Differentiable Molecular Simulations for Learning and Control

Wujie Wang

Department of Material Science and Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
wwj@mit.edu

Simon Axelrod
Department of Chemistry and Chemical Biology
Harvard University
Cambridge, MA 02138, USA
saxelrod@mit.edu

Rafael Gómez-Bombarelli
Department of Material Science and Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
rafagb@mit.edu

Abstract

哈密顿量描述了系统的状态及其与环境的相互作用

Molecular simulations use statistical mechanics at the atomistic scale to enable both the elucidation of fundamental mechanisms and the engineering of matter for desired tasks. Non-quantized molecular behavior is typically simulated with differential equations parameterized by a Hamiltonian, or energy function. The Hamiltonian describes the state of the system and its interactions with the environment. In order to derive predictive microscopic models, one wishes to infer a molecular Hamiltonian from macroscopic quantities. From the perspective of engineering, one wishes to control the Hamiltonian to achieve desired macroscopic structures or simulation outcomes as in self-assembly and quantum control. In both cases, the goal is to modify the Hamiltonian such that bulk properties of the simulated system match a given target. We demonstrate how this can be achieved using differentiable simulations where bulk target observables and simulation outcomes can be analytically differentiated with respect to Hamiltonians. Our work opens up new routes for parameterizing Hamiltonians to infer macroscopic models and develops control protocols

控制哈密顿量以获得所需的宏观结构或模拟结果

修改哈密顿量，使模拟系统的总体属性与给定目标匹配

1 Introduction

At the atomic level, physical processes are governed by differential equations containing many degrees of freedom. Macroscopic phenomena in matter emerge from microscopic interactions that can be simulated through numerically integrating the equations of motion. In classical simulations, these equations of motion are derived from a Hamiltonian function. In quantum simulations, they are derived from a Hamiltonian operator. Examples of microscopic quantities emerging from simulations are time series of positions, velocities, and forces on atoms and molecules. From these, a rich family

Machine Learning for Molecules Workshop at NeurIPS 2020. https://ml4molecules.github.io

of macroscopic observables can be calculated to describe the configurational and temporal correlation functions of atoms.

Classically, simulating the positions of points that preserves energy requires integrating the Hamiltonian equations of motions:

势能 位置 哈密顿量

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i} \qquad \frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \tag{1}$$

where p_i and q_i are the respective momentum and position of the i^{th} particle. H is the Hamiltonian of the systems; for conservative systems, it is given by the sum of kinetic energy and the potential energy,

对于保守系统，H是动能和势能之和。

$$H(\mathbf{p}, \mathbf{q}) = \underset{\text{势能}}{U(\mathbf{q})} + \sum_i^N \frac{p_i^2}{2m_i}, \text{第i个粒子的动能} \tag{2}$$

where boldface denotes the set of quantities for all particles, $U(\mathbf{q})$ is the potential energy and $p_i^2/(2m_i)$ is the kinetic energy of the i^{th} particle.

Simulating an entire system containing all degrees of freedom is computationally intractable. Typically one is interested in a small subset of a system, such as a molecule or protein, and concerned only with the influence of the environment on the system, but not the details of the environment itself. For this reason, one usually incorporates an environment Hamiltonian H_b with coarse-grained macroscopic variables of interest into the original Hamiltonian: $H_{tot} = H + H_b$. The inclusion of H_b is important in simulating systems under certain thermodynamic conditions. For example, the crystallization and melting of water occur under constant pressure conditions. These conditions are imposed by the environment, which must therefore be incorporated into H_b. The environment, and therefore H_b, can also be explicitly controlled and optimized in an experiment. For example, H_b can represent an external laser that is varied to control chemical reaction dynamics.

