Revising the N-body Problem as a Benchmark for Geometric Deep Learning

Patrik Šimurka, Martin Karas, Juraj Mečír, Andrej Kozák, Ondrej Hovorka, Pavol Skubák, Ján Kmeťko
Simona Biosystems, Bratislava, Slovakia
patrik.simurka@simonabio.com

Code available at https://github.com/Simona-Biosystems/Revising-the-N-body-Problem-as-a-Benchmark-for-Geometric-Deep-Learning

Abstract

The N-body problem deals with predicting the trajectories of bodies by solving Newton's equations of motion using a numerical integrator[1]. Physical constraints require small time steps to maintain sufficient precision. This limits the applicability and usefulness of such simulations. Training a neural network to predict these trajectories using larger time steps than the integrator while maintaining accuracy can alleviate this problem. Due to its nature, the problem also lends itself well to be used as a benchmark for geometric deep learning. However, we argue that it has not yet been formalized properly as such in the literature. This paper aims to change that and (re)define how the N-body problem is used for this purpose. To this end, we take the Ponita model[2] (state-of-the-art, well-documented equivariant graph neural network) and predict the evolution of the system. To evaluate the generated trajectories and underline the importance of holistic evaluation, we introduce a comprehensive suite of so-called "macro-property tests".

1 Introduction

The practicality of dynamical simulations of many-body phenomena in physics, chemistry, and biology is often limited by computational complexity. This complexity arises from the vast number of interacting degrees of freedom evolving through multiple timescales. Many-body simulations typically rely on iteratively propagating coupled dynamical equations of motion over time, requiring time steps smaller than the fastest degree of freedom present in the system.

Consequently, the number of time steps needed to access long timescales of experimental observables becomes computationally prohibitive.

Training neural networks to predict these trajectories using larger time steps than traditional numerical integrators—while maintaining accuracy—can alleviate this problem. This approach doubles as a promising candidate for use as a benchmark for geometric deep learning.

However, benchmarking the predictions of the general N-body problem is challenging due to the potential for chaotic particle motion and the absence of physical constraints like the Boltzmann distribution in equilibrium molecular motion. To address this challenge, we developed a suite of statistical macro-property tests to visualize and quantify different probabilistic aspects of particle trajectories. We demonstrate the effectiveness of our approach by comparing these statistical measures between ground truth trajectories (generated using the integrator) and predicted trajectories (generated using the trained network) and showing that our benchmark successfully gauges the validity of generated trajectories by observing how the resulting macro-properties improve with more training and worsen with increasing the step size used for prediction.

With regards to the scope of this work, we decided to only experiment on the Ponita architecture, but our results naturally generalize to other classes of networks as well. We perform all experiments on the gravitational N-body problem, operating under the assumption that our findings are transferable to more complex dynamics, such as those in molecular dynamics—a hypothesis we plan to scrutinize in future work.

2 Related Work

Neural networks based on geometric deep learning are being rapidly developed to tackle the computational complexity in many-body simulations. Such state-of-the-art neural network architectures include Ponita, SEGNN[3] SE(3)-transformer[4], tensor field networks[5], and Clifford group equivariant neural networks[6], to name a few examples. These architectures have been reported to achieve considerably improved performance metrics (e.g. mean squared error) while requiring fewer training samples than classical deep learning models.

However, most of these studies focus on showcasing the generality and applicability of the network architectures across different mathematical problem classes, and as such, evaluating the accuracy of predictions regarding many-body dynamics and the statistical properties of associated macroscopic observables often falls beyond their scope. These limitations include:

a) evaluating certain subsets of statistical properties (such as only looking at energy conservation)[7–9];
b) not evaluating self-feed (rollout) performance and instead focusing on what can be called "static inference" (predicting just one future step in isolation)[8,9];
c) focusing on "micro-properties" such as specific trajectories and their alignment to the corresponding ground truth trajectories (which will never match exactly due to floating point rounding error accumulation – "butterfly effect")[10].

Therefore, in this work, we extend beyond evaluating performance on the training objective by quantifying the ability of neural networks to accurately predict the actual dynamics over extended periods. To achieve this, we developed a suite of statistical tests for evaluation, emphasizing the macro-properties of the system.

3 Defining the Benchmark

To establish a robust and meaningful benchmark for evaluating geometric deep learning models on the N-body problem, we propose a standardized framework that addresses the shortcomings identified in previous studies. Our approach emphasizes the following key components:

Rollout Evaluation
Rather than focusing solely on static inference—predicting one future step in isolation—our benchmark evaluates models based on their rollout performance. This involves iteratively feeding the model's predictions back as inputs for subsequent steps, simulating the dynamics over extended periods. Rollout evaluation is crucial for assessing how errors accumulate over time and whether the model can sustain accurate predictions without diverging from realistic behavior.

Focus on Macro-Properties
Previous benchmarks often concentrated on predicting micro-level properties, such as the alignment of the individual generated trajectories to the ground truth. However, given the chaotic nature of the N-body problem and the sensitivity to initial conditions, exact trajectory matching is impractical due to the accumulation of floating-point rounding errors—the so-called "butterfly effect". Instead, our benchmark prioritizes the accurate prediction of macro-properties—statistical measures that capture the emergent behavior of the system. These include the frequency of events like collisions, escapes, sharp turns, and particle stickings, which are more relevant to practical applications and provide a holistic evaluation of model performance.

Standardization Using the Ponita Architecture
We selected the Ponita model as the foundational architecture for this benchmark due to its state-of-the-art performance and good documentation. By providing detailed implementation specifics—such as hyperparameters, training procedures, and evaluation metrics—we ensure that the benchmark is replicable and serves as a fair platform for comparing different models.

Comprehensive Statistical Testing
The benchmark includes a suite of macro-property tests designed to visualize and quantify different probabilistic aspects of particle trajectories. We compare the distributions of macro-properties between the model's predictions and the ground truth generated by the integrator. This comparison utilizes statistical methods, notably the

Kolmogorov–Smirnov two-sample test, to numerically assess the similarity of distributions and quantify deviations.

By integrating these components, our benchmark provides a rigorous and comprehensive framework for evaluating geometric deep learning models on the N-body problem. It moves beyond traditional metrics that may not fully capture the quality of the simulation and instead focuses on emergent behaviors that are critical in many practical applications. We believe that this benchmark will facilitate more meaningful comparisons and drive improvements in modeling many-body dynamics using geometric deep learning techniques.

It is worth noting that the details of how we compute the macro-properties (e.g. the specific distance and time thresholds) are not that important as long as we are consistent when computing macro-properties of the ground-truth vs predicted data. However, for exact reproducibility, the code that we used will be published.

Main Macro-Properties Considered:

Number of Collisions:

- *Definition*: Counts the instances where any two bodies come into contact or pass within a predefined proximity threshold.
- *Significance*: Reflects the frequency of direct interactions between bodies, indicative of the system's density and dynamism.

