Pacific Northwest
NATIONAL LABORATORY

PNNL-34961

GRaman

Graph Network based Simulator for Forecasting Molecular Polarizability

September 2023

Marco Minutoli
Mahantesh Halappanavar
Edoardo Aprà
Patrik El-Khoury
Niri Govind

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

PNNL-34961

GRaman

Graph Network based Simulator for Forecasting Molecular Polarizability

September 2023

Marco Minutoli
Mahantesh Halappanavar
Edoardo Aprà
Patrik El-Khoury
Niri Govind

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

This report presents the work performed under the GRaman project, sponsored by the PCSD LDRD Seed program. The project aimed at accelerating ab initio molecular dynamics simulation using Graph Networks. The Graph Network framework is a ML framework that has been successfully employed to simulate the dynamics of several physical systems: including water splashing in a container and flags moving with the wind.

In this effort, we performed a data collection campaign for 3 different molecules of interest. We have built tools for preprocessing the trajectories obtained by simulating Raman Spectroscopy with NWChem and translating them into a suitable format for training. We have developed a training algorithm to train the Graph Network based simulators based on our data and developed a simulator that produces trajectories in the same NWChem format.

While the tool has improved with each iteration of development and subsequent experiments, the current state of the tool does not allow to directly incorporate the technology within the NWChem framework because the trajectories produced by the tool are not yet accurate enough. However, the technology has proved to hold significant promise to accelerate molecular dynamics simulations upon further research and development.

Acknowledgments

This research was supported by the PCSD LDRD Mission Seeds Program, under the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Contents

Figures

Tables

1.0 Introduction

The key problem addressed in the GRaman project is the acceleration of the computation of molecular properties in ab initio molecular dynamics (AIMD) simulation using artificial intelligence techniques. AIMD simulations have shown remarkable accuracy for the predicting Raman spectra, either ensemble-averaged molecular spectra or single molecule Raman scattering. However, a considerable number of long simulations are required to reach accurate results for medium size molecules. Previous attempts of applying time series forecasting demonstrated the possibility of shortening the length of AIMD simulations required for the dynamical evaluation of molecular properties.

We aim to build on the novel concept of graph network-based simulator for studying the time-domain (dynamic) Raman spectral simulations and comparing these theoretical results with experimental studies. A “graph network” is a type of graph neural network designed for learning the forward dynamics of a system using the message-passing framework (Battaglia 2018). This approach has been demonstrated for learning the dynamics in a variety of contexts such as simulation of rigid body, mass-spring, n-body, and robotic control systems, as well as non-physical systems, such as multi-agent dynamics and algorithm execution (Sanchez-Gonzalez 2020). Several downstream tasks such as classification and prediction at node-, subgraph- and graph-level can be performed. The decoder component can be used for forecasting the time series among other generative tasks.

1.1 Contributions and Achievements

In summary the project made the following contributions:

- Development of a PyTorch based software library implementing the fundamental blocks of a graph-network simulator;
- Development of data preprocessing tools to transform NWChem trajectory in a format suited for training;
- Data collection of NWChem-based datasets for training;
- Scaling the simulator on multi-GPU platforms using PyTorch-Lightening framework; and
- Preliminary exploration of the tool using the newly developed software library.

2.0 Data Collection

This section gives an overview of the data collection effort that was part of this project. We start by providing details on the molecular trajectories that we have collected to train the model (see 2.1). Then, we provide an overview of the pre-processing steps and give a high-level overview of a tool that we have developed to generate suitable training data for the GRaman simulator (see 2.2). We conclude by giving an overview of the training data released as part of the project (see 0).

2.1 NWChem Trajectories

We have collected trajectories of four different molecules by running simulations using NWChem (Apra 2020). Each simulation consists of and AIMD run using a timestep of 25 atomic units (corresponding to 0.60 femtoseconds) for a total time of around 6 picosends. For each molecular system described below, we ran multiple trajectories in order to reach a total simulation time of the order of 100s of picoseconds. More details about the computational protocol used here can be found in (Fischer, et al. 2016).

Our study used:

- H_2: the hydrogen molecule was chosen for its small size the simplicity of its vibrational spectrum due to its linear structure.
- pNTP (p-nitrothiophenol): this molecule was chosen since there is an extensive literature of the experimental and theoretical studies of its Raman spectra.
- NTP dimer (Ling 2016): this molecule was chosen to see how the Raman spectrum would change going for the pNTP monomer to its dimer.
- dmab (dimercaptoazobenzene): dimer structure like the pNTP dimer. Chosen to see if the modeling tools would be able to distinguish the two different dimers

Table 1: Summary of NWChem trajectories collected.

