Coarse-Graining of Molecular Dynamics Using Neural Ordinary Differential Equations

Jakub Lála
Department of Materials
Imperial College London
London, United Kingdom
jakub.lala18@imperial.ac.uk

Stefano Angiolleti-Uberti
Department of Materials
Imperial College London
London, United Kingdom
s.angioletti-uberti@imperial.ac.uk

***Abstract*—Coarse-graining (CG) improves computational feasibility of simulating complex molecular systems (e.g. proteins, or polymer chains) by reducing the number of degrees of freedom considered. Here, we collect many particles making up a composite body to a single centre of mass and orientation. Defining the CG interaction potential between the bodies that minimizes loss of information is non-trivial with no clear analytical solution. We use neural ordinary differential equations (ODE) to learn such CG potentials in a data-driven manner. We show a proof-of-concept application on a toy problem and outline the next steps towards an automated CG software pipeline.**

***Index Terms*—deep learning, neural ordinary differential equations, molecular dynamics, coarse-graining, rigid body dynamics**

I. Introduction

Molecular dynamics (MD) experiments attempt to understand macroscopic material properties by running computer simulations according to microscopic physical laws. To find a thermodynamic average of a property, interatomic potential energy functions have been empirically fine-tuned to describe the particles' interactions.

Machine learning (ML) potentials, specifically neural network (NN) potentials, provide a data-driven approach to defining these energy landscapes. NNs interpolate the potential from datasets without the need for any manual fine-tuning. Since their first use with Behler-Parinello NNs in 2007 [1], success has been shown with the popular graph NN architecture called SchNet [2], or the more recent state-of-the-art approaches of Allegro [3] and NequIP [4].

As the feasibility to simulate molecular systems depends on the system's size and complexity, coarse-graining (CG) of the system's coordinates is an effective way of reducing the dimensionality of the problem and thus its computational demand. As one reduces the representation, a new CG potential must be defined as a function of the coarser set of variables. For instance, a protein backbone or DNA could be modelled as a chain of beads, rather than on the atomistic resolution. Systems such as DNA nanostars [5], carbon nanotubes [6], or colloidal systems [7] (shown in Fig. 1) all exhibit a vast amount of degrees of freedom (df), limiting the computationally feasible system sizes. Deep learning efforts such as CGNets [8], VAMPnets [9], or autoencoders [10] show promising approximations of effective CG potentials.

Learning an ML potential usually involves an energy- or a force-matching procedure, where frozen time snapshots of the system are sampled and configuration-energy (or -force) pairs are used for supervised training of the network. It has been suggested that training on forces should yield better results when predicting time-dependent, dynamic behaviour [11].

Greener et al. [12] use recurrent neural networks (RNNs) to train a model that parametrizes a physics-informed CG force fields of proteins. They show that differentiating MD and backpropagating gradients through the time-evolved trajectory leads to an efficient way to update a lot of parameters at once. Nevertheless, they are limited by the trajectory length due to GPU memory constraints, as they are required to store intermediate values during the forward pass. Moreover, they impose an inductive bias on the functional form of the potential, having a pair-wise distance, bond angle and torsion angle components, rather than letting the NN define its own internal structure for the potential.

We address both of these limitations and use a novel method of neural ordinary differential equations (ODEs) [13] to MD. Neural ODEs advance RNNs by reducing the demand for memory. During the backward pass, they compute gradients without backpropagating through the ODE solver. That means the model trains with constant memory cost, allowing us to integrate long trajectories. So far they have shown success across a variety of tasks, including Hamiltonian dynamics with control [14], computational fluid modelling [15], spintronic experiments [16] and chemical kinetics [17].

Wang et al. [10] were the first ones to use neural ODEs within the context of MD simulations. They use control theory to bias dynamics towards a target folded state of a protein, defined by a set of macroscopic observables (e.g. torsion angles along the protein chain). Although this method enables learning control protocols that could be implemented experimentally, the target state may be unphysical, making this application unpractical for traditional MD simulations, where one samples physical thermodynamic averages.

This paper describes a general method to learn CG potentials for any rigid body, regardless of its shape and complexity. The minimum number of df to describe a rigid body is only six - three for the position of the centre of mass (COM) and three for the orientation. Thus, by collecting MD trajectories

