Deep Learning Based Integrators for Solving Newton's Equations with Large Timesteps

JCS Kadupitiya,[1,*] Geoffrey C. Fox,[1,†] and Vikram Jadhao[1,‡]

[1]*Intelligent Systems Engineering, Indiana University, Bloomington IN 47408, USA*

Classical molecular dynamics simulations are based on solving Newton's equations of motion. Using a small timestep, numerical integrators such as Verlet generate trajectories of particles as solutions to Newton's equations. We introduce integrators based on recurrent neural networks that accurately solve Newtons equations utilizing sequences of past trajectory data and produce energy-conserving dynamics of particles using timesteps up to 4000 times larger compared to the Verlet timestep. We demonstrate significant speedup in many example problems including 3D systems of up to 16 particles.

Newton's equations of motion [1] are the basis of powerful computational methods such as classical molecular dynamics (MD) that are used to understand the microscopic origins of a wide range of material and biological phenomena [2, 3]. In the MD method, Newton's equations are integrated for a system of many particles using numerical integrators such as Verlet [4] to produce particle trajectories. The time evolution is performed one small timestep at a time for long times to accurately sample enough representative configurations in order to extract useful information. Consider the 2nd order ordinary Verlet integrator $\vec{x}(t+\Delta) = 2\vec{x}(t) - \vec{x}(t-\Delta) + \Delta^2 \vec{f}/m$ that updates the current position $\vec{x}(t)$ of a particle of mass m at time t to position $\vec{x}(t+\Delta)$ after timestep Δ using the previous position $\vec{x}(t-\Delta)$ and the force $\vec{f}$ at time t. This integrator produces an error of $O(\Delta^4)$ in each local update and incurs the global error of $O(\Delta^2)$ [3, 5]. These discretization errors in solving Newton's equations are reduced by choosing a small Δ which often makes the simulations computationally expensive.

The ordinary Verlet integrator requires a sequence of 2 positions $(\vec{x}_{t-\Delta}, \vec{x}_t)$ to update the particle position using other quantities such as $\vec{f}$ and m. These other quantities can be inferred using the information encoded in a longer stream of position data such that the time evolution can be done with only the history of positions as input. We illustrate this with a 1-dimensional example of a particle experiencing simple harmonic motion governed by the force $f = -kx$. One can show that the particle position can be evolved to $t+\Delta$ using a sequence of 3 positions via the function $\mathcal{V} = x_{t-\Delta}^{-1}\left(x_t^2 - x_{t-\Delta}^2 + x_t x_{t-2\Delta}\right)$, which also incurs a global error of $O(\Delta^2)$. This idea generalizes for higher-order integrators [6] and many-particle systems such that the time evolution can be performed via the operator $\mathcal{V}\left(\vec{x}_t, \vec{x}_{t-\Delta}, \ldots \vec{x}_{t-s\Delta}\right)$ that takes a sequence of s positions. The longer history of input positions enables integrators to perform accurate time evolution with larger Δ. However, this advantage generally comes at the expense of higher computing costs per timestep that often offset the net speedup.

The use of deep learning (DL) in sequence processing and time series prediction problems has been well studied by the industry for different applications including voice recognition and translation [7], pattern recognition in stock market data [8], and ride-hailing [9]. Recurrent neural networks (RNNs) are established DL tools in these applications. We demonstrate that RNNs can be used to design integrators that perform accurate time evolution utilizing sequences of past configurations. The RNN-based integrators are trained using the ground truth results obtained with the 2nd order Verlet integrator. They possess a complex mathematical structure described with up to $100,000$ parameters. We demonstrate that the network complexity enables the integrators to perform time evolution of systems of many particles for a wide range of force fields using a timestep of up to $4000\times$ the baseline Verlet timestep. The relatively small time for inferring the positions as predictions of the DL model keeps overhead costs low and we demonstrate significant net speedups using larger timesteps.

Machine learning has been used to enhance the performance of MD simulations in many recent studies [10–24]. Of particular relevance to our work are approaches that use DL to learn differential equations and replicate the outputs of numerical integrators [25–35]. Recently, such efforts have focused on obtaining solutions of differential equations with discretization steps larger than the baseline [34, 35]. However, most approaches have been evaluated on relatively simple 1D systems. Integrators producing accurate time evolution of 3D systems with larger timesteps have remained elusive.

