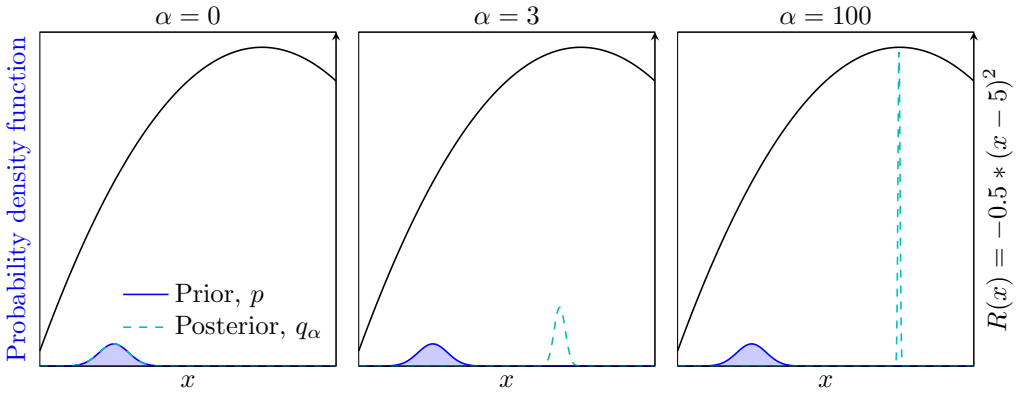

Figure 5: Visualization of Example 1 for three values of α, showing interpolation between no update and towards the Newton update.

A Extended Discussion Points and Results

This section discusses specific details not covered at length in the main section.

Posterior policy iteration as optimization. For clarity, we can ground inference-based optimization clearly by considering Gaussian inference and quadratic optimization (Example 1).

Example 1. *(Newton method via Gaussian inference). Consider minimizing a function $f(\boldsymbol{x})$ from a Gaussian prior $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Considering a second-order Taylor expansion about the prior mean, $f(\boldsymbol{x}) \approx f(\boldsymbol{\mu}_i) + \nabla_{\boldsymbol{x}} f(\boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^\top \nabla_{\boldsymbol{xx}} f(\boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)$, the Gaussian pseudo-posterior $q_\alpha(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_{i+1}, \boldsymbol{\Sigma}_{i+1}) \propto \exp(-\alpha f(\boldsymbol{x}))\, p(\boldsymbol{x})$ is*

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i - \alpha \boldsymbol{\Sigma}_{i+1} \nabla_{\boldsymbol{x}} f(\boldsymbol{\mu}_i), \;\; \boldsymbol{\Sigma}_{i+1} = (\boldsymbol{\Sigma}_i + \alpha \nabla^2_{\boldsymbol{xx}} f(\boldsymbol{\mu}_i))^{-1}.$$

As $\alpha \to 0$, $q_\alpha \to p$, while as $\alpha \to \infty$, the mean update tends to the Newton step,

$$\boldsymbol{\mu}_{i+1} \to \boldsymbol{\mu}_i - \nabla^2_{\boldsymbol{xx}} f(\boldsymbol{\mu}_i)^{-1} \nabla_{\boldsymbol{x}} f(\boldsymbol{\mu}_i).$$

Therefore, α acts as a learning rate and regularizer, with repeated inference performing regularized Newton decent. This regularization is beneficial for non-convex optimization, as the Hessian may be negative definite.

Minimum relative entropy problems. To motivate REPS as a form of constrained Gibbs posterior, we reverse the objective and constraint, which yields the minimum relative entropy problem (Lemma 1). The posterior of this problem is the same as REPS and the temperature is also a Lagrangian multiplier, but the constraint is now defined by the expectation value rather than the KL divergence.

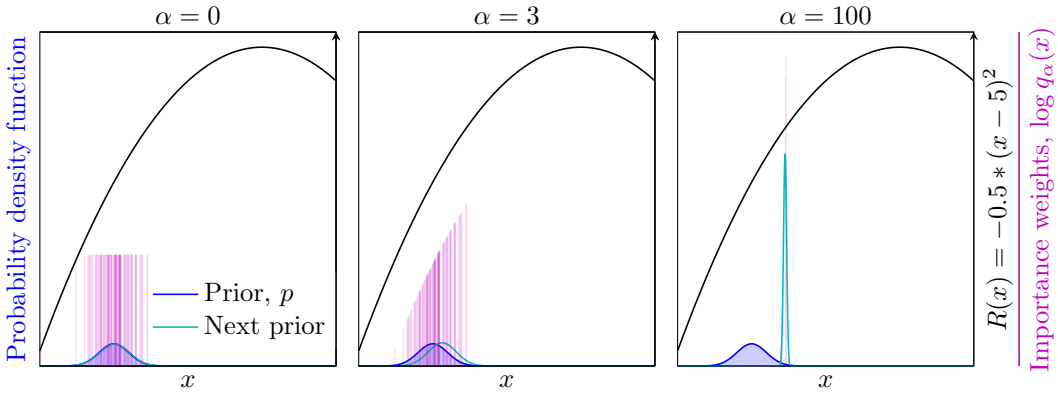

Figure 6: Visualization of Example 1 for three values of α, using self-normalized importance sampling. High values of α lead to the posterior collapsing around a few samples. Moreover, compared to the Gaussian case, the update is limited to the support of the samples.

Lemma 1. *(Minimum relative entropy [48]) Find q that minimizes the relative entropy from p, given a function $\boldsymbol{f}$ that describes random variable $\boldsymbol{x}$, $\min_q \mathbb{D}_{\mathrm{KL}}[q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x})]$ s.t. $\mathbb{E}_{\boldsymbol{x}\sim q(\cdot)}[\boldsymbol{f}(\boldsymbol{x})] = \hat{\boldsymbol{f}}$. The solution to this problem is $q(\boldsymbol{x}) \propto \exp(-\boldsymbol{\lambda}^\top \boldsymbol{f}(\boldsymbol{x}))p(\boldsymbol{x})$, where $\boldsymbol{\lambda}$ is the Lagrange multiplier required to satisfy the constraint. When $p(\boldsymbol{x})$ is uniform, $q(\boldsymbol{x})$ is a maximum entropy distribution.*

Interestingly, in maximum entropy problems, function $\boldsymbol{f}$ defines the sufficient statistics of the random variable. For our setting, $\boldsymbol{f}$ is based on the reward function (episodic return), which indicates the reward objective is used to 'summarize' the random policy samples.

Rényi-2 divergence and the effective sample size. The LBPS objective uses the effective sample size as a practical estimator for the exponentiated Rényi-2 divergence (Lemma 2).

Lemma 2. *(Exponentiated Rényi-2 divergence estimator [51]) The effective sample size diagnostic ($\hat{N}$) is an estimate of the exponentiated Rényi-2 divergence for finite samples, as $d_2(p||q)/N \approx \hat{N}^{-1}$,*

$$\lim_{N\to\infty} \frac{1}{N} \frac{(\sum_n w_{q/p}(\boldsymbol{x}_n))^2}{\sum_n w_{q/p}(\boldsymbol{x}_n)^2} = \int \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} = \exp \mathbb{D}_2(p \mid\mid q)^{-1} = d_2(p \mid\mid q)^{-1}.$$

This result connects divergence-based stochastic optimization methods, such as REPS, to 'elite'-based stochastic solvers such as CEM, as the number of elites is equal to the effective sample size of the weights, as discussed below.

Time Shifting with Gaussian Process We derive the multivariate normal posterior shifting update introduced in Section 4, that is facilitated by continuous-time Gaussian process priors.

Proposition 1. *Given a Gaussian process prior $\mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\Sigma}(t))$ and multivariate normal posterior $q_\alpha(\boldsymbol{a}_{t_1:t_2}) = \mathcal{N}(\boldsymbol{\mu}_{t_1:t_2|R}, \boldsymbol{\Sigma}_{t_1:t_2|R})$ for t_1 to t_2, the posterior for t_3 to t_4 is expressed as*

$$\boldsymbol{\mu}_{t_3:t_4|R} = \boldsymbol{\mu}_{t_3:t_4} + \boldsymbol{\Sigma}_{t_3:t_4,t_1:t_2}\boldsymbol{\nu}_{t_1:t_2}, \quad \boldsymbol{\Sigma}_{t_3:t_4|R} = \boldsymbol{\Sigma}_{t_3:t_4} - \boldsymbol{\Sigma}_{t_3:t_4,t_1:t_2}\boldsymbol{\Lambda}_{t_1:t_2}\boldsymbol{\Sigma}^\top_{t_3:t_4,t_1:t_2}, \quad (7)$$

where $\boldsymbol{\nu}_{t_1:t_2} = \boldsymbol{\Sigma}^{-1}_{t_1:t_2}(\boldsymbol{\mu}_{t_1:t_2|R} - \boldsymbol{\mu}_{t_1:t_2})$ and $\boldsymbol{\Lambda}_{t_1:t_2} = \boldsymbol{\Sigma}^{-1}_{t_1:t_2}(\boldsymbol{\Sigma}_{t_1:t_2} - \boldsymbol{\Sigma}_{t_1:t_2|R})\boldsymbol{\Sigma}^{-1}_{t_1:t_2}$.

Proof. For this update, we require the Gaussian likelihood $\mathcal{N}(\boldsymbol{\mu}_{R|t_1:t_2}, \boldsymbol{\Sigma}_{R|t_1:t_2})$ that achieves the posterior $q_\alpha(\boldsymbol{a}_{t_1:t_2})$ given the prior $p(\boldsymbol{a}_{t_1:t_2})$. Using $\boldsymbol{K}_{t_1:t_2} = \boldsymbol{\Sigma}_{t_1:t_2}\boldsymbol{\Sigma}^{-1}_{R|t_1:t_2}$, we write the Gaussian posterior update in the Kalman filter form

$$\begin{aligned}
\boldsymbol{\mu}_{t_1:t_2|R} &= \boldsymbol{\mu}_{t_1:t_2} + \boldsymbol{K}_{t_1:t_2}(\boldsymbol{\mu}_{R|t_1:t_2} - \boldsymbol{\mu}_{t_1:t_2}), \\
\boldsymbol{\Sigma}_{t_1:t_2|R} &= \boldsymbol{\Sigma}_{t_1:t_2} - \boldsymbol{K}_{t_1:t_2}\boldsymbol{\Sigma}_{R|t_1:t_2}\boldsymbol{K}^\top_{t_1:t_2}.
\end{aligned}$$

We introduce $\boldsymbol{\nu}$ and $\boldsymbol{\Lambda}$ to capture the unknown terms involving $\boldsymbol{a}_{R|t_1:t_2}$ w.r.t. $\boldsymbol{a}_{t_1:t_2}$ and $\boldsymbol{a}_{t_1:t_2|R}$,

$$\begin{aligned}
\boldsymbol{\nu}_{t_1:t_2} &= \boldsymbol{\Sigma}^{-1}_{R|t_1:t_2}(\boldsymbol{\mu}_{R|t_1:t_2} - \boldsymbol{\mu}_{t_1:t_2}) = \boldsymbol{\Sigma}^{-1}_{t_1:t_2}(\boldsymbol{\mu}_{t_1:t_2|R} - \boldsymbol{\mu}_{t_1:t_2}), \\
\boldsymbol{\Lambda}_{t_1:t_2} &= \boldsymbol{\Sigma}^{-1}_{R|t_1:t_2} = \boldsymbol{\Sigma}^{-1}_{t_1:t_2}(\boldsymbol{\Sigma}_{t_1:t_2} - \boldsymbol{\Sigma}_{t_1:t_2|R})\boldsymbol{\Sigma}^{-1}_{t_1:t_2}.
\end{aligned}$$

When extrapolating to the new time sequence, we use the Kalman posterior update again, but with the prior for the new time sequence. This step computes the joint over old and new timesteps, conditions on the objective (i.e. Bayes' rule), and marginalizes out the old timesteps (see Equation 6), so

$$\begin{aligned}
\boldsymbol{\mu}_{t_3:t_4|R} &= \boldsymbol{\mu}_{t_3:t_4} + \boldsymbol{\Sigma}_{t_3:t_4,t_1:t_2}\boldsymbol{\nu}_{t_1:t_2}, \\
\boldsymbol{\Sigma}_{t_3:t_4|R} &= \boldsymbol{\Sigma}_{t_3:t_4} - \boldsymbol{\Sigma}_{t_3:t_4,t_1:t_2}\boldsymbol{\Lambda}_{t_1:t_2}\boldsymbol{\Sigma}^\top_{t_3:t_4,t_1:t_2}.
\end{aligned}$$

□

Using the geometric interpretation of the Kalman filter [83], this update can be seen as projecting the solution from the previous sequence into the new time sequence, given the correlation structure defined by the GP prior.