Molecule	Number of Trajectories	Total Size (MB)
H_2	1	
pNTP	26	739
NTP dimer	7	375
dmab	10	489

```
  15
        0     -834.4533212853  -2.831682E-01  -1.841542E+00   7.614573E-02
C                0.01807016    -1.66019861     0.07174881     -0.01369492     0.00528296     0.00270122
C                1.22567418    -0.93673432     0.05012922      0.00512525    -0.00094507     0.00190160
C                1.21117338     0.45073622    -0.00997326     -0.00575017     0.00153900    -0.01093851
C               -0.01390123     1.11911751    -0.04828929     -0.00430450    -0.00054291    -0.00506063
C               -1.22218040     0.42116787    -0.02769028     -0.00025986    -0.00221377     0.00185230
C               -1.20520400    -0.96698908     0.03191528      0.01169176    -0.00302471     0.00056753
H                2.17918223    -1.46683906     0.08029143      0.01637977     0.00689143     0.02209759
H                2.13470679     1.02800482    -0.02807517     -0.00602898     0.00834933     0.00075779
H               -2.15896992     0.97637296    -0.05866843      0.01854725     0.00876233     0.00549589
H               -2.14818405    -1.51591408     0.04747281     -0.00623053    -0.02600382     0.01955735
S                0.12382937    -3.41549828     0.14910381      0.00002045     0.00016498    -0.00067662
H               -1.20849352    -3.65168088     0.14395116      0.02578014     0.04537042     0.01102705
N               -0.03243068     2.58989253    -0.11215519      0.00421751    -0.00331932    -0.00170225
O                1.05440807     3.17558899    -0.13092410      0.00078984    -0.00382266     0.00829895
O               -1.13369421     3.14768394    -0.14314952     -0.00217962     0.00359452    -0.00243518
  15
        1     -834.4533101352  -2.639659E-01  -1.831274E+00   8.266613E-02
C                0.00978843    -1.65700402     0.07338225     -0.01347950     0.00522530     0.00270074
C                1.22877357    -0.93730578     0.05127914      0.00508848    -0.00092671     0.00190139
C                1.20769610     0.45166700    -0.01658807     -0.00569973     0.00150886    -0.01092161
C               -0.01650432     1.11878914    -0.05134956     -0.00430199    -0.00054749    -0.00506697
C               -1.22233759     0.41982910    -0.02657022     -0.00021454    -0.00221521     0.00184922
C               -1.19813366    -0.96881810     0.03225848      0.01145786    -0.00304176     0.00057654
H                2.18908811    -1.46267183     0.09365486      0.01592914     0.00697707     0.02202558
H                2.13106147     1.03305394    -0.02761611     -0.00616682     0.00824228     0.00070855
H               -2.14775410     0.98167150    -0.05534655      0.01787676     0.00890106     0.00548835
H               -2.15195094    -1.53163912     0.05930209     -0.00503394    -0.02513771     0.01946503
S                0.12384177    -3.41539851     0.14869457      0.00005549     0.00018185    -0.00067658
H               -1.19290509    -3.62424348     0.15062065      0.02433975     0.04487760     0.01104645
N               -0.02987989     2.58788535    -0.11318459      0.00414866    -0.00329094    -0.00169902
O                1.05488544     3.17327720    -0.12590550      0.00080109    -0.00381933     0.00829711
O               -1.13501235     3.14985765    -0.14462213     -0.00213823     0.00357111    -0.00243538
```

Figure 1: A trajectory as output from NWChem. The snippet includes two consecutive steps of the simulation. The format includes local properties of atoms (i.e., type, position, and velocities) and global properties of the molecule (i.e., dipole).

2.2 Data Preprocessing Tool

Figure 1 show the structure of the language that is output by NWChem. The language represents lists of molecule states at each step of the simulation. Each molecule state includes the energy, the dipole, and, for each atom in the molecule, positions and velocities. In this format, the information that is associated with each atom at every time step of the simulation only describes the current state of the molecule and does not provide information about the history. Therefore, we have built a language translator (Minutoli, Halappanavar, et al., GRaman Utils: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-utils 2022) that converts NWChem trajectories files into a JSON format that we have designed to store the training data for the GRaman Simulator.