Recurrent neural networks process input sequence data and maintain a vector $\vec{h}_t$ known as the "hidden state" for each recurrent cell to model the temporal behavior of sequences through directed cyclic connections between cells. $\vec{h}_t$ is updated by applying a function f to the previous hidden state $(\vec{h}_{t-1})$ and the current input $(\vec{x}_t)$. The cells are arranged in a fashion where they fire when the right sequence is fed. A common choice for f is the Long Short Term Memory (LSTM) networks [36]. An LSTM unit contains a cell which is the memory of the unit, and three gates (input gate, output gate, and forget gate) which regulate the flow of information into and out of the cells, enabling LSTM to remember longer depen-

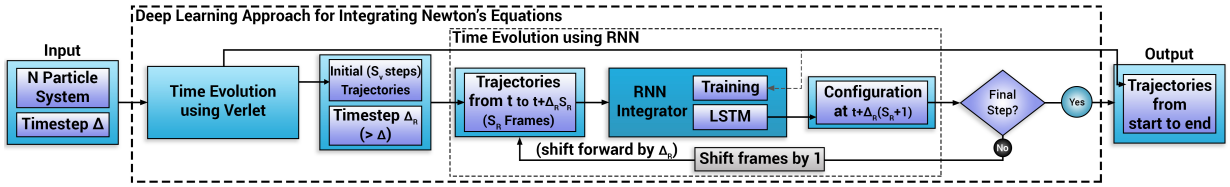

FIG. 1. Overview of the RNN-based approach to integrate Newton's equations and perform time evolution in MD simulations.

dencies of the sequences fed into the network. We now show how RNNs with LSTMs can describe integrators evolving positions of particles. Similar process describes integrators that evolve both positions and velocities.

Each component of the position vector of a particle is identified as a feature. The feature size of both inputs and outputs for N particles in $\mathcal{D}$ physical dimensions is $d = N \times \mathcal{D}$. For example, for $N = 16$ particles interacting in 3D, the feature size is $d = 48$. An operator $\mathscr{R}$ based on RNN is introduced to predict the future position at time $t + \Delta_{\mathrm{R}}$ by employing a position sequence $\{x\} = \vec{x}_t, \vec{x}_{t-\Delta_{\mathrm{R}}}, \ldots, \vec{x}_{t-S_{\mathrm{R}}\Delta_{\mathrm{R}}}$ of length S_{R} up to time t. $\mathscr{R}$ can be expressed as $\mathscr{R}[\{x\}] = \mathscr{D}[\mathscr{L}_2[\mathscr{L}_1[\{x\}]]]$, where $\mathscr{D}$, $\mathscr{L}_1$, $\mathscr{L}_2$ are operators associated with the dense layer, first LSTM layer, and second LSTM layer of the RNN respectively. The layers are stacked up on each other such that the output of one (e.g., $\mathscr{L}_1$) becomes the input for another ($\mathscr{L}_2$) [37]. Each LSTM layer consists of n number of LSTM units and contains a set of parameters in the form of weights, biases, and activation functions. For example, $\mathscr{L}_1$ has n_1 LSTM units and is characterized with weights W and U, and bias b. It takes input feature vector $\{x\}$ and outputs hidden state vectors $\{h\}$ which are fed as input to the $\mathscr{L}_2$ layer characterized with its own set of weights and biases. A similar connection is made between $\mathscr{L}_2$ and the dense layer $\mathscr{D}$. Post training, these layers acquire well-determined values for all the parameters and the integrator $\mathscr{R}$ emerges as:

$$\vec{x}_{t+\Delta_{\mathrm{R}}} = \mathscr{D}\left[\mathscr{L}_2\left[\mathscr{L}_1\left[\{x\}, \{P_1\}\right], \{P_2\}\right], \{P_D\}\right], \quad (1)$$

where $\{P_1\}$, $\{P_2\}$, $\{P_D\}$ are trained parameters associated with LSTM layer 1, LSTM layer 2, and the dense layer respectively. $\mathscr{R}$ has a complex mathematical structure characterized with up to $100,000$ parameters which accounts for its ability to handle larger timesteps.

