Accelerating Long-Term Molecular Dynamics with Physics-Informed Time-Series Forecasting

Hung Le*, Sherif Abbas, Minh Hoang Nguyen, Van Dai Do, Huu Hiep Nguyen, Dung Nguyen
Applied Artificial Intelligence Initiative
Deakin University
Geelong, Australia

***Abstract*—Efficient molecular dynamics (MD) simulation is vital for understanding atomic-scale processes in materials science and biophysics. Traditional density functional theory (DFT) methods are computationally expensive, which limits the feasibility of long-term simulations. We propose a novel approach that formulates MD simulation as a time-series forecasting problem, enabling advanced forecasting models to predict atomic trajectories via displacements rather than absolute positions. We incorporate a physics-informed loss and inference mechanism based on DFT-parametrised pair-wise Morse potential functions that penalize unphysical atomic proximity to enforce physical plausibility. Our method consistently surpasses standard baselines in simulation accuracy across diverse materials. The results highlight the importance of incorporating physics knowledge to enhance the reliability and precision of atomic trajectory forecasting. Remarkably, it enables stable modeling of thousands of MD steps in minutes, offering a scalable alternative to costly DFT simulations.**

***Index Terms*—molecular dynamics, time-series forecasting, physics-informed machine learning**

I. Introduction

Molecular dynamics (MD) simulations are critical for examining atomic-scale interactions in many scientific fields such as materials science and drug discovery. Classical MD, based on numerically integrating Newton's equations of motion at the atomic scale, has long been a fundamental tool for probing molecular behavior [14], [22]. Despite its generality, this method offers only an approximate description of how atoms interact with one another. To be stable, it needs very small time steps, on the order of femtoseconds. This results in high computational costs when simulating longer-timescale phenomena. On the other hand, ab initio molecular dynamics (AIMD) methods such as density functional theory (DFT) provide a more accurate quantum-level description of atomic interactions but are orders of magnitude more expensive, making them impractical for large systems or long simulations [10], [13]. Both approaches, therefore, face fundamental limitations in scalability and efficiency. As a result, the growing demand for faster and more scalable MD modeling has led to increasing interest in machine learning (ML)-based alternatives [1], [23].

State-of-the-art ML methods, including generative models [3], [8], graph neural networks [12], [29], and recurrent architectures [4], [21], [23], have demonstrated potential in deciphering molecular interactions from existing MD trajectory data. *Nonetheless, three limitations remain*. First, most current sequential MD models predict exact atomic positions, which makes it hard for them to work with different molecule structures and typically leads to unstable long-term rollouts. This challenge arises from position-based models relying heavily on the spatial patterns seen during training, making them particularly sensitive to distribution shifts at test time. As prediction errors accumulate over successive steps, these models can quickly diverge from reasonable trajectories. Second, generative approaches suffer from exposure bias and dependence on carefully tuned priors [3], [16], which hinder their robustness and scalability. Third, all existing approaches lack explicit physical constraints, leading to unrealistic atomic configurations that fail to align with real-world molecular behavior. Without these constraints, models may generate unstable high-energy states, where atoms exhibit unphysical overlaps or diverge from expected motion patterns.

To overcome these limitations, we propose a physics-informed time-series forecasting framework, dubbed **PhysTimeMD**, that treats molecular dynamics as a time-series prediction problem. Moreover, by forecasting atomic displacements instead of coordinate positions, our approach captures relative motion and encodes local physical dynamics while remaining invariant to global spatial shifts. Rather than sampling atomic trajectories, our approach formulates molecular dynamics as a deterministic displacement forecasting task. This avoids handcrafted prior tuning and complicated sampling processes in generative rollouts, leverages local temporal dependencies, and enables the incorporation of physical constraints later. By reframing the problem in this way, we improve simulation robustness across temperatures and molecular systems while benefiting from advances in time-series forecasting [24], [26], [27], offering a stable and scalable framework for long-range trajectory prediction. More importantly, we incorporate explicit physical constraints by integrating knowledge from the Morse potential [15] into both the training loss and the inference process. Specifically, the loss function penalizes atomic pairs that violate realistic bonding distances, while the inference mechanism adjusts predicted displacements to discourage unphysical overlaps and excessively high interatomic forces. This dual enforcement of physical plausibility not only reduces the likelihood of unstable or energetically unfavorable configurations but also promotes more accurate

*Corresponding author: thai.le@deakin.edu.au

and chemically meaningful trajectory generation throughout extended simulations. By combining state-of-the-art time-series forecasting with physics-informed regularization, our method outperforms existing ML approaches that model atom positions without physical insights, delivering more accurate, physically consistent, and computationally efficient molecular dynamics simulations.

In summary, our contributions are three-fold:

- We formulate MD simulation as a displacement-based time-series forecasting framework, allowing application of state-of-the-art time-series models while avoiding the complexity and instability associated with generative sampling methods.
- We pioneer incorporating physics-informed constraints by integrating the Morse potential knowledge into both the training loss and inference, promoting realistic atomic interactions.
- We conduct extensive experiments across diverse materials, showing that our method achieves higher accuracy and can generate stable molecular trajectories for thousands of steps, significantly outperforming existing ML-based MD approaches in both precision and efficiency.

II. Method

We introduce the PhysTimeMD framework, which predicts future atomic positions by combining time-series learning and physics-informed regularization. It processes past atomic positions into displacement vectors, forecasts future movements with any time-series models, and reconstructs future positions. A final correction step, based on the Morse potential, ensures the predicted configurations obey physical interaction constraints during both training and inference. An overview of our framework is depicted in Fig. 1

A. Displacement-based Time-series Forecasting Formulation

Let $\mathcal{S} = \{S_t\}_{t=1}^T$ denote a molecular trajectory, where each state S_t consists of N atoms with atomic positions $\mathbf{r}_t = \{\mathbf{r}_t^i\}_{i=1}^N$ in three-dimensional space, i.e, $\mathbf{r}_t^i = \{P_{i,t}^x, P_{i,t}^y, P_{i,t}^z\}$. The goal of forecasting is to predict future states $S_{t+1}, \ldots, S_T$ given historical observations $S_1, \ldots, S_t$. In practice, to enable efficient training, fixed-size input and output windows are typically used. A generic forecaster f_θ models the conditional distribution of future states given past observations:

$$\hat{S}_{t+1:t+L} = f_\theta(S_{t-H:t}), \tag{1}$$

where H is the historical window size, L is the forecasting horizon, and f_θ is a parameterized function that captures temporal patterns within the data. However, we argue that predicting absolute atomic positions is not robust due to the sensitivity to global transformations and the accumulation of errors over time. Instead of directly predicting absolute positions $\mathbf{r}_{t+1}$, we model atomic displacements Δ_t as:

$$\Delta_t^i = \mathbf{r}_{t+1}^i - \mathbf{r}_t^i = \begin{bmatrix} P_{i,t+1}^x - P_{i,t}^x \\ P_{i,t+1}^y - P_{i,t}^y \\ P_{i,t+1}^z - P_{i,t}^z \end{bmatrix} = \begin{bmatrix} \Delta_{i,t}^x \\ \Delta_{i,t}^y \\ \Delta_{i,t}^z \end{bmatrix}, \quad i = 1, .., N. \tag{2}$$

This displacement-based formulation improves generalization by ensuring predictions remain invariant to global coordinate transformations and enhances stability by reducing the accumulation of positional errors.