Augmented Cost Design versus Prior Design. Prior work enforces smoothness using an augmented reward function term, e.g. $-\lambda_s||\boldsymbol{a}_t - \boldsymbol{a}_{t+1}||^2$ per timestep [68]. For PPI, the PPI objective also augments the reward objective with a KL term (Equation 3). For the Monte Carlo setting here, this KL term is $\frac{1}{\alpha}\sum_n w^{(n)}(\log w^{(n)} - \log p(\boldsymbol{A}^{(n)}))$, where $w^{(n)} \propto \exp(\alpha R^{(n)})$. When $p(\boldsymbol{A})$ is a

Gaussian process, the log probability has the familiar quadratic form, but this time applies to the whole sequence $\boldsymbol{A}$, due to the episodic setting. Independent white noise only applies a quadratic penalty per action, due to its factorized covariance. A first-order Markovian GP would regularize one-step correlations, as its inverse covariance is banded [84]. The squared exponential kernel, used in this work, has infinite order [33] and a dense inverse covariance matrix and therefore regularizes the whole sequence, which is why it was chosen to achieve smoothness.

Episodic or sequential inference? For the Monte Carlo setting, sequential Monte Carlo (SMC) [85] is an obvious choice for inference given its connection to Bayesian smoothing methods. However, we found that an SMC approach provided limited benefits for control. The variance reduction when resampling decreases exploration from the optimization viewpoint.

Moreover, SMC smoothing reweighs the particles sampled in the forward pass, with $\mathcal{O}(N^2T)$ complexity for N particles [85]. Therefore, no exploration occurs during the backward pass, in contrast to Gaussian message passing where smoothing performs Riccati equation updates on the state-action distribution. When considering just sequential importance sampling for trajectories, we arrive at the episodic case, as $w_T^{(n)} = \exp\left(\alpha r_T^{(n)}\right) w_{T-1}^{(n)} = \exp\left(\alpha \sum_{\tau=0}^{T} r_\tau^{(n)}\right)$.

Connection to rare event simulation and maximum estimators. The cross entropy method is a popular and effective Monte Carlo optimizer [19]. It broadly works by moment matching an exponential family distribution onto 'elite' samples. The elite samples are chosen according to $\mathbb{P}_{q(\boldsymbol{x})}(f(\boldsymbol{x}) > f^*)$, a rare event simulator, where improving upon f^* is treated as the rare event. In practice, elites are chosen by sorting the top k samples according to the objective. Our lower-bound reflects CEM in two ways. One, the rare event inequality reflects the expected return bound used in the LBPS objective. Secondly, the k in the top-k estimator matches the effective sample size, used in LBPS and ESSPS. One can view CEM and IS approaches through their maximum estimators

$$\text{Max}_k[\boldsymbol{R}] = \frac{\sum_{n=1}^{k} R_n}{\sum_{n=1}^{k} 1}, \quad \text{Max}_\alpha[\boldsymbol{R}] = \frac{\sum_{n=1}^{N} \exp(\alpha R_n) R_n}{\sum_{n=1}^{k} \exp(\alpha R_n)}.$$

We can see that the SNIS expectation is equivalent to the Boltzmann softmax, used in reinforcement learning [86]. Using k or α this estimator is bounded between the true maximum over the samples and the mean. Therefore the estimators behave in a similar fashion when α is chosen such that the ESS is k. The key distinction is the sparse and uniform weights of CEM vs. the posterior weights of PPI methods, which seems to have an equal or more important influence on optimization, based on the black-box optimization results.

Connection to mirror descent. Functional mirror descent for online learning optimizes a distribution q and introduces a Bregman divergence $\mathbb{D}$ for information-geometric regulation, i.e. $q_{i+1} = \arg\min_{q \in \mathbb{Q}} \mathbb{E}_{\boldsymbol{x} \sim q(\cdot)}[f(\boldsymbol{x})] + \alpha_i \mathbb{D}[q \mid\mid q_i]$ [87]. The temperature typically follows a predefined schedule that comes with convergence guarantees, e.g. $\alpha_i \propto \sqrt{i}$, such that $\alpha \to \infty$ as $i \to \infty$ [88]. Considering Equation 3, the DMD view applies to our setting when using KL regularization, so DMD view opens up alternative Bregman divergences for regularization.

B Extended Related Work

The methods presented in the main section have been investigated extensively in the prior literature. This section provides a more in-depth discussion of the differences in approach and implementation.

Temperature tuning approaches. Across control-as-inference methods, the temperature value plays an important role. By the construction of the policy updates, mis-specification of the temperature leads overly greedy or conservative optimization. Table 1 provides a summary. A popular and often effective approach is to tune a fixed temperature, as done in AICO [23] and MPPI [18]. However, the optimization behavior depends the values of the objective, and nonlinear optimization methods such as Levenberg–Marquardt and mirror descent suggest an adaptive regularization scheme may perform better. Reward-weighted regression (RWR) [22], and later input inference for control (I2C) [27], take a completely probabilistic view of the Gibbs likelihood and used a closed-form update to optimize the temperature using expectation maximization. However, this update is motivated by the probabilistic interpretation of the optimization, not the optimization itself, so there is no guarantee this approach improves optimization, beyond the attractive closed-form update. Follow-up work regulated optimization by normalizing the returns, using the range or standard deviation [64, 56].

Table 1: A review of different adaptive temperature strategies across episodic control-as-inference methods and settings. We propose LBPS and ESSPS as optimization-driven methods with intuitive hyperparameters.

Algorithm	Method	Description
MPPI [18], AICO [23]	Constant	α
PI^2 [64], POWER [56]	Normalization	$\bar{\alpha}/(\max[\boldsymbol{R}] - \min[\boldsymbol{R}]), \quad \bar{\alpha}/\sigma_R$
RWR [22], I2C [37]	EM	$\alpha_{i+1} = \sum_n \exp\left(\alpha_i R_i^{(n)}\right) / \sum_n R_i^{(n)} \exp\left(\alpha_i R_i^{(n)}\right)$
REPS [47], MORE [17]	KL bound	$\arg\min_\alpha \epsilon/\alpha + \frac{1}{\alpha} \log \sum_n \exp(\alpha R^{(n)})$
LBPS (this work)	IS lower-bound	$\arg\max_\alpha \mathbb{E}_{q_\alpha}[R] - \mathcal{E}_R(\delta, \hat{N}_\alpha)$
ESSPS (this work)	ESS	$\arg\min_\alpha \lvert\hat{N}_\alpha - N^*\rvert$

While this resolved the objective dependency, it is a heuristic and not interpretable from the optimization perspective. Later, REPS framed the update as a constrained optimization problem, subject to a KL constraint [47, 9]. The temperature was them obtained by minimizing the Lagrangian dual function. However, in practice this dual is approximated using Monte Carlo integration which limits the constraint accuracy. Moreover, a hard KL constraint now requires a distribution-dependent hyperparameter. This work introduces LBPS and ESSPS. LBPS optimizes a lower-bound of the importance-sampled expected return, with the hyperparameter controlling greediness via the confidence of the concentration inequality and therefore is objective- and distribution independent. ESSPS bridges LBPS and CEM, optimizing for a desired effective sample size as an analogy to elite samples. Finally, self-paced contextual episodic policy search has previously adopted the minimum KL form of the REPS optimization problem, in order to assess the expected performance per context task [78].

Gaussian message passing methods. Using Bayesian smoothing of the state-action distribution, trajectory optimization can be performed by treating a step-based objective analogously to the observation log likelihood in state estimation [23, 27]. While the control prior defined as temporally independent in prior work, the smoothing of the state-action distribution imbues an inherent smoothness to the solution (e.g. Figure 6, [27]). Approximate inference can be performed using linearization or quadrature, which, while effective are less amenable to parallelization and therefore do not scale so gracefully to high-dimensional state and action spaces [27]. Moreover, these methods require access to the objective function for linearization, rather than just rollout evaluations.

Mirror descent model predictive control. Mirror descent, used for proximal optimization, introducing information geometric regularization through a chosen Bregman divergence $\mathbb{D}_\Psi$. *Dynamic* mirror descent (DMD), motivates an autonomous update Φ to the optimization variable to improve performance for online learning. DMD-MPC [46] incorporates this into an MPC scheme by incorporating an explicit time shift operator Φ, so $\boldsymbol{\theta}_{t+1} = \Phi(\boldsymbol{\theta})$, and $\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}\in\Theta} \nabla l_t(\boldsymbol{\theta}_t)^\top \boldsymbol{\theta} + \mathbb{D}_\psi(\boldsymbol{\theta}||\boldsymbol{\theta}_t)$. Unlike mirror descent, DMD-MPC does not consider an adaptive temperature strategy. Moreover, it introduces CEM- and MPPI- like updates by explicitly transforming the objective and preforming gradient descent, rather than estimating the posterior.

Path integral control. Path integral theory connects Monte Carlo esimations to partial differential equation (PDE) solutions [28]. Using this, path integral control is used to motivate sample-approximation to the continuous-time Hamiltonian-Jacobian PDE. Crucially, the exponential transform is introduced to the value function term to make the PDE linear, a requirement to path integral theory. Path integral theory is limited to estimating a optimal action trajectory, given an initial state, and requires several additional assumptions on the dynamics and disturbance noise [64]. Later, a divergence-minimization perspective was applied to path integral methods, to produce algorithms for more general settings and discrete time, referred to both as (information theoretic) IT-MPC [10] and also MPPI [89]. This alternative view is closer to PPI and no longer contains the key assumptions that apply to path integral theory.

Variational Inference MPC Variational MPC [5] is the same posterior policy iteration scheme outlined in the main text. The key difference is a posterior entropy bonus, like MORE, added to the objective

$$\min_q \mathbb{E}_q[R] + \frac{1}{\lambda_1}\mathbb{D}_{\mathrm{KL}}[q \mid\mid p] + \frac{1}{\lambda_2}\mathbb{H}[q].$$

This results, once reparameterized, in the usual posterior update with an annealing coefficient κ on the prior

$$w_n \propto \exp(\alpha r_n)\, p(\boldsymbol{a}_n)^{-\kappa}.$$

Due to hyperparameter sensitivity, this entropy regularization required a normalization step itself

$$p(\boldsymbol{a})^{-\kappa} = \exp\left(-\kappa \log p(\boldsymbol{a})\right) \rightarrow \exp\left(-\bar{\kappa}\frac{\log p - \max \log p}{\min \log p - \max \log p}\right). \tag{8}$$

As a result, the entropy regularization is dynamic in practice. The mechanism of this update is to increase the weight of low-likelihood samples such that the entropy of the posterior increases. In the ablation study, the effect of this regularization was relatively small with limited statistical significance. The work also uses a Gaussian mixture model as the variational family for multi-modal action sequences. As an MPPI baseline, the authors use PI^2 for MPC with an adaptive temperature through normalization with $\bar{\alpha}$ set to 10.

