Supplementary Material

A. Proofs and derivations

A.1. Proof of thm. 1

Let $V = \mathbb{R}^n$ and $\rho\colon V \to \mathbb{R}_{\geq 0}$ be a probability density on V. Let G be a group acting on V and let $R\colon G \to GL(n), g \to R_g$ be a representation of G over V. As V is finite-dimensional every R_g is represented by a matrix and thus $\det R_g$ is well-defined. Furthermore, for a function $f \in C^1(\mathbb{R}^n, \mathbb{R}^m)$ let $J_f(x) \in \mathbb{R}^{n\times m}$ denote its Jacobian evaluated at x and define the push-forward density of ρ along a diffeomorphism $f \in C^1(V, V)$ by $\rho_f(x) := \rho(f^{-1}(x)) \left|\det J_{f^{-1}}(x)\right|$.

Lemma 1. *Let $A \in GL(n)$, if $\rho(Ax) = \rho(x)$ for all $x \in V$, then* $\det A \in \{-1, 1\}$

Proof. Set $a\colon V \to V, x \mapsto Ax$. By substituting $y = a^{-1}x$ we get

$$\begin{aligned}
1 &= \int_V \rho(x)dx \\
&= \int_{a^{-1}(V)} \rho(a(y)) \left|\det A\right| dy \\
&= \int_V \rho(y) \left|\det A\right| dy \\
&= \left|\det A\right| \underbrace{\int_V \rho(z)dy}_{=1} \\
&= \left|\det A\right|
\end{aligned}$$

□

Let $G > H$ and $h \in H$. From Lemma 1 we get $\det R_h \in \{-1, 1\}$ for each $h \in H$. Define the transformation $T_h\colon V \to V,\ x \mapsto R_h x$. If $f \in C^1(V, V)$ is H-equivariant, it means $f \circ T_h = T_h \circ f$ for each $h \in H$. If ρ is an G-invariant density it means $\rho \circ T_g = \rho$. Together with the lemma we obtain

$$\begin{aligned}
\rho_f(R_h x) &= \rho_f(T_h(x)) \\
&= \rho_f(T_h(x)) \underbrace{\left|\det J_{T_h}(x)\right|}_{=\left|\det R_h\right|=1} \\
&= \rho_{T_{h^{-1}} \circ f}(x) \\
&= \rho_{f \circ T_{h^{-1}}}(x) \\
&= \rho((T_h \circ f^{-1})(x)) \left|\det J_{T_h \circ f^{-1}}(x)\right| \\
&= (\rho \circ T_h \circ f^{-1})(x) \left|\det J_{T_h}(f^{-1}(x)) J_{f^{-1}}(x)\right| \\
&= (\rho \circ f^{-1})(x) \left|\det J_{T_h}(f^{-1}(x))\right| \left|\det J_{f^{-1}}(x)\right| \\
&= \rho(f^{-1}(x)) \left|\det R_h\right| \left|\det J_{f^{-1}}(x)\right| \\
&= \rho(f^{-1}(x)) \left|\det J_{f^{-1}}(x)\right| \\
&= \rho_f(x)
\end{aligned}$$

A.2. Proof of thm. 2

Proof. Let $h \in H$ and R_h be its representation. Let v be an H-equivariant vector field. Then

$$\begin{aligned} F_{v,T}(R_h z) &= R_h x_{v,z}(0) + \int_0^T dt\ v(R_h x_{v,z}(t), t) \\ &= R_h x_{v,z}(0) + \int_0^T dt\ R_h v(x_{v,z}(t), t) \\ &= R_h \left(x_{v,z}(0) + \int_0^T dt\ v(x_{v,z}(t), t) \right). \end{aligned}$$

This implies that the bijection $F_{v,T}$ for each $T \in [0, \infty)$ given by solving

$$\begin{aligned} x_{v,z}(0) &= z \\ \frac{d}{dt} x(t) &= v(x_{v,z}(t), t) \end{aligned}$$

is H-equivariant. □

A.3. Invariant prior density

Subtracting the CoM of a system $x \in \mathbb{R}^{N \cdot D}$ and obtaining a CoM-free $\tilde{x}$, can be considered a linear transformation

$$\tilde{x} = Ax$$

with

$$A = \mathbf{I}_D \otimes \left(\mathbf{I}_N - \tfrac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right)$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{1}_k$ the k-dimensional vector containing all ones.