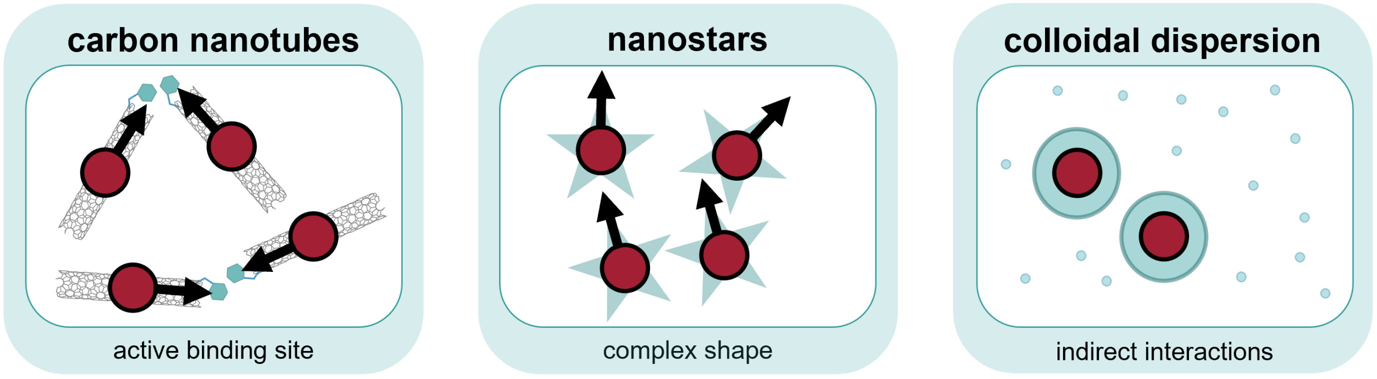

Fig. 1. Various systems may have their representation coarse-grained to fewer df, hence enabling faster computation times. In the schematics, the red circle represents the centre of mass and the black arrow the body's orientation. A carbon nanotube with an active binding site or a nanostar with a complex shape can both be interpreted as a rigid body with a centre of mass position and orientation only. Thus, for a system of N bodies, one only needs $6N$ df, compared to simulating the composite bodies at their atomistic resolution. The interaction between such bodies is then given by a new coarse-grained potential V_{CG} with no trivial analytical form. A different example of coarse-graining is a colloidal dispersion of larger particles surrounded by smaller particles, where the larger particles indirectly interact due to entropic effects caused by the smaller ones. Rather than simulating all the particles, the aggregate effect of the smaller particles can be described by V_{CG}. Here, we may even omit the rotation of the large particles if they are symmetrical, hence reducing the df even further.

of any molecule, composite particle, nanoparticle, etc., we are able to train a numerical approximation of the effective CG potential without the need to know the exact analytical form.

After describing our approach in Section II, we benchmark the method on a toy problem of two 7-particle bodies. We give the results, discuss the limitations and outline further improvements in Section III. We also consider the prospects of learning pair-body potentials and how it allows for cheaper computation when simulating larger systems with more bodies.

II. Method

A. Neural Ordinary Differential Equations

Neural ODEs come from the seminal paper by Chen et al. [13]. They are a continuous limit extension of RNNs and parametrize an ODE as

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t, \theta) \tag{1}$$

where $\mathbf{z}$ is a state variable, f is the neural network parametrized by θ and t is time.

It is straightforward to propagate a trajectory forward in time using any ODE solver. However, computing gradients with respect to θ is not. To overcome this, the *adjoint sensitivity* method must be used during the backward pass.

The adjoint $\mathbf{a}(t) = \frac{\partial L}{dt}$ is an instantaneous analogue of the chain rule, where $L(\mathbf{z}(t_f), \hat{\mathbf{z}}(t_f))$ is the loss between the predicted and true final state of the trajectory, $\mathbf{z}(t_f)$ and $\hat{\mathbf{z}}(t_f)$ respectively. It is governed by

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}} \tag{2}$$

The gradients used to update θ are then given by

$$\frac{\partial L}{\partial \theta} = \int_{t_f}^{t_0} \mathbf{a}(t)^\top \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt \tag{3}$$

For more detailed derivation, see the original paper [13].

B. Rigid Body Dynamics

We use rigid body dynamics governed by the Hamiltonian

$$H(\mathbf{x}, \mathbf{p}, \mathbf{q}, \mathbf{l}) = V_{CG}(\mathbf{x}, \mathbf{q}) + K_{\text{trans}}(\mathbf{p}) + K_{\text{rot}}(\mathbf{l}) \tag{4}$$

where $\mathbf{x}$ and $\mathbf{p}$ are the positions and momenta of COM, respectively. $\mathbf{q}$ are the quaternions of the rigid bodies, and $\mathbf{l}$ are their angular momenta given in the body-fixed coordinates. We use quaternions over Euler angles as Euler angles are prone to geometrical singularities and are computationally less efficient.

