A Appendix

A.1 Details on Gaussian Mixture Regression (GMR)

In GMR we start from a GMM in state-action space $\pi(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{N} \omega_i \mathcal{N}\big([\boldsymbol{s}\, \boldsymbol{a}]^\mathsf{T}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big)$ from which a policy, i.e. a probability distribution on the action space, can be obtained by conditioning on the state, as follows

$$\pi(\boldsymbol{a}|\boldsymbol{s}) = \frac{\pi(\boldsymbol{s}, \boldsymbol{a})}{\int \pi(\boldsymbol{s}, \boldsymbol{a}) \mathrm{d}\boldsymbol{a}}. \tag{15}$$

The resulting conditional distribution is another GMM on the action sapce, with state dependent parameters, given by:

$$\pi(\boldsymbol{a}_t|\boldsymbol{s}_t) = \sum_{i=1}^{N} \omega_i(\boldsymbol{s}_t) \mathcal{N}(\boldsymbol{a}_t; \boldsymbol{\mu}_i^a(s_t), \boldsymbol{\Sigma}_i^a), \quad \text{with} \tag{16}$$

$$\boldsymbol{\mu}_i^a(s_t) = \boldsymbol{\mu}_i^a + \boldsymbol{\Sigma}_i^{as}(\boldsymbol{\Sigma}_i^s)^{-1}\left(\boldsymbol{s}_t - \boldsymbol{\mu}_i^s\right), \tag{17}$$

$$\boldsymbol{\Sigma}_i^a = \boldsymbol{\Sigma}_i^a - \boldsymbol{\Sigma}_i^{as}(\boldsymbol{\Sigma}_i^s)^{-1}\boldsymbol{\Sigma}_i^{sa}, \tag{18}$$

$$\omega_i(\boldsymbol{s}_t) = \frac{\omega_i \mathcal{N}(\boldsymbol{s}_t; \boldsymbol{\mu}_i^s, \boldsymbol{\Sigma}_i^s)}{\sum_k^n \omega_k \mathcal{N}(\boldsymbol{s}_t; \boldsymbol{\mu}_k^s, \boldsymbol{\Sigma}_k^s)}. \tag{19}$$

Note that we have split the GMM parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ into their state and action components according to

$$\boldsymbol{\mu}_i = \begin{pmatrix} \boldsymbol{\mu}_i^s \\ \boldsymbol{\mu}_i^a \end{pmatrix}, \quad \boldsymbol{\Sigma}_i = \begin{pmatrix} \boldsymbol{\Sigma}_i^s & \boldsymbol{\Sigma}_i^{sa} \\ \boldsymbol{\Sigma}_i^{as} & \boldsymbol{\Sigma}_i^a \end{pmatrix}. \tag{20}$$

A.2 Riemannian gradients and retractions

For completeness we give here the explicit expressions of the Riemannian gradients and the retractions used in § 3.1. As the mean vectors are assumed to lie in the Euclidean space, their Riemannian gradients actually coincide with the Euclidean gradients and no retraction is required, so Eq. 12 reduces to the well-known Euclidean gradient descent

$$\hat{\boldsymbol{\mu}}_{k+1} = \hat{\boldsymbol{\mu}}_k + \nabla_{\hat{\boldsymbol{\mu}}} J(\pi_k), \tag{21}$$

where $\nabla_{\hat{\boldsymbol{\mu}}}$ denotes the Euclidean gradient w.r.t. $\hat{\boldsymbol{\mu}}$. For the covariance matrices we use the gradient and retraction w.r.t. the Bures-Wasserstein manifold, taken from [59, 60]. The gradient is given by

$$\mathrm{grad}_{\hat{\boldsymbol{\Sigma}}} J(\pi_k) = 4\{\nabla_{\hat{\boldsymbol{\Sigma}}} J(\pi_k)\hat{\boldsymbol{\Sigma}}\}_S, \tag{22}$$

where again $\nabla_{\hat{\boldsymbol{\Sigma}}}$ denotes the Euclidean gradient w.r.t. $\hat{\boldsymbol{\Sigma}}$ and $\{\boldsymbol{X}\}_S = \frac{(\boldsymbol{X}+\boldsymbol{X}^\mathsf{T})}{2}$.

Furthermore, the retraction is given by

$$\mathrm{R}_{\boldsymbol{\Sigma}_k}\left(\hat{\boldsymbol{X}}\right) = \hat{\boldsymbol{\Sigma}}_k + \hat{\boldsymbol{X}} + \mathcal{L}_{\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{\Sigma}}_k\right]\hat{\boldsymbol{X}}\mathcal{L}_{\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{\Sigma}}_k\right], \tag{23}$$

where $\mathcal{L}_{\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{\Sigma}}_k\right]$ is the Lyapunov operator, defined as the solution to the matrix linear system

$$\mathcal{L}_{\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{\Sigma}}_k\right]\hat{\boldsymbol{X}} + \hat{\boldsymbol{X}}\mathcal{L}_{\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{\Sigma}}_k\right] = \hat{\boldsymbol{\Sigma}}_k. \tag{24}$$