当条件是由环境施加的，因此必须纳入Hb

展示了在分子模拟的背景下使用可微模拟

我们证明了可以学习哈密顿量，使得通过模拟轨迹计算的宏观观测值与给定目标匹配。

Recent advances in differentiable solvers have shown that *differentiable simulations* may be performed, in which the result of a simulation may be analytically differentiated with respect to its inputs (1; 2; 3; 4; 5; 6). Our paper demonstrates the use of differentiable simulations in the context of molecular simulation. We show that a Hamiltonian can be learned such that the macroscopic observables computed through simulation trajectory to match a given target. This is done through automatic differentiation of the macroscopic observables computed from simulation trajectory with respect to a wide class of functional forms for the energy function including Graph Neural Networks (GNN). We show that the same principles can be used to control the system Hamiltonian to force the system towards a target state. 控制系统的哈密顿量，迫使系统走向目标状态。

2 Differentiable Molecular Simulations with Adjoint Sensitivity method

使用灵敏度方法的可微分子模拟

To be able to differentiate molecular simulations to reach a control target, we adopt the reverse-mode automatic differentiation method from Chen *et al.*, which uses adjoint sensitivity methods (7; 1). Taking derivatives requires computation of the adjoint state $a(t) = dL/d(p(t), q(t))$. Evaluating the loss requires the reverse-time integration of the vector-Jacobian product:

$$\frac{dL}{d\theta} = \int_{t_{i+1}}^{t_i} a(t) \frac{df(q(t), p(t), \theta)}{d\theta} dt, \tag{3}$$

where $f(q, p, t)$ represents the Hamiltonian ODE defined in Eq. 1. The reverse-mode automatic differentiation computes the gradient through the adjoint states without backpropagating through the fowardpass computations in the ODE solver. This has the advantage of linear memory cost as function of simulation steps because the alogrithm does not save intermediate system states but instead integrating backward in time to compute gradients of the parameters and other system state variables. The ability to output positions and momenta at individual timesteps allow one to directly compute observables and correlation functions from a trajectory. For the classical MD simulations we applied velocity verlet for the forward computation of molecular trajectory. We apply the mid-point integration for the background call to compute gradients of GNN parameters. For the optimization of the two-state quantum dynamics, we use the fourth order Runge Kutta for both of the forward and backward computation.

3 Control Protocol for Molecular Quantum Dynamics

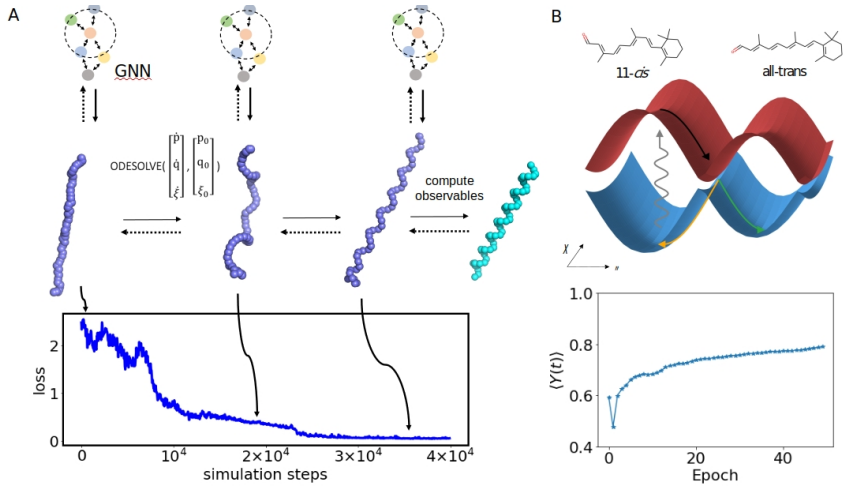

Figure 1: **A** We perform continuous model training during the simulations to bias a harmonic polymer chain toward a targeted helix shape. This is done by training a bias Hamiltonian parameterized by a GNN. We run the simulations for 4000 steps, and the loss is computed and differentiated to update GNN weights every 40 simulation steps. **B** Controlled isomerization of the model retinal Hamiltonian with a time-dependent electric field. The model consists of two electronic states, denoted with blue and red, a vibrational mode x, and a torsional mode ϕ. We show that the time-averaged quantum yield increases as a function of training epoch with differentiable control.

We use the model introduced in Ref. (8) for the retinal chromophore. The model Hamiltonian consists of two diabatic electronic states, a single torisonal mode ϕ for the isomerizing double bond, and a single stretching mode x (see Fig. 1B). Details of the model, the construction of the Hamiltonian and the operators of interest can be found in Refs. (8; 9; 10).

