A. Simulation and Cost Details

The swimmer consisted of 3 links, with 10 state dimensions corresponding to joint angles, joint angular velocities, and the position, angle, velocity, and angular velocity of the head, with two action dimensions corresponding to the torques between the joints. The simulation applied drag on each link of the swimmer to roughly simulate a fluid, allowing it to propel itself. The simulation step was set to 0.05s, and the reward weights were $w_{\mathbf{u}} = 0.0001$, $w_v = 1$, and $w_h = 0$, with the desired velocity was $v_x^\star = 1$m/s.

The bipedal walker consisted of seven links: a torso and three links for each leg, for a total of 18 dimensions, including joint angles, the global position and orientation of the torso, and the corresponding velocities. The action space had six dimensions, corresponding to each of the joints. MuJoCo was used to simulate soft, differentiable contacts to allow gradient-based optimization to proceed even in the presence of contact forces. The simulation step was set to 0.01s, and the reward weights for all walker tasks were $w_{\mathbf{u}} = 0.0001$, $w_v = 1$, and $w_h = 10$, with desired velocity and height $v_x^\star = 2.1$m/s and $p_y^\star = 1.1$m.

B. Sample-Based Gradients

As discussed in Section 3.1, the Laplace approximation may not accurately capture the structure of $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ in the entire region where $q(\mathbf{x}_t)$ is large, and since the policy is trained by sampling states from $q(\mathbf{x}_t)$, policy optimization optimizes a different objective. This can lead to nonconvergence when the policy is highly nonlinear. To reconcile this problem, we can approximate the policy terms in the objective with M random samples $\mathbf{x}_{ti}$, drawn from $q(\mathbf{x}_t)$, rather than by using a linearization of the policy:

$$\begin{aligned}\mathcal{L}(q) \approx \sum_{t=1}^{T} \ell(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \frac{1}{2}\text{tr}\left(\Sigma_t \ell_{\mathbf{xu},\mathbf{xu}t}\right) - \frac{1}{2}\log|\mathbf{A}_t| + \\ \frac{\lambda_t}{2M}\sum_{i=1}^{M}(\mathbf{u}_{ti} - \mu_t^\pi(\mathbf{x}_{ti}))^{\mathrm{T}}\mathbf{A}_t^{-1}(\mathbf{u}_{ti} - \mu_t^\pi(\mathbf{x}_{ti})) + \\ \frac{\lambda_t}{2}\text{tr}\left(\mathbf{A}_t^{-1}\Sigma_t^\pi\right) + \frac{\lambda_t}{2}\log|\mathbf{A}_t|,\end{aligned}$$

where the actions are given by $\mathbf{u}_{ti} = \mathbf{K}_t\mathbf{x}_{ti} + \hat{\mathbf{u}}_t$. Note that the samples $\mathbf{x}_{ti}$ depend on $\hat{\mathbf{x}}_t$, according to $\mathbf{x}_{ti} = \hat{\mathbf{x}}_t + \mathbf{L}_t^{\mathrm{T}}\mathbf{s}_{ti}$, where $\mathbf{s}_{ti}$ is a sample from a zero-mean spherical Gaussian, and $\mathbf{L}_t$ is the upper triangular Cholesky decomposition of $\mathbf{S}_t$.[4] As before, we differentiate with respect to $\hat{\mathbf{u}}_t$, substituting $Q_{\mathbf{u},\mathbf{u}t}$ and $Q_{\mathbf{u}t}$ as needed:

$$\begin{aligned}\mathcal{L}_{\mathbf{u}t} &= Q_{\mathbf{u}t} + \lambda_t\mathbf{A}_t^{-1}\hat{\mu}_t^\pi \\ \mathcal{L}_{\mathbf{u},\mathbf{u}t} &= Q_{\mathbf{u},\mathbf{u}t} + \lambda_t\mathbf{A}_t^{-1},\end{aligned}$$

[4] Keeping the same samples $\mathbf{s}_{ti}$ across iterations reduces variance and can greatly improve convergence.

where $\hat{\mu}_t^\pi = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{u}_{ti} - \mu_t^\pi(\mathbf{x}_{ti}))$ is the average difference between the linear feedback and the policy. This yields the following correction and feedback terms:

$$\begin{aligned}\mathbf{k}_t &= -\left(Q_{\mathbf{u},\mathbf{u}t} + \lambda_t\mathbf{A}_t^{-1}\right)^{-1}\left(Q_{\mathbf{u}t} + \lambda_t\mathbf{A}_t^{-1}\hat{\mu}_t^\pi\right) \\ \mathbf{K}_t &= -\left(Q_{\mathbf{u},\mathbf{u}t} + \lambda_t\mathbf{A}_t^{-1}\right)^{-1}\left(Q_{\mathbf{u},\mathbf{x}t} - \lambda_t\mathbf{A}_t^{-1}\hat{\mu}_{\mathbf{x}t}^\pi\right),\end{aligned}$$