Using the derivation from Shivarama et al. [18], the complete set of Hamilton's equation to evolve body i is

$$\frac{d\mathbf{x}_i}{dt} = \frac{\mathbf{p}_i}{m_i} = \mathbf{v}_i \tag{5}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial V_{\text{CG}}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{q}_i, \mathbf{q}_j)}{\partial \mathbf{x}_i} = \mathbf{F}_{ij} \tag{6}$$

$$\frac{d\mathbf{q}_i}{dt} = \frac{1}{2}\mathbf{G}^\top \mathbf{J}_i^{-1} \mathbf{l}_i \tag{7}$$

$$\frac{d\mathbf{l}_i}{dt} = -\mathbf{\Omega} \mathbf{l}_i - \frac{1}{2}\mathbf{G}\frac{\partial V_{CG}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{q}_i, \mathbf{q}_j)}{\partial \mathbf{q}_i} \tag{8}$$

where m_i is the body's mass, $\mathbf{J}_i$ is the matrix with its principal moments of inertia, $\mathbf{v}_i$ is its COM velocity, $\mathbf{F}_{ij}$ is the force acted on it by body j,

$$\mathbf{G} = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{9}$$

and $\mathbf{\Omega}$ is defined by the time derivative as

$$\mathbf{\Omega} = 2\mathbf{G}\dot{\mathbf{G}}^\top \tag{10}$$

Note that both the translational and rotational kinetic components of H, i.e. K_{trans} and K_{rot} respectively, can be explicitly evolved by (5) and (7). The NN then models the CG potential V_{CG} as a function of $\mathbf{x}$ and $\mathbf{q}$ only, summarized in Fig. 2.

Instead of the Runge-Kutta solver used in the original neural ODE paper [13], we use the Velocity-Verlet integrator as it

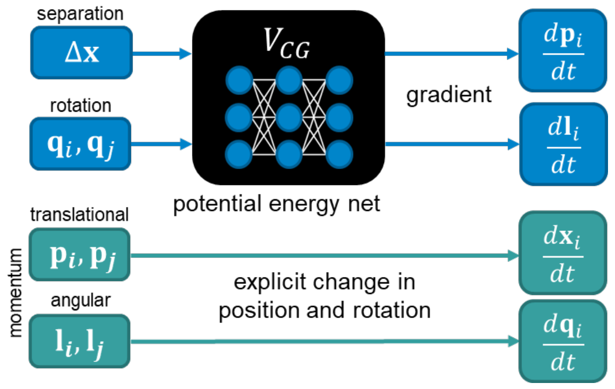
separation
$\Delta\mathbf{x}$
rotation
$\mathbf{q}_i, \mathbf{q}_j$
V_{CG}
potential energy net
gradient
$\frac{d\mathbf{p}_i}{dt}$
$\frac{d\mathbf{l}_i}{dt}$
momentum
translational
$\mathbf{p}_i, \mathbf{p}_j$
angular
$\mathbf{l}_i, \mathbf{l}_j$
explicit change in position and rotation
$\frac{d\mathbf{x}_i}{dt}$
$\frac{d\mathbf{q}_i}{dt}$

Fig. 2. In a single time step for body i, the momenta are explicitly evolved according to Hamilton's equations in a physics-informed fashion. The CG potential energy net is a function of 11 variables: 3 for the separation vector $\Delta\mathbf{x}$, 4 for the quaternion $\mathbf{q}_i$ and 4 for the quaternion $\mathbf{q}_j$. Note that we set the COM position of i to the origin to further reduce the necessary df for the NN input. To obtain the ODEs from (7) and (8), we take the gradient of V_{CG} with respect to $\mathbf{x}_i$ and $\mathbf{q}_i$ using `PyTorch` [19], which conveniently finds the gradient using the computation graph.

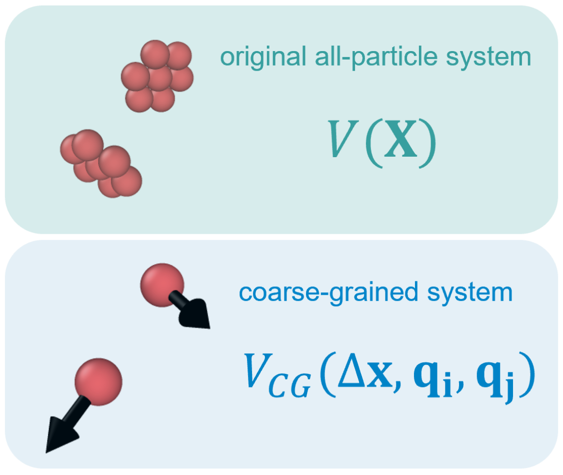
original all-particle system
$V(\mathbf{X})$
coarse-grained system
$V_{CG}(\Delta\mathbf{x}, \mathbf{q_i}, \mathbf{q_j})$

Fig. 3. Schematic of the two 7-particle hexagons, comparing the full (top) and CG representations (bottom). The hexagon consists of a central particle surrounded by 6 particles at a distance of σ^{LJ} from (20). In the atomistic resolution, all particles' positions $\mathbf{X}$ need to be considered, whereas after coarse-graining one considers only the CG coordinates.

is time-reversible and energy-conserving even with long time steps - a crucially desired property when simulating in the microcanonical ensemble. More specifically, to resemble the ODE solver from `LAMMPS` [20], we implement the Richardson iteration. Below we give the algorithm to evolve in time by Δt. Note that whenever we update a quaternion, it is followed by a normalisation step as $\mathbf{q} \rightarrow \frac{\mathbf{q}}{||\mathbf{q}||}$.