The total Hamiltonian of the system and the control field is given by

$$\hat{H}(t) = \hat{H}_{\mathrm{S}} + \hat{H}_b = \hat{H}_{\mathrm{S}} - \hat{\mu}E(t), \tag{4}$$

where $\hat{H}_{\mathrm{S}}$ is the system Hamiltonian, $\hat{H}_b$ is the control Hamiltonian, $\hat{\mu}$ is the dipole operator, and $E(t)$ is the electric field. The system wave function evolves under the Schrödinger equation,

$$\frac{\partial}{\partial t}\left|\psi(t)\right\rangle = -i\hat{H}(t)\left|\psi(t)\right\rangle, \tag{5}$$

where $|\psi(t)\rangle$ is the wave function. Computationally, the wave function is represented as a vector and the Hamiltonian as a matrix, and the system dynamics are obtained by solving Eq. (5). Physical quantities are obtained as expectation values of operators, $A(t) = \langle\psi(t)|\hat{A}|\psi(t)\rangle$, where $A(t)$ is an observable and $\hat{A}$ is the corresponding operator.

The quantity to be optimized is the quantum yield, i.e. the efficiency of isomerization Y (the higher the better; further details are provided in the Appendix). During simulations, we backpropagate through the simulation trajectory to optimize both the magnitude and phase of the temporal electric field. The numerical results are shown in Fig. 1 B. The quantum yield begins at approximately 0.6, and after 50 epochs reaches 0.8 as the electric field is improved. These results show that differentiable simulations can be used to learn control protocols for electric fields driven isomerization.

4 Controlling Molecular Dynamics

We trained a GNN to represent H_b, to bias a linear chain with a harmonic Hamiltonian into a helix fold (see Fig. 1). The polymer simulations are performed at a constant temperature using the Nose-Hoover Chain integrator described above and in the Appendix. We back-propagate through the simulations during the course of the simulations to continuously update the GNN so that the

loss function $L = \sum_i^{observables} (\phi_i(q(t_1)) - \phi_i(q_{helix}))^2$ is minimized where the set of functions ϕ_i include structural variables of the polymer chain: bond distances, angles and dihedrals angles.

5 Learning from Observables

We demonstrate an example of fitting pair correlations for liquid water. Pair correlation functions characterize structural and thermodynamic properties of condensed phase systems (11). We demonstrate that by differentiating through the simulation trajectories, one can actively modify parameters to match a target distributions function. To make the distribution function our differentiable target, we implement a differentiable histogram to approximate the typical non-differentiable histogram operation. This is done by summing over pair distances expanded in a basis of Gaussians ("Gaussian smearing"), followed by normalization to ensure that the histogram integration yields the total number of pair distances (see Appendix). We use a GNN to fit a Coarse-Grained Potential which reproduces the experimental oxygen-oxygen distribution (g_{oo}) from Ref. [(12)]. For each epoch, we first run simulations and backpropagate to optimize GNN weights, a potential is learned to best reproduce g_{oo}. We use the symmetric Jensen-Shannon entropy to quantify the difference between the simulated distribution function and experimental distribution, adn as the loss function for backpropatation.