Variational Inference MPC using Tsallis Divergence Adopting the 'generalized' variational inference approach, Wang et al. replace the KL divergence with the Tsallis divergence [66], resulting in discontinuous, rather than exponential, expression for the posterior weights

$$w_n \propto \max(0, 1 + (\gamma - 1)\, \alpha\, r_n)^{\frac{1}{\gamma - 1}}, \quad \gamma > 0. \tag{9}$$

In practice, like in the CEM, k elites are used to define the r_* threshold

$$w_n \propto \begin{cases} \exp\left(\frac{1}{\gamma - 1} \log\left(1 - \frac{r_n}{r_*}\right)\right) & \text{if } r_n < r_*, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The resulting parameterization, essentially combines the CEM- and importance sampling approaches, using elites but weighing them through γ. For the implementation, the number of elites and γ were tuned but kept constant during MPC. From the perspective outlined in this work, this approach limits the effective sample size at a maximum value, but allows it to drop if there is sufficient range in the elite returns. As a result, its optimization is greedier than CEM, resulting in better downstream control performance. In the paper, the authors compare to MPPI with a static temperature and do not consider adaptive schemes. In theory, the Tsallis divergence could be combined with LBPS to optimize γ adaptively.

Stein Variational Inference Model Predictive Control Stein variational gradient descent (SVGD) is an approximate inference method for learning multi-modal posteriors parameterized by particles and a kernel function, $q(\cdot) = \sum_n k(\cdot, \boldsymbol{x}_n)$. The key quality of SVGD is the kernel, which provides a repulsion force during learning to encourage particle diversity.

For control, SVGD can be used to infer multi-modal action sequence which exhibit high entropy for exploration, as done in SV-MPC [65]. Due to the limitation of kernel methods with high-dimensional inputs, the kernel is designed to be factorized in time and action. The kernel design is also Markovian, so it considers the temporally adjacent variables as well, encouraging smoothness. This design decomposes the kernel into the sum of $H(H-1)d_a$ one dimensional kernels for H timesteps. Moreover, extrapolation into the future is realized by copying the last timestep (i.e. a zero-order hold). SVGD requires gradients of the loglikelihood, using backpropagation through the dynamics or Monte Carlo estimates. SV-MPC has an additional learning rate hyperparameter and uses an independent Gaussian action prior across timestemps.

Algorithm 1 Open-loop Episodic Monte Carlo Posterior Policy Iteration

Requires:
Markov decision process `MDP`, initial policy π_1, posterior strategy `GibbsPosterior`, initial state $\boldsymbol{s}_0$.

for $i \leftarrow 1$ to N **do**
  Sample action sequences and associated parameters $\boldsymbol{\theta}$, $\boldsymbol{A}^{(n)}$, $\boldsymbol{\theta}^{(n)} \sim \pi_i(\cdot|\boldsymbol{s}_0)$
  Obtain returns $R^{(n)} \sim \texttt{MDP}(\boldsymbol{A}^{(n)})$
  Compute importance weights, $\boldsymbol{w} \leftarrow \texttt{GibbsPosterior}(\boldsymbol{R})$ to estimate $q_\alpha(\boldsymbol{A}|\mathcal{O})$
  Update policy $\pi_{i+1} \leftarrow \texttt{MProjection}(\boldsymbol{w}, \boldsymbol{\theta})$, performing $\min_\pi \mathbb{D}_{\mathrm{KL}}[q_\alpha \mid\mid \pi]$
end for

C Implementation Details

Algorithm 1 describes the general routine of Monte Carlo PPI. The specific `GibbsPosterior` update must be chosen, e.g. REPS, PI^2, MPPI, LBPS or ESSPS. CEM corresponds to uniform importance weights applied to the elite samples in `GibbsPosterior`. By definition, MPPI has a fixed covariance. For MPC, CEM methods reset their covariance each timestep. For actuator limits, we treat them as properties of the dynamics, and later fit the posterior using the *clipped* actions. This recognizes that the applied policy is not necessarily the same as the sampled one and is observed along with the reward. For feature regression weights, clipping is applied but cannot be incorporated into the fitting, as the model weights are being fit, which may explain its worse performance for MPC.

As done in REPS and MORE, the temperature optimization uses off-the-shelf minimizers, such as those found in `scipy.opt.minimize` [90]. While REPS and MORE use gradient-based solvers such as LBFGS-B, for LBPS and ESSPS, the `brent` method in `scipy.opt.minimize_scalar` was found to be slighly faster, presumably due to its quasi-convex objective (see Section F).

For the matrix normal distribution, a weighted maximum likelihood fit and sampling is very similar to the normal distribution. Sampling a matrix normal $\mathcal{MN}(\boldsymbol{M}, \boldsymbol{K}, \boldsymbol{\Sigma})$ is achieved using

$$\boldsymbol{X}^{(n)} = \boldsymbol{M} + \boldsymbol{A}\boldsymbol{W}^{(n)}\boldsymbol{B}, \quad \boldsymbol{K} = \boldsymbol{A}\boldsymbol{A}^\top, \quad \boldsymbol{\Sigma} = \boldsymbol{B}^\top\boldsymbol{B}, \quad W_{ij}^{(n)} \sim \mathcal{N}(0,1).$$

The weighted maximum likelihood fit of the input covariance is computed using [91]

$$\boldsymbol{K} = \textstyle\sum_n w_n (\boldsymbol{X}^{(n)} - \boldsymbol{M})(\boldsymbol{X}^{(n)} - \boldsymbol{M})^\top \boldsymbol{\Sigma}^{-1}, \quad \text{where } \textstyle\sum_n w_n = 1.$$

For the feature approximation of the squared exponential kernel, we used RBF features and quadrature RFFs (QRFF). RBF features require careful normalization in order to approximate the SE kernel at the limiting case [33]. A d-dimensional RBF feature for a SE kernel approximation with lengthscale l is defined as

$$\boldsymbol{\phi}(t) = \frac{1}{\sqrt{\sqrt{\pi}\, d\lambda}} \exp\left(-\frac{(t-\boldsymbol{c})^2}{2\lambda^2}\right),$$

where $\lambda = l/\sqrt{2}$. The d-dimensional centers $\boldsymbol{c}$ are linearly placed along the task horizon.

Quadrature random Fourier features require $d = (2\nu)^k$ features for order ν and input dimension k. Therefore, for time-series we require 2ν features where

$$\phi_j(t) = \begin{cases} w_j \cos(\omega_j t) & j \leq \nu \\ w_{j-\nu} \sin(\omega_{j-\nu} t) & \nu < j \leq 2\nu \end{cases}$$

Gauss-Hermite quadrature provides points $\boldsymbol{u}$ and weights $\boldsymbol{v}$ for a given order, to be used form approximating integrals. In QRFFs, the Monte Carlo integration with Gaussian frequencies [60] is replaced with quadrature. As a results, $w_i = 2v_j/\sqrt{\pi}$ Incorporating the lengthscale, $\omega_i = \sqrt{2}u_i/l$. See Mutný et al. for a more in-depth description of QRFFs, including for higher-dimensional inputs [62].

A practical issue in stochastic search methods is maintaining a sufficiently exploratory search distribution. In accordance with Bayesian methods, PPI methods maintain a belief and do not keep a fixed covariance like MPPI. However, for some tasks it was found that the search distribution had insufficient variance to solve the task effecively, therefore, we introduced an adjustment to Equation 7 to

Algorithm	n_samples		
	16	128	1024
MPPI (white noise)	0.06	0.37	2.63
iCEM (coloured noise)	0.07	0.18	1.08
LBPS (SE kernel)	0.06	0.36	2.74
MPPI (SE kernel)	0.06	0.36	2.74
LBPS (RBF features)	0.06	0.37	2.79

Table 2: Wall-clock time (s) of one MPC calculation for `HumanoidStandup-v2`. Averaged over 10 timesteps and 5 seeds. Computation was performed on a AMD Ryzen 9 3900X 12-Core Processor @ 3.8GHz, parallelized across 24 processes.

'anneal' the covariance, recognizing the the update subtracts a likelihood-based term from the prior. Using

$$\boldsymbol{\Sigma}_{t_3:t_4,t_3:t_4|\mathcal{O}} = \boldsymbol{\Sigma}_{t_3:t_4,t_3:t_4} - \gamma \boldsymbol{\Sigma}_{t_3:t_4,t_1:t_2} \boldsymbol{\Lambda}_{t_1:t_2} \boldsymbol{\Sigma}_{t_3:t_4,t_1:t_2}^\top, \quad 0 \leq \gamma \leq 1,$$

we perform standard PPI for $\gamma = 1$, but adopt an MPPI-like approach for $\gamma = 0$. For `HumanoidStandup-v2`, $\gamma = 0.5$ was required to transition from standing to stabilization effectively. Appendix D.3.4 provides ablation studies of this aspect.

Investigating the runtime of these methods, comparing iCEM, MPPI and LBPS in Table 2, PPI methods are slower than CEM/iCEM, increasing with sample size. Rather than due to the α optimization or kernel-based prior construction, the bottleneck is using all the samples in the matrix normal MLE step for the covariance, computed using `einsum` operations on the sampled parameters. Based on this study, one approximation to speed up PPI methods is use the ESS to reduce the number of samples used in the matrix normal MLE step, pruning samples that have negligible weight, like in CEM.

D Extended Experimental Results

D.1 Black-box Optimization

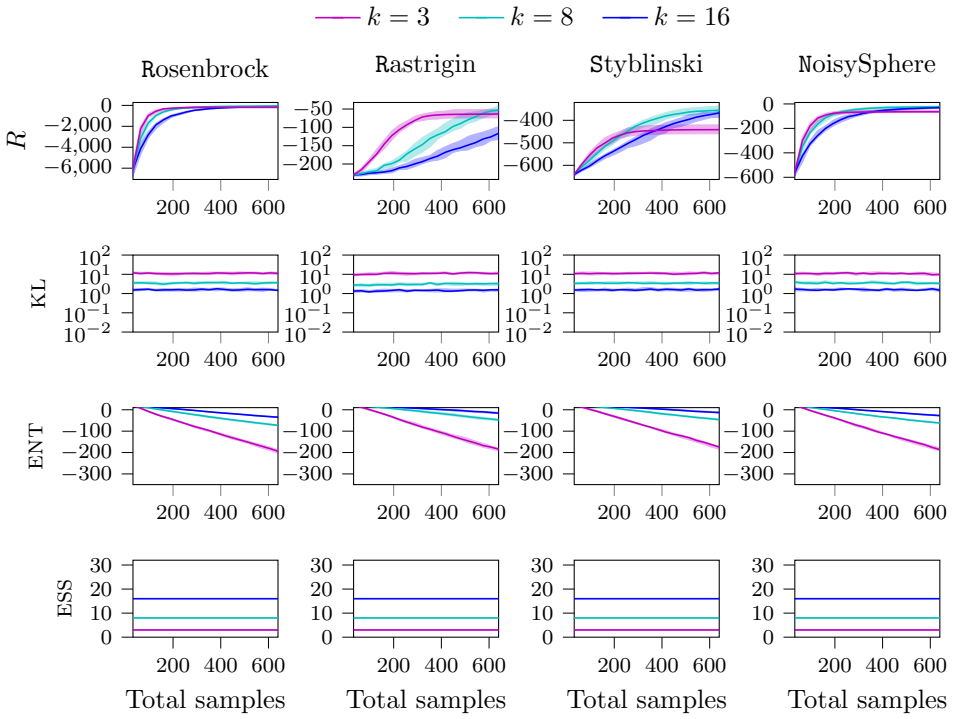

Figure 7: Black-box optimization with CEM and Monte Carlo sampling, with 32 samples over 20 episodes, displaying quartiles over 25 seeds. k is the number of 'elite' samples.