1) translational and angular momentum at half-step ('half-kick', $\mathbf{x}_i$ and $\mathbf{q}_i$ constant)

$$\mathbf{p}_i\left(t_{1/2}\right) = \mathbf{p}_i(t_0) + \frac{\Delta t}{2}\frac{d\mathbf{p}_i}{dt}\bigg|_{t=t_0} \tag{11}$$

$$\mathbf{l}_i\left(t_{1/2}\right) = \mathbf{l}_i(t_0) + \frac{\Delta t}{2}\frac{d\mathbf{l}_i}{dt}\bigg|_{t=t_0} \tag{12}$$

2) centre of mass position at full-step ('translational drift', free flight with $\mathbf{p}_i$ constant)

$$\mathbf{x}_i\left(t_1\right) = \mathbf{x}_i(t_0) + \frac{\Delta t}{m_i}\mathbf{p}_i\left(t_{1/2}\right) \tag{13}$$

3) Richardson iteration for quaternion leapfrog ('rotational drift', free rotation with $\mathbf{l}_i$ constant)
   a) full-step Richardson update

$$\mathbf{q}_i^{\text{full}}(t_1) = \mathbf{q}_i(t_0) + \frac{\Delta t}{2}\mathbf{G}^\top\mathbf{J}^{-1}\mathbf{l}_i\left(t_{1/2}\right) \tag{14}$$

   b) half-step Richardson update

$$\mathbf{q}_i^{\text{half}}\left(t_{1/2}\right) = \mathbf{q}_i(t_0) + \frac{\Delta t}{4}\mathbf{G}^\top\mathbf{J}^{-1}\mathbf{l}_i\left(t_{1/2}\right) \tag{15}$$

   c) re-compute body-fixed $\mathbf{l}_i(t_{1/2})$ at $\mathbf{q}_i^{\text{half}}(t_{1/2})$ from system-fixed $\tilde{\mathbf{l}}_i(t_{1/2})$ at $\mathbf{q}_i(t_0)$
   d) second half-step Richardson update

$$\mathbf{q}_i^{\text{half}}\left(t_1\right) = \mathbf{q}_i^{\text{half}}\left(t_{1/2}\right) + \frac{\Delta t}{4}\mathbf{G}^\top\mathbf{J}^{-1}\mathbf{l}_i\left(t_{1/2}\right) \tag{16}$$

   e) corrected Richardson update

$$\mathbf{q}_i(t_1) = 2\mathbf{q}_i^{\text{half}}\left(t_1\right) - \mathbf{q}_i^{\text{full}}(t_1) \tag{17}$$

4) translational and angular momentum at full-step ('half-kick', $\mathbf{x}_i$ and $\mathbf{q}_i$ constant)

$$\mathbf{p}_i\left(t_1\right) = \mathbf{p}_i\left(t_{1/2}\right) + \frac{\Delta t}{2}\frac{d\mathbf{p}_i}{dt}\bigg|_{t=t_{1/2}} \tag{18}$$

$$\mathbf{l}_i\left(t_1\right) = \mathbf{l}_i\left(t_{1/2}\right) + \frac{\Delta t}{2}\frac{d\mathbf{l}_i}{dt}\bigg|_{t=t_{1/2}} \tag{19}$$

C. *Toy Problem*

Our system consists of two 7-particle hexagons (shown in Fig. 3), where each particle with unit mass m_p acts with a purely repulsive Lennard-Jones potential given by

$$V_{LJ}(r) = 4\epsilon^{LJ}\left[\left(\frac{\sigma^{LJ}}{r}\right)^{12} - \left(\frac{\sigma^{LJ}}{r}\right)^{6}\right] \quad r < r_c \tag{20}$$

where r is the particles' separation, ϵ^{LJ} is the depth of the potential well (set to 1), σ^{LJ} is the distance at which the potential energy is zero (or equivalently the size of the particle; set to 1) and r_c is the cutoff radius (set to $2^{\frac{1}{6}}$).

To sample the phase space of interest, i.e. when the two bodies are interacting, we add a harmonic restraint between the COMs given by

$$V_{\text{ext}}(\Delta x) = \frac{1}{2}k(\Delta x - \Delta x_0)^2 \tag{21}$$

where $\Delta x = ||\mathbf{x}_j - \mathbf{x}_i||$, k is the spring constant (set to $6T/r_c^2$), and Δx_0 is the equilibrium distance (set to 2). The parameters are defined such that there is $3k_BT$ potential energy when the bodies stop interacting. When training, we add V_{ext} to the overall potential acting on the bodies, hence the NN only learns the difference, which is the desired V_{CG}.