A.3 Expressions of the free functional $J(\pi)$ and its Euclidean gradients

For completeness sake, we provide here the explicit expression of the Euclidean gradients for the objective $J(\pi)$ w.r.t. the parameters of the GMM, which are used in the construction of the Riemannian gradients. Using the policy gradient theorem, we obtain the gradient of Eq. 11 w.r.t to a

parameter $\boldsymbol{\xi}$ as follows

$$\begin{aligned}\nabla_{\boldsymbol{\xi}} J(\pi) &= \nabla_{\boldsymbol{\xi}} \int \Pi_t \mathrm{d}\boldsymbol{s}_0 \mathrm{d}\boldsymbol{s}_t \mathrm{d}\boldsymbol{a}_t \rho(\boldsymbol{s}_0)\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)\sum_{t'>t}\gamma^t r(\boldsymbol{s}_{t'},\boldsymbol{a}_{t'}), \\ &= \mathbb{E}_\tau\left[\sum_t \nabla_{\boldsymbol{\xi}} \log(\pi(\boldsymbol{a}_t|\boldsymbol{s}_t))\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right], \\ &= \mathbb{E}_\tau\left[\sum_t \nabla_{\boldsymbol{\xi}} \log\left(\frac{\pi(\boldsymbol{s}_t,\boldsymbol{a}_t)}{\int \mathrm{d}\boldsymbol{a}_t \pi(\boldsymbol{s}_t,\boldsymbol{a}_t)}\right)\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right], \\ &= \sum_t \mathbb{E}_\tau\left[\left(\frac{\nabla_{\boldsymbol{\xi}}\pi(\boldsymbol{s}_t,\boldsymbol{a}_t)}{\pi(\boldsymbol{s}_t,\boldsymbol{a}_t)} - \frac{\int \mathrm{d}\boldsymbol{a}_t \nabla_{\boldsymbol{\xi}}\pi(\boldsymbol{s}_t,\boldsymbol{a}_t)}{\int \mathrm{d}\boldsymbol{a}_t \pi(\boldsymbol{s}_t,\boldsymbol{a}_t)}\right)\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right].\end{aligned} \tag{25}$$

In this work, we focus on GMM policies, for which the objective $J(\pi)$ takes the form:

$$\begin{aligned}J(\pi) = &\int \Pi_t \mathrm{d}\boldsymbol{s}_0 \mathrm{d}\boldsymbol{s}_t \mathrm{d}\boldsymbol{a}_t \rho(\boldsymbol{s}_0)\sum_{i=1}^n \omega_i(\boldsymbol{s}_t)\mathcal{N}(\boldsymbol{a}_t;\boldsymbol{\mu}_i(\boldsymbol{s}_t),\boldsymbol{\Sigma}_i(\boldsymbol{s}_t))p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,a_t)\sum_t \gamma^t r(\boldsymbol{s}_t,\boldsymbol{a}_t) \\ &+ \beta\int \mathrm{d}\boldsymbol{a}_t \sum_{i=1}^n \omega_i(\boldsymbol{s}_t)\mathcal{N}(\boldsymbol{a}_t;\mu_i(\boldsymbol{s}_t),\Sigma_i(\boldsymbol{s}_t))p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t) \\ &\log\left(\sum_{i=1}^n \omega_i(s_t)\mathcal{N}(\boldsymbol{a}_t;\mu_i(\boldsymbol{s}_t),\Sigma_i(\boldsymbol{s}_t))p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)\right).\end{aligned} \tag{26}$$

By inserting Eq. 26 into Eq. 25 we obtain for the individual parameters of the GMM

$$\begin{aligned}\nabla\boldsymbol{\mu}_l J(\pi) &= \mathbb{E}_\tau\left[\sum_t\left(\frac{\boldsymbol{\omega}_l\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)\boldsymbol{\Sigma}_l^{-1}\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)}{\sum_j \boldsymbol{\omega}_j\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}\right.\right. \\ &\left.\left. -\frac{\boldsymbol{\omega}_l\int \mathrm{d}\boldsymbol{a}\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)\boldsymbol{\Sigma}_l^{-1}\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)}{\sum_j \boldsymbol{\omega}_j\int \mathrm{d}\boldsymbol{a}\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}\right)\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right],\end{aligned} \tag{27}$$

$$\begin{aligned}&= \mathbb{E}_\tau\left[\sum_t\left(\frac{\boldsymbol{\omega}_l\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)\boldsymbol{\Sigma}_l^{-1}\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)}{\sum_j \boldsymbol{\omega}_j\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}\right.\right. \\ &\left.\left. -\delta_{\boldsymbol{s}}\frac{\boldsymbol{\omega}_l\mathcal{N}(\boldsymbol{s}_t;\boldsymbol{\mu}_{l,s}\boldsymbol{\Sigma}_{l,ss})\boldsymbol{\Sigma}_{l,ss}^{-1}\left(\boldsymbol{s}_t-\boldsymbol{\mu}_{l,s}\right)}{\sum_j \boldsymbol{\omega}_j\mathcal{N}(\boldsymbol{s}_t;\boldsymbol{\mu}_{j,s},\boldsymbol{\Sigma}_{j,ss})}\right)\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right],\end{aligned} \tag{28}$$

$$= \mathbb{E}_\tau\left[\sum_t\left(\zeta_{l,\boldsymbol{s}_t,\boldsymbol{a}_t}\boldsymbol{\Sigma}_l^{-1}\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)-\delta_s\zeta_{l,\boldsymbol{s}_t}\boldsymbol{\Sigma}_{l,ss}^{-1}\left(\boldsymbol{s}_t-\boldsymbol{\mu}_{l,s}\right)\right)\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right].$$