Number of Group Collisions:

- *Definition*: Counts collisions involving a pair and a triplet of bodies (which, in case of 5-body simulations, means all 5 bodies)
- *Significance*: Similar to number of collisions, but our initial motivation for including this test was to see how accurately rare events are captured.

Number of Escapes:

- *Definition*: Measures how many bodies exceed a certain distance threshold from the system's center of mass.
- *Significance*: Indicates the tendency of bodies to disperse, reflecting the balance between gravitational attraction and kinetic energy.

Number of Sharp Turns (30° Threshold):

- *Definition*: Counts occurrences where a body's velocity vector changes direction by more than 30 degrees between consecutive time steps.
- *Significance*: Captures abrupt changes in motion, which may be due to close encounters or interactions with other particles.

Number of Sharp Turns (45° Threshold):

- *Definition*: Similar to the previous test but with a 45-degree threshold.

- *Significance*: Focuses on even more significant directional changes, highlighting extreme dynamic events.

Number of Stickings:
- *Definition*: Identifies events where particles remain in close proximity (within a specified threshold) for a set number of time steps or longer.
- *Significance*: Reflects temporary or permanent clustering behavior.

Additional Macro-Properties

For a more complete view, we also examine various other statistical properties, such as energies (and total energy conservation) and trajectory plots. However, only the six main macro-properties listed in the previous section are used to calculate numerical performance using the Kolmogorov–Smirnov two-sample test.

4 Methodology

Building upon the defined benchmark, we proceeded to train the Ponita model using the specified procedures and evaluated its performance using the macro-property tests.

Training the Model

We trained the Ponita model on trajectories generated with the Verlet integrator, using the code from EGNN[11] (a simple Verlet integrator implementation which many subsequent works, including Ponita, also use).

Using our macro-property tests, we determined that a time step of 0.01 (unitless[12]) is small enough for generation, such that the macro-properties are unchanged when compared to smaller time steps. We verified this assumption up to a time step of $1*10^{-5}$ and saw no signs of divergence. Further investigation revealed that we could probably get away with a time step of 0.05, but we will concern ourselves with this fact in future works. To determine this "largest accurate time step", we found looking at the energy conservation statistic mentioned in the previous section was generally enough, as its behavior closely mirrored the behavior of other macro-properties as we varied the time step. We illustrate how we used it to guide our thinking in figure 1 (below).

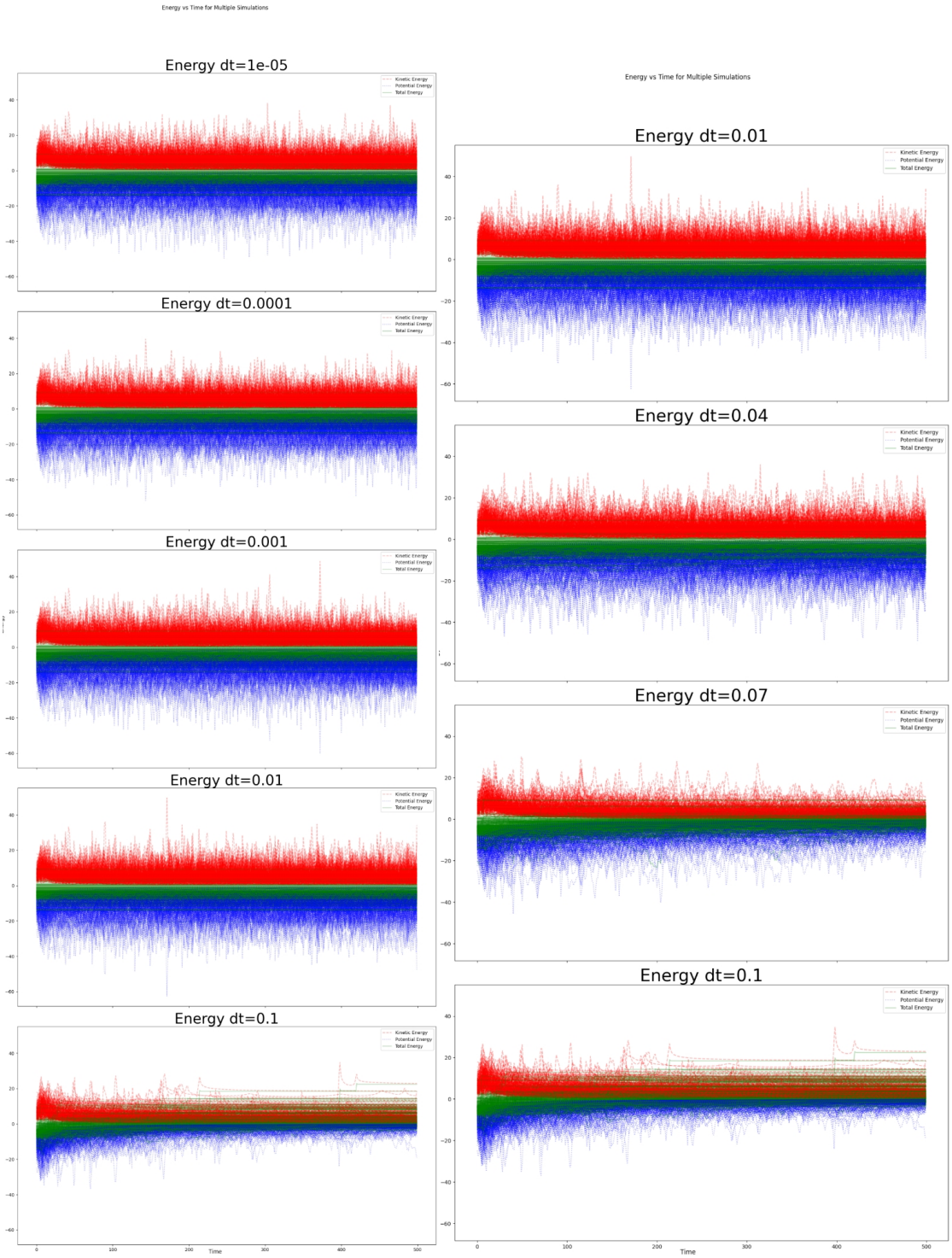


Figure 1: Plots of how kinetic (red), potential (blue), and total (green) energy evolved for the 500 simulated steps across the 128 trajectories in the batch generated with the Velvet integrator.

Having generated the ground truth trajectories using the Verlet integrator, we took every *N*-th step and used these to train the model. Since we were interested in seeing if the model can learn to "simulate" with time steps larger than those required by the integrator, we tried various *N*'s. However, to demonstrate the effectiveness of our benchmark (and to not bloat the scope of this paper), all results presented in the results section were achieved with *N*=10. For curious readers, we also provide results for *N*=40 in the appendix.