Figure 1 describes our DL approach using $\mathscr{R}$ to evolve the dynamics of an N particle system with timestep Δ_{R}. $\mathscr{R}$ is trained using the ground-truth particle trajectories generated via the Verlet integrator $\mathscr{V}$ with small timestep $\Delta = 0.001$. First, input system attributes and Δ are fed to $\mathscr{V}$ to simulate the dynamics with timestep Δ up to $S_{\mathrm{V}} = \Delta_{\mathrm{R}}(S_{\mathrm{R}} - 1)/\Delta$ computational steps, where S_{R} is the sequence length. Out of the full trajectory data up to S_{V} steps, S_{R} number of configurations (frames) separated by Δ_{R} are distilled to feed the $\mathscr{R}$ operator. $\mathscr{R}$ predicts the time evolution of the system after timestep Δ_{R}. Then, the input sequence to $\mathscr{R}$ is left shifted to discard the oldest time frame, and the latest frame predicted by $\mathscr{R}$ is appended to the right of the sequence. The adjusted input sequence is fed back to $\mathscr{R}$ to evolve the system Δ_{R} further in time until the end of the simulation.

We now present the results obtained from simulations using $\mathscr{R}$ as an integrator. Single particle experiments in 1D are performed for a variety of potentials including simple harmonic oscillator (SHO), double well (DW), Lennard-Jones (LJ), and rugged [24] [38]. Many-particle experiments in 3D are performed on particles interacting with LJ potential. We adopt units such that the system parameters and predicted quantities are around 1. In these initial studies, training and testing datasets are generated by sweeping parameters that shift the initial configuration (e.g., particle positions), scale the particle attributes (e.g., particle mass) and, in some cases, change the shape of the potential energy (e.g., spring constant k in SHO). $\mathscr{R}$ uses a sequence length of $S_{\mathrm{R}} = 5$.

Our first experiments test $\mathscr{R}$ to predict the dynamics of single particle systems in 1D. Training datasets of trajectories simulated using $\mathscr{V}$ with $\Delta = 0.001$ up to time $t = 200$ are used. For all potentials considered, we find that the errors in positions (trajectory errors) predicted by $\mathscr{R}$ do not increase with time t up to 10000, and are $O(10^{-3})$ for all $\Delta_{\mathrm{R}} \in (100\Delta, 4000\Delta)$ (Figure 2A and 2B). For the same systems, the trajectory errors incurred using $\mathscr{V}$ with timestep 10Δ show the expected exponential increase with t (Supplemental Material). The errors rise with increasing complexity of the potential (e.g., higher for rugged than SHO) and with an increase in timestep Δ_{R} from 100Δ to 4000Δ, but remain within an order of magnitude ($O(10^{-3})$; Figure 2C). In comparison, errors incurred using $\mathscr{V}$ rise much more rapidly with increasing timestep and are already three orders of magnitude larger for timestep of 100Δ (Figure 2C inset). Figure 2 (bottom row) shows the long time dynamics of a particle in an LJ potential. The positions and velocities predicted by $\mathscr{R}$ for timestep $\Delta_{\mathrm{R}} \in [100\Delta, 4000\Delta]$ track the ground truth. Further, the energy deviation (from the initial energy at $t = 0$) produced by $\mathscr{R}$ is small and does not rise with Δ_{R}. On the other hand, trajectories and energy deviation obtained using $\mathscr{V}$ show significant de-