Here $\delta_{\boldsymbol{s}} \in \{0,1\}$ indicates which terms the gradient acts on. In this case, the gradient act on the state components and it is absent for the action dimensions.

$$\begin{aligned}\nabla_{\boldsymbol{\Sigma}_l} J(\pi) &= \mathbb{E}_\tau\left[\sum_t\left(-\frac{1}{2}\frac{\boldsymbol{\omega}_l\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)\boldsymbol{\Sigma}_l^{-1}\left(1-\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)^\mathsf{T}\boldsymbol{\Sigma}_l^{-1}\right)}{\sum_j \boldsymbol{\omega}_j\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}\right.\right. \\ &\left.\left. +\frac{1}{2}\frac{\boldsymbol{\omega}_l\int \mathrm{d}\boldsymbol{a}\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)\boldsymbol{\Sigma}_l^{-1}\left(1-\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)^\mathsf{T}\boldsymbol{\Sigma}_l^{-1}\right)}{\sum_j \boldsymbol{\omega}_j\int \mathrm{d}a\mathcal{N}(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}\right)\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right],\end{aligned} \tag{29}$$

$$\begin{aligned}&= \mathbb{E}_\tau\left[\sum_t\left(-\frac{\zeta_{l,\boldsymbol{s}_t,\boldsymbol{a}_t}}{2}\boldsymbol{\Sigma}_l^{-1}\left(1-\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)\left((\boldsymbol{s}_t,\boldsymbol{a}_t)-\boldsymbol{\mu}_l\right)^\mathsf{T}\boldsymbol{\Sigma}_l^{-1}\right)\right.\right. \\ &\left.\left. +\delta_s\frac{\zeta_{l,\boldsymbol{s}_t}}{2}\boldsymbol{\Sigma}_{l,s}^{-1}\left(1-\left(\boldsymbol{s}_t-\boldsymbol{\mu}_{l,s}\right)\left(\boldsymbol{s}_t-\boldsymbol{\mu}_{l,s}\right)^\mathsf{T}\boldsymbol{\Sigma}_{l,s}^{-1}\right)\right)\sum_{t'>t} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right].\end{aligned}$$

$$\nabla \boldsymbol{\omega}_l J(\pi) = \mathbb{E}_\tau \left[\sum_t \left(\frac{\mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}{\sum_j \boldsymbol{\omega}_j \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} - \frac{\int \mathrm{d}\boldsymbol{a} \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}{\sum_j \boldsymbol{\omega}_j \int \mathrm{d}\boldsymbol{a} \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \right) \sum_{t'>t} r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \tag{30}$$

$$= \mathbb{E}_\tau \left[\sum_t \frac{(\zeta_{l,\boldsymbol{s}_t,\boldsymbol{a}_t} - \zeta_{l,\boldsymbol{s}_t})}{\boldsymbol{\omega}_l} \sum_{t'>t} r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]. \tag{31}$$

Note that we introduced the responsibilities $\zeta_{l,\boldsymbol{s}_t,\boldsymbol{a}_t}$ and $\zeta_{l,\boldsymbol{s}_t}$, which are defined as follows

$$\zeta_{l,\boldsymbol{s}_t,\boldsymbol{a}_t} = \frac{\boldsymbol{\omega}_l \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}{\sum_j \boldsymbol{\omega}_j \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \quad \text{and} \tag{32}$$

$$\zeta_{l,\boldsymbol{s}_t} = \frac{\boldsymbol{\omega}_l \int \mathrm{d}\boldsymbol{a} \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}{\sum_j \boldsymbol{\omega}_j \int \mathrm{d}\boldsymbol{a} \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} = \frac{\boldsymbol{\omega}_l \mathcal{N}(\boldsymbol{s}_t; \boldsymbol{\mu}_{l,s}, \boldsymbol{\Sigma}_{l,ss})}{\sum_j \boldsymbol{\omega}_j \mathcal{N}(\boldsymbol{s}_t; \boldsymbol{\mu}_{j,s}, \boldsymbol{\Sigma}_{j,ss})}. \tag{33}$$

A.4 Relation between forward and backward discretization in the Bures-Wasserstein metric

In this section we outline the relation between the implicit and explicit optimization schema w.r.t. the Bures-Wasserstein metric, which is leveraged to formulate our policy optimization in § 3. We closely follow [58]. For the sake of simplicity, we group the Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ into a single parameter vector $\boldsymbol{\theta}$. Furthermore, we restrict our explanation to a single Gaussian component, which is possible without loosing generality, as each of the N components live in its own manifold $\mathbb{R}^d \times \mathcal{S}^d_{++}$.

