32–47, Jan. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5350445/

[12] "Near-optimal sensor placements in Gaussian processes | Proceedings of the 22nd international conference on Machine learning." [Online]. Available: https://dl.acm.org/doi/10.1145/1102351.1102385

[13] A. Fiorino, O. Neopane, and A. Singh, "Gaussian Processes for Episodic Experimental Design," p. 7.

[14] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian Active Learning for Classification and Preference Learning," *arXiv:1112.5745 [cs, stat]*, Dec. 2011, arXiv: 1112.5745. [Online]. Available: http://arxiv.org/abs/1112.5745

[15] A. McHutchon, "Nonlinear Modelling and Control using Gaussian Processes," p. 207.

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, ser. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006, oCLC: ocm61285753.

[17] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2022, arXiv:1312.6114 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1312.6114

[18] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[19] "JAX: Autograd and XLA," Feb. 2023, original-date: 2018-10-25T21:25:02Z. [Online]. Available: https://github.com/google/jax

[20] T. Pinder and D. Dodd, "GPJax: A Gaussian Process Framework in JAX," *Journal of Open Source Software*, vol. 7, no. 75, p. 4455, Jul. 2022. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.04455