A is a symmetric projection operator, i.e. $A^2 = A$ and $A^T = A$. Furthermore rank $[A] = (N-1)D$. Finally, we have $Ay = y$ for each $y \in U$.

If we equip $\mathbb{R}^n$ with an isotropic density $\rho = \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$, this implies the subspace density $\tilde{\rho} = \mathcal{N}(\mathbf{0}, A\mathbf{I}_n A^T) = \mathcal{N}(\mathbf{0}, AA^T)$. Thus, sampling from ρ and projecting by A achieves sampling from $\tilde{\rho}$ trivially. On the other hand, if we have $y \in U$, then $\|y\|_2^2 = \|Ay\|_2^2$ and thus $\rho(y) = \tilde{\rho}(y)$.

If f is an equivariant flow w.r.t. symmetries (S1-3) we see that any CoM-free system is mapped onto another CoM-free system and thus defines a well-defined flow on the subspace spanned by A.

A.4. Derivations for the RBF gradient field

We first show that v as defined in (1) is indeed a gradient field. Define

$$\alpha(x, a, b) = \sqrt{\frac{\pi a b^2}{2}} \operatorname{erf}\left(\frac{x-b}{\sqrt{2a}} \right) - a \exp\left(-\frac{(x-b)^2}{2a} \right), \tag{2}$$

where erf denotes the Gaussian error function. Then we have

$$\frac{\partial \alpha(x, a, b)}{\partial x} = \exp\left(-\frac{(x-b)^2}{2a} \right) \cdot x. \tag{3}$$

Now by setting

$$\kappa(d) = \frac{1}{2} \begin{pmatrix} \alpha(d, \mu_1, \sigma_1) & \dots & \alpha(d, \mu_M, \sigma_M) \end{pmatrix} \tag{4}$$

and

$$\tilde{\Phi}(d_{ij}, t) = R(t) W \kappa(d_{ij}) \tag{5}$$

we obtain

$$\frac{\partial\tilde{\Phi}(d_{ij},t)}{\partial x_i} = R(t)W\frac{\partial\kappa(d_{ij})}{\partial d_{ij}}\frac{\partial d_{ij}}{\partial x_i} \tag{6}$$

$$= R(t)W\frac{\partial\kappa(d_{ij})}{\partial d_{ij}}\frac{r_{ij}}{d_{ij}} \tag{7}$$

$$= \frac{1}{2}R(t)WK(d_{ij})r_{ij} \tag{8}$$

where

$$K(d) = \left(\ \exp\left(-\tfrac{(d-\mu_1)^2}{2\sigma_1}\right)\quad\dots\quad\exp\left(-\tfrac{(d-\mu_M)^2}{2\sigma_M}\right)\ \right). \tag{9}$$

Similarly, we have

$$\frac{\partial\tilde{\Phi}(d_{ji},t)}{\partial x_i} = -\frac{1}{2}R(t)WK(d_{ji})r_{ji} \tag{10}$$

$$= \frac{1}{2}R(t)WK(d_{ij})r_{ij}. \tag{11}$$

Thus, we obtain

$$\left(\nabla_x\Phi(x,t)\right)_i = \frac{\partial\Phi(x,t)}{\partial x_i} \tag{12}$$

$$= \sum_{lj}\frac{\partial\tilde{\Phi}(d_{lj},t)}{\partial x_i} \tag{13}$$

$$= \sum_{j}R(t)WK(d_{ij})r_{ij} \tag{14}$$

$$= \sum_{j}\phi(d_{ij})r_{ij} \tag{15}$$

Finally, by using $v(x,t) = \frac{\partial x(t)}{\partial t} = \nabla_x\Phi(x,t)$ we can compute the divergence as

$$\operatorname{div} v(x,t) = \operatorname{tr}\left[\frac{\partial v(x,t)}{\partial x}\right] \tag{16}$$