The Riemannian gradient w.r.t the Gaussian parameters $\boldsymbol{\theta}$, $\mathrm{grad}_{\boldsymbol{\theta}}\, J(\pi(\boldsymbol{\theta}))$, satisfies by definition

$$g_{\boldsymbol{\theta}}(\mathrm{grad}_{\boldsymbol{\theta}}\, J(\pi(\boldsymbol{\theta}), \boldsymbol{\xi}) = \nabla_{\boldsymbol{\theta}} J(\pi(\boldsymbol{\theta})) \cdot \boldsymbol{\xi}, \tag{34}$$

where $\nabla_{\boldsymbol{\theta}}$ denotes the Euclidean gradient, $\boldsymbol{\xi}$ is an arbitrary vector on the tangent space $\mathcal{T}_{\boldsymbol{\theta}}\mathcal{M}$, and $g_{\boldsymbol{\theta}}$ is the Riemannian metric tensor, defining the inner product on $\mathcal{T}_{\boldsymbol{\theta}}\mathcal{M}$. The Riemannian metric $g_{\boldsymbol{\theta}}$ can be written as

$$g_{\boldsymbol{\theta}}(\boldsymbol{\zeta}, \boldsymbol{\xi}) = \boldsymbol{\zeta}^\mathsf{T} G_W(\boldsymbol{\theta}) \boldsymbol{\xi}, \tag{35}$$

with two arbitrary tangent vectors $\boldsymbol{\zeta}, \boldsymbol{\xi}$, and $G_W(\boldsymbol{\theta})$ being a positive definite matrix. Moreover, note that the Wasserstein distance $W_2^2\big(\mathcal{N}(\boldsymbol{\theta}), \mathcal{N}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta})\big)$, where $\Delta\boldsymbol{\theta}$ denotes a small perturbation in the Gaussian parameters $\boldsymbol{\theta}$, can be expressed as

$$W_2^2\big(\mathcal{N}(\boldsymbol{\theta}), \mathcal{N}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta})\big) = \frac{1}{2}\big(\Delta\boldsymbol{\theta}^\mathsf{T}\big) G_W(\boldsymbol{\theta}) \big(\Delta\boldsymbol{\theta}\big) + O\big((\boldsymbol{\Delta\theta})^2\big), \tag{36}$$

for $\Delta\boldsymbol{\theta} \to 0$. Similarly, we can approximate the objective evaluated at $J(\boldsymbol{\theta} + \Delta\boldsymbol{\theta})$ via the Taylor theorem as

$$J(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \cdot \Delta\boldsymbol{\theta} + O((\Delta\boldsymbol{\theta})^2). \tag{37}$$

With this, we can approximate

$$\boldsymbol{\theta}_{k+1} = \arg\min_{\boldsymbol{\theta}} \left(\frac{W_2^2(\pi(\boldsymbol{\theta}), \pi(\boldsymbol{\theta}_k))}{2\tau} - J(\pi(\boldsymbol{\theta})) \right), \tag{38}$$

$$\approx \arg\min_{\boldsymbol{\theta}} \left(\frac{(\boldsymbol{\theta} - \boldsymbol{\theta_k})^\mathsf{T} G_W (\boldsymbol{\theta} - \boldsymbol{\theta_k})}{2\tau} - \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta_k}) \right), \tag{39}$$

from which we obtain the update equation for $\boldsymbol{\theta}$ as follows

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \tau G_W(\boldsymbol{\theta}_k)^{-1} \nabla_{\boldsymbol{\theta}} J(\pi(\boldsymbol{\theta}_k)), \tag{40}$$

which corresponds to the Wasserstein natural gradient with respect to the Bures-Wasserstein metric. Note that Eq. 40 in turn corresponds to an approximation of the exact Riemannian gradient descent

$$\boldsymbol{\theta}_{k+1} = \mathrm{R}_{\boldsymbol{\theta}_k} \left(\tau \cdot \mathrm{grad}_{\boldsymbol{\theta}}\, J(\pi(\boldsymbol{\theta}_k)) \right). \tag{41}$$

This approximation can be obtained by considering a first-order approximation of the geodesic on the BW manifold. As the exponential map (a.k.a. the retraction) is defined via the geodesic, the retraction operator in Eq. 41 turns into a simple addition operation under a first-order approximation, leading to Eq. 40. Notice that such approximation does not guarantee that the updated parameters $\boldsymbol{\theta}$ stay on the manifold, except for the cases in which $\boldsymbol{\theta} \in \mathbb{R}^d$. This means that the positive-definite constraints arising from the covariance matrices can be easily violated when using natural gradient approaches as they do not guarantee that the updates stay on the underlying manifold. However, it is worth noting that the use of an inexact Riemannian gradient descent is often motivated by the difficulty of computing the exponential map (or geodesic) necessary to calculate the exact Riemannian update. In our case, we leverage the retraction and Riemannian gradients of [59, 60], which allow us to apply the exact Riemannian gradient descent of 41. This avoids to rely on first-order approximations and in turn we can guarantee that the updates of the Gaussian distribution parameters always lie on on the product manifold $\left(\mathbb{R}^d \times \mathcal{S}_{++}^d\right)^N$.