To construct the training dataset, we process molecular trajectory data as follows. Given T molecular configurations $\{S_t\}_{t=1}^T$, we extract both absolute positions and displacements. Each sample consists of the atomic coordinates $\mathbf{r}_t$ at time step t and the corresponding displacements Δ_t. The dataset is structured as a time series with features at step t as:

$$\begin{aligned} X_t &= [\mathbf{r}_t, \Delta_{t-1}] \\ &= \{P_{i,t}^x, P_{i,t}^y, P_{i,t}^z, \Delta_{i,t-1}^x, \Delta_{i,t-1}^y, \Delta_{i,t-1}^z\}_{i=1}^N \end{aligned} \tag{3}$$

where P and Δ denote position and displacement components, respectively. At $t = 1$, we set $\Delta_0 = \mathbf{0}$. The target becomes:

$$\Delta_{t:t+L-1} = \{\Delta_{i,t:t+L-1}^x, \Delta_{i,t:t+L-1}^y, \Delta_{i,t:t+L-1}^z\}_{i=1}^N, \tag{4}$$

which fully determines S_t since the original states can be reconstructed from the predicted displacements and the original position $\{P_{i,1}^x, P_{i,1}^y, P_{i,1}^z\}_{i=1}^N$.

We translate the forecasting problem (Eq. 1) into predicting the displacement target series $\Delta_{t:t+L-1}$ given the input $X_{t-H:t}$. Specifically, we train a model f_θ to learn the mapping:

$$\hat{\Delta}_{t:t+L-1} = f_\theta(X_{t-H:t}). \tag{5}$$

By learning displacement dynamics, the model captures local interactions while remaining invariant to rigid-body transformations. The predicted states $\hat{S}_{t+1:t+L}$ can be reconstructed from the predicted displacements and the initial positions:

$$\hat{S}_{t+1} = \{\hat{\mathbf{r}}_{t+1}^i\}_{i=1}^N = \left\{ \mathbf{r}_1^i + \sum_{s=1}^{t} \hat{\Delta}_{i,s} \right\}_{i=1}^N. \tag{6}$$

During inference, we adopt an autoregressive strategy over fixed forecasting windows of length L. Specifically, at each autoregression step, the model takes as input the recent window of atomic positions and displacements, predicts the displacements for the next interval, and reconstructs future atomic coordinates via Eq. 6. These predicted positions, combined with their corresponding displacements, are used as features for the next prediction window. This iterative procedure enables long-range trajectory rollout to arbitrary trajectory lengths. By forecasting displacements segment by segment rather than sampling full trajectories in one pass, our approach avoids compounding generation noise and retains the flexibility to incorporate physical constraints or corrections between steps.

B. Physics-Informed Training (PIT)

Morse Potential To ensure physically consistent predictions, we introduce a physics-informed regularization term derived from the Morse potential, which models interatomic interactions as:

$$E(d) = D_e \left(1 - e^{-a(d-d_e)}\right)^2 + b, \tag{7}$$

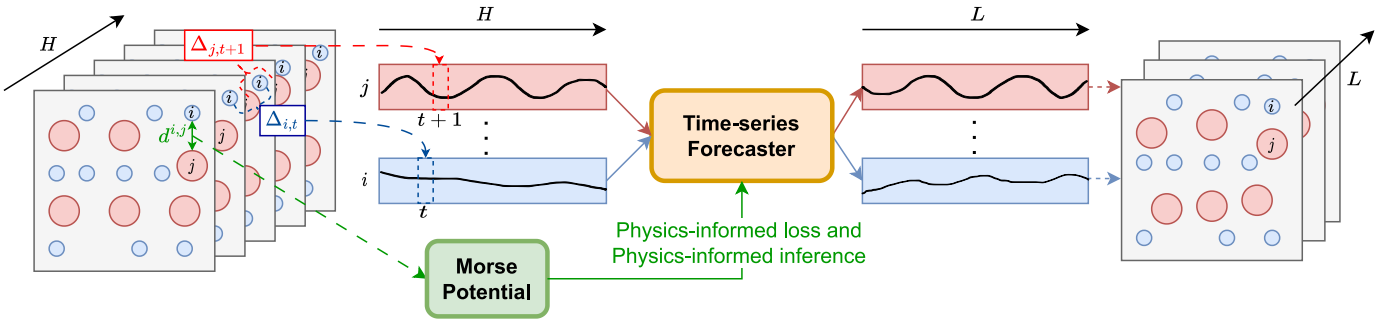

Fig. 1. Overview of the PhysTimeMD framework. The framework takes as input a sequence of atomic positions over the past H timesteps and converts them into relative displacements for each atom (e.g., $\Delta_{i,t}$ and $\Delta_{j,t+1}$ represent the movement vectors of atoms i and j between consecutive steps). Then, it uses a time-series forecaster to predict L-step future displacements. These predictions are then integrated to reconstruct future atomic positions (see Sec. II-A). A physics-informed mechanism, guided by the Morse potential based on the distance between 2 atoms (e.g, $d^{i,j}$), refines the predicted configurations through a physics-based loss during training (see Sec. II-B) and a physics-based correction during inference (see Sec. II-C), ensuring adherence to physical constraints.

where d is the interatomic distance, D_e is the bond dissociation energy, d_e is the equilibrium bond length, a controls the steepness of the potential, and b is a constant energy offset. We fit these parameters (d, D_e, d_e, a, and b) against DFT-computed energies for a number of atom-atom pairs. Specifically, we compute these DFT energies for each atom-atom pair by performing electronic structure optimization using VASP 5.4.4 [9]. The calculations use the generalized gradient approximation (GGA) of Perdew, Burke, and Ernzerhof (PBE) [18]. We apply an energy cutoff for the set of plane wave basis sets of 520 eV, and the energy tolerance is 10^{-6} eV. For each atom-atom pair, we perform the DFT optimization at 20 separation distances, and then fit Eq. 7 against the energies obtained at these separation distances.

Physics-Informed Loss Function For a given molecular system at step t, the model predicts atomic displacements Δ_t, which are used to update the atomic coordinates:

$$\mathbf{r}_{t+1} = \mathbf{r}_t + \Delta_t. \tag{8}$$

We then sample M pairs of atoms $\{(i_m, j_m)\}_{m=1}^{M}$ and compute their pairwise distances using the predicted positions:

$$d_{t+1}^{i_m,j_m} = \left\| \mathbf{r}_{t+1}^{i_m} - \mathbf{r}_{t+1}^{j_m} \right\|_2. \tag{9}$$

To encourage physically plausible predictions at step t, we define a physics-informed loss based on the Morse potential:

$$\mathcal{L}_{\text{phys}} = \frac{1}{|\mathbf{M}'|} \sum_{(i_m,j_m)\in M'} E\big(d_{t+1}^{i_m,j_m}\big), \tag{10}$$

where

$$\mathbf{M}' = \Big\{ (i_m, j_m) \in \{(i_m, j_m)\}_{m=1}^{M} \;\Big|\; E\big(d_{t+1}^{i_m,j_m}\big) > \tau_{\text{train}}^{i,j} \Big\}. \tag{11}$$

Here, $E(d)$ is the Morse potential energy calculated using Eq. 7 and $\tau_{\text{train}}^{i,j}$ is a predefined energy threshold for a pair of atoms (i, j). The core idea is to selectively apply the physics constraint only to atomic pairs whose predicted interactions violate physical plausibility, as determined by the energy threshold. We compute $\tau_{\text{train}}^{i,j}$ from the training dataset. For each atomic pair (i, j), we iterate over all time steps in the training trajectories and compute the Morse potential energy $E(d_t^{ij})$ at each step. The threshold $\tau_{\text{train}}^{i,j}$ is then set as the maximum energy observed for that pair across the entire training set:

$$\tau_{\text{train}}^{i,j} = \max_{t\in\text{Train}} E(d_t^{i,j}). \tag{12}$$

Optimization and Training Details The training objective consists of a weighted sum of the mean squared error (MSE) between predicted and ground-truth displacements, along with a physics-informed regularization term:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \lambda \mathcal{L}_{\text{phys}}, \tag{13}$$

where λ is the physics-informed hyperparameter controlling the trade-off between predictive accuracy and physical consistency. We optimize this objective using the Adam optimizer with a learning rate η and a mini-batch size of B. The training process updates the model parameters θ via gradient descent:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}. \tag{14}$$

To ensure stability during training, we apply gradient clipping and use an adaptive learning rate schedule. The model is trained with early stopping based on validation loss. This loss-based approach encourages the model to learn to predict stable molecular trajectories.