It is worth noting that while it is very encouraging that the model seems to be capable of accurately modeling the dynamics using a time step larger than the integrator, for this to be practical the inference must not take much longer (more than 10x longer including overhead for *N*=10, for example). We will be concerning ourselves with these practical aspects in future works, since this undertaking is of considerable scope: trying out different integrators and models, scaling the models, and finding optimal hyperparameters, to mention just a few.

Returning to training, our objective was, given the positions and velocities of the bodies at some step, to simply predict the ones in the next step. Since we generated new trajectories on-the-fly (during training), the model never saw the same one twice. This has the obvious advantage that we did not need to worry about overfitting. We used the Mean Squared Error (MSE) criterion for both the positions and velocities. AdamW[13] Optimizer was used, with an inverse square root decay with warmup learning rate schedule, with the learning rate calculated as follows: *model_size**(-0.5)*min(step**(-0.5),step*warmup_steps**(-1.5))*

We use *warmup_steps*=2048 and *model_size=hidden_dim*=128

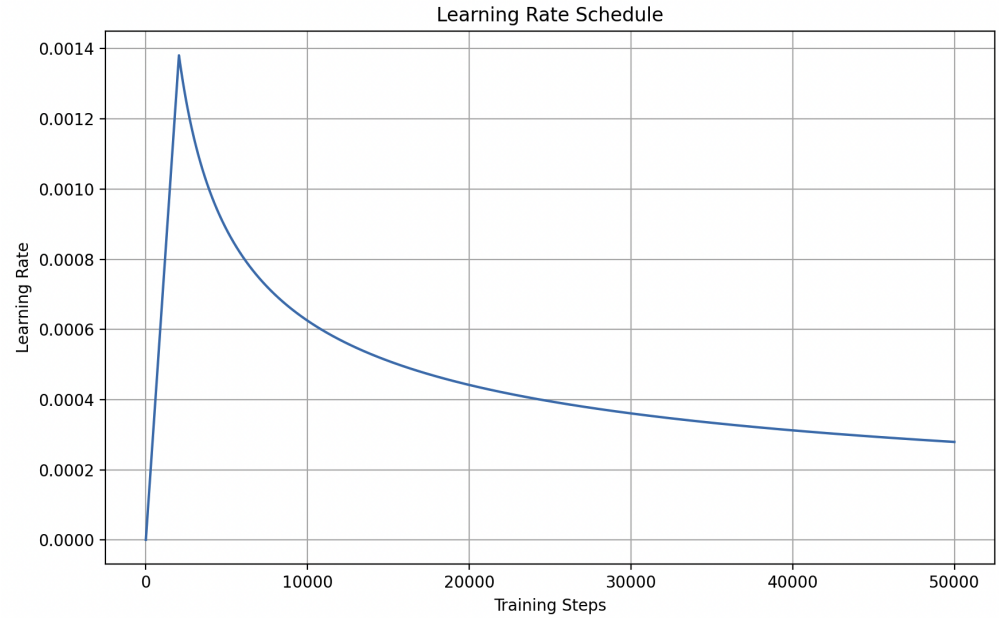


Figure 2: The learning schedule used.

In terms of model parameters, we used *hidden_dim*=128, *n_layers*=5, *radius*=None, *num_ori*=20, *basis_dim*=256, *degree*=3, *widening_factor*=4, and *multiple_readouts*=True.

For the sake of simplicity, we only used simulations of 5 bodies for training. Scaling to different numbers of bodies will be the subject of future works.

Obtaining the Macro-Properties

To obtain the macro-properties, we used batches of 128 trajectories. For each trajectory, both the integrator (*ground truth*) and the model (*predicted*) start with the same initial configuration. As with training—courtesy of our on-the-fly method—the configuration is generated at random and thus new to the model.

To obtain the ground truth, we used the integrator to predict *X* steps. For the model predictions, we used the trained neural network (SEGNN) to iteratively predict *X*//*N* steps, feeding each prediction back as input for the next iteration. Since the network was trained to predict every *N*th step, the time length of the simulated trajectory is effectively the same as that of the ground truth integrator's. This allows us to compare both the trajectories themselves (not crucial) and the macro-properties (crucial) side-by-side.

Statistical Analysis Using the Kolmogorov–Smirnov Test

To numerically quantify how well the macro-properties of the inferred trajectories match those of the ground truth trajectories, we employed the Kolmogorov–Smirnov (K-S) two-sample test. The K-S test is a non-parametric test that measures the maximum difference (*D*) between the empirical cumulative distribution functions (CDFs) of two samples. It is defined as:

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)|$$

where $F_{1,n}$ and $F_{2,m}$ are the empirical CDFs of the two samples.
The advantages of using the K-S test in our context are:

- Simplicity and Intuitiveness: It provides an elegant and straightforward measure of the difference between two distributions, which is accessible even to those unfamiliar with advanced statistical methods. You can find one such intuitive explanation of the KS test and the reasoning behind choosing it in the appendix.
- Non-parametric Nature: The K-S test does not assume any specific distribution for the data, making it suitable for our diverse set of macro-properties.

We applied the K-S test in the following way:

1. Baseline Comparison: As a sanity check, we first compared two batches of ground truth trajectories to confirm that they come from the same distribution. This provided a baseline K-S statistic and p-value.
2. Model Evaluation: We then compared the distributions of macro-properties from the model's inferred trajectories with those from the ground truth trajectories. By calculating

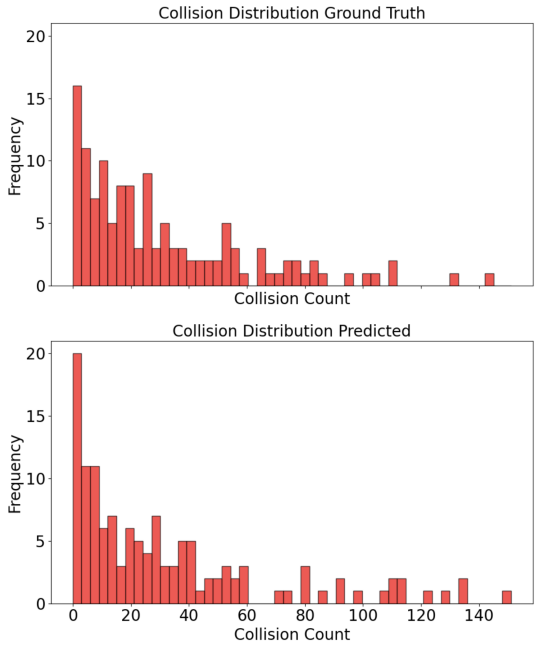
the K-S statistic and p-value for each macro-property, we quantified the differences between the model predictions and the ground truth.

