Supplemental Material for "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*

IMPLEMENTATION DETAILS

There are several architectures of LSTM units. A common architecture is composed of a cell (the memory part of the LSTM unit) and three "regulators", usually called gates, that regulate the flow of information inside the LSTM unit. An input gate (i_t) controls how much new information is added from the present input (x_t) and past hidden state (h_{t-1}) to our present cell state (c_t). A forget gate (f_t) decides what is removed or retained and carried forward to the current cell state (c_t) from the previous cell state (c_{t-1}). An output gate (o_t) decides what to output as the current hidden state (h_t) from the current cell state (c_t). The LSTM formulation can be expressed as:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t). \qquad (1)
\end{aligned}
$$

Here, $x_t \in \mathbf{R}^d$ is the input vector to the LSTM unit, $f_t \in \mathbf{R}^h$ is the forget gate's activation vector, $i_t \in \mathbf{R}^h$ is the input gate's activation vector, $o_t \in \mathbf{R}^h$ is the output gate's activation vector, $h_t \in \mathbf{R}^h$ is the hidden state vector also known as the output vector of the LSTM unit, $c_t \in \mathbf{R}^h$ is the cell state vector, and $\circ$ is the Hadamard product operator. $W \in \mathbf{R}^{h\times d}$ and $U \in \mathbf{R}^{h\times h}$ are the weight matrices and $b \in \mathbf{R}^h$ are the bias vector parameters which need to be learned during training. σ_g and σ_h represent sigmoid function and hyperbolic tangent functions respectively. d and h refer to the number of input features and the number of hidden units respectively.

We introduced an integrator $\mathscr{R}$ derived using LSTMs (Figure 1) that employs a sequence of current and past configurations (e.g., positions and velocities) up to time t to predict the future configuration at time $t+\Delta_\mathrm{R}$. $\mathscr{R}$ integrator with LSTM layer 1, LSTM layer 2, and final dense layer (Figure 1) is implemented in TensorFlow for regression of the particle trajectories using n_1, n_2, and n_D number of hidden units respectively. $\mathscr{R}$ takes a $B\times S_\mathrm{R}\times d$ dimensional vector as input where B is a training parameter denoting batch size and d is the feature size. When used as an integrator (testing phase), $B = 1$. All the parameters $\{P\}$ (weights, biases) describing the layers are trained with an error backpropagation algorithm, implemented via a stochastic gradient descent. Adam optimizer is used to optimize the error backpropagation. Outputs of the LSTM layers are wrapped with the $\tanh$ activation function and no activation functions are used in the final dense layer. The L2 error (mean square loss) between target and predicted trajectories is used for error calculation. LSTM implementation, training, and testing are programmed using scikit-learn, Keras, and TensorFlow machine learning libraries [1–3]. Scikit-learn is used for grid search and scaling, Keras is used to save and load models, and TensorFlow is used to create and train the integrator $\mathscr{R}$. Prototype implementation written using Python/C++ is available on GitHub.

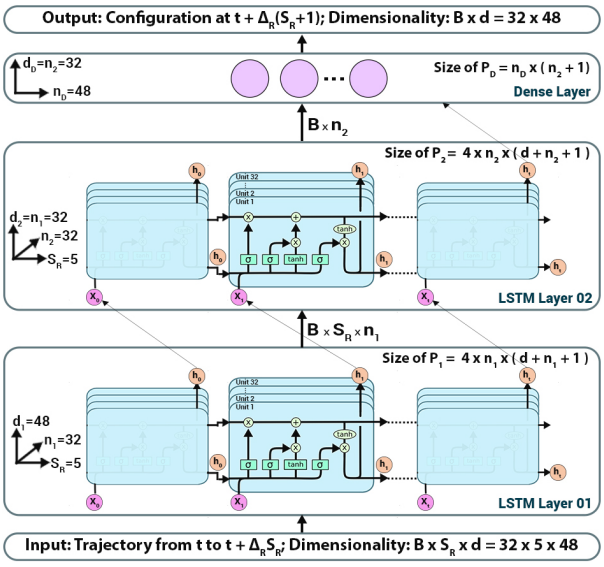

FIG. 1. $\mathscr{R}$ integrator designed to perform a single timestep (Δ_R) evolution of an N particle system characterized by features of size d. Input system and LSTM parameters are shown for 16 particles in 3D. Symbols are defined in the main text.