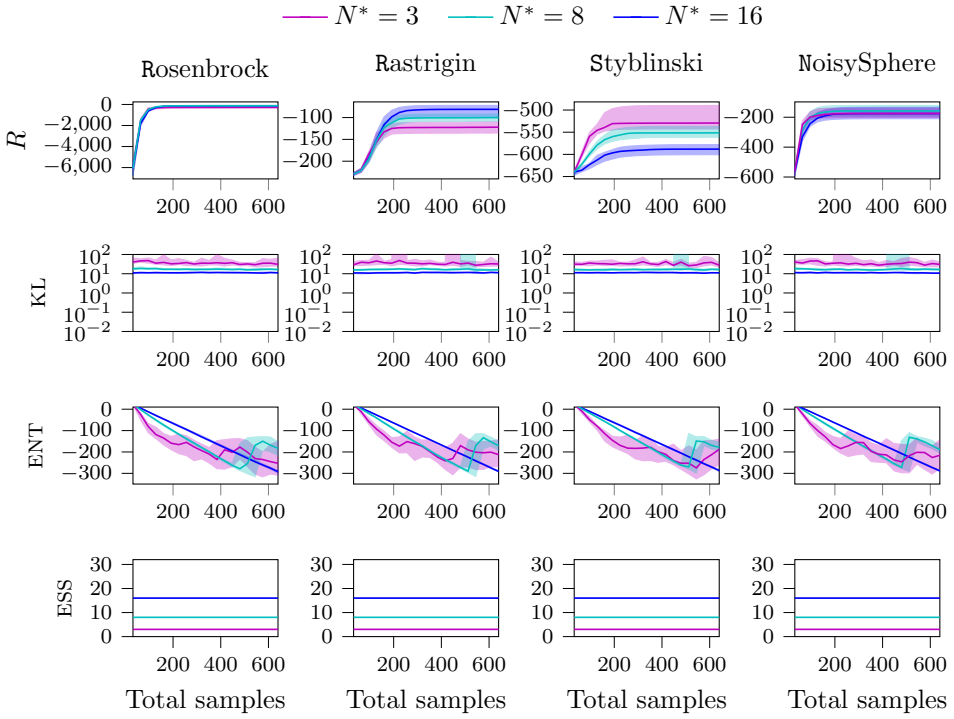

Figure 8: Black-box optimization with ESSPS and Monte Carlo sampling, with 32 samples over 20 episodes, displaying quartiles over 25 seeds. N^* is the desired effective sample size.

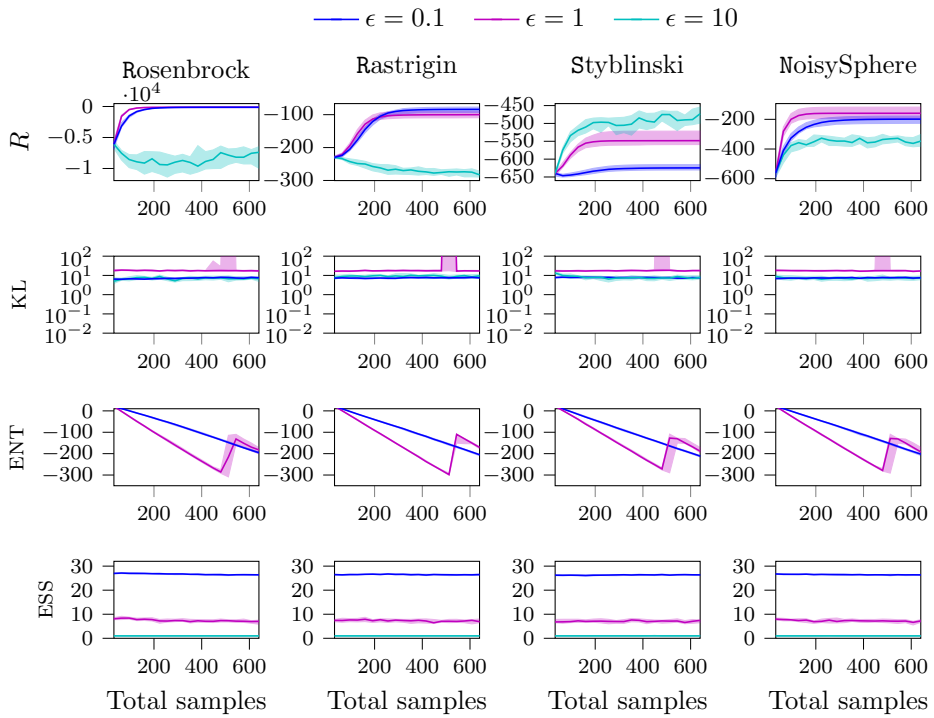

Figure 9: Black-box optimization with REPS and Monte Carlo sampling, with 32 samples over 20 episodes, displaying quartiles over 25 seeds. ϵ is the KL bound.

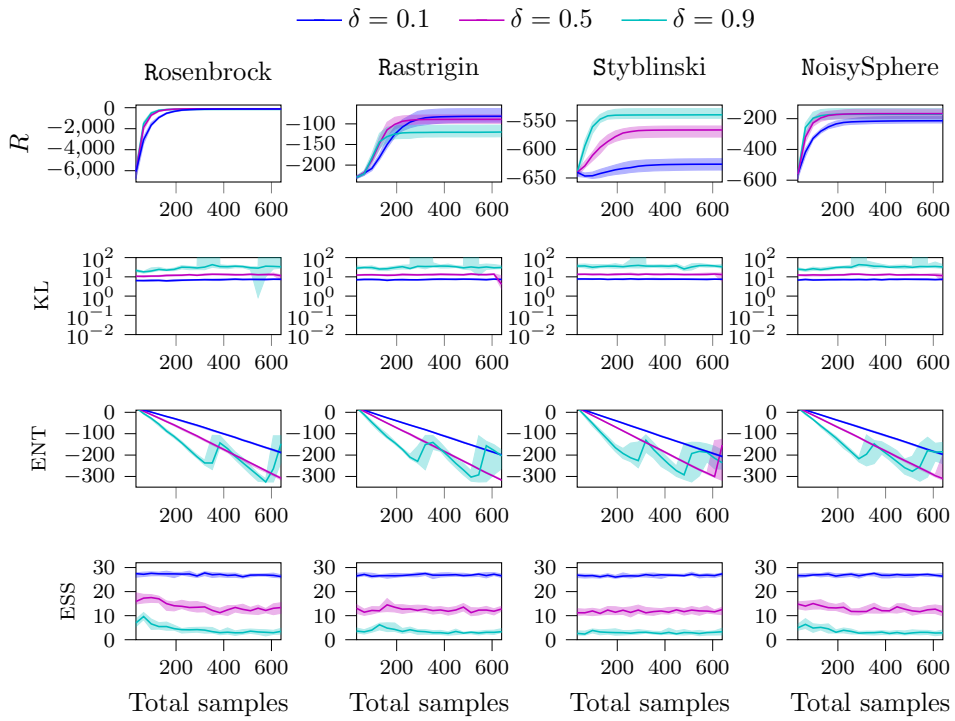

Figure 10: Black-box optimization with LBPS and Monte Carlo sampling, with 32 samples over 20 episodes, displaying quartiles over 25 seeds. δ is the probability of the lower bound.

D.2 Policy Search

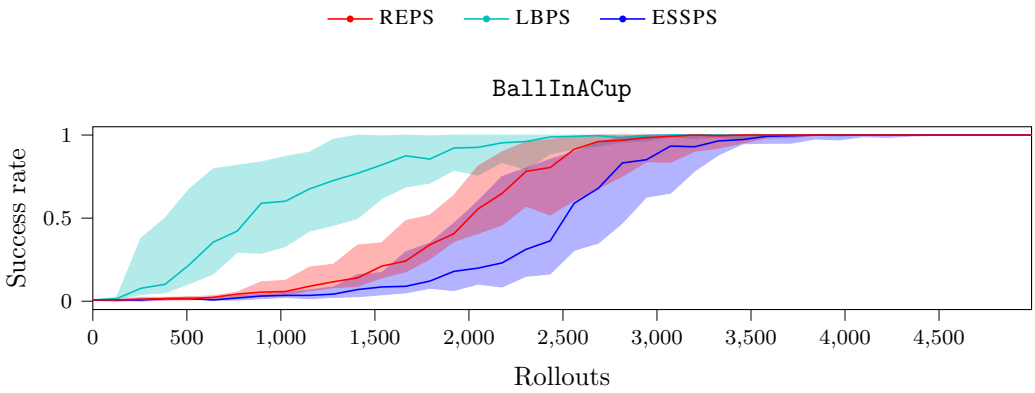

Figure 11: Success rate for policy search with RBF features over 20 seeds, using 128 rollout samples per episode. Displaying uncertainty in quartiles

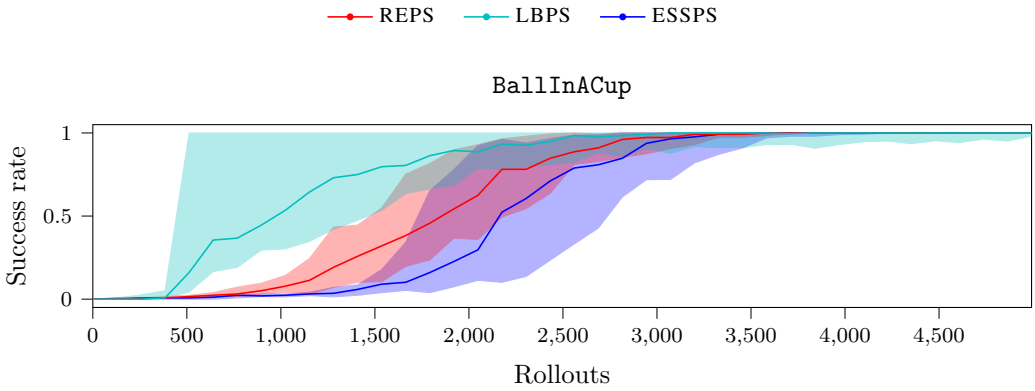

Figure 12: Success rate for policy search with RFF features over 20 seeds, using 128 rollout samples per episode. Displaying uncertainty in quartiles

D.3 Model Predictive Control

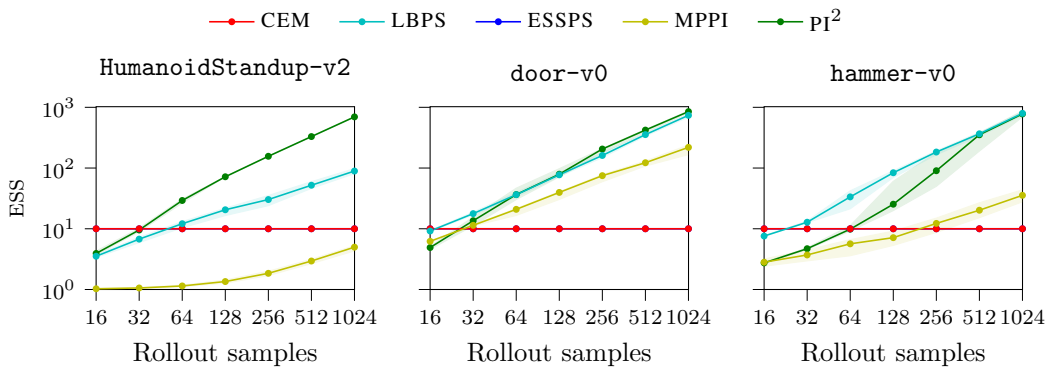

Figure 13: Average ESS for Monte Carlo MPC with white noise priors. Displaying quartiles over 50 seeds.