Figure 2 shows a formal description of the output file format from NWChem expressed as a grammar of the language of valid trajectories. This grammar has been used to implement the parsing module needed by the translator in two ways. First the lexer and parser for the input file are generated by the Parsec library through a description of the grammar in Figure 2 in the DSL defined by Parsec. Second, the internal data structures to represent the NWChem trajectory mimic the representation in Figure 2 (e.g., the trajectory is a list of TrajectoryPoints).

$$
\begin{aligned}
Trajectory &\leftarrow TrajectoryPoint\ Trajectory \mid TrajectoryPoint \\
TrajectoryPoint &\leftarrow NumAtoms\ TotalEnergy\ Dipole\ AtomStates \\
NumAtoms &\leftarrow <integer\ number> \\
TotalEnergy &\leftarrow <float\ number> \\
Dipole &\leftarrow Vector3D \\
Vector3D &\leftarrow <float\ number><float\ number><float\ number> \\
AtomStates &\leftarrow AtomState\ AtomStates \mid AtomState \\
AtomState &\leftarrow AtomSymbol\ Position\ Velocity \\
AtomSymbol &\leftarrow <An\ Atom\ symbol\ from\ the\ periodic\ table> \\
Position &\leftarrow Vector3D \\
Velocity &\leftarrow Vector3D
\end{aligned}
$$

Figure 2: The grammar of NWChem trajectory files

Once the input file has been parsed and its content translated into a list of *trajectory point*, the tool performs a series of transformations on the information contained in the list with three objectives:

1. *Compute missing information.* For computational efficiency reasons, NWChem trajectory files often represent a subsample of the simulation run with the tool. For example, each state in a trajectory file could have been obtained by saving the state of the molecule every 10 simulation steps. In this case, the velocity recorded in the trajectory does not match the actual position of the atoms at the following step. To overcome the issue, the tool computes the "average" velocity that justifies the new position of the atoms in the molecule.

2. *Provide a window of the trajectory in the past for each data point.* One of the input arguments of the tool is the size of a *window* of past states that is used to augment the feature vector associated with the atoms in the system and constructs the data points that will be given to the model for training. Augmenting the current state with information from the past simulation steps was proposed by (Sanchez-Gonzalez 2020). We leverage the same approach and have built a tool that enables to explore different window sizes.

3. *Transform the internal tool representation into JSON file for reloading by the downstream tool.* Once step two is completed, the tool lowers the internal representation into a JSON format that we have designed. The format includes for each data point constructed:

    a. the edge list of the graph encoding the molecule;

    b. the feature vectors associated with each atom;

    c. and, the next state to be predicted by the model.

More details on the algorithms used and the implementation details of the tool can be found in the git repository implementing the GRaman Utils (Minutoli, Halappanavar, et al., GRaman Utils: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-utils 2022).

2.3 Training Data

(Minutoli, Halappanavar, et al., GRaman Experiments: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-experiments 2022) contains all the data, the NWChem scripts, and descriptions of the trajectories. We have worked closely with the DataHub team (e.g., Shannon Sheridan) to comply to the FAIR standards[1] and to make the data available on DataHub.

The tool described in Section 2.2 allows to efficiently regenerate all the training data that we have used on a need basis. In fact, the formatted training data is much larger than the raw NWChem trajectory files due to the richer information associated with each step of the simulation. This approach enables to more easily share the data avoiding unnecessarily large downloads.

[1] https://www.go-fair.org/

3.0 The Graph Networks Framework

This section will follow the same notation from (Battaglia 2018). We will define a graph as a tuple of the form $G = (\boldsymbol{u}, V, E)$, where $\boldsymbol{u}$ is a global attribute of the system being modelled, V is a set of vertices modelling the entities (or actors) of the system. Each vertex i has attributes $\boldsymbol{v_i}$ and each edge $e_k = (r_k, s_k)$ has attributes $\boldsymbol{e}_k$.

A Graph Network block uses three update functions (ϕ), and three aggregation functions (ρ):

$$
\begin{aligned}
\boldsymbol{e}'_k &= \phi^e\left(\boldsymbol{e}_k, \boldsymbol{v}_{r_k}, \boldsymbol{v}_{s_k}, \boldsymbol{u}\right) \\
\bar{\boldsymbol{e}}_i &= \rho^{e\to v}(E'_i) \\
\boldsymbol{v}'_i &= \phi^v\left(\overline{\boldsymbol{e}_i}', \boldsymbol{v}_i, \boldsymbol{u}\right) \\
\bar{\boldsymbol{e}'} &= \rho^{e\to u}(E') \\
\boldsymbol{u}' &= \phi^u(\bar{\boldsymbol{e}'}, \bar{\boldsymbol{v}}', \boldsymbol{u}) \\
\bar{\boldsymbol{v}}' &= \rho^{v\to u}(V')
\end{aligned}
$$

where E'_i is the set of edges incident on vertex i with updated attributes, V' is the set of vertices with updated attributes, and E' is the edge list of the graph with updated attributes. Therefore, the function ϕ^e is mapped over each edge in the graph to compute the update on the edges, while the ϕ^v function is mapped over each vertex in the graph to compute the update attributes. The function ϕ^u is applied only once to compute the update global attributes. The aggregation functions (ρ) are instead computed on sets of elements and their results in aggregated information. The ρ functions take a variable number of arguments and must be permutation invariant. Examples of such functions are summation, mean, minimum, and maximum.