$$= \sum_i \operatorname{tr}\left[\frac{\partial v_i(x,t)}{\partial x_i}\right] \tag{17}$$

$$= \sum_{ij}\operatorname{tr}\left[\frac{\partial\phi(d_{ij})}{\partial d_{ij}}r_{ij}\frac{\partial d_{ij}}{\partial x_i}^T + \phi(d_{ij})\frac{\partial r_{ij}}{\partial x_i}\right] \tag{18}$$

$$= \sum_{ij}\frac{\partial\phi(d_{ij})}{\partial d_{ij}}\operatorname{tr}\left[r_{ij}\frac{r_{ij}}{d_{ij}}^T\right] + \phi(d_{ij})\operatorname{tr}\left[I_{D\times D}\right] \tag{19}$$

$$= \sum_{ij}\frac{\partial\phi(d_{ij})}{\partial d_{ij}}\frac{r_{ij}^T r_{ij}}{d_{ij}} + \phi(d_{ij})D \tag{20}$$

$$= \sum_{ij}\frac{\partial\phi(d_{ij})}{\partial d_{ij}}d_{ij} + \phi(d_{ij})D. \tag{21}$$

B. Additional experiments

B.1. Comparison with other equivariant gradient flows

Next to the equivariant flow as proposed in the main text (furthermore referred to as *Kernel Flow*) we experimented with two neural-network-based equivariant flow architectures relying on the CNF framework. Both performed inferior in terms of accuracy and execution time.

One ad-hoc way to model a rotation and permutation invariant potential Φ matching the given target systems could be given by only feeding pairwise particle distances to Φ. Similarly to the architecture presented in the main text, we can model $\tilde{\Phi}(d_{ij}(t), t)$, where $\Phi(x(t)) = \sum_{ij} \tilde{\Phi}(d_{ij}(t), t)$ and then take its gradient field for the CNF. To this end we embed the distances with Gaussian RBF-kernels and use a simple multi-layer perceptron to obtain a scalar output for each embedded distance. We refer to this flow as *simple gradient flow*.

Another invariant, possibly more complex, potential Φ can be obtained by using a molecular message passing architecture similar to *SchNet* (Schütt et al., 2017). We refer to this flow as *gradient flow with SchNet*. For the implementation details see C.2.

As before, the Hutchinson estimator is used during training, while sampling and reweighing is done brute-force.

In this first additional experiment we compare training / sampling wall-clock-times for the equivariant gradient flows and the kernel flow. To this end we measure the time per iteration for training / sampling with a batch / sample size of 64. Our results show that kernel flows are considerably faster during training and at least one order of magnitude faster during sampling (Table 2). The speedup during training is manly rooted in AD required to calculate $\nabla_{x(t)}\Phi(x(t))$. The overhead during sampling is a consequence of brute-force computing $\Delta_{x(t)}\Phi(x(t))$.

Table 2. Training and sampling wall-clock-times in seconds: displayed are means and standard deviation over 10 epochs with 10 iterations each and a batch size of 64.

	TRAINING		SAMPLING	
MODEL	DW-4	LJ-13	DW-4	LJ-13
KERNEL FLOW	$\mathbf{0.879 \pm 0.012}$	$\mathbf{0.498 \pm 0.010}$	$\mathbf{0.222 \pm 0.003}$	$\mathbf{0.178 \pm 0.002}$
SIMPLE GRADIENT FLOW	2.58 ± 0.091	1.760 ± 0.056	4.18 ± 0.112	6.670 ± 0.269
GRADIENT FLOW WITH SCHNET	6.140 ± 0.583	4.770 ± 0.500	7.690 ± 0.018	25.400 ± 3.690

B.2. Density estimation with coupling flows and other equivariant gradient flows

In a second additional experiment we compare density estimation of the equivariant flow as proposed in the main text to other equivariant gradient flows and flows based on non-equivariant coupling layers. For the coupling layers we use real NVP (rNVP) transformations as introduced by (Dinh et al., 2016) and used for Boltzmann Generators in (Noé et al., 2019). The experiment follow section 7.3. As coupling layers do not conserve the CoM we add a Gaussian distributed CoM to the training data. We further use a Gaussian prior with the same CoM in the latent space. Equivariant / CoM-preserving flows will not change the CoM. Thus running an equivariant flow on the CoM-free system and then adding then energy of the CoM perturbation yields the negative log-likelihood corresponding to running the rNVP flow on the perturbed system.