3. Obtaining Combined P-Values: To get a single number expressing the overall difference of the macro-properties of model predictions and the ground truth, we needed a way to combine the p-values of the individual macro-properties obtained in the previous step. We opted to use the Fisher method. We won't go into details of the Fisher method as it is easy enough to look up, and the technical minutia would only serve to distract here. More importantly, this method is very sensitive to outliers (much more so than a simple average), which is important, since the resulting (combined) p-value should never be large if one of the constituent p-values of the individual macro-properties is really low.
4. Interpretation of the combined p-value: While p-values are commonly used to reject or fail to reject a null hypothesis at a certain significance level (e.g., 0.05), we recognize that overreliance on arbitrary thresholds can be misleading. Therefore, we used the p-values and K-S statistics as continuous measures of discrepancy rather than strict decision criteria.
5. Visualization of Training Convergence: To assess training convergence, we plotted the combined p-value as a function of the training step. This allowed us to observe how the model's performance improved over time.

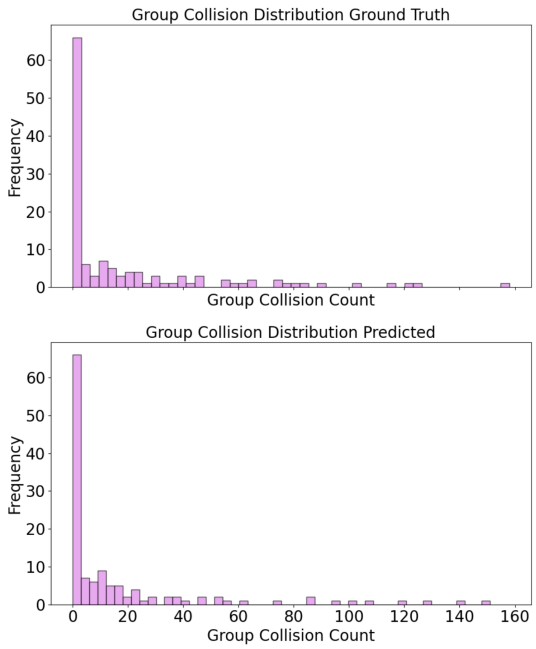
5 Results

Observed Macro-Properties

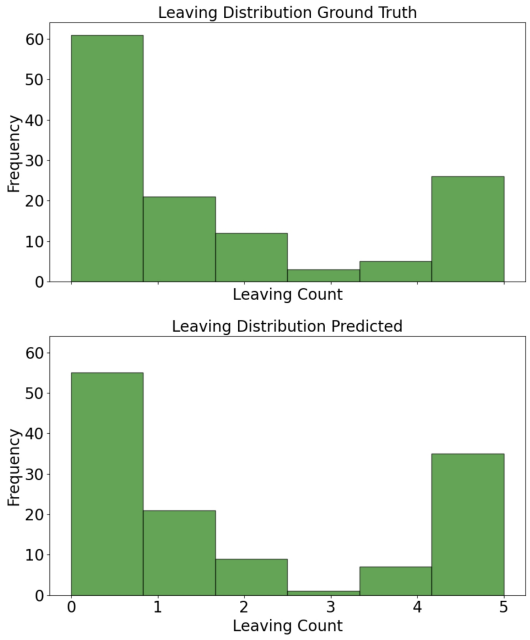
As shown in figure 3, the macro-properties obtained from the model trained with N=10 are quite close to ground truth counterparts. We numerically quantify this in the next section.


Collision Distribution Ground Truth
Frequency
Collision Count
Collision Distribution Predicted
Frequency
Collision Count

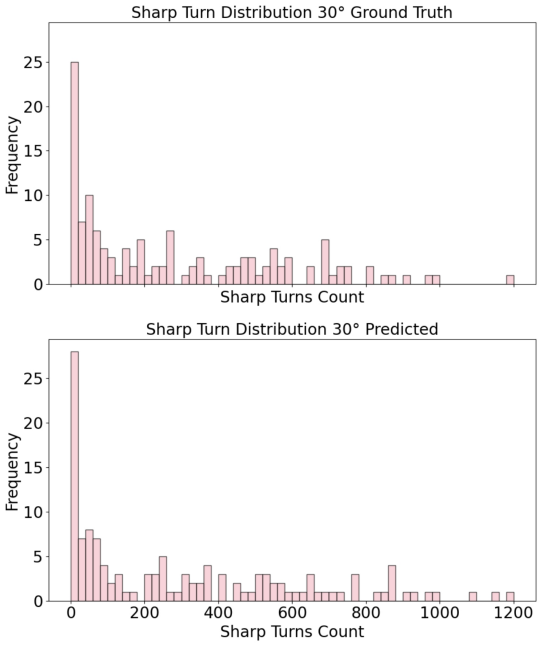
a)


Group Collision Distribution Ground Truth
Frequency
Group Collision Count
Group Collision Distribution Predicted
Frequency
Group Collision Count

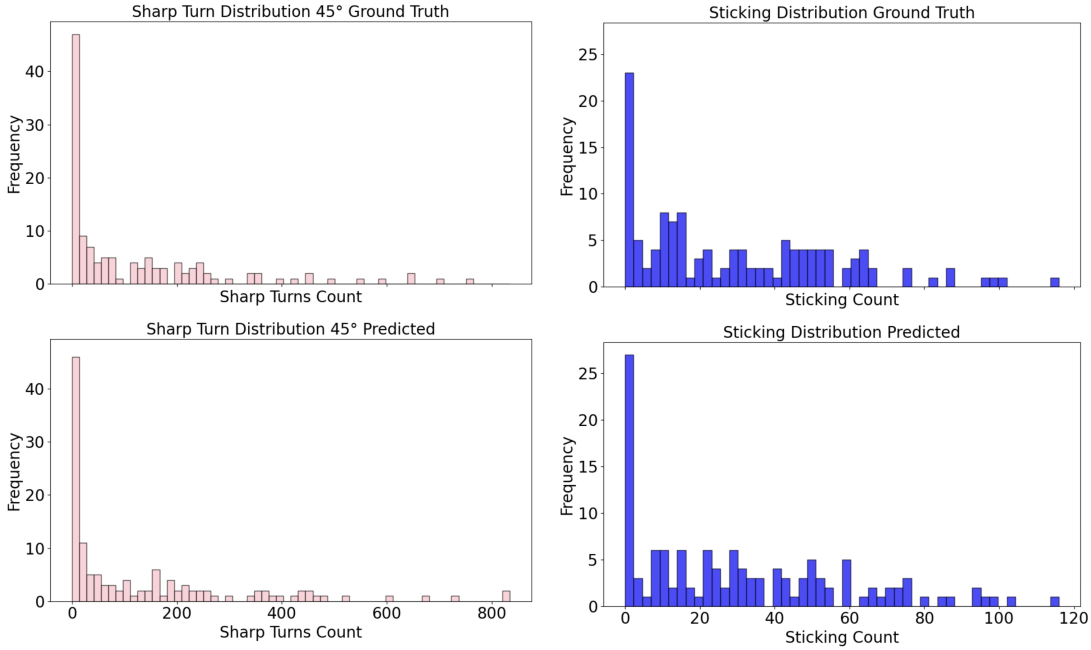
b)