The theoretical framework of Graph Networks is rather general and has the potential to be adapted for learning the dynamics of systems that can be modelled using graphs. For example, the work in (Sanchez-Gonzalez 2020) that has motivated this investigation adapted the framework of Graph Networks to learn the dynamics of several different substances including water, sand, and goop. In their implementation, Sanchez-Gonzalez *et al.* simplified the framework by removing the functions related to the global state and by concatenating the global state to each feature vector associated with each of the particle in the systems that they have considered.

3.1 GRaman Simulator

3.1.1 The Machine Learning Task

We aimed at building a trained Graph Network model that given a state of an input molecule $S = (G, F_v, F_e, F_u)$ at a time t will predict the state of the same molecule $S' = (G', F'_v, F'_e, F'_u)$ at time $t + 1$. We define the state of a molecule S as a 4-tuple (G, F_v, F_e, F_u), where G is a graph representation of the molecule where vertices are atoms and bonds are edges, F_v is a feature vector associated with every atom (vertices of G), F_e is a feature vector associated with the bonds (edges in G), and F_u is a global feature vector.

The graph representation of a molecule used in GRaman is isomorphic to the chemical graphs (or molecular graphs) as described in (McNaught and Wilkinson. 2019). However, the feature vectors associated with vertices (F_v) and edges (F_e) are richer than those used in chemical graphs. We define F_{v_1} as a tuple $(T, p_t, p_{t-1}, \dots, p_{t-w,} v_t, v_{t-1}, \dots, v_{t-w})$, where T is the atom type, and p_i and v_i include a window of length w of past positions and velocities of the atom. We define F_e as the distance in 3-D space between the atoms forming the edge e (bond) in G. The same structure of the feature vectors F_v and F_e can be used with positions and velocities in different coordinate systems (e.g., absolute or relative with respect to other atoms).

3.1.2 The Model Architecture

Figure 3 presents a block diagram of the model architecture used in GRaman. The architecture mimics the one of (Sanchez-Gonzalez 2020). The model takes as input the state S (as described in 3.1.1) at time t and produces as output the state S' of the molecule at time $t + 1$.

The pipeline is constituted by three blocks: an *Encoder*, the *Graph Neural Network*, and a *Decoder.* The Encoder/Decoder pair enable the model to learn a representation of the state space of the simulation that is optimized for the task. In our implementation, both Encoder and Decoders are Multi-Level Perceptrons. The Graph Neural Network block in Figure 3 is the core of the architecture and, conceptually, it is responsible of learning an approximator of the state transition function of the system we want to simulate.