C. Physics-Informed Inference (PII)

Although the model is trained with a physics-informed loss that penalizes high-energy atomic states, this alone does not guarantee physically plausible predictions at inference time. The learned model may still occasionally generate displacements that result in unphysical atom-atom interactions, such as atoms collapsing too closely, especially when extrapolating to long trajectories or out-of-distribution inputs.

To address this, we introduce a physics-informed inference correction step as a safeguard during trajectory rollout. Specifically, after predicting the atomic displacements and updating positions, we check whether any interatomic distances violate

physical constraints by computing their Morse potential energy. If the predicted configuration results in excessive potential energy for any sampled pair, we discard the displacement and freeze the atoms' positions at the current step.

In particular, at each autoregressive step, similar to constructing the physics-informed loss, we sample a set of atomic pairs (i, j) and compute the corresponding interatomic distances $d_{t+1}^{i,j}$ using the updated coordinates:

$$d_{t+1}^{i,j} = ||\mathbf{r}_{t+1}^{i} - \mathbf{r}_{t+1}^{j}||_2. \tag{15}$$

The Morse potential energy for each pair is evaluated as

$$E(d_{t+1}^{i,j}) = D_e \left(1 - e^{-a(d_{t+1}^{i,j} - d_e)}\right)^2 + b. \tag{16}$$

To avoid the generation of unphysical configurations, we again apply the energy threshold $\tau_{train}^{i,j}$ in Eq. 12. If the computed energy $E(d_{t+1}^{i,j})$ for any sampled pair exceeds its corresponding threshold $\tau_{train}^{i,j}$, the predicted displacement Δ_{t+1} is rejected and replaced with a zero vector:

$$\Delta_{t+1} = \begin{cases} \Delta_{t+1}, & \text{if } E(d_{t+1}^{i,j}) < \tau_{train}^{i,j} \text{ for all sampled pairs} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{17}$$

This correction serves as a real-time filter that enforces conservative updates, effectively preventing the propagation of physically invalid configurations. This is particularly important during autoregressive inference at test time, where error accumulation can lead to out-of-distribution inputs, making the training-time loss in Eq. 10 insufficient to guarantee constraint satisfaction. Inference pseudocode is provided in Algo. II-C, with a runtime complexity that scales linearly with the prediction length T.

Algorithm 1 Physics-Informed Inference

Require: Initial atomic positions $\mathbf{r}_1$, $\Delta_0 = \mathbf{0}$, forecasting model f_θ, energy threshold $\tau_{train}^{i,j}$, total steps T, forecasting window size L

Ensure: Forecasted trajectory $\{\mathbf{r}_t\}_{t=1}^{T}$

for $t = 1$ to T **do**
  Construct input $X_t = [\mathbf{r}_t, \Delta_{t-1}]$
  Predict future displacements: $\Delta_{t:t+L-1} = f_\theta(X_t)$
  for $\ell = 0$ to $L - 1$ **do**
    Compute $\mathbf{r}_{t+\ell+1} = \mathbf{r}_{t+\ell} + \Delta_{t+\ell}$
    Sample atomic pairs $\mathcal{P} = \{(i, j)\}$ from $[1, N]$
    `violation` $\leftarrow$ `False`
    for each $(i, j) \in \mathcal{P}$ **do**
      Compute $d_{t+\ell}^{i,j} = \|\mathbf{r}_{t+\ell}^{i} - \mathbf{r}_{t+\ell}^{j}\|_2$
      Compute $E(d_{t+\ell}^{i,j})$ using Morse potential
      if $E(d_{t+\ell}^{i,j}) > \tau_{train}^{i,j}$ **then**
        `violation` $\leftarrow$ `True`
        break
      end if
    end for
    if `violation` **then**
      $\Delta_{t+\ell} \leftarrow \mathbf{0}$
      $\mathbf{r}_{t+\ell+1} = \mathbf{r}_{t+\ell} + \Delta_{t+\ell}$
    end if
  end for
end for

III. Experimental Results

A. Generation of AIMD Atomic Data

To create training and testing data for machine learning models, we carried out AIMD simulations using the Vienna Ab initio Simulation Package (VASP) [9]. We simulated a diverse set of materials containing different atomic species, with system sizes up to 300 atoms. These simulations were run at temperatures ranging from 600 K to 1200 K. At these temperatures, atomic vibrations become significant, making it possible to probe various thermodynamic behaviors. However, the resulting complex and noisy trajectories present a major challenge for learning-based models aiming to accurately predict molecular dynamics over time.

Each simulation used a time step of 1 fs, updating atomic positions and velocities via the Verlet integration scheme [6]. All simulations were performed in the NVT ensemble using the Nosé thermostat to maintain constant temperature. The atomic interactions were modeled using a projector-augmented wave (PAW) pseudopotential, along with the Perdew–Burke–Ernzerhof (PBE) exchange-correlation functional under the generalized gradient approximation (GGA). A plane-wave energy cutoff of 500 eV and single Γ-point sampling was used.

TABLE I
Dataset statistics across different temperatures and materials.

Dataset	Material	Temperature (K)	Train/Valid/Test Size
A	$Li_{10}GeP_2S_{12}$	600	28,000 / 4,000 / 1,000
B	$Li_{10}GeP_2S_{12}$	800	28,000 / 4,000 / 1,000
C	$Li_{10}GeP_2S_{12}$	1,000	28,000 / 4,000 / 1,000
D	$Li_{10}GeP_2S_{12}$	1,200	28,000 / 4,000 / 1,000
E	$Li_7Ta_1O_6$	800	28,000 / 4,000 / 1,000
F	$Li_7Ta_1O_6$	1,000	19,144 / 2,735 / 1,000
G	$LiGaBr_3$	1,000	5,967 / 852 / 1,000

The simulation produces six datasets, each corresponding to a distinct combination of temperature and sequence length. For each dataset, we apply a train/validation/test split and summarize the resulting dataset statistics in Table I.

B. Training and Evaluation Setup

All models were trained using a single NVIDIA A100 GPU, with training times ranging from 1 to 1.6 hours depending on the dataset size. To assess result variability, we initially conducted multiple training runs and measured the mean and standard deviation of the final performance metrics. However, we observed that the variations were negligible. Therefore, in line with common practice in time-series forecasting, we report results from a single representative training run.

To evaluate each model, we begin with an initial window of atomic positions of length H. Using this as input, the model autoregressively predicts the future atomic trajectory over a specified forecast horizon (1000 testing steps). Importantly, during this prediction phase, the model does not receive ground-truth positions at each step. Instead, it feeds its own previously predicted positions back as input for subsequent predictions. This setup reflects a more realistic and challenging scenario, as it simulates deployment conditions where ground-truth data is unavailable beyond the observed history.