A.5 Additional details on the implementation

We extended the Pymanopt [64] by adding a custom line-search routine that accounts for a constraint on the Wasserstein distance between the old and the optimized GMMs. The details of this line-search can be found in Algorithm 2.

Algorithm 2 Constrained line-search. The constraint function $c(x_0, \cdot)$ is arbitrary in general. We use the L^2-Wasserstein distance between two points on the manifold of GMMs as constraint.

Input: point $\boldsymbol{x_0}$ on the manifold, descent direction $\boldsymbol{d}$, initial step size λ_0, decrement α, constraint $c(\boldsymbol{x_0}, \cdot)$, maximum allowed value for constraint c_{max}, minimum step size λ_{min}
Output: step size s, updated point on manifold $\boldsymbol{x}$

$\boldsymbol{x} = \boldsymbol{x_0} + \lambda_0 \cdot \boldsymbol{d}$
$\lambda = \lambda_0$
while $c(\boldsymbol{x_0}, \boldsymbol{x}) > c_{\text{max}}$ and $\lambda > \lambda_{\text{min}}$ **do**
    decrease step size: $\lambda = \alpha \cdot \lambda$
    update point on manifold: $\boldsymbol{x} = \boldsymbol{x_0} + \lambda \cdot \boldsymbol{d}$
end while
if $\lambda < \lambda_{\text{min}}$ **then**
    return λ_0, $\boldsymbol{x_0}$
else
    return λ, $\boldsymbol{x}$
end if

A.6 Additional details on experiments

A.6.1 Additional results

Fig. 5 shows the convergence curves for the two baselines as in Fig. 3 of the main paper, however, we extended the horizontal axis up to the maximum number of environment steps used for training.

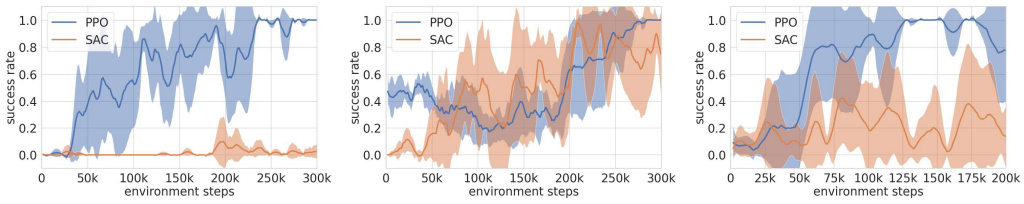


Figure 5: The success rate of the two baselines on the reaching task (*left*), the collision-avoidance task (*middle*) and the multiple-goal task (*right*). The shaded area indicates the standard deviation over 5 runs.

Fig. 6 shows the variance of the success rate for the three methods at their time step of convergence for all three robotic tasks. Concerning SAC, which did not converge after the maximum number of

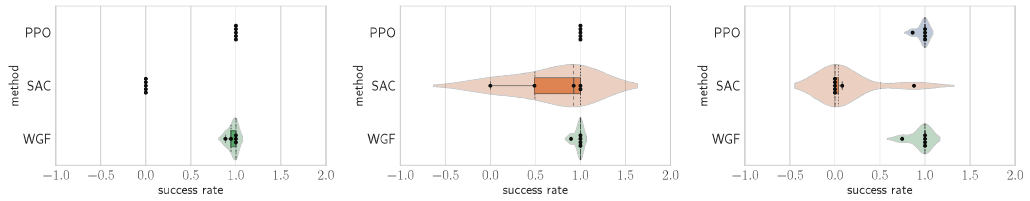


Figure 6: Variance of the success rate over the 5 runs for our method (WGF) and the two baselines on the reaching task (*left*), the collision avoidance task (*middle*) and the multiple-goal task (*right*). The violine plots are overlaid with box plots, quartile lines and a swarm plot, where dots indicate the success rates of individual runs. The time steps at which we determined the variance are for PPO, SAC and WGF for the three tasks from left to right: $(280000, 400000, 80000)$, $(275000, 300000, 90000)$, $(130000, 200000, 95000)$.