D. Training

We use LAMMPS [20] and its RIGID package to sample 10 trajectories of the two hexagons interacting. These trajectories are split into training, test and validation datasets with the ratio of 80%-10%-10% respectively. Each trajectory runs for 10^7 steps, where each time step is $10^{-5}\,\tau_{\text{unit}}$, but we only log every 100th step and thus the logged time step in the datasets is $10^{-3}\,\tau_{\text{unit}}$. We initialise each trajectory with a canonical ensemble simulation for $10^5\,\tau_{\text{unit}}$ with a random velocity assignment, defined by the temperature (set to 0.5).

The validation dataset is used to find suitable hyperparameters using SigOpt [21]. The net has 68-48-32 neurons and uses the tanh activation function in the hidden layers. The output layer is a single linear neuron. We train with Adamax optimizer with a learning rate of 0.02 and we use a batch size of 10 000 initial conditions. The loss function is defined as the mean-squared error (MSE) of the final state as

$$L(\mathbf{x},\mathbf{p},\mathbf{q},\mathbf{l}) = \frac{1}{13N}\sum_{n}^{N}\Big((\mathbf{x}_n-\hat{\mathbf{x}}_n)^2+(\mathbf{p}_n-\hat{\mathbf{p}}_n)^2 \tag{22}$$

$$+(\mathbf{q}_n-\hat{\mathbf{q}}_n)^2+(\mathbf{l}_n-\hat{\mathbf{l}}_n)^2\Big) \tag{23}$$

where we take the mean over all outputs n from a single batch of N trajectories. The hat symbol refers to the true final states and the factor of 13 comes from adding together the dimensions of the four variables, i.e. $3+3+4+3=13$.

Initially, the training iterations involve propagating trajectories for 20 time steps. This is done in order to learn some initial dynamics first. We then increase the number of time steps by 10 every 100 epochs, up until we have trained for 500 epochs in total.

The PyTorch [19] code used in this paper extends the adjoint method from [10] and is available at [22].

III. Results & Discussion

Fig. 4 shows predicted trajectories using V_{CG}. Note that we do not plot predicted trajectories from the test dataset, as our current solution fails to generalize well.

Fig. 5 shows the energy landscape when the hexagons interact within the xy-plane. We would expect a flat landscape with an elliptic hill in the middle, nevertheless, our solution also shows other peaks and is not very smooth. Note, however, that direct xy-plane interactions area statistically unlikely to appear in the dataset, hence this is an extrapolation from interactions at other than direct angles.

Below we discuss the shortcomings of our approach and the future prospects towards an automated CG pipeline. Note that the project is currently work in progress and the majority of the advancements were made in implementing rigid body MD into neural ODEs. Nevertheless, even though our solution does not compare with all-particle LAMMPS simulations in terms of the accuracy or computational efficiency, we produce this paper as a proof-of-concept of this neural ODE application.

A. Generazibility

Accuracy on the test dataset could be improved by finding a more suitable set of hyperparameters - primarily the NN architecture, including activation functions and widths of each layer. V_{CG} of the toy problem should be relatively easy to learn, hence one should expect a small net to approximate the solution well. As always, more training data should provide more configurations to sample a general V_{CG}, nevertheless, efforts should be made to minimise the amount of training data necessary to reduce the initial computation demand of the all-particle simulation.

The potential energy landscape shows a lot of jagged regions, suggesting our net either overfits or is insufficiently small to define a smooth interpolation. L2 regularization, temporal regularization [23] or both, should be employed to improve generalizability. Temporal regularization also speeds up training convergence. One may then also fine-tune the time step Δt and trajectory length to manage the trade-off between efficiency and accuracy.

B. Loss Function Choice

Our simple approach to define the loss function as the MSE of the final state could be revised. Firstly, as the loss of quaternions will always be smaller due to their unit magnitude, one should normalise each variable by its distribution within the dataset. Another approach might be normalising the distance and velocities by the size of the simulation box, or one could only consider the relative separation and velocities between the bodies to avoid the effects of the box size completely.

Secondly, a more robust method of evaluating the divergence should be implemented. In some of our experiments, we tried using the MSE of all states within a predicted trajectory, which sometimes produced better results, but was fundamentally insufficient as a metric for measuring divergence, as some models minimised the test error when $V_{CG}=0$ for all configurations, i.e. the energy landscape was flat. This also highlights that a clearer divergence metric, such as the Lyapunov divergence [24], should be used to evaluate model accuracy. Furthermore, the accuracy could then also be assessed on more general simulation metrics, for example the pair-correlation function, rather than naïvely evaluating the loss function on the test dataset.

Also, note that the choice of the loss function may primarily depend on the objective of our simulations. If one wants to primarily predict the system's energy, then one could define the loss as a function of energies leading to a more accurate approximation. Moreover, as the loss can be a function of any thermodynamic variable, one could experiment with functions of force or even the pair-correlation function.