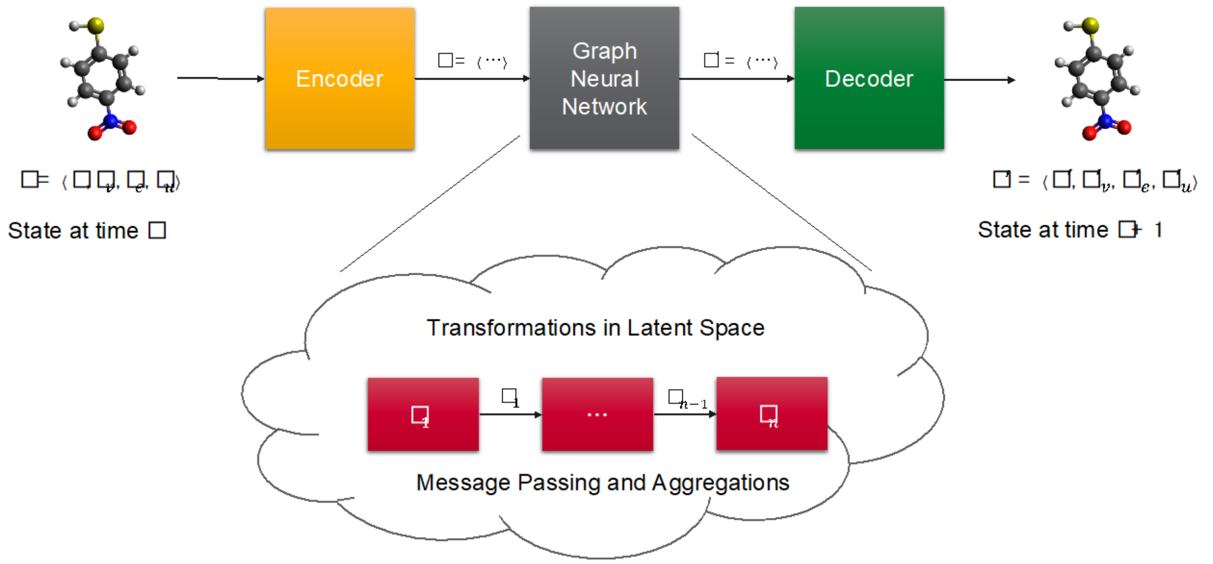

Figure 3: A block diagram of the GRaman simulator model architecture. Given the current state S the trained model predicts the state of the molecule at the next time step of the simulation S'.

The Graph Neural Network block is itself a pipeline of components working in the latent space where the states S are projected from the Encoder. Each element of the pipeline constitutes one step of Message Passing and Aggregation in the Graph Network framework (described in 3.0). The length of this pipeline constitutes one of the hyper-parameters that must be tuned for optimal performance. Our implementation uses the guidance given in (Sanchez-Gonzalez 2020) and sets its length to 5.

3.2 GRaman Simulator Software Tool

The GRaman simulator software tool (Minutoli, Halappanavar, et al., Graman Simulator: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-sim 2022) comprises of two components:

1. the **train** module;
2. the **simulate** module.

The **train** module implements the training algorithm used for the training of the GRaman Graph Network based simulator. This module went through several iteration before reaching its current state. Its first implementation attempted at predicting both position and velocity (as reported in the output of NWChem) as a single output. Over the length of the project, we have evolved the module in its current state where the Graph Network based simulator uses two different predictors (trained separately) to predict position and velocities. While the predictor for the velocity is trained on the output from NWChem, the predictor for the position is trained using the average velocity in the data and the position is then obtained through integration during simulation. The train module outputs periodic checkpoints of the two predictors retaining the best performing predictors. The module leverages GPU acceleration as provided by Pytorch lightning and uses early stopping to terminate the training loop when no more improvements are observed from the model.

The **simulate** module implements a simulator using the trained Graph Network model. The module takes as input an initial trace (usually 5 steps) used to establish enough history to compute the node features for the GRaman simulator and the checkpoint files to reload the two predictors composing the Graph Network simulator. At each step of the simulation, the simulator computes the current feature vectors associated with the graph representation of the molecule and uses the GRaman model to predict the instantaneous velocity of each atom and the average velocity that will move the molecule from the current state to the next state of the simulation. The position in space of each of the atoms is then obtained through a simple Euler integrator. The choice of the integrator was made to match the work from (Sanchez-Gonzalez 2020). The output of the simulate module has the same format as a NWChem trajectory as shown in Figure 1. During the project, we have focused on reproducing the dynamics of the molecule and, therefore, some of the global properties in the trajectory are mocked out. More precisely, our software does not make predictions about the energy of the system nor its dipole.

4.0 Experimental Evaluation

4.1 Experimental Setup

All our experiments have been conducted on a DGX-2 box (Enigma) with 4 NVIDIA A100 GPU. The implementation of the GRaman simulator (Minutoli, Halappanavar, et al., Graman Simulator: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-sim 2022) leverages PyTorch 1.13.1, PyTorch Geometric 2.3.0, and Torch Lightning 2.0.3.

The model is trained using the Mean Square Error (MSE) of predicted quantities as performance metric. We have set the maximum number of epochs to 100, but allowed the framework to stop the training sooner if no sensible improvement on the validation MSE is detected. We have observed that generally between 3 and 6 epochs the training always stopped.

Once the training of the predictors was completed, we have used the checkpoints of the best performing iteration in our experiments to perform one step predictions and long simulation roll outs. To evaluate the goodness of long simulation, we have used Avogadro 1 to load the trajectory files produced by the GRaman simulator and to qualitatively assess the quality of the trajectory obtained.

4.2 One Step Predictions

In our first experiment, we tested the performance of the model when predicting the next step of the simulation starting from the ground truth states extracted from the NWChem trajectories. The trajectories used in these experiments were part of the data used during the training process. For each prediction, the model was given as input the graph representing the molecule and the feature vectors associated with each atom constructed from a NWChem trajectory. The model was then asked to predict the state of the molecule at the next step of the simulation.

Figure 4 presents the results of this experiments for the NTP molecule. Each row of the plot represents the dynamics of each of the 15 atoms composing the NTP molecule while the columns plot the x, y, and z components. It can be observed that from the picture that the model performs well when provided as input data coming from NWChem trajectories. In fact, the actual NWChem dynamic (blue line on the plots) and the one step prediction dynamics (red line on the plots) are almost identical.