Forecasting metrics To assess how close the predicted atomic trajectories are to the ground-truth data, we adopt standard metrics from the time-series forecasting literature, namely the Mean Squared Error (MSE) and Mean Absolute Error (MAE). These metrics can be computed between the predicted and ground-truth values, which can refer to either atomic displacements $\Delta_t^i \in \mathbb{R}^3$ or positions $\mathbf{r}_t^i \in \mathbb{R}^3$. For a forecast horizon of length L over N atoms, the metrics are:

a) Displacement:

$$\text{MSE}_\Delta = \frac{1}{LN}\sum_{t=1}^{L}\sum_{i=1}^{N}\left\|\hat{\Delta}_t^i - \Delta_t^i\right\|_2^2, \tag{18}$$

$$\text{MAE}_\Delta = \frac{1}{LN}\sum_{t=1}^{L}\sum_{i=1}^{N}\left\|\hat{\Delta}_t^i - \Delta_t^i\right\|_2, \tag{19}$$

b) Position:

$$\text{MSE}_\mathbf{r} = \frac{1}{LN}\sum_{t=1}^{L}\sum_{i=1}^{N}\left\|\hat{\mathbf{r}}_t^i - \mathbf{r}_t^i\right\|_2^2, \tag{20}$$

$$\text{MAE}_\mathbf{r} = \frac{1}{LN}\sum_{t=1}^{L}\sum_{i=1}^{N}\left\|\hat{\mathbf{r}}_t^i - \mathbf{r}_t^i\right\|_2. \tag{21}$$

These metrics quantify the average squared and absolute deviations, serving as reliable indicators of forecasting accuracy for both displacement and position predictions.

Physical metrics To evaluate the physical plausibility of predicted trajectories, we introduce physical metrics that quantify how often the model produces energetically infeasible atomic configurations. Specifically, for each forecasted step t, we sample a subset of atomic pairs (i, j) and compute their pairwise distances $d_t^{ij} = \|\mathbf{r}_t^i - \mathbf{r}_t^j\|_2$, which are then used to calculate the corresponding Morse potential energies $E(d_t^{ij})$.

A violation is recorded if the energy between any sampled pair exceeds a threshold $\tau_{i,j}$. We define a binary indicator:

$$\mathbb{I}_t^{ij} = \begin{cases} 1, & \text{if } E(d_t^{ij}) > \tau_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

Here, $\tau_{i,j}$ can be estimated using the test dataset as follows: for each atomic pair (i, j), we iterate over all time steps in the test trajectories and record the Morse potential energy $E(d_t^{ij})$ at each step. The threshold $\tau_{i,j}$ is then set to the maximum energy observed for that pair across the entire test set:

$$\tau_{i,j} = \max_{t \in \text{Test}} E(d_t^{ij}). \tag{23}$$

TABLE II

PERFORMANCE COMPARISON BETWEEN OUR METHOD *PhyTimeMD* AND CONVENTIONAL MD BASELINES. LOWER VALUES OF MSE_r, MAE_r, AND V_r INDICATE BETTER PERFORMANCE. BEST RESULTS ARE **BOLDED**.

Dataset	Model	MSE_r	MAE_r	$V_r\%$
A	LSTM	30.57	4.65	33.833
	TimeMixer	10.08	2.05	3.384
	PhysTimeMD (Ours)	**6.20**	**0.98**	**0.028**
B	LSTM	27.14	4.36	31.074
	TimeMixer	13.67	2.66	2.677
	PhysTimeMD (Ours)	**6.78**	**1.13**	**0.034**
C	LSTM	31.65	4.71	33.754
	TimeMixer	12.52	2.53	5.410
	PhysTimeMD (Ours)	**5.92**	**1.11**	**0.000**
D	LSTM	31.17	4.64	27.797
	TimeMixer	17.78	3.89	4.110
	PhysTimeMD (Ours)	**6.08**	**1.13**	**0.026**
E	LSTM	32.35	4.40	44.090
	TimeMixer	10.35	1.78	9.984
	PhysTimeMD (Ours)	**4.49**	**0.82**	**6.760**
F	LSTM	33.39	4.48	42.164
	TimeMixer	7.63	1.25	6.725
	PhysTimeMD (Ours)	**3.76**	**0.77**	**2.310**
G	LSTM	52.92	6.09	31.237
	TimeMixer	31.38	4.51	7.578
	PhysTimeMD (Ours)	**9.54**	**1.18**	**1.140**

This data-driven approach ensures that the violation criterion reflects the highest physically valid interaction energy encountered under ground-truth dynamics. By using pair-specific thresholds instead of a global constant, we accommodate natural variations in interaction strength and equilibrium distances across different atom types or bonded states.

Let $\mathcal{P}_t$ denote the set of sampled atomic pairs at time t with $|\mathcal{P}_t| = M$, the *total number of violations* V_n over a forecast horizon of length L is computed as:

$$V_n(L, M) = \sum_{t=1}^{L}\sum_{(i,j)\in\mathcal{P}_t}\mathbb{I}_t^{ij}. \tag{24}$$

To allow for a fair comparison across different systems and trajectories, we also report the normalized *violation rate* V_r, which captures the average proportion of violating pairs per time step:

$$V_r = \frac{1}{LM}V_n. \tag{25}$$

These violation-based metrics are orthogonal to traditional error metrics and serve as a proxy for the degree of physical constraint adherence in the predicted dynamics.

C. Comparison with Conventional MD models

To evaluate the effectiveness of our approach, we compare it against conventional MD models that typically predict future atomic positions directly from past positions without any physics constraint. Specifically, we follow prior work on modeling MD trajectories [4], [21], [23] and implement an LSTM network for position prediction. We also include an additional baseline based on TimeMixer [24], a strong deep-learning model for time-series forecasting. This baseline represents a straightforward approach that directly applies a time-series model to predict atomic positions. For a fair comparison,

our PhysTimeMD framework also adopts TimeMixer as the forecast backbone. We evaluate the models using MSE_r and MAE_r. It is worth noting that generative and graph methods [3], [16], [20] cannot apply to our datasets, as they rely on prior structural information that is unavailable in our setting.

The results in Table II show that our PhysTimeMD framework consistently outperforms both LSTM and TimeMixer across all datasets. For example, **in dataset C**, PhysTimeMD achieves an MSE_r of 5.92 and an MAE_r of 1.11, compared to 12.52 and 2.53 for TimeMixer, and 31.65 and 4.71 for LSTM, respectively. The violation rate V_r is reduced to 0.0%, in stark contrast to 5.4% for TimeMixer and 33.7% for LSTM. Similar improvements are observed across the other datasets. **In dataset E**, PhysTimeMD achieves an MSE_r of 4.49 and MAE_r of 0.82, improving upon TimeMixer (MSE_r: 10.35, MAE_r: 1.78) and LSTM (MSE_r: 32.35, MAE_r: 4.40). **In dataset G**, where the baseline errors are particularly high, PhysTimeMD reduces the MSE_r from 52.92 (LSTM) and 31.38 (TimeMixer) to just 9.54, and MAE_r from 6.09 and 4.51 down to 1.18.

Moreover, our violation rate V_r is consistently reduced by at least an order of magnitude. LSTM models yield V_r values between 27.8% and 44.1%, while TimeMixer achieves 2.7% to 9.9%. In contrast, PhysTimeMD keeps V_r below 1.2% across all datasets. For example, V_r drops to 0.028% **in dataset A**, 0.034% in B, and 0.026% in D. These results confirm the effectiveness of our approach, which combines displacement modeling with physics-informed components.

D. Benchmarking with Time-Series Baselines

In this section, we examine our PhysTimeMD as a general framework that can be applied to any modern time-series model to equip them with physics-informed components. To create an equal impact of displacement modeling, we apply the Displacement-based Time-Series Forecasting Formulation (Sec. II-A) uniformly across all forecasting models. Under this setup, we evaluate several state-of-the-art time-series architectures both with and without the integration of PhysTimeMD. These include *TimeMixer* [24], *ITransformer* [11], *Mamba* [7], and *TSMixer* [2]. We implement these baselines using a public code repository introduced in [25].