The other equivariant gradient flows achieve similar albeit slightly larger log-likelihoods on the training and test set (Figure 8). The non equivariant flow based on rNVP achieves a log-likelihood on the test similar to the non equivariant nODEs (Figure 9) when data augmentation is applied. Without data augmentation the log-likelihood of the rNVP between train and test set differs in the range of ~ 200.

B.3. Energy based training for the DW-4 system

This experiment follows a similar training procedure as done in (Noé et al., 2019). The models are pretrained with *ML-training* on 1000 samples from a long MCMC trajectory. Then the we use a combination of *ML-* and *KL-training* (see sec 2 bottom or (Noé et al., 2019)), where we increase the proportion of the KL-loss from $\lambda = 0$ to $\lambda = 0.5$ over the course of training. Aftwerwards we sample $10,000$ samples and compare their energy and reweighted energy to the expected energy of the samples from the long MCMC trajectory (Figure 10). The non-equivariant models without data augmentation are capable of producing samples with low energies, but the reweighing fails in these cases (Figure 10 d, f). This is due to mode collapsing onto a small part of the target space, i.e. most of the produced samples are only from a single rotation/permutation of a certain configuration. Hence, a random sample lying in another region will yield a very large reweighing weight. This behaviour was also observed in (Noé et al., 2019) and gets worse with larger proportions of the KL-loss. The non-equivariant models with data augmentation produce samples with high energies making reweighing difficult as well (Figure 10 e, g). These models are prone to mode collapse as well if the proportion of the KL-loss becomes larger than $\lambda = 0.5$. We thus chose $\lambda = 0.5$ throughout the reported experiments. In contrast, the equivariant models (kernel flow, simple gradient flow

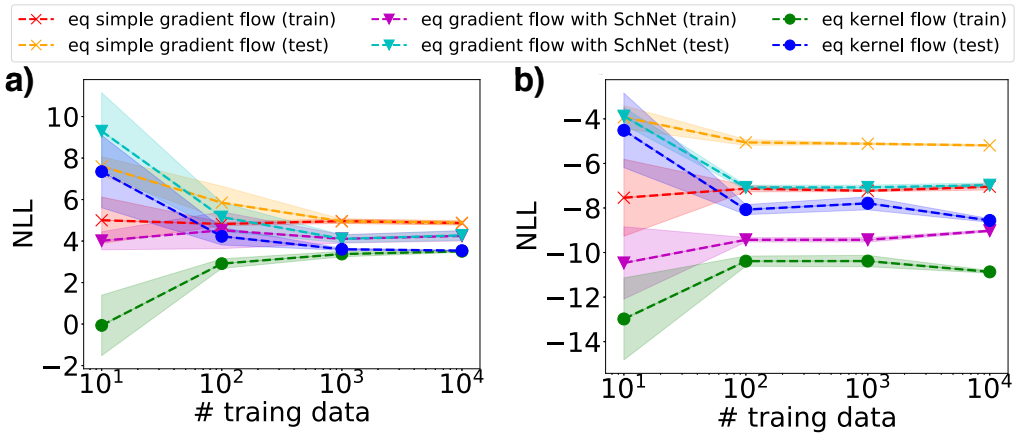

Figure 8. Log-likelihood on train and test data for both **a)** the DW-4 and **b)** the LJ-13 system after training on an increasing number of data points. All equivariant flows generalizes quickly to unseen trajectories, but the kernel flow achieves the lowest log-likelihoods on the train and test data.

and the gradient flow with SchNet) produce low energies and allow for correct reweighing (Figure 10 a, b, c).

B.4. Visualisation of the weights of the kernel flow

Due to the simple structure of the kernel flow we can visualize the learned dynamics of the equivariant flow, by plotting W after training (Figure 11).