While these initial results were promising, we will see in the next section that the problem of predicting these molecule trajectories is harder than what the experimental results presented in Figure 4.

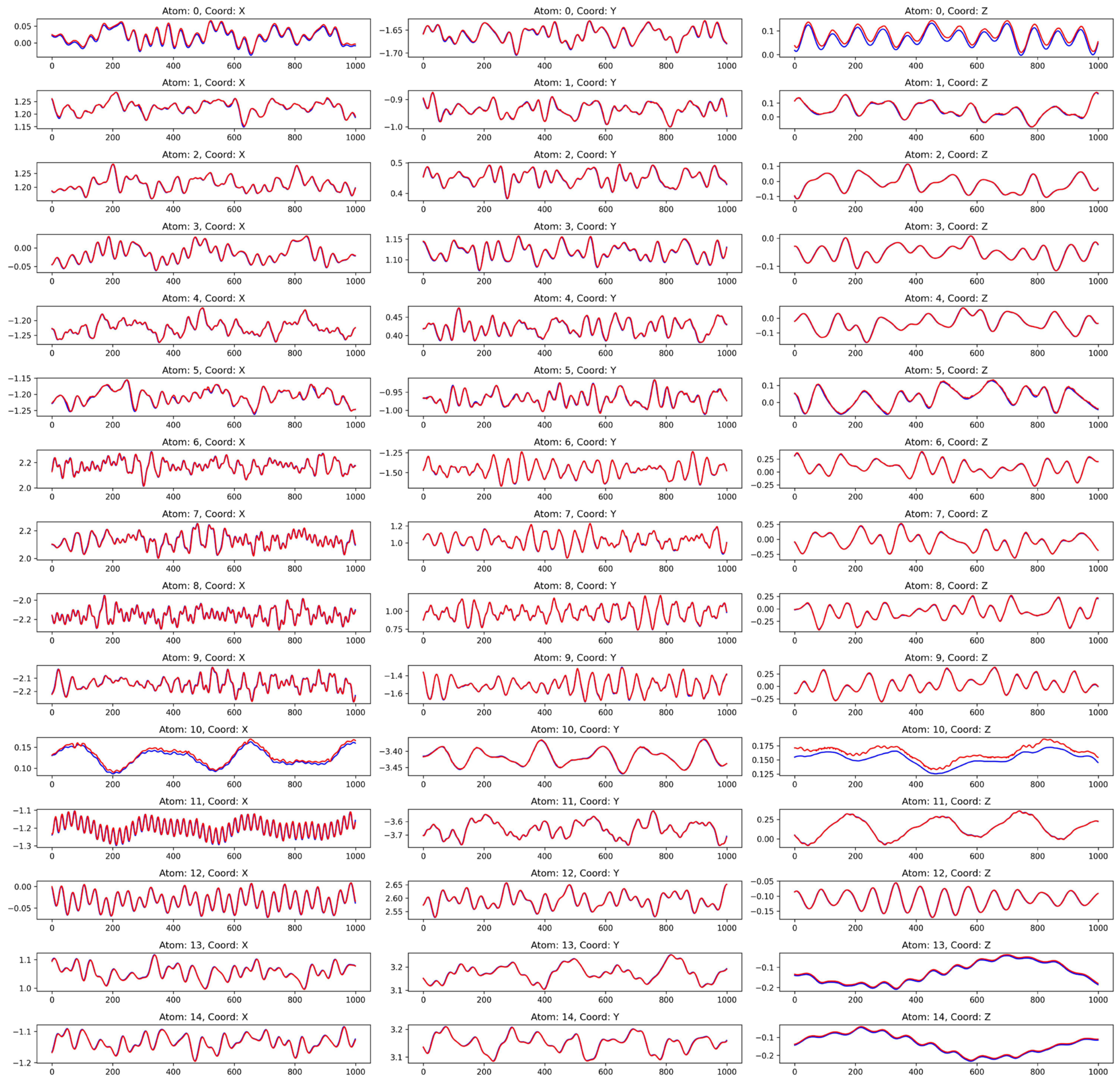

Figure 4: One Step Prediction of the model when using ground truth as input. Blue line is the ground truth while red line is what the model predicts.