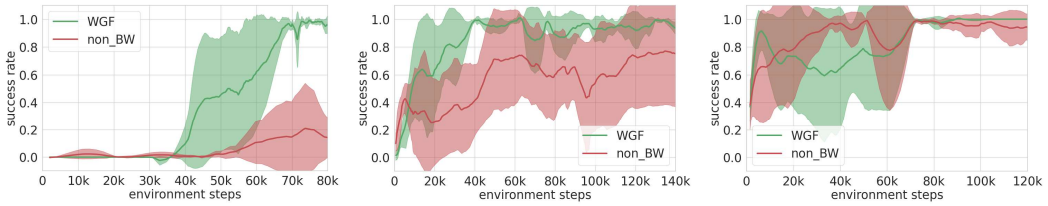


Figure 7: The success rate of our method and an ablated version, not using the Bures-Wasserstein formulation for the reaching task (*left*), the collision-avoidance task (*middle*) and the multiple-goal task (*right*). The shaded area indicates the standard deviation over 5 runs.

environment steps used for training, we chose the last time step. Specifically, we chose the following time steps for PPO, SAC and WGF, respectively: reaching task $(280000, 400000, 80000)$, collision avoidance task $(275000, 300000, 90000)$, multiple goal task $(130000, 200000, 95000)$. These plots show that PPO may also reach low-variance success rate over the five runs at the time step of convergence, at the cost of a prohibitively large number of steps. SAC showed huge variance in all tasks, apart from the reaching task, where all runs collapsed to a success rate of 0.

A.6.2 Ablation study

In order to assess the influence of leveraging a Riemannian optimization approach on the Bures-Wasserstein manifold, we conducted an ablation of our method by eliminating the Riemannian formulation. Instead of the explicit Euler scheme update in Eq. 12, which corresponds to Riemannian gradient descent w.r.t. the Bures-Wasserstein metric, we use the implicit Euler scheme

$$\hat{\boldsymbol{\mu}}_{k+1} = \arg\min_{\hat{\boldsymbol{\mu}}} \left(\frac{W_2^2(\pi_k(\hat{\boldsymbol{\mu}}), \pi_k)}{2\tau} - J(\pi_k(\hat{\boldsymbol{\mu}})) \right), \tag{42}$$

$$\hat{\boldsymbol{\Sigma}}_{k+1} = \arg\min_{\hat{\boldsymbol{\Sigma}}} \left(\frac{W_2^2(\pi_k(\hat{\boldsymbol{\Sigma}}), \pi_k)}{2\tau} - J(\pi_k(\hat{\boldsymbol{\Sigma}})) \right). \tag{43}$$

To guarantee that the updated covariance matrices do not leave the manifold of symmetric positive definite matrices, we parameterize them in terms of Cholesky factors. The results obtained with this non-Riemannian version of our method are shown in Fig. 7 in direct comparison to our method and Fig. 8 for an extended range.

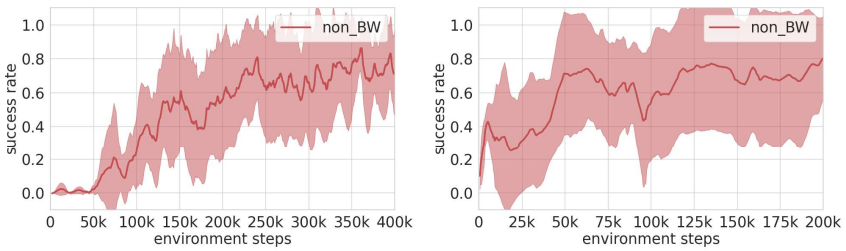


Figure 8: Extended plot of the success rate of and ablated version of our method, not using the Bures-Wasserstein-based formulation for the reaching task (*left*), the collision-avoidance task (*middle*) and the multiple-goal task (*right*). The shaded area indicates the standard deviation over 5 runs.

The results clearly show that the non-Riemannian method struggles to reach a success rate of 1 for the reaching task and the collision-avoidance task. Furthermore, we observe a high variance over different runs in the same settings (see Fig. 9 and Fig. 10). We attribute this to the fact that the our method takes exact gradient steps in the direction of steepest descent w.r.t. the underlying BW metric, whereas the implicit scheme only approximates this direction. For this reason the non-Riemannian method is much more noisy, which in turn leads to the aforementioned high variance. Nevertheless, the multiple-goal task constitutes an exception. Here we observed a similar performance for our approach and the ablated method. The reason for this is that the optimization of this task is mainly dominated by the weight updates, which are identical for both methods. This result is therefore expected and confirms that correctness of our ablation strategy.

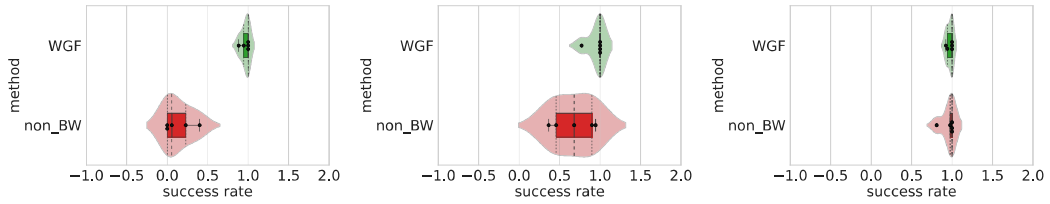


Figure 9: Variance of the success rate over 5 runs for our method (WGF) and the ablated method (non-BW) on the reaching task (*left*), the collision avoidance task (*middle*) and the multiple-goal task (*right*). The violine plots are overlaid with box plots, quartile lines and a swarm plot, where dots indicate the success rates of individual runs. The time steps at which we determined the variance are $80000, 90000, 85000$.