For illustrative purposes, Fig. 2 reports MAE_Δ and normalized physical violation rate V_r (in %) for datasets G and F, while the comprehensive results are provided in Appendix Tab. IV with all datasets and metrics. **In dataset F**, all architectures benefit from PhysTimeMD: the MAE_Δ of TSMixer decreases from 1.32×10^{-2} to 6.88×10^{-3} with V_r halved (from 6.5% to 2.6%), and ITransformer, despite only a modest improvement in MAE_Δ (from 7.53×10^{-3} to 6.97×10^{-3}), exhibits a substantial decrease in violation rate (from 6.3% to 2.7%). **In dataset G**, PhysTimeMD yields remarkable improvements: the MAE_Δ of TimeMixer decreases from 5.87×10^{-3} to 3.79×10^{-3} with V_r reduced from 15.4% to 11.4%; TSMixer's MAE_Δ is reduced fivefold (from 2.48×10^{-2} to 3.79×10^{-3}) while V_r declines nearly fourfold (from 45.8% to 11.8%). Two notable cases further highlight the benefits of PhysTimeMD. **In dataset B**, although TimeMixer without PhysTimeMD slightly outperforms the version with PhysTimeMD in terms of MAE_Δ and MSE_Δ, its violation rate is 6.1% compared to 0.034% of PhysTimeMD, rendering it impractical to be used in reality. This underscores the capability of PhysTimeMD to enforce physical consistency independently of the forecast error. **In dataset E**, the backbone TSMixer without PhysTimeMD diverges, producing unbounded position predictions that preclude reliable MAE/MSE computation, whereas with PhysTimeMD it remains stable (see Appendix VII-A for an analysis of this). Across all remaining datasets, PhysTimeMD consistently enhances both forecasting accuracy and adherence to physical constraints, demonstrating that integrating physics-informed methodologies into contemporary time series models yields predictions that are both more precise and inherently physically plausible.

TABLE III
ABLATION STUDY OF PHYSTIMEMD COMPONENTS—PHYSICS-INFORMED TRAINING (PIT) AND PHYSICS-INFORMED INFERENCE (PIF)—USING THE TIMEMIXER BACKBONE ON DATASET A. REPORTED METRICS INCLUDE MAE_Δ, MSE_Δ, AND THE PHYSICAL VIOLATION RATE V_r (%).

PIT	PIF	MAE_Δ	MSE_Δ	V_r %
✗	✗	6.05×10^{-3}	6.47×10^{-5}	4.36
✓	✗	5.73×10^{-3}	5.89×10^{-5}	2.83
✗	✓	4.77×10^{-3}	4.10×10^{-5}	0.045
✓	✓	$\mathbf{4.60 \times 10^{-3}}$	$\mathbf{3.82 \times 10^{-5}}$	**0.028**

E. Ablation Study and Hyperparameter Analysis

Effectiveness of Physics-Informed Training and Inference. In this experiment, we investigate the contributions of the individual components of PhysTimeMD, specifically Physics-Informed Training (PIT) and Physics-Informed Inference (PIF). The evaluation is conducted using the TimeMixer backbone on Dataset A. The hyperparameter λ is fixed at 0.0001, as this value yields the best performance (refer to Tab. IV). The ablation results in Tab. III demonstrate that both PIT and PIF significantly enhance the TimeMixer backbone's performance. Enabling PIT alone reduces MAE from 6.05×10^{-3} to 5.73×10^{-3} (a 5.3% improvement), MSE (a 9.0% improvement), and V_r (a 35.1% reduction). Enabling PIF alone yields larger gains: MAE decreases with an improvement of 21.2%, MSE (36.6% improvement), and V_r (a 99.0% reduction). Combining both PIT and PIF (full PhysTimeMD) achieves the best overall results, with MAE 4.60×10^{-3} (a 24.0% improvement), MSE 3.82×10^{-5} (a 40.9% improvement), and V_r 0.028% (a 99.4% reduction). These findings confirm that each component contributes uniquely to accuracy and physical consistency, and their joint application delivers the best results.

Sensitivity to the Hyperparameter λ We perform an ablation study to examine the effect of the physics-based hyperparameter λ on model accuracy, as shown in Fig. 3. We explore three tuned values—0.0001, 0.0005, and 0.001—alongside a value of $\lambda = 0$, which corresponds to removing the physics-informed training component. Both models show improved

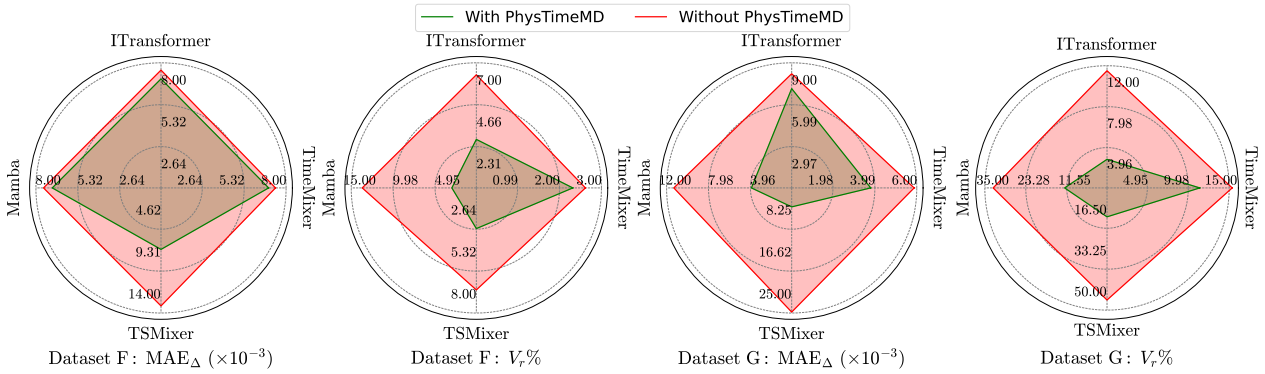

Fig. 2. Radar plot comparing forecasting error (MAE$_\Delta$) and normalized physical violation rate V_r (%) across datasets G (left) and F (right). The lower the metrics the better the results. Each axis represents one of the four baseline architectures—TimeMixer, ITransformer, Mamba, and TSMixer—with and without the PhysTimeMD framework, highlighting improvements in both accuracy and physical consistency.

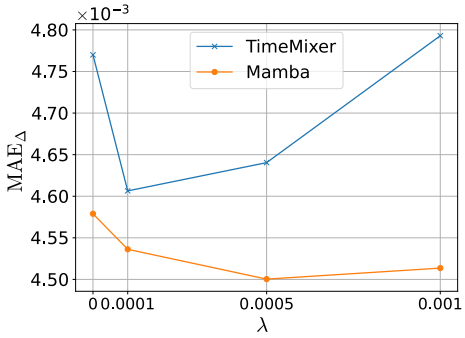

Fig. 3. Effect of λ on MAE$_\Delta$ performance for TimeMixer and Mamba on Dataset A. Each curve illustrates how careful tuning of this hyperparameter can positively influence prediction accuracy.

performance within the tuned range. Specifically, Mamba achieves its lowest MAE of 4.51×10^{-3} at $\lambda = 0.0005$, while TimeMixer performs best at $\lambda = 0.0001$ with a MAE of 4.60×10^{-3}. These findings indicate that the hyperparameter λ plays a critical role in model accuracy; however, excessively large values may degrade performance.