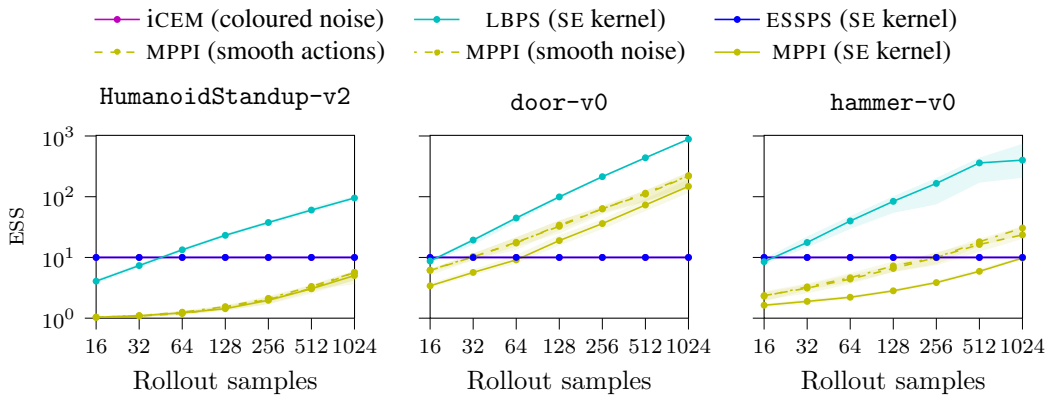

Figure 14: Average ESS for Monte Carlo MPC with smooth noise priors. Displaying quartiles over 50 seeds. iCEM and ESSPS both have an ESS of 10, due to iCEM having 10 elites.

D.3.1 MPC with finite feature approximations

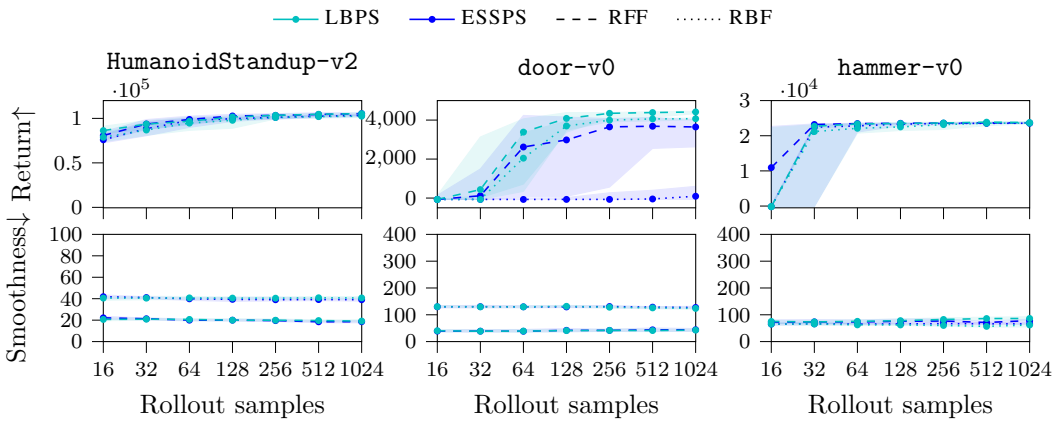

Figure 15: Monte Carlo MPC with with RBF and RFF policies. Displaying quartiles over 50 seeds.

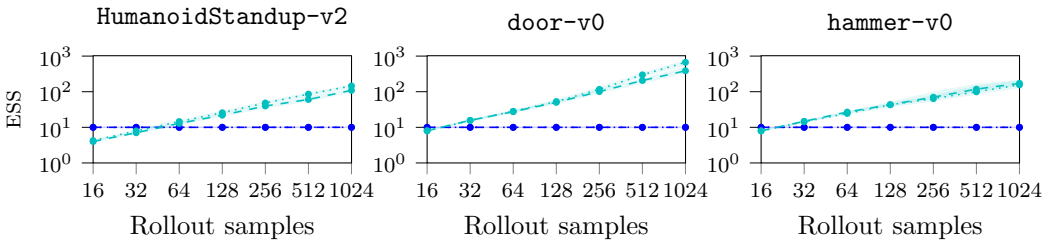

Figure 16: Average ESS for Monte Carlo MPC with RBF and RFF policies. Displaying quartiles over 50 seeds.

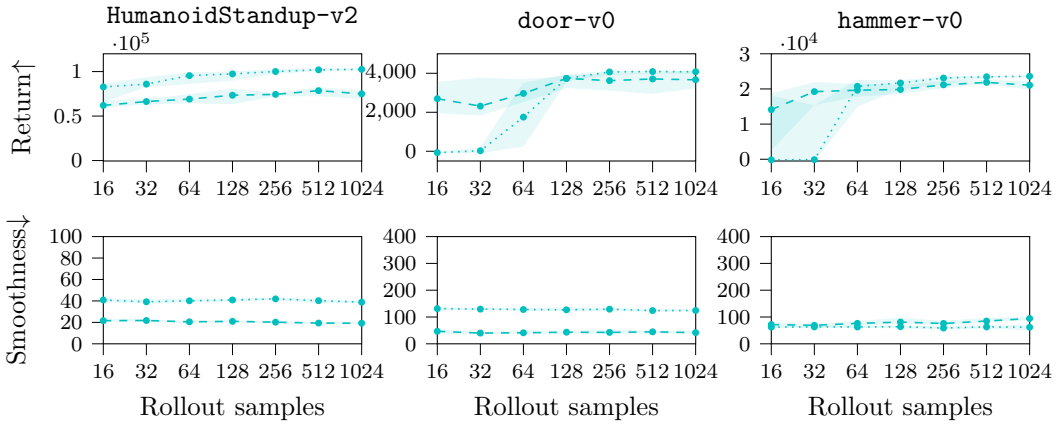

Figure 17: Monte Carlo MPC with RBF and RFF policies, using a (shrinking) planning horizon of 250 rather than 30, the full task duration. Displaying quartiles over 20 seeds.

D.3.2 Visualizing actuation profiles

To accompany the smoothness metric used in the MPC evaluation, we showed how the SE kernel with PPI produces smoother and lower amplitude policies than alternative priors. Due to the high-dimensional action spaces, we depict all actions overlapped as they are normalized.

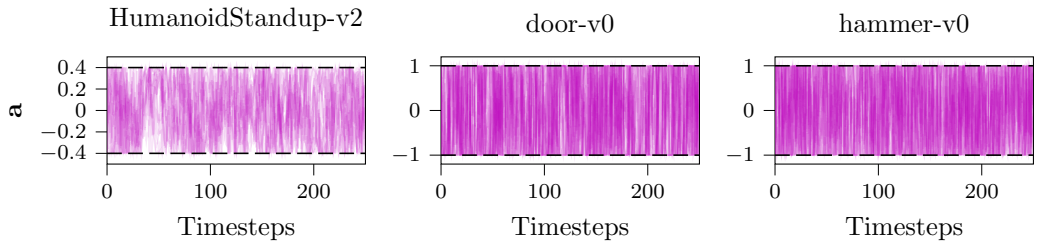

Figure 18: iCEM MPC, with coloured noise, action sequence using 16 rollouts.

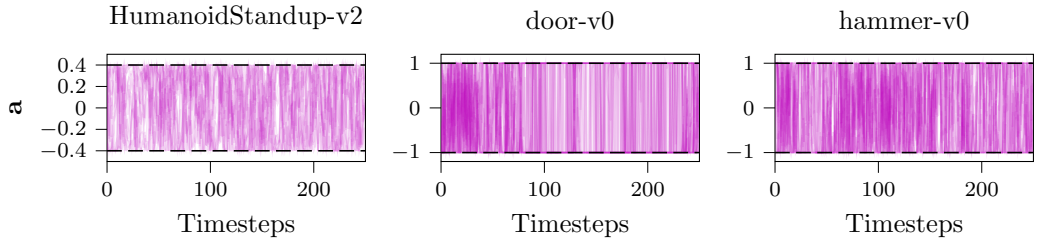

Figure 19: iCEM MPC, with coloured noise, action sequence for using 1024 rollouts.

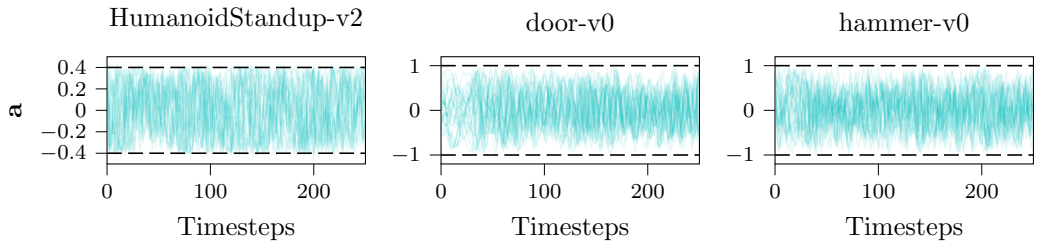

Figure 20: LBPS MPC, using the SE kernel, action sequence using 16 rollouts.

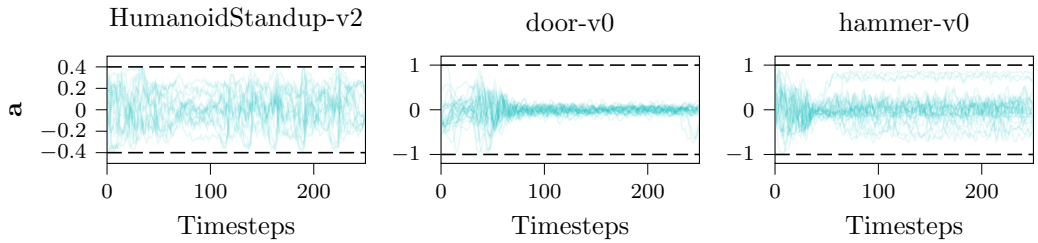

Figure 21: LBPS MPC, using the SE kernel, action sequence using 1024 rollouts.

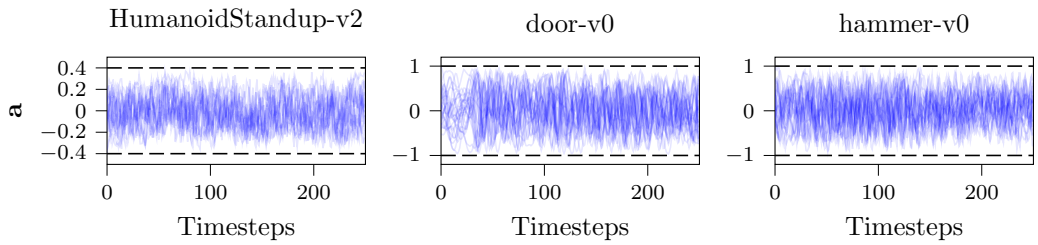

Figure 22: ESSPS MPC, using the SE kernel, action sequence using 16 rollouts.

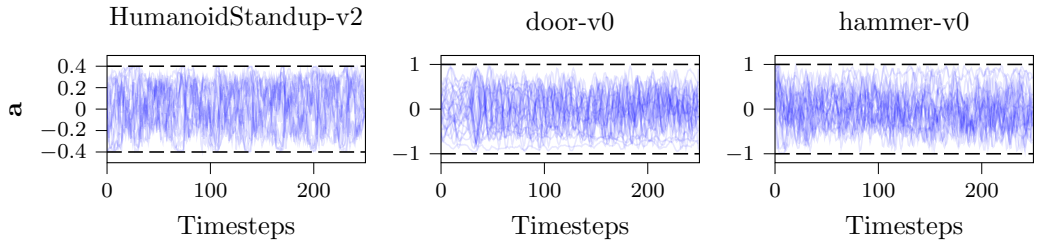

Figure 23: ESSPS MPC, using the SE kernel, action sequence using 1024 rollouts.