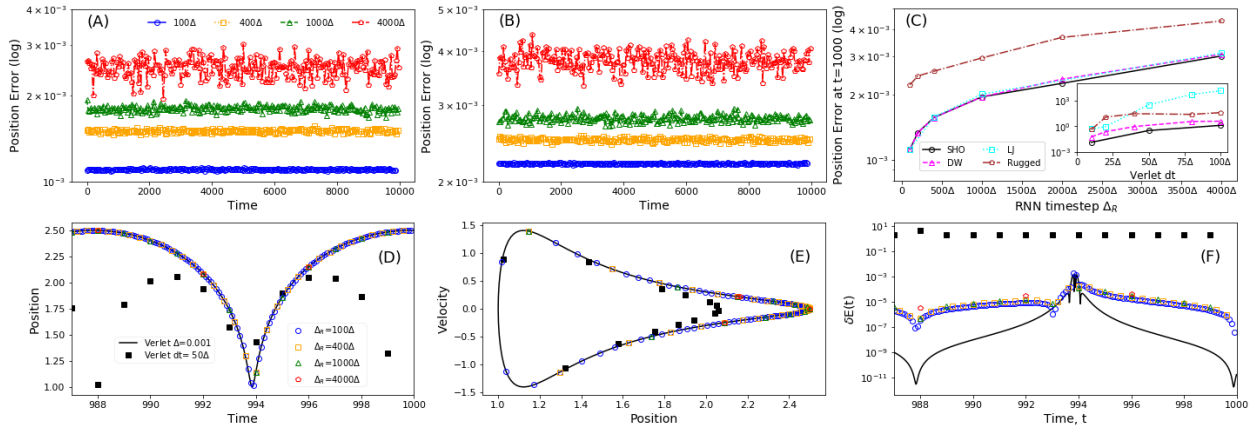

FIG. 2. Trajectory errors and particle dynamics for 1D potentials. Top row: Errors (log scale) in position updates using $\mathscr{R}$ for $\Delta_{\rm R} = 100\Delta$ (circles), 400Δ (squares), 1000Δ (triangles), and 4000Δ (pentagons). (A) Error vs. time for SHO with mass $m = 10$, spring constant $k = 1$, initial position $x_0 = -10$; (B) Error vs. time for rugged potential with $m = 1, x_0 = -6$; and (C) Trajectory error using $\mathscr{R}$ at $t = 1000$ vs. timestep for different potentials; inset shows corresponding error using Verlet integrator $\mathscr{V}$. Note the order of magnitude difference between the outset and inset y-axis values. Bottom row: Dynamics of a particle in a 1D LJ potential ($m = 1, x_0 = 2.5, \epsilon = 1$) from $t = 987$ to 1000. Open and solid symbols are results produced using $\mathscr{R}$ with timestep $\Delta_{\rm R} \in (100\Delta, 4000\Delta)$ and $\mathscr{V}$ with timestep 50Δ respectively. Black lines are the ground truth results obtained using $\mathscr{V}$ with $\Delta = 0.001$. (D) Position vs. time. (E) Velocity vs. position. (F) Deviation in the total energy vs. time.

viation from the ground truth for timestep of 50Δ.

Next, we performed experiments to assess the stability of the trajectories predicted by $\mathscr{R}$. This is typically discussed in terms of Lyapunov instability which states that close-by trajectories diverge exponentially [39, 40]. Lyapunov instability is present in standard Verlet integrators for many potentials. Trajectories of a particle in 1D LJ potential generated using $\mathscr{V}$ timestep $\Delta = 0.001$ and $\mathscr{R}$ with timestep $\Delta_{\rm R} = 100\Delta$ are used for these experiments. A small perturbation in the momentum δp is introduced at the start to investigate its effects on long-time evolution of the trajectory. As Figure 3 shows, $\mathscr{R}$ inherits the characteristic Lyapunov instability of $\mathscr{V}$. As δp increases from 10^{-4} to 10^{-3}, the trajectories predicted by $\mathscr{R}$ and $\mathscr{V}$ exhibit similar average divergences from 0.0153 to 0.149 for $\mathscr{R}$ and 0.0152 to 0.148 for $\mathscr{V}$.

It is instructive to examine the role of sequence length $S_{\rm R}$ in determining the accuracy of $\mathscr{R}$. For smaller $S_{\rm R} =$ 3 or 4, $\mathscr{R}$ is only able to accurately propagate the dynamics for $\Delta_{\rm R} \lesssim 10\Delta$ (Supplemental Material). For $S_{\rm R} = 3$, the trajectory error associated with $\mathscr{R}$ rises steeply for $\Delta_{\rm R} > 10\Delta$, similar to the results for the 2nd order Verlet integrator. For $S_{\rm R} = 4$, the accuracy improves and error scaling is similar to that produced by traditional 4th order integrators [6]. Trained with $S_{\rm R} = 5$, $\mathscr{R}$ shows a much weaker rise in error limited to within an order of magnitude as $\Delta_{\rm R}$ rises up to 4000Δ.