Leaving Distribution Ground Truth
Frequency
Leaving Count
Leaving Distribution Predicted
Frequency
Leaving Count

c)


Sharp Turn Distribution 30° Ground Truth
Frequency
Sharp Turns Count
Sharp Turn Distribution 30° Predicted
Frequency
Sharp Turns Count

d)



e) f)

Figure 3: The main macro-properties of the inferred trajectories (128 of them) compared to ground truth. Each data point represents the number of times an event of a certain type occurred in a given trajectory. As such, each macro-property distribution consists of 128 data points (one for each trajectory). We distinguish—and show the distribution of—the following types of events. It's worth noting that the specifics of what constitutes an event (such as the minimum distance threshold for 'escapes') are consistent between ground truth and inferred data.

a) Number of collisions;
b) Same as a), but while a) counts collisions of any two particles, b) counts collisions of a pair and a triplet of particles;
c) Number of 'escapes' (how many bodies went beyond a distance threshold from the center of mass);
d) Number of sharp turns (defined as the number of times that a body's velocity vector turns by more than 30 degrees);
e) Same as d), but with a 45-degree threshold;
f) Number of "stickings" (similar to collisions, but instead of rebounding, the bodies stay together for a set number of time steps or longer).

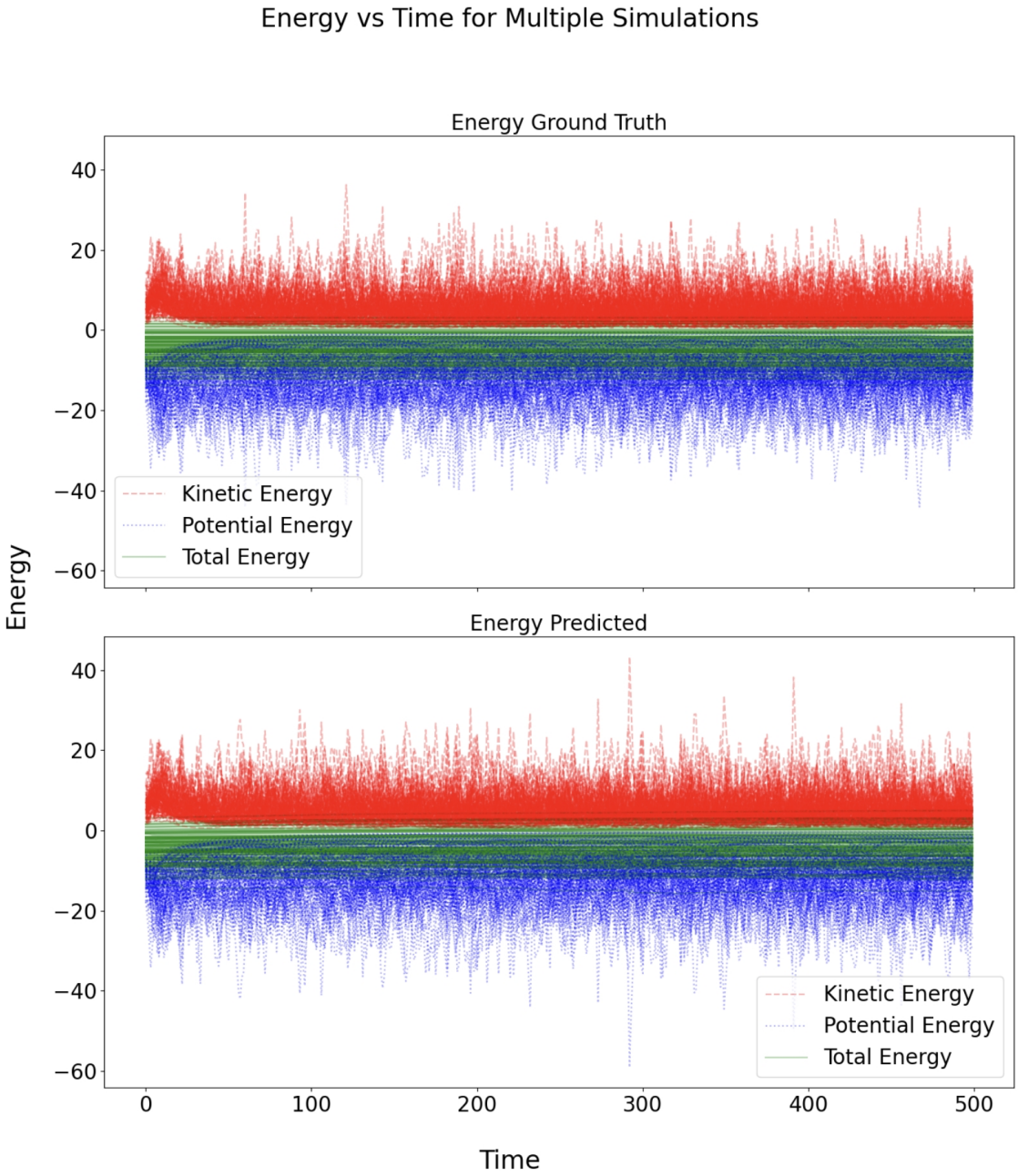


Figure 4: Auxiliary macro-property – energy conservation – visualized: kinetic, potential, as well as total (kinetic + potential) energy of the system.