C. Optimization

In terms of compute time, both training and prediction times are fairly high compared to the well-optimized LAMMPS. For instance, training the model took 16 hours on a single GPU (NVIDIA Quadro P4000). The major bottleneck compared to the original neural ODE implementation is the use of

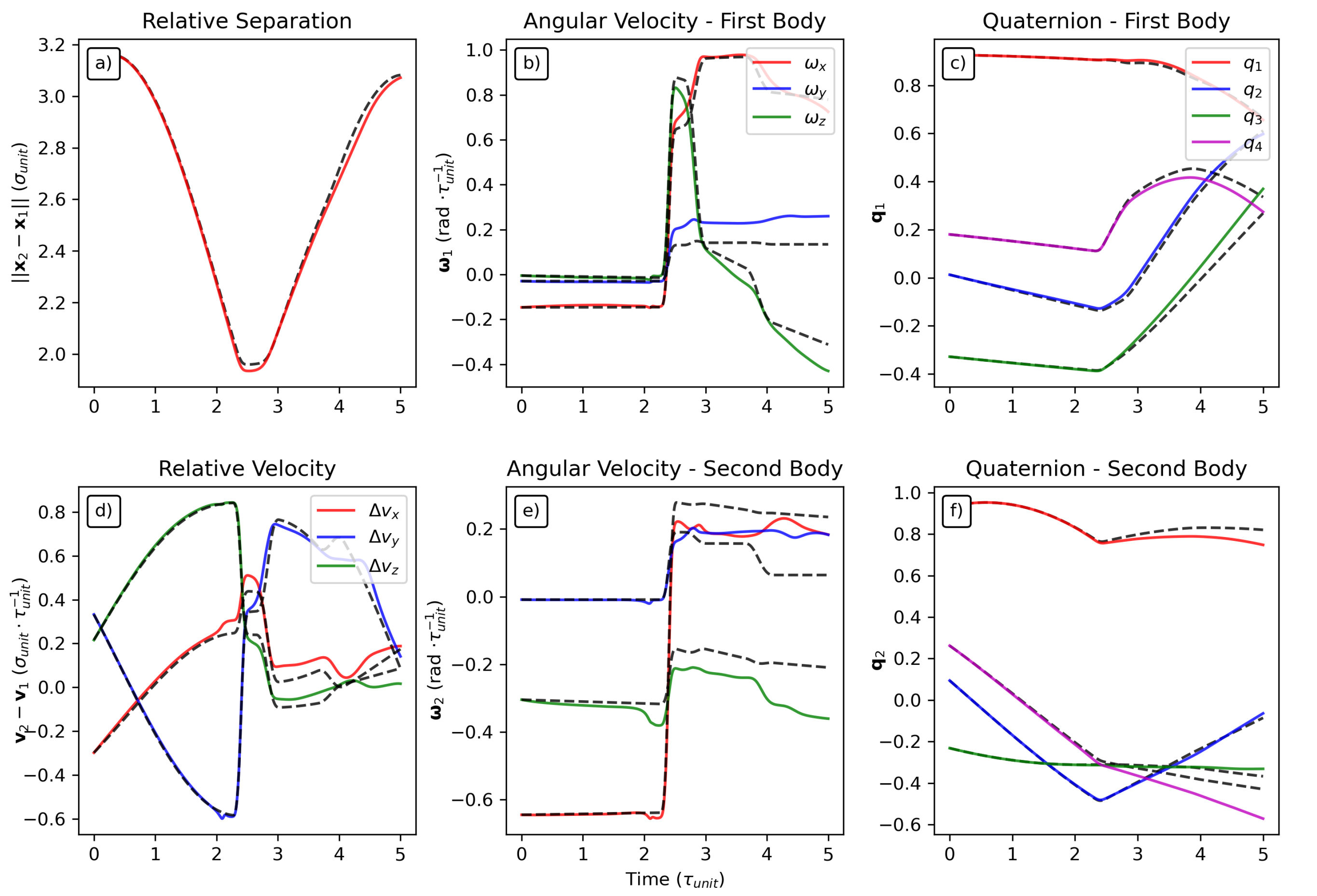

Fig. 4. Predicted trajectory of the two hexagons using the learned V_{CG}. The initial conditions are taken from the training dataset. The dashed lines represent the true trajectories. The reduced units of distance and time are given by $\sigma_{\text{unit}} = \sigma^{LJ}$ and $\tau_{\text{unit}} = \sigma^{LJ} \cdot \sqrt{m_p/\epsilon^{LJ}}$ respectively. We are able to train a potential that somewhat approximates the collisions between the bodies, although, at longer trajectories, our numerical solution starts accumulating substantial error. Note that the maximum trajectory length trained on is 100 time steps, whereas here we plot 5 000 time steps, suggesting that one can learn dynamics on a long time-scale even with many short trajectories.

quaternions. Here note that we have purposely chosen this trade-off of longer computation in order to avoid memory overflow. Nevertheless, such memory issues may arise only with a certain set of molecular systems, hence those will be the ones taking full advantage of our method. Such systems will presumably involve more complex bodies in terms of shape and surface structure. The usefulness of this method will thus come to test only when simulating more complex systems such as carbon nanotubes, nanostars, or colloidal dispersions mentioned above.