Values for n_1, n_2, and n_D are chosen depending on the problem complexity and data dimensions. We discuss these choices and other details of the feature extraction and regression process for the most complex case of 16 particles interacting with LJ potential in 3D with PBC. For this system the feature size $d = 96$. By performing a grid search, hyper-parameters such as the number of units for each of the two LSTM layers (n_1, n_2), number of units in the final dense layer (n_D), batch size (B),

and the number of epochs are optimized to 32, 32, 96, 256, and 2500 respectively. Here we have not done a thorough hyper-parameter optimization but have identified reasonable values for key choices – the architecture and the size of the network, and the length of time series. The learning rate of Adam optimizer is set to 0.0005 and the dropout rate is set to 0.15 to prevent overfitting. Both learning and dropout rates are selected using a trial-and-error process. The weights in the hidden layers and in the output layer are initialized for better convergence using a Xavier normal distribution at the beginning [4].

DATASET PREPARATION DETAILS

Prior domain experience and backward elimination using the adjusted R squared method is employed to determine the important input parameters that significantly change the desired output. In some cases, such as particles interacting with the Lennard-Jones (LJ) potential, only a subset of the important input parameters are varied to create the dataset to train the $\mathscr{R}$ integrator for the initial set of experiments presented in this work. It should be noted that unlike traditional deep neural networks where the physical inputs are mapped to outputs generally distinct from inputs, the inputs and outputs for the RNN approach are both trajectory data.

The datasets associated with systems characterized by different potential energies and particle attributes used in the experiments for training and testing $\mathscr{R}$ are described below. The potential energy functions are shown in Figure 2. For the 1D systems, the timestep and the total time associated with generating the ground truth using the Verlet integrator are $\Delta = 0.001$ and $t_f = 100$. For the 3D many particle systems, $\Delta = 0.001$ and $t_f = 2000$. In each case, the entire dataset is randomly shuffled (along the axis of the number of samples) and separated into training and testing sets using a ratio of 0.8:0.2.

Simple harmonic oscillator (SHO) For this system, the potential energy is given by

$$U = \frac{1}{2}kx^2. \tag{2}$$

The dataset is generated by varying three input parameters: mass of the particle $m \in [1, 10]$, spring constant $k \in [1, 10]$, and initial position of the particle $x_0 \in [-10, -1]$. The parameter sweep generated a dataset of 500 simulations, each having 50,000 position and velocity values.

Particle in a double well (DW) For this system, the potential energy is given by

$$U = \frac{1}{4}x^4 - \frac{1}{2}x^2. \tag{3}$$

The dataset is generated by varying two input parameters: mass of the particle $m \in [1, 10]$ and the initial position of the particle $x_0 \in [-3.1, 3.1]$. The parameter sweep generated a dataset of 500 simulations, each having 50,000 position and velocity values.

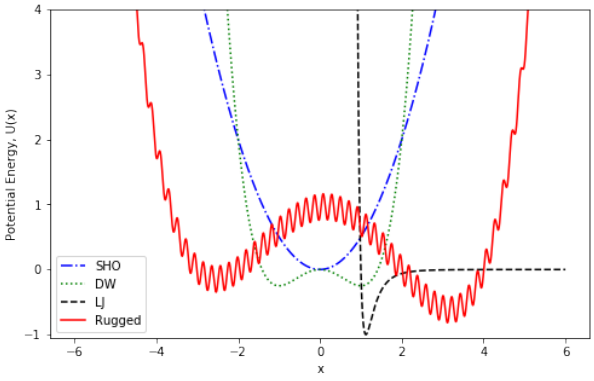

FIG. 2. Potential energies associated with the 1D experiments. Dash-dotted, dotted, dashed, and solid lines represent SHO, DW, LJ, and rugged potential respectively.

1D Lennard-Jones (LJ) system For this system, the potential energy is given by

$$U(x) = 4\epsilon\left(\left(\frac{1}{x}\right)^{12} - \left(\frac{1}{x}\right)^{6}\right). \tag{4}$$

The dataset is generated by varying two input parameters: mass of the particle $m \in [1, 10]$ and the initial position of the particle $x_0 \in [1.0, 3.0]$. The parameter sweep generated a dataset of 500 simulations, each having 50,000 position and velocity values.