F. Evaluation of Material Properties

To assess the physical consistency of the predicted atomic trajectories, we compute the **diffusion coefficient** D, which captures how atoms diffuse over time. Diffusivity is a key material property related to ionic transport, defect motion, and phase behavior in solids and fluids. We estimate D using the Einstein relation:

$$\left\langle |\mathbf{r}(t) - \mathbf{r}(0)|^2 \right\rangle = 2nDt, \tag{26}$$

where $\mathbf{r}(t)$ is the position of an atom at time t, n is the number of spatial dimensions, and $\langle \cdot \rangle$ denotes an average over atoms and simulation time. While it is important for a model to accurately predict the positions of atoms in a dynamics simulation, it is more important for the model to satisfy key quantitative and qualitative features, such as the diffusivity of ions in the material.

In Fig. 4, we report the diffusivity estimated from our model's generated trajectories and compare it against that from a strong time-series forecaster (TimeMixer) and ground-truth data of dataset A. Closer agreement with the ground-truth indicates stronger physical fidelity in the long-term behavior of the predictions. PhysTimeMD achieves a diffusivity magnitude around 10^{-17}, closely matching the ground-truth range, while the baseline model TimeMixer significantly overestimates diffusivity at approximately 10^{-15}. More importantly, all the baseline curves exhibit a consistent upward drift over time, reflecting crystal melting. In contrast, PhysTimeMD maintains stable diffusion trends, with only two cases showing minor deviation (Ge and S), compared to just one in the ground truth (Ge). This demonstrates that PhysTimeMD not only captures accurate and stable long-term atomic dynamics but also generates trajectories with realistic diffusivity values, making them a viable replacement for ground-truth data in downstream material simulations.

Regarding running time, our method can generate 1,000 simulation steps in approximately 6 minutes on a single NVIDIA A100 GPU. In contrast, DFT-based molecular dynamics simulations are significantly more computationally intensive, requiring approximately 24 hours on a 96-CPU core computational node. This highlights the substantial efficiency advantage of our approach over traditional DFT simulations. To reproduce the results, we will release the source code and data upon publication.

IV. Related Works

Machine learning has emerged as a powerful paradigm for enhancing MD simulations, addressing the long-standing trade-off between simulation accuracy and computational cost. Early machine learning approaches, such as support vector machines and autoencoders, were primarily used to extract thermodynamic properties like free energy from MD trajectories and to approximate full Boltzmann distributions for small biomolecules [17]. However, these methods fall short of capturing high-resolution femtosecond (fs) dynamics—one of the central goals of MD simulations.

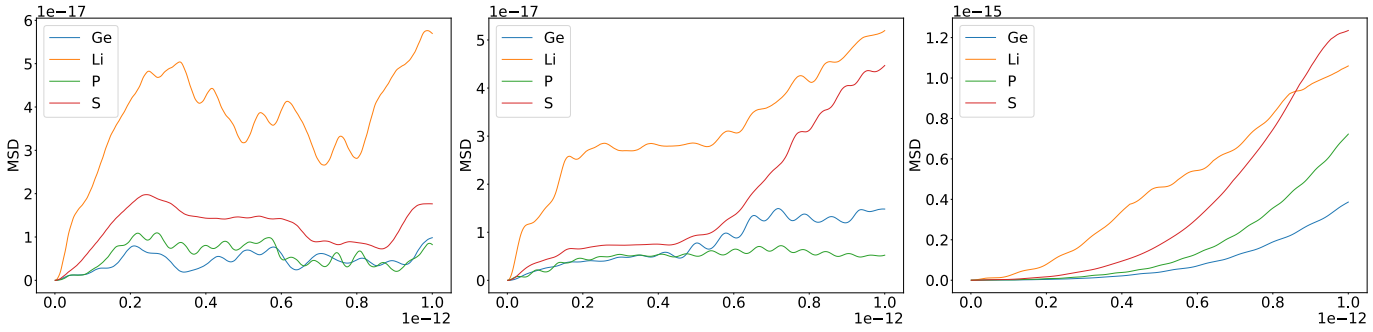

Fig. 4. Diffusivity computed from ground-truth trajectories (left), PhysTimeMD-generated trajectories (middle), and TimeMixer-generated trajectories (right).

Recent advances in machine learning have significantly enhanced the modeling and acceleration of MD simulations. Broadly, three classes of deep learning approaches have emerged as particularly influential: recurrent neural networks, generative models, and graph neural networks.

Recurrent neural networks (RNNs), including LSTMs, have been applied to model MD trajectories as high-frequency, high-dimensional time series data [4], [21], [23]. These models can capture short-term dynamics but often struggle with long-range dependencies and scalability, especially when modeling complex biomolecular systems with more than a few dozen particles. This limitation stems from their reliance on predicting atomic coordinates directly, without integrating physical principles, which leads to error accumulation and poor generalization.

Generative models—such as GANs, diffusion, and flow-based approaches—have been explored to produce long MD trajectories by stitching together short segments [3], [5], [8], [16], [19], [28]. While these methods show promise, they often suffer from exposure bias, where small errors compound over iterations, leading to unrealistic long-term dynamics. Training generative models is often unstable and data-hungry, requiring careful balancing of adversarial losses or diffusion schedules, and they lack built-in physical constraints, making it difficult to ensure energy conservation or adherence to molecular symmetries. This limits their reliability for simulating fine-grained, high-resolution trajectories over extended timescales.

Graph neural networks (GNNs) offer a compelling alternative by modeling molecular systems as graphs, with atoms as nodes and interactions as edges [12], [20], [29]. GNNs excel at capturing both local atomic environments and global physical effects and eliminate the need for handcrafted features through automatic representation learning. However, GNNs also present challenges: they require careful graph construction based on spatial proximity, which may be unstable at high temperatures or during rapid conformational changes, and their complex architectures demand extensive tuning and computational resources. Given these limitations, modeling MD as a standard time-series forecasting provides a more tractable and scalable framework.

V. Discussion

In this paper, we introduce a displacement-based, physics-informed forecasting framework for molecular dynamics. By predicting atomic displacements and incorporating the Morse potential into both training and inference, our method achieves stable and physically plausible simulations over long horizons while remaining computationally efficient. Experimental results on various materials at different temperatures show that our approach not only improves the accuracy of state-of-the-art time-series forecasting models for molecular dynamics but also outperforms conventional MD methods in both precision and efficiency, producing stable trajectories for up to 1,000 simulation steps. Although this marks a significant improvement, small prediction errors still accumulate over time, gradually degrading physical fidelity. Future work will aim to address this by extending the stability and accuracy of the framework to enable robust long-range simulations over 10,000 steps.

VI. Acknowledgment

This research was funded (partially or fully) by the Australian Government through the Australian Research Council. Dr Hung Le is the recipient of an Australian Research Council Discovery Early Career Researcher Award (project number DE250100355) funded by the Australian Government.

References

[1] Martina Audagnotto, Werngard Czechtizky, Leonardo De Maria, Helena Käck, Garegin Papoian, Lars Tornberg, Christian Tyrchan, and Johan Ulander. Machine learning/molecular dynamic protein structure prediction approach to investigate the protein conformational ensemble. *Scientific Reports*, 12(1):10018, 2022.

[2] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2024.

[3] Katsuhiro Endo, Katsufumi Tomobe, and Kenji Yasuoka. Multi-step time series generator for molecular dynamics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[4] Mohammad Javad Eslamibidgoli, Mehrdad Mokhtari, and Michael H Eikerling. Recurrent neural network-based model for accelerated trajectory analysis in aimd simulations. *arXiv preprint arXiv:1909.10124*, 2019.

[5] Cong Fu, Keqiang Yan, Limei Wang, Wing Yee Au, Michael Curtis McThrow, Tao Komikado, Koji Maruhashi, Kanji Uchino, Xiaoning Qian, and Shuiwang Ji. A latent diffusion model for protein structure generation. In *Learning on Graphs Conference*, pages 29–1. PMLR, 2024.