Our next set of experiments probe the extension of the approach to 3D systems of $N = 3, 8, 16$ particles interacting with LJ potential. $\mathscr{R}$ is trained using trajectories generated by $\mathscr{V}$ in a cubic box with periodic boundary conditions (PBC) or in a spherical hard-wall confinement with reflective boundaries. In the former case, $\mathscr{R}$ observed the positions of the particles governed by Newton's equations of motion and the re-mapping dictated by the use of PBC. We find that $\mathscr{R}$ successfully evolved the positions and velocities of particles up to time $t = 10^6$ for all the N-particle systems studied with $\Delta_{\rm R}$ up to 4000Δ. In the interest of brevity, we discuss the results of experiments on the $N = 16$ system in PBC. The training dataset for this study involved time evolution up to $t = 2000$ and the characteristic LJ potential energy $\epsilon = 1$. Results are shown for time evolution using $\mathscr{R}$ for $\epsilon = 2$. The initial positions were selected randomly from outside the training dataset and initial velocities were set to 0. The total energy E_t and the energy deviation $\delta E_t = (E_t - E_0)/E_0$ of the system (Figure 4) show that $\mathscr{R}$ exhibits excellent energy conservation ($\delta E_t \lesssim 10^{-3}$) for up to $t = 10^6$ for $\Delta_{\rm R} \in (100\Delta, 4000\Delta)$. In strike contrast, $\mathscr{V}$ suffers from growing accumulated error when simulated with timestep of 40Δ and exhibits a rapid divergence in the energy for $t > 10^5$.

We introduce metrics to capture the performance enhancement of simulations resulting from the use of DL-based integrators. Our DL approach uses Verlet integrator $\mathscr{V}$ to kickstart the simulation and $\mathscr{R}$ to evolve the dynamics forward in time. Incorporating this detail, we propose the speedup metric $S_p = S_{\rm T} t_{\rm V} / [S_{\rm V} t_{\rm V} + (S_{\rm T} - S_{\rm V})\, t_{\rm R}\Delta/\Delta_{\rm R}]$, where $S_{\rm T}$ is the total number of steps needed if the time evolution is performed only using $\mathscr{V}$, $S_{\rm V} = \Delta_{\rm R}(S_{\rm R} - 1)/\Delta$ is the total number of steps that generate the initial tra-

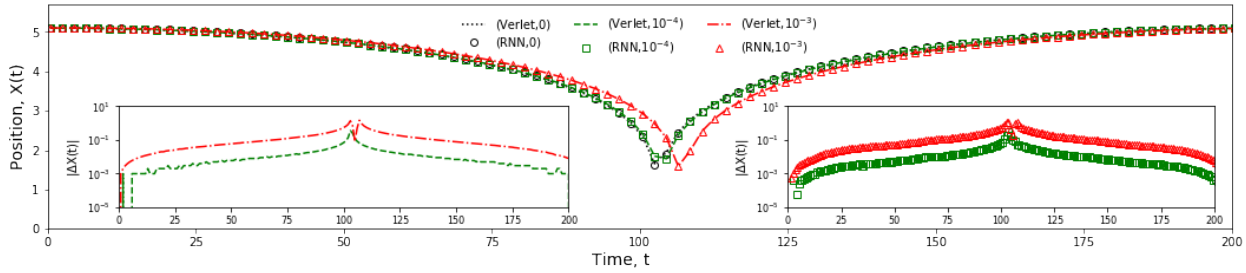

FIG. 3. Lyapunov instability tests in 1D simulations of a particle in an LJ potential with $m = 1, x_0 = 5.1$. Legend indicates (integrator, δp) where δp represents the small momentum shift introduced at the start of trajectory. Lines and markers are the results of simulations driven by $\mathscr{V}$ and $\mathscr{R}$ integrators respectively. Outset shows trajectories for $\delta p = 0, 10^{-4}, 10^{-3}$. Left and right insets show the difference $\Delta x(t)$=$|x(t,0) - x(t,\delta p)|$ in trajectories produced by $\mathscr{V}$ and $\mathscr{R}$ respectively for $\delta p = 10^{-4}, 10^{-3}$.