C. Technical details

In this section we show the hyperparameters and optimization details used for the experiments presented in this work. For all experiments we used ML-training to train the models on the given training data ($\lambda = 0$) if not stated otherwise. For the discovery of new meta-stable states (7.5) for the *DW-4* system, where we used a combination of *ML-training* and *KL-training* with $\lambda = 0.5$ after pretraining both models with *ML-training*.

C.1. Equivariant kernel flow

For the *DW-4* system we fixed 50, 10 kernel means $\mu_{K,l}$, $\mu_{R,l}$ equispaced in $[0, 8]$, $[0, 1]$ for distances and times respectively. The bandwidths $\gamma_{K,l}, \gamma_{R,l}$ of the kernels have been initialized with 0.5, 0.3 and were optimized during the training process. The total model ended up having 620 trainable parameters.

For the *LJ-13* system we fixed 50 kernel means $\mu_{K,l}$ in $[0, 16]$ concentrated around $r_m = 1$ with increasing distance to each other towards the interval bounds. Similarly bandwidths $\gamma_{K,l}$ are initialized narrowly close to r_m increasing towards the interval bounds. We placed the 10 kernels $\mu_{R,l}$ for the time-dependent component equispaced in $[0, 1]$. The bandwidths $\gamma R, l$ where initialized with narrower bandwidths around $t = 0.5$ and smearing out the closer they reach the interval boundaries. Again bandwidths were optimized during the training process. This resulted in a total of 620 trainable parameters.

As regularization is important to efficiently train our architecture using fixed step-size solvers our models were optimized using `AdamW`, a modified implementation with fixed weight-decay (Loshchilov & Hutter, 2017) using a learning rate of 0.005, weight decay of 0.01 and a batch size of 64 samples until convergence.

C.2. Equivariant gradient flows

We used the same distance embedding for both equivariant gradient flows.

For the *DW-4* system we fixed 50 rbf-kernel means equispaced in $[0, 8]$. The bandwidths have been initialized with 0.5 and were not optimized during the training process.

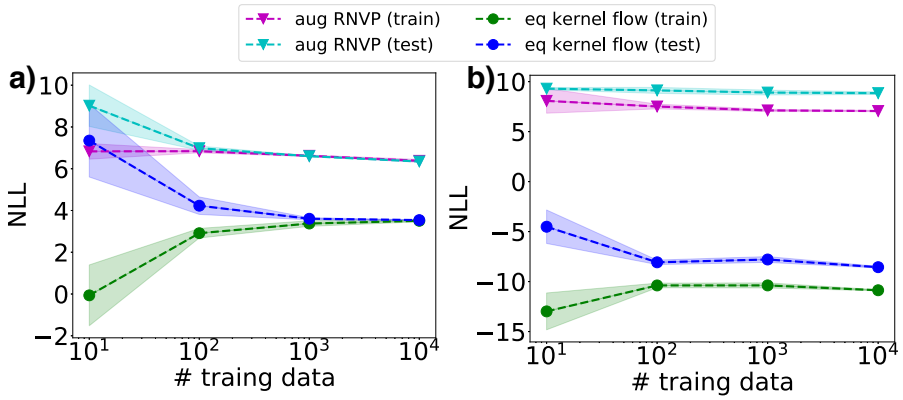

Figure 9. Log-likelihood on train and test data for both **a)** the DW-4 and **b)** the LJ-13 system after training on an increasing number of data points. The non-equivariant coupling flow with data augmentation (*aug RNVP*) performs significantly worse compared to the equivariant flow on both test systems. For the LJ-13 system it is even unable to fit the augmented data at all and remains close to the prior.