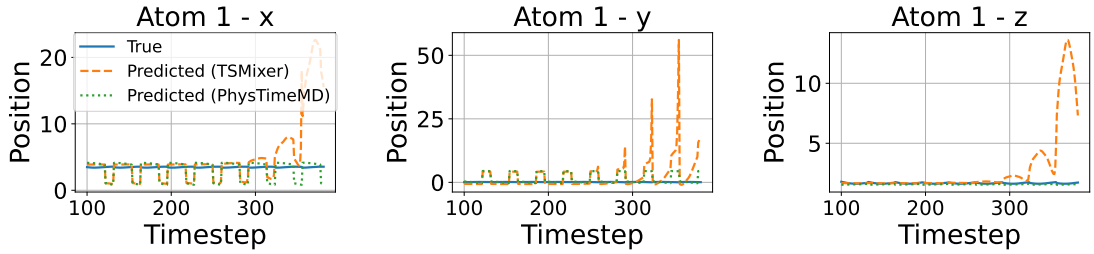

Fig. 5. Comparison of TSMixer and PhysTimeMD-enhanced TSMixer on Dataset E. We visualize the predicted positions of an atom (the same observation was observed for other atoms), each along the three Cartesian axes. The model TSMixer without physics-informed regularization ($\lambda = 0$) exhibits divergence (after 200 timesteps), while incorporating the PhysTimeMD term stabilizes rollout and significantly reduces error.

[6] Helmut Grubmüller, Helmut Heller, Andreas Windemuth, and Klaus Schulten. Generalized verlet algorithm for efficient molecular dynamics simulations with long-range interactions. *Molecular Simulation*, 6(1-3):121–142, 1991.

[7] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[8] Bowen Jing, Hannes Stärk, Tommi Jaakkola, and Bonnie Berger. Generative modeling of molecular dynamics trajectories. *arXiv preprint arXiv:2409.17808*, 2024.

[9] Georg Kresse and Jürgen Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical review B*, 54(16):11169, 1996.

[10] Thomas D Kühne. Second generation car–parrinello molecular dynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 4(4):391–406, 2014.

[11] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[12] Debasis Maji, Atish Ghosh, Debaditya Barman, and Pranab Sarkar. Accelerating molecular dynamics with a graph neural network: A scalable approach through e (q) c-gnn. *The Journal of Physical Chemistry Letters*, 16(9):2254–2264, 2025.

[13] Dominik Marx and Jurg Hutter. Ab initio molecular dynamics: Theory and implementation. *Modern methods and algorithms of quantum chemistry*, 1(301-449):141, 2000.

[14] J Andrew McCammon, Bruce R Gelin, and Martin Karplus. Dynamics of folded proteins. *nature*, 267(5612):585–590, 1977.

[15] Philip M Morse. Diatomic molecules according to the wave mechanics. ii. vibrational levels. *Physical review*, 34(1):57, 1929.

[16] Juno Nam, Sulin Liu, Gavin Winter, KyuJung Jun, Soojung Yang, and Rafael Gómez-Bombarelli. Flow matching for accelerated simulation of atomic transport in materials. *arXiv preprint arXiv:2410.01464*, 2024.

[17] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

[18] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77:3865–3868, Oct 1996.

[19] Mathias Schreiner, Ole Winther, and Simon Olsson. Implicit transfer operator learning: Multiple time-resolution models for molecular dynamics. *Advances in Neural Information Processing Systems*, 36:36449–36462, 2023.

[20] Harikrishnan Sibi, Jovita Biju, and Chandra Chowdhury. Advancing 2d material predictions: superior work function estimation with atomistic line graph neural networks. *RSC advances*, 14(51):38070–38078, 2024.

[21] Sun-Ting Tsai, En-Jui Kuo, and Pratyush Tiwary. Learning molecular dynamics with simple language model built upon long short-term memory neural network. *Nature communications*, 11(1):5115, 2020.

[22] Loup Verlet. Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical review*, 159(1):98, 1967.

[23] Jiaqi Wang, Chengcheng Li, Seungha Shin, and Hairong Qi. Accelerated atomic data production in ab initio molecular dynamics with recurrent neural network for materials research. *The Journal of Physical Chemistry C*, 124(27):14838–14846, 2020.

[24] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[25] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. 2024.

[26] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

[27] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[28] Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, et al. Predicting equilibrium distributions for molecular systems with deep learning. *Nature Machine Intelligence*, 6(5):558–567, 2024.

[29] Tianze Zheng, Weihao Gao, and Chong Wang. Learning large-time-step molecular dynamics with graph neural networks. *arXiv preprint arXiv:2111.15176*, 2021.

VII. Appendix

A. Analyzing Divergence in Autoregressive Forecasts

We observe a divergence (See Fig. 5) where predicted values grow unbounded in the predictions on Dataset E. The comparison involves the TSMixer baseline and our proposed PhysTimeMD variant. The divergence starts to happen around 250 timesteps across all three dimensions of the visualized atom. We hypothesize that the divergence stems from a mismatch between training and inference. During training, ground-truth inputs are used, while inference depends on the model's own predictions, causing errors to accumulate over time. This distribution shift leads to unstable rollouts. PhysTimeMD reduces such instability through physics-informed constraints, resulting in more stable and physically plausible long-term predictions.

B. Hyperparameters Setup

The physics loss coefficient λ is selected from $\{0.001, 0.0005, 0.0001\}$ through tuning, with the best-performing value reported. At each training step, $M = 500$ atom pairs are sampled to compute the energy-based physics loss, which enforces physically plausible interactions. All backbone models are trained using a batch size $B = 16$, an adaptive learning rate $\eta = 0.01$, gradient clipping, and an early stopping strategy with a patience of 3 epochs, applied

TABLE IV

COMPARISON OF MODEL PERFORMANCE WITH AND WITHOUT PHYSTIMEMD ON MULTIPLE DATASETS. FOR EACH MODEL, WE SELECT THE λ VALUE THAT YIELDS THE LOWEST MAE_Δ AND THEN REPORT THAT MAE ALONGSIDE CORRESPONDING, MAE_r, MSE_Δ, MSE_r AND ITS CORRESPONDING $V_r\%$. IN THIS EXPERIMENT, WE CALCULATE THE DISPLACEMENTS AND CONVERT IT BACK INTO THE RAW POSITIONS TO CALCULATE MAE_r AND MSE_r. AN ASTERISK (*) DENOTES CASES WHERE THE MODEL DIVERGED, MAKING MAE, MSE, AND $V_r\%$ UNRELIABLE. **BOLD** VALUES INDICATE IMPROVEMENTS ACHIEVED BY INCORPORATING PHYSTIMEMD.