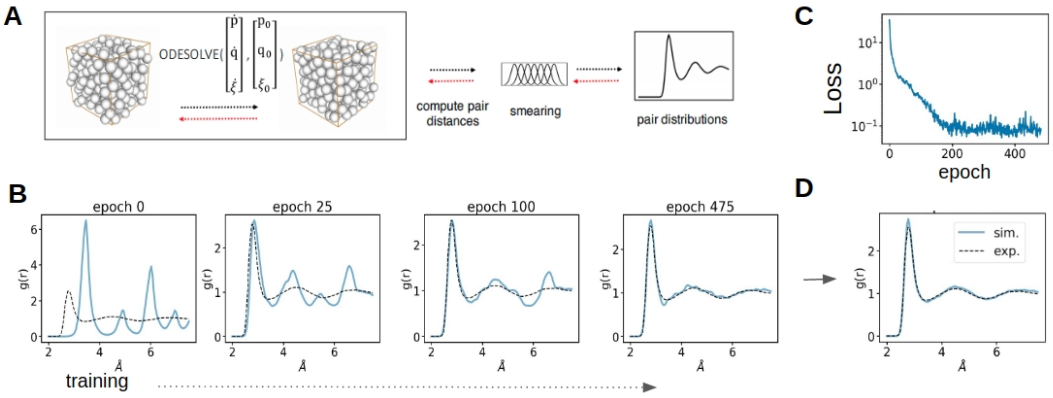

Figure 2: A. Computational workflow to fit Oxygen-Oxygen pair distribution functions for water at 298K and 1 atm. We use a Graph Neural Networks to parameterize the force fields. For each training epoch, we compute the pair distribution functions from simulated trajectories. We back-propagate the mean square loss between simulated and target pair distribution functions to update the GNN parameters. B. Change of simulated g_{oo} during training. C. The training loss curve. D. g_{oo} simulated by the converged model.

6 Conclusions

In this work we proposed a framework for training molecular simulations based on macroscopic quantities to develop learning and control protocols. Our method is based on model learning through simulation time feedback from bulk observables. Our method also opens up new possibilities for designing control protocols for equilibrium and non-equilibrium simulations by incorporating bias Hamiltonians. This work can be extended to the simulation of other types of molecular systems with different thermodynamic boundary conditions and different control scenarios.

7 Related Work

Several works have incorporated physics-based simulations to control and infer movements of mechanical objects. These are done by incorporating inductive biases that obey Hamiltonian dynamics (13; 14). Many works also focus on performing model control over dynamical systems (3; 15; 16). Differentiable simulations with automatic differentiation have also been utilized in constructing models from data in many differential equation settings like computational fluid dynamics (17), physics simulations (18; 19; 4), quantum chemistry (20), protein simulations (21) and normalizing

flows (22). Much progress has been made in developing differentiable frameworks for molecular dynamics (23), PDEs (24; 25; 26; 2) and ODEs (1).

References

[1] Chen, R. T. Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. Neural Ordinary Differential Equations. *Advances in neural information processing systems* (2018). 1806.07366.

[2] Lu, P. Y., Kim, S. & Soljačić, M. Extracting Interpretable Physical Parameters from Spatiotemporal Systems using Unsupervised Learning (2019). 1907.06011.

[3] Li, Y., He, H., Wu, J., Katabi, D. & Torralba, A. Learning Compositional Koopman Operators for Model-Based Control (2019). 1910.08264.

[4] Liang, J., Lin, M. & Koltun, V. Differentiable Cloth Simulation for Inverse Problems. *Advances in Neural Information Processing Systems 32* 771–780 (2019).

[5] Holl, P., Koltun, V. & Thuerey, N. Learning to Control PDEs with Differentiable Physics (2020). 2001.07457.

[6] Lu, L., Meng, X., Mao, Z. & Karniadakis, G. E. DeepXDE: A deep learning library for solving differential equations (2019). 1907.04502.

[7] Pontryagin, L. *Mathematical Theory of Optimal Processes* (Routledge, 2018).

[8] Hahn, S. & Stock, G. Quantum-mechanical modeling of the femtosecond isomerization in rhodopsin. *The Journal of Physical Chemistry B* **104**, 1146–1149 (2000).

[9] Tscherbul, T. V. & Brumer, P. Quantum coherence effects in natural light-induced processes: cis–trans photoisomerization of model retinal under incoherent excitation. *Physical Chemistry Chemical Physics* **17**, 30904–30913 (2015).

[10] Axelrod, S. & Brumer, P. An efficient approach to the quantum dynamics and rates of processes induced by natural incoherent light. *The Journal of chemical physics* **149**, 114104 (2018).

[11] Jones, J. E. On the Determination of Molecular Fields. II. From the Equation of State of a Gas. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **106**, 463–477 (1924).

[12] Skinner, L. B. *et al.* Benchmark oxygen-oxygen pair-distribution function of ambient water from x-ray diffraction measurements with a wide Q-range. *Journal of Chemical Physics* **138**, 074506 (2013). URL http://aip.scitation.org/doi/10.1063/1.4790861.