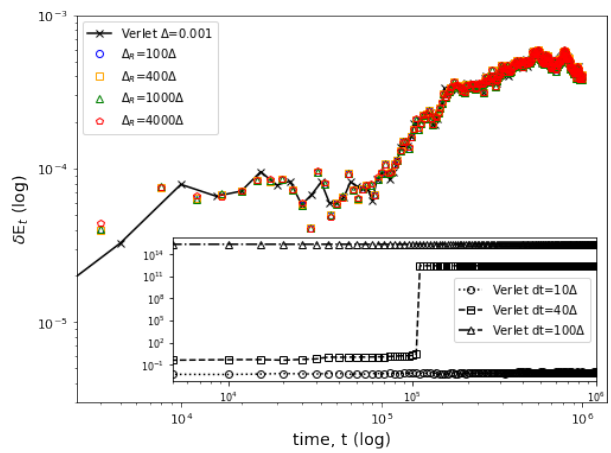

FIG. 4. Energy conservation using $\mathscr{R}$ in a 16 particle 3D simulation with LJ interactions in PBC. Black crosses are the ground truth results obtained using $\mathscr{V}$ with $\Delta = 0.001$. Outset shows the energy deviation δE vs. time t predicted by $\mathscr{R}$ for $\Delta_{\rm R} = 100\Delta, 400\Delta, 1000\Delta, 4000\Delta$. Inset shows the corresponding results using $\mathscr{V}$ with timestep 10Δ, 40Δ, and 100Δ.

jectory using $\mathscr{V}$, and $t_{\rm V}$ and $t_{\rm R}$ are the times for one forward step propagation using $\mathscr{V}$ and $\mathscr{R}$ respectively. We have not accounted for the time spent on creating training datasets (one time investment of < 24 hours for the experiments shown). S_p is 1 if $S_{\rm T} = S_{\rm V}$ (no time evolution using $\mathscr{R}$). In the limit $S_{\rm T} \gg S_{\rm V}$, we obtain $S_p \approx t_{\rm V}\Delta_{\rm R}/(t_{\rm R}\Delta)$. Clearly, the greater the ratio $\Delta_{\rm R}/\Delta$, the higher the speedup. For example, for the system of 16 LJ particles with $t_{\rm V} \approx 0.04392$ seconds, $t_{\rm R} \approx 0.0026$ seconds, $\Delta = 0.001$ and $\Delta_{\rm R} = 4000\Delta$, we find $S_p > 10^4$. We note that S_p can be less than 1 when $\Delta_{\rm R}$ is small and $t_{\rm R} \gg t_{\rm V}$, which can happen in performing time evolution of simpler 1D systems. As the complexity of the system rises, we find generally $t_{\rm R} < t_{\rm V}$. For dynamics up to $t = 10^6$ requiring $S_T = 10^9$ steps, we find S_p up to $45\times$ for the 1D systems, and up to $32000\times$ for the many particle 3D systems [41]. While the main source of speedup is the use of large timestep, $\mathscr{R}$ also exhibits a relatively small forward step propagation (inference) time $t_{\rm R}$ that is largely independent of $\Delta_{\rm R}$ and exhibits $O(N)$ scaling.

While state-of-the-art results in terms of timesteps, number of particles, and the potential complexity are shown for a diverse set of example benchmark problems, research is needed to make further progress towards the long-term goal of machine-learning-assisted MD simulations of many particle systems. Scaling the approach to large N or needing the RNN to "see" a variety of distinct configurations in systems where kinetic barriers can trap particles in metastable pathways may require networks with greater architectural complexity and much longer training times. While these limitations represent a challenge, there are emerging physics-informed neural network architectures [30, 42–45] that can be integrated with the RNN-based approach. The use of hierarchical RNNs [46, 47] and transformers [48] can also be explored. Another key goal will be to test the accuracy of the RNN-based integrators in different ensembles.

We have introduced integrators derived using recurrent neural networks that learn both the interaction potentials and the dynamics of the particles based on their experience with the ground-truth solutions of Newton's equations of motion. We showed that these integrators exhibit excellent energy conservation, inherit the Lyapunov instabilities of the Verlet integrator, and produce accurate predictions for the time evolution of particles over a variety of force fields using up to $4000\times$ larger timestep than the Verlet integrators. The idea of formulating the dynamics of particles into a sequence processing problem solved via the use of recurrent neural networks illustrates an important approach to learn the time evolution operators, which is applicable across different fields [31, 33, 49] including fluid dynamics and robotics.

This work is partially supported by the NSF through awards 1720625 and DMR-1753182. G.C.F was par-

tially supported by NSF CIF21 DIBBS 1443054 and CINES 1835598 awards. V.J. thanks M. O. Robbins for useful discussions.