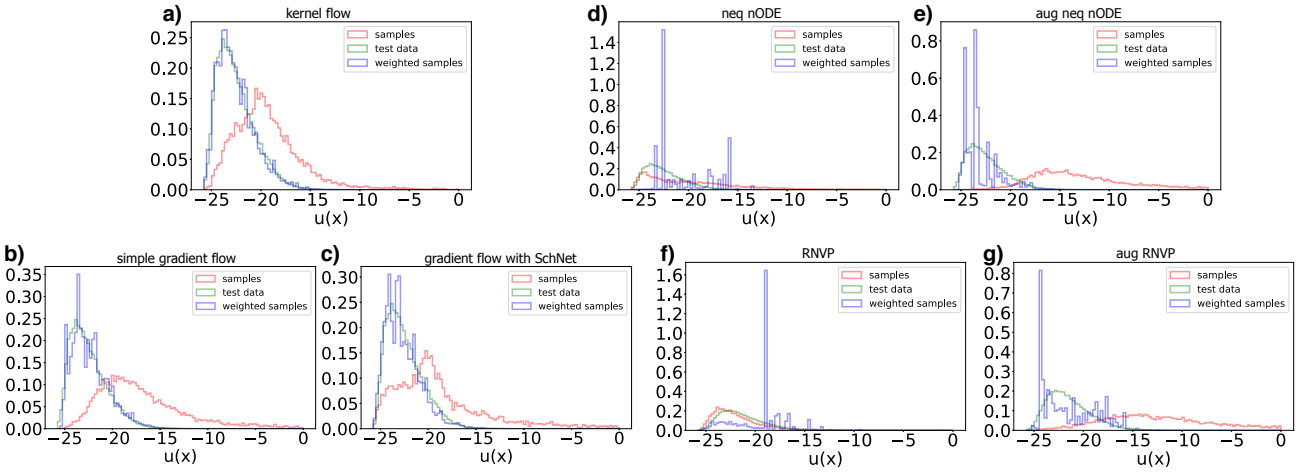

Figure 10. Energy histograms for samples from the DW-4 system with different models.

For the *LJ-13* system we fixed 50 rbf-kernel means in $[0, 16]$ concentrated around $r_m = 1$ with increasing distance to each other towards the interval bounds. Similarly bandwidths $\gamma_{K,l}$ are initialized narrowly close to r_m increasing towards the interval bounds and were not optimized during the training process..

For the equivariant gradient flows with the simple potential the transformation of each embedded distance was modeled with a dense neural network with layer sizes $[50, 64, 32, 1]$ and `tanh` activation functions. This resulted in a total of 5377 trainable parameters.

For the equivariant gradient flows with the SchNet (Schütt et al., 2017) inspired potential we used 16 features and 3 interaction blocks. For the feature encoding we used a simple network with layer sizes $[1, 16]$. For the continuous convolutions in the three interaction blocks we used dense neural networks with layer sizes $[50, 32, 32, 16]$ and `tanh` activation functions. Finally, a dense neural networks with layer sizes $[16, 8, 4, 1]$ was used to compute the invariant energy after the interaction blocks. This resulted in a total of 9857 trainable parameters.