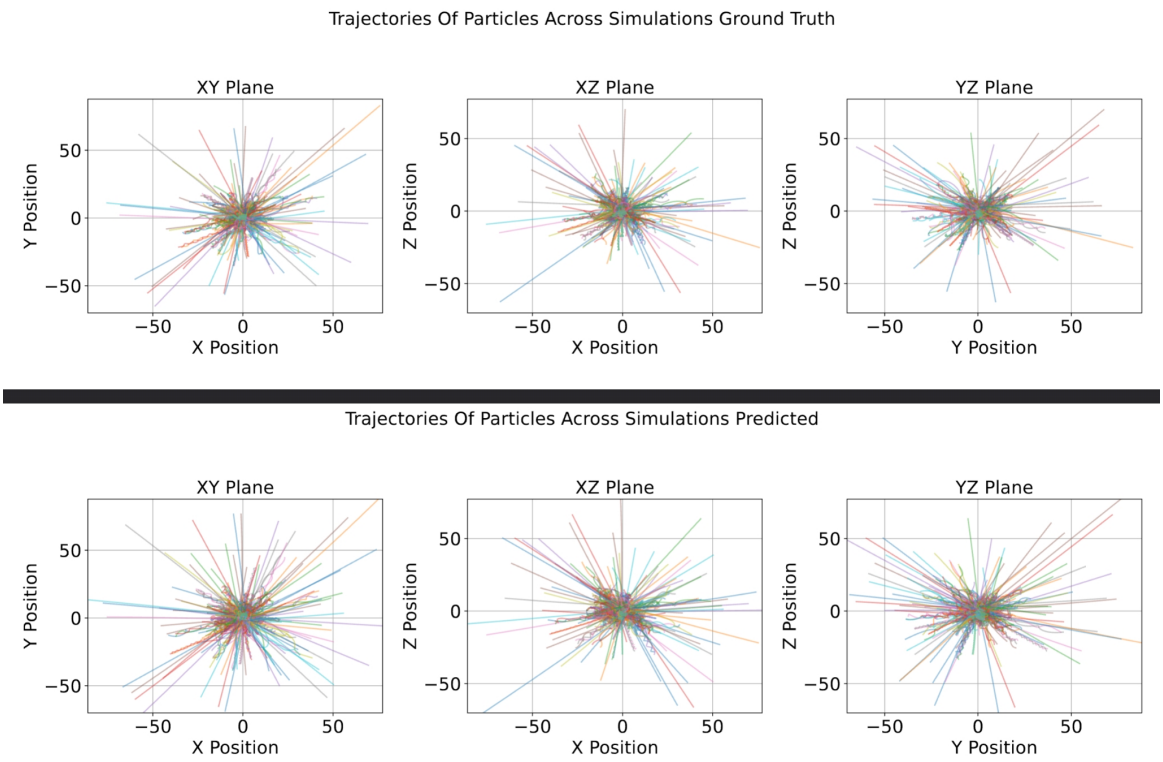


Figure 5: Auxiliary macro-property – all the batch trajectories (128 of them) – visualized in their entirety from 3 planes.

Results of the Statistical Analysis

To numerically evaluate the alignment between the macro-properties of the inferred trajectories and those of the ground truth, we applied the K-S test as described in the Methodology section.

Baseline Verification

- Ground Truth vs. Ground Truth: The K-S test comparisons between two ground truth trajectory batches yielded high p-values (close to 1) and low K-S statistics, confirming that they come from the same distribution. This baseline verifies that the test functions as expected in our context.

Model Evaluation Findings

- Inferred vs. Ground Truth: For N=10, the p-values obtained from comparing the model's inferred trajectories with the ground truth were relatively high (nearing 0.05 towards the end of the training run), and the K-S statistics were low, indicating that the distributions are statistically similar.
- Effect of Increasing Time Steps: As N increased, we observed a gradual decrease in p-values and an increase in K-S statistics, reflecting a divergence between the inferred and ground truth distributions. This trend highlights the limitations of the model at larger time steps and emphasizes the importance of selecting an appropriate N.

Visualization

- Training Convergence: Figure 6 shows the combined p-values plotted against the training steps for a training run using *N*=10. The convergence of the p-values towards higher values over training steps indicates that the model is learning to align its predictions more closely with the ground truth distributions. A reference line at p-value = 0.05 is included to provide context, although we caution against rigidly interpreting this threshold.

Interpretation

These results demonstrate that our model can generalize well and has learned the underlying dynamics governing gravitational motion. The macro-properties of the inferred trajectories closely match those of the ground truth, validating our approach. The successful application of the K-S test as a quantitative metric reinforces the utility of macro-properties in evaluating N-body simulations.

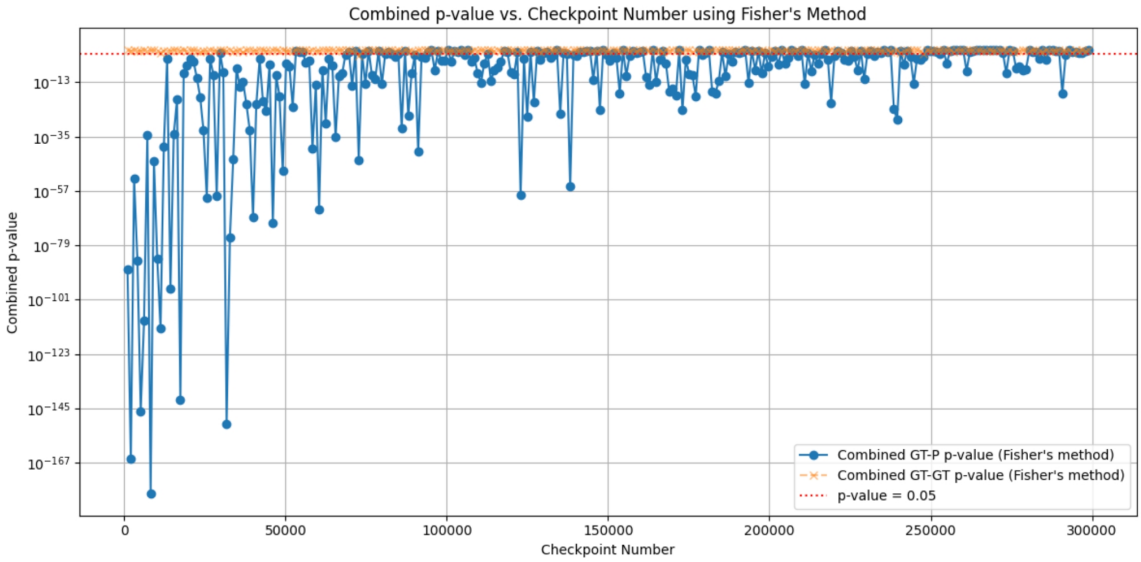


Figure 6: Combined p-values plotted against the checkpoint (training step) number of the training run whose macro-properties are visualized in figures 2-4.

The Correspondence Between Observed Macro-Properties and Statistical Analysis (KS-Test P-Values)

To illustrate how our statistical analysis is useful to isolate checkpoints (or models) that have learned to predict the underlying dynamics with good accuracy (as determined by our macro-property tests), we attach figure 7 that shows selected macro-properties of the following 2 snapshots of the training run visualized in figure 6:

1. One of the initial checkpoints, combined p-value is very low
2. A checkpoint halfway into the training, combined p-value is mediocre
3. The best checkpoint (late in the training) that is visualized in figures 3-5.