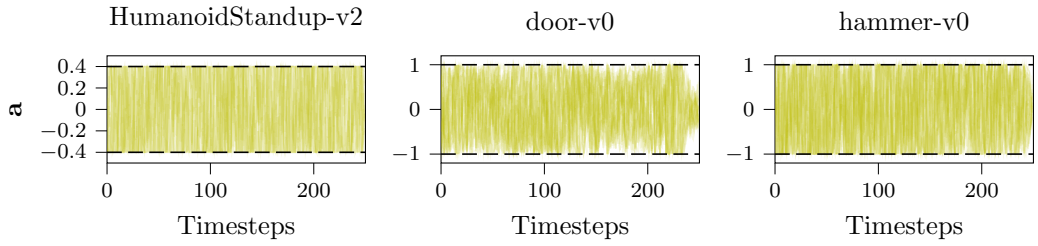

Figure 24: MPPI MPC, using the SE kernel, action sequence using 16 rollouts.

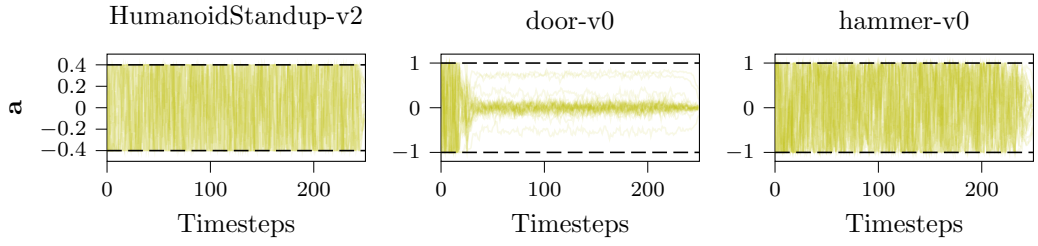

Figure 25: MPPI MPC, using the SE kernel, action sequence using 1024 rollouts.

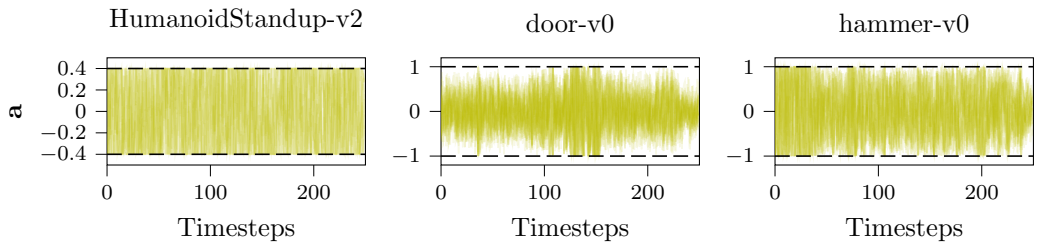

Figure 26: MPPI MPC, using smooth action noise, action sequence using 16 rollouts.

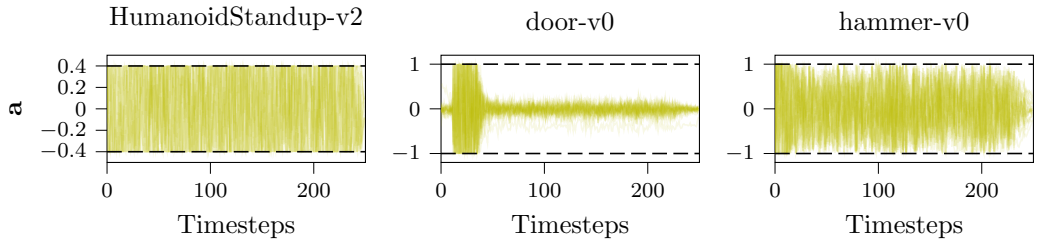

Figure 27: MPPI MPC, using smooth action noise, action sequence using 1024 rollouts.

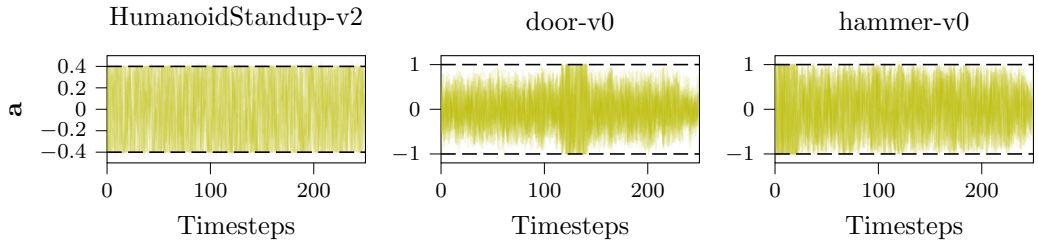

Figure 28: MPPI MPC, using smooth exploration noise, action sequence using 16 rollouts.

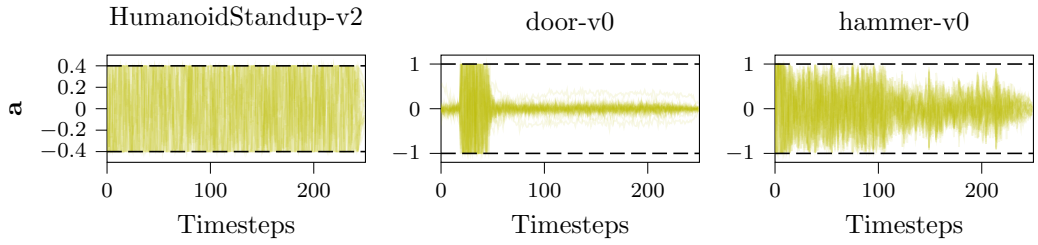

Figure 29: MPPI MPC, using smooth exploration noise, action sequence using 1024 rollouts.

Setting	Return	Smoothness	Lengthscale
door-v0 (expert)	[3301, 4001, 4028]	[139 145 150]	1×10^{-5}
door-v0 (LBPS MPC)	[4326, 4370, 4388]	[50, 55, 58]	0.05
hammer-v0 (expert)	[14042, 17054, 21272]	[142, 147, 151]	1×10^{-5}
hammer-v0 (LBPS MPC)	[23810, 23828, 23904]	[51, 53, 62]	0.025
HumanoidStandup-v2 (GAC)	[92429, 92972, 93300]	–[†]	1×10^{-5}
HumanoidStandup-v2 (LBPS MPC)	[99902 100147 101337]	[20 22 23]	0.05

Table 3: Comparing expert demonstrations to PPI MPC performance w.r.t. return, smoothness and lengthscale. LBPS MPC performance is reported for 1024 rollouts. Uncertainty shows quartiles over dataset demonstrations or experiment seeds. See the discussion text for the omitted result [†].

D.3.3 Learning priors from data

A benefit of the Gaussian process view of policy priors is the ability to perform model selection from data, rather the hyperparameter tuning. This data could be expert demonstrations or a partial solution, such as the 'warm start' in MPC.

We investigate data driven priors by taking expert demonstrations from human and RL agents, analyzing them using the matrix normal distribution. The covariances of the matrix normal allow us to assess the stationarity of the temporal correlations, as well as the correlations between actions. For example, in Figures 30–31 we can see non-stationarity of door-v0 and hammer-v0, due to the motions before and after completing the task, whereas HumanoidStandUp-v2 appears surprisingly stationary due to the task having 'stand up' and 'stabilize' components. However, we found the smoothness that benefited MPC agents in Section 6.3 was not present in the demonstrations. For door-v0 and hammer-v0, the action space is desired joint positions and the demonstrations were collected using motion capture technology [82]. With this in mind, the non-smoothness may be the result of motion capture artefacts or the inverse kinematics used. While the video results of the demonstrations do not suggest rough motion, the action sequences in the dataset do appear rough (Figures 30–31). Moroever, there may be unknown components of the control stack that smooth out the desired joint setpoints. However, the issue may lie in the matrix normal factorization, which assumes each action share the same temporal correlations. This coupling may result in missing smooth sequences if many dimensions have a rough actions.

For HumanoidStandUp-v2, we train and use demonstrations from a gac (generative actor critic) agent [92]. To our knowledge, this is the only model-free deep RL algorithm that solves HumanoidStandUp-v2. However, the policy learned by GAC is a bang-bang controller that operates at the action limits. As a result, the smoothness measure introduced in Section 6.3 'breaks', as the norm of this action sequence is *constant*, suggesting a smoothness of 0. Looking per action independently, the smoothness metric varies around 100 to 300. This result suggests that the IID noise used in deep RL exploration may influence its optimal policy estimate towards rough behaviors such as bang-bang control, which limits their usefulness as expert demonstrators.

The second quantity of interest is the action covariance $\mathbf{\Sigma}$. The demonstrations of door-v0 and hammer-v0 suggest that the independence assumption of $\mathbf{\Sigma}$ is a reasonable one. The action covariance of HumanoidStandUp-v2 depicts much stronger correlations between actions, which could be used to improve exploration through coordination.

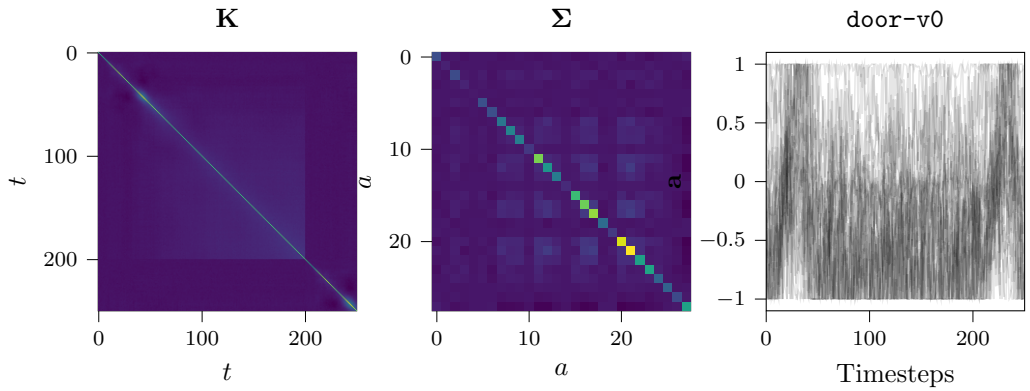

Figure 30: `door-v0` expert demonstrations, showing matrix normal covariance fit and action sequence. The `viridis` colourmap is used to express correlations.

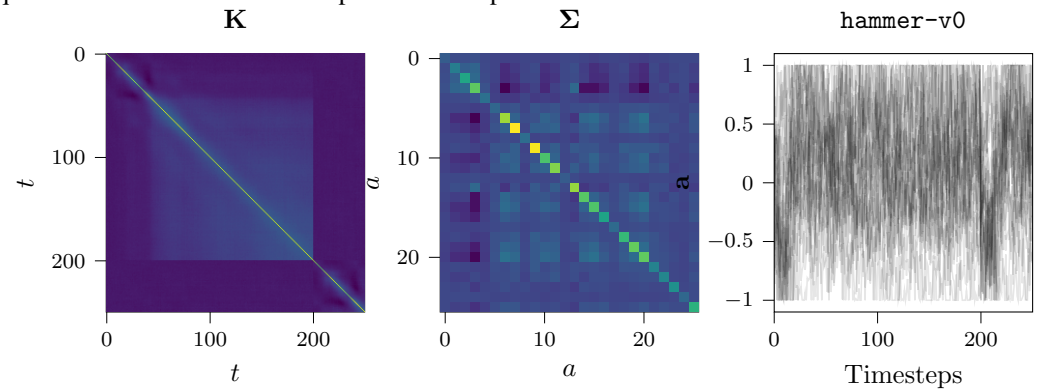

Figure 31: `hammer-v0` expert demonstrations, showing matrix normal covariance fit and action sequence. The `viridis` colourmap is used to express correlations.