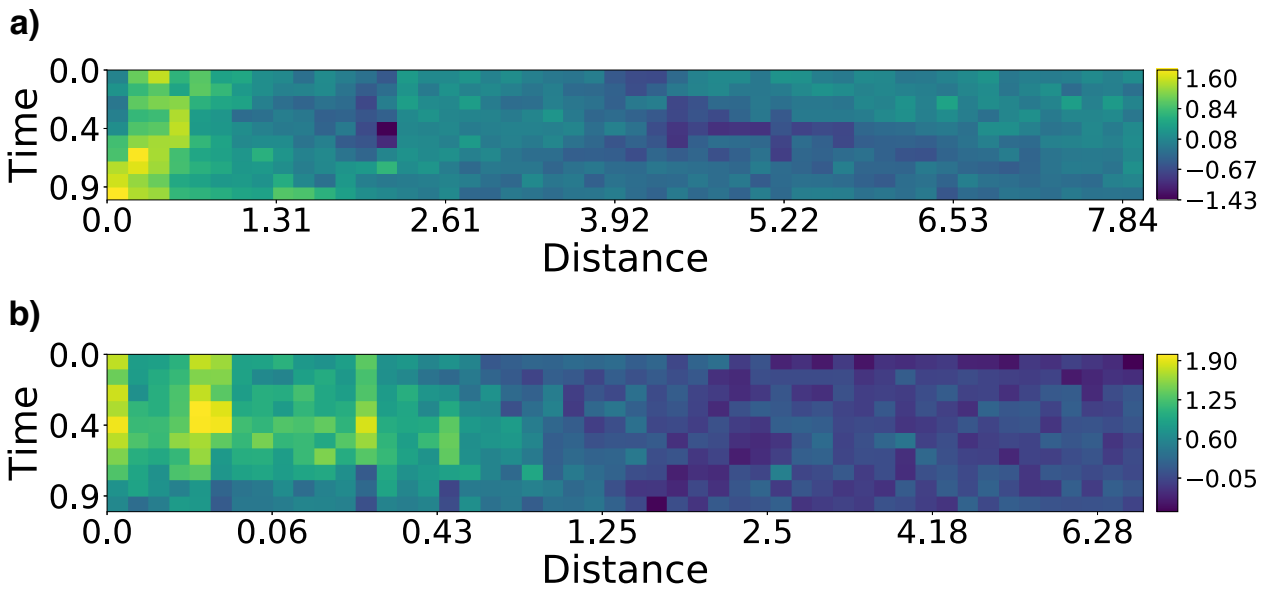

Figure 11. Kernel weights W visualized after training for the **a)** DW-4 system and the **b)** LJ-13 system.

C.3. Non equivariant nODE flow

For the *DW-4* system we used a dense neural network with layer sizes $[64, 64]$ and `tanh` activation functions. This resulted in a total of 5256 trainable parameters.

For the *LJ-13* system we used a dense neural network with layer sizes $[64, 128, 64]$ and `tanh` activation functions. This resulted in a total of 21671 trainable parameters. For the optimization we used `AdamW` with a learning rate of 0.005. We optimized the model with a batch size of 64 samples until convergence.

C.4. MCMC trajectories

For each system, a training and a test trajectory were obtained with Metropolis Monte-Carlo, where we optimized the width of the Gaussian proposal density by maximizing $\alpha \cdot s$, with α being the acceptance rate computed from short trajectories and s the Gaussian standard deviation (step size). The optimal step sizes are $s = 0.5$ for the $DW - 4$ system and $s = 0.025$ for the $LJ - 13$ system. To ensure that all samples steam from the equilibrium distribution we discard a large number of initial samples. For the *DW-4* system the initial 1000 samples are discarded, while we discard 20000 for the *LJ-13* system.

C.5. Non equivariant coupling flows

For the *DW-4* system we used 8 coupling blocks. For the translation transformation we used a dense neural network with layer sizes $[4, 64, 64, 4]$ and `ReLU` activation functions. For the scaling transformation we used a dense neural network with layer sizes $[4, 64, 64, 4]$ and `tanh` activation functions. This resulted in a total of 21576 trainable parameters.

For the *LJ-13* system we used 16 coupling blocks. For the translation transformation we used a dense neural network with layer sizes $[19/20, 64, 64, 20/19]$ and `ReLU` activation functions. For the scaling transformation we used a dense neural network with layer sizes $[19/20, 64, 64, 20/19]$ and `tanh` activation functions. The number of input and output neurons for each network is either 19 or 20 due to the uneven number of total dimensions. This resulted in a total of 215680 trainable parameters.

C.6. Benchmark systems

Throughout all experiments we chose the same parameters for our two benchmark systems.

For the *DW-2 / DW-4* system we chose $a = 0, b = -4, c = 0.9, d_0 = 4$ and a dimensionless temperature factor of $\tau = 1$.

For the *LJ-13* system we chose $r_m = 1, \epsilon = 1$ and a dimensionless temperature factor of $\tau = 1$.

C.7. Error bars

Error bars in all plots are given by one standard deviation.

In Figure 3 a) we show errors for 1000 estimations per particle count. In Figure 3 b) errors are displayed for 100 reweighed bootstrapped sub-samples. In Figure 3 c) time was measured for 100 estimations per particle count per method.

In Figure 4 a) we show 3 runs per method.

In Figure 5, Figure 8, and Figure 9 we show 5 runs per model/system/training set size.

C.8. Computing infrastructure

All experiments were conducted on a *GeForce GTX 1080 Ti* with 12 GB RAM.