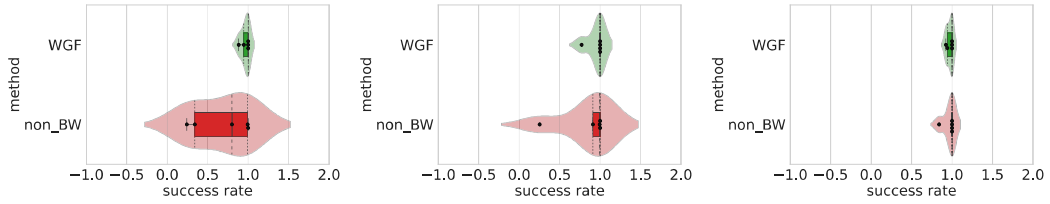


Figure 10: Variance of the success rate over 5 runs for our method (WGF) and the ablated method (non-BW) on the reaching task (*left*), the collision avoidance task (*middle*) and the multiple-goal task (*right*). The violine plots are overlaid with box plots, quartile lines and a swarm plot, where dots indicate the success rates of individual runs. The time steps at which we determined the variance are $(80000, 400000), (90000, 200000), (85000, 90000)$.

A.6.3 Additional experiments with 7-DoF robotic manipulator

We carried out two additional experiments to show that our method can be employed on tasks performed by off-the-shelf robotic manipulators (e.g. a 7-DoF Franka Emika Panda robot). Specifically, we extended the collision-avoidance task described in § 4 to a 3D environment, where the state and action depend on the task space representation. The first task was represented in the robot operational space, i.e., the state $\boldsymbol{s} = \boldsymbol{x} \in \mathbb{R}^3$ and the action $\boldsymbol{a} = \dot{\boldsymbol{x}} \in \mathbb{R}^3$. The second task was represented in the robot joint space, consequently the state $\boldsymbol{s} = \boldsymbol{q} \in \mathbb{R}^7$ and the action $\boldsymbol{a} = \dot{\boldsymbol{q}} \in \mathbb{R}^7$. This experiment was aimed at assessing the capabilities of our approach to adapt robot motion policies in state-action spaces of higher dimensions. The initial 3-components GMM policy was trained using 10 human demonstrations featuring linear reaching 3D trajectories. For policy optimization, we used a sparse reward defined as a function of the position error between the robot end-effector position and the target at the end of the rollout. Moreover, two sparse penalty terms were added to punish collision with obstacles and divergent trajectories.

Similarly to the planar task reported in the main paper, we tested whether our method was able to adapt a trajectory tracking skill in order to avoid collisions with newly added obstacles. This means that the robot end-effector needed to pass through a narrow path between two spherical obstacles. For the operational space representation, the robot end-effector pose was controlled using a full-pose Cartesian velocity controller at a frequency of 100Hz, where the end-effector orientation was kept constant. For the joint space representation, the robot joint configuration was controlled using a joint velocity controller at a frequency of 100Hz. Figure 11 shows the results for 3D operational space representation, where our method reached a success rate of 1.0 very quickly, taking approximately

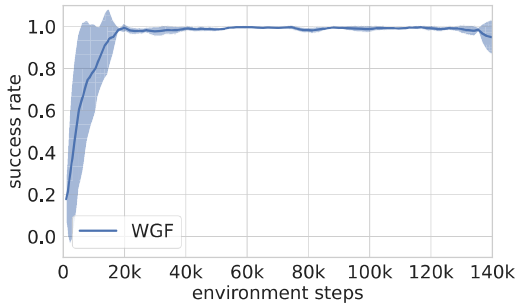


Figure 11: The success rate of our method applied to the 3D narrow-path task performed by the 7-DoF Panda robotic manipulator. The shaded area indicates the standard deviation over 5 runs.