Dataset	Model	PhysTimeMD	λ	MAE_Δ	MAE_r	MSE_Δ	MSE_r	V_r %
A	TimeMixer	✗	—	6.05×10^{-3}	1.81	6.47×10^{-5}	9.62	4.36
		✓	**0.0001**	$\mathbf{4.60 \times 10^{-3}}$	**0.98**	$\mathbf{3.82 \times 10^{-5}}$	**6.20**	**0.028**
	ITransformer	✗	—	5.94×10^{-3}	2.50	6.07×10^{-5}	13.23	6.08
		✓	**0.0005**	$\mathbf{4.76 \times 10^{-3}}$	**1.01**	$\mathbf{4.00 \times 10^{-5}}$	**6.17**	**0.005**
	Mamba	✗	—	5.37×10^{-3}	2.04	5.25×10^{-5}	11.55	3.59
		✓	**0.0005**	$\mathbf{4.51 \times 10^{-3}}$	**0.96**	$\mathbf{3.64 \times 10^{-5}}$	**6.11**	**0.128**
	TSMixer	✗	—	6.91×10^{-3}	2.32	8.41×10^{-5}	13.78	8.62
		✓	**0.001**	$\mathbf{4.51 \times 10^{-3}}$	**1.10**	$\mathbf{3.68 \times 10^{-5}}$	**6.21**	**0.002**
B	TimeMixer	✗	—	1.48×10^{-3}	0.89	6.76×10^{-6}	4.37	6.06
		✓	0.0005	5.24×10^{-3}	1.13	5.00×10^{-5}	6.78	0.034
	ITransformer	✗	—	7.72×10^{-3}	2.55	1.05×10^{-3}	13.65	10.50
		✓	**0.001**	$\mathbf{5.53 \times 10^{-3}}$	**1.13**	$\mathbf{5.45 \times 10^{-5}}$	**6.73**	**0.090**
	Mamba	✗	—	7.82×10^{-3}	2.58	1.09×10^{-4}	14.85	5.92
		✓	**0.0005**	$\mathbf{6.47 \times 10^{-3}}$	**1.34**	$\mathbf{7.54 \times 10^{-5}}$	**6.61**	**0.117**
	TSMixer	✗	—	7.75×10^{-3}	2.33	1.32×10^{-4}	15.20	11.20
		✓	**0.001**	$\mathbf{4.61 \times 10^{-3}}$	**1.12**	$\mathbf{3.87 \times 10^{-5}}$	**6.39**	**0.058**
C	TimeMixer	✗	—	7.52×10^{-3}	1.89	9.63×10^{-5}	9.12	1.95
		✓	**0.0001**	$\mathbf{5.98 \times 10^{-3}}$	**1.11**	$\mathbf{6.55 \times 10^{-5}}$	**5.92**	**0.000**
	ITransformer	✗	—	9.66×10^{-3}	3.34	1.50×10^{-4}	24.48	24.60
		✓	**0.0005**	$\mathbf{6.06 \times 10^{-3}}$	**1.06**	$\mathbf{6.69 \times 10^{-5}}$	**5.94**	**0.004**
	Mamba	✗	—	6.92×10^{-3}	2.14	8.44×10^{-5}	10.42	2.64
		✓	**0.0001**	$\mathbf{5.89 \times 10^{-3}}$	**1.12**	$\mathbf{6.36 \times 10^{-5}}$	**5.69**	**0.196**
	TSMixer	✗	—	1.33×10^{-2}	3.87	3.74×10^{-4}	40.46	17.70
		✓	**0.0005**	$\mathbf{5.84 \times 10^{-3}}$	**1.19**	$\mathbf{6.31 \times 10^{-5}}$	**6.11**	**0.073**
D	TimeMixer	✗	—	7.47×10^{-3}	2.57	9.72×10^{-5}	14.69	8.08
		✓	**0.001**	$\mathbf{5.94 \times 10^{-3}}$	**1.13**	$\mathbf{6.51 \times 10^{-5}}$	**6.08**	**0.026**
	ITransformer	✗	—	7.87×10^{-3}	2.43	1.06×10^{-4}	12.49	6.08
		✓	**0.0001**	$\mathbf{6.60 \times 10^{-3}}$	**1.41**	$\mathbf{7.88 \times 10^{-5}}$	**6.99**	**0.026**
	Mamba	✗	—	7.58×10^{-3}	2.54	9.99×10^{-5}	13.73	5.12
		✓	**0.0005**	$\mathbf{6.48 \times 10^{-3}}$	**1.36**	$\mathbf{7.54 \times 10^{-5}}$	**6.73**	**0.018**
	TSMixer	✗	—	8.66×10^{-3}	2.50	2.34×10^{-4}	15.88	6.06
		✓	**0.0005**	$\mathbf{5.86 \times 10^{-3}}$	**1.15**	$\mathbf{6.35 \times 10^{-5}}$	**6.06**	**0.092**
E	TimeMixer	✗	—	6.63×10^{-3}	1.74	7.29×10^{-5}	7.95	9.85
		✓	**0.0001**	$\mathbf{6.17 \times 10^{-3}}$	**0.82**	$\mathbf{6.50 \times 10^{-5}}$	**4.49**	**6.76**
	ITransformer	✗	—	7.15×10^{-3}	1.31	8.57×10^{-5}	5.40	17.00
		✓	**0.001**	$\mathbf{6.38 \times 10^{-3}}$	**1.31**	$\mathbf{6.89 \times 10^{-5}}$	**5.40**	**7.69**
	Mamba	✗	—	6.74×10^{-3}	1.71	7.74×10^{-5}	8.29	20.40
		✓	**0.0001**	$\mathbf{6.19 \times 10^{-3}}$	**0.91**	$\mathbf{6.54 \times 10^{-5}}$	**4.56**	**8.52**
	TSMixer	✗	—	*	*	*	*	*
		✓	**0.001**	$\mathbf{6.20 \times 10^{-3}}$	**0.89**	$\mathbf{6.55 \times 10^{-5}}$	**4.59**	**7.85**
F	TimeMixer	✗	—	7.32×10^{-3}	1.06	8.94×10^{-5}	4.59	2.62
		✓	**0.001**	$\mathbf{6.88 \times 10^{-3}}$	**0.73**	$\mathbf{8.08 \times 10^{-5}}$	**3.78**	**2.31**
	ITransformer	✗	—	7.53×10^{-3}	1.47	8.57×10^{-5}	6.80	6.33
		✓	**0.0001**	$\mathbf{6.97 \times 10^{-3}}$	**0.78**	$\mathbf{8.29 \times 10^{-5}}$	**3.78**	**2.71**
	Mamba	✗	—	7.51×10^{-3}	1.64	9.62×10^{-5}	7.26	13.70
		✓	**0.0005**	$\mathbf{6.95 \times 10^{-3}}$	**0.78**	$\mathbf{8.23 \times 10^{-5}}$	**3.78**	**2.93**
	TSMixer	✗	—	1.06×10^{-2}	1.06	4.59×10^{-4}	4.59	6.52
		✓	**0.0001**	$\mathbf{6.88 \times 10^{-3}}$	**0.77**	$\mathbf{8.06 \times 10^{-5}}$	**3.76**	**2.60**
G	TimeMixer	✗	—	5.87×10^{-3}	2.16	6.60×10^{-5}	14.54	15.40
		✓	**0.001**	$\mathbf{3.79 \times 10^{-3}}$	**1.18**	$\mathbf{3.23 \times 10^{-5}}$	**9.54**	**11.40**
	ITransformer	✗	—	8.21×10^{-3}	1.94	1.11×10^{-4}	9.04	11.50
		✓	**0.0005**	$\mathbf{7.14 \times 10^{-3}}$	**0.74**	$\mathbf{8.68 \times 10^{-5}}$	**3.78**	**2.81**
	Mamba	✗	—	1.13×10^{-2}	3.86	2.55×10^{-3}	72.10	32.60
		✓	**0.0005**	$\mathbf{3.86 \times 10^{-3}}$	**1.22**	$\mathbf{3.30 \times 10^{-5}}$	**9.81**	**12.10**
	TSMixer	✗	—	2.48×10^{-2}	7.17	1.20×10^{-2}	423.99	45.80
		✓	**0.001**	$\mathbf{3.79 \times 10^{-3}}$	**1.21**	$\mathbf{3.21 \times 10^{-5}}$	**9.65**	**11.80**

over a maximum of 10 epochs. This training setup ensures stable and efficient convergence while maintaining physical consistency in predictions.

C. Benchmarking with Time-Series Baselines

See Tab. IV for detailed results of PhysTimeMD.