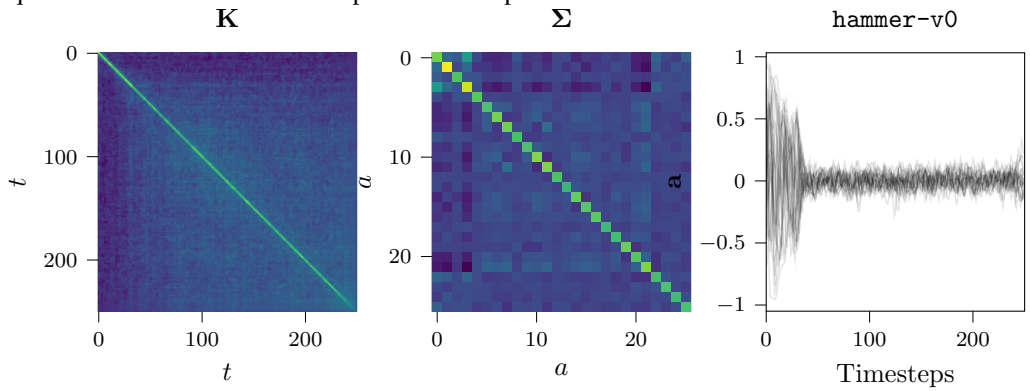

Figure 32: `hammer-v0` expert demonstrations from a MPC solvers (MPPI, LBPS, ESSPS) using the SE kernel, showing matrix normal covariance fit and action sequence. The `viridis` colourmap is used to express correlations.

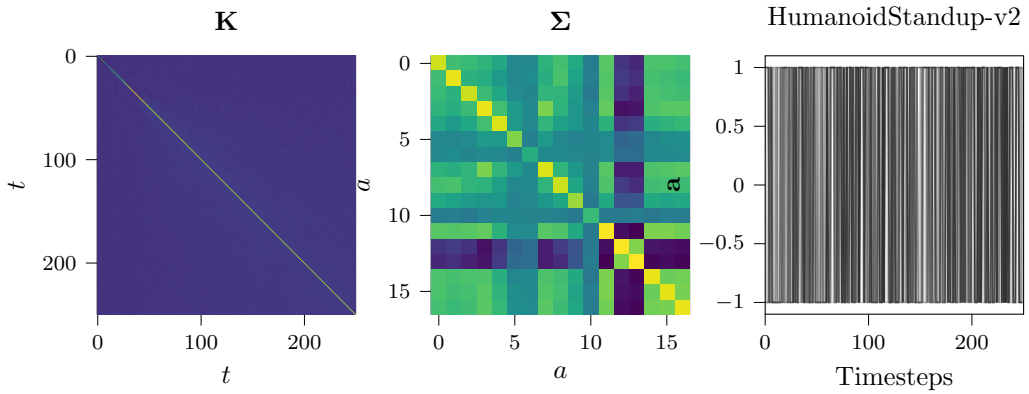

Figure 33: `HumanoidStandup-v2` expert demonstrations from a GAC agent, showing matrix normal covariance fit and action sequence. The `viridis` colourmap is used to express correlations.

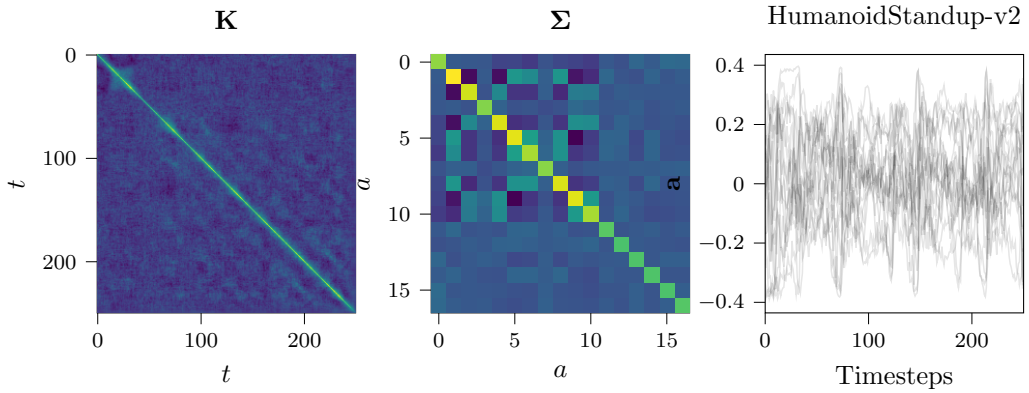

Figure 34: `HumanoidStandup-v2` expert demonstrations from a MPC solvers (MPPI, LBPS, ES-SPS) using the SE kernel, showing matrix normal covariance fit and action sequence. The `viridis` colourmap is used to express correlations.

D.3.4 Model predictive control ablation studies

The MPC methods compared in Section 6.3 have subtle variations in their implementation. MPPI has a constant covariance during optimization, while CEM MPC resets the covariance each timestep. To understand the implications of these details, we provide ablations over these design decisions, in comparison to the results in Section 6.3.

LBPS and ESSPS with constant covariances

On the whole the performance drops compared to the updated covariance results (Figure 4), which suggets that a fixed variance requires greedier optimization (i.e. MPPI) or more iterations per timestep for good performance.

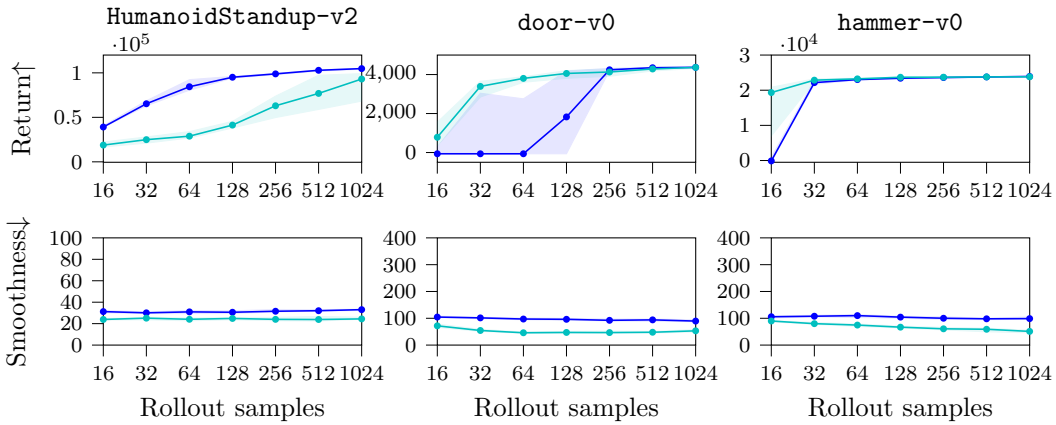

Figure 35: Monte Carlo MPC with LBPS and ESSPS with fixed covariance, like MPPI, as opposed to updated each timestep. Displaying quartiles over 20 seeds.

MPPI with covariance updates

Conversely to the result above, MPPI's greedier optimization does not work effectively with covariance updates, as entropy is quickly lost during the control.

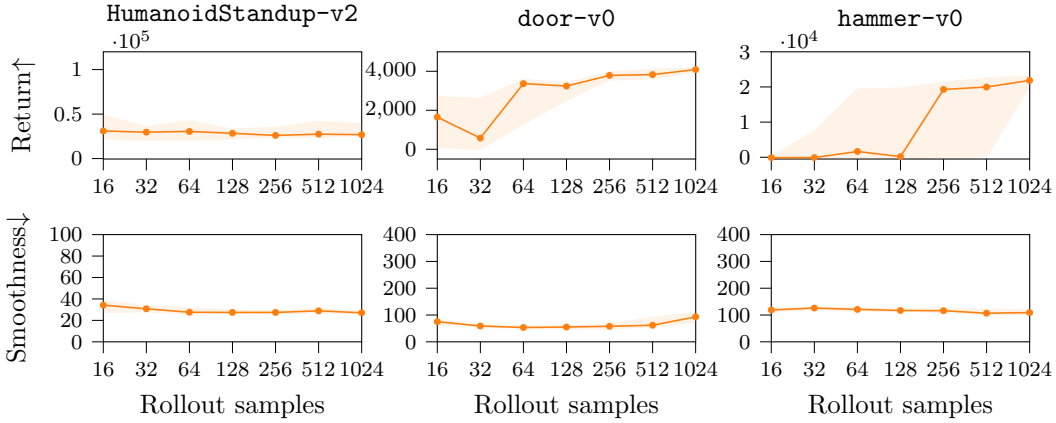

Figure 36: Monte Carlo MPC using MPPI with covariance updates. Displaying quartiles over 20 seeds.

CEM with a SE kernel prior

Compared to the white noise prior in Figure 3, the SE kernel provides an improvement boost for `HumanoidStandup-v2` and `hammer-v0`. There is a smoothness improvement, but slightly smaller than compared to Figure 4. Based on the results of Figure 35, we can attribute this to CEM MPC resetting the policy covariance each timestep.

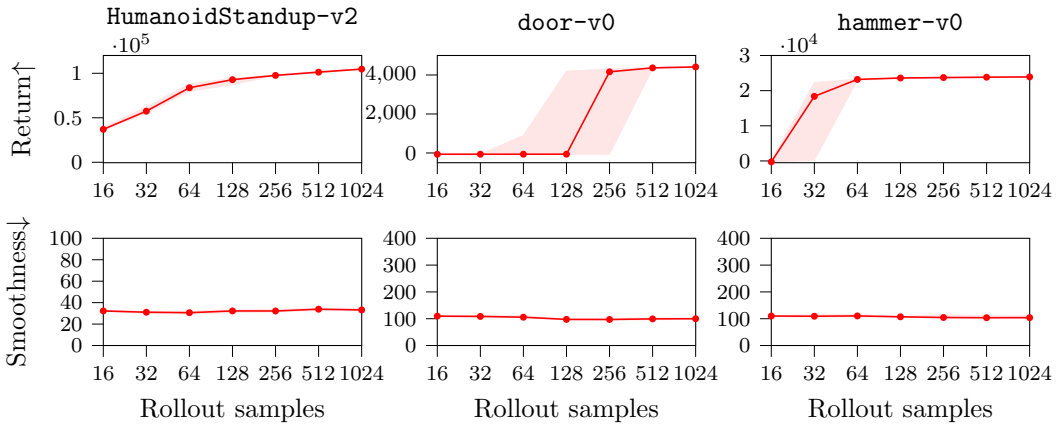

Figure 37: Monte Carlo MPC using CEM with the SE. Displaying quartiles over 20 seeds. Results closely match ESSPS with the SE kernel and fixed covariance (Figure 35).

E Experimental Details

E.1 Hyperparameter Selection

Parameters were swept with there respective solver for best average performance with 256 sample rollouts, averaged over 5 seeds. Parameters were not swept if prior work had identified effective values.

Table 4: Hyperparameter selection for model predictive control methods

	α (MPPI)	$\bar{\alpha}$ (PI^2) [64]	δ (LBPS)	k, N^* (CEM, iCEM, ESSPS) [4]
Considered	[0.1, 1, 10]	-	[0.1, 0.5, 0.9]	-
`HumanoidStandup-v2`	10	10	0.9	10
`door-v0`	10	10	0.5	10
`hammer-v0`	10	10	0.5	10

Table 5: Hyperparameter selection for model predictive control policies

	β (smooth noise)	β (smooth action)	β (coloured noise) [4]	l (SE kernel)
Considered	[0.5, 0.7, 0.9]	[0.5, 0.7, 0.9]	-	[0.01, 0.025, 0.05, 0.1]
`HumanoidStandup-v2`	0.5	0.7	2.0	0.05
`door-v0`	0.9	0.9	2.5	0.05
`hammer-v0`	0.7	0.7	2.5	0.025

E.2 Experiment Hyperparameters

Black-box Optimization. We evaluated the solvers on the 20-dimensional version of the test functions. The initial search distribution was $\mathcal{N}(\mathbf{1}, 0.5\boldsymbol{I})$ for all methods. 32 samples were used over 20 iterations.