* kadu@iu.edu
† gcf@iu.edu
‡ vjadhao@iu.edu

[1] Issac Newton, Philosophiae Naturalis Principia Mathematica (G. Brookman, 1687).
[2] Berni J Alder and Thomas Everett Wainwright, "Studies in molecular dynamics. i. general method," The Journal of Chemical Physics **31**, 459–466 (1959).
[3] Daan Frenkel and Berend Smit, Understanding Molecular Simulation, 2nd ed. (Academic Press, 2001).
[4] Loup Verlet, "Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules," Physical review **159**, 98 (1967).
[5] Hans C Andersen, "Rattle: A velocity version of the shake algorithm for molecular dynamics calculations," Journal of Computational Physics **52**, 24–34 (1983).
[6] John Charles Butcher, Numerical methods for ordinary differential equations (John Wiley & Sons, 2016).
[7] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv preprint arXiv:1609.08144 (2016).
[8] Eunsuk Chong, Chulwoo Han, and Frank C Park, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," Expert Systems with Applications **83**, 187–205 (2017).
[9] X Huang, G C Fox, S Serebryakov, A Mohan, P Morkisz, and D Dutta, "Benchmarking deep learning for time series: Challenges and directions," in 2019 IEEE International Conference on Big Data (Big Data) (ieeexplore.ieee.org, 2019) pp. 5679–5682.
[10] Andrew L Ferguson, "Machine learning and data science in soft materials engineering," Journal of Physics: Condensed Matter **30**, 043002 (2017).
[11] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh, "Machine learning for molecular and materials science," Nature **559**, 547 (2018).
[12] Geoffrey Fox, James A. Glazier, JCS Kadupitiya, Vikram Jadhao, Minje Kim, Judy Qiu, James P. Sluka, Endre Somogyi, Madhav Marathe, Abhijin Adiga, Jiangzhuo Chen, Oliver Beckstein, and Shantenu Jha, "Learning everywhere: Pervasive machine learning for effective High-Performance computation," in HPDC Workshop at IPDPS 2019 (2019).
[13] Tristan A. Sharp, Spencer L. Thomas, Ekin D. Cubuk, Samuel S. Schoenholz, David J. Srolovitz, and Andrea J. Liu, "Machine learning determination of atomic dynamics at grain boundaries," Proceedings of the National Academy of Sciences **115**, 10943–10947 (2018), https://www.pnas.org/content/115/43/10943.full.pdf.
[14] Matthew Spellings and Sharon C Glotzer, "Machine learning for crystal identification and discovery," AIChE Journal **64**, 2198–2206 (2018).
[15] Ashley Z Guo, Emre Sevgen, Hythem Sidky, Jonathan K Whitmer, Jeffrey A Hubbell, and Juan J de Pablo, "Adaptive enhanced sampling by force-biasing using neural networks," The Journal of chemical physics **148**, 134108 (2018).
[16] Venkatesh Botu and Rampi Ramprasad, "Adaptive machine learning framework to accelerate ab initio molecular dynamics," International Journal of Quantum Chemistry **115**, 1074–1083 (2015).
[17] JCS Kadupitiya, Geoffrey C Fox, and Vikram Jadhao, "Machine learning for parameter auto-tuning in molecular dynamics simulations: Efficient dynamics of ions near polarizable nanoparticles," The International Journal of High Performance Computing Applications (2020), 10.1177/1094342019899457.
[18] Andrew W Long, Jie Zhang, Steve Granick, and Andrew L Ferguson, "Machine learning assembly landscapes from particle tracking data," Soft Matter **11**, 8141–8153 (2015).
[19] Alireza Moradzadeh and Narayana R Aluru, "Molecular dynamics properties without the full trajectory: A denoising autoencoder network for properties of simple liquids," The journal of physical chemistry letters **10**, 7568–7576 (2019).
[20] Yangzesheng Sun, Robert F DeJaco, and J Ilja Siepmann, "Deep neural network learning of complex binary sorption equilibria from molecular simulation data," Chemical science **10**, 4377–4388 (2019).
[21] Florian Hse, Ignacio Fdez. Galvn, Aln Aspuru-Guzik, Roland Lindh, and Morgane Vacher, "How machine learning can assist the interpretation of ab initio molecular dynamics simulations and conceptual understanding of chemistry," Chem. Sci. **10**, 2298–2307 (2019).
[22] JCS Kadupitiya, Geoffrey C Fox, and Vikram Jadhao, "Machine learning for performance enhancement of molecular dynamics simulations," in International Conference on Computational Science (2019) pp. 116–130.
[23] JCS Kadupitiya, Fanbo Sun, Geoffrey Fox, and Vikram Jadhao, "Machine learning surrogates for molecular dynamics simulations of soft materials," Journal of Computational Science , 101107 (2020).
[24] Jiang Wang, Simon Olsson, Christoph Wehmeyer, Adrià Pérez, Nicholas E Charron, Gianni De Fabritiis, Frank Noé, and Cecilia Clementi, "Machine learning of coarse-grained molecular dynamics force fields," ACS central science **5**, 755–767 (2019).
[25] Maziar Raissi and George Em Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," Journal of Computational Physics **357**, 125–141 (2018).
[26] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong, "Pde-net: Learning pdes from data," arXiv preprint arXiv:1710.09668 (2017).
[27] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud, "Neural ordinary differential equations," in