D. Further Work

As it is likely that the approximation will always yield to some numerical error, especially when the error accumulates for longer trajectories, one may employ Hamiltonian Monte Carlo (HMC) [25] to correct for the deviation from the ergodic ensemble. This involves taking a Monte Carlo acceptance step after several dynamic time steps were taken. The Master's thesis that laid the foundations for this paper suggested the use of neural HMC as an effective way to improve sampling [26].

Moreover, further effort should be given to incorporating our `PyTorch` models within `LAMMPS`. Packages such as `ML-IAP` and `ML-SNAP` do just that, although more work is needed to integrate CG potentials parameterised by body orientation. A future outlook is to have a fully automated coarse-grainer with an adjustable trade-off between accuracy and efficiency, where lower accuracy would allow for more rapid experimentation.

Finally, in more complex systems, we also need to consider the temperature of the training data, as the effective intermolecular forces will be temperature-dependent. To elaborate, composite particles at different temperatures might have different variances in their spatial configurations, thus the effective interaction between the bodies is also different. Our toy problem was defined in such a way that there were no thermal fluctuations between the particles within a single hexagon.

IV. Conclusion

This paper showed the proof-of-concept application of neural ODEs to learning CG potentials. We generated full-atomistic MD trajectories of composite bodies to train an

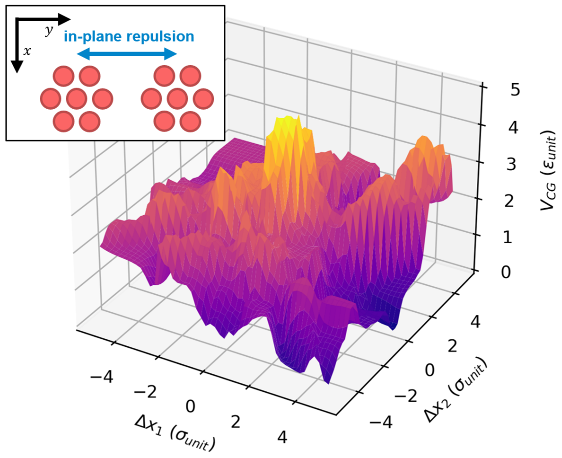

Fig. 5. Potential energy landscape of V_{CG} for in-plane interaction, given in the inset schematic. The inputs to V_{CG} are thus $[\Delta x_1, \Delta x_2, 0]$, $[1, 0, 0, 0]$ and $[1, 0, 0, 0]$, where we scan over Δx_1 and Δx_2 values, i.e. we scan across the x and y spatial dimensions, and then the quaternions represent no rotation, i.e. the hexagons lie flat in the xy-plane. The reduced energy ϵ_{unit} equals ϵ^{LJ}.

effective CG potential as a function of the minimum df for a rigid body. We confirmed the feasibility on a toy problem. Our learned CG potential showed promising results predicting trajectories from the training dataset, but so far failed to perform well on the test dataset. Regularization methods were proposed to generalise our model. Further accuracy and efficiency improvements have also been suggested in order to properly compare this method to the more traditional ML potentials, or to full-atomistic MD simulations themselves.

References

[1] J. Behler and M. Parrinello, “Generalized neural-network representation of high-dimensional potential-energy surfaces,” *Physical Review Letters*, vol. 98, no. 14, Apr. 2007. [Online]. Available: https://doi.org/10.1103/physrevlett.98.146401

[2] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, and K.-R. Müller, “SchNetPack: A deep learning toolbox for atomistic systems,” *Journal of Chemical Theory and Computation*, vol. 15, no. 1, pp. 448–455, Nov. 2018. [Online]. Available: https://doi.org/10.1021/acs.jctc.8b00908

[3] A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, “Learning Local Equivariant Representations for Large-Scale Atomistic Dynamics,” *arXiv e-prints*, p. arXiv:2204.05249, Apr. 2022.