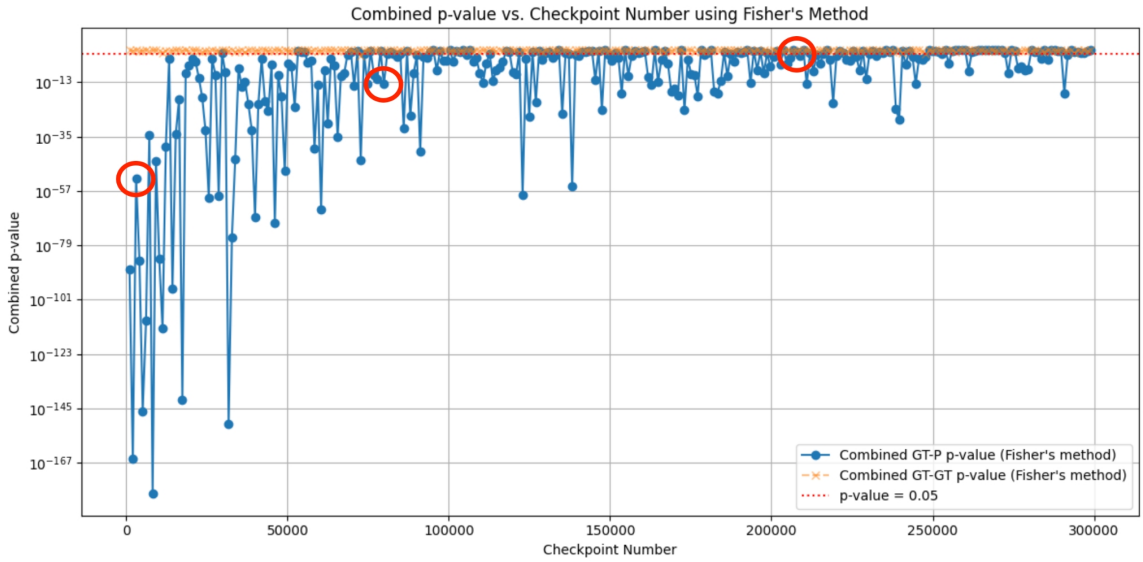


Figure 7: The same plot as in figure 6, with checkpoints that are visualized in figures 3-5 and 8-9 highlighted.

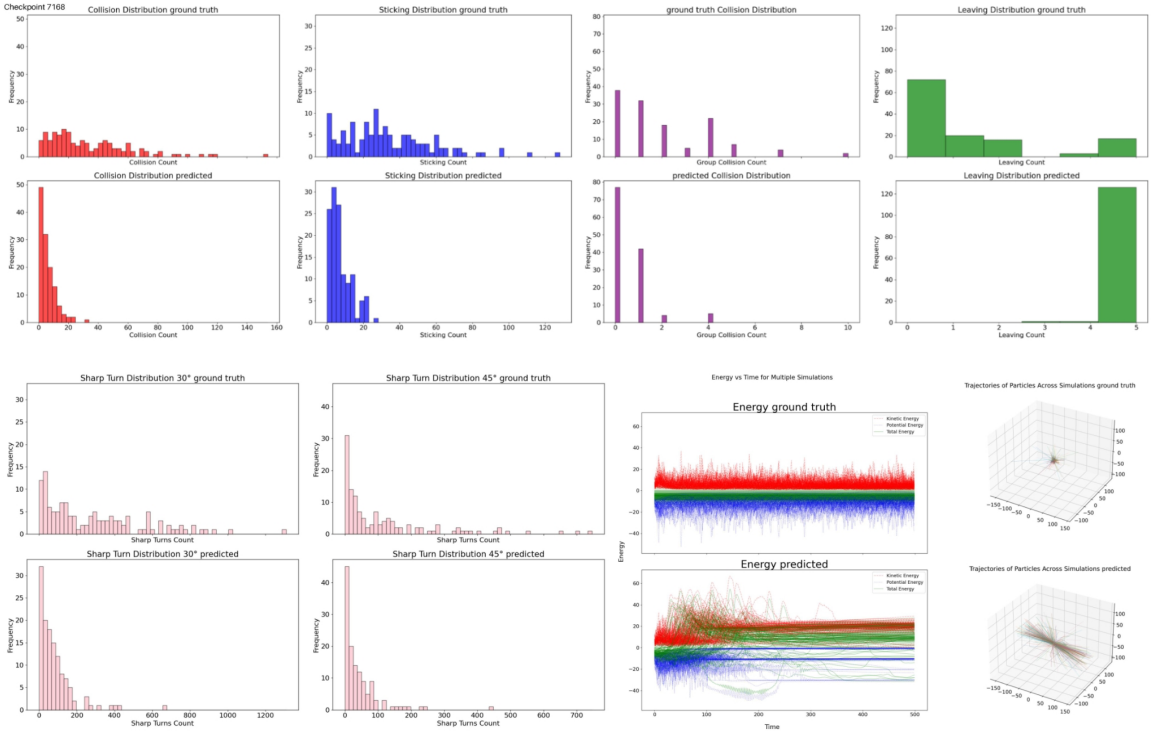


Figure 8: Macro-properties of one of early checkpoints (first highlighted in figure 7).

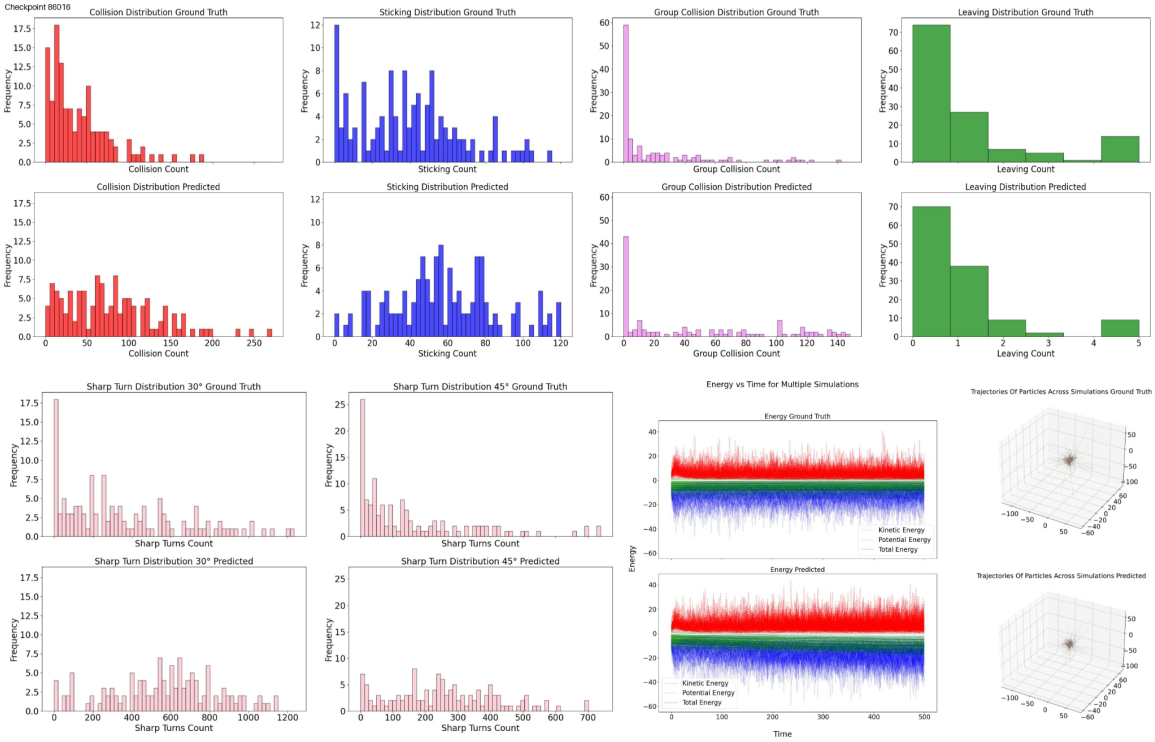


Figure 9: Macro-properties of one of mid-training checkpoints (second highlighted in figure 7).