Particle in a rugged potential For this system, the potential energy is given by

$$U(x) = \frac{x^4 - x^3 - 16x^2 + 4x + 48}{50} + \frac{\sin\left(30(x+5)\right)}{5}. \tag{5}$$

The dataset is generated by varying two input parameters: mass of the particle $m \in [1, 10]$ and the initial position of the particle $x_0 \in [-6.1, 6.1]$. The parameter sweep generated a dataset of 640 simulations, each having 64,000 position and velocity values.

Many particles interacting with LJ potential For this 3D system, the interaction potential energy between any two particles is given by the LJ potential:

$$U(r) = 4\epsilon\left(\left(\frac{1}{r}\right)^{12} - \left(\frac{1}{r}\right)^{6}\right) + 0.0163\epsilon \quad \text{for } r \leq 2.5. \tag{6}$$

For $r > 2.5$, U is 0. We prepared two different types of simulation boxes to generate the datasets: cubic box with

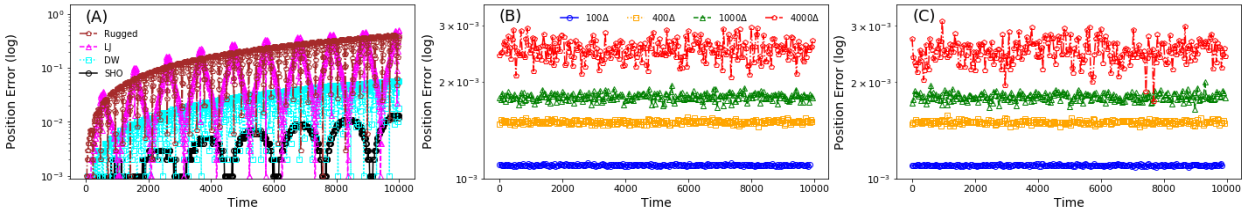

FIG. 3. (A) Errors using $\mathscr{V}$ with timestep $dt = 10\Delta$. (B) and (C) show errors using $\mathscr{R}$ for DW and LJ potentials respectively for $\Delta_{\rm R} = 100\Delta$ (circles), 400Δ (squares), 1000Δ (triangles), and 4000Δ (pentagons).

periodic boundary conditions and spherical box with reflective boundary. In each box, we performed simulations with $N = 3, 8$ and 16 particles. Each of these simulations was created as a separate dataset that yielded 6 different datasets to train and test $\mathscr{R}$. The dataset is generated by varying two input parameters: mass of the particle $m \in [1, 10]$ and the initial position of the particles (x_0, y_0, z_0) with each Cartesian coordinate chosen between -3.0 and 3.0. The well depth was held constant to $\epsilon = 1$ in creating the training dataset. Initial velocities were chosen to be zero. The parameter sweep generated a dataset of 5000 simulations for each of the aforementioned cases (or 30,000 simulations in total).

SUPPORTING EXPERIMENTS AND RESULTS

Trajectory errors in 1D simulations Figure 3 shows the errors (log scale) in position updates for 1D simulations as a function of time. Results are shown for errors obtained using Verlet integrator $\mathscr{V}$ with timestep dt= 10Δ for 4 different potentials: SHO with mass $m = 10$, spring constant $k = 1$, initial position $x_0 = -10$; DW with $m = 1$, $x_0 = -2$; LJ with $m = 1$, $x_0 = 2$; and rugged with $m = 1$, $x_0 = -6$. Results are also shown for errors incurred in time evolution using RNN integrator $\mathscr{R}$ for 2 potentials: DW with $m = 1$, $x_0 = -2$ and LJ with $m = 1$, $x_0 = 2$. Trajectories obtained using $\mathscr{V}$ with $\Delta = 0.001$ are taken as the ground truth.