Policy Search. The experimental setting of prior work [14] was reproduced, with the same `MuJoCo` simulation and episodic reward function.

Table 6: Hyperparameters for policy search tasks.

	T	d_a	`n_iter`	`n_samples`	RBF features	RFF order	lengthscale, l
`BallInACup`	1000	4	40	128	20	10	$\sqrt{0.03}$

Model Predictive Control. For the MPC tasks, the action mean and variance for each dimension i was set using

$$\mu_i = \frac{a_{\max,i} + a_{\min,i}}{2}, \quad \sigma_i^2 = \frac{(a_{\max,i} - a_{\min,i})^2}{4}.$$

This specification means that $\mu_i \pm \sigma_i$ reaches the actuator limits, ensuring coverage across the actuation range when sampling. μ_i defines a 'mean function' applied as a bias to the policy, which had nominally zero mean. σ_i^2 defines the diagonal of the matrix normal action covariance $\boldsymbol{\Sigma}$. To match the covariance size of the kernel, 30 features were used for the RBF and RFF approximations, i.e. order $\nu = 15$.

Table 7: Hyperparameters for model predictive control tasks.

	T	H	d_a	`n_iters`	`n_warmstart_iters`
`HumanoidStandup-v2`	250	30	17	2	50
`door-v0`	250	30	25	1	50
`hammer-v0`	250	30	25	1	50

F Characterizing the Lower-bound Objective

The temperature objective of REPS is the dual function, which is convex (Definition 3). For the lower bound introduced Section 3, we characterize its nature to understand its suitability for optimization.

First, we express objective in terms of functions f, g and h of α,

$$\max_{\alpha} \mathcal{L}(\alpha) = \frac{\sum_n \exp(\alpha r_n) r_n}{\sum_n \exp(\alpha r_n)} - \lambda \sqrt{\frac{1}{\hat{N}(\alpha)}}, \quad \hat{N}(\alpha) = \frac{(\sum_n \exp(\alpha r_n))^2}{\sum_n \exp(2\alpha r_n)}$$
$$= \frac{\sum_n \exp(\alpha r_n) r_n - \lambda \sqrt{\sum_n \exp(2\alpha r_n)}}{\sum_n \exp(\alpha r_n)} = \frac{f(\alpha) - \lambda g(\alpha)}{h(\alpha)}.$$

The gradient is available in closed form

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathcal{L}(\alpha) = \frac{(\sum_n w_n)(\sum_n w_n r_n^2) - \lambda(\sum_n w_n)\sum_n r_n w_n^2/\sqrt{\sum_n w_n^2} - (\sum_n w_n r_n)^2 + (\sum_n r_n w_n)(\lambda\sqrt{\sum_n w_n^2})}{(\sum_n w_n)^2}.$$

This objective is not concave in α, but is quasi-concave. With $\alpha \in \mathbb{R}_+$, we enforce $r_n \leq 0$ and $\exp(\alpha r_n)$ is convex in α for $r_n \in \mathbb{R}$. We rewrite the objective using in dot product form

$$\min_{\alpha} \mathcal{L}(\alpha) = \frac{\sum_n \exp(\alpha r_n) r_n + \lambda \sqrt{\sum_n \exp(2\alpha r_n)}}{\sum_n \exp(\alpha r_n)} = \frac{\boldsymbol{w}_\alpha^\top \boldsymbol{r} - \lambda \sqrt{\boldsymbol{w}_\alpha^\top \boldsymbol{w}_\alpha}}{\boldsymbol{w}_\alpha^\top \mathbf{1}},$$

where $\boldsymbol{w}_\alpha = [\exp(\alpha r_1), \ldots, \exp(\alpha r_N)]^\top$. The term $\boldsymbol{w}_\alpha^\top \boldsymbol{r}$ is concave as it is a negative weighted sum of convex functions. The term $\boldsymbol{w}_\alpha^\top \mathbf{1}$ is convex as it is a positive sum of weighted convex functions. The remaining term $g(\alpha)$ can be shown to be convex in α (Lemma 3) using standard technques [93].

Lemma 3. *The function $g(\alpha) = \sqrt{\sum_n \exp(2\alpha r_n)}$ is convex in α for $r_n \leq 0 \; \forall n$.*

Proof. For convexity, where $\theta \in [0, 1]$, $\alpha \in \mathbb{R}_+$, $\beta \in \mathbb{R}_+$,

$$\sqrt{\textstyle\sum_n \exp(2(\theta\alpha + (1-\theta)\beta) r_n)} \leq \theta \sqrt{\textstyle\sum_n \exp(2\alpha r_n)} + (1-\theta)\sqrt{\textstyle\sum_n \exp(2\beta r_n)}$$

Starting with the right-hand term, we take interpolation term θ inside

$$\theta \sqrt{\textstyle\sum_n \exp(2\alpha r_n)} + (1-\theta)\sqrt{\textstyle\sum_n \exp(2\beta r_n)} = \sqrt{\textstyle\sum_n (\theta \exp(\alpha r_n))^2} + \sqrt{\textstyle\sum_n ((1-\theta)\exp(\beta r_n))^2}$$

Apply Jensen's inequality to both exponential terms, where $\exp(xy) \leq x \exp(y)$, as they are inside Euclidean norms so we can use $\sqrt{\sum_i p_i^2} \leq \sqrt{\sum_i q_i^2}$ if $p_i < q_i$,

$$\sqrt{\textstyle\sum_n \exp(\theta\alpha r_n)^2} + \sqrt{\textstyle\sum_n \exp((1-\theta)\beta r_n)^2} \leq \sqrt{\textstyle\sum_n (\theta \exp(\alpha r_n))^2} + \sqrt{\textstyle\sum_n ((1-\theta)\exp(\beta r_n))^2}.$$

Apply Minkowski's inequality to the LHS, where $(\sum_i |x_i + y_i|^p)^{1/p} \leq (\sum_i |x_i|^p)^{1/p} + (\sum_i |y_i|^p)^{1/p}$

$$\sqrt{\textstyle\sum_n (\exp(\theta\alpha r_n) + \exp((1-\theta)\beta r_n))^2} \leq \sqrt{\textstyle\sum_n \exp(\theta\alpha r_n)^2} + \sqrt{\textstyle\sum_n \exp((1-\theta)\beta r_n)^2}$$

Expand the terms of the LHS and remove the (non-negative) squared terms

$$\sqrt{\textstyle\sum_n 2\exp((\theta\alpha + (1-\theta)\beta) r_n)} \leq \sqrt{\textstyle\sum_n \exp(2\theta\alpha r_n) + \exp(2(1-\theta)\beta r_n) + 2\exp((\theta\alpha + (1-\theta)\beta) r_n)}$$

Apply Jensen's inequality again to recover the initial left hand term

$$\sqrt{\textstyle\sum_n \exp(2(\theta\alpha + (1-\theta)\beta) r_n)} \leq \sqrt{\textstyle\sum_n 2\exp((\theta\alpha + (1-\theta)\beta) r_n)}.$$

□

With the lemma, the negative penalty term is concave as $\lambda \geq 0$.

For quasi-convexity, we require the t-level sets to be convex $\{\alpha \in \mathbb{R}_+ \mid \mathcal{L}(\alpha) \leq t\}, t \in \mathbb{R}$. Due to the structure of the objective, we require $f(\alpha) - \lambda g(\alpha) \leq t\, h(\alpha)$. As $f(\alpha) \leq 0$, $-\lambda g(\alpha) \leq 0$ and $h(\alpha) \geq 0$, for $t > 0$ we have the empty set, which is convex. For $t \leq 0$, we have the sum of two concave function which are both less or equal to zero, so the set is also convex.

G Stochastic processes, Gaussian processes and coloured noise

This section summarizes the connections between stochastic processes, smoothed noise, coloured noise and Gaussian processes to motivate the use of kernels in action priors. For further details, we refer to Section 12.3 of Särkkä et al. [94] and Chapters 4 and Appendix B of Rasmussen et al. [33].

Section 4 introduced smooth Gaussian noise processes of the form

$$n_t^{(n)} = \sum_{i=1}^{p} a_i \, n_{t-i}^{(n)} + b_0 \, v^{(n)}, \quad v^{(n)} \sim \mathcal{N}(0, 1).$$

to sample action sequences. This is known as a discrete-time autoregressive AR(p) process. The ARMA(p, q) process introduces a noise history, such that ARMA(p, 0) is a AR(p) process

$$n_t^{(n)} = \sum_{i=1}^{p} a_i \, n_{t-i}^{(n)} + \sum_{j=1}^{q} b_j \, v_{t-j}^{(n)}, \quad v_t^{(n)} \sim \mathcal{N}(0, 1).$$

In continuous-time, the AR(1) is analogous to the Ornstein–Uhlenbeck (OU) process

$$\mathrm{d}n(t) = a_0 \, n(t) \, \mathrm{d}t + v(t) \, \mathrm{d}t.$$

The OU covariance function is $\mathrm{Cov}(t, t') = \sigma^2 \exp(-a_0|t - t'|)$, which is also known as the exponential kernel. For additional smoothness we can consider higher-order derivatives, which results in the Matérn family of kernels. In stochastic differential equation form, they are written as

$$n(t) = \boldsymbol{H} \boldsymbol{f}(t), \quad \mathrm{d}\boldsymbol{f}(t) = \boldsymbol{A} \, \boldsymbol{f}(t) \, \mathrm{d}t + \boldsymbol{L} \, v(t) \, \mathrm{d}t,$$

where $\boldsymbol{f}$ contains $n(t)$ and its derivatives, describing the state. The order ν of the Matérn defines the size of the state space and $\boldsymbol{A}$. A first-order kernel reduces to the exponential kernel. These Matérn kernels are Markovian in their state space, following a linear Gaussian dynamical system (LGDS). Therefore, they can be compared to the Gaussian processes used in STOMP [8] and GPMP [30], which are also LGDSs but with different state space models that perform Euler integration, producing priors with Gaussian noise on the velocity, acceleration or jerk. Extending the derivatives for the Matérn kernel, as the order $\nu \to \infty$, we arrive at the squared exponential kernel $\mathrm{Cov}(t, t') = \sigma^2 \exp\left(-|t - t'|^2 / 2l^2\right)$. Comparing to the exponential kernel earlier, we observe the a_0 in the OU process defines the lengthscale of the covariance function.

Considering stationary covariance functions, where $\mathrm{Cov}(t, t+\tau) = \mathrm{Cov}(\tau)$, the power spectral density is defined as the Fourier transform of the covariance function

$$S(\omega) = \int \mathrm{Cov}(\tau) \exp(i\omega\tau) \, \mathrm{d}\tau.$$

From the linear systems perspective, the parameters a and b of an ARMA process describe a linear filter $F(\omega)$ where $N(\omega) = F(\omega)V(\omega)$. In the frequency domain, such is realized as a filter

$$F(\omega) = \frac{|B(\exp i\omega)|^2}{|A(\exp i\omega)|^2}, \quad \text{where} \quad A(\omega) = \sum_{k=1}^{p} a_i \exp(ki\omega).$$

Conversely, *coloured* noise with parameter β applies a filter $\propto 1/\omega^\beta$ to Gaussian noise. The power spectrum of the ν-order Matérn kernel is $S(\omega) \propto (l^2 + \omega^2)^{\nu/2+1}$ and the squared exponential is $S(\omega) \propto \exp(-l^2\omega^2/2)$. While there is no explicit connection between coloured noise and Gaussian processes, by reasoning about their power spectrums it can be seen that they can produce similar sampled paths.