[4] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, “E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials,” *Nature Communications*, vol. 13, no. 1, May 2022. [Online]. Available: https://doi.org/10.1038/s41467-022-29939-5

[5] S. Naskar, D. Bhatia, S.-T. Lin, and P. K. Maiti, “A minimal coarse-grained model to study the gelation of multi-armed dna nanostars,” 2021. [Online]. Available: https://arxiv.org/abs/2110.11251

[6] N. M. Uddin, F. M. Capaldi, and B. Farouk, “Molecular dynamics simulations of carbon nanotube dispersions in water: Effects of nanotube length, diameter, chirality and surfactant structures,” *Computational Materials Science*, vol. 53, no. 1, pp. 133–144, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927025611004332

[7] X. Shen and I. C. Bourg, “Molecular dynamics simulations of the colloidal interaction between smectite clay nanoparticles in liquid water,” *Journal of Colloid and Interface Science*, vol. 584, pp. 610–621, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021979720313527

[8] J. Wang, S. Olsson, C. Wehmeyer, A. Pérez, N. E. Charron, G. de Fabritiis, F. Noé, and C. Clementi, “Machine learning of coarse-grained molecular dynamics force fields,” *ACS Central Science*, vol. 5, no. 5, pp. 755–767, Apr. 2019. [Online]. Available: https://doi.org/10.1021/acscentsci.8b00913

[9] A. Mardt, L. Pasquali, H. Wu, and F. Noé, “VAMPnets for deep learning of molecular kinetics,” *Nature Communications*, vol. 9, no. 1, Jan. 2018. [Online]. Available: https://doi.org/10.1038%2Fs41467-017-02388-1

[10] W. Wang and R. Gómez-Bombarelli, “Coarse-graining auto-encoders for molecular dynamics,” *npj Computational Materials*, vol. 5, no. 1, Dec. 2019. [Online]. Available: https://doi.org/10.1038/s41524-019-0261-5

[11] F. Noé, A. Tkatchenko, K.-R. Müller, and C. Clementi, “Machine learning for molecular simulation,” *Annual Review of Physical Chemistry*, vol. 71, no. 1, pp. 361–390, Apr. 2020. [Online]. Available: https://doi.org/10.1146/annurev-physchem-042018-052331

[12] J. G. Greener and D. T. Jones, “Differentiable molecular simulation can learn all the parameters in a coarse-grained force field for proteins,” *PLOS ONE*, vol. 16, no. 9, p. e0256990, Sep. 2021. [Online]. Available: https://doi.org/10.1371%2Fjournal.pone.0256990

[13] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, “Neural ordinary differential equations,” 2018. [Online]. Available: https://arxiv.org/abs/1806.07366

[14] Y. D. Zhong, B. Dey, and A. Chakraborty, “Symplectic ode-net: Learning hamiltonian dynamics with control,” 2019. [Online]. Available: https://arxiv.org/abs/1909.12077

[15] K. Lee and E. J. Parish, “Parameterized neural ordinary differential equations: applications to computational physics problems,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 477, no. 2253, p. 20210162, Sep. 2021. [Online]. Available: https://doi.org/10.1098/rspa.2021.0162

[16] X. Chen, F. A. Araujo, M. Riou, J. Torrejon, D. Ravelosona, W. Kang, W. Zhao, J. Grollier, and D. Querlioz, “Forecasting the outcome of spintronic experiments with neural ordinary differential equations,” *Nature Communications*, vol. 13, no. 1, Feb. 2022. [Online]. Available: https://doi.org/10.1038/s41467-022-28571-7

[17] T. Zhang, Y. Zhang, W. E, and Y. Ju, “A deep learning-based ode solver for chemical kinetics,” 2020. [Online]. Available: https://arxiv.org/abs/2012.12654

[18] R. Shivarama and E. P. Fahrenthold, “Hamilton’s equations with euler parameters for rigid body dynamics modeling,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 126, no. 1, pp. 124–130, Mar. 2004. [Online]. Available: https://doi.org/10.1115%2F1.1649977

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[20] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in ’t Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, “LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales,” *Comp. Phys. Comm.*, vol. 271, p. 108171, 2022.

[21] S. Clark and P. Hayes, “SigOpt Web page,” https://sigopt.com, 2019. [Online]. Available: https://sigopt.com

[22] J. Lála, “Coarse-Graining of Rigid Bodies in Molecular Dynamics with Neural ODEs,” 10 2022. [Online]. Available: https://github.com/jakublala/coarsegrained-md-neural-ode

[23] A. Ghosh, H. S. Behl, E. Dupont, P. H. S. Torr, and V. Namboodiri, “Steer: Simple temporal regularization for neural odes,” 2020. [Online]. Available: https://arxiv.org/abs/2006.10711

[24] I. D. J. Rodriguez, A. D. Ames, and Y. Yue, “LyaNet: A Lyapunov Framework for Training Neural ODEs,” Feb. 2022, arXiv:2202.02526 [cs]. [Online]. Available: http://arxiv.org/abs/2202.02526

[25] M. Betancourt, “A conceptual introduction to hamiltonian monte carlo,” 2017. [Online]. Available: https://arxiv.org/abs/1701.02434

[26] J. Lála, “Coarse-graining of molecular dynamics using neural ordinary differential equations,” Master’s thesis, Imperial College London, 2022.