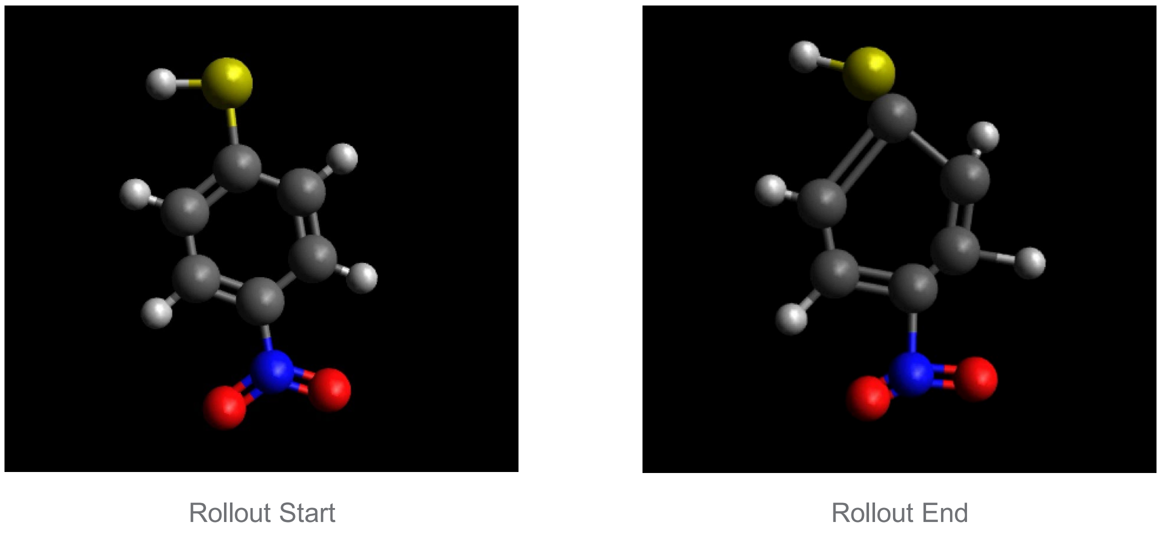

Figure 5: Long Trajectory Rollouts obtained by training the model on ground truth.

4.3 Long Trajectory Rollout

After assessing that a trained GNS model could well predict the next state when the input is provided by a NWChem trace, the next question is if the model has the same level of accuracy when being fed its own predictions. Therefore, we built a simulator module that is described in section 3.2.

After building a simulator module where predictions made by the GRaman model are fed back into it as part of the history of the trajectory, we notice that trajectories obtained by the tools were not behaving as expected. Figure 5 shows the outcome of one such simulation. Many of the simulations output showed deformation of the aromatic ring and the molecule breaking the bonds on the longest trajectories.

Our initial hypothesis was that a single monolithic predictor was not sufficiently accurate to perform well at the task. Therefore, we decided to split the model in two independent predictors: one predicting the atom velocity and the other predicting its position. We experimented with the training process and attempted to optimize the two separate predictors

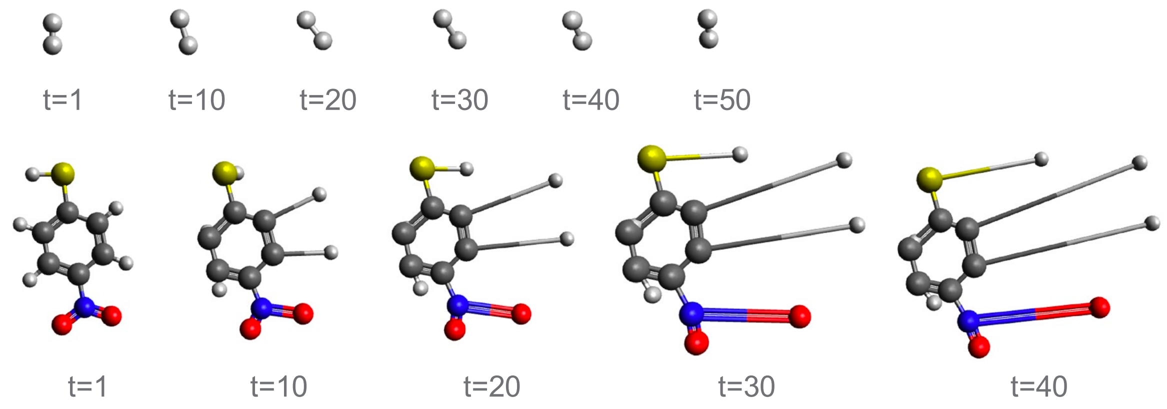

Figure 6: Long Trajectory Rollouts after training the model with noisy inputs. Top: H_2. Bottom: NTP.

both conjunctly and independently. Optimizing the two predictors independently provided the best performing model. However, the simulator was still generating unrealistic trajectories.