6 Discussion & Future Work

We showed that neural networks can learn to simulate the N-body problem close to reality, perhaps with even bigger time steps than a simple Verlet integrator, and we substantiated this claim with our suite of macro-properties. But a few big questions (and hopefully research directions) remain:

Efficiency in Wall-Clock Time

- Are these "shortcuts" sufficient to achieve faster simulations in terms of actual computational time?
- How far can we push this by employing better architectures, larger models, or more (and better) training data?

Scaling to Larger Systems

- How does the performance change as we increase the number of bodies?
- Can models trained on a small number of bodies generalize to systems with many more bodies?

Impact of Different Integrators

- How would using different numerical integrators to generate ground truth trajectories influence the results?

Application to Molecular Dynamics

- How does this translate to specific, more complex problems, such as those in molecular dynamics?

- While the N-body problem may seem toy-ish, we believe it's also fundamental – and while we think our diverse macro-property tests give us reasonable assurance that we are approximating the fundamental function governing N-body-like dynamics (which extend to the domain of molecular dynamics also), there still may be undersampled regions which our model never encountered (and never generalized to) and which may or may not prove relevant when we try to transfer to actual molecular dynamics.

We plan to explore these questions in future works, aiming to further validate and extend our approach.

7 Conclusion

We introduce a comprehensive suite of macro-property tests to evaluate trajectories of the gravitational N-body problem generated by a state-of-the-art geometric neural network – Ponita – in an attempt to address limitations in previous approaches and redefine how the N-body problem is used as a benchmark for geometric deep learning. The results suggest that neural networks can learn the dynamics of gravitational simulations effectively while preserving important macro-properties. Our hope is that this opens up new possibilities for more efficient N-body simulations and potentially extends to other, more complex domains like molecular dynamics.

References

1. Kipf, T., Fetaya, E., Wang, K.-C., Welling, M. & Zemel, R. Neural Relational Inference for Interacting Systems. in *Proceedings of the 35th International Conference on Machine Learning* 2688–2697 (PMLR, 2018).
2. Bekkers, E. J., Vadgama, S., Hesselink, R. D., van der Linden, P. A. & Romero, D. W. Fast, Expressive SE(n) Equivariant Networks through Weight-Sharing in Position-Orientation Space. Preprint at http://arxiv.org/abs/2310.02970 (2024).
3. Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J. & Welling, M. Geometric and Physical Quantities Improve E(3) Equivariant Message Passing. Preprint at https://doi.org/10.48550/arXiv.2110.02905 (2022).
4. Fuchs, F. B., Worrall, D. E., Fischer, V. & Welling, M. SE(3)-Transformers: 3D Roto-

Translation Equivariant Attention Networks. Preprint at http://arxiv.org/abs/2006.10503 (2020).

5. Thomas, N. *et al.* Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. Preprint at http://arxiv.org/abs/1802.08219 (2018).
6. Ruhe, D., Brandstetter, J. & Forré, P. Clifford Group Equivariant Neural Networks. Preprint at https://doi.org/10.48550/arXiv.2305.11141 (2023).
7. Li, X., Lu, F., Tao, M. & Ye, F. NySALT: Nystr\"{o}m-type inference-based schemes adaptive to large time-stepping. *J. Comput. Phys.* **477**, 111952 (2023).
8. Mohanty, S., Yoo, S., Kang, K. & Cai, W. Evaluating the Transferability of Machine-Learned Force Fields for Material Property Modeling. *Comput. Phys. Commun.* **288**, 108723 (2023).
9. Chen, T. *et al.* GeoMFormer: A General Architecture for Geometric Molecular Representation Learning. Preprint at https://doi.org/10.48550/arXiv.2406.16853 (2024).
10. Zheng, T., Gao, W. & Wang, C. Learning Large-Time-Step Molecular Dynamics with Graph Neural Networks. Preprint at http://arxiv.org/abs/2111.15176 (2021).
11. Satorras, V. G., Hoogeboom, E. & Welling, M. E(n) Equivariant Graph Neural Networks. Preprint at http://arxiv.org/abs/2102.09844 (2022).
12. Portegies Zwart, S. & McMillan, S. *Astrophysical Recipes: The Art of AMUSE*. (IOP Publishing, 2018). doi:10.1088/978-0-7503-1320-9.
13. Loshchilov, I. & Hutter, F. Decoupled Weight Decay Regularization. Preprint at https://doi.org/10.48550/arXiv.1711.05101 (2019).
14. Greenland, S. *et al.* Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations. *Eur. J. Epidemiol.* **31**, 337–350 (2016).

Appendix

On Choosing the Kolmogorov–Smirnov Test

To numerically quantify how good the macro-properties of inferred trajectories match the macro-properties of ground truth trajectories, we needed a test statistic. Given a test statistic, you can measure the size of the apparent effect δ^* (in our case the distance between the two sample distributions). Then, you can compute a p-value, which is the probability of seeing an effect at least as big as δ^* under the null hypothesis (the two samples come from the same underlying distribution). For example, a low value of 0.05 (5%) is often used to “reject” the null hypothesis - but this is purely a matter of consensus, and is often misused[10]. That's why, instead of relying on arbitrary critical values, we will only use the test statistic as a metric to numerically quantify the difference between the ground truth and inferred distributions, and let you, the reader, use your own judgment to gauge significance. To make this as easy as possible, one of our two main criteria was that the test used must be elegantly simple and intuitive, even to the uninitiated. The second criterion was that the test must be universal and non-parametric, which is important in the face of the diversity of the macro-properties under scrutiny. This led us to the Kolmogorov–Smirnov statistic and the associated Kolmogorov–Smirnov two-sample test. The test works by finding the biggest vertical difference between the cumulative distribution functions (CDFs) of two probability distributions (in our case the ground truth macro-properties and the inferred macro-properties). This maximum value is called the K-S statistic. The p-value is derived from the K-S statistics by calculating how likely it is, if we resample the same sample size again and again, to get a K-S statistic value bigger than the one obtained, given than the null hypothesis (the samples come from the same underlying distribution) is true.

Selected Macro-Properties a Larger Time Step

For additional context, we also provide the combined p-value plot and macro-property plots of the best checkpoint within a training run with a time step (N) of 40. As expected, increasing the time step to 40 (as opposed to 10 in the main text) yields worse results. Training length was kept approximately the same as for the N=10 run in the main text. We'll look at scaling to larger

time steps more properly in future works.