Testing $\mathscr{R}$ with different sequence lengths In this experiment, we tested the dependence of the accuracy of time evolution performed by $\mathscr{R}$ integrator on the sequence length of input configurations used to train the operator. We trained $\mathscr{R}$ with different input sequence length $S_{\rm R}$ experimenting with $S_{\rm R} = 3, 4$ and 5. The training was performed using the same dataset employed in the original 1D LJ system experiment. The resulting integrator was used to evolve the dynamics of the 1D system of a particle in an LJ potential. Figure 4 shows the position error δr incurred in the time evolution up to $t = 1000$ vs. Δ using different integrators. δr is evaluated using the ground truth results obtained with the 2nd order baseline Verlet integrator $\mathscr{V}$ with $\Delta = 0.001$.

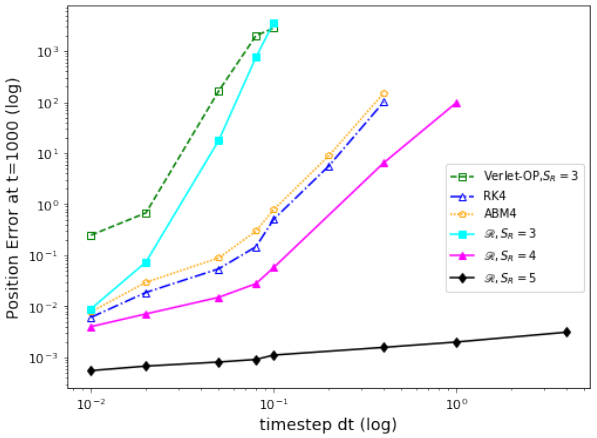

FIG. 4. Error in position at time $t = 1000$ for different timestep $dt \geq 10^{-2}$ in 1D simulations of a particle of mass $m = 1$ in an LJ potential with initial position $x_0 = 2.0$. Open symbols correspond to traditional numerical integrators and closed symbols are results from the $\mathscr{R}$ integrator. Open circles are the errors obtained with the traditional Verlet integrator ($\mathscr{V}$) with $dt \geq 10^{-2}$. Open triangles and pentagons are the errors obtained with 4th order Runge-Kutta (RK4) and 4th order Adams-Bashforth-Moulton (ABM4) respectively. Closed squares, triangles, and diamonds are errors incurred using $\mathscr{R}$ with $S_{\rm R} = 3, 4$ and 5 respectively.

We find that for smaller sequence length $S_{\rm R} = 3$ or 4, $\mathscr{R}$ fails to accurately perform the time evolution for $\Delta_{\rm R} \gtrsim 10\Delta$ incurring errors $\delta r \gtrsim O(\Delta^{-2})$ (SI Figure 4). For $S_{\rm R} = 3$, the error associated with $\mathscr{R}$ rises steeply for $\Delta_{\rm R} > 10\Delta$ and spans over 4 orders of magnitude, 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 such as Runge-Kutta or Adams-Bashforth-Moulton method. $\mathscr{R}$ integrator trained with sequence length $S_{\rm R} = 5$ shows a much weaker rise in error limited to within an order of magnitude as $\Delta_{\rm R}$ rises up to 4000Δ.

Speedup Table I shows S_p for different 1D experiments (first 4 rows) and 3D experiments (last 2 rows) at different $\Delta_{\rm R}$ computed using the metric proposed in the main text for time evolution up to $t = 10^6$ requiring

$S_T = 10^9$ steps. Note that $S_p < 1$ data is for 1D systems with $\Delta_{\mathrm{R}} = 100\Delta$, 200Δ, where $t_{\mathrm{V}} < t_{\mathrm{R}}$.

TABLE I. Net speedup S_p using the $\mathscr{R}$ integrator

Expt. / Δ_{R}	**100**	**200**	**400**	**1000**	**2000**	**4000**
SHO	0.5	1.3	3.2	8.6	20.0	45.0
Double-well	0.6	1.2	2.8	8.7	17.3	38.0
LJ	0.9	1.5	3.9	12.8	22.5	42.3
Rugged	0.4	0.8	2.1	4.7	9.7	20.6
LJ, 8	600	1000	1500	5500	8300	12000
LJ, 16	3000	4900	7100	20000	28000	32000

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

[1] François Chollet et al., "Keras," (2015).
[2] Lars Buitinck et al., "Api design for machine learning software: experiences from the scikit-learn project," arXiv:1309.0238 (2013).
[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: a system for large-scale machine learning." in OSDI, Vol. 16 (2016) pp. 265–283.
[4] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," (2010) pp. 249–256.