[13] Greydanus, S., Dzamba, M. & Yosinski, J. Hamiltonian Neural Networks (2019). 1906.01563.

[14] Sanchez-Gonzalez, A., Bapst, V., Cranmer, K. & Battaglia, P. Hamiltonian Graph Networks with ODE Integrators (2019). 1909.12790.

[15] Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D. & Kavukcuoglu, K. Interaction Networks for Learning about Objects, Relations and Physics. *Advances in Neural Information Processing Systems* 4509–4517 (2016). 1612.00222.

[16] Zhong, Y. D., Dey, B. & Chakraborty, A. Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control (2019). 1909.12077.

[17] Schenck, C. & Fox, D. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks (2018). 1806.06094.

[18] Hu, Y. *et al.* ChainQueen: A real-time differentiable physical simulator for soft robotics. In *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, 6265–6271 (Institute of Electrical and Electronics Engineers Inc., 2019). 1810.01054.

[19] Hu, Y. *et al.* DiffTaichi: Differentiable Programming for Physical Simulation (2019). 1910.00935.

[20] Tamayo-Mendoza, T., Kreisbeck, C., Lindh, R. & Aspuru-Guzik, A. Automatic Differentiation in Quantum Chemistry with Applications to Fully Variational Hartree-Fock. *ACS Central Science* **4**, 559–566 (2018). 1711.08127.

[21] Ingraham, J., Riesselman, A., Sander, C. & Marks, D. Learning Protein Structure with a Differentiable Simulator. In *International Conference on Learning Representations* (2019).

[22] Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I. & Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019* (International Conference on Learning Representations, ICLR, 2019). 1810.01367.

[23] Schoenholz, S. S. & Cubuk, E. D. JAX, M.D.: End-to-End Differentiable, Hardware Accelerated, Molecular Dynamics in Pure Python (2019). 1912.04232.

[24] Han, J., Jentzen, A. & Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences of the United States of America* **115**, 8505–8510 (2018). 1707.02568.

[25] Long, Z., Lu, Y., Ma, X. & Dong, B. PDE-Net: Learning PDEs from Data. *35th International Conference on Machine Learning, ICML 2018* **7**, 5067–5078 (2017). 1710.09668.

[26] Long, Z., Lu, Y. & Dong, B. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics* **399** (2019). 1812.04426.

[27] Nosé, S. A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics* **81**, 511–519 (1984).

[28] Martyna, G. J., Klein, M. L. & Tuckerman, M. Nosé-Hoover chains: The canonical ensemble via continuous dynamics. *The Journal of Chemical Physics* **97**, 2635–2643 (1992).

[29] Parrinello, M. & Rahman, A. Strain fluctuations and elastic constants. *The Journal of Chemical Physics* **76**, 2662–2666 (1982).

[30] Duvenaud, D. K. *et al.* Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems*, 2215–2223 (2015). 1509.09292.

[31] Zhang, L., Han, J., Wang, H., Car, R. & Weinan, E. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Physical Review Letters* **120**, 143001 (2018). 1707.09571.

[32] Yao, K., Herr, J. E., Toth, D., Mckintyre, R. & Parkhill, J. The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics. *Chemical Science* **9**, 2261–2269 (2018).

[33] Mailoa, J. P. *et al.* A fast neural network approach for direct covariant forces prediction in complex multi-element extended systems. *Nature Machine Intelligence* **1**, 471–479 (2019). 1905.02791.

[34] Paszke, A. *et al.* Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems 32*, 8024–8035 (2019).

[35] Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems (2016). 1603.04467.