We hypothesize that the unrealistic rollouts produced by the model obtained by combining the velocity and position predictors in our latest implementation are caused by a lack of correction to the noisy input fed to the model itself. Therefore, we experimented with injecting noise in the training data aiming at improving robustness in the model. Noise was injected in the custom Data Loader for NWChem trajectories that we have implemented in the GRaman simulator. The approach ensured that the training data was perturbed with different noise at every epoch of the training process. The Data Loader leaves the ground truth values representing the next state unaffected by the added noise. The spirit of the approach was to have the model learn to predict the correct next state when only the input is noisy.

Figure 6 show the results of our model after training with noisy data. The picture was obtained taking snapshots from Avogadro. While the top row shows a roll out from the H_2 molecule, the bottom row shows a rollout obtained for NTP by this last version of the simulator. The trajectory of H_2 produced by training the model adding noise is very close to what is simulated by NWChem. The only difference is that the GRaman simulator is inducing a rotational moment. In contrast to the H2 molecule, the NTP molecule produces improved but still unrealistic rollouts. In fact, while the aromatic ring is now preserved by the simulation, it is affected by a translational moment that separates it from the rest of the atoms.

5.0 Dissemination Activities and Business Development

During the length of the project, the team engaged with peers to disseminate the work performed under the auspices of this project and to developed business opportunities.

The effort in GRaman has enabled the PI Marco Minutoli to get in contact with other scientists within PNNL to discuss about the technology developed in GRaman. These interactions have proved to extremely fruitful. In fact, these interactions materialized into four proposals:

- LDRD Open Call (PI: Dr. Laura M Fierce): the proposal will explore the training simulators for turbulent microphysics and chemistry to expand the representation of complex aerosol-cloud-precipitation processes in particle-based simulations. (Funded)

- NIH Proposal (PI: Dr. Laura M. Fierce): the aim of the proposal is to understand the influence of mechanism controlling exposure to airborne pathogens within buildings. The project aims at extending the PREEMPT model for network intervention to incorporate these mechanisms. The project plans to train AI/ML based model to extend the simulation of pathogen transmission to incorporate more precise indoor dynamics. (Submitted 09/2023).

- DOE ASCR Proposal (PI: Antonino Tumeo): The RADARSCOPE project proposed to develop a framework that will facilitate deployment of next-generation intelligent scientific workflows on highly heterogeneous dispersed systems. Part of the proposal builds on the technology developed in GRaman to build digital twins of the sensing infrastructure. The proposal was not selected for funding.

- DOE ASCR Proposal (PI: Mahantesh Halappanavar): SciQ proposed to develop a novel multi-stage uncertainty quantification framework for complex dynamical networks. One of the central pieces of the proposal was to develop a theory to understand and quantify uncertainties in Graph Networks. The proposal was not selected for funding.

6.0 References

Apra, E., Bylaska, E.J., de Jong, W.A., Govind, N., Kowalski, K., Straatsma, T.P., Valiev, M., van Dam, H.J.J., Alexeev, Y., Anchell, J. and Anisimov, V. 2020. "NWChem: Past, present, and future." *The Journal of Chemical Physics* 152 (18): 184102.

Battaglia, Peter W., Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti et al. 2018. "Relational inductive biases, deep learning, and graph networks." ArXiv preprint, arXiv:1806.01261.

Fischer, Sean A, Tyler W Ueltschi, Patrick Z El-Khoury, Amanda L Miffin, Wayne P Hess, Hong-Fei Wang, Christorpher J Cramer, and Niranjan Govind. 2016. "Infrared and Raman Spectroscopy from Ab Initio Molecular Dynamics and Static Normal Mode Analysis: The C–H Region of DMSO as a Case Study." *The Journal of Physical Chemistry B* 120 (8): 1429-1436.

Ling, Y., Xie, W., Liu, G. et al. 2016. "The discovery of the hydrogen bond from p-Nitrothiophenol by Raman spectroscopy: Guideline for the thioalcohol molecule recognition tool." *Scientific Report* 6 (1): 31981.

McNaught, A. D., and A. Wilkinson. 2019. *IUPAC. Compendium of Chemical Terminology, 2nd ed. (the "Gold Book").* Oxford: Blackwell Scientific Publications.

Minutoli, Marco, Mahantesh Halappanavar, Edoardo Apra, Niri Govind, and Patrick El Khoury. 2022. "GRaman Experiments: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-experiments."

—. 2022. "Graman Simulator: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-sim."

—. 2022. "GRaman Utils: https://stash.pnnl.gov/projects/GRAMAN/repos/graman-utils."

Sanchez-Gonzalez, Alvaro, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. 2020. "Learning to simulate complex physics with graph networks." *International conference on machine learning.* PMLR. 8459-8468.