where $\hat{\mu}_{\mathbf{x}t}^\pi = \frac{1}{M}\sum_{i=1}^{M}\mu_{\mathbf{x}t}^\pi(\mathbf{x}_{ti})$ is the average policy gradient. So far, the change to the mean is identical to simply averaging the policy values and policy gradients over all the samples. In fact, a reasonable approximation can be obtained by doing just that, and substituting the sample averages directly into the equations in Section 3.1. This is the approximation we use in our experiments, as it is slightly faster and does not appear to significantly degrade the results. However, the true gradients with respect to $\mathbf{A}_t$ are different. Below, we differentiate the objection with respect to $\mathbf{A}_t$ and $\hat{\mathbf{K}}_t$ as before, where we use $\hat{\mathbf{K}}_t$ to distinguish the covariance term from the feedback in the first dynamic programming pass, which is no longer identical. The derivatives with respect to $\mathbf{A}_t$ and $\hat{\mathbf{K}}_t$ are

$$\begin{aligned}\mathcal{L}_{\mathbf{A}t} &= \frac{1}{2}Q_{\mathbf{u},\mathbf{u}t} + \frac{\lambda_t - 1}{2}\mathbf{A}_t^{-1} - \frac{\lambda_t}{2}\mathbf{A}_t^{-1}\mathbf{M}\mathbf{A}_t^{-1} \\ \mathcal{L}_{\hat{\mathbf{K}}t} &= Q_{\mathbf{u},\mathbf{u}t}\hat{\mathbf{K}}_t\mathbf{S}_t + Q_{\mathbf{u},\mathbf{x}t}\mathbf{S}_t + \lambda_t\mathbf{A}_t^{-1}(\hat{\mathbf{K}}_t\hat{\mathbf{S}}_t - \mathbf{C}_t),\end{aligned}$$

where $\mathbf{M} = \Sigma_t^\pi + \sum_{i=1}^{M}(\mathbf{u}_{ti} - \mu_t^\pi(\mathbf{x}_{ti}))(\mathbf{u}_{ti} - \mu_t^\pi(\mathbf{x}_{ti}))$, $\hat{\mathbf{S}}_t = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_{ti}\mathbf{x}_{ti}^{\mathrm{T}}$, and $\mathbf{C}_t = \frac{1}{M}\sum_{i=1}^{M}\mu_t^\pi(\mathbf{x}_{ti})\mathbf{x}_{ti}^{\mathrm{T}}$, where we simplified using the assumption that $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ are zero. We again solve for $\mathbf{A}_t$ by solving the CARE in Equation 8. To solve for $\hat{\mathbf{K}}_t$, we rearrange the terms to get

$$\hat{\mathbf{K}}_t\hat{\mathbf{S}}_t\mathbf{S}_t^{-1} + \frac{1}{\lambda_t}\mathbf{A}_t Q_{\mathbf{u},\mathbf{u}t}\hat{\mathbf{K}}_t = \mathbf{C}_t\mathbf{S}_t^{-1} - \frac{1}{\lambda_t}\mathbf{A}_t Q_{\mathbf{u},\mathbf{x}t}.$$

This equation is linear in $\hat{\mathbf{K}}_t$, but requires solving a sparse linear system with dimensionality equal to the number of entries in $\hat{\mathbf{K}}_t$, which increases the computational cost.

Differentiating with respect to $\mathbf{S}_t$, we get:

$$\begin{aligned}\mathcal{L}_{\mathbf{S}t} = \frac{1}{2}\big[&Q_{\mathbf{x},\mathbf{x}t} + \mathbf{K}_t^{\mathrm{T}}Q_{\mathbf{u},\mathbf{x}t} + Q_{\mathbf{x},\mathbf{u}t}\mathbf{K}_t + \mathbf{K}_t^{\mathrm{T}}Q_{\mathbf{u},\mathbf{u}t}\mathbf{K}_t \\ &+\text{choldiff}(\mathbf{D}_t) + \text{choldiff}(\mathbf{D}_t)^{\mathrm{T}}\big]\end{aligned}$$

where $\mathbf{D}_t = \frac{\lambda_t}{M}\sum_{i=1}^{M}(\hat{\mathbf{K}}_t - \mu_{\mathbf{x}t}^\pi(\mathbf{x}_{ti}))^{\mathrm{T}}\mathbf{A}_t^{-1}(\mathbf{u}_{ti} - \mu_t^\pi(\mathbf{x}_{ti}))\mathbf{s}_{ti}^{\mathrm{T}}$ and $\text{choldiff}(\dots)$ indicates the differentiation of the Cholesky decomposition, for example using the method described by Giles in "An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation" (2008). While this will provide us with the correct gradient, $\text{choldiff}(\mathbf{D}_t) + \text{choldiff}(\mathbf{D}_t)^{\mathrm{T}}$ is not guaranteed to be positive definite. In this case, we found it useful to regularize by interpolating the gradient with the one obtained from the Laplace approximation.