8 Appendix

8.1 Nose-Hover Chain Integrator

Here we describe the Nose-Hover Chain (27; 28), the constant temperature integrator algorithm mentioned in the paper. We applied this integrator to the coarse-grained water and polymer examples to simulate systems with constant temperature control. Here we define the variables used in the integrator:

- N: number of particles
- K: number of virtual variables used in the chain
- i: index for individual degrees of freedom, $i : 1, ..., 3N$
- j: index for virtual variables in the chain $j : 1, ..., K$
- p_i: momentum for each degree of freedom i
- q_i: position for each degree of freedom i
- m_i: mass for each particle in the simulation
- Q_j: coupling strengths to the heat baths variable in the chain
- η_j: virtual momenta

The coupled equations of motion are:

$$\begin{aligned}
\frac{dp_i}{dt} &= -\frac{\partial H}{\partial q_i}, \\
\frac{dq_i}{dt} &= \frac{\partial H}{\partial p_i} - p_i \frac{\eta_1}{Q_1} \\
\frac{d\eta_1}{dt} &= \big(\sum_i^{3N} \frac{p_i^2}{2m_i} - N k_B T\big) - \eta_1 \frac{\eta_2}{Q_2} \\
&... \\
\frac{d\eta_j}{dt} &= \big(\frac{\eta_{j-1}}{Q_{j-1}} - k_B T\big) - \eta_j \frac{\eta_{j+1}}{Q_{j+1}} \\
&... \\
\frac{d\eta_K}{dt} &= \big(\frac{\eta_{j-1}}{Q_{j-1}} - k_B T\big)
\end{aligned} \tag{6}$$

The Nose Hover Chain integrator performs effective temperature control, and the integrated dynamics sample the Boltzmann distribution. The integrator is deterministic and time-reversible. The control of other thermodynamic variables can be realized with other integrator protocols, such as the Rahman-Parrinelo method to maintain constant pressure (29).

8.2 More on two-state Isomerization

Here we provide details for the two-state isomerization example in this section. The control incident electric field is initialized with a Gaussian form.

$$E(t) = E_0 \cos(\omega_0(t - t_p)) \exp(-(t - t_p)^2/\tau^2), \tag{7}$$

where E_0 is the amplitude of the field, ω_0 is the center frequency, t_p is the pulse arrival time, and τ is the pulse duration. The pulse duration is set to $\tau = 10$ fs, the center frequency to $\omega_0 = 2.4$ eV, and the arrival time to $t_p = 3\tau = 30$ fs. The short pulse duration is chosen to approximate a delta-function, the arrival time to ensure that the pulse is completely contained within the simulation, and the center frequency to approximately match the electronic excitation energy. The field amplitude is chosen as $E_0 = 1$ in atomic units, since, as explained below, the quantum yield is normalized with respect to excited state population, and hence to the field intensity.

The quantity to be optimized is the quantum yield, i.e. the efficiency of isomerization. The *cis* projection operator is denoted as $\hat{P}_c$ and the *trans* projection operator as $\hat{P}_T$. In position space, the operators are given by $\hat{P}_c = \theta(\pi/2 - |\phi|)$ and $\hat{P}_T = \theta(|\phi| - \pi/2)$, where θ is the Heaviside step function and ϕ the molecular rotation coordinate. The quantum yield is then

$$Y(t) = \frac{\langle \hat{P}_T^{11}(t) \rangle}{\langle \hat{P}_T^{11}(t) + \hat{P}_C^{00}(t) \rangle - p_g(t)}, \tag{8}$$

where $\langle ... \rangle = \langle \psi | ... | \psi \rangle$ denotes a quantum expectation value, $\hat{P}_T^{11} = \hat{P}_T \left| \psi_{e_1} \right\rangle \left\langle \psi_{e_1} \right|$ is the projection of $\hat{P}_T$ onto the diabatic excited electronic state, $\hat{P}_C^{00} = \hat{P}_C \left| \psi_{e_0} \right\rangle \left\langle \psi_{e_0} \right|$ projects onto the ground

diabatic state, and p_g is the ground state population. Subtraction of p_g ensures that any population remaining in the ground state does not contribute to the quantum yield. Since the quantum yield depends on time, we optimize its average over a time period after the pulse is over:

$$\langle Y(t) \rangle_t = \frac{1}{T} \int_{t_0}^{t_0+T} dt\, Y(t), \tag{9}$$

where $\langle ... \rangle_t$ denotes a time average. Here, t_0 is the time at which the yield is first recorded, and T is the averaging time. We set $t_0 = 0.5$ ps and $T = 1.5$ ps.