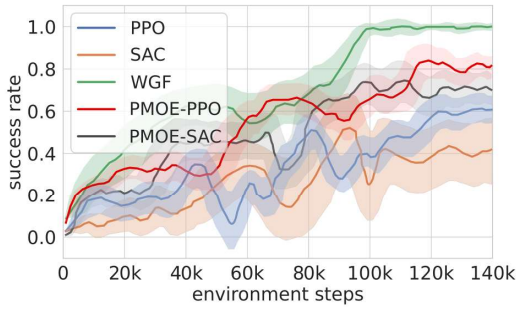


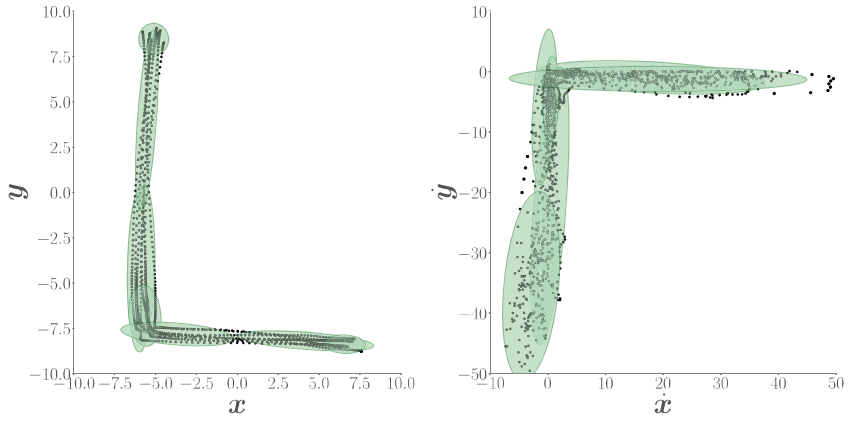
Figure 12: The success rate of our method (WFG) and the baselines on a 3D narrow-path task performed by a simulated 7-DoF Panda robotic manipulator. The task is represented in the robot joint space, therefore the state $\boldsymbol{s} = \boldsymbol{q} \in \mathbb{R}^7$, and the action $\boldsymbol{a} = \dot{\boldsymbol{q}} \in \mathbb{R}^7$, correspond to the joint position and velocity, respectively. The shaded area indicates the standard deviation over 15 runs.

20000 environment steps[3]. Moreover, the solution variance of our method was also very low, which is consistent with our observations concerning the performance of our policy optimization on the three planar tasks analyzed in the main paper. Figure 12 shows the results of the collision avoidance task using the joint space representation. As observed, our approach was able to adapt the robot motion policy so that the robot end-effector safely passes through a narrow path defined by two spherical obstacles (the narrow-path task description is provided in Sec. 4.1). It is evident that our approach outperformed all the baselines in this simulated robotic task, providing evidence that our approach scales and it is able to adapt robot motion policies in higher-dimensional tasks.

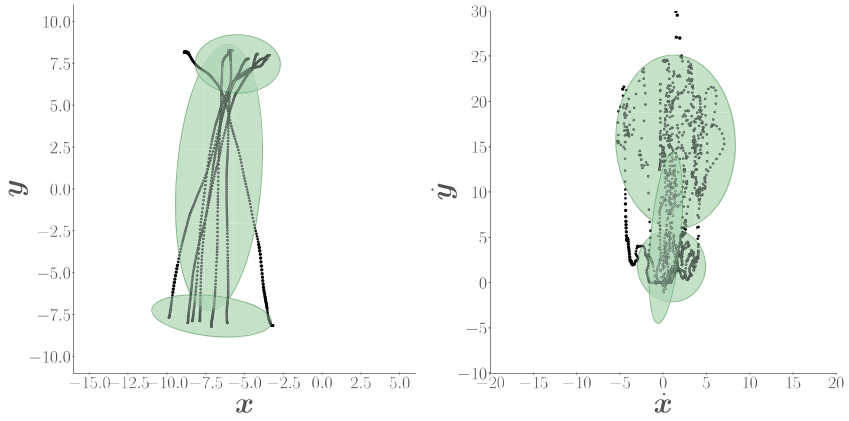
A.6.4 Initial GMM policies

For the sake of completeness, Fig. 13 provides 2D projections of the initial GMM policies learned from demonstrations for the three robotic settings considered in the main paper: the reaching motion skill, the collision-free trajectory tracking, and the multiple-goal task. Figure 13 also provides the demonstration data used to train the initial policies. Note that these models are then adapted according to the policy optimization approach introduced in § 3.2.

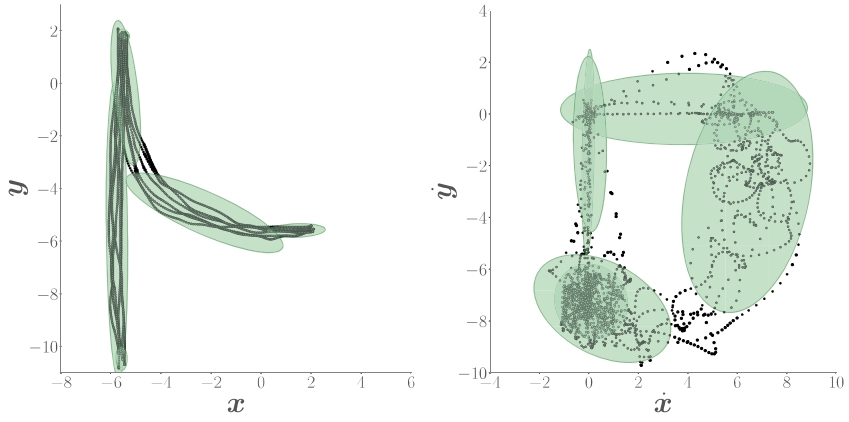
[3] As the baselines underperformed in the 2D case, they were not tested in this specific setting.



(a) Reaching motion skill



(b) Collision-free trajectory tracking



(c) Multiple-goal task

Figure 13: Green Gaussian components ( ) represent the initial GMM policy learned from demonstrations, projected on the Cartesian position (*left*) and velocity (*left*) spaces. The recorded position and velocity data are depicted as black dots (•).