Advances in neural information processing systems (2018) pp. 6571–6583.

[28] Katsuhiro Endo, Katsufumi Tomobe, and Kenji Yasuoka, "Multi-step time series generator for molecular dynamics," in Thirty-Second AAAI Conference on Artificial Intelligence (2018).

[29] Philip G Breen, Christopher N Foley, Tjarda Boekholt, and Simon Portegies Zwart, "Newton vs the machine: solving the chaotic three-body problem using deep neural networks," arXiv preprint arXiv:1910.07291 (2019).

[30] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou, "Symplectic recurrent neural networks," arXiv preprint arXiv:1909.13334 (2019).

[31] Peiyao Shen, Xuebo Zhang, and Yongchun Fang, "Essential properties of numerical integration for time-optimal path-constrained trajectory planning," IEEE Robotics and Automation Letters **2**, 888–895 (2017).

[32] Maziar Raissi, Paris Perdikaris, and George E Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational Physics **378**, 686–707 (2019).

[33] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner, "Learning data-driven discretizations for partial differential equations," Proceedings of the National Academy of Sciences **116**, 15344–15349 (2019).

[34] Xing Shen, Xiaoliang Cheng, and Kewei Liang, "Deep euler method: solving odes by approximating the local truncation error of the euler method," arXiv preprint arXiv:2003.09573 (2020).

[35] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," arXiv preprint arXiv:1801.01236 (2018).

[36] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," Neural computation **9**, 1735–1780 (1997).

[37] Supplemental Material provides implementation and training details.

[38] Supplemental Material provides the details for the potentials and the training and testing datasets.

[39] HA Posch and Wm G Hoover, "Lyapunov instability of dense lennard-jones fluids," Physical Review A **38**, 473 (1988).

[40] GD Venneri and William G Hoover, "Simple exact test for well-known molecular dynamics algorithms," Journal of computational physics (Print) **73**, 468–475 (1987).

[41] Supplemental Material provides tabulated speedups for different experiments.

[42] Sam Greydanus, Misko Dzamba, and Jason Yosinski, "Hamiltonian neural networks," arXiv preprint arXiv:1906.01563 (2019).

[43] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho, "Lagrangian neural networks," arXiv preprint arXiv:2003.04630 (2020).

[44] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller, "Machine learning of accurate energy-conserving molecular force fields," Science advances **3**, e1603015 (2017).

[45] Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia, "Hamiltonian graph networks with ode integrators," arXiv preprint arXiv:1909.12790 (2019).

[46] Yong Du, Wei Wang, and Liang Wang, "Hierarchical recurrent neural network for skeleton based action recognition," (2015) pp. 1110–1118.

[47] Deniz Eroglu, Norbert Marwan, Martina Stebich, and Jürgen Kurths, "Multiplex recurrence networks," Physical Review E **97**, 012312 (2018).

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in Advances in neural information processing systems (2017) pp. 5998–6008.

[49] Julian Kates-Harbeck, Alexey Svyatkovskiy, and William Tang, "Predicting disruptive instabilities in controlled fusion plasmas through deep learning," Nature **568**, 526–531 (2019).