The dynamics are discretized with a timestep of $\Delta t = 0.05$ fs. The electric field is discretized with a time step of $5\Delta t = 0.25$ fs. This is done to limit the timescale over which the electric field can vary.

To avoid the use of complex numbers in backpropagation, the real and imaginary parts of the wave function are stored as separate vectors $|\psi_R(t)\rangle$ and $|\psi_I(t)\rangle$, respectively. They then follow the coupled equations of motion

$$\begin{aligned} \frac{\partial}{\partial t} |\psi_R(t)\rangle &= \hat{H} |\psi_I(t)\rangle , \\ \frac{\partial}{\partial t} |\psi_I(t)\rangle &= -\hat{H} |\psi_R(t)\rangle . \end{aligned} \tag{10}$$

The expectation value of an operator $\hat{A}$ is given by

$$\begin{aligned} &\langle \psi(t)|\hat{A}|\psi(t)\rangle = \langle \psi_R(t)|\hat{A}|\psi_R(t)\rangle + \langle \psi_I(t)|\hat{A}|\psi_I(t)\rangle \\ &+ i \left(\langle \psi_R(t)|\hat{A}|\psi_I(t)\rangle - \langle \psi_I(t)|\hat{A}|\psi_R(t)\rangle \right. . \end{aligned} \tag{11}$$

8.3 Graph Neural Networks

The model is based on graph convolution, which has achieved state-of-the-art predictive performance for chemical properties and molecular energies/forces (30**?** ; 31; 32; 33). In our work, we utilized the SchNet (**?**) architecture to learn the control Hamiltonian. The model consists of a message step and update step to systematically gather information from neighboring atoms. A 3D molecular graph is used, and the existence of a connection between atoms is decided by a fixed distance cutoff. Defining v as the index for each atom and its neighbors as $N(v)$, the graph convolutions process iteratively updates the atomic embedding h_v by aggregating "messages" from their connected atoms v and their edge features e_{uv}. This update process is summarized by

$$h_v^t = h_v^{t-1} + \sum_{u \in N(v)} Message^t(h_u, e_{uv}). \tag{12}$$

By performing this operation several times, a many-body correlation function can be constructed to represent the potential energy surface of a molecular system. In the case of SchNet, the update function is simply a summation over the atomic embeddings. The message function is parameterized by the following equations:

$$Message^t(e_{uv}, h_v) = MLP_3(MLP_1(e_{uv}) \circ MLP_1(h_v) \tag{13}$$

where the MLP_i are independent multi-layer perceptrons (MLP). For each convolution t, a separate message function is applied to characterize atom correlations at different scales. After taking element-wise products of atomic fingerprints h_v and pair-interaction fingerprints e_{uv}, the joint fingerprint is further parameterized by another MLP to incorporate more non-linearity into the model. The final updated fingerprints are used as inputs to two fully-connected layers that yield atom-wise energies. The sum of these energies gives the total energy of the system. The atomic forces are the negative gradients of the energy with respect to atomic positions. They are easily computed through automatic differentiation implemented in PyTorch (34) and Tensorflow (35). We used PyTorch in our demonstrations.

8.4 Differential Histograms

In this section, we present the calculation of an approximate histogram with Gaussian smearing functions. Given an observation r that we wish to approximately map onto a histogram with k bins, the Gaussian-smeared function $\rho_k(r)$ is given by

$$\rho_k(r) = e^{-\frac{(r-r_k)^2}{\delta}} / \sum_k e^{-\frac{(r-r_k)^2}{\delta}}, \tag{14}$$

where δ approximates the bin width. A Gaussian basis is used to replace the non-differentiable Dirac Delta functions to compute an approximate histogram. The total normalized histogram is the expected value over individual samples of r over the all the observations of pair distances (between atoms i and j) in the trajectory:

$$p(r) =_{i,j} \rho(r_{ij}). \tag{15}$$

The pair correlation function $g(r)$ is obtained by normalizing by differential volume increases:

$$g(r) = \frac{V}{N^2 4\pi r^2} p